JOURNAL OF APPLIED STATISTICS
https://doi.org/10.1080/02664763.2021.1982878

Regularized regression when covariates are linked on a network: the 3CoSE algorithm

Matthias Weber[a], Jonas Striaukas[b], Martin Schumacher[c] and Harald Binder[c]

[a]School of Finance, University of St. Gallen, St. Gallen, Switzerland; [b]F.R.S.-FNRS, Université Catholique de Louvain, Louvain-la-Neuve, Belgium; [c]Institute of Medical Biometry and Statistics, Faculty of Medicine and Medical Center, University of Freiburg, Freiburg im Breisgau, Germany

ABSTRACT
Covariates in regressions may be linked to each other on a network. Knowledge of the network structure can be incorporated into regularized regression settings via a network penalty term. However, when it is unknown whether the connection signs in the network are positive (connected covariates reinforce each other) or negative (connected covariates repress each other), the connection signs have to be estimated jointly with the covariate coefficients. This can be done with an algorithm iterating a connection sign estimation step and a covariate coefficient estimation step. We develop such an algorithm, called 3CoSE, and show detailed simulation results and an application forecasting event times. The algorithm performs well in a variety of settings. We also briefly describe the publicly available R-package developed for this purpose.

ARTICLE HISTORY
Received 19 January 2021
Accepted 13 September 2021

KEYWORDS
Regressions on networks; network penalty; high-dimensional data; machine learning

1. Introduction

Network data have become increasingly important in the last few decades. This concerns a variety of types of data and disciplines, including pathway data in biology that reveals information relevant for medical research, data on social networks from social media websites with up to a couple of billion users, or networks of financial institutions relevant for assessing financial stability and designing regulation. There are already some statistical methods to deal with network data, but the development of such methods seems to be generally still in early stages.

In this paper, we introduce a method to incorporate network information into a regularized regression setting and provide simulations showing that the method performs well. The method is an algorithm based on Li and Li [9] who propose a particular network penalty on top of a Lasso penalty to incorporate network information about the covariates (that is, it is known which covariates are linked on a network). This method assumes that all network connections have a positive sign (that is that they are of an activating or enforcing type). However, there may also be network connections with negative signs (connections of a repressing type). Often, the signs of the connections are not even known ex-ante.

CONTACT Jonas Striaukas jonas.striaukas@uclouvain.be, jonas.striaukas@gmail.com

Therefore, the algorithm described in this paper estimates the covariate coefficients and the connection signs simultaneously.

The idea of the algorithm can be summarized as follows. There are two steps. In a covariate coefficient estimation step, the covariate coefficients are estimated by maximizing a penalized log-likelihood given estimates of the connection signs. Then, in a connection sign estimation step, the signs of the connections are estimated, taken the covariate coefficients as given from the last coefficient estimation step. These two steps are then iterated.

One of the goals of this method is to improve prediction performance. The method can, for example, be used to forecast event times of cancer patients and thereby improve patients' treatments or add in the development of new medical treatments. In addition to this, the method can help to gain knowledge about which covariates have an influence on the outcome and which do not (sensitivity and specificity) and about the signs of the coefficients. The latter can, in cases where it is important to understand the signs of the connections, be a goal in its own.

Situating our approach in the literature, there are already several approaches that make use of double penalties on the log-likelihood and that can deal with covariates connected on a network [8–12,23].[1] However, there are only very few methods thus far allowing for the explicit incorporation of negative connection signs in the regularized regression framework [10,11,17]. These few existing methods estimate the connection signs in a simple one-step procedure (one step for the sign estimation, so that estimating connection signs and covariate coefficients comprise two steps). Our method differs in that it contains an algorithm to estimate connection signs and covariate coefficients jointly. Gained information on one of these two components is used for the estimation of the other component in an iterative procedure.

This paper is organized as follows. Section 2 describes the algorithm to estimate covariate coefficients and connection signs and its implementation in a new accompanying R-package. Section 3 shows simulation studies analyzing prediction performance, sensitivity and specificity, and the fraction of correctly identified connection signs. This section also shows simulations illustrating in which cases the iteration of the coefficient estimation step and the connection sign estimation step improves over only estimating both parts once. Section 4 applies the method to time-to-event microarray data. Section 5 concludes.

2. Method

We first describe the setting and background. Then, we introduce the new algorithm and thereafter discuss its implementation.

