Revisiting VERTIGO and VERTIGO-CI: Confidentiality, Efficiency and Comparability

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

[1]Groupe de recherche interdisciplinaire en informatique de la santé (GRIIS), `https://griis.ca/`, [2]Département de mathématiques, Faculté des sciences, Université de Sherbrooke, Sherbrooke, Canada, [3]Département de médecine, Faculté de médecine et des sciences de la santé, Université de Sherbrooke, Sherbrooke, Canada, [4]Health Data Research Network Canada, [5]Institut Imagine, Université Paris Cité, Paris, France and [6]Department of Medical Informatics, Necker Hospital, AP-HP, Paris, France

*Corresponding author.
Jean-François Ethier, MD, PhD
GRIIS
Université de Sherbrooke, 3001, 12e Avenue Nord, Sherbrooke (Québec) J1H 5N4, Canada.
Phone: +1 (819) 821-8000, ext. 74977
Email: Jean-Francois.Ethier@USherbrooke.ca

Abstract

Objectives: This study aims to guide future use of the VERTIGO and VERTIGO-CI algorithms by (1) Highlighting confidentiality issues; (2) minimizing the number of communication rounds required; (3) offering an alternative algorithm that excludes the intercept from the penalty.

Materials and Methods: Quantities exchanged in the original algorithms were inspected and linear algebra was used to uncover reverse-engineering processes that the coordinating center can perform to re-create raw data. The objective function of the optimization problem was investigated to propose an alternative where one communication is needed to run the algorithm while being able to exclude the intercept from the penalty.

Results: It was shown that the coordinating center can retrieve all individual data using matrix operations when VERTIGO-CI is run. Individual data could also be retrieved in the case where parameter estimates associated with binary covariates are shared. The VERTIGO algorithm was adapted with reduced communications, and an algorithm was proposed to exclude the intercept from the penalty.

Discussion: While the use of VERTIGO-CI or VERTIGO with binary covariates does not imply directly sharing raw data, this works highlights the risks of reverse-engineering enabling the coordination node to effectively obtain the raw dataset of a data node. While inspecting quantities exchanged allowed to reduce the number of communications rounds, investigating the modeling choice, which translates as including or not the intercept in the penalty, was essential for interpretation matters.

Conclusion: This work highlights confidentiality breaches that may arise in distributed analytics methods, but also proposes an improved algorithm with reduced operational costs and enriching model interpretablity.

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

BACKGROUND AND SIGNIFICANCE

As the health research and management community strive to improve care for our citizens and ultimately their health, questions arising require more and more diverse datasets to find the answers. From genomic to environmental data, from quantified-self information to clinical care data warehouses, factors driving outcomes are collected by different organizations and stored in different systems. Traditionally, the approach to analysis has been to copy all required data and bring it a to central location, but the variety, scope and geographical coverage of data required often prevents this as legal, ethical, and social acceptability constraints arise. For example, Health Data Research Canada (HDRN) is tasked with facilitating large scale health data research like statistical analyses using genomic data from the Canadian Partnership for Tomorrow's Health (CanPath) cohort [1], in conjunction with provincial medico-administrative data. This scenario illustrates a case of vertically partitioned data [2], where different institutions hold distinct subsets of features corresponding to the same individuals. The intended analyses involved estimating parameters of statistical regression models and their associated confidence intervals (CIs), specifically using logistic regression for binary outcomes. To support these objectives, we explored methods capable of accommodating such data-sharing constraints.

Among the proposed solutions, the field of distributed analysis focuses on approaches that rely solely on the exchange of summary statistics to produce analytical results. It must be noted that the absence of direct raw data sharing does not inherently guarantee confidentiality, as participating entities may still be able to infer sensitive information through reverse-engineering of the exchanged quantities.

The ideal method does not require any raw data sharing outside the environments of the original data stewards (data nodes); provides claims of confidentiality, including retro-engineering; offers published reports of clinical use; and supports vertical distributed logistic regressions. When reviewing literature, no such method has been identified.

The next step was to look for a method with minimal sharing requirements. The algorithm VERTIGO [3] and its extension VERTIGO-CI [4] have been recently proposed for the setting of vertically partitioned data. These methods respectively provide parameter estimates for the penalized logistic regression model and enable CI estimation. Recent studies have integrated this approach into the analysis of distributed health data [5, 6]. 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.

While confidentiality is of paramount importance, other factors are also examined to identify the most suitable methods. As of now, human review of information leaving a secure environment for analysis is required for the majority of the data partners. Therefore, minimizing the number of communication rounds between nodes can have a significant impact on the time required to obtain the results. Finally, comparability with pooled methods is important to convince our colleagues that the results obtained are indeed a good substitute to the pooled version they have been using until now.

