VdistCox: Vertically distributed Cox proportional hazards model with hyperparameter optimization

Ji Ae Park[1] and Yu Rang Park[1]
[1]*Department of Biomedical Systems Informatics, Yonsei University College of Medicine, 50-1 Yonsei-ro, Seodaemun-gu, Seoul, 03722, Republic of Korea*
Email: jiaepark1717@yonsei.ac.kr; yurangpark@yuhs.ac

Vertically partitioned data is distributed data in which information about a patient is distributed across multiple sites. In this study, we propose a novel algorithm (referred to as VdistCox) for the Cox proportional hazards model (Cox model), which is a widely-used survival model, in a vertically distributed setting without data sharing. VdistCox with a single hidden layer feedforward neural network through extreme learning machine can build an efficient vertically distributed Cox model. VdistCox can tune hyperparameters, including the number of hidden nodes, activation function, and regularization parameter, with one communication between the master site, which is the site set to act as the server in this study, and other sites. In addition, we explored the randomness of hidden layer input weights and biases by generating multiple random weights and biases. The experimental results indicate that VdistCox is an efficient distributed Cox model that reflects the characteristics of true centralized vertically partitioned data in the model and enables hyperparameter tuning without sharing information about a patient and additional communication between sites.

Keywords: Cox proportional hazards model; vertically partitioned data; privacy protection; hyperparameter tuning; extreme learning machine.

1. Introduction

1.1. *Characteristics of biomedical data*

Biomedical data are distributed in different locations in the form of various sources. Distributed data can be divided into horizontally or vertically partitioned data based on their distributed form. When the sites (e.g., government agencies, business establishments, or hospitals) have the same variables but different data subjects, the distributed data across the sites are known as horizontally partitioned data. On the other hand, when the sites hold disjoint sets of features for the same data subjects, the distributed data are known as vertically partitioned data. Utilizing the distributed data can increase the generalizability of research, provide insights that can prevent disease, and deliver highly customizable care to patients by considering more information about the patient. However, the confidential nature and privacy issues of patient data limit the sharing of distributed data. The data protection law in the USA, HIPAA, restricts the sharing of important data. In the European Union, the General Data Protection Regulation established a well-formulated guideline for securing the confidentiality and privacy of citizens.[1] Additionally, Canada's PIPEDA, the UK's Data Protection Act (PDA), and Russia's federal law on personal data reflect the growing global awareness of the importance of data privacy and confidentiality.[2-4] Patients are increasingly aware of the use of personal data and they are reluctant to share their data. Furthermore, owners of distributed data sources may not want to share data with other agencies, according to their institutional policies.

1.2. *Vertically distributed survival model*

Survival analysis for a time-to-event outcome (i.e., the length of time from the starting point to an event of interest, such as mortality or disease) is widely used in biomedical research. The most common model in survival analysis is the Cox proportional hazards model (Cox model). To utilize the distributed data without data sharing for privacy preservation, many studies have developed horizontally[5-9] or vertically[10,11] partitioned data-based distributed algorithms for deep learning or statistical models. The various features required for predicting a patient's prognosis do not exist in a single institution. The features have mutually exclusive characteristics in the form of vertically partitioned data. A patient's prognosis can be predicted more precisely by using information about the same patient from different institutions such as hospitals, insurance companies, and government agencies. VERTICOX[11] is the only distributed Cox model based on vertically partitioned data. VERTICOX using alternating direction method of multipliers (ADMM) has an advantage of obtaining almost the same estimated parameter as the global model. However, the algorithm deals with the standard Cox model with a linearity assumption, which limits its application in many real-world data. Because the vertically partitioned data can easily become high-dimensional data and it is difficult to confirm the interaction relationship between features distributed across sites, assuming only a simple linear relationship can be a limitation. Furthermore, ADMM requires many iterations (i.e., 2,000 and 1,500 for real data with 20 and 10 features) to obtain stable model parameters.

1.3. *Objective*

To overcome the limitation of the linearity assumption, nonparametric approaches such as neural networks can be useful alternatives. Faraggi and Simon (1995)[12] proposed an approach for modeling survival data with a simple feed-forward neural network as the basis for a nonlinear proportional hazards model. We used the optimization technique of extreme learning machine (ELM)[13] under the framework of Faraggi and Simon for the nonlinear Cox model. ELM has single hidden layer feedforward neural networks (SLFNs) that randomly choose the input weights and analytically determine the output weights. In this study, we developed a vertically distributed Cox model (referred as to VdistCox) while avoiding the transmission of patient features, which considers various functional forms in the Cox model using ELM, including hyperparameter tuning in a one-shot manner.

2. Materials and Methods

2.1. *Cox model in non-partitioned data*

In the Cox model,[14] the hazard of individual i with risk vector $\boldsymbol{x_i}$ at time t can be rewritten as the product of a baseline hazard $h_0(t)$, and a positive function of the covariates as follows:

$$h_i(t) = h_0(t)\exp\left(f(\boldsymbol{x_i})\right), \quad (1)$$

where $f(\boldsymbol{x_i})$ can be any function of $\boldsymbol{x_i}$, and for a standard Cox model, $f(x_i) = x_i\beta$. In Faraggi and Simon,[12] $f(x_i)$ is replaced with the output of a neural network for a nonlinear proportional hazards model rather than a linear functional form. We consider the output of the ELM as $f(x_i)$ under the framework of Faraggi and Simmon. ELM is an efficient learning algorithm for SLFNs that randomly chooses the input weights and analytically determines the output weights.[13]

2.2. *Vertically distributed Cox model*

We considered the Cox model with neural networks by replacing $f(x_i)$ in Eq. (1) with the ELM output. The proposed model (VdistCox) is communication efficient without iterative communication between the server and sites owing to the characteristics of ELM optimization.