2.1. Background

We consider a continuous response y and covariate matrix X. The numbers of observations and covariates are n and p, respectively. Thus, $y = (y_1, \dots, y_n)^{\mathrm{T}}$ and X is an $n \times p$-matrix with rows $x_i^T = (x_{i1}, \dots, x_{ip})$. $x_{(j)}$ denotes the jth column. We assume y to be centered and X standardized, which means $\sum_{i=1}^{n} y_i = 0$, $\sum_{i=1}^{n} x_{ij} = 0$ and $\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1$ for $j = 1, \dots, p$. In the classical linear model $y_i = x_i^T\beta + \epsilon_i$, with $\epsilon_i \sim N(0, \sigma^2)$.

The additional information for the regression problem is given by a network depicted by a weighted graph. The vertices are the covariates and the edges indicate the relationship between the covariates. The network information is incorporated into the network penalty via the normalized Laplace matrix of the associated graph. This is a $p \times p$ -matrix defined as

$$(L)_{uv} = \begin{cases} 1 - \dfrac{w(u,v)}{d_u} & \text{if } u = v \text{ and } d_u \neq 0, \\ \dfrac{-w(u,v)}{\sqrt{d_u d_v}} & \text{if } u \text{ and } v \text{ are adjacent}, \\ 0 & \text{otherwise}, \end{cases}$$

where u and v stand for the uth and vth covariate and $w(u,v)$ denotes the weight of the edge that links the uth and vth covariate (see [3]). Often the information is given via a connection matrix which consists only of zeros and ones indicating only which covariates are connected, thus $w(u,v)$ is usually zero or one. d_u is the degree of vertex u defined as the sum of $w(u,v)$ over all vertices v that are linked to vertex u.

Li and Li [9] propose to estimate β by minimizing the following penalized residual sum of squares (selecting λ_1 and λ_2 via 10-fold cross-validation):

$$\mathrm{RSS}(\lambda_1, \lambda_2, \beta) = (y - X\beta)^{\mathrm{T}}(y - X\beta) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \beta^T L \beta.$$

The set of all edges is denoted by $\{u \sim v\}$ (this is the set of all directly linked pairs of predictors). It is then $\beta^T L \beta = \sum_{u \sim v} (\frac{\beta_u}{\sqrt{d_u}} - \frac{\beta_v}{\sqrt{d_v}})^2 w(u,v)$.

2.2. Ideas behind the algorithm

It is implicitly assumed in Li and Li [9] that all connections between the connected covariates are positive, that is, that they influence the outcome in the same direction. It is likely, though, that some of the connections have a negative sign (in a biological application, for example, if a transcription factor suppresses another gene). In this case, it would be suitable to add a penalty of the form $\lambda_2(\frac{\beta_u}{\sqrt{d_u}} + \frac{\beta_v}{\sqrt{d_v}})^2 w(u,v)$ rather than $\lambda_2(\frac{\beta_u}{\sqrt{d_u}} - \frac{\beta_v}{\sqrt{d_v}})^2 w(u,v)$ (see [10]).

It is also possible to look at a different penalty matrix. In addition to the normalized Laplacian (or mutations of it that arise through negative connection signs), we also use the combinatorial Laplacian, which is defined by

$$(L_{\mathrm{comb}})_{uv} = \begin{cases} d_u - w(u,u) & \text{if } u = v \text{ and } d_u \neq 0, \\ -w(u,v) & \text{if } u \text{ and } v \text{ are adjacent}, \\ 0 & \text{otherwise}. \end{cases}$$

With positive connection signs, the combinatorial Laplacian leads to the following penalized log-likelihood (maximizing this penalized log-likelihood is equivalent to minimizing the penalized RSS). Thereby, $\ell(\beta)$ denotes the log-likelihood of the unpenalized linear regression problem (scaled, so that the same tuning parameters are obtained as when minimizing the penalized RSS)

$$l(\lambda_1, \lambda_2, \beta) = \ell(\beta) - \lambda_1 \sum_{j=1}^{p} |\beta_j| - \lambda_2 \sum_{u \sim v} (\beta_u - \beta_v)^2 w(u, v).$$

The difference between using the normalized and the combinatorial Laplacian as penalty matrices is thus that the normalized Laplacian penalizes the difference of regression coefficients after dividing them by the square root of their degree, while the combinatorial Laplacian penalizes just the difference between two connected covariates (independent of their degree, i.e. independent of how many other covariates a covariate is connected to).