In order to ensure due diligence vis-à-vis the 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;

- **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 [7]. As a result, VERTIGO estimates are not directly comparable to those obtained from other algorithms [8], 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) optimize the operations performed on shared quantities to reduce the number of communication rounds in VERTIGO, and (3) propose an alternative approach to VERTIGO that excludes the intercept from the penalty.

MATERIALS AND METHODS

General methodological framework and mathematical foundations

The work presented here is expressed using a mathematical notation framework proposed in our previous work on horizontally distributed analyses [9]. The general methodological framework, including mathematical notations, is described below.

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 $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{a}) := \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{a}}$.

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:

$$\begin{aligned} \boldsymbol{\mathcal{K}} &:= [\boldsymbol{X}\ \mathbf{1}_n][\boldsymbol{X}\ \mathbf{1}_n]^\top = \boldsymbol{X}\boldsymbol{X}^\top + \mathbf{1}_n\mathbf{1}_n^\top \\ &= \begin{bmatrix} \sum_{i=1}^p x_{1i}^2 + 1 & \cdots & \sum_{i=1}^p x_{1i}x_{ni} + 1 \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^p x_{ni}x_{1i} + 1 & \cdots & \sum_{i=1}^p x_{ni}^2 + 1 \end{bmatrix}. \end{aligned} \tag{3}$$

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 $i, j \in \{1, \ldots, n\}$:

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

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.

Penalized logistic regression

VERTIGO and VERTIGO-CI rely on a penalized maximum likelihood estimation procedure, specifically the ridge logistic regression [10], 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 [8]. 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 Supplementary Material C). Let $\widehat{\beta}_{0,\text{alt}}^\lambda$ and $\widehat{\boldsymbol{\beta}}_{\text{alt}}^\lambda$ denote this solution.

Overview of VERTIGO

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

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

where $\widehat{\boldsymbol{\alpha}}^\lambda := [\widehat{\alpha}_1^\lambda,\ldots,\widehat{\alpha}_n^\lambda]^\top \in (0,1)^n$ 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

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

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

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

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

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

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 $\lambda^{-1}\sum_{i=1}^{n} \widehat{\alpha}_i^\lambda y_i$.

Overview of VERTIGO-CI

The VERTIGO-CI algorithm [4] 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 Supplementary Material A, 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 Supplementary Material A). 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.

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 $\boldsymbol{a} = \boldsymbol{A}\boldsymbol{d}$. Define the set $\mathcal{U}(\boldsymbol{d}) := \{z \in \mathbb{R} : z = \boldsymbol{b}^\top \boldsymbol{d},\ \boldsymbol{b} \in \{0,1\}^p\}$. 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 $\boldsymbol{A}$.

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 Supplementary Material B), 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.

Methodology pertaining to objective 3

The design of 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 Supplementary Material C.

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

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

RESULTS

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.

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 Supplementary Material A (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

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

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 $\widehat{\beta}^{\lambda}_0 + \boldsymbol{x}_j^{\top} \widehat{\boldsymbol{\beta}}^{\lambda}$ 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 Supplementary Material D for details).

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 $\lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)}\,\mathrm{diag}(\widehat{\boldsymbol{\alpha}}^{\lambda})\boldsymbol{y}$ 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 $\lambda^{-1}\boldsymbol{\mathcal{K}}^{(k)}\,\mathrm{diag}(\widehat{\boldsymbol{\alpha}}^{\lambda})\boldsymbol{y}$ 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 $\boldsymbol{X}^{(k)}$.

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

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

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 VERTIGO Optimized

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 VERTIGO Optimized

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.

Results pertaining to objective 3: Comparability

Our proposed modification to VERTIGO, 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 [11]. We refer to [12] 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

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

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

Algorithm 2 VERTIGO Optimized with no penalty over 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. Do:
   a. CC: solve problem (15) using an algorithm adapted for equality constrained optimization.

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

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 other 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 the sharing of raw data. This context might lead to assume that a method that avoids direct transmission of raw data renders it *de facto* compliant with these restrictions. These constraints rather mean that raw data must remain unknowable to any unauthorized party, regardless of the form in which data is exchanged with the original 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. 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 the problem 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. Other situations with a mix of binary and continuous predictors will require careful investigations by users before deploying them. On the other hand, we did not identify situations in which data nodes with more than three continuous covariates would be at risk of retro-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. Nevertheless, such brute force attempts cannot guarantee that individual-level data cannot be reconstructed. A clear and systematic framework for assessing the risk of re-identification based on shared quantities remains lacking in the current literature.

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 an alternative 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. 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 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.

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 VERTIGO algorithm to reduce communication costs and introduced an alternative formulation 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.

