Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 2 April 2015
Applications Note

OXFORD

Collaborative Generalized Linear Mixed Model

Privacy-preserving Construction of Generalized Linear Mixed Model for Biomedical Computation

Rui Zhu [1,], Chao Jiang [2] Xiaofeng Wang [1,] Shuang Wang [1,] Hao Zheng [3,]and Haixu Tang [1,*]

[1]Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, 47408, IN.
[2]Department of Computer Science and Software Engineerin, Auburn University, Auburn, 36849,AL.
[3]Department of Bioinformatics, Hangzhou Nuowei Information Technology, Hangzhou, 310053,China.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: The Generalized Linear Mixed Model (GLMM) is an extension of the generalized linear model (GLM) in which the linear predictor takes random effects into account. Given its power of precisely modeling the mixed effects from multiple sources of random variations, the method has been widely used in biomedical computation, for instance in the genome-wide association studies (GWAS) that aim to detect genetic variance significantly associated with phenotypes such as human diseases. Collaborative GWAS on large cohorts of patients across multiple institutions is often impeded by the privacy concerns of sharing personal genomic and other health data. To address such concerns, we present in this paper a privacy-preserving Expectation-Maximization (EM) algorithm to build GLMM collaboratively when input data are distributed to multiple participating parties and cannot be transferred to a central server. We assume that the data are *horizontally* partitioned among participating parties: i.e., each party holds a subset of records (including observational values of fixed effect variables and their corresponding outcome), and for all records, the outcome is regulated by the same set of known fixed effects and random effects.
Results: Our *collaborative* EM algorithm is mathematically equivalent to the original EM algorithm commonly used in GLMM construction. The algorithm also runs efficiently when tested on simulated and real human genomic data, and thus can be practically used for privacy-preserving GLMM construction. We implemented the algorithm for collaborative GLMM (cGLMM) construction in R. The data communication was implemented using the *rsocket* package.
Availability: The software is released in open source at https://github.com/huthvincent/cGLMM.
Contact: hatang@indiana.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Owing to the rapid advances in DNA sequencing technologies, human genomic studies, such as genome-wide association studies (GWAS) have been increasingly used to identify the genetic variants susceptible to human diseases Hirschhorn and Daly (2005) McCarthy *et al.* (2008). Meanwhile, many computational methods have been developed to enhance the sensitivity and statistical power of GWAS McCulloch (2003a). Among them, the linear mixed model (LMM) aims to explain the variation of a target phenotype in a population by a mixture of fixed effects (variables of interests) and random effects (unknown variables), which are shown to improve the identification rate of potentially causal variants of human disease Golan and Rosset (2018). However, LMM is designed for quantitative traits (e.g., blood pressure), and cannot be directly applied to categorical phenotypes. The generalized linear mixed model (GLMM) extends the LMM to support both categorical and quantitative phenotypes, and thus were frequently used in GWAS (e.g., for binary case-control studies or ordered disease stages) McCulloch (2003b). In a generic setting, a GLMM can be constructed using human genomic data (i.e., genotypes

The Author . Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

inferred from genome sequences) from a cohort of phenotyped human individuals, which requires data analysts to have direct access to the individual-level genomic data for every member in the cohort. In practice, it may be of great biomedical interest to assemble a large cohort of human genomes from multiple studies of the same disease for GWAS (often referred to as the *meta-analysis* Pharoah *et al.* (2013) Jeck *et al.* (2012) Begum *et al.* (2012)). For this purpose, the genomic data need to be collected and stored across several institutions, since it is difficult to move the data to a central site due to the challenges in data transmission (large data size), privacy protection (personal human genomic data are identifiable and thus sensitive) and the restriction of institutional data disclosure policy. As a result, privacy-preserving algorithms should be in place to enable computation on distributed genomic data without sharing individual-level genomic variants.

In the past decade, privacy-preserving algorithms have been developed for protecting various statistical methods, including survival analyses (e.g., using the Cox model Yu *et al.* (2008); Lu *et al.* (2015) Bradburn *et al.* (2003)), missing data imputation Jagannathan and Wright (2008), and logistic regression Wu *et al.* (2012). These algorithms follow the same *collaborative* computation approach, where the computation for targeted statistical methods is partitioned depending on the required input data: some computation is done on the locally retained genomic data to obtain often less-sensitive intermediate results, whereas the other computation is performed on a central server using the intermediate results as the input. As a step forward on this direction, in this paper, we present a privacy-preserving method for constructing a GLMM using a collaborative Expectation-Maximization (EM) algorithm that combines the Metropolis-Hasting (MH) algorithm in the E-step and the Newton-Raphson (NR) algorithm in the M-step to estimate parameters of the GLMM. We partition the computation for both the MH and the NR algorithm into the *private* and *joint* components, which are carried out by each participating party, whose intermediate results are then combined by a central server. The resulting collaborative EM algorithm is mathematically equivalent to the original EM algorithm (that is commonly used in GLMM model construction Booth and Hobert (1999)). We implemented the collaborative GLMM algorithm in R, and evaluated its performance using both simulated and real-world human genome data. The results show that the collaborative EM algorithm is efficient, and also accurate. We note that the collaborative GLMM method can be applied to other biomedical computation tasks where privacy-preserving approaches are needed, e.g., for building predictive models of human diseases using diagnostic information retrieved from patients' Electronic Health Records (EHRs) that are held by multiple medical institutions but cannot be shared outside each respective institution.

2 Methods

2.1 Background

2.1.1 Generalized Linear Mixed Model (GLMM)

In an LMM, the outcome is a continuous variable(Please see LMM in Appendix). In GWAS, however, the outcome is often categorical, e.g., the binary outcome representing disease or healthy in a case-control study. The generalized linear mixed model (GLMM) extends LMM by incorporating a link function to convert a continuous outcome into a categorical outcome McCulloch (2003b). Here, we use the logit function: $f(z) = log\frac{z}{1-z}$ as the link function to deal with the binary outcome often used in the case-control GWAS:

$$log(\frac{P(Y=1|X)}{1-P(Y=1|X)}) = \beta X + Zu \tag{1}$$

, where X represents fixed effects and Z represents the random effect, while β and u are the coefficients for the fixed and the random effects, respectively.