We define for all $i, j \in \{1, \ldots, p\}$

$$\xi_{ij} = \begin{cases} -1 & \text{if there is a negative connection between covariates } i \text{ and } j, \\ 1 & \text{otherwise.} \end{cases}$$

Given an initial penalty matrix M_1 with $(M_1)_{ij} = 0$ if covariates $i \neq j$ are not connected (e.g. the normalized or combinatorial Laplacian), the optimal solution would be to use the penalty matrix with entries $-\xi_{ij}|(M_1)_{ij}|$ for $i \neq j$ and $|(M_1)_{ii}|$ otherwise. However, in many cases the ξ_{ij} are unknown and have to be estimated.[2] Similar to $\{u \sim v\}$, $\{u \overset{+}{\sim} v\}$ denotes the set of all connections assumed to have positive signs and $\{u \overset{-}{\sim} v\}$ that with negative signs. Then, we have

$$\sum_{u \overset{+}{\sim} v} \left(\frac{\beta_u}{\sqrt{d_u}} - \frac{\beta_v}{\sqrt{d_v}} \right)^2 w(u, v) + \sum_{u \overset{-}{\sim} v} \left(\frac{\beta_u}{\sqrt{d_u}} + \frac{\beta_v}{\sqrt{d_v}} \right)^2 w(u, v)$$
$$= \sum_{u \sim v} \left(\frac{\beta_u}{\sqrt{d_u}} - \hat{\xi}_{uv} \frac{\beta_v}{\sqrt{d_v}} \right)^2 w(u, v).$$

Given estimates of the covariate coefficients (obtained via maximizing a penalized log-likelihood with some set of connection signs), the connection signs can be estimated (anew) as follows. To estimate the connection sign between covariates i and j, all covariate coefficients $\hat{\beta}_k$ are kept fixed, except for the two coefficients whose connection sign is being estimated ($k \neq i, j$), and then a small linear model is fitted for covariates i and j only. A connection sign is estimated to be positive if the coefficient estimates of the connected covariates in this small linear model have the same sign. This is a reasonable estimate of the connection sign as in many applications positively connected covariates influence the output variable in the same direction, while negatively connected covariates influence the output variable in opposite directions. These small linear models should thus reveal information about the sign of the connection.

To be more detailed, the connection signs can be estimated as follows. $X^{-(i,j)}$ denotes the input matrix excluding columns i and j, $\hat{\beta}^{-(i,j)}$ denotes the estimate of β excluding $\hat{\beta}_i$ and $\hat{\beta}_j$, while $x_{(i)}$ denotes again the ith column of the input matrix. Fitting the small linear model for covariates i and j means considering the new response $\tilde{y} = y - X^{-(i,j)}\hat{\beta}^{-(i,j)}$ and minimizing

$$\sum_{k=1}^{n} (\tilde{y}_k - X_{ki}\beta_i - X_{kj}\beta_j)^2 = \left(\tilde{y} - (x_{(i)}, x_{(j)}) \begin{pmatrix} \beta_i \\ \beta_j \end{pmatrix}\right)^{\mathrm{T}} \left(\tilde{y} - (x_{(i)}, x_{(j)}) \begin{pmatrix} \beta_i \\ \beta_j \end{pmatrix}\right)$$

over $(\beta_i, \beta_j)^{\mathrm{T}}$. Let $(\hat{\beta}_i{}^*, \hat{\beta}_j{}^*)^{\mathrm{T}}$ denote the minimizer of the above residual sum of squares. If and only if the signs of $\hat{\beta}_i{}^*$ and $\hat{\beta}_j{}^*$ are different, the connection between covariates i and j is estimated to have negative sign.

As starting values for the connection sign estimates, the signs of the empirical covariance of the columns of the input matrix can be taken. Because the covariate matrix is standardized, this boils down to the sign of $x_{(i)}^T x_{(j)}$. Thus, the starting value of a connection sign is positive if $x_{(i)}^T x_{(j)} \geq 0$ and negative otherwise.

2.3. The 3CoSE algorithm

The description above contains the ingredients of the algorithm. We propose to call it **3CoSE** (pronounced 'three-cose'), standing for **Co**variate **Co**efficient and **Co**nnection **S**ign **E**stimation. Given a log-likelihood $\ell(\beta)$, a penalty matrix M_1, and fixed penalty parameters, the algorithm consists of the following steps:

(1) Determine starting values for the connection sign estimates via the empirical covariance, $\hat{\xi}_{ij} = \operatorname{sign}(x_{(i)}^T x_{(j)})$.
(2) Estimate β by maximizing the penalized log-likelihood

$$l(\lambda_1, \lambda_2, \beta) = \ell(\beta) - \lambda_1 \sum_{j=1}^{p} |\beta_j| - \lambda_2 \beta^T M \beta$$

with the current estimates of the connection signs, where $(M)_{ij} = -\hat{\xi}_{ij}|(M_1)_{ij}|$ for $i \neq j$ and $(M)_{ii} = |(M_1)_{ii}|$.
(3) Update the connection sign estimates by running mini OLS models, with the current estimate of β from step 2, so that $\hat{\xi}_{ij} \leftarrow \operatorname{sign}(\hat{\beta}_i^*) \cdot \operatorname{sign}(\hat{\beta}_j^*)$.
(4) Iterate steps 2 and 3 until convergence (or for a fixed number of repetitions).