To implement VdistCox, we set one of the sites as the master site, which plays the role of a server, to aggregate the intermediate results from the sites and derive the final model. Throughout this study, the first site was the master site. The setting of the master site does not affect the model results. VdistCox requires the following assumptions before implementation:

- There is a unique identifier for each patient (e.g., study ID) shared across institutions.
- It is not necessary to store event and time outcome information in every site. One of the sites stored the outcome should be the master party.

To illustrate VdistCox, some notations are summarized in Table 1.

Table 1. Summary of notations for VdistCox

Notation	Description
K	Number of sites
N ($= n + \tilde{n}$)	Number of patients
X	$(n \times M)$ Feature matrix for model training
$\tilde{X}$	$(\tilde{n} \times M)$ Feature matrix for model validation
M	Number of features distributed across K sites
L	Number of nodes
S	Number of randomly generated input weight
$R(s)$	$((M+1) \times L)$ Random matrix of s-th input weight
$\beta(s)$	Output weight of s-th random input weight. L- dimensional vector
$g(.)$	Activation function
M_k	Number of features for the k-th party, k = 1, …, K
$R_k(s)$	$(M_k \times L)$ Random matrix of s-th input weight at k site, k = 2, …, K
X^k	$(n \times M_k)$ Feature matrix of k party for model training, k = 2, …, K
$\tilde{X}^k$	$(\tilde{n} \times M_k)$ Feature matrix of k party for model validation, k = 2, …, K

At the master site, N patients are randomly divided into n patients for model training and $\tilde{n}$ patients for model validation, and the information is shared to the other sites. The feature matrices of the training and validation sets are denoted by

$$X^{\boldsymbol{k}} = \begin{bmatrix} x_{1(1+\sum_{i=1}^{k-1} M_i)} & \cdots & x_{1(1+\sum_{i=1}^{k} M_i)} \\ \vdots & \ddots & \vdots \\ x_{n(1+\sum_{i=1}^{k-1} M_i)} & \cdots & x_{n(1+\sum_{i=1}^{k} M_i)} \end{bmatrix}_{n \times M_k}, \ \tilde{X}^k = \begin{bmatrix} \tilde{x}_{1(1+\sum_{i=1}^{k-1} M_i)} & \cdots & \tilde{x}_{1(1+\sum_{i=1}^{k} M_i)} \\ \vdots & \ddots & \vdots \\ \tilde{x}_{\tilde{n}(1+\sum_{i=1}^{k-1} M_i)} & \cdots & \tilde{x}_{\tilde{n}(1+\sum_{i=1}^{k} M_i)} \end{bmatrix}_{\tilde{n} \times M_k}. \quad (2)$$

$X^{\mathbf{1}}$ and $\tilde{X}^1$ of the master are $(n \times (1+M_1))$ matrices in which the first column of Eq. (2) is all 1. Each site randomly generates a hidden layer input weight matrix corresponding to the M_k features under a non-overlapping seed number range between sites, and the master site generates an input weight matrix including the hidden layer bias. The random matrix on the hidden layer input weights and biases is generated S times at each site. The s-*th* random matrix is denoted as

$$R_1(s) = \begin{bmatrix} b_1(s) & \dots & b_L(s) \\ r_{11}(s) & \cdots & r_{1L}(s) \\ \vdots & \ddots & \vdots \\ r_{M_1 1}(s) & \cdots & r_{M_1 L}(s) \end{bmatrix}_{(1+M_1) \times L}, R_k(s) = \begin{bmatrix} r_{(1+\sum_{i=1}^{k-1} M_i)1}(s) & \cdots & r_{(1+\sum_{i=1}^{k-1} M_i)L}(s) \\ \vdots & \ddots & \vdots \\ r_{(1+\sum_{i=1}^{k} M_i)1}(s) & \cdots & r_{(1+\sum_{i=1}^{k} M_i)L}(s) \end{bmatrix}_{M_k \times L}. \quad (3)$$

R(s), which is a centralized random matrix, is not known in reality, but it can be considered as $R(\mathrm{s})^T = [R_1(\mathrm{s})^T | \quad \ldots \quad | R_K(\mathrm{s})^T]$. Each k site (k = 2, …, K) calculates $T_k(s) = X^{\boldsymbol{k}} R_k(\mathrm{s})$ and $\tilde{T}_k(s) = \tilde{X}^{\boldsymbol{k}} R_k(\mathrm{s})$, and sends $\{T_k(\mathrm{s})\}_{s=1}^{S}$ and $\{\tilde{T}_k(\mathrm{s})\}_{s=1}^{S}$ to the master site. The master site calculates $T(s) = \sum_{k=1}^{K} T_k(s)$ and $\tilde{T}(s) = \sum_{k=1}^{K} \tilde{T}_k(s)$. Subsequently, the master site takes any activation function on $T(s)$ and $\tilde{T}(s)$, and hidden layer output matrices, $H(s) = g(T(s))$ of size $(n \times L)$ and $\tilde{H}(s) = g(\tilde{T}(s))$ of size $(\tilde{n} \times L)$, are derived at master site. The master site estimates L output weights, $(\beta(s))^{\mathrm{T}} = (\beta_1(s), \beta_2(s), \ldots, \beta_L(s))^{\mathrm{T}}$, which minimizes $-LL(\boldsymbol{\beta}(\boldsymbol{s}))$ of Eq. (4) using the Newton–Raphson method.