Below, we summarize the Expectation-Maximization (EM) algorithm for GLMM parameter estimation from a total of N input records, each with the given fixed and random effects as well as the desirable outcome (0 or 1). Let T be the number of random effect categories and M be the number of fixed effect variables. We consider $t \in [1, ..., T]$ as one of the random effect categories, and $m \in [1, ..., M]$ as one of the fixed effect variables. For the convenience of presenting the privacy-preserving EM algorithm for parameter estimation in the next section, here we assume each random effect category contains the equal number of P input records (thus, $N = T \times P$), and $p \in [1, ..., P]$ indexes each record. Our implementation does not have this constraint and allows for each category to contain different number of records. Hence, the outcomes for all records can be represented as a *response matrix*,

$$\mathbf{Y} = \begin{bmatrix} Y_{11} & Y_{1,2} & \cdots & Y_{1,P} \\ Y_{2,1} & Y_{2,2} & \cdots & Y_{2,P} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{T,1} & Y_{T,2} & \cdots & Y_{T,P} \end{bmatrix} \tag{2}$$

The input random effects are represented as,

$$\mathbf{Z} = \begin{bmatrix} Z_1 & Z_2 & \cdots & Z_T \end{bmatrix}^\top \tag{3}$$

Z is a vector with length T. For i^{th} element Z_i ($i \in \{1, 2, ..., T\}$) is sampled from a normal distribution $N(0, \sigma)$. Note that different with most notation of GLMM, We remove "Membership matrix" u by reshape input matrix X as eq(5). For example, in (5) X_1 is the first fixed effect variable. It is a matrix which has T rows, where T is number of levels of random effect. So for the i^{th} row of X_1, all the record has same random effect level, so that they are all correspond to Z_i which is the level of random effect of i^{th} record. It different from most notation in GLMM, but this is the only notation way that we think can elaborate cGLMM. Throughout the rest of this paper, we will use $X_{t,p,m}$ to represent the m^{th} input fixed effect values in t^{th} level of random effect, p^{th} record, and use $Y_{t,p}$ to represent the desirable outcomes, where $X_{t,p}$ is a M-dimensional vector (as shown below), and $Y_{t,p} \in \{0, 1\}$ is the binary outcome(as shown on (3)).

$$\mathbf{X}_1 = \begin{bmatrix} X_{1,1,1} & X_{2,1,1} & \cdots & X_{P,1,1} \\ X_{1,2,1} & X_{2,2,1} & \cdots & X_{P,2,1} \\ \vdots & \vdots & \ddots & \vdots \\ X_{1,T,1} & X_{2,T,1} & \cdots & X_{P,T,1} \end{bmatrix}$$

$$\vdots \tag{4}$$

$$\mathbf{X}_M = \begin{bmatrix} X_{1,1,M} & X_{2,1,M} & \cdots & X_{P,1,M} \\ X_{1,2,M} & X_{2,2,M} & \cdots & X_{P,2,M} \\ \vdots & \vdots & \ddots & \vdots \\ X_{1,TM} & X_{2,T,M} & \cdots & X_{P,T,M} \end{bmatrix}$$

m^{th} fixed effect is a $T \times P$ matrix, denote as X_m. So the corresponding fixed effect coefficients β are represented in an M-dimensional vector,

$$\beta = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_M \end{bmatrix}^\top \tag{5}$$

We use an EM algorithm to estimate the parameters (Z and β) of an optimal GLMM: in the M-step, given the current latent variables (σ), we compute the fixed effect coefficients β by using the Newton-Raphson algorithm, and in the E-step, we compute the latent variables (σ)

based on the current model parameters β by using the Metropolis-Hasting (MH) algorithm. The EM algorithm iterates between these two steps until convergence.

Specifically, in the MH algorithm, we started from randomly selected σ_i for each category i, and then sample random effects $\mathbf{Z} = (z_1, ..., z_T)$, where each z_i is randomly sampled from $N(0, \sigma)$. Then in each MH sampling step, we first sample new random effects z' in the neighborhood of $z \in \mathbf{Z}$ based on the current $N(0, \sigma)$, and then compute a *proposal* probability A between z' and z using the current model parameters β to decide if z should be replaced by z' in $\mathbf{Z}$ for the next step. The proposal function is defined as

$$A\left(\mathbf{z}, \mathbf{z}'\right) = \min\left(1, \frac{p^*\left(\mathbf{z}\right) q\left(\mathbf{z}'\right)}{p^*\left(\mathbf{z}'\right) q\left(\mathbf{z}\right)}\right) \in [0, 1] \tag{6}$$

where $p^*(\cdot)$ is the likelihood of random effect according to the current GLMM model, and $q(\cdot)$ represents the likelihood of observing a record according to the current $N(0, \sigma)$. Specifically, $A(z', z)$ can be written as McCulloch (2003a):

$$A\left(z, z'\right) = \min\left\{1, e^{\sum_{p=1}^{P} Y_p\left(z'-z\right)} \prod_{p=1}^{P} \frac{1 + e^{\beta X_p + z}}{1 + e^{\beta X_p + z'}}\right\} \tag{7}$$

Let M_{MH} be the total number of sampling steps in the MH algorithm, which can be set to between 500 and 1000. After we complete M_{MH} iterations, the variance of the final samples are used to compute the updated σ. The details of the MH algorithm will be further illustrated in section 2.3, when we present the privacy-preserving version of the GLMM construction algorithm.

In the M-step of the EM algorithm, we use the Newton-Raphson (NR) algorithm to compute the fixed effect coefficients β based on the current random effects $\mathbf{Z}$ and the parameters σ (updated in the E-step). NR is a second-order optimization algorithm, which uses the first-order derivative to choose the optimization direction, and the second-order derivative to choose the step length. Comparing with the first-order optimization algorithms, the NR algorithm takes fewer steps to converge. Specifically, in each NR iteration i, β is updated by

$$\beta_i = \beta_{i-1} + f' \cdot H^{-1} \tag{8}$$

where the first-order derivative f' and the Hessian matrix H can be computed by

$$f' = \sum_{t=1}^{T} \sum_{p=1}^{P} \left[Y_{t,p} X_{t,p} - \frac{X_{t,p} e^{\beta X_{t,p} + Z_t}}{1 + e^{\beta X_{t,p} + Z_t}}\right]$$

$$and \tag{9}$$

$$H = -\sum_{t=1}^{T} \sum_{p=1}^{P} \frac{X_{t,p}^2 e^{\beta X_{t,p} + Z}}{(1 + e^{\beta X_{t,p} + Z})^2}$$

We use a threshold ϵ to determine if the optimization converges, i.e., the NR algorithm terminates when the distance between the βs from two consecutive steps is smaller than ϵ. The details of the NR algorithm will be illustrated in Section 2.4.

