Revisiting VERTIGO and VERTIGO-CI: Identifying confidentiality breaches and introducing a statistically sound, efficient alternative

Marie-Pier Domingue[1,2], Jean-François Ethier[3,4*], Jean-Philippe Morissette[4,5], Simon Lévesque[1,2], Anita Burgun[2,6], Félix Camirand Lemyre[1,4]

[1]Groupe de recherche interdisciplinaire en informatique de la santé (GRIIS) (Mathematics), https://griis.ca/, Université de Sherbrooke, Sherbrooke,Canada.
[2]Institut Imagine, Université Paris Cité, Paris, France.
[3]Groupe de recherche interdisciplinaire en informatique de la santé (GRIIS) (Medicine/Computer science), https://griis.ca/, Université de Sherbrooke, Sherbrooke, Canada.
[4]Health Data Research Network Canada.
[5]Groupe de recherche interdisciplinaire en informatique de la santé (GRIIS) (Medicine), https://griis.ca/, Université de Sherbrooke, Sherbrooke, Canada.
[6]Department of Medical Informatics, Necker Hospital, AP-HP, Paris, France.

*Corresponding author(s). E-mail(s): jean-francois.ethier@usherbrooke.ca;

Abstract

Background: Health Data Research Network Canada is tasked with facilitating large-scale health data research, such as statistical analyses that integrate, within a single model, data collected by different organizations, each holding distinct subsets of features corresponding to the same individuals, thereby forming a vertical data partition. To support logistic regression analyses in this setting,

we assessed two recently proposed algorithms, VERTIGO and VERTIGO-CI, which enable parameter estimation and confidence interval computation, respectively, with respect to three aspects: the risk of re-identifying patient feature data, communication efficiency, and the extent to which model interpretability is preserved. This study has three main objectives: (1) highlighting confidentiality issues that arise with VERTIGO-CI, as well as those that may occur with VERTIGO when a data node holds only binary covariates; (2) reducing the number of required communication rounds; and (3) proposing an alternative (RidgeLog-V) to VERTIGO that excludes the intercept from the penalty term, which VERTIGO otherwise includes.
Methods: We inspected the quantities exchanged in the original algorithms and used linear algebra to identify reverse-engineering procedures that the coordinating center could employ to reconstruct raw data. We also analyzed the objective function of the optimization problem, leading to the proposal of an alternative formulation that requires only a single round of communication while allowing the intercept to be excluded from the penalty term.
Results: We showed that, when the VERTIGO-CI algorithm is executed, the coordinating center can reconstruct all individual-level data using simple vector-matrix operations. When the VERTIGO algorithm is executed and a data node has binary covariates only, the coordinating center may be able to recover individual data when parameter estimates are shared. We adapted the VERTIGO algorithm to reduce the number of communications and proposed a variant that excludes the intercept from the penalty term.
Conclusions: While the use of VERTIGO-CI, or of VERTIGO with binary covariates does not involve directly sharing raw data, confidentiality breaches may arise through reverse-engineering, illustrating that that the distributed nature of an algorithm does not inherently guarantee data privacy. This work also proposed a new algorithm (RidgeLog-V) that reduces operational costs and enhances model interpretability.

Keywords: Vertically partitioned data, Data privacy, Distributed analysis, Logistic regression model, Distributed algorithm

1 Background

As the health research and management community strives to improve care and ultimately health outcomes for citizens, the questions being asked require access to more diverse and extensive datasets. From genomic to environmental data, and from quantified-self information to clinical care data warehouses, the factors influencing health outcomes are often collected by various organizations and stored across different systems. Traditionally, the standard approach to analysis has been to centralize

data by copying it to a single location. However, the growing variety, scale, and geographical distribution of the data needed often make this impractical. Legal, ethical, and social acceptability constraints frequently arise, preventing data from being shared or centralized. Health Data Research Network Canada (HDRN) is tasked with facilitating large-scale health data research, such as statistical analyses that integrate, within a single model, individual-level data collected by different organizations—for example, linking genomic data from participants in the Canadian Partnership for Tomorrow's Health (CanPath) cohort [1] with their corresponding provincial medico-administrative records. This scenario illustrates a case of vertically partitioned data [2], where different institutions hold distinct subsets of features corresponding to the same individuals. In such settings, the intended analyses often involve estimating parameters of statistical models and their associated confidence intervals (CIs). To support these objectives, we explored methods designed to accommodate such data-sharing constraints across institutions in the context of vertically partitioned data, with a focus on logistic regression—widely used in health research for modeling binary outcomes.

The field of distributed analysis offers relevant approaches in which analytical results are produced solely through the exchange of summary statistics. However, the absence of direct raw data sharing does not inherently guarantee confidentiality, as participating entities may still infer sensitive information by reverse-engineering the exchanged quantities, a risk that has been illustrated in the literature [3], in a context however different from the one considered in this paper.

In supporting vertically distributed logistic regressions, the ideal method does not require any raw data to leave the environments of the original data stewards (data nodes); does provide formal confidentiality guarantees, including protection against reverse-engineering; offers documented use in clinical research settings; is communication-efficient, as most data partners require human review of any information leaving a secure environment; and yields results that are comparable to those from

classical pooled, maximum likelihood-based analyses (see e.g. [4]), which is important to reassure colleagues that the method is a valid alternative to the centralized approaches they have relied on until now. However, our review of the literature did not identify any method that satisfies all these criteria.

To our knowledge, only one method [5, 6] has been proposed that addresses the maximum likelihood-based logistic regression inference problem considered here, while fully avoiding the need for raw data to leave its original environment (while CIs are not specifically covered, it is straightforward to see that they can be obtained by a minor modification of the procedure described). However, as noted by the original authors and in a recent review [7], it requires a high number of communication rounds between parties, and concerns about data leakage persist—issues that may be inherently difficult to resolve. Taken together, these limitations rendered the method unsuitable for our intended applications.

The next step was to identify a method with minimal data-sharing requirements. The VERTIGO algorithm (JAMIA 2016) [8] and its extension developed by the same group, VERTIGO-CI (AMIA 2021) [9], have recently been proposed for the analysis of vertically partitioned data. They have since been applied in distributed health data research [10], including a study published in *Nature Communications* [11]. To implement the method, the vector of binary responses must be accessible to all involved parties, including the coordinating center (CC), which is responsible for aggregating the quantities shared by parties contributing with local data (called *nodes*). While this requires sharing the outcome outside the originating site, we hypothesized that it might be helpful to support some projects at HDRN if the other criteria were met, given the authors' assertion that the VERTIGO algorithm "harmonize[s] information in a privacy-preserving manner" [9].

In order to ensure due diligence with respect to HDRN data partners, we conducted an assessment of VERTIGO and VERTIGO-CI regarding the risk of re-identifying

patient feature data, the communication efficiency, and the extent to which they preserve the integrity of model interpretation. Four areas of concern were identified:

- **Confidentiality:** The use of VERTIGO-CI enables reverse engineering at the CC level, meaning that the quantities exchanged by local data nodes allow the CC to reconstruct individual feature data using simple vector-matrix operations;
- **Confidentiality:** The use of VERTIGO (estimation only, without CIs) may allow the CC to reconstruct individual feature data when a data includes holds binary features and discloses associated parameter estimates;
- **Communication:** The use of VERTIGO (estimation only, without CIs) involves non-essential communication rounds between local data nodes and the CC;
- **Comparability:** The use of VERTIGO (estimation only, without CIs) incorporates the intercept in the penalty term. However, in most applications of penalized logistic regression, the intercept is excluded from the penalty [12]. As a result, VERTIGO estimates are not directly comparable to those obtained from widely adopted algorithms such as the one proposed in [13], leading to inconsistencies in interpretation.

In light of these observations, the primary aim of this article is to guide the future use of this method. Specifically, our objectives are to: (1a) highlight the confidentiality issues associated with VERTIGO-CI and (1b) the privacy breaches that may arise from estimates disclosed when a node includes binary features under VERTIGO, (2) propose an optimized way to perform the operations on shared quantities to reduce the number of communication rounds, and (3) propose an alternative approach to VERTIGO that excludes the intercept from the penalty.

While the method discussed below is supported by mathematical arguments, we also illustrate the different aims with R code [14] available at https://github.com/OpenLHS/Revisiting_VERTIGO using a publicly available dataset that was previously analyzed in the original VERTIGO-CI paper. For issues 1a and 1b, we numerically

demonstrate how vector-matrix operations can be used by the CC to reconstruct data from other nodes using shared quantities.

The implementation of our alternative approach—named RidgeLog-V (Ridge Logistic Regression from Vertically Partitioned Data)—which is more communication-efficient and excludes the intercept from the penalty, is also available in the same repository. We illustrate that RidgeLog-V produces results that are coherent—yielding near-identical estimates to those obtained using glmnet [13]—while differing from those produced by the VERTIGO algorithm.

2 Methods

2.1 General methodological framework and mathematical foundations

The work presented here is expressed using a mathematical notation framework aligned with the one proposed in our previous work on horizontally distributed analyses [15]. The general methodological framework, including mathematical notations, is described below.

2.1.1 Mathematical notations and definitions

In the following, non-italic uppercase letters represent random variables, while non-italic bold uppercase letters denote random vectors. Vector-valued quantities and matrices are represented by lowercase italic bold letters and uppercase italic bold letters, respectively. The jth element of a vector $\boldsymbol{a} \in \mathbb{R}^p$ is denoted by a_j. For any vector $\boldsymbol{a} \in \mathbb{R}^p$, $\text{diag}(\boldsymbol{a})$ represents the $p \times p$ diagonal matrix with the entries of $\boldsymbol{a}$ on the diagonal. In the case of iterative computations over $\boldsymbol{a}$, the step count s appears as an index in parentheses (e.g., $\boldsymbol{a}_{(s)}$). The gradient of a function $f(\boldsymbol{\theta})$ with parameter $\boldsymbol{\theta}$ evaluated at $\boldsymbol{\theta} = \boldsymbol{a}$ is denoted as the column vector $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{a}) := \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{a}}$. Notation glossaries are provided in Additional file 1-A.

We use the following formulation of the binary logistic regression model with p covariates. Let $\mathrm{Y} \in \{-1, 1\}$ be a random variable representing the binary response, and let the corresponding random vector of covariates be denoted by $\mathbf{X} \in \mathbb{R}^p$. The conditional distribution of $\mathrm{Y} \mid \mathbf{X} = \boldsymbol{x}$ is given, for $y \in \{-1, 1\}$, by

$$\mathrm{P}(\mathrm{Y} = y \mid \mathbf{X} = \boldsymbol{x}) = \frac{1}{1 + \exp\{-y(\beta_{0\star} + \boldsymbol{x}^\top \boldsymbol{\beta}_\star)\}}, \tag{1}$$

where $\beta_{0\star} \in \mathbb{R}$ and $\boldsymbol{\beta}_\star = [\beta_{\star 1}, \ldots, \beta_{\star p}]^\top \in \mathbb{R}^p$ are respectively the true unknown intercept and the true unknown covariate parameters. The data involved in the analysis consists in a sample of n realizations of the random vector $(\mathbf{X}, \mathrm{Y})$, denoted by $\mathcal{D} := \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$, with $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{ip}]^\top$.

In the definition of the logistic model above, we adopt a different notation convention for the intercept parameter compared to the original VERTIGO and VERTIGO-CI articles. Specifically, we separate the intercept parameter from the covariate-associated parameters. In the original papers, a value of 1 is included in the covariate vector $\mathbf{x}$, and the intercept parameter is treated as part of the covariate-associated parameters.