$$-LL(\boldsymbol{\beta}(\boldsymbol{s})) = \sum_{t=1}^{T} d_t \log\left(\sum_{j \in \mathcal{R}_t} \exp\left(\sum_{l=1}^{L} \beta_l(s)\, g(x_j, b_l(s), r_l(s))\right)\right) - \sum_{t=1}^{T} \sum_{u \in \mathcal{D}_t} \left(\sum_{l=1}^{L} \beta_l(s)\, g(x_u, b_l(s), r_l(s))\right) + \lambda \|\boldsymbol{\beta}(\boldsymbol{s})\|_2^2 \quad (4)$$

Here, T denotes the number of distinct event times. At time t, $\mathcal{D}_t$ is the event set of all samples whose event occurs at time t, d_t is the number of events, and $\mathcal{R}_t$ is the risk set of all samples who caused the event or censoring after t. The negative log-partial likelihood in Eq. (4) for the estimation of the output weights includes a regularization term with tuning parameter λ. The master site computes $\hat{f}(\tilde{x}) = \tilde{H}(\mathrm{s})\hat{\beta}(s)$. Subsequently, the concordance index[15] of R(s), $Cindex(R(\mathrm{s}))$, is calculated using $\hat{f}(\tilde{x})$ as a risk score. The master site selects R(s^*) and $\hat{\beta}(s^*)$ as the final hidden layer input weights, biases, and output weights of VdistCox, corresponding to s with the largest $Cindex(R(\mathrm{s}))$, where $s^* = argmax_s Cindex(R(\mathrm{s}))$. VdistCox is exactly the same as its centralized model because $T(s)$=$\sum_{k=1}^{K} T_k(s)$ and $\tilde{T}(s)$=$\sum_{k=1}^{K} \tilde{T}_k(s)$ are the same as $XR(\mathrm{s})$ and $\tilde{X}R(\mathrm{s})$. Fig. 1 shows the overall communication process and model structure of VdistCox.

There are three hyperparameters: $g(.)$, λ, and L, in VdistCox. The activation function and the regularization parameter can be adjusted at the master site. The two hyperparameters can be explored by setting various candidate values after obtaining $T(s)$ and $\tilde{T}(s)$ at the master site. The number of hidden nodes affects the size of the random matrix R; moreover, an additional communication between the master site and other sites is required to consider various L values. A more efficient method is to generate $\{R_k(\mathrm{s})\}_{s=1}^{S}$ of size $(M_k \times L_{max})$ by setting the maximum number of nodes, L_{max}. Subsequently, $R_k(\mathrm{s})$ is divided into various sizes of $(M_k \times L_1)$, $(M_k \times L_2)$, …, and $(M_k \times L_{max})$ at the master site, where $L_1 < L_2 < \ldots < L_{max}$. The number of nodes is adjusted by generating $R_k(\mathrm{s})$ of various sizes. Therefore, all three hyperparameters can be explored within one communication between the master site and other sites.

2.3. *Experimental Settings*

Two simulations were performed to confirm the characteristics of VdistCox in a vertically distributed setting, assuming two sites and four features. It was assumed that x_1 and x_2 are at site 1, x_3 and x_4 are at site 2, and site 1 is the master site with outcomes.

For various simulated data generations, the function of Eq. (1) was considered as follows:

- f_l (Linear): $\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$
- f_q (Quadratic + interaction): $\beta_1 {x_1}^2 + \beta_2 {x_2}^2 + \beta_3 {x_3}^2 + \beta_4 {x_4}^2 + \beta_5 x_1 x_3 + \beta_6 x_2 x_4$
- f_g (Gaussian + interaction):

$$log(5)\,exp\left(-\frac{{x_1}^2+{x_2}^2}{2(0.5)^2}\right)+log(5)\,exp\left(-\frac{{x_3}^2+{x_4}^2}{2(0.5)^2}\right)+\beta_1x_1x_2+\beta_2x_3x_4$$

We set $[0.5, 1, 0.5, 1]$ as $[\beta_1, \beta_2, \beta_3, \beta_4]$ of f_l, $[2, 1, 2, 1, 1, 1]$ as $[\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6]$ of f_q, and $[1, 1]$ as $[\beta_1, \beta_2]$ of f_g. x_1, x_2, x_3, and x_4 were randomly generated from a uniform distribution, U(-1, 1). The baseline hazard was derived from a Weibull distribution with a scale of 20 and a shape of 5. Given x_1, x_2, x_3, x_4, β's, and the baseline hazard, the event rate was set to 30%.

In the first simulation, we confirmed whether VdistCox can represent the true function by setting f_l and $f_q$or whether the interaction relationship between the vertically partitioned features can be elucidated. We manually selected the hyperparameter setting in this first simulation under several settings without a criterion for the hyperparameter optimization as follows: 10, 30, 100, and 300 for the hidden node, TanHRe, Sigmoid, ELU, Softplus, and LReLU[16] for the activation function, and 0.1, 10, 100, and 300 for the regularization parameter. (Sigmoid, 30, and 300 in the setting of f_l) and (Softplus, 30, and 0.1 in the setting of f_q) were selected as the activation functions L, and λ, respectively. The size of the simulated data was set to N = 2000, and the training and validation sets were randomly divided in an 8:2 ratio. S was set to 100.