This procedure is often still feasible when straightforward maximization of the penalized log-likelihood over all coefficients and connection signs is computationally prohibitive (which is not uncommon for big or high-dimensional data). The algorithm usually converges. When it does not converge, it can be run for a certain number of repetitions. It is very unlikely that a large number of connections have still changing signs and that their coefficients are important. It is much more likely that the few connections with still changing signs have coefficients that are estimated to be zero. No convergence is thus not necessarily problematic.

2.4. Implementation and R-Package LassoNet

For these simulations, the accompanying R-package LassoNet was developed, which is publicly available [16]. It implements the covariate coefficient estimation step via coordinate descent (see [5]). This means that only one covariate is updated at a time; all β_k, $k = 1, \dots p$ except one, say β_j, are kept fixed at their current value and the maximization of the log-likelihood is conducted only for β_j, which is then updated. This is carried out for all $\beta_j, j = 1, \dots, p$, and then the whole cycle is repeated until convergence.

With network penalty matrix M, maximizing the log-likelihood is equivalent to minimizing this residual sum of squares:

$$\begin{aligned}\mathrm{RSS}(\lambda_1, \lambda_2, \beta) &= \sum_{i=1}^{n}\left(y_i - \sum_{h=1}^{p} x_{ih}\beta_h\right)^2 + \lambda_1 \sum_{h=1}^{p} |\beta_h| + \lambda_2 \beta^T M \beta \\ &= \sum_{i=1}^{n}\left(y_i - \sum_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j\right)^2 + \lambda_1 \sum_{h=1}^{p} |\beta_h| \\ &\quad + \lambda_2 \sum_{h=1}^{p} M_{hh}\beta_h^2 + \lambda_2 \sum_{u\sim v} 2M_{uv}\beta_u\beta_v.\end{aligned}$$

One wants to minimize over one β_j while keeping all other coefficients fixed at their current values $\tilde{\beta}_k$. For $\beta_j > 0$ and $\tilde{y}_i^{(j)} := \sum_{k\neq j} x_{ik}\tilde{\beta}_k$, the derivative with respect to β_j becomes

$$\frac{\partial}{\partial \beta_j}\mathrm{RSS}(\lambda_1, \lambda_2, \beta) = \sum_{i=1}^{n}(-2x_{ij}(y_i - \tilde{y}_i^{(j)}) + 2x_{ij}^2\beta_j) + \lambda_1 + 2\lambda_2 M_{jj}\beta_j + 2\lambda_2 \sum_{j\neq v} M_{jv}\tilde{\beta}_v.$$

Setting this equal to zero, we get

$$\beta_j = \frac{\sum_{i=1}^{n} 2x_{ij}(y_i - \tilde{y}_i^{(j)}) - 2\lambda_2 \sum_{j\neq v} M_{jv}\tilde{\beta}_v - \lambda_1}{2n + 2\lambda_2 M_{jj}}.$$

We have used $\sum_{i=1}^{n} x_{ij}^2 = n$, which is the case as the input matrix is standardized. The case of $\beta_j < 0$ leads to a similar term.[3]

This finally leads to coordinate updates of the form

$$\tilde{\beta}_j \leftarrow \frac{S(\sum_{i=1}^{n} 2x_{ij}(y_i - \tilde{y}_i^{(j)}) - 2\lambda_2 \sum_{j\neq v} M_{jv}\tilde{\beta}_v, \lambda_1)}{2n + 2\lambda_2 M_{jj}},$$

where $S(\cdot,\cdot)$ is the soft thresholding operator defined by

$$S(x, \kappa) = \mathrm{sign}(x)(|x| - \kappa)_+ = \begin{cases} x - \kappa & \text{if } x > 0 \text{ and } |x| > \kappa, \\ x + \kappa & \text{if } x < 0 \text{ and } |x| > \kappa, \\ 0 & \text{if } |x| \leq \kappa. \end{cases}$$