We represent the covariate data of the n individuals in matrix form and the response data in a vector form such that

$$\boldsymbol{X} := \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}, \text{ and } \boldsymbol{y} := \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}. \tag{2}$$

Our derivations involve the Gram matrix $\boldsymbol{\mathcal{K}}$ defined as follows, where $\mathbf{1}_n \in \mathbb{R}^n$ denotes the vector with all entries equal to one:

$$\boldsymbol{\mathcal{K}} := [\boldsymbol{X}\ \mathbf{1}_n][\boldsymbol{X}\ \mathbf{1}_n]^\top = \boldsymbol{X}\boldsymbol{X}^\top + \mathbf{1}_n \mathbf{1}_n^\top$$

$$= \begin{bmatrix} \sum_{i=1}^p x_{1i}^2 + 1 & \cdots & \sum_{i=1}^p x_{1i}x_{ni} + 1 \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^p x_{ni}x_{1i} + 1 & \cdots & \sum_{i=1}^p x_{ni}^2 + 1 \end{bmatrix}. \tag{3}$$

The (i, j) entry of $\boldsymbol{\mathcal{K}}$ for $i, j \in \{1, \ldots, n\}$ will be denoted $\mathcal{K}_{ij}$. It can be readily deduced from (3) that the following holds for all $i, j \in \{1, \ldots, n\}$:

$$\mathcal{K}_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j + 1\,. \tag{4}$$

2.1.2 Vertically partitioned data setting and properties of the Gram matrix

In the case of vertically partitioned data, where covariate data are distributed across K locations (referred to as nodes), the matrix $\boldsymbol{X}$ is partitioned into K submatrices denoted as $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(K)}$. Each node $k \in \{1, \ldots, K\}$ holds a matrix $\boldsymbol{X}^{(k)} \in \mathbb{R}^{n \times p^{(k)}}$ containing $p^{(k)}$ covariates, ensuring that $\sum_{k=1}^K p^{(k)} = p$. We assume that the columns of $\boldsymbol{X}$ are arranged such that the first $p^{(1)}$ columns correspond to the covariates stored at node 1, followed by the next $p^{(2)}$ columns corresponding to those stored at node 2, and so forth. Under this ordering, $\boldsymbol{X}$ can be expressed in a block matrix notation as

$$\boldsymbol{X} = \left[\boldsymbol{X}^{(1)}\ \boldsymbol{X}^{(2)}\ \cdots\ \boldsymbol{X}^{(K-1)}\ \boldsymbol{X}^{(K)}\right].$$

For $k \in \{1, \ldots, K\}$, the ith row of $\boldsymbol{X}^{(k)}$, which we will denote by $\boldsymbol{x}_i^{(k)}$, represents the $p^{(k)}$ covariate values of individual i stored at node k. Using (4), it follows that

$$\mathcal{K}_{ij} = \sum_{k=1}^K (\boldsymbol{x}_i^{(k)})^\top \boldsymbol{x}_j^{(k)} + 1\,. \tag{5}$$

In VERTIGO and VERTIGO-CI, each node $k \in \{1, \ldots, K\}$ is assumed to have access to the response vector $\boldsymbol{y}$ in addition to the covariate data $\boldsymbol{X}^{(k)}$. It is also assumed that the datasets are aligned across all partitions $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(K)}$ and that this alignment corresponds to the structure of $\boldsymbol{y}$, i.e., Patient 1 appears in row 1 of all partitions and in the first entry of $\boldsymbol{y}$, and so forth. We will maintain these assumptions throughout.

2.1.3 Penalized logistic regression

VERTIGO and VERTIGO-CI rely on a penalized maximum likelihood estimation procedure, specifically the ridge logistic regression [16], to estimate the unknown coefficients $\beta_{0\star}$ and $\boldsymbol{\beta}_\star$ in (1). A strictly positive parameter λ is used to control the strength of the penalty. In these algorithms, the penalty is applied to all model coefficients, including the intercept. The corresponding optimization problem is given by

$$\begin{aligned} &\max_{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^\top \in \mathbb{R}^p} \left(l_n^\lambda(\beta_0, \boldsymbol{\beta}) := \ell_n(\beta_0, \boldsymbol{\beta}) - \frac{\lambda}{2} \sum_{j=0}^{p} \beta_j^2 \right), \\ &\text{where } \ell_n(\beta_0, \boldsymbol{\beta}) := \sum_{i=1}^{n} \log \left(\frac{1}{1 + \exp\{-y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\}} \right) . \end{aligned} \tag{6}$$

For $\lambda > 0$, the optimization problem in (6) always has a unique solution. Let $\widehat{\beta}_0^\lambda$ and $\widehat{\boldsymbol{\beta}}^\lambda$ denote this solution.

As mentioned in the introduction, in most applications, the intercept is excluded from the penalty term [13]. When this is the case, the following alternative penalized maximum log-likelihood optimization problem is used to derive the model parameter estimates:

$$\max_{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^\top \in \mathbb{R}^p} \left(l_{n,\text{alt}}^\lambda(\beta_0, \boldsymbol{\beta}) := \ell_n(\beta_0, \boldsymbol{\beta}) - \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2 \right), \tag{7}$$

where $\ell_n(\beta_0, \boldsymbol{\beta})$ is as in (6). For $\lambda > 0$, when there is at least one y_i equal to 1 and one y_i equal to 0, the optimization problem in (7) always has a unique solution (see Additional file 1-D). Let $\widehat{\beta}^{\lambda}_{0,\text{alt}}$ and $\widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}}$ denote this solution.

2.2 Overview of VERTIGO

The VERTIGO algorithm leverages an application of dual optimization techniques, which shows that, under our notations, the unique solution $(\widehat{\beta}^{\lambda}_0, \widehat{\boldsymbol{\beta}}^{\lambda})$ to the problem in (6) can be computed as

$$\widehat{\beta}^{\lambda}_0 = \frac{1}{\lambda} \sum_{i=1}^{n} \widehat{\alpha}^{\lambda}_i y_i \,, \quad \widehat{\boldsymbol{\beta}}^{\lambda} = \frac{1}{\lambda} \sum_{i=1}^{n} \widehat{\alpha}^{\lambda}_i y_i \boldsymbol{x}_i \,, \tag{8}$$

where $\widehat{\boldsymbol{\alpha}}^{\lambda} := [\widehat{\alpha}^{\lambda}_1, \ldots, \widehat{\alpha}^{\lambda}_n]^{\top} \in (0,1)^n$ is the unique solution to the following minimization problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha} \in (0,1)^n} \left(J^{\lambda}(\boldsymbol{\alpha}) := \frac{1}{2\lambda} \sum_{i=1}^{n} \sum_{j=1}^{n} \{\alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^{\top} \boldsymbol{x}_j + 1)\} + \sum_{i=1}^{n} L(\alpha_i) \right), \\ &\text{with } L(\alpha_i) := (1 - \alpha_i) \log(1 - \alpha_i) + \alpha_i \log(\alpha_i) \,. \end{aligned} \tag{9}$$

The minimization problem in (9) is called the dual problem to the maximization problem in (6), where the vector $\boldsymbol{\alpha}$ is the dual parameter.

To compute $\widehat{\boldsymbol{\alpha}}^{\lambda}$ in the vertically partitioned setting described above, the VERTIGO algorithm employs an iterative Newton method. At each iteration, the gradient and Hessian of $J^{\lambda}(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ can be computed exactly at the CC by requiring each node to send intermediate numerical outputs. Specifically, for $k \in \{1, \ldots, K\}$, letting $\boldsymbol{e}^{(k)}(\boldsymbol{\alpha})$ denote a vector whose j^{th} entry is given by

$$e^{(k)}_j(\boldsymbol{\alpha}) = \begin{cases} \lambda^{-1} \sum_{i=1}^{n} \alpha_i y_i y_j (\boldsymbol{x}^{(k)}_i)^{\top} \boldsymbol{x}^{(k)}_j & \text{if } k \leq K - 1 \\ \lambda^{-1} \sum_{i=1}^{n} \alpha_i y_i y_j \{(\boldsymbol{x}^{(k)}_i)^{\top} \boldsymbol{x}^{(k)}_j + 1\} & \text{if } k = K \,, \end{cases}$$

the gradient of $J^\lambda(\boldsymbol{\alpha})$ in (9) is expressed as

$$\nabla_{\boldsymbol{\alpha}} J^\lambda(\boldsymbol{\alpha}) = \sum_{k=1}^{K} \boldsymbol{e}^{(k)}(\boldsymbol{\alpha}) + \left[\log\left(\frac{\alpha_1}{1-\alpha_1}\right), \cdots, \log\left(\frac{\alpha_n}{1-\alpha_n}\right)\right]^\top . \tag{10}$$

The distinct definitions for nodes $k \in \{1, \ldots, K-1\}$ and node K reflect the fact that, in VERTIGO, a column of ones is appended to the data matrix $\boldsymbol{X}^{(K)}$ at node K to treat the intercept as part of the covariate-associated parameters.

The Hessian of $J^\lambda(\boldsymbol{\alpha})$ is expressed as

$$\boldsymbol{H}(\boldsymbol{\alpha}) = \frac{1}{\lambda} \operatorname{diag}(\boldsymbol{y}) \boldsymbol{\mathcal{K}} \operatorname{diag}(\boldsymbol{y}) + \operatorname{diag}\left\{\left[\frac{1}{\alpha_1(1-\alpha_1)}, \cdots, \frac{1}{\alpha_n(1-\alpha_n)}\right]^\top\right\}. \tag{11}$$

At the start of the VERTIGO algorithm, each node k shares the local matrix $\boldsymbol{\mathcal{K}}^{(k)}$ with the CC, where

$$\boldsymbol{\mathcal{K}}^{(k)} := \begin{cases} \boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top & \text{if } k \leq K-1 \\ \boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top + \mathbf{1}_n \mathbf{1}_n^\top & \text{if } k = K\,. \end{cases}$$

This allows the CC to construct $\boldsymbol{\mathcal{K}}$ because $\boldsymbol{\mathcal{K}} = \sum_{k=1}^{K} \boldsymbol{\mathcal{K}}^{(k)}$ (from (5)). Starting with $\boldsymbol{\alpha}_{(0)}$ shared across all nodes, at each iteration $s \geq 0$ until convergence, each node sends its locally updated quantity $\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$ to the CC. The CC then updates $\boldsymbol{\alpha}_{(s)}$ using the Newton-Raphson algorithm such that $\boldsymbol{\alpha}_{(s+1)} = \boldsymbol{\alpha}_{(s)} - \boldsymbol{H}^{-1}(\boldsymbol{\alpha}_{(s)}) \nabla_{\boldsymbol{\alpha}} J^\lambda(\boldsymbol{\alpha}_{(s)})$, and subsequently sends the updated dual parameters to all nodes.

Once the optimal dual parameter $\widehat{\boldsymbol{\alpha}}^\lambda$ has been obtained and shared with all nodes, each node k computes the components of $\widehat{\boldsymbol{\beta}}^\lambda$ corresponding to the covariates stored

at their node using the equation $\widehat{\boldsymbol{\beta}}^{\lambda(k)} = \lambda^{-1}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i \boldsymbol{x}_i^{(k)}$ where this relationship follows from (8). The coefficient $\widehat{\beta}_0^{\lambda}$ is computed at node K using the formula $\lambda^{-1}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i$.

2.2.1 Overview of VERTIGO-CI

The VERTIGO-CI algorithm [9] is an extension of the algorithm VERTIGO that aims at estimating the variance-covariance matrix of the estimator $(\widehat{\boldsymbol{\beta}}^{\lambda}, \widehat{\beta}_0^{\lambda})$ using $\{[\boldsymbol{X}\,\boldsymbol{1}_n]^\top \boldsymbol{V}^{\lambda}[\boldsymbol{X}\,\boldsymbol{1}_n]\}^{-1}$, where $\boldsymbol{V}^{\lambda}$ denotes a diagonal matrix whose entry (i,i) is given by

$$V_{ii}^{\lambda} = \left(\frac{\exp\{\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}\}}{1+\exp\{\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}\}}\right) \times \left(1 - \frac{\exp\{\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}\}}{1+\exp\{\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}\}}\right). \tag{12}$$

To achieve this, the VERTIGO-CI algorithm (provided in Additional file 1-B, see Algorithm 4 therein) begins by executing the VERTIGO algorithm to obtain $(\widehat{\boldsymbol{\beta}}^{\lambda}, \widehat{\beta}_0^{\lambda})$ with a simplified Hessian matrix. Once the parameter estimates obtained, the nodes compute the matrix $\boldsymbol{V}^{\lambda}$ using node-to-node communications (step 8 Algorithm 4 in Additional file 1-B). Every node can then compute and send $(\boldsymbol{X}^{(k)})^\top(\boldsymbol{V}^{\lambda})^{1/2}$ to the CC ($[\boldsymbol{X}^{(k)}\,\boldsymbol{1}_n]^\top(\boldsymbol{V}^{\lambda})^{1/2}$ for $k=K$). Finally, the CC builds the matrix $[\boldsymbol{X}\,\boldsymbol{1}_n]^\top \boldsymbol{V}^{\lambda}[\boldsymbol{X}\,\boldsymbol{1}_n]$ by block.

The quantities shared between the data nodes and the CC throughout the algorithms VERTIGO and VERTIGO-CI are summarized in Table 1.