In the second simulation, the results of VdistCox based on various hyperparameter settings were explored under the settings of f_l and f_g. As discussed in Section 2.2, to proceed with the hyperparameter tuning without additional communication, L_{max} was set to 300, and 10, 30, 100, and 300 hidden nodes were considered. TanHRe, Sigmoid, ELU, Softplus, and LReLU[16] were considered as the activation functions, and the values of 0.1, 10, 100, and 300 were considered as the regularization parameters. We explored the results of the hyperparameter settings for 4 hidden nodes×5 activation functions×4 regularization parameters = 80. The size of the second simulated data was set to N = 1500, out of which the external (N = 500) dataset was randomly extracted and then randomly divided into training (N = 800) and validation (N = 200) from the remaining N = 1000. In each function setting, S was set to 100, and the distribution of the 100 performances in the validation set under 80 hyperparameter settings was confirmed. We selected four hyperparameter settings from each of f_l and f_g, based on the following criteria among the 80 hyperparameter settings:

1. Activation function: By comparing the *Cindex*(R(s*)) values of five activation functions under L = 10, two activation functions with the largest and smallest values were selected.
2. L and λ: In the two selected activation functions, among the total L and λ combinations of 16, two combinations with the largest or smallest values of Cindex(R(s*)) were selected.

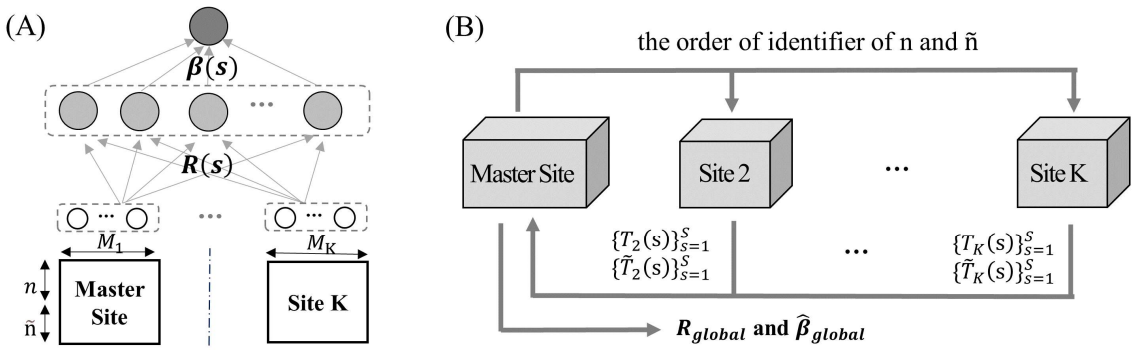

Fig. 1 Illustration of the VdistCox. (A) Model structure. (B) Process of communication.

In addition, we compared the performance of the test set between the centralized standard Cox model and the proposed model under the four hyperparameter settings selected for each function. The results were confirmed according to s^*, s^{min}, and s^{med} to examine the advantage of generating the random input matrix S times, where $s^* = argmax_s Cindex$ ($R(\mathrm{s})$), $s^{min} = argmin_s Cindex(R(\mathrm{s}))$, and s^{med} is s when $Cindex(R(\mathrm{s}))$ has a median value. The centralized standard Cox model was performed with N = 1,000, combined with both training and validation sets. For the second simulation, 100 different simulated data were generated. The four hyperparameter settings based on the aforementioned two criteria were selected using the first simulated data among 100 simulated data. The 100 simulations were performed under the selected four hyperparameter settings and the results thus obtained were compared with those of the standard Cox model.

Furthermore, we confirmed the validity of VdistCox with real-data using electronic Intensive Care Unit (eICU) Collaborative Research Database.[17] We considered 27 factors included in Acute Physiology, Age, and Chronic Health Evaluation (APACHE) scores as features and the length of stay from the date of ICU admission to the date of mortality during the ICU stay as the outcome of the Cox model. We extracted 2,486 stays with all 27 features and outcomes, hospitals corresponding to the number of beds ≥500, and Caucasians; 19 hospitals were included in 2,486 stays. We randomly selected 486 stays as the test set, and divided 2,000 stays 8:2 into the training and validation sets. The comparative analysis with the standard Cox model was also performed using the eICU data, and the same 2,000 stays were used for both VdistCox and the standard Cox model. After setting the centralized eICU data with 2,000 stays and 27 features, two vertical sites were assumed. Site 1 was a master site with 14 features and outcomes, where site 2 was a site with only 13 features. For hyperparameter setting, L_{max} was set to 500, and 10, 30, 100, 300, and 500 hidden nodes were considered. As the activation functions, the five functions were used in the same manner as the simulation, and six regularization parameters of 0.1, 10, 100, 300, 500, and 1,000 were considered. Hyperparameter settings of 5 hidden nodes × 5 activation functions × 6 regularization parameters = 150 were explored.

VdistCox was implemented with R software and the source code is available from the authors upon request.