Covariance updates can lead to a reduction of compute time (we adapt the version proposed in Friedman *et al.* [6]). It is $y_i - \tilde{y}_i^{(j)} = y_i - \tilde{y}_i + x_{ij}\tilde{\beta}_j = r_i + x_{ij}\tilde{\beta}_j$, where $\tilde{y}_i =$

$\sum_{j=1}^{p} x_{ij}\tilde{\beta}$ and r_i is the current residual. Because the input matrix is standardized, it is $\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}) = \sum_{i=1}^{n} x_{ij}r_i + n\tilde{\beta}_j$ and then one can write $\sum_{i=1}^{n} x_{ij}r_i = \langle x_{(j)}, y\rangle - \sum_{k=1}^{p} \langle x_{(j)}, x_{(k)}\rangle \tilde{\beta}_k$, where $x_{(j)}$ is the jth column of the input matrix and $\langle\cdot,\cdot\rangle$ is the inner product. This leads to coordinate updates of the form

$$\tilde{\beta}_j \leftarrow \frac{S(2(n\tilde{\beta}_j + \langle x_{(j)}, y\rangle - \sum_{k=1}^{p} \langle x_{(j)}, x_{(k)}\rangle \tilde{\beta}_k) - 2\lambda_2 \sum_{j\neq v} M_{jv}\tilde{\beta}_v, \lambda_1)}{2n + 2\lambda_2 M_{jj}}.$$

The inner products of y with all columns of the covariate matrix as well as all inner products of two columns of the covariate matrix can then be computed in the beginning and stored. At each coordinate update, they can be accessed.

Note that the coordinate descent algorithm always converges to the global minimum. The proof is contained in the following footnote.[4]

3. Simulations

The motivation for the simulations is of biomedical nature, where network information about gene expression data is of great interest. Such data, which is usually high-dimensional, can be used for prediction purposes (e.g. to predict event times of patients in medical applications), but the signs of the network connections are also of interest in themselves for biomedical research. The simulations that we conduct are similar to the ones reported in Li and Li [9].

3.1. Setting

We consider two similar sets of simulations with four scenarios each. The difference between the two sets is that the first one contains fewer variables, 1100 covariates as compared to 2200 in the second set, while the number of non-zero covariate coefficients is identical in both sets. That is, in the first set, there are relatively more informative covariates. We first describe the first set in detail and briefly mention the small modifications for the second set thereafter.

Note that of the four scenarios in each of the two sets of simulations, the third and fourth scenarios should be seen more as robustness checks. In these scenarios, all connections between covariates are, per definition, positive. These scenarios do thus not promise any potential for the 3CoSE algorithm to do better than the penalty introduced by Li and Li [9], which already assumes positive connections. Nevertheless, it is interesting to see whether the algorithm does significantly worse in case that all connections are indeed positive.

We use the terminology of gene regulation in the description of the simulation scenarios to keep the proximity to Li and Li [9]. However, the simulation scenarios are very general (covariates that are connected on a network, with only a small subset having true non-zero coefficients), so that the motivation of gene regulation is not crucial for the simulation scenarios to be relevant.[5]

3.1.1. Simulations with 1100 covariates (100 transcription factors).

Suppose that information is available about a regulatory network with 100 transcription factors and 1000 genes that are controlled by these transcription factors. Each transcription factor controls ten genes. The resulting network thus consists of 1100 genes and the connections between the transcription factors and the genes that they regulate. The covariate matrix X consists of 1100 columns, where the first column consists of the expression levels of the first transcription factor, the next ten columns are the expression levels of the genes regulated by the transcription factor in the first column, and so on.

The four scenarios differ mainly in the true covariate coefficient vector β. We assume a linear model, i.e. $y_i = x_i^T\beta + \epsilon_i$. The number of observations is 100 and $\epsilon_i \sim N(0, \sigma^2)$ with $\sigma^2 = (\sum_j \beta_j^2)/4$. The expression levels of the 100 transcription factors are independently and identically distributed according to a standard normal distribution. The expression level of a gene depends on the level of the transcription factor that regulates it. The expression level of a gene of observation i that depends on the jth transcription factor ($TF_{i,j}$) follows a $N(\mathbb{1} \cdot 0.7 \cdot TF_{i,j}, 0.51)$ distribution, where $\mathbb{1}$ is the indicator function, which depends on the connection sign and is equal to 1 or -1.