Table 1 Overview of the information shared in VERTIGO and VERTIGO-CI

Step	From nodes to CC	From CC to nodes
Initialization[1]	$\boldsymbol{\mathcal{K}}^{(k)}$	-
Every iteration s	$\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$	$\boldsymbol{\alpha}_{(s)}$
Estimation results	$\widehat{\boldsymbol{\beta}}^{\lambda(k)}$	-
CIs (VERTIGO-CI)	$(\boldsymbol{X}^{(k)})^\top(\boldsymbol{V}^{\lambda})^{1/2}$	-

[1] It is assumed that all parties (nodes and CC) have access to $\boldsymbol{y}$ and λ.

2.3 Methodology pertaining to objective 1

To highlight the confidentiality issues of VERTIGO-CI, using the list of available quantities at the CC in Table 1, we demonstrate how one can construct an invertible system of equations that, through linear algebra manipulations, enables the recovery of each node's data matrix $\boldsymbol{X}^{(k)}$.

To highlight the privacy breaches that may arise when a node includes binary variables and shares the associated parameter estimates under VERTIGO, we present a case in which the CC is able to reverse-engineer individual-level data for $\boldsymbol{X}^{(k)}$ involving solely binary features. This is achieved using the local Gram matrix $\boldsymbol{\mathcal{K}}^{(k)}$ and the estimated coefficients $\widehat{\boldsymbol{\beta}}^{\lambda(k)}$ shared from node k. The case also relies on the following property of binary matrices, which can be established using basic tools from linear algebra:

> Let $\boldsymbol{A}$ denote an $n \times p$ matrix with binary (0-1) entries. Consider $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{d} \in \mathbb{R}^p$ such that $\boldsymbol{a} = \boldsymbol{A}\boldsymbol{d}$, and let $\mathcal{U}(\boldsymbol{d}) := \{z \in \mathbb{R} : z = \boldsymbol{b}^\top \boldsymbol{d},\ \boldsymbol{b} \in \{0,1\}^p\}$. If $\mathcal{U}(\boldsymbol{d})$ has cardinality 2^p, then any binary matrix $\boldsymbol{A}'$ satisfying $\boldsymbol{a} = \boldsymbol{A}'\boldsymbol{d}$ must be equal to $\boldsymbol{A}$.

2.4 Methodology pertaining to objective 2

Our optimization of the VERTIGO algorithm's execution relies on the observation that, as the term $\boldsymbol{x}_i^\top \boldsymbol{x}_j + 1$ in the definition of $J^\lambda(\boldsymbol{\alpha})$ in (9) is equal to $\mathcal{K}_{ij}$ (see equation (4)), $J^\lambda(\boldsymbol{\alpha})$ can be expressed as

$$J^\lambda(\boldsymbol{\alpha}) = \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \operatorname{diag}(\boldsymbol{y})\boldsymbol{\mathcal{K}} \operatorname{diag}(\boldsymbol{y})\boldsymbol{\alpha} + \sum_{i=1}^{n} L(\alpha_i)\,. \tag{13}$$

Based on this observation (details in Additional file 1 - C), we designed an algorithm that requires only a single communication from the nodes to the CC, in which the nodes transmit the same quantities they send in Step 1 of the VERTIGO algorithm.

2.5 Methodology pertaining to objective 3

The design of RidgeLog-V, our proposed alternative approach to VERTIGO which excludes the intercept from the penalty, leverages dual optimization techniques very similar to those used in VERTIGO to derive the expression of $J^\lambda(\boldsymbol{\alpha})$. These techniques show that, if there exists at least one i such that $y_i = 1$, at least one j such that $y_j = -1$ and $\lambda > 0$, the solution $(\widehat{\beta}_{0,\text{alt}}, \widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}})$ of the optimization problem (7) is unique and satisfies:

$$\begin{aligned} \widehat{\beta}_{0,\text{alt}} &= \frac{1}{n}\sum_{i=1}^n \left(y_i \log\{(\widehat{\alpha}^\lambda_{i,\text{alt}})^{-1} - 1\} - \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}}\right), \\ \widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}} &= \lambda^{-1}\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{y})\widehat{\boldsymbol{\alpha}}^\lambda_{\text{alt}}, \end{aligned} \tag{14}$$

where $\widehat{\boldsymbol{\alpha}}^\lambda_{\text{alt}} = [\widehat{\alpha}^\lambda_{1,\text{alt}}, \ldots, \widehat{\alpha}^\lambda_{n,\text{alt}}]^\top$ is the unique solution to the following minimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} J^\lambda_{\text{alt}}(\boldsymbol{\alpha}) \quad \text{s.t.} \sum_{i=1}^n y_i\alpha_i = 0, \\ &\text{where } J^\lambda_{\text{alt}}(\boldsymbol{\alpha}) = \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \operatorname{diag}(\boldsymbol{y})\boldsymbol{X}\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{y})\boldsymbol{\alpha} + \sum_{i=1}^n L(\alpha_i). \end{aligned} \tag{15}$$

The proof of the latter result is available in Additional file 1-D.

To show how $\widehat{\beta}_{0,\text{alt}}$ can be computed in our vertically partitioned data setting, we will also use the fact that as (14) implies $\boldsymbol{X}\widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}} = \lambda^{-1}\boldsymbol{X}\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{y})\widehat{\boldsymbol{\alpha}}^\lambda_{\text{alt}}$, it holds that

$$n^{-1}\sum_{i=1}^n \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}} = (n\lambda)^{-1}\mathbf{1}_n^\top \boldsymbol{X}\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{y})\widehat{\boldsymbol{\alpha}}^\lambda_{\text{alt}}. \tag{16}$$

3 Results

3.1 Results pertaining to objective 1: Confidentiality

We first show that once the VERTIGO-CI algorithm has been executed, the quantities available at the CC allow it to reverse-engineer the feature data of all individuals. We

then illustrate the privacy risks that can arise when binary features are used under VERTIGO and their associated parameter estimates are shared (even without using the CI extension).

3.1.1 VERTIGO-CI

Once VERTIGO-CI has been executed, the following quantities are available at the CC. The corresponding steps (indicated in parentheses) refer to those shown in the VERTIGO-CI algorithm reproduced in Additional file 1-(B) (see Algorithm 4):

- The response vector $\boldsymbol{y}$ and the parameter λ (as part of the input of the algorithm);
- The local matrices $\boldsymbol{\mathcal{K}}^{(1)}, \ldots, \boldsymbol{\mathcal{K}}^{(K)}$ (Step 1);
- $\widehat{\boldsymbol{\alpha}}^{\lambda}$ (Step 4);
- $(\boldsymbol{X}^{(1)})^{\top}(\boldsymbol{V}^{\lambda})^{1/2}, \ldots, (\boldsymbol{X}^{(K)})^{\top}(\boldsymbol{V}^{\lambda})^{1/2}$ (Steps 9 & 10).

We next show how the CC can reconstruct the individual-level data held by each node using these quantities. From the relationships in (8), it holds, for all $1 \leq j \leq n$, that

$$\begin{aligned}
\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda} &= \frac{1}{\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i + \boldsymbol{x}_j^{\top}\Big(\frac{1}{\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i \boldsymbol{x}_i\Big) \\
&= \frac{1}{\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i (\boldsymbol{x}_i^{\top}\boldsymbol{x}_j + 1) \\
&= \frac{1}{\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i \mathcal{K}_{ij}\,. \qquad (17)
\end{aligned}$$

To obtain the last equality, we used the relationship at (4). Since the CC can calculate each entry of the Gram matrix $\boldsymbol{\mathcal{K}} = \sum_{k=1}^{K}\boldsymbol{\mathcal{K}}^{(k)}$ (it does so at Step 2), and as it has access to $\widehat{\boldsymbol{\alpha}}^{\lambda}$, $\boldsymbol{y}$ and λ, the CC is able to compute $\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda}$ for each $j \in \{1, \ldots, n\}$ through the formula proved at (17).

Because it can compute $\widehat{\beta}_0^\lambda + \boldsymbol{x}_j^\top \widehat{\boldsymbol{\beta}}^\lambda$ for each $j \in \{1, \ldots, n\}$, the CC is therefore able to compute each entry of the diagonal matrix $\boldsymbol{V}^\lambda$ defined in (12) through the formula

$$V_{jj}^\lambda = \left(\frac{\exp\{\lambda^{-1} \sum_{i=1}^n \widehat{\alpha}_i^\lambda y_i \mathcal{K}_{ij}\}}{1 + \exp\{\lambda^{-1} \sum_{i=1}^n \widehat{\alpha}_i^\lambda y_i \mathcal{K}_{ij}\}} \right) \times \left(1 - \frac{\exp\{\lambda^{-1} \sum_{i=1}^n \widehat{\alpha}_i^\lambda y_i \mathcal{K}_{ij}\}}{1 + \exp\{\lambda^{-1} \sum_{i=1}^n \widehat{\alpha}_i^\lambda y_i \mathcal{K}_{ij}\}} \right) .$$

The last formula implies that the CC is also able to compute the inverse of the matrix $(\boldsymbol{V}^\lambda)^{1/2}$. Indeed, since $\boldsymbol{V}^\lambda$ is diagonal with strictly positive entries, $(\boldsymbol{V}^\lambda)^{-1/2}$ is also a diagonal matrix whose jth diagonal entry can be calculated by the CC using $\left[(\boldsymbol{V}^\lambda)^{-1/2}\right]_{jj} = \left(V_{jj}^\lambda\right)^{-1/2}$.

Recalling that the CC has access to $(\boldsymbol{X}^{(k)})^\top (\boldsymbol{V}^\lambda)^{1/2}$ for each $k \in \{1, \ldots, K\}$, and given that we have just shown how the CC can construct the matrix $(\boldsymbol{V}^\lambda)^{-1/2}$ using quantities available to it after the execution of the VERTIGO-CI algorithm, we conclude that the CC can reverse-engineer each data matrix $\boldsymbol{X}^{(k)}$ via the relationship

$$\begin{aligned} (\boldsymbol{V}^\lambda)^{-1/2} \left((\boldsymbol{X}^{(k)})^\top (\boldsymbol{V}^\lambda)^{1/2} \right)^\top &= (\boldsymbol{V}^\lambda)^{-1/2} (\boldsymbol{V}^\lambda)^{1/2} \boldsymbol{X}^{(k)} \\ &= \boldsymbol{X}^{(k)} . \end{aligned}$$

This is not the only way for the CC to proceed with reverse-engineering as another path can be used to construct the matrix $\boldsymbol{V}^\lambda$ at the CC (see Additional file 1-E for details).

The reverse-engineering process is numerically illustrated using R code available at https://github.com/OpenLHS/Revisiting_VERTIGO.

3.1.2 Binary features in VERTIGO

We now elaborate the case in which the CC is able to reverse-engineer an individual patient data matrix $\boldsymbol{X}^{(k)}$ comprising binary (0-1) features. We demonstrate it

for a case with all binary predictors but mix cases (binary and continuous) warrant additional investigations before confirming or infirming their confidentiality properties.

For simplicity, to avoid dealing with the intercept in our example, we choose $k \neq K$. We note that the relationship in (8) implies

$$(\boldsymbol{x}_j^{(k)})^\top \widehat{\boldsymbol{\beta}}^{\lambda(k)} = \frac{1}{\lambda} \sum_{i=1}^{n} \widehat{\alpha}_i^\lambda y_i \mathcal{K}_{ij}^{(k)} \qquad \text{for all } j \in \{1, \ldots, n\}. \tag{18}$$

Since the CC has access to the matrix $\boldsymbol{\mathcal{K}}^{(k)}$ and the response vector $\boldsymbol{y}$ as part of the VERTIGO algorithm's initialization, and also has access to $\widehat{\boldsymbol{\alpha}}^\lambda$, which is computed at the CC, it can compute $(\boldsymbol{x}_j^{(k)})^\top \widehat{\boldsymbol{\beta}}^{(k)}$ for all $j \in \{1, \ldots, n\}$ using formula (18). Therefore, when the parameter estimates $\widehat{\boldsymbol{\beta}}^{\lambda(k)}$ are made available to the CC, the only quantity unknown to the CC in (18) are the entries of $\boldsymbol{x}_j^{(k)}$. Using a vector-matrix notation, the CC can attempt to reverse-engineer the data matrix $\boldsymbol{X}^{(k)}$ by identifying a binary matrix $\boldsymbol{A}$ such that $\boldsymbol{A}\widehat{\boldsymbol{\beta}}^{\lambda(k)} = \lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)} \operatorname{diag}(\widehat{\boldsymbol{\alpha}}^\lambda)\boldsymbol{y}$. When the features of data node k are all binary, even if the number of unknowns exceeds the number of equations, if the cardinality of the set $\mathcal{U}(\widehat{\boldsymbol{\beta}}^{\lambda(k)})$ is equal to $2^{p^{(k)}}$, then any binary matrix satisfying $\boldsymbol{A}\widehat{\boldsymbol{\beta}}^{\lambda(k)} = \lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)} \operatorname{diag}(\widehat{\boldsymbol{\alpha}}^\lambda)\boldsymbol{y}$ must be equal to $\boldsymbol{X}^{(k)}$. In this case, to identify line j of $\boldsymbol{X}^{(k)}$, the CC has to search over all binary vectors $\boldsymbol{b} \in \{0,1\}^{p^{(k)}}$ and identify the unique one such that $\boldsymbol{b}^\top \widehat{\boldsymbol{\beta}}^{(k)}$ equals the j component of $\lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)} \operatorname{diag}(\widehat{\boldsymbol{\alpha}}^\lambda)\boldsymbol{y}$.