2.2 Privacy-Preserving GLMM Construction on Horizontally Partitioned Data

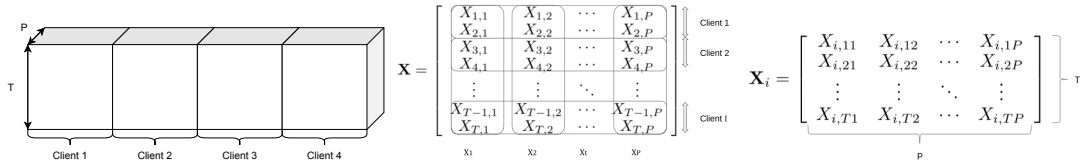
In this paper, we consider a scenario of collaborative computation for privacy-preserving GLMM construction, where each of the collaborative parties (e.g., medication institutions) holds a sensitive dataset (e.g., the genotypes from a cohort of human individuals including disease patients and healthy controls) and attempts to build a GLMM model collaboratively without sharing its raw dataset to the other parties. This scenario is often referred to as the collaborative computation on *horizontally* partitioned data Wang *et al.* (2013); Jiang *et al.* (2013), where each participating party holds the *complete records* from a subset of subjects, whereas, in a different scenario called *vertically* partitioned data Li *et al.* (2016), each participating party holds a different portion of records from the same set of subjects that can be matched among all parties. The general idea of collaborative computation is to partition a target algorithm (e.g., to construct the GLMM) into two parts: the first part of *private* computation can be performed on the partial data held by each participating party that generate intermediate results required for the second part of computation, and the second part *joint* computation has to be performed on a central server using the intermediate results from the private computation of all parties. In this scenario, the central server is considered to be *semi-trusted* (honest-but-curious), and thus the sensitive raw data cannot be directly sent to the central server by each participating party.

In practice, collaborative algorithms for many tasks (e.g., uncertainty quantification Sciacchitano *et al.* (2015)) have been developed. In many cases, they involved multi-round communications where in each round, not only the intermediate results were transferred from each party to the central server for the joint computation, but the intermediate results from joint computation needs to be sent back to each party for the next round of private computation. Notably, the collaborative computation for building a machine learning model is also referred to as the *federated learning* approach Konečnỳ *et al.* (2016), which aims to save the cost of transferring a large amount of training data among participating parties while protecting the privacy of sensitive training data. In some cases, the intermediate results may carry sensitive information and thus can be used to infer the presence of a record (e.g., by using a re-identification attack El Emam *et al.* (2011)). In these cases, the joint computation should be performed by using encryption protocols (e.g., homomorphic encryption (HE) Gentry (2009); Wang *et al.* (2016)), or in a Trusted Execution Environment (TEE) Sabt *et al.* (2015).

Consider the input data matrix X containing the fix effects (e.g., the genotypes) on a total of N records (disease patients) in each of the T random effect categories (Figure 1a). In the horizontal data partition scenario, the entire data matrix was not held by a single user, but instead by K different parties: the ith party holds a subset of $T \times P$ records (Figure 1c). Again, for the convenience of presentation, here we assume each party holds the same number (P) of records in each of the T random effect categories; as a result, $N = T \times P \times K$. However, in our implementation, we can handle the situation where different parties hold a different number of records, and also the records may be distributed unevenly among random effect categories. Using the separately held input data matrix, our goal is to develop the collaborative computation algorithm for building a GLMM model among the K participating parties, through the partition of private and joint computation for the EM algorithm as laid out above.

The collaborative GLMM (cGLMM) presented here takes as the input the data matrix jointly held by multiple parties and uses the EM algorithm to estimate the parameters of the GLMM (i.e., β for fixed effect coefficients and σ for the random effect). In each iteration of the E-step (MH algorithm) and M-step (NR algorithm), each party first computes the intermediate results from their own data, and then transfer the results to a central server to compute the updated parameters, which will then be sent back to each party for the next iteration.

Figure 2 illustrates the workflow of the collaborative EM algorithm for cGLMM, in which each iteration consists of the E-step (MH algorithm; Figure 2 left) and M-step (NR algorithm; Figure 2 right), and each of them is performed in a collaborative manner. Both the MH and NR algorithms can be partitioned into the private computation (executed by each party separately) and the joint computation (executed on a central server) such



(a) Data partition among participating parties (b) Horizontally-partitioned input matrix (c) The data matrix held by the ith party

Fig. 1. The input fixed effect matrix is jointly held by K parties, each holding a subset of records. Hence, the input matrix can be viewed as horizontally partitioned (b), and each party holds a submatrix containing a subset of rows (c).

that the intermediate results generated by the private computation of each party are sent to the central server for joint computation (①), and the intermediate results generated by the joint computation are then sent back to each party for the private computation in the next iteration ②. As a result, the collaborative EM algorithm involves multiple rounds of data communications, where the number of rounds is equal to the number of iterations in the EM algorithm.

In the next two sections, we will present the details of the collaborative EM algorithm, specifically the partition of the private and joint computation for the MH and NR algorithms, respectively.

2.3 Collaborative Metropolis-Hasting (MH) Algorithm

As described in Section 2.1.1, in the E-step of the EM algorithm for the parameter estimation of GLMM, we attempt to estimate the fixed effect coefficients (β) based on the current estimation of the random effect parameters (σ), using the MH algorithm. Given the horizontally partitioned data, we need to modify the MH algorithm into a collaborative version that consists of the private computation, in which each participating party computes some intermediate results from its partial data, and the joint computation, in which the intermediate results from all parties are transferred to the central server to compute the new fixed effect coefficients for the next step.