3. Results

3.1. *Simulations*

Fig. 2 shows the results of the first simulation. The contour plot shows the relationship between x_1 and x_3 when x_2 and x_4 are zero and the relationship between x_2 and x_4 when x_1 and x_3 are zero. In addition, graphs (a) to (h) confirm that the proposed model adequately describes the interaction relationship between variables under $R(s^*)$ and $\hat{\beta}(s^*)$. The graphs in (a) and (b) represent the results of $\hat{f}_l$, which is the output of VdistCox, according to x_1 when x_3 is -1 and x_3 is 1, where $f_l = 0.5x_1+0.5x_3$. Because x_1 and x_3 have no interaction, the slopes of the graphs of (a) and (b) should not change regardless of whether x3 is -1 or 1, and the results reflect this fact efficiently. In addition, (c) and (d) show the result of $\hat{f}_l$ according to x_2 when x_4 is -1 and x_4 is 1, where $f_l = x_2+x_4$, and they have a parallel shape with no change in the slope. The true slopes of (a) and (b) are smaller than those of (c) and (d), which is also reflected in the results. In the setting of $f_q = 2\mathrm{x}_1^{\,2} + 2\mathrm{x}_3^{\,2} +$

x_1x_3, the results of VdistCox represent the true functions of x1 and f_q when x_3 is -1 and x_3 is 1 (see the results of graphs (e) and (f)). Further, because the interaction of x_1 and x_3 exists, the vertices of (e) and (f) are different under the same quadratic function. In $f_q = {x_2}^2 + {x_4}^2 + x_2x_4$, when x_4 is -1 and x_4 is 1, (g) and (h) on the graph of $\hat{f}_q$ according to x_2 have different vertices under the same quadratic function form owing to the interaction of x_2 and x_4. The true coefficient of quadratic terms (e) and (f) is larger than that of (h) and (g), and the result of VdistCox efficiently reflects the true relationship, as (e) and (f) are more concave than (h) and (g).

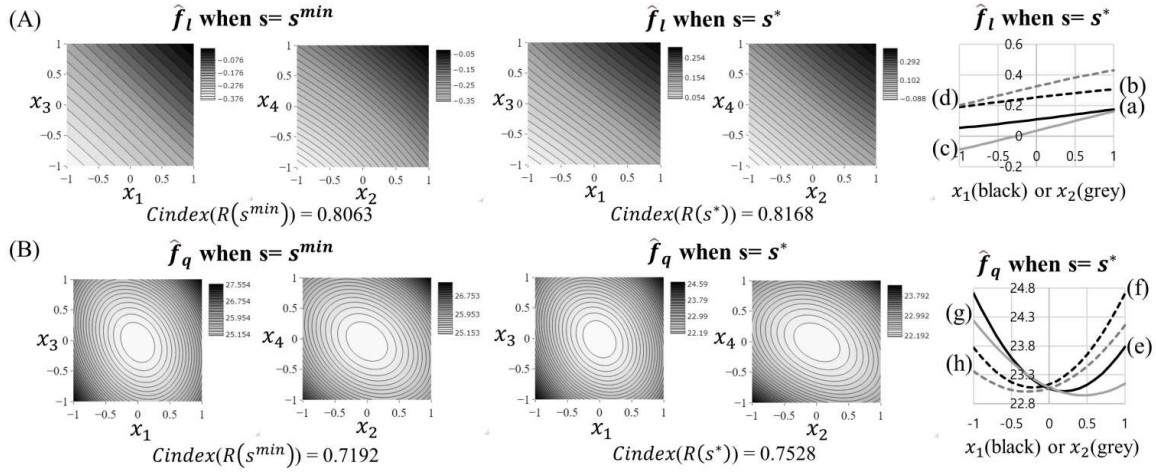

Fig. 2. Simulation results under (A) f_l and (B) f_q. Site 1 stores x_1 and x_2 and Site 2 stores x_3 and x_4. $s^* = argmax_s Cindex(R(s))$, $s^{min} = argmin_s Cindex(R(s))$, The true functions of a (black solid), b (black dashed), c (grey solid), d (grey dashed), e (black solid), f (black dashed), g (grey solid), and h (grey dashed) are $f(x) = 0.5x_1 - 0.5$, $f(x) = 0.5x_1 + 0.5$, $f(x) = x_2 - 1$, $f(x) = x_2 + 1$, $f(x) = 2{x_1}^2 - x_1 + 2$, $f(x) = 2{x_1}^2 + x_1 + 2$, $f(x) = {x_2}^2 - x_2 + 1$, and $f(x) = {x_2}^2 + x_2 + 1$, respectively.

Fig. 3 and 4 show the results of the second simulation. Fig. 3 shows the distribution of 100 Cindex(R(s))s at 80 hyperparameter settings. In the linear function setting, the performance distribution tended to increase as λ increased from 0.1 to 300. Additionally, as the number of nodes increased, the distribution of the performance did not significantly increase. Moreover, the value of Cindex(R(s^*)) was overall large in the Sigmoid among the five activation functions. However, in the nonlinear setting, as λ was small and the number of nodes increased, the performance generally increased. The LReLU had a high overall performance distribution compared to the other activation functions. The change in performance according to hyperparameter selection is larger in the nonlinear function than in the linear function. According to the two criteria of hyperparameter selection described in Section 2.3, in the linear function, Sigmoid was selected as the activation function with max(Cindex(R(s*))), and TanHRe was selected as the activation function with min(Cindex(R(s*)). The four settings of Sigmoid/L = 30/λ = 300, Sigmoid/L = 300/λ = 0.1, TanHRe/L = 10/λ = 300, and TanHRe/L = 300//λ = 0.1 were selected as the hyperparameter settings with $Cindex(R(s^*))$ values of 0.8610, 0.8287, 0.8510, and 0.8143, respectively. In the nonlinear function, LReLU was selected as the activation function with max($Cindex(R(s^*))$, and TanHRe was selected as the activation function with min($Cindex(R(s^*))$. The four settings of LReLU/L = 30/λ = 0.1, LReLU/L = 30/λ = 300, TanHRe/L = 30/λ = 0.1, and TanHRe/L = 100/λ = 300 were

selected as the hyperparameter settings with $Cindex(R(s^*))$ values of 0.7405, 0.5381, 0.7033, and 0.4711.