To illustrate this, consider the case $p^{(k)} = 3$, and suppose that, upon running the algorithm, the vector $\widehat{\boldsymbol{\beta}}^{\lambda(k)} = [0.5,\, 0.6\,, 0.7]^\top$ is obtained. In this case, it is straightforward to verify that the cardinality of the set $\mathcal{U}(\widehat{\boldsymbol{\beta}}^{\lambda(k)})$ is $8 = 2^3$, and that the only possible values for the entries of $\lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)} \operatorname{diag}(\widehat{\boldsymbol{\alpha}}^\lambda)\boldsymbol{y}$ are

$$\{0, 0.5, 0.6, 0.7, 0.11, 0.12, 0.13, 0.18\}\,.$$

Therefore, if, for example, the jth component of the vector $\lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)}\,\text{diag}(\widehat{\boldsymbol{\alpha}}^{\lambda})\boldsymbol{y}$ is equal to 0.12, the only possible value for the jth row of $\boldsymbol{X}^{(k)}$ is $[1,\,0,\,1]^{\top}$. The CC can then repeat this process to reconstruct the entire matrix $\boldsymbol{X}^{(k)}$.

These confidentiality issues are numerically illustrated using R code available at https://github.com/OpenLHS/Revisiting_VERTIGO.

3.2 Results pertaining to objective 2: Communication

The dual objective function used in VERTIGO, and its associated gradient and Hessian with respect to $\boldsymbol{\alpha}$, only depends on the data through $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{y}$. It follows that optimization procedure used to calculating $\widehat{\boldsymbol{\alpha}}^{\lambda}$ can be entirely executed at the CC with the knowledge of $\boldsymbol{\mathcal{K}}$ and $\boldsymbol{y}$. To illustrate, we develop the objective function, the gradient and the Hessian in a way that highlights that only $\boldsymbol{y}$ and $\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}$ for all $k \in \{1,\ldots,K\}$ are needed for the CC to complete the estimation procedure.

First, since $\boldsymbol{\mathcal{K}} = (\sum_{k=1}^{K}\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}) + \mathbf{1}_n\mathbf{1}_n^{\top}$ (see (5)), it directly follows that

$$\begin{aligned} J^{\lambda}(\boldsymbol{\alpha}) =& \frac{1}{2\lambda}\boldsymbol{\alpha}^{\top}\,\text{diag}(\boldsymbol{y})\Big\{\sum_{k=1}^{K}\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}\Big\}\,\text{diag}(\boldsymbol{y})\boldsymbol{\alpha} \\ &+ \frac{1}{2\lambda}\boldsymbol{\alpha}^{\top}\,\text{diag}(\boldsymbol{y})\mathbf{1}_n\mathbf{1}_n^{\top}\,\text{diag}(\boldsymbol{y})\boldsymbol{\alpha} + \sum_{i=1}^{n}L(\alpha_i)\,. \end{aligned} \tag{19}$$

Consequently, if each node k sends the matrix $\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}$ to the CC, and assuming the CC has access to the response vector $\boldsymbol{y}$, the CC can fully reconstruct the objective function of the optimization problem in (9). As will be shown, the CC is also able to solve this problem and compute $\widehat{\boldsymbol{\alpha}}^{\lambda}$ using Newton iterations without requiring any additional communication with the nodes.

From (10), since

$$\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}) = \begin{cases} \lambda^{-1}\,\mathrm{diag}(\boldsymbol{y})\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top\,\mathrm{diag}(\boldsymbol{y})\boldsymbol{\alpha} & \text{for } k \leq K-1 \\ \lambda^{-1}\,\mathrm{diag}(\boldsymbol{y})\{\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top + \mathbf{1}_n\mathbf{1}_n^\top\}\,\mathrm{diag}(\boldsymbol{y})\boldsymbol{\alpha} & \text{for } k = K\,, \end{cases}$$

the gradient of $J^\lambda(\boldsymbol{\alpha})$ can be expressed as

$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} J^\lambda(\boldsymbol{\alpha}) =& \frac{1}{\lambda}\,\mathrm{diag}(\boldsymbol{y})\Big\{\sum_{k=1}^{K}\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top\Big\}\,\mathrm{diag}(\boldsymbol{y})\boldsymbol{\alpha} \\ &+ \frac{1}{\lambda}\,\mathrm{diag}(\boldsymbol{y})\mathbf{1}_n\mathbf{1}_n^\top\,\mathrm{diag}(\boldsymbol{y})\boldsymbol{\alpha} \\ &+ \left[\log\left(\frac{\alpha_1}{1-\alpha_1}\right), \cdots, \log\left(\frac{\alpha_n}{1-\alpha_n}\right)\right]^\top . \end{aligned} \tag{20}$$

This expression for the gradient can also be derived directly from (19) using standard properties of vector calculus.

Moreover, from (11), the Hessian matrix of $J^\lambda(\boldsymbol{\alpha})$ expresses as

$$\begin{aligned} \boldsymbol{H}(\boldsymbol{\alpha}) =& \frac{1}{\lambda}\,\mathrm{diag}(\boldsymbol{y})\Big\{\sum_{k=1}^{K}\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top\Big\}\,\mathrm{diag}(\boldsymbol{y}) + \frac{1}{\lambda}\,\mathrm{diag}(\boldsymbol{y})\mathbf{1}_n\mathbf{1}_n^\top\,\mathrm{diag}(\boldsymbol{y}) \\ &+ \mathrm{diag}\left\{\left[\frac{1}{\alpha_1(1-\alpha_1)}, \cdots, \frac{1}{\alpha_n(1-\alpha_n)}\right]^\top\right\}. \end{aligned} \tag{21}$$

The expressions for $\nabla_{\boldsymbol{\alpha}} J^\lambda(\boldsymbol{\alpha})$ and $\boldsymbol{H}(\boldsymbol{\alpha})$ make it clear that, once the CC has received the matrices $\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^\top$, it is able to compute the Newton update without any additional communication with the nodes.

The version of the method with optimized communication is presented in Algorithm 1. The equivalence between the two algorithms follows from the fact that their Newton updates are mathematically identical. The information exchanged during the communication steps

in the optimized version of the algorithm is summarized in Table 2.

Algorithm 1 Optimized communications

Input Local data in each node $\boldsymbol{X}^{(k)}$, $k = 1, ..., K$, shared response $\boldsymbol{y}$, parameter λ, initial solution $\boldsymbol{\alpha}_{(s)}, s = 0$

Output Global solution $\widehat{\boldsymbol{\alpha}}^{\lambda}$

Procedure:

1. Node k: compute the local matrix $\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}$ and send $\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}$ to CC.
2. Do:
   (a) CC: compute the gradient $\nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha}_{(s)})$ using equation (20).
   (b) CC: compute the Hessian matrix $\boldsymbol{H}(\boldsymbol{\alpha}_{(s)})$ using equation (21).
   (c) CC: update $\boldsymbol{\alpha}_{(s)} \leftarrow \boldsymbol{\alpha}_{(s)} - \boldsymbol{H}^{-1}(\boldsymbol{\alpha}_{(s)})\nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha}_{(s)})$.
   $s \leftarrow s + 1$, until $\boldsymbol{\alpha}_{(s)}$ converges.
3. $\widehat{\boldsymbol{\alpha}}^{\lambda} \leftarrow \boldsymbol{\alpha}_{(s)}$.

Table 2 Overview of the information shared in the communication optimized algorithm

Step	From nodes to CC	From CC to nodes
Initialization[1]	$\boldsymbol{\mathcal{K}}^{(k)}$	-
Last iteration	-	$\widehat{\boldsymbol{\alpha}}^{\lambda}$
Estimation results	$\widehat{\boldsymbol{\beta}}^{\lambda(k)}$	-

[1] It is assumed that all parties (nodes and CC) have access to $\boldsymbol{y}$ and λ.

3.3 Results pertaining to objective 3: Comparability

The alternative approach proposed, which excludes the intercept from the penalty, is outlined in Algorithm 2. Once each node has sent the matrix $\boldsymbol{X}^{(k)}(\boldsymbol{X}^{(k)})^{\top}$ to the CC, the constrained dual problem in (15) is solved at the CC level using constrained optimization techniques that require no further communication or information exchange between the nodes and the CC. Multiple constrained convex optimization algorithms exist and some are implemented in software packages, such as the CVXR package in R [17]. We refer to [18] for further information regarding constrained optimization. After computing $\widehat{\boldsymbol{\alpha}}^{\lambda}_{\text{alt}}$, the CC broadcasts $\widehat{\boldsymbol{\alpha}}^{\lambda}_{\text{alt}}$ to all nodes. From this, to obtain the

covariate-associated coefficients, the nodes use equation (14) with their locally stored data such that

$$\widehat{\boldsymbol{\beta}}_{\text{alt}}^{\lambda(k)} = \lambda^{-1}(\boldsymbol{X}^{(k)})^\top \operatorname{diag}(\boldsymbol{y})\widehat{\boldsymbol{\alpha}}_{\text{alt}}^{\lambda}\,. \tag{22}$$

Using (14) and (16), $\widehat{\beta}_{0,\text{alt}}^{\lambda}$ can be computed by combining $\boldsymbol{X}\boldsymbol{X}^\top$, $\boldsymbol{y}$ and $\widehat{\boldsymbol{\alpha}}_{\text{alt}}^{\lambda}$. Since, at this stage, these quantities are all available at the CC, the CC is able to compute $\widehat{\beta}_{0,\text{alt}}^{\lambda}$.

Algorithm 2 RidgeLog-V: Ridge Logistic Regression from Vertically partitioned data with no penalty on the intercept

Input Local data in each node $\boldsymbol{X}^{(k)}$, $k = 1, ..., K$, shared response $\boldsymbol{y}$, parameter λ, initial solution $\boldsymbol{\alpha}_{(s)}, s = 0$

Output Global solution $\widehat{\boldsymbol{\alpha}}^{\lambda}$

Procedure:

1. Node k: compute the local matrix $\boldsymbol{X}^{(k)}\boldsymbol{X}^{(k)\top}$ and send $\boldsymbol{X}^{(k)}\boldsymbol{X}^{(k)\top}$ to CC.
2. CC: compute the global matrix $\boldsymbol{X}\boldsymbol{X}^\top = \sum_{k=1}^{K} \boldsymbol{X}^{(k)}\boldsymbol{X}^{(k)\top}$.
3. CC: solve problem (15) using an algorithm adapted for equality constrained optimization until convergence is attained at a given iteration s.
4. $\widehat{\boldsymbol{\alpha}}_{\text{alt}}^{\lambda} \leftarrow \boldsymbol{\alpha}_{(s)}$.

The R implementation of RidgeLog-V (covering objectives (2) and (3)) along with an example comparing its results to those obtained in a centralized setting using the glmnet R package [13] and to those produced by the VERTIGO algorithm, is available at https://github.com/OpenLHS/Revisiting_VERTIGO.

4 Discussion

To conduct logistic regression analyses in the context of vertically partitioned data, the algorithms VERTIGO and VERTIGO-CI were identified and evaluated with respect to confidentiality, communication efficiency, and comparability to pooled methods.