Figure 3a illustrates the procedure of the collaborative MH algorithm. As an initial step, based on the current estimated σ, the central server samples a set of random effects $Z_0 \sim N(0, \sigma)$ as the initial pool of samples, and sends them to all participating parties. In each subsequent iteration i, the central server first samples a new set of random effects Z_i from $N(0, \sigma)$, and sends them to all parties. Then each party k ($k = 1, 2, ..., K$) computes the intermediate results A_k on its private server by using the random effect from the previous step (Z'_l), the new random effect (Z_{l+1}), as well as the private data records $X_{k,p}$ and $Y_{k,p}$ held by the party. Then by using A_k we can update the result to get Z'_{l+1}.

$$A_k = \sum_{p=1}^{P} Y_{k,p} \left(Z'_l - Z_{l+1}\right) + \sum_{p=1}^{P} log(1 + e^{X_{k,p}\beta + Z_{l+1}}) - \sum_{p=1}^{P} log(1 + e^{X_{k,p}\beta + Z'}) \quad (10)$$

, where P is the same number of records in each category of random effect, and $Y_{k,p}$ is the outcome (e.g., $\in \{0, 1\}$ in the case-control GWAS study) of the p^{th} record, and $X_{k,p}$ is the input fixed effect variables (e.g., the genotypes in GWAS) in all T random effect categories, which are held privately by the k^{th} participating party. Again, for the convenience of presentation, here we assume each party k holds the same number (P) of records in each random effects category. In our implementation, however, different numbers of records are allowed to be held by different parties and in different random effects categories.

The intermediate result A_k computed in the private computation by the k^{th} party is then sent to the central server, where the joint computation is performed for computing the proposal probability density A,

$$A = min(1, \sum_{k=1}^{K} A_k) \quad (11)$$

to decide if the previous samples of the random effect z_{i-1} should be replaced by the new estimates z_i. The proposal probability A acts like a filter to replace the less likely (i.e., fitting improperly to the outcome according to the current estimates of fixed effect parameters β) random effects parameters sampled in the previous step: if $A = 1$, the previous parameters are definitively replaced; otherwise, it is replaced with the probability A. After the update of iteration i, the random effects Z_i are stored in the central server for the subsequent iterations in the MH algorithm. The iteration of the algorithm continues until the parameter estimation converges. In our experiments, the MH algorithm usually converges after 1000 iterations.

After convergence, the central server will retain a pool of random effects Z^* obtained in each iteration. In order to better estimate the parameters Z^*, we adopted the burn-in strategy commonly used in MH algorithms Chib and Greenberg (1995), which eliminates some sampled parameters in the initial steps of MH sampling. Based on the final sample pool of random effects Z^*, the central server updates the parameters σ, which will be used in the M-step (see the collaborative NR algorithm in Section 2.4) for estimating the fixed effect coefficients β.

The key idea of the collaborative MH algorithm presented here is the partition of the computation of proposal probability into two components, the private and joint computation, respectively. This partitioning approach is equivalent to the non-collaborative MH algorithm presented in Section 2.1.1. As a result, the collaborative MH algorithm offers mathematically equivalent solutions as the original MH algorithm in the E-step of the EM algorithm for GLMM parameter estimation, even though in practice, the solutions may not be identical due to the stochastic sampling in the MH algorithm.

Apparently, the collaborative MH algorithm requires the central server and the server at each participating party to remain online to facilitate the in-time communication between servers for the coordinated private/joint computation. The entire computation involves M_{MH} rounds of communications, where M_{MH} represents the number of MH iterations before it converges. In each iteration, the central server sends a P dimensional vector (i.e., z_i) to each participating party, and receives another P dimensional vector i.e., A_k, from each party. In addition, the server at each party computes A_k privately, which takes $O(P)$ running time and the central server computes the aggregate A (Equation 11), which takes $O(P)$ time for each iteration. The private computation completed by the participating parties spends a majority of the running time during the entire process.

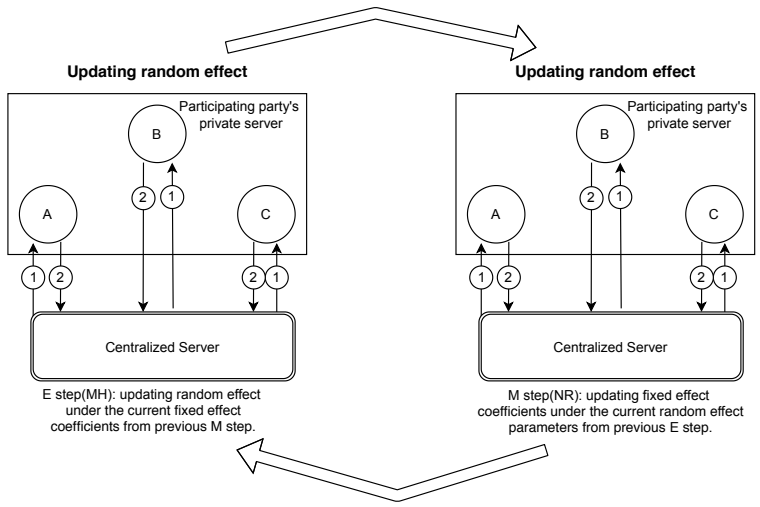


Fig. 2. The workflow of the collaborative EM algorithm for building cGLMM jointly by multiple participating parties.

2.4 Collaborative Newton-Raphson (NR) Algorithm

In the M-step, we use a Newton-Raphson (NR) algorithm to estimate the fixed effect coefficients (β) based on the current estimates of the random effect parameters σ. Similar to the E-step, we propose a collaborative NR algorithm that partitions the computation of the first-order derivative and Hessian matrix of the likelihood function, which are required to update the fixed effect coefficients, into the private and joint computation.

Figure 3b illustrates the procedure of the collaborative NR algorithm in the M-step. Before the iteration starts, the central server will pick up the random β_0 and Z as input. In each subsequent iteration i, we attempt to update β_i by using Equation 8, and thus we need to compute the Hessian matrix H and the first-order derivative f'.