Fig. 4 shows the distribution of $Cindex(R(s^*))$, $Cindex(R(s^{med}))$, and $Cindex(R(s^{min}))$ in the validation and test sets of 100 simulations performed under four settings selected from linear and nonlinear, respectively. In a linear setting, the standard Cox model, which can be viewed as a true model, showed a higher performance distribution than VdistCox, and the performance results of s^* and s^{med} were similar. The two hyperparameter settings of Sigmoid/L = 30/λ = 300 and TanHRe/L = 10/λ = 300, which showed similar performance in the validation set, showed similar performance in the test set, and the performance distributions s^* and s^{med} in the two settings were similar to that of the standard Cox model. The s^* of Sigmoid/L = 30/λ = 300 showed the highest performance, with an average performance of 0.7821. The average performance of the standard Cox model is 0.7860. In all settings of nonlinear function of Fig.4, s^*, s^{med}, and s^{min} showed a higher distribution of performance for the test set than the standard Cox model. The two hyperparameter settings of LReLU/L = 30/λ = 0.1 and TanHRe/L = 30/λ = 0.1, which showed similar performance in the validation set, showed similar performance in the test set, and the s^* of LReLU/L = 30/λ = 0.1 showed the highest performance with an average performance of 0.6677. In both the linear and nonlinear functions, s^* under the hyperparameter setting, which had the highest performance in the validation set, showed the highest performance in the test set on average.

3.2. *Real data*

Additionally, we explored 150 hyperparameter settings to confirm validity in real data, and four settings of ELU/L = 300/λ = 1000, ELU/L = 500/λ = 0.1, Sigmoid/L = 500/λ = 10, and Sigmoid/L = 500/λ = 0.1 were selected. As summarized in Table 2, the differences in performance in the validation and test sets between the four settings was quite large. Similar to the simulation results, the performance in the test set was also the highest at ELU/L = 300/λ = 1000, which had the highest performance in the validation set; s^{med} and s* in this setting showed higher performance than the standard Cox model.

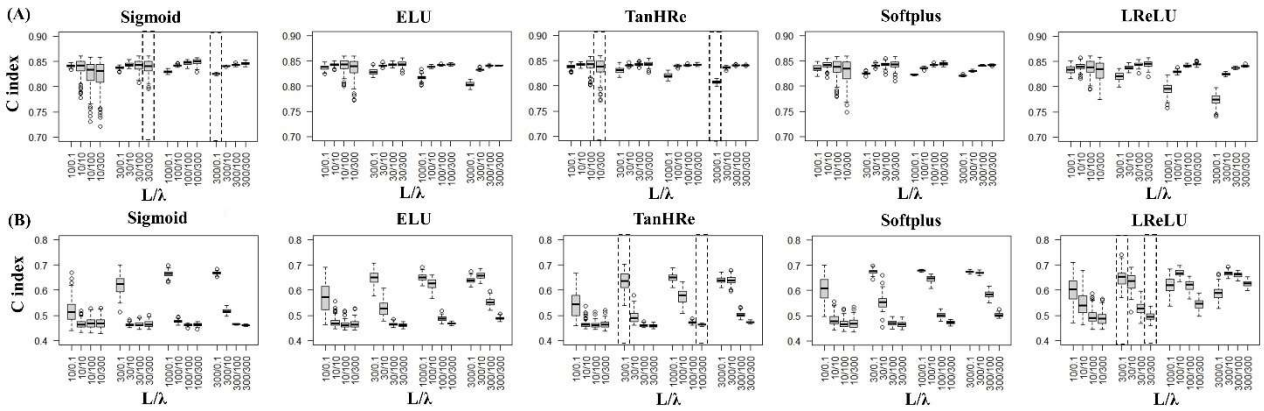

Fig. 3. Simulation results on distribution of $\{Cindex(R(s))\}_{s=1}^{100}$ at each hyperparameter setting under (A) f_l and (B) f_g settings. Dashed boxes represent selected four hyperparameter settings based on the two criteria described in section 2.3.

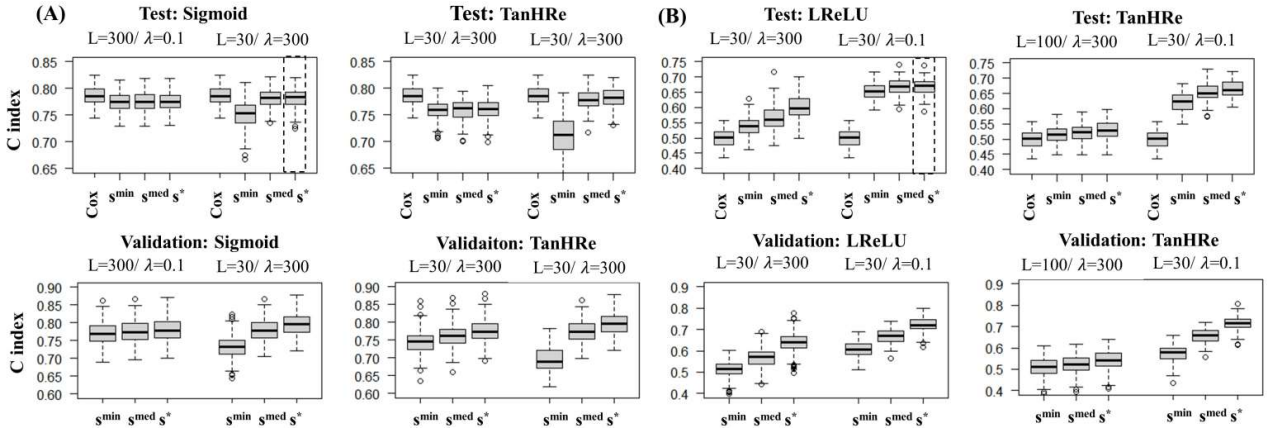

Fig. 4. Results on performances distribution in validation and test sets based on 100 simulations under the four hyperparameter settings of (A) f_l and (B) f_g. Dashed boxes represent the best results of performance among four hyperparameter settings.