Competing interests

No competing interest is declared.

Acknowledgments

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.

References

1. The canadian partnership for tomorrow's health (canpath), 2024. Accessed May 20 2025. https://canpath.ca/en/.
2. Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: A systematic literature review. *ACM Transactions on Internet Technology*, 21(2):1–31, 2021.
3. Yong Li, Xiaoqian Jiang, Shuang Wang, Hongkai Xiong, and Lucila Ohno-Machado. VERTIcal grid lOgistic regression (VERTIGO). *Journal of the American Medical Informatics Association*, 23(3):570–579, 2016.
4. Jihoon Kim, Wentao Li, Tyler Bath, Xiaoqian Jiang, and Lucila Ohno-Machado. VERTIcal grid lOgistic regression with confidence intervals (VERTIGO-CI). *AMIA Summits on Translational Science Proceedings*, 2021:355–363, 2021.
5. Tsung-Ting Kuo, Rodney A. Gabriel, Jejo Koola, Robert T. Schooley, and Lucila Ohno-Machado. Distributed cross-learning for equitable federated models - privacy-preserving prediction on data from five California hospitals. *Nature Communications*, 16(1), 2025.
6. Arturo Moncada-Torres, Frank Martin, Melle Sieswerda, Johan Van Soest, and Gijs Geleijnse. VANTAGE6: an open source priVAcy preserviNg federaTed leArninG infrastructurE for Secure Insight eXchange. *AMIA Annual Symposium Proceedings*, 2020:870–877, 2021.
7. Hana Šinkovec, Georg Heinze, Rok Blagus, and Angelika Geroldinger. To tune or not to tune, a case study of ridge logistic regression in small or sparse datasets. *BMC Medical Research Methodology*, 21(199), 2021.
8. Jerome Friedman, Robert Tibshirani, and Trevor Hastie. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.
9. Félix Camirand Lemyre, Simon Lévesque, Marie-Pier Domingue, Klaus Herrmann, and Jean-François Ethier. Distributed Statistical Analyses: A Scoping Review and Examples of Operational Frameworks Adapted to Health Analytics. *JMIR Medical Informatics*, 12(1), 2024.
10. S. Le Cessie and J. C. Van Houwelingen. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(1):191–201, 1992.
11. Anqi Fu, Balasubramanian Narasimhan, and Stephen Boyd. CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, 94(14):1–34, 2020.
12. Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, Cambridge, United Kingdom, 2004.

Supplementary Material A. 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 Main text equation (10)).
   c. CC: compute the Hessian matrix $\boldsymbol{H}(\boldsymbol{\alpha}_{(s)})$ (using Main text 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. $\widehat{\boldsymbol{\alpha}}^{\lambda} \leftarrow \boldsymbol{\alpha}_{(s)}$.

Supplementary Material B. 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

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

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

Supplementary Material C. Mathematical development for the dual problem in the case of a non-penalized 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 $(\widehat{\beta}_{0,alt}^{\lambda}, \widehat{\boldsymbol{\beta}}_{alt}^{\lambda})$ to the optimization problem*

$$\max_{\beta_0 \in \mathbb{R}, \boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]^{\top} \in \mathbb{R}^p} l_{n,alt}^{\lambda}(\beta_0, \boldsymbol{\beta}) \,. \tag{1}$$

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

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

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

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 (1) has a unique finite solution. To do this, we first compute that

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

and that

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

From this, we obtain that

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

Hence, $l^{\lambda}_{n,\text{alt}}(\beta_0,\boldsymbol{\beta})$ is strictly concave in $(\beta_0,\boldsymbol{\beta})$. Therefore, if $l^{\lambda}_{n,\text{alt}}(\beta_0,\boldsymbol{\beta})$ admits a maximizer, it must be unique. To conclude that the optimization problem in (1) has a unique solution, it remains to show that $l^{\lambda}_{n,\text{alt}}(\beta_0,\boldsymbol{\beta})$ 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 (1). 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

$$\frac{\partial}{\partial \beta_0} l^{\lambda}_{n,\mathrm{alt}}(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}}, \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}) = \sum_{i=1}^{n} \frac{y_i \exp\{-y_i(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})\}}{1+\exp\{-y_i(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})\}}$$
$$= \sum_{i=1}^{n} \Big\{ \mathbb{I}(y_i = 1) - \frac{\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}{1+\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})} \Big\} = 0\,, \tag{2}$$

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 (1) can be narrowed to a compact set (see (5) below).