Notably, the computation of H and f' in the non-collaborative NR algorithm requires the entire input dataset, i.e., Y_t and X_t of each record t (Equations 9). To compute them without sharing the data among participating parties, we re-write the equations for computing the Hessian matrix and the first-order derivative. As a result, each participating party k can compute the intermediate results on its private server, including the partial Hessian matrix H_k and the partial first-order derivatives f'_k. We denote the fixed effects of the pth record in the t^{th} category of random effect held by the k^{th} party as $X_{t,p,k}$. The $k * th$ party then computes

$$H_k = -\sum_{t=1}^{T}\sum_{p=1}^{P} \frac{X_{t,p,k}^2 e^{\beta X_{t,p,k}+Z_t}}{(1+e^{\beta X_{t,p,k}+Z_t})^2} \tag{12}$$

$$f'_k = \sum_{t=1}^{T}\sum_{p=1}^{P} \left[Y_{t,p,k} X_{t,p,k} - \frac{X_{t,p,k} e^{\beta_m X_{t,p,k}+Z_t}}{1+e^{\beta_m X_{t,p,k}+Z_t}} \right] \tag{13}$$

and sends the intermediate results to the central server for the joint computation of the full Hessian matrix and first-order derivatives,

$$H = \sum_{k=1}^{K} H_k \quad and \quad f' = \sum_{k=1}^{K} f'_k \tag{14}$$

which are sent back to each participating party to update the fixed effect coefficients of its records β_{i+1} (Equation 8) for the next iteration. Finally, the iterative procedure terminates after the estimates of the fixed effect coefficients converge or it reaches the preset maximum number of iterations.

Notably, the collaborative computation for the Hessian matrix and the first-order derivatives is equivalent to the non-collaborative NR algorithm presented in Section 2.1.1. Therefore, the collaborative NR algorithm generates identical results (i.e., the fixed effect coefficients β) as the original NR algorithm in the M-step of the EM algorithm for GLMM parameter estimation.

Similar to the collaborative MH algorithm, the collaborative NR algorithm also requires the central server and all participating parties' servers to remain online for in-time communication. The entire computation involves M_{NR} rounds of communications, where M_{NR} represents the number of NR iterations before it converges. In each iteration, the central server receives the Hessian matrix (in $M \times M$ dimension) and first-order derivative (i.e., a M dimensional vector) computed by each party and then sends back the full Hessian matrix and first-order derivative vector of the same size to each party. In the private computation, the server at each party first computes the partial Hessian matrix and partial first-order derivatives (Equations 12 and 13), which takes $O(|X| * P^2)$ and $O(|X| * P)$ running time, respectively, and also updates the fixed effect coefficients β, which takes $O(|X|)$ time, where $|X|$ represents the number of fixed effect variables. On the other hand, in the joint computation, the central server combines these intermediate results into the full Hessian matrix and first-order derivatives (Equation 14), which only takes $O(T \times P \times K \times |X|) \approx O(N \times |X|)$ time, where N is the total number of records held by all parties. Overall, the private and the joint computation takes about the same amount of running time in the collaborative NR algorithm.

Above we presented the collaborative algorithms for the E-step and M-step, respectively. It is straightforward to combine these them into a collaborative EM algorithm, which iterates between these two steps until the estimated parameters (including the fixed effect coefficients β and the random effect parameter σ) converge. The entire process requires $C \times (M_{MH} + M_{NR})$ rounds of communications, where C represents the number of iterations of the EM algorithm.

2.5 Implementation

We implemented the algorithm for collaborative GLMM (cGLMM) construction in R. The data communication was implemented using the *rsocket* package. The software is released in open source at https://github.com/huthvincent/cGLMM.

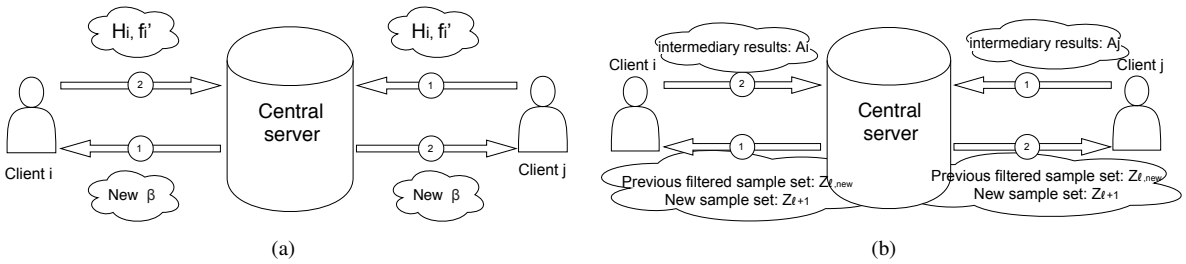


Fig. 3. The data communication between the central server and participating parties in the collaborative MH algorithm (E-step; a) and the collaborative NR algorithm (M-step; b).

3 Results

3.1 Simulation Experiments

We first conducted simulated experiments to test the performance of the collaborative GLMM (cGLMM) construction. We simulated three different data sets with 50, 100 and 150 fixed effects over 1000, 2000, and 3000 records, respectively, which were held by two participating parties. For each data set, there is one random effect with 10 different levels. We note that the numbers of fixed effects simulated here resemble the real cases in a collaborative study, in which a participating party have already identified a small number of putative effects from its own data, and hope to use the data from the other parties to validate these candidates. We compared the results of cGLMM using the collaborative EM algorithm with results of the regular GLMM using the non-collaborative EM algorithm. We considered only two parties; the running time may not be much longer when more parties are involved because the computation carried out by each party remains the same as long as each party has about the same number of records.

For comparison purpose, we used the same convergence conditions and the same initial selection of parameters in the cGLMM and regular GLMM. Nevertheless, the final results were similar but not identical because the MH algorithm in the E-step may introduce some randomness. We set the maximum MH iteration as 1000, and only retain the last two sets of samples in the pool (i.e., burn-in=998). The convergence condition of the EM algorithm is that the Euclidean distances between all parameters in three consecutive iterations are all below a threshold of 0.08. The same threshold was also used to determine the convergence of the MH algorithm.

We simulated the genotypes as the fixed effects. The variable of each fixed effect can take $X \in \{00, 01, 11\}$, where 00, 01 and 11 represents the genotypes of homozygous major, heterozygous and homozygous minor, respectively. The frequencies of the three genotypes were simulated following the Hardy-Weinberg equilibrium with a specific minor allele frequency; for example, if the minor allele frequency is 0.3, the simulated frequencies of three genotypes follow 0.49, 0.42 and 0.09, respectively. Finally, in the simulation of the linear mixed model, we assume the minor allele has an additive effect on the outcome; as a result, the fixed effect variables were simulated as integer values of {0, 1, 2}, representing the three genotypes, respectively. GLMM can be applied to other models of genetic effect (e.g., the dominant effect model), which were not tested here.