Unlike in fields where data partitioning may result from limitations in computing resources, the distributed nature of data in many health studies is often due to ethical and legal restrictions that prohibit sharing raw data. In this context, one might

mistakenly assume that any method avoiding the direct transmission of raw data is *de facto* compliant with these constraints. However, such constraints rather impose that raw data must remain unknowable to any unauthorized party, regardless of the form in which information is exchanged with the original data source. VERTIGO-CI fails to meet this requirement, as it allows an unauthorized party (the CC) to reconstruct raw data, without the need for intensive computing resources, even though individual data was not directly transmitted (notwithstanding the outocome values). VERTIGO-CI thus serves as another salient example that a distributed algorithm is not inherently privacy-preserving.

While retrieving individual data in VERTIGO-CI required only simple vector-matrix operations involving quantities already available at the CC level, highlighting a flaw in the algorithm's design, the potential confidentiality breach that arises in all discussed algorithms when a participating node possesses only binary covariates is more subtle. In this case, the binary nature of the features allows for the construction of a system of equations based on the shared quantities. The structure of this system of equations can lead to a unique solution, thereby enabling full reconstruction of the individual data. Our paper identifies conditions under which such recovery is possible. Binary covariates are quite common in this type of analysis. Situations involving a mix of binary and continuous predictors, however, will require users to conduct careful assessments before deploying the method. On the other hand, we did not identify any scenarios in which data nodes with more than three continuous covariates were at risk of reverse-engineering.

This highlights the importance of thoroughly investigating potential data leakage risks when developing methods for distributed analysis. Proving confidentiality is challenging, and often the next best approach is to make as many informed attempts as possible to challenge the potential for reverse-engineering. We found that exploring various forms of representation, such as the matrix form used throughout our approach,

can help uncover potential vulnerabilities. Overall, a clear and systematic framework for assessing re-identification risk based on shared quantities remains lacking in the current literature.

What is clear, however, is that the privacy-preserving capabilities of a method cannot be evaluated solely based on the confidentiality properties of individual algorithmic operations. Instead, they must be assessed in light of all quantities that may become available to a potentially adversarial party throughout the entire procedure. For example, the VERTIGO paper claims that when a sufficient number of covariates are present, "the [CC] can obtain the global Gram matrix [...] from each party and avoid the disclosure of patient data", noting that since the Gram matrix is constructed from dot products between patient records, "it is not possible to reverse-engineer the product to obtain the values as long as there are enough covariates from individual databases". While this may hold under typical data distributions (though it can fail in extreme scenarios, such as when covariates are bounded and all entries lie on boundary values) our findings show that this protection breaks down even under typical conditions when parameter estimates are disclosed and one of the data nodes holds only binary covariates. Such claims may be misleading with respect to the perceived safety of the procedure, especially in the absence of a thorough privacy analysis.

We also identified unnecessary communication steps in VERTIGO and have optimized the algorithm's communication efficiency. While such changes may appear minor, they can significantly impact the practical applicability of the method to health data. Due to the sensitivity of such data, many institutions require manual review of intermediate results before any further communication with other nodes or the CC. As a result, each additional communication step substantially affects the feasibility of the approach and increases the resources needed to implement it.

This work also underscores the importance of understanding the impact of modeling choices introduced by adapting a method for use in a distributed environment—in

this case, those related to penalizing the intercept—to ensure that the resulting estimates remain interpretable in the same way as those typically obtained from similar models in a non-distributed setting. We proposed the alternative RidgeLog-V approach that excludes the intercept from the penalty during estimation, thereby avoiding bias in the average predicted probability of the outcome being one toward one-half.

While the current work has identified important limitations to the context in which the method can be used, it does not mean the method should be entirely discarded. We have not identified confidentiality risks when binary features are included but their associated estimates are withheld. We have also not identified any issue in settings involving enough continuous covariates. In fact, penalized logistic regression is particularly well-suited to scenarios with continuous covariates, especially when these variables are correlated, as the penalty helps mitigate multicollinearity and supports more stable estimation of their relationship with a binary outcome [12]. While the notion of enough covariates was also mentioned in the original VERTIGO paper, no quantification was given. By simple observation of the Gram matrix, cases where a node holds only one or two variables can be problematic, but such issues do not seem to arise with at least three continuous variables. However, in the context of analyses that require privacy guarantees, further investigations should be led before sharing estimates associated with continuous covariates when any of the proposed algorithms are employed.

5 Conclusion

We identified confidentiality issues with VERTIGO-CI, concluding that this extension should not be used in settings involving confidential feature data. We also demonstrated the risk of individual data leakage when parameter estimates associated with binary features are shared following the execution of VERTIGO. For use with continuous feature data, we proposed an optimized version of the algorithm to reduce

communication costs and introduced an alternative formulation, the RidgeLog-V algorithm, that excludes the intercept from the penalty term. This modification is consistent with standard applications of penalized logistic regression and facilitates a more meaningful interpretation of the intercept.

List of abbreviations.

- CC: Coordinating center
- HDRN: Health Data Research Network
- CI: Confidence interval

Supplementary information. See Additional file 1.

Declarations

- **Ethics approval and consent to participate**

  Not applicable

- **Consent for publication**

  Not applicable

- **Availability of data and materials**

  The code used in this article is available at https://github.com/OpenLHS/Revisiting_VERTIGO.

- **Competing interests**

  The authors declare that they have no competing interests.

- **Funding**

  This work was supported by the Natural Sciences and Engineering Research Council of Canada – Discovery Grant; the Health Data Research Network Canada, an initiative funded by the Canadian Institutes of Health Research; and the Chaire

de recherche en informatique de la santé de l'Université de Sherbrooke. Marie-Pier Domingue and Simon Lévesque both received a scholarship from the Natural Sciences and Engineering Research Council of Canada.

- **Author contribution**

  M.P.D., J.F.E., A.B., and F.C.L. conceived and designed the study. M.P.D., J.F.E., and F.C.L. wrote the manuscript. The mathematical derivations were conducted by M.P.D. and F.C.L., with contributions from S.L. and J.P.M. Implementations were carried out by M.P.D., J.F.E., and J.P.M. All authors read and approved the final manuscript.

- **Acknowledgments**

  Not applicable

References

[1] CanPath.: The Canadian Partnership for Tomorrow's Health. Available from: https://canpath.ca/en/.

[2] Pfitzner B, Steckhan N, Arnrich B. Federated Learning in a Medical Context: A Systematic Literature Review. ACM Transactions on Internet Technology. 2021;21(2):1–31. https://doi.org/10.1145/3412357.

[3] Brink C, Hansen CR, Field M, Price G, Thwaites D, Sarup N, et al.: Distributed learning optimisation of Cox models can leak patient data: Risks and solutions. Available from: https://arxiv.org/abs/2204.05856.

[4] Agresti A. Foundations of linear and generalized linear models. John Wiley & Sons; 2015.

[5] Slavkovic AB, Nardi Y, Tibbits MM. "Secure" Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases. In: Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007). Omaha, NE, USA: IEEE; 2007. p. 723–728.

[6] Fienberg SE, Nardi Y, Slavković AB. Valid Statistical Analysis for Logistic Regression with Multiple Sources. In: Protecting Persons While Protecting the People. Berlin, Heidelberg: Springer; 2009. p. 82–94.

[7] Gonçalves C, Bessa RJ, Pinson P. A critical overview of privacy-preserving approaches for collaborative forecasting. International Journal of Forecasting. 2021;37(1):322–342. https://doi.org/10.1016/j.ijforecast.2020.06.003.

[8] Li Y, Jiang X, Wang S, Xiong H, Ohno-Machado L. VERTIcal Grid lOgistic regression (VERTIGO). Journal of the American Medical Informatics Association. 2016;23(3):570–579. https://doi.org/10.1093/jamia/ocv146.

[9] Kim J, Li W, Bath T, Jiang X, Ohno-Machado L. VERTIcal Grid lOgistic regression with Confidence Intervals (VERTIGO-CI). AMIA Summits on Translational Science Proceedings. 2021;p. 355–363.

[10] Moncada-Torres A, Martin F, Sieswerda M, Van Soest J, Geleijnse G. VANTAGE6: an open source priVAcy preserviNg federaTed leArninG infrastructurE for Secure Insight eXchange. AMIA Annual Symposium Proceedings. 2021;2020:870–877.

[11] Kuo TT, Gabriel RA, Koola J, Schooley RT, Ohno-Machado L. Distributed cross-learning for equitable federated models - privacy-preserving prediction on data from five California hospitals. Nature Communications. 2025;16(1). https://doi.org/10.1038/s41467-025-56510-9.

[12] Šinkovec H, Heinze G, Blagus R, Geroldinger A. To tune or not to tune, a case study of ridge logistic regression in small or sparse datasets. BMC Medical Research Methodology. 2021;21(199). https://doi.org/10.1186/s12874-021-01374-y.

[13] Friedman J, Tibshirani R, Hastie T. Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software. 2010;33(1):1–22. https://doi.org/10.18637/jss.v033.i01.

[14] R Core Team.: R: A Language and Environment for Statistical Computing. Vienna, Austria. Available from: https://www.R-project.org/.

[15] Camirand Lemyre F, Lévesque S, Domingue MP, Herrmann K, Ethier JF. Distributed Statistical Analyses: A Scoping Review and Examples of Operational Frameworks Adapted to Health Analytics. JMIR Medical Informatics. 2024;12(1). https://doi.org/10.2196/53622.

[16] Le Cessie S, Van Houwelingen JC. Ridge Estimators in Logistic Regression. Journal of the Royal Statistical Society Series C (Applied Statistics). 1992;41(1):191–201. https://doi.org/10.2307/2347628.

[17] Fu A, Narasimhan B, Boyd S. CVXR: An R Package for Disciplined Convex Optimization. Journal of Statistical Software. 2020;94(14):1–34. https://doi.org/10.18637/jss.v094.i14.

[18] Boyd SP, Vandenberghe L. Convex optimization. Cambridge, United Kingdom: Cambridge university press; 2004.

Additional file 1 - *Revisiting VERTIGO and VERTIGO-CI: Identifying confidentiality breaches and introducing a statistically sound, efficient alternative*

Appendix A Notation

Table A1 Glossary for general notation conventions

Random variable in $\mathbb{R}$	A	Uppercase Non-italic
Random vector in $\mathbb{R}^p$	$\mathbf{A}$	Uppercase Non-italic bold
Scalar in $\mathbb{R}$	a	Lowercase Italic
Vector in $\mathbb{R}^p$	$\boldsymbol{a}$	Lowercase Italic bold
Vector in $\mathbb{R}^p$ with all components equal to 1	$\mathbf{1}_p$	-
Matrix in $\mathbb{R}^{n\times p}$	$\boldsymbol{A}$	Uppercase Italic bold
Identity matrix in $\mathbb{R}^{n\times n}$	$\boldsymbol{I}_n$	-
Gradient of $f(\boldsymbol{\theta})$ (column vector)	$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$	$\boldsymbol{H}$ for Hessian
$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})\vert_{\boldsymbol{\theta}=\boldsymbol{a}}$	$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{a})$	$\boldsymbol{H}(\boldsymbol{a})$ for Hessian
$\max_{1\leq j\leq p} \vert a_j\vert$	$\vert\vert\boldsymbol{a}\vert\vert_\infty$	Infinite norm
$\sum_{j=1}^p \vert a_j\vert$	$\vert\vert\boldsymbol{a}\vert\vert_1$	$\ell 1$-norm
$\sqrt{\sum_{j=1}^p a_i^2}$	$\vert\vert\boldsymbol{a}\vert\vert_2$	$\ell 2$-norm
Diagonal matrix with entries of $\boldsymbol{a}$ on diagonal	$\mathrm{diag}(\boldsymbol{a})$	Dimension $p\times p$ for $\boldsymbol{a}\in\mathbb{R}^p$
Quantity $\cdot$ at iteration t (step count)	$\cdot_{(t)}$	Starts with $\cdot_{(0)}$