From the first equation, we deduce that

$$\|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \le \lambda^{-1} \sum_{i=1}^{n} \|\boldsymbol{x}_i\|_\infty\,. \tag{3}$$

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

$$\sum_{i:y_i=1} \Big\{ 1 - \frac{\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}{1+\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})} \Big\}$$
$$= \sum_{i:y_i=-1} \frac{\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}{1+\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}\,. \tag{4}$$

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 $n_1 \ge 1$, we have

$$\sum_{i:y_i=1} \Big\{ 1 - \frac{\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}{1+\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})} \Big\}$$
$$\ge 1 - \max_{i:y_i=1} \frac{\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}{1+\exp(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})}$$
$$= 1 - \max_{i:y_i=1} \frac{1}{1+\exp\{-(\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} + \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}})\}}$$
$$\ge 1 - \frac{1}{1+\exp\{-\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} - \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty \max_{1\le i\le n} \|\boldsymbol{x}_i\|_1\}}\,,$$

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 (4) satisfies

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

Consequently,

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

which implies that

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

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

Since from similar arguments we can show that $\widehat{\beta}^{\lambda}_{0,\mathrm{alt}} \le \log(n) + p\max_{1\le i\le n} \|\boldsymbol{x}_i\|_\infty \|\widehat{\boldsymbol{\beta}}^{\lambda}_{\mathrm{alt}}\|_\infty$, we conclude using (3) that

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

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

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

and

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

Consequently, the search space $\mathbb{R} \times \mathbb{R}^p$ in the optimization problem at (1) can be restricted to $\Theta_{\boldsymbol{x},\lambda}$, and the optimization problem at (1) is equivalent to

$$\max_{\beta_0 \in \Theta^0_{\boldsymbol{x},\lambda}, \boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p} l^{\lambda}_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta})\,. \tag{5}$$

Now from the relationship

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

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

$$\begin{aligned}
& l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) = \min_{\boldsymbol{\alpha} \in (\Theta^2_{\boldsymbol{x},\lambda})^n} l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) \\
& \text{with } l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^n \Big\{ \alpha_i y_i (\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) + L(\alpha_i) \Big\} - \frac{\lambda}{2} \boldsymbol{\beta}^\top \boldsymbol{\beta} \\
& \text{and } \Theta^2_{\boldsymbol{x},\lambda} = \big[\frac{1}{1 + \exp(c_{n,\boldsymbol{x},\lambda})}, \frac{1}{1 + \exp(-c_{n,\boldsymbol{x},\lambda})} \big] \,.
\end{aligned} \tag{6}$$

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

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

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

$$\Theta^3_{\boldsymbol{x},\lambda} := (\Theta^2_{\boldsymbol{x},\lambda})^n \cap \{\boldsymbol{\alpha} : \sum_{i=1}^n \alpha_i y_i = 0\} \,. \tag{7}$$

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 (5) can be expressed as

$$\max_{\beta_0 \in \Theta^0_{\boldsymbol{x},\lambda}, \boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p} l^\lambda_{n,\mathrm{alt}}(\beta_0, \boldsymbol{\beta}) = \max_{\boldsymbol{\beta} \in (\Theta^1_{\boldsymbol{x},\lambda})^p} \left(\min_{\boldsymbol{\alpha} \in \Theta^3_{\boldsymbol{x},\lambda}} l^\lambda_{n,\mathrm{alt}}(\boldsymbol{\beta}, \boldsymbol{\alpha}) \right). \tag{8}$$

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 (7).

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 (6), 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:

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

□

Supplementary Material D. 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 Main text Equation (12). Also, recall the expression of $\nabla_{\boldsymbol{\alpha}} J^\lambda(\boldsymbol{\alpha})$ (Main text equation (10)), that $\widehat{\boldsymbol{\alpha}}^\lambda$ satisfies $\nabla_{\boldsymbol{\alpha}} J^\lambda(\widehat{\boldsymbol{\alpha}}^\lambda) = 0$, that $\widehat{\boldsymbol{\beta}}^{\lambda(k)} = \lambda^{-1} \sum_{i=1}^n \widehat{\alpha}_i^\lambda y_i \boldsymbol{x}_i^{(k)}$ and that $\widehat{\beta}_0^\lambda = \lambda^{-1} \sum_{i=1}^n \widehat{\alpha}_i^\lambda y_i$.

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. Yong Li, Xiaoqian Jiang, Shuang Wang, Hongkai Xiong, and Lucila Ohno-Machado. VERTIcal grid lOgistic regression (VERTIGO). *Journal of the American Medical Informatics Association*, 23(3):570–579, 2016.
2. Jihoon Kim, Wentao Li, Tyler Bath, Xiaoqian Jiang, and Lucila Ohno-Machado. VERTIcal grid lOgistic regression with confidence intervals (VERTIGO-CI). *AMIA Summits on Translational Science Proceedings*, 2021:355–363, 2021.
3. Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.