Table 1 compares the results from the collaborative (cGLMM) and non-collaborative (GLMM) EM algorithm on the three simulated datasets. Both algorithms converge after a small number of iterations, while cGLMM runs slower than GLMM, which is mostly due to the communication overhead. We note that our evaluation was conducted using three separate jobs (simulating two servers of the participating parties and the central server, respectively) on the same computer; thus, in reality, the running time of cGLMM can be significantly longer, depending on network bandwidth among the participating servers. However, even though the number of rounds of communication between servers is overall high, the total amount of data transferred is still moderate. Interestingly, we observe that with the increasing size of the GLMM model (i.e., with more fixed effect variables), the overhead of cGLMM comparing with GLMM actually decreases, probably because for complex GLMM problems, significantly more time is spent on the actual computation comparing to data communication. Finally, in all cases, cGLMM reported fixed effect coefficients that are nearly identical to the actual values used in the simulation (Pearson Correlation Coefficient PCC = 0.99) and the results from GLMM, suggesting cGLMM achieved the same accuracy as GLMM.

Figure 4 compares the results of GLMM and cGLMM on the simulated dataset containing 150 fixed effect variables (genotypes) on 3,000 records. As shown in Fig. 3.1a and 3.1b, respectively, the distances between the fixed effect coefficients (β) and the random effect parameters (σ) in consecutive iterations becomes close to zero after only a few iterations of the collaborative EM algorithm, indicating it converges as fast as the non-collaborative EM algorithm. Furthermore, the fixed effect coefficients reported in cGLMM are very to those in GLMM and the actual coefficients used in the simulation (Fig. 4c; PCC > 0.99 with the actual values), indicating cGLMM reported comparable results as GLMM. The results from the other two datasets (containing 50 and 100 fixed effect variables) are generally similar, and are shown in Supplemental Figures 1-2. These results suggested the collaborative EM algorithm is accurate and efficient for constructing GLMM collaboratively among multiple parties.

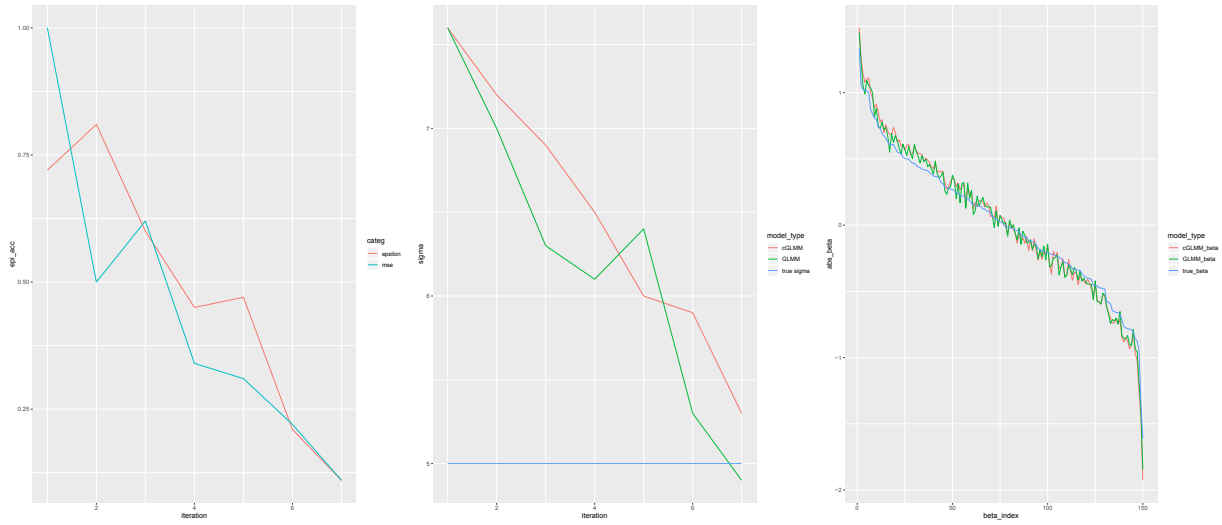
3.2 Real Human Genomic Data

Next, we used the real human genomic data from the 1000 Genomes Project Wang *et al.* (2018) to test our algorithm. The 1000 Genomes Project has total 2000 records, contain 1000 records in control group and 1000 records in case group. We selected all 2000 records from the whole data set. We then selected the 10 most significant SNPs (by using a χ^2 test; including 5 positively and 5 negatively correlated with Group I versus Group II) between these two groups (Group I and II, simulating the case and control groups in GWAS), and another randomly selected 40, 90 and 140 SNPs to form the three testing datasets containing 50, 100 and 150 SNPs (used as the fixed effect variables in GLMM), respectively. We considered the gender of the individuals as the random effect: Group I contains 505 males and 495 females while Group II contains 479 males and 521 females. Similar to the simulation experiment, we consider the additive genetic model, in which the fixed effect variables take 0, 1, or 2, representing the genotypes of homozygous major, heterozygous and homozygous minor, respectively. We randomly split the genomes into two subsets containing 907 and 1093 records, respectively, assuming each participating party holds one of them.

Table 2 compares the results from the collaborative (cGLMM) and non-collaborative (GLMM) EM algorithm on the three real human genomic

Table 1. We run cGLMM and GLMM 10 times, below is the average comparison of GLMM and cGLMM on simulated datasets. *PCC: Pearson Correlation Coefficients between the actual fixed effect coefficients used in simulation and those reported by GLMM and cGLMM, respectively.

No. of SNPs	iterations		running time (mins)		communication round(cGLMM)	communication size(cGLMM) (KB)	PCC*
	GLMM	cGLMM	GLMM	cGLMM			
50	7	7	0.062	2.9	6107	369.62	0.99994
100	8	7	0.096	4.2	6505	807.40	0.99999
150	9	8	2.6	26.6	7825	1403.22	0.99996



(a) (b) (c)

Fig. 4. The comparison between the results from cGLMM and GLMM on the simulated dataset containing 150 fixed effects in a typical run, for the distances between the fixed effect coefficients (β) (a), and the distances between the random effect parameters (σ) in consecutive iterations of the collaborative (in red) and non-collaborative (in green) EM algorithm, as well as the actual fixed effect coefficients (β, sorted in decreasing order; in blue) and those reported by GLMM (in green) and cGLMM (in red), respectively.