In the first scenario, the true coefficient vector is of the following form:

$$\beta_1 = (5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{3}, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{7}, 5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{3}, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{7},$$
$$5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{3}, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{7}, 5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{3}, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{7}, \underbrace{0, \dots, 0}_{1056}).$$

In the second scenario, the denominators of $\sqrt{10}$ are replaced by 10, keeping all else equal. In these two scenarios, the indicator function $\mathbb{1}$ takes the value -1 for the first three regulated genes of a transcription factor, while it takes the value 1 for the other seven genes.

The coefficient vector in the third scenario is

$$\beta_3 = (5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{10}, -5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{10}, 5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{10},$$
$$-5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{10}, \underbrace{0, \dots, 0}_{1056}).$$

The indicator function $\mathbb{1}$ in this scenario always equals 1 (this is intuitive as the coefficients of the transcription factors and the regulated genes always have the same sign). The fourth scenario is identical to the third one, except that the denominator values of $\sqrt{10}$ are replaced by 10.

3.1.2. Simulations with 2200 covariates (200 transcription factors).

In addition to the first set of four scenarios described above, we also consider very similar scenarios with additional zero-coefficients. That is, instead of considering 100 transcription factors, we consider 200 transcription factors, again each one regulating ten genes. The four scenarios resemble the four previously discussed scenarios but with additional 1100 zero coefficients in the coefficient vector β. The coefficient vector in the first scenario, for example, is then as follows:

$$\beta_1 = (5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{3}, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{7}, 5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{3}, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{7},$$
$$5, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{3}, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{7}, 5, \underbrace{\frac{5}{\sqrt{10}}, \dots, \frac{5}{\sqrt{10}}}_{3}, \underbrace{\frac{-5}{\sqrt{10}}, \dots, \frac{-5}{\sqrt{10}}}_{7}, \underbrace{0, \dots, 0}_{2156}).$$

3.2. Simulation details and definitions

For each scenario, we simulate 50 training data sets and 50 test data sets. In each training set, we select the lasso and the network penalty parameters with 10-fold cross-validation. The regression coefficients and connection signs are then computed using the full training data set and selected penalty parameters. Prediction mean squared errors are then calculated on one full test data set. Note that this means that we evaluate the prediction performance out of sample.

To evaluate the prediction performance, we use the following definitions:

(1) Prediction mean squared error (k here indicates the index of the data set, $k = 1, \dots, T$, and i indicates the observation number, $i = 1, \dots, N$; in our case $T = 50$ and $N = 100$),

$$\mathrm{PMSE}(y_k) = \frac{1}{N}\sum_{i=1}^{N}(y_{k,i} - \hat{y}_{k,i})^2 = \frac{1}{N}\sum_{i=1}^{N}(y_{k,i} - X_{k,i}\hat{\beta}_{k,i|\lambda_1,\lambda_2})^2,$$
$$\mathrm{PMSE}(y) = \frac{1}{T}\sum_{k=1}^{T}(\mathrm{PMSE}(y_k)).$$

(2) Standard errors, denoted by s, for the estimated variable of interest, i.e. $\mathrm{PMSE}(y)$, are computed as follows:

$$s = \frac{1}{T}\sqrt{\sum_{k=1}^{T}[\mathrm{PMSE}(y_k) - \mathrm{PMSE}(y)]^2}.$$

The grid from which the possible values of λ_1 and λ_2 are chosen in the cross validation is the following:

$$\lambda_1 = \{\underbrace{600, \dots, 100}_{\text{in steps of 100}}, \underbrace{99, \dots, 20}_{\text{in steps of 3}}\},$$
$$\lambda_2 = \{\underbrace{0, \dots, 100}_{\text{in steps of 5}}\}.$$

This grid was chosen based on values found in a variety of small model simulations (these models were similar to the scenarios that we consider but with fewer covariates).

In the simulation studies, we compare the following methods. First, the regular 3CoSE algorithm, using the normalized Laplacian as penalty matrix (abbreviated as 3CoSE in the tables). Second, the 3CoSE algorithm using the combinatorial Laplacian as penalty matrix (abbreviated as 3CoSE-CL). Third, penalized regression with a network penalty as in Li and Li [9], which is abbreviated as Net (Li Li) in the tables.[6] Fourth, we use the Lasso as a benchmark model, which incorporates information about the levels of the covariates but not about the network structure. As a comparison, we also partly show a null model ignoring all covariate information (that is, always forecasting the intercept) and the true model (that is, forecasting with the β used to create the data).

3.3. Simulation results with 1100 covariates

Now, we consider the prediction mean square errors (PMSEs), sensitivity and specificity estimates, and the estimates of the fractions of correctly estimated connection signs for the simulations with 1100 covariates (100 transcription factors). Standard errors are always given in parentheses.