Table A2 Glossary for quantities that pertain to the regression settings

Analytical dataset with sample size n	$\mathcal{D}=\{\ldots\}_{i=1}^n$
Covariate vector for *ith* individual	$\boldsymbol{x}_i=[x_{i1},\ldots,x_{ip}]^\top$
Covariate matrix in $\mathbb{R}^{n\times p}$	$\boldsymbol{X}=\begin{bmatrix} x_{11} & \cdots & x_{1p}\\ \vdots & \ddots & \vdots\\ x_{n1} & \cdots & x_{np}\end{bmatrix}=\begin{bmatrix}\boldsymbol{x}_1^\top\\ \vdots\\ \boldsymbol{x}_n^\top\end{bmatrix}$
Gram matrix	$\boldsymbol{\mathcal{K}}=[\boldsymbol{X}\ \mathbf{1}_n]\,[\boldsymbol{X}\ \mathbf{1}_n]^\top$
True (unknown) parameters	$\beta_{0\star},\boldsymbol{\beta}_\star$
Penalized estimate of the parameter (penalty parameter λ)	$\widehat{\beta}_0^\lambda,\widehat{\boldsymbol{\beta}}^\lambda$
Penalized estimate of the parameter (intercept not penalized)	$\widehat{\beta}_{0,\mathrm{alt}}^\lambda,\widehat{\boldsymbol{\beta}}_{\mathrm{alt}}^\lambda$
Log-likelihood	$\ell_n(\beta_0,\boldsymbol{\beta})=\frac{1}{n}\sum_{i=1}^n\log(\cdot)$
Penalized log-likelihood	$l_n^\lambda(\beta_0,\boldsymbol{\beta})$
Penalized log-likelihood (intercept not penalized)	$l_{n,\mathrm{alt}}^\lambda(\beta_0,\boldsymbol{\beta})$

Table A3 Glossary for quantities specific to the vertical setting

Number of covariate-nodes	K
Number of covariates at covariate-node k	$p^{(k)}$
Covariate matrix at covariate-node k	$\boldsymbol{X}^{(k)}$
Dual parameter estimates	$\widehat{\boldsymbol{\alpha}}^\lambda$
Penalized estimate associated with covariate-node k	$\widehat{\boldsymbol{\beta}}^{\lambda(k)}$

Appendix B VERTIGO and VERTIGO-CI Algorithms

Algorithm 3 provides the original VERTIGO algorithm [1], while algorithm 4 provides the original VERTIGO-CI algorithm [2]. Some adjustments have been made to the representation of the original algorithms using the harmonized notation to facilitate the comparison. We note that constrained optimization technique should be considered when numerically applying the Newton-Raphson steps because the components of the parameter $\boldsymbol{\alpha}$ are constrained in $(0, 1)$.

Algorithm 3 VERTIGO Original [1]

Input Local data in each node $\boldsymbol{X}^{(k)}$, $k = 1, ..., K$, shared response $\boldsymbol{y}$, parameter λ, initial solution $\boldsymbol{\alpha}_{(s)}$, $s = 0$

Output Global solution $\widehat{\boldsymbol{\alpha}}^{\lambda}$

Procedure:

1. Node k: compute the local matrix $\boldsymbol{\mathcal{K}}^{(k)}$ and send $\boldsymbol{\mathcal{K}}^{(k)}$ to CC.
2. CC: compute the global matrix $\boldsymbol{\mathcal{K}}$.
3. Do:
   (a) Node k: compute $\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$, and send $\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$ to CC.
   (b) CC: calculate $\boldsymbol{e}_{(s)} = \sum_{k=1}^{K} \boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$ and $\nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha}_{(s)})$ (using Equation (10)).
   (c) CC: compute the Hessian matrix $\boldsymbol{H}(\boldsymbol{\alpha}_{(s)})$ (using Equation (11)).
   (d) CC: update $\boldsymbol{\alpha}_{(s)} \leftarrow \boldsymbol{\alpha}_{(s)} - \boldsymbol{H}^{-1}(\boldsymbol{\alpha}_{(s)}) \nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha}_{(s)})$ and send update $\boldsymbol{\alpha}_{(s)}$ to each node.

   $s \leftarrow s + 1$, until $\boldsymbol{\alpha}_{(s)}$ converges.
4. $\widehat{\boldsymbol{\alpha}}^{\lambda} \leftarrow \boldsymbol{\alpha}_{(s)}$.

Algorithm 4 VERTIGO-CI Original [2]

Input Local data in each node $\boldsymbol{X}^{(k)}$, $k = 1, ..., K$, shared response $\boldsymbol{y}$, parameter λ, initial solution $\boldsymbol{\alpha}_{(s)}, s = 0$

Output Global solution $\widehat{\boldsymbol{\beta}}^{\lambda}$ and associated variances.

Procedure:

1. Node k: compute the local matrix $\boldsymbol{\mathcal{K}}^{(k)}$ and send $\boldsymbol{\mathcal{K}}^{(k)}$ to CC.
2. CC: compute the global matrix $\boldsymbol{\mathcal{K}}$.
3. Do:
   (a) Node k: compute $\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$, and send $\boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$ to CC.
   (b) CC: calculate $\boldsymbol{e}_{(s)} = \sum_{k=1}^{K} \boldsymbol{e}^{(k)}(\boldsymbol{\alpha}_{(s)})$ and $\nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha}_{(s)})$ (using Equation (10)).
   (c) CC: compute the modified Hessian matrix $\boldsymbol{H}(\boldsymbol{\alpha}_{(s)}) = \lambda^{-1} \operatorname{diag}(\boldsymbol{y}) \boldsymbol{\mathcal{K}} \operatorname{diag}(\boldsymbol{y}) + C\boldsymbol{I}_n$.
   (d) CC: update $\boldsymbol{\alpha}_{(s)} \leftarrow \boldsymbol{\alpha}_{(s)} - \boldsymbol{H}^{-1}(\boldsymbol{\alpha}_{(s)}) \nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha}_{(s)})$ and send update $\boldsymbol{\alpha}_{(s)}$ to each node.

   $s \leftarrow s + 1$, until $\boldsymbol{\alpha}_{(s)}$ converges.
4. CC: $\widehat{\boldsymbol{\alpha}}^{\lambda} \leftarrow \boldsymbol{\alpha}_{(s)}$.
5. CC: send $\widehat{\boldsymbol{\alpha}}^{\lambda}$ to each node.
6. Node k, $k \leq K-1$: Calculate $\widehat{\boldsymbol{\beta}}^{\lambda(k)} = \lambda^{-1} \sum_{i=1}^{n} \widehat{\alpha}_i^{\lambda} y_i \boldsymbol{x}_i^{(k)}$. and send $\widehat{\boldsymbol{\beta}}^{\lambda(k)}$ to CC.
7. Node K: Calculate $\widehat{\beta}_0^{\lambda} = \lambda^{-1} \sum_{i=1}^{n} \widehat{\alpha}_i^{\lambda} y_i$ and $\widehat{\boldsymbol{\beta}}^{\lambda(K)} = \lambda^{-1} \sum_{i=1}^{n} \widehat{\alpha}_i^{\lambda} y_i \boldsymbol{x}_i^{(K)}$, and send $\widehat{\beta}_0^{\lambda}$ and $\widehat{\boldsymbol{\beta}}^{\lambda(K)}$ to CC.
8. Client-to-client communication:
   (a) Node k, $k \leq K-1$: compute $\exp\{(\boldsymbol{x}_i^{(k)})^{\top} \widehat{\boldsymbol{\beta}}^{\lambda(k)}\}$ for all $i \in \{1, \ldots, n\}$ and send result to node 1.
   (b) Node K: compute and send $\exp\{\widehat{\beta}_0^{\lambda}\} \exp\{(\boldsymbol{x}_i^{(K)})^{\top} \widehat{\boldsymbol{\beta}}^{\lambda(K)}\}$ for all $i \in \{1, \ldots, n\}$ and send result to node 1.
   (c) Node 1: combine the results using $\exp\{\widehat{\beta}_0^{\lambda} + \boldsymbol{x}_i^{\top} \widehat{\boldsymbol{\beta}}^{\lambda}\} = \exp\{\widehat{\beta}_0^{\lambda}\} \prod_{k=1}^{K} \exp\{(\boldsymbol{x}_i^{(k)})^{\top} \widehat{\boldsymbol{\beta}}^{\lambda(k)}\}$ for all $i \in \{1, \ldots, n\}$, compute $\boldsymbol{V}^{\lambda}$ (using Equation (12)) and send V to all nodes k, $k \in \{2, \ldots, K\}$.
9. Node k, $k \leq K-1$: calculate $(\boldsymbol{X}^{(k)})^{\top} (\boldsymbol{V}^{\lambda})^{1/2}$ and send this quantity to CC.
10. Node K: calculate $[\boldsymbol{X}^{(K)} \, \boldsymbol{1}_n]^{\top} (\boldsymbol{V}^{\lambda})^{1/2}$ and send this quantity to CC.
11. CC: build $[\boldsymbol{X} \, \boldsymbol{1}_n]^{\top} \boldsymbol{V}^{\lambda} [\boldsymbol{X} \, \boldsymbol{1}_n]$ by block (see Algorithm 2 in [2] for details), and invert the result to obtain variance estimates.

Appendix C Equivalence between matrix and expanded notation in dual optimization

To obtain the expression of $J^\lambda(\boldsymbol{\alpha}) = \frac{1}{2\lambda}\boldsymbol{\alpha}^\top \operatorname{diag}(\boldsymbol{y})\boldsymbol{\mathcal{K}}\operatorname{diag}(\boldsymbol{y})\boldsymbol{\alpha} + \sum_{i=1}^n L(\alpha_i)$, it suffices to note that

$$
\begin{aligned}
\boldsymbol{\alpha}^\top \operatorname{diag}(\boldsymbol{y})\boldsymbol{\mathcal{K}}\operatorname{diag}(\boldsymbol{y})\boldsymbol{\alpha} &= \begin{bmatrix} \alpha_1 y_1 & \cdots & \alpha_n y_n \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1^\top \boldsymbol{x}_1 + 1 & \cdots & \boldsymbol{x}_1^\top \boldsymbol{x}_n + 1 \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_n^\top \boldsymbol{x}_{1+1} & \cdots & \boldsymbol{x}_n^\top \boldsymbol{x}_n + 1 \end{bmatrix} \begin{bmatrix} \alpha_1 y_1 \\ \vdots \\ \alpha_n y_n \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^n \alpha_i y_i (\boldsymbol{x}_i^\top \boldsymbol{x}_1 + 1) & \cdots & \sum_{i=1}^n \alpha_i y_i (\boldsymbol{x}_i^\top \boldsymbol{x}_n + 1) \end{bmatrix} \begin{bmatrix} \alpha_1 y_1 \\ \vdots \\ \alpha_n y_n \end{bmatrix} \\
&= \sum_{j=1}^n \sum_{i=1}^n \alpha_i y_i (\boldsymbol{x}_i^\top \boldsymbol{x}_j + 1) \alpha_j y_j \\
&= \sum_{j=1}^n \sum_{i=1}^n \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^\top \boldsymbol{x}_j + 1)\,.
\end{aligned}
$$

Appendix D Mathematical development for the dual problem of the ridge logistic regression maximum log-likelihood problem with no penalty over the intercept

Proposition 1 *Consider any $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, with $y_i \in \{-1, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^p$ for $i = 1, \ldots, n$ and $n \geq 2$. Assume that $\sum_{i=1}^n \mathbb{I}(y_i = 1) \in \{1, \ldots, n-1\}$ (i.e., there exists at least one i such that $y_i = 1$, and at least one j such that $y_j = -1$). Then, for any $\lambda > 0$,*

there exists a unique solution $(\widehat{\beta}^{\lambda}_{0,alt}, \widehat{\boldsymbol{\beta}}^{\lambda}_{alt})$ *to the optimization problem*

$$\max_{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^\top \in \mathbb{R}^p} l^{\lambda}_{n,alt}(\beta_0, \boldsymbol{\beta}) \,. \tag{D1}$$

Furthermore, $(\widehat{\beta}^{\lambda}_{0,alt}, \widehat{\boldsymbol{\beta}}^{\lambda}_{alt})$ *satisfies*

$$\widehat{\beta}^{\lambda}_{0,alt} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i \log\{(\widehat{\alpha}^{\lambda}_{i,alt})^{-1} - 1\} - \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{alt} \right) ,$$

$$\widehat{\boldsymbol{\beta}}^{\lambda}_{alt} = \lambda^{-1} \boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{y}) \widehat{\boldsymbol{\alpha}}^{\lambda}_{alt} \,,$$

where $\widehat{\boldsymbol{\alpha}}^{\lambda}_{alt} = [\widehat{\alpha}^{\lambda}_{1,alt}, \ldots, \widehat{\alpha}^{\lambda}_{n,alt}]^\top$ *is the unique solution to the following minimization problem:*

$$\min_{\boldsymbol{\alpha} \in (0,1)^n} J^{\lambda}_{alt}(\boldsymbol{\alpha}) \quad s.t. \sum_{i=1}^{n} y_i \alpha_i = 0.$$

Proof of Proposition 1. We begin by showing that the optimization problem in (D1) has a unique finite solution. To do this, we first compute that

$$\nabla_{\boldsymbol{\beta}} l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{y_i \boldsymbol{x}_i}{1 + \exp\{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\}} - \lambda \boldsymbol{\beta} \,,$$

and that

$$\frac{\partial}{\partial \beta_0} l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{y_i}{1 + \exp\{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\}} \,.$$

From this, we obtain that

$$\begin{aligned} &\nabla^2_{\boldsymbol{\beta}, \beta_0} l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} \frac{-\exp\{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\}}{(1 + \exp\{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})\})^2} \begin{bmatrix} \boldsymbol{x}_i \boldsymbol{x}_i^\top & \boldsymbol{x}_i \\ \boldsymbol{x}_i^\top & 1 \end{bmatrix} - \lambda \begin{bmatrix} \boldsymbol{I}_p & 0 \\ 0 & 0 \end{bmatrix} \,. \end{aligned}$$

Hence, $l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta})$ is strictly concave in $(\beta_0, \boldsymbol{\beta})$. Therefore, if $l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta})$ admits a maximizer, it must be unique. To conclude that the optimization problem in (D1) has a unique solution, it remains to show that $l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta})$ indeed has a maximizer.