Table 2. Comparison between the results from cGLMM and GLMM on real human genomic datasets. *PCC: Pearson Correlation Coefficients between the fixed effect coefficients reported by GLMM and those reported by cGLMM. All resource list in here are averages of 10 runs.

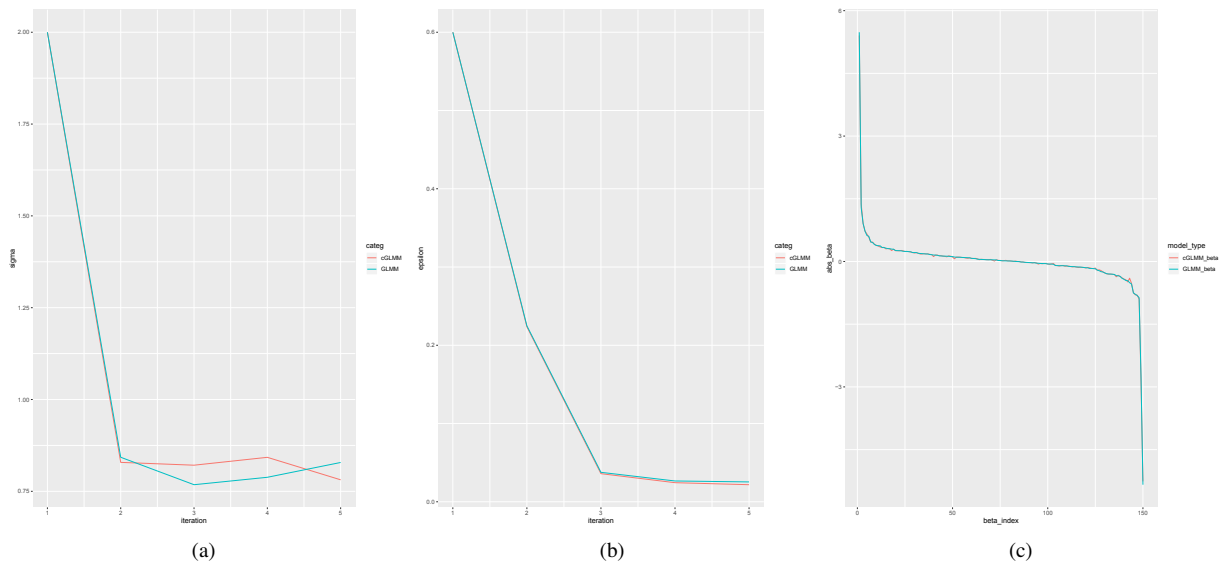
No. of SNPs	iterations		running time (mins)		communication round(cGLMM)	communication size(cGLMM) (KB)	PCC*
	GLMM	cGLMM	GLMM	cGLMM			
50	4	4	0.05	1.6	3006	731.84	0.99994
100	5	5	0.06	3.1	4004	4749.71	0.99999
150	5	5	0.2	5.0	4006	17121.65	0.99996

datasets. Similar to the results from the simulated data, both algorithms converge after a few EM iterations, while cGLMM runs about 30 times slower than GLMM, which is mostly due to the communication overhead. In all cases, cGLMM reported fixed effect coefficients that nearly identical to the results from GLMM (Pearson Correlation Coefficient PCC > 0.99).

Figure 5 compares the results of GLMM and cGLMM on the real human genomic dataset containing 150 SNPs (including 10 significant SNPs between the two groups) over the 2000 genomes in two groups. Similar to the results from the simulated datasets, the cGLMM algorithm converges as fast as the GLMM algorithm (Fig. 5a and 5b), and achieved nearly identical results as the regular GLMM algorithm (Fig. 5c), where the 10 SNPs (leftmost and rightmost SNPs in Figure 5c) which are highly correlated with the two groups receive high fixed effect coefficients. The testing on the other two datasets (containing 50 and 100 SNPs) showed similar results (see Supplemental Figures 3-4). These results confirmed the satisfactory performance of our cGLMM algorithm on real-world human genomic data.

4 Discussions

Our current collaborative EM algorithm follows a distributed computing approach, in which the input data are partitioned and the computation is split among participating institutions. As a result, each partition of the input data can remain on the server of the respective participant that holds the partial data, and only the intermediate results (e.g., the partial proposal probabilities and the partial Hessian matrix) need to be sent out. Still, some computation needs to be performed on a central server, and the intermediate results need to be communicated between the central server and the server at each participating party. Our experimental results showed that even though several thousands rounds of communication are



(a) (b) (c)

Fig. 5. The comparison between the results from cGLMM and GLMM on the human genomic dataset in a typical run, containing 150 SNPs (including 10 SNPs significantly associated with the two groups), for the distances between the fixed effect coefficients (β) (a), and the distances between the random effect parameters (σ) in consecutive iterations of the collaborative (in red) and non-collaborative (in blue) EM algorithm, as well as the fixed effect coefficients (β, sorted in decreasing order) reported by GLMM (in blue) and cGLMM (in red), respectively.

required to complete a reasonable-size GLMM task, the total amount of data transferred between servers is moderate (less than 100 MB). Therefore, we think our privacy-preserving algorithm can be practically used for collaboratively building GLMMs. However, we note that the many rounds of communication due to the required data exchange in each iteration of the MH algorithm may reduce the applicability of the method. We plan to explore the approximation approaches to the MH algorithm that may reduce the rounds of communication while without sacrificing its accuracy.