The prediction performance of the different methods is given in Table 1. The 3CoSE algorithm with the normalized Laplacian and with the combinatorial Laplacian perform much better than the other methods in the first scenario. In the other three scenarios, there are no large differences in PMSEs. This includes the scenarios with all positive connection signs. Thus, the algorithm seems to be able to improve prediction performance considerably in some cases while not leading to worse performance even in the extreme cases with only positive connection signs.

Measures of sensitivity and specificity can be found in Table 2. The sensitivity shows the fraction of non-zero coefficients that have been correctly estimated to be non-zero, the specificity shows the fraction of zero coefficients that have been correctly estimated to be zero. 3CoSE identifies non-zero coefficients similarly well as Net (Li Li) in most scenarios and considerably better in the first one. The standard algorithm with the normalized Laplacian, in general, performs better than the one with the combinatorial Laplacian, which still performs quite well. Specificity estimates are similar for 3CoSE and Net (Li Li) in all scenarios, while the values are very close to one for the Lasso, which estimates many variables to be zero (reflected in very good specificity but very poor sensitivity).

Furthermore, we compute the fractions of correctly estimated connections signs. We consider the connection sign to be correctly identified if the coefficient estimates of the

Table 1. Prediction mean squared errors (PMSEs).

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso	True	Null
Scenario 1	105.94	124.45	134.72	136.49	53.44	1121.22
	(3.24)	(4.81)	(4.82)	(4.34)	(1.05)	(22.11)
Scenario 2	52.58	51.75	51.75	61.30	28.11	320.79
	(1.45)	(1.38)	(1.38)	(1.97)	(0.56)	(7.17)
Scenario 3	104.18	115.33	100.25	131.88	52.28	1150.07
	(4.13)	(4.13)	(4.25)	(4.26)	(0.93)	(23.46)
Scenario 4	50.78	50.17	51.64	56.85	27.20	315.56
	(1.58)	(1.39)	(1.36)	(1.58)	(0.46)	(6.45)

Table 2. Sensitivity and specificity estimates.

Sensitivity	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Scenario 1	0.93	0.75	0.67	0.52
	(0.02)	(0.03)	(0.02)	(0.01)
Scenario 2	0.47	0.41	0.41	0.26
	(0.03)	(0.01)	(0.01)	(0.01)
Scenario 3	0.94	0.84	0.96	0.52
	(0.02)	(0.03)	(0.02)	(0.01)
Scenario 4	0.46	0.42	0.57	0.27
	(0.02)	(0.02)	(0.04)	(0.01)
Specificity				
Scenario 1	0.94	0.95	0.95	0.99
	(0.0023)	(0.0026)	(0.0025)	(0)
Scenario 2	0.96	0.96	0.96	1
	(0.003)	(0.0029)	(0.0029)	(0)
Scenario 3	0.94	0.94	0.94	0.99
	(0.0026)	(0.0025)	(0.0028)	(0)
Scenario 4	0.96	0.97	0.96	1
	(0.0026)	(0.0025)	(0.0029)	(0)

Table 3. Fractions of correctly estimated connection signs.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Scenario 1	0.95	0.82	0.80	0.66
	(0.03)	(0.05)	(0.06)	(0.07)
Scenario 2	0.62	0.58	0.58	0.48
	(0.07)	(0.07)	(0.07)	(0.07)
Scenario 3	0.94	0.84	0.96	0.52
	(0.03)	(0.05)	(0.03)	(0.07)
Scenario 4	0.46	0.42	0.57	0.27
	(0.07)	(0.07)	(0.07)	(0.06)

two connected covariates are non-zero and either both estimates and both true coefficients have the same signs or if both have different signs (this allows us to also talk about correctly identified connections for Net (Li Li) and the Lasso, although these methods are not intended to estimate network connection signs. As can be seen in Table 3, 3CoSE performs much better than Net (Li Li) in the first scenario, similarly in the second and third and a bit worse in the fourth (remember, however, that scenarios 3 and 4 are designed as favorably to Net (Li Li) as possible as all connection signs are positive). The Lasso does much worse in identifying the connection signs than the other methods across the different scenarios.

The 3CoSE algorithm converges in all simulations and scenarios. In the first scenario, only in 10% of cases, the algorithm needs more than one iteration to converge. In the second scenario, convergence happens after the first iteration in 36% of cases, while in the third and fourth scenarios the algorithm converges after the first iteration in all cases. The selected tuning parameters can be found in the appendix.