To establish this, it suffices to show that $l^{\lambda}_{n,\text{alt}}(\beta_0, \boldsymbol{\beta}) \to -\infty$ as $\max\{|\beta_0|, \|\boldsymbol{\beta}\|_\infty\} \to \infty$, where we denote the infinite norm $||\boldsymbol{a}||_\infty := \max_{1\le j\le p} |a_j|$ for $a \in \mathbb{R}^p$. Observe that the log-likelihood term is bounded above by 0. Therefore, if $\|\boldsymbol{\beta}\|_\infty \to \infty$, then regardless of whether $|\beta_0| \to \infty$ or $|\beta_0| < \infty$, the penalty term $\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}$ dominates, and we have $l^{\lambda}_{n,\text{alt}}(\beta_0, \boldsymbol{\beta}) \to -\infty$.

Now suppose $\|\boldsymbol{\beta}\|_\infty \not\to \infty$ but $\max\{|\beta_0|, \|\boldsymbol{\beta}\|_\infty\} \to \infty$, which implies that $|\beta_0| \to \infty$ while $\|\boldsymbol{\beta}\|_\infty$ remains bounded. Because there is at least one $y_i = 1$ and at least one $y_j = -1$, there exists at least one term in the log-likelihood of the form $\log\left([(1 + \exp\{-y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta})\}]^{-1}\right)$ that diverges to $-\infty$ as $|\beta_0| \to \infty$ (since $\boldsymbol{x}_i^\top\boldsymbol{\beta}$ remains bounded), while the others remain bounded from above by 0. Therefore, $l^{\lambda}_{n,\text{alt}}(\beta_0, \boldsymbol{\beta}) \to -\infty$ in this case as well.

The conclusion of the above discussion is that for any $\lambda > 0$, there exists a unique solution $(\widehat{\beta}^{\lambda}_{0,\text{alt}}, \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}})$ to the optimization problem in (D1). This concludes the proof of the first part of the proposition. The rest of the proof is dedicated to show the second part of the proposition.

Since $l^{\lambda}_{n,\text{alt}}(\beta_0, \boldsymbol{\beta})$ is strictly concave and two times continuously differentiable, $(\widehat{\beta}^{\lambda}_{0,\text{alt}}, \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}})$ is a stationary point of $l^{\lambda}_{n,\text{alt}}(\beta_0, \boldsymbol{\beta})$, i.e.,

$$\begin{aligned}
&\nabla_{\boldsymbol{\beta}} l^{\lambda}_{n,\text{alt}}(\widehat{\beta}^{\lambda}_{0,\text{alt}}, \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}}) \\
&\qquad = \sum_{i=1}^{n} \frac{y_i \boldsymbol{x}_i \exp\{-y_i(\widehat{\beta}^{\lambda}_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}})\}}{1 + \exp\{-y_i(\widehat{\beta}^{\lambda}_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}})\}} - \lambda\boldsymbol{\beta} = 0\,,
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial}{\partial\beta_0} l^{\lambda}_{n,\text{alt}}(\widehat{\beta}^{\lambda}_{0,\text{alt}}, \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}}) &= \sum_{i=1}^{n} \frac{y_i \exp\{-y_i(\widehat{\beta}^{\lambda}_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}})\}}{1 + \exp\{-y_i(\widehat{\beta}^{\lambda}_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}})\}} \\
&= \sum_{i=1}^{n} \left\{\mathbb{I}(y_i = 1) - \frac{\exp\left(\widehat{\beta}^{\lambda}_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}}\right)}{1 + \exp\left(\widehat{\beta}^{\lambda}_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}}\right)}\right\} = 0\,, \qquad \text{(D2)}
\end{aligned}$$

where, on the last line, $\mathbb{I}(A)$ denotes the indicator function taking the value 1 if A is true, and 0 otherwise.

Using this result, we will next show that the search space in the optimization problem at (D1) can be narrowed to a compact set (see (D5) below).

From the first equation, we deduce that

$$\|\widehat{\boldsymbol{\beta}}^{\lambda}_{\text{alt}}\|_\infty \le \lambda^{-1} \sum_{i=1}^{n} \|\boldsymbol{x}_i\|_\infty\,. \qquad \text{(D3)}$$

Rearranging the equation for $\frac{\partial}{\partial \beta_0} l^{\lambda}_{n,\mathrm{alt}}(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}}, \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})$ gives

$$\begin{aligned}
&\sum_{i:y_i=1} \Big\{1 - \frac{\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}{1+\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}\Big\} \\
&\quad = \sum_{i:y_i=-1} \frac{\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}{1+\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)} \,. \qquad \text{(D4)}
\end{aligned}$$

From the latter equation, we will derive in turns a lower bound and an upper bound for $\widehat{\beta}^{\lambda}_{0,\mathrm{alt}}$. Starting with the upper bound, let $n_1 = \#\{i : y_i = 1\}$ and $n_{-1} = \#\{i : y_i = -1\}$. Since we have assumed $n_1 \geq 1$, we have

$$\begin{aligned}
&\sum_{i:y_i=1} \Big\{1 - \frac{\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}{1+\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}\Big\} \\
&\quad \geq 1 - \max_{i:y_i=1} \frac{\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}{1+\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)} \\
&\quad = 1 - \max_{i:y_i=1} \frac{1}{1+\exp\{-(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})\}} \\
&\quad \geq 1 - \frac{1}{1+\exp\{-\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} - \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_1\}} \,,
\end{aligned}$$

where $||\boldsymbol{a}||_1 := \sum_{j=1}^p |a_j|$ for $a \in \mathbb{R}^p$. On the other hand, the term on the second line of (D4) satisfies

$$\begin{aligned}
&\sum_{i:y_i=-1} \frac{\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)}{1+\exp\big(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\big)} \\
&\qquad \leq \frac{n_{-1}}{1+\exp\{-\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} - \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_1\}} \,.
\end{aligned}$$

Consequently,

$$\begin{aligned}
&1 - \frac{1}{1+\exp\{-\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} - \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_1\}} \\
&\qquad \leq \frac{n_{-1}}{1+\exp\{-\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} - \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_1\}} \,,
\end{aligned}$$

which implies that

$$\begin{aligned}
\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} &\geq -\log(n_{-1}) - \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_1 \\
&\geq -\log(n) - p\|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty \,,
\end{aligned}$$

where, to obtain the last line, we used the fact that $n_{-1} \leq n$, that for any $\boldsymbol{a} \in \mathbb{R}^p$, $\|\boldsymbol{a}\|_1 \leq p\|\boldsymbol{a}\|_\infty$, and that $\max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_\infty \leq \sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty$.

Since from similar arguments we can show that $\widehat{\beta}^\lambda_{0,\mathrm{alt}} \leq \log(n) + p\max_{1\leq i\leq n} \|\boldsymbol{x}_i\|_\infty \|\widehat{\boldsymbol{\beta}}^\lambda_{\mathrm{alt}}\|_\infty$, we conclude using (D3) that

$$|\widehat{\beta}^\lambda_{0,\mathrm{alt}}| \leq \log(n) + \lambda^{-1}p\big\{\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\big\}^2 .$$

To summarize, we have just shown that the solution $(\widehat{\beta}^\lambda_{0,\mathrm{alt}}, \widehat{\boldsymbol{\beta}}^\lambda_{\mathrm{alt}})$ of (D1) lies in the compact set $\Theta_{\boldsymbol{x},\lambda} = \Theta^0_{\boldsymbol{x},\lambda} \times (\Theta^1_{\boldsymbol{x},\lambda})^p$, with

$$\Theta^0_{\boldsymbol{x},\lambda} = \{\beta \in \mathbb{R} : |\beta| \leq \log(n) + \lambda^{-1}p\big\{\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\big\}^2\}$$

and

$$\Theta^1_{\boldsymbol{x},\lambda} = \{\beta \in \mathbb{R} : |\beta| \leq \lambda^{-1}\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\} .$$

Consequently, the search space $\mathbb{R} \times \mathbb{R}^p$ in the optimization problem at (D1) can be restricted to $\Theta_{\boldsymbol{x},\lambda}$, and the optimization problem at (D1) is equivalent to

$$\max_{\beta_0 \in \Theta^0_{\boldsymbol{x},\lambda}, \boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p} l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) . \tag{D5}$$

Now from the relationship

$$\begin{aligned}\min_{\alpha\in(0,1)} \alpha x + (1-\alpha)\log(1-\alpha) + \alpha\log(\alpha) \\ = \log\Big(\frac{1}{1+e^{-x}}\Big) ,\end{aligned}$$

since for any $\beta_0 \in \Theta^0_{\boldsymbol{x},\lambda}$ and $\boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p$, we have

$$\begin{aligned}|\beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x}_j| &\leq \log(n) + \lambda^{-1}p\big\{\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\big\}^2 \\ &\qquad + \lambda^{-1}p\big\{\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\big\} \max_{1\leq j\leq n} \|\boldsymbol{x}_j\|_\infty \\ &\leq \log(n) + 2\lambda^{-1}p\big\{\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\big\}^2 \\ &< \log(n) + 2 + 2\lambda^{-1}p\big\{\sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty\big\}^2 := c_{n,\boldsymbol{x},\lambda} ,\end{aligned}$$

where we used the fact that $|\boldsymbol{\beta}^\top \boldsymbol{x}_j| \le p\|\boldsymbol{\beta}\|_\infty \|\boldsymbol{x}_j\|_\infty \le p\|\boldsymbol{\beta}\|_\infty \sum_{i=1}^n \|\boldsymbol{x}_i\|_\infty$, we conclude that

$$
\begin{aligned}
& l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) = \min_{\boldsymbol{\alpha} \in (\Theta^2_{\boldsymbol{x},\lambda})^n} l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) \\
& \text{with } l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^n \Big\{ \alpha_i y_i (\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) + L(\alpha_i) \Big\} - \frac{\lambda}{2} \boldsymbol{\beta}^\top \boldsymbol{\beta} \\
& \text{and } \Theta^2_{\boldsymbol{x},\lambda} = \Big[\frac{1}{1+\exp(c_{n,\boldsymbol{x},\lambda})}, \frac{1}{1+\exp(-c_{n,\boldsymbol{x},\lambda})} \Big] . \qquad \text{(D6)}
\end{aligned}
$$

We recall $L(\alpha_i) = (1-\alpha_i)\log(1-\alpha_i) + \alpha_i \log(\alpha_i)$. Since the solution $\boldsymbol{\alpha}(\beta_0, \boldsymbol{\beta})$ of the optimization problem $\min_{\boldsymbol{\alpha} \in (\Theta^2_{\boldsymbol{x},\lambda})^n} l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})$ satisfies

$$
\alpha_i(\beta_0, \boldsymbol{\beta}) = \frac{\exp(-y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}))}{1+\exp(-y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}))} \quad \text{for all } i \in \{1, \dots, n\},
$$

and as the optimal $(\widehat{\beta}^\lambda_{0,\mathrm{alt}}, \widehat{\boldsymbol{\beta}}^\lambda_{\mathrm{alt}})$ are such that

$$
\begin{aligned}
0 &= \sum_{i=1}^n y_i \frac{\exp(-y_i(\widehat{\beta}^\lambda_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^\lambda_{\mathrm{alt}}))}{1+\exp(-y_i(\widehat{\beta}^\lambda_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^\lambda_{\mathrm{alt}}))} \\
&= \sum_{i=1}^n y_i \alpha_i(\widehat{\beta}^\lambda_{0,\mathrm{alt}}, \widehat{\boldsymbol{\beta}}^\lambda_{\mathrm{alt}}) ,
\end{aligned}
$$