We note that our current approach does not protect the intermediate results sent by each participating institution to the central server: they are sent in plain text. The assumption here is that the intermediate results do not carry sufficient sensitive information about the individual records, which is commonly adopted by existing privacy-preserving algorithms for other biomedical computation tasks, e.g., for logistic regression Wu *et al.* (2012). However, our method can be combined with encryption methods to provide additional protection on the intermediate results. For example, we may use homomorphic encryption (HE) Kim *et al.* (2018) to compute the proposal probability (Equation 11), the Hessian matrix and the first-order derivatives (Equation 14), respectively, on the central server. Because only additions are needed for these computations, one can use the light-weight Paillier cryptosystem with additive homomorphic properties Parmar *et al.* (2014), which introduces moderate overhead on running time and communication. A more efficient solution may also be implemented using the recently available Trusted Execution Environments (TEEs) Chen *et al.* (2017b,a, 2016), such as Intel's Software Guard Extensions (SGX) McKeen *et al.* (2016).

Our current approach addresses the collaborative GLMM construction on horizontally partitioned data, when each participating institution holds the complete records of a subset of subjects. In comparison, the data may be *vertically* partitioned in some scenarios of collaborative computation. For examples, two medical institutions may have complementary clinical data (i.e., a subset of fixed effects) of the same patients, and attempt to combine their partial data for some joint analyses. Privacy-preserving methods have been developed for some data mining and machine learning algorithms on vertically partitioned data, e.g., for association rule mining **?**, k-means clustering Vaidya and Clifton (2003), naive Bayes classifier Vaidya and Clifton (2004) and support vector machine (SVM) Yu *et al.* (2006). We plan to develop privacy-preserving algorithms for collaborative GLMM construction on vertically partitioned data in the future.

Funding information. The research was partially supported by the National Institute of Health grants U01EB023685 and R01HG010798, and Indiana University (IU) Precision Health Initiative (PHI).

Acknowledgements. We thank Diyue Bu for her help of using the human genomic data from the 1000 Genomes Project.

References

Begum, F. *et al.* (2012). Comprehensive literature review and statistical considerations for gwas meta-analysis. *Nucleic acids research*, **40**(9), 3777–3784.

Booth, J. G. and Hobert, J. P. (1999). Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**(1), 265–285.

Bradburn, M. J. *et al.* (2003). Survival analysis part ii: multivariate data analysis–an introduction to concepts and methods. *British journal of cancer*, **89**(3), 431–436.

Chen, F. *et al.* (2016). Premix: Privacy-preserving estimation of individual admixture. In *AMIA Annual Symposium Proceedings*, volume 2016, page 1747. American Medical Informatics Association.

Chen, F. *et al.* (2017a). Presage: Privacy-preserving genetic testing via software guard extension. *BMC medical genomics*, **10**(2), 48.

Chen, F. *et al.* (2017b). Princess: Privacy-protecting rare disease international network collaboration via encryption through software guard extensions. *Bioinformatics*, **33**(6), 871–878.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The american statistician*, **49**(4), 327–335.

El Emam, K. *et al.* (2011). A systematic review of re-identification attacks on health data. *PloS one*, **6**(12).

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178.

Golan, D. and Rosset, S. (2018). Mixed models for case-control genome-wide association studies: major challenges and partial solutions. *Borgan, Breslow N, Chatterjee N, et al.(1st edn). Handbook of Statistical Methods for Case-Control Studies. Boca Raton, FL: Chapman and Hall/CRC*, pages 495–514.

Hirschhorn, J. N. and Daly, M. J. (2005). Genome-wide association studies for common diseases and complex traits. *Nature reviews genetics*, **6**(2), 95–108.

Jagannathan, G. and Wright, R. N. (2008). Privacy-preserving imputation of missing data. *Data & Knowledge Engineering*, **65**(1), 40–56.

Jeck, W. R. *et al.* (2012). a meta-analysis of gwas and age-associated diseases. *Aging cell*, **11**(5), 727–731.

Jiang, W. *et al.* (2013). Webglore: a web service for grid logistic regression. *Bioinformatics*, **29**(24), 3238–3240.

Kim, M. *et al.* (2018). Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, **6**(2), e19.

Konečnỳ, J. *et al.* (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Li, Y. *et al.* (2016). Vertical grid logistic regression (vertigo). *Journal of the American Medical Informatics Association*, **23**(3), 570–579.

Lu, C.-L. *et al.* (2015). Webdisco: a web service for distributed cox model learning without patient-level data sharing. *Journal of the American Medical Informatics Association*, **22**(6), 1212–1219.

McCarthy, M. I. *et al.* (2008). Genome-wide association studies for complex traits: consensus, uncertainty and challenges. *Nature reviews genetics*, **9**(5), 356–369.

McCulloch, C. (2003a). Chapter 4: Generalized linear mixed models (glmms). *Generalized linear mixed models*, **7**, 28–33.

McCulloch, C. E. (2003b). Generalized linear mixed models. In *NSF-CBMS regional conference series in probability and statistics*, pages i–84. JSTOR.

McKeen, F. *et al.* (2016). Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, pages 1–9.

Parmar, P. V. *et al.* (2014). Survey of various homomorphic encryption algorithms and schemes. *International Journal of Computer Applications*, **91**(8).

Pharoah, P. D. *et al.* (2013). Gwas meta-analysis and replication identifies three new susceptibility loci for ovarian cancer. *Nature genetics*, **45**(4), 362–370.

Sabt, M. *et al.* (2015). Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64. IEEE.

Sciacchitano, A. *et al.* (2015). Collaborative framework for piv uncertainty quantification: comparative assessment of methods. *Measurement Science and Technology*, **26**(7), 074004.

Vaidya, J. and Clifton, C. (2003). Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215.

Vaidya, J. and Clifton, C. (2004). Privacy preserving naive bayes classifier for vertically partitioned data. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 522–526. SIAM.

Wang, S. *et al.* (2013). Expectation propagation logistic regression (explorer): distributed privacy-preserving online model learning. *Journal of biomedical informatics*, **46**(3), 480–496.

Wang, S. *et al.* (2016). Healer: homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. *Bioinformatics*, **32**(2), 211–218.

Wang, X. *et al.* (2018). idash secure genome analysis competition 2017.

Wu, Y. *et al.* (2012). Grid binary lo gistic re gression (glore): building shared models without sharing data. *Journal of the American Medical Informatics Association*, **19**(5), 758–764.

Yu, H. *et al.* (2006). Privacy-preserving svm classification on vertically partitioned data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 647–656. Springer.

Yu, S. *et al.* (2008). Privacy-preserving cox regression for survival analysis. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1034–1042.