Table 2. Results of performance as measured by the C-index in validation and test sets under vertical two sites setting based on eICU dataset.

VdistCox	**ELU**				Sigmoid			
	300/1000 (L/λ)		500/0.1 (L/λ)		500/10 (L/λ)		500/0.1 (L/λ)	
	validation	test	validation	test	validation	test	validation	test
s^{min}	0.8170	0.7149	0.4216	0.4458	0.7938	0.7162	0.2563	0.4253
s^{med}	0.8296	0.7204	0.5419	0.5086	0.8144	0.7154	0.3686	0.4686
s^*	**0.8466**	**0.7294**	0.7440	0.6017	0.8422	0.7159	0.7539	0.6502
Standard Cox model	test: 0.7160							

Bold represents the best results in the validation and the test sets in VdistCox.

4. Discussion

VdistCox shares only the value obtained by multiplying the feature value by the random value independently generated at each site in a privacy-preserving manner, and it has an efficient process that requires only one communication between the master site and other sites. Because VdistCox derives the exact same model as its centralized model without data sharing, it can provide a stable distributed model if the centralized ELM-based Cox model is valid. We confirmed the validity and characteristics of the proposed model through experiments using simulated and real data.

According to the results of the first simulation (Fig.2), VdistCox showed the real functional form between the variables, and it also reflected the interaction relationship between the vertically partitioned features.

To overcome the instability caused by the randomness, of the input weights and hidden biases, we generated the matrix of random input weights and hidden biases S times and selected the best random matrix among them. In the results of the performance of the test and validation sets of the second simulation (Fig.4), the performance of s^* and s^{med} was similar in the linear function setting, however it was different in the nonlinear function setting. This indicates that it is efficient to generate the R matrix multiple times in the nonlinear function setting. However, even in the nonlinear function, there was no difference in the performances of s^* and s^{med} depending on the hyperparameter selection (in the case of LReLU/L = 30/λ = 0.1). This means that hyperparameter selection could be

a more important factor than the randomness of R. However, exploring multiple R can prevent choosing the worst random weights. The results for the performance of s^{min} were worse compared to those of $s^{*}$and s^{med} in all cases. However, in the results on real data (Fig.4), the performance on the test set of s^{min} was slightly better than that of $s^{*}$and s^{med} in Sigmoid/L = 500/λ = 10. This indicates that the selection of a random value with good performance in the validation dataset may be a selection with low generalizability in external validation. However, considering the overall results, the best performance on the test dataset was s^{*}.

Hyperparameter tuning can be crucial for obtaining a good trade-off between accuracy and convergence in models with neural networks; it could affect the quality of the learned model.[18] To train a distributed model under different hyperparameter settings, many computing resources are required, and the evaluation of hyperparameters is extremely expensive for a large-scale distributed dataset.[19] In the framework of VdistCox, the three hyperparameters can be explored without additional communication between the master and other sites after obtaining the T and $\tilde{T}$ matrices at the master site. The importance of hyperparameter selection was confirmed through experiments. The results of the second simulation showed a large difference in performance according to the 80 hyperparameter settings, and the importance of the hyperparameter was greater in the nonlinear function than in the linear function settings (Fig. 3). Further, we confirmed that the setting with good performance in the validation set also showed good performance in the test set (Fig.4 and Table 1). Assuming a distributed model with iterative communication, if we want to explore 80 hyperparameter settings, the distributed model will have to be run 80 times, which consumes a significant amount of computing resources. In VdistCox, a wide range of hyperparameter choices can be implemented in a one-shot manner.

Comparing the results of VdistCox and the centralized standard Cox model, in the linear function setting of the second simulation, VdistCox (Sigmoid/L = 30/λ = 300) showed a similar performance to the standard Cox model, which is a true model. In addition, in real data where the true function is unknown, the performance of VdistCox (ELU/ L = 300/ λ =1000) was higher than that of the standard Cox model, which may indicate that the true relationship between the 27 features is not linear. Vertically partitioned data combines features of various characteristics for the same patient from different sites. Therefore, compared to the data from a single site, the number of features in vertically partitioned data is more likely to become high dimensional, and the $f(x_i)$ of Eq. (1) cannot be determined in advance because we cannot distinguish which interaction exists between the numerous distributed variables. Compared to the standard Cox model based VERTICOX, the VdistCox may flexibly reflect $f(x_i)$ based on the real data characteristics in the distributed data that is difficult to share between the sites. Additionally, there is a possibility that the number of features exceeds the number of patients in vertically partitioned data in which only the number of features increases in a certain patient group (N<<M). In these data characteristics, the parameter estimation in the standard Cox model may become unstable and the accuracy of prediction may decrease. Therefore, compared to VERTICOX, which aims to accurately estimate the parameter of the standard Cox model, the VdistCox can provide a stable predictive model in high-dimensional vertically partitioned data of N<<M. Moreover, VERTICOX requires several iterations to obtain stable parameter estimates (i.e., 2,000 and 1,500 for real data with 20 and 10 features). By contrast, VdistCox requires only one communication including hyperparameter optimization.