(see (D2)), then, the search space $(\Theta^2_{\boldsymbol{x},\lambda})^n$ can be further narrowed to

$$
\Theta^3_{\boldsymbol{x},\lambda} := (\Theta^2_{\boldsymbol{x},\lambda})^n \cap \{\boldsymbol{\alpha} : \sum_{i=1}^n \alpha_i y_i = 0\} . \qquad \text{(D7)}
$$

Since for any $\boldsymbol{\alpha} \in \Theta^3_{\boldsymbol{x},\lambda}$ the function $l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})$ satisfies

$$
\begin{aligned}
l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) &:= l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta}, \boldsymbol{\alpha}) \\
&= \sum_{i=1}^n \Big\{ \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{\beta} + L(\alpha_i) \Big\} - \frac{\lambda}{2} \boldsymbol{\beta}^\top \boldsymbol{\beta} ,
\end{aligned}
$$

the optimization problem in (D5) can be expressed as

$$
\max_{\beta_0 \in \Theta^0_{\boldsymbol{x},\lambda}, \boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p} l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) = \max_{\boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p} \Big(\min_{\boldsymbol{\alpha} \in \Theta^3_{\boldsymbol{x},\lambda}} l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta}, \boldsymbol{\alpha}) \Big) . \qquad \text{(D8)}
$$

Now for any fixed $\boldsymbol{\beta}$ the function $\boldsymbol{\alpha} \mapsto l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta}, \boldsymbol{\alpha})$ is convex, and for any fixed $\boldsymbol{\alpha}$ the function $\boldsymbol{\beta} \mapsto l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta}, \boldsymbol{\alpha})$ is concave. Since $\Theta^1_{\boldsymbol{x},\lambda}$ and $\Theta^3_{\boldsymbol{x},\lambda}$ are compact convex sets, we can apply Sion's

minimax theorem [3] and swap the max and the min in the above equation and conclude that

$$\max_{\boldsymbol{\beta}\in(\Theta^1_{\boldsymbol{x},\lambda})^p}\left(\min_{\boldsymbol{\alpha}\in\Theta^3_{\boldsymbol{x},\lambda}} l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta},\boldsymbol{\alpha})\right) = \min_{\boldsymbol{\alpha}\in\Theta^3_{\boldsymbol{x},\lambda}}\left(\max_{\boldsymbol{\beta}\in(\Theta^1_{\boldsymbol{x},\lambda})^p} l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta},\boldsymbol{\alpha})\right)$$

The inner problem can be solved exactly, by equating to 0 the gradient of $l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta},\boldsymbol{\alpha})$ with respect to $\boldsymbol{\beta}$. The solution $\boldsymbol{\beta}(\boldsymbol{\alpha})$ of this equation is given by the following equation:

$$\boldsymbol{\beta}(\boldsymbol{\alpha}) = \lambda^{-1}\sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i\,.$$

Plugging the expression of $\boldsymbol{\beta}(\boldsymbol{\alpha})$ into the definition of $l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta},\boldsymbol{\alpha})$ yields $l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta}(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = J^\lambda_{\mathrm{alt}}(\boldsymbol{\alpha})$, and therefore

$$\begin{aligned}\max_{\boldsymbol{\beta}\in(\Theta^1_{\boldsymbol{x},\lambda})^p}\left(\min_{\boldsymbol{\alpha}\in\Theta^3_{\boldsymbol{x},\lambda}} l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta},\boldsymbol{\alpha})\right) &= \min_{\boldsymbol{\alpha}\in\Theta^3_{\boldsymbol{x},\lambda}}\left(J^\lambda_{\mathrm{alt}}(\boldsymbol{\alpha})\right)\\ &= \min_{\boldsymbol{\alpha}\in(\Theta^2_{\boldsymbol{x},\lambda})^n}\left(J^\lambda_{\mathrm{alt}}(\boldsymbol{\alpha})\right) \quad \text{s.t.} \sum_{i=1}^n y_i\alpha_i = 0\,.\end{aligned}$$

To obtain the last line, we used the definition of $\Theta^3_{\boldsymbol{x},\lambda}$ in (D7).

The optimization problem on the last line is strongly convex with a linear equality constraint. It therefore has a unique solution provided the feasible set $\{\boldsymbol{\alpha} \in (\Theta^2_{\boldsymbol{x},\lambda})^n : \sum_{i=1}^n \alpha_i y_i = 0\}$ is non empty. This is the case, since, as we have assumed $n_1 \geq 1$ and $n_{-1} \geq 1$ with $n \geq 2$, the point $\boldsymbol{\alpha}^f = (\alpha^f_1, \ldots, \alpha^f_n)^\top$ with $\alpha^f_i = 1/(2n_{-1})$ if $y_i = -1$ and $\alpha^f_i = 1/(2n_1)$ if $y_i = 1$ is feasible. To see why this is the case, recalling the definition of $\Theta^2_{\boldsymbol{x},\lambda}$ in (D6), it suffices to note that the inequality $1/(1+\exp\{x\}) \leq \exp(-x)$ implies that

$$\begin{aligned}\frac{1}{2\max(n_{-1},n_1)} &\geq (2n)^{-1} \geq \exp\{-\log(n)-\log(2)\}\\ &\geq \exp\{-\log(n)-2\} \geq \frac{1}{1+\exp\{\log(n)+2\}}\\ &\geq \frac{1}{1+\exp(c_{n,\boldsymbol{x},\lambda})}\,,\end{aligned}$$

and the inequality $1/2 \leq (1+\exp\{-2\})^{-1}$ implies that

$$\frac{1}{2\min(n_{-1},n_1)} \leq \frac{1}{2} \leq \frac{1}{1+\exp(-2)} \leq \frac{1}{1+\exp(-c_{n,\boldsymbol{x},\lambda})}\,.$$

Combining these relationships shows that $\alpha_i^f \in \Theta^2_{\boldsymbol{x},\lambda}$ for all $1 \leq i \leq n$. The proof of the proposition follows from the fact that, since the solution to the optimization problem

$$\min_{\boldsymbol{\alpha} \in (\Theta^2_{\boldsymbol{x},\lambda})^n} \left(J^\lambda_{\text{alt}}(\boldsymbol{\alpha}) \right) \quad \text{s.t.} \sum_{i=1}^n y_i \alpha_i = 0$$

is reached at a stationary point that cancels the Lagrangian, it can be equivalently expressed as

$$\min_{\boldsymbol{\alpha} \in (0,1)^n} \left(J^\lambda_{\text{alt}}(\boldsymbol{\alpha}) \right) \quad \text{s.t.} \sum_{i=1}^n y_i \alpha_i = 0\,.$$

Finally, the expression $\widehat{\beta}^\lambda_{0,\text{alt}} = \frac{1}{n} \sum_{i=1}^n \left(y_i \log\{(\widehat{\alpha}^\lambda_{i,\text{alt}})^{-1} - 1\} - \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}} \right)$ can directly be retrieved using the following:

$$\widehat{\alpha}^\lambda_{i,\text{alt}} = \frac{1}{1 + \exp(y_i(\widehat{\beta}^\lambda_{0,\text{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^\lambda_{\text{alt}}))} \quad \text{for all } i \in \{1, \ldots, n\}.$$

□

Appendix E Alternative reverse-engineering procedure

An alternative procedure can be employed to construct the diagonal matrix $\boldsymbol{V}^\lambda$ at the CC when executing VERTIGO-CI, which enables the CC to follow the steps outlined in the main text and subsequently reverse-engineer the feature data of all individuals. Recall the definition of the diagonal matrix $\boldsymbol{V}^\lambda$, with its entries specified in Equation (12). Also, recall the expression of $\nabla_{\boldsymbol{\alpha}} J^\lambda(\boldsymbol{\alpha})$ (Equation (10)), that $\widehat{\boldsymbol{\alpha}}^\lambda$ satisfies $\nabla_{\boldsymbol{\alpha}} J^\lambda(\widehat{\boldsymbol{\alpha}}^\lambda) = 0$, that $\widehat{\boldsymbol{\beta}}^{\lambda(k)} = \lambda^{-1} \sum_{i=1}^n \widehat{\alpha}^\lambda_i y_i \boldsymbol{x}_i^{(k)}$ and that $\widehat{\beta}^\lambda_0 = \lambda^{-1} \sum_{i=1}^n \widehat{\alpha}^\lambda_i y_i$.

Using these expression, we derive that the jth component of $\nabla_{\boldsymbol{\alpha}} J^{\lambda}(\boldsymbol{\alpha})$ satisfies

$$
\begin{aligned}
0 &= [\nabla_{\boldsymbol{\alpha}} J^{\lambda}(\widehat{\boldsymbol{\alpha}}^{\lambda})]_j \\
&= \sum_{k=1}^{K} e_j^{(k)}(\widehat{\boldsymbol{\alpha}}^{\lambda}) + \log\left(\frac{\widehat{\alpha}_j^{\lambda}}{1-\widehat{\alpha}_j^{\lambda}}\right) \\
&= \sum_{k=1}^{K}(\lambda^{-1}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i y_j (\boldsymbol{x}_i^{(k)})^{\top}\boldsymbol{x}_j^{(k)}) + \lambda^{-1}\sum_{i=1}^{n}\widehat{\alpha}_i^{\lambda} y_i y_j + \log\left(\frac{\widehat{\alpha}_j^{\lambda}}{1-\widehat{\alpha}_j^{\lambda}}\right) \\
&= y_j \sum_{k=1}^{K}((\boldsymbol{x}_j^{(k)})^{\top}\widehat{\boldsymbol{\beta}}^{\lambda(k)}) + y_j\widehat{\beta}_0^{\lambda} + \log\left(\frac{\widehat{\alpha}_j^{\lambda}}{1-\widehat{\alpha}_j^{\lambda}}\right) \\
&= y_j(\boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda} + \widehat{\beta}_0^{\lambda}) + \log\left(\frac{\widehat{\alpha}_j^{\lambda}}{1-\widehat{\alpha}_j^{\lambda}}\right).
\end{aligned}
$$

By isolating $\widehat{\alpha}_j^{\lambda}$, we obtain that

$$
\widehat{\alpha}_j^{\lambda} = \left[1+\exp\{y_j(\boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda} + \widehat{\beta}_0^{\lambda})\}\right]^{-1}.
$$

Since $1-\widehat{\alpha}_j^{\lambda} = [1+\exp\{-y_j(\boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda} + \widehat{\beta}_0^{\lambda})\}]^{-1}$, and as $y_j \in \{-1,1\}$, we deduce from the latter equation that

$$
\begin{aligned}
&\widehat{\alpha}_j^{\lambda}(1-\widehat{\alpha}_j^{\lambda}) \\
&= \left[1+\exp\{(\boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda} + \widehat{\beta}_0^{\lambda})\}\right]^{-1}\left[1+\exp\{-(\boldsymbol{x}_j^{\top}\widehat{\boldsymbol{\beta}}^{\lambda} + \widehat{\beta}_0^{\lambda})\}\right]^{-1} \\
&= V_{jj}^{\lambda}.
\end{aligned}
$$

Since $\widehat{\alpha}_j^\lambda$ is computed at the CC, the latter can determine each diagonal entry $\boldsymbol{V}_{jj}^\lambda$ for all $j \in 1, \ldots, n$ and proceed, as outlined in the main text, to reverse-engineer the individual feature data. Notably, this approach allows the CC to reconstruct the matrix $\boldsymbol{V}^\lambda$ without requiring access to the response vector $\boldsymbol{y}$.

References

[1] Li Y, Jiang X, Wang S, Xiong H, Ohno-Machado L. VERTIcal Grid lOgistic regression (VERTIGO). Journal of the American Medical Informatics Association. 2016;23(3):570–579. https://doi.org/10.1093/jamia/ocv146.

[2] Kim J, Li W, Bath T, Jiang X, Ohno-Machado L. VERTIcal Grid lOgistic regression with Confidence Intervals (VERTIGO-CI). AMIA Summits on Translational Science Proceedings. 2021;p. 355–363.

[3] Sion M. On general minimax theorems. Pacific Journal of Mathematics. 1958;8(1):171–176.