In this study, we confirmed the characteristics and validity of our novel model, VdistCox. However, because it was performed using restricted simulated and real data, it is possible that the validity of VdistCox has not been sufficiently proven in this paper. Additionally, we have not proposed an index that can interpret the influence of features such as the hazard ratio provided by the VERTICOX. However, in the results of the first simulation, the relative influence between features from VdistCox were identified. For example, in the setting of f_l, true β_1 and β_2 were set to 0.5 and 1, respectively, and the slope of x_2 was greater than that of x_1 in (a) to (d) of Fig. 2. Furthermore, in the setting of f_q, true β_1 and β_2 were set to 2 and 1, respectively, and the concave degree of x_1 was greater than that of x_2 in (e) to (h) of Fig. 2. Explaining the influence of each feature in terms of interpretation of the model is important and further discussion in VdistCox on the interpretation is required.

5. Conclusion

The model proposed in this study, VdistCox, is communication-efficient vertically distributed Cox model by sharing once the intermediate results that are obtained by multiplying the features of each site to the input weight randomly generated at each site, while avoiding data sharing. In VdistCox using ELM, we proposed generating random input weights multiple times and a hyperparameter tuning process. In our experiments, the importance of randomness on input weights and hyperparameter selection depended on the data type (e.g., linear or nonlinear relationship between features). However, because confirming the true relationship between features in a real vertically distributed environment is difficult, considering multiple random input weights and hyperparameter tuning can be an effective means for a stable vertically distributed Cox model.

6. Acknowledgments

This work was supported by the Bio-Industrial Technology Development Program (20014841) and funded by the Ministry of Trade, Industry, and Energy (MOTIE, Korea). This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number : HI19C1330).

References

1. Oya Beyan, Ananya Choudhury, Johan van Soest, Oliver Kohlbacher, Lukas Zimmermann, Holger Stenzhorn, Md. Rezaul Karim, Michel Dumontier, Stefan Decker, Luiz Olavo Bonino da Silva Santos, Andre Dekker; Distributed Analytics on Sensitive Medical Data: The Personal Health Train. Data Intelligence 2020; 2 (1-2): 96–107. doi:
2. Office of the Privacy Commissioner of Canada. The Personal Information Protection and Electronic Documents Act (PIPEDA). Available at: https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in canada/the-personal-information-protection-and-electronic-documents-act-pipeda/.
3. The Data Protection Act. Available at: https://www.gov.uk/data-protection.

4. Federal Law of 27 July 2006 N 152-FZ on Personal Data. Available at: https://pd.rkn.gov.ru/authority/p146/p164/.
5. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication efficient learning of deep networks from decentralized data. In A. Singh and X. J. Zhu, editors, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, volume 54 of Proceedings of Machine Learning Research, pages 1273–1282. PMLR, 2017. URL http://proceedings. mlr.press/v54/mcmahan17a.html.
6. K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards federated learning at scale: System design. CoRR, abs/1902.01046, 2019. URL http://arxiv.org/abs/1902.01046
7. Lu C, Wang S, Ji Z, Wu Yuan, Xiong Li, Jiang Xiaoqian, Ohno-Machado Lucila. WebDISCO: a web service for distributed cox model learning without patient-level data sharing. J Am Med Inform Assoc. 2015 Nov;22(6):1212–9. doi: 10.1093/jamia/ocv083.
8. Duan R. Learning from local to global-an efficient distributed algorithm for modeling time-to-event data. bioRxiv. 2021 doi: 10.1101/2020.03.04.977298.
9. PARK, Ji Ae, et al. Weight-Based Framework for Predictive Modeling of Multiple Databases With Noniterative Communication Without Data Sharing: Privacy-Protecting Analytic Method for Multi-Institutional Studies. JMIR medical informatics, 2021, 9.4: e21043.
10. S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. CoRR, abs/1711.10677, 2017. URL http://arxiv.org/abs/1711.10677.
11. DAI, Wenrui, et al. VERTICOX: Vertically Distributed Cox Proportional Hazards Model Using the Alternating Direction Method of Multipliers. IEEE Transactions on Knowledge and Data Engineering, 2020.
12. Faraggi, D. and Simon, R. (1995). A neural network model for survival data. Statistics in medicine, 14(1):73–82.
13. Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2004, July). Extreme learning machine: a new learning scheme of feedforward neural networks. In 2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541) (Vol. 2, pp. 985-990). Ieee.
14. K. Dietz et al., Stat. Biol. Health Logistic Regression SelfLearn. Text., vol. 2, pp. 102–124, 2002.
15. UNO, Hajime, et al. On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. Statistics in medicine, 2011, 30.10: 1105-1117.
16. RATNAWATI, Dian Eka, et al. Comparison of activation function on extreme learning machine (ELM) performance for classifying the active compound. In: AIP Conference Proceedings. AIP Publishing LLC, 2020. p. 140001.
17. Le Gall J. A New Simplified Acute Physiology Score (SAPS II) Based on a European/North American Multicenter Study. JAMA 1993 Dec 22;270(24):2957.
18. CHARLES, Zachary; KONEČNÝ, Jakub. On the outsized importance of learning rates in local update methods. arXiv preprint arXiv:2007.00878, 2020.
19. KAIROUZ, Peter, et al. Advances and open problems in federated learning. Foundations and Trends® in Machine Learning, 2021, 14.1–2: 1-210.