3.4. *Simulation results with 2200 covariates*

The results for simulations with 200 transcription factors (and thus 2200 covariates) are similar to the results for the simulations with 100 transcription factors. These results are reported in Tables 4–6 (the selected tuning parameters have again been relegated to the appendix).

Table 4. Prediction mean squared errors (PMSEs), 2200 covariates.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso	True	Null
Scenario 1	123.17	139.28	146.92	144.94	51.59	1140.37
	(6.15)	(5.75)	(6.12)	(6.1)	(0.98)	(27.61)
Scenario 2	52.27	52.03	51.89	57.82	27.32	308.51
	(1.69)	(1.66)	(1.65)	(1.95)	(0.55)	(6.33)
Scenario 3	118.61	131.76	111.98	136.6	51.65	1152.17
	(4.21)	(4.47)	(3.94)	(4.43)	(0.99)	(23.61)
Scenario 4	52.15	50.93	53.18	58.26	27.89	315.07
	(2.06)	(1.99)	(2.07)	(2.75)	(0.61)	(6.59)

Table 5. Sensitivity and specificity estimates, 2200 covariates.

Sensitivity	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Scenario 1	0.86	0.72	0.6	0.48
	(0.03)	(0.03)	(0.02)	(0.01)
Scenario 2	0.40	0.37	0.37	0.25
	(0.02)	(0.01)	(0.01)	(0.01)
Scenario 3	0.84	0.69	0.92	0.49
	(0.03)	(0.03)	(0.02)	(0.01)
Scenario 4	0.44	0.39	0.54	0.27
	(0.03)	(0.01)	(0.04)	(0.01)
Specificity				
Scenario 1	0.97	0.97	0.98	0.99
	(0.0013)	(0.0012)	(0.0011)	(0)
Scenario 2	0.98	0.98	0.98	1
	(0.0014)	(0.0014)	(0.0014)	(0)
Scenario 3	0.97	0.97	0.97	0.99
	(0.0011)	(0.0013)	(0.0012)	(0)
Scenario 4	0.98	0.98	0.97	1
	(0.0015)	(0.0015)	(0.0015)	(0)

Table 6. Fractions of correctly estimated connection signs, 2200 covariates.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Scenario 1	0.91	0.80	0.74	0.63
	(0.04)	(0.06)	(0.06)	(0.07)
Scenario 2	0.59	0.57	0.56	0.48
	(0.07)	(0.07)	(0.07)	(0.07)
Scenario 3	0.84	0.69	0.92	0.49
	(0.05)	(0.07)	(0.04)	(0.07)
Scenario 4	0.44	0.39	0.54	0.27
	(0.07)	(0.07)	(0.07)	(0.06)

In short, in terms of PMSEs, 3CoSE performs again extremely well in the first scenario, while prediction errors are similar to those of Net (Li Li) in the other scenarios. The sensitivity and specificity estimates are comparable to the ones in the simulations with 100 transcription factors, meaning that the sensitivity is considerably better in the first scenario for 3CoSE than for Net (Li Li) while differences are relatively small in the other scenarios. Specificity is similar between the methods. The Lasso again estimates many variables to be zero leading to high specificity but low sensitivity.

Again, in all cases, the algorithm converged. Convergence took place similarly fast as in the set with 1100 covariates suggesting that the number of covariates is not a driving factor in determining the rate of convergence of the algorithm. In the first two scenarios, for example, convergence took more than one iteration (of the connection sign estimation step) in only 16% and 32% of the cases, respectively. It is also worth mentioning here that in both sets of simulations, the algorithm converged after at most 4 iterations. Average selected penalty parameters in the second set of simulations are also similar to the ones in the regressions with 1100 covariates, with slightly lower values for λ_2 in the 2200 covariate setup.

3.5. Improvement of the algorithm over one single estimation step

One may wonder in what cases the algorithm improves over a simple-one step estimation of the connection signs. To show this, we compare our proposed approach of iteratively re-estimating the connection signs with only conducting one iteration of the algorithm (which corresponds to using a simple one-step estimator for the connection signs based on the initial sample covariance). We run small scale simulation studies that are as *Scenario 1* above, with a lower number of zero covariates ($p = 200$) and with different levels of noise.

In the simulations, we vary the signal-to-noise ratio (SNR) by multiplying the standard deviation σ with different values, denoted by κ. The signal-to-noise ratio in (the shortened version of) *Scenario 1* is by definition

$$\mathrm{SNR}_1 = \frac{\sqrt{\|X\beta_1\|_2^2/n}}{\sigma_1},$$

where β_1 and σ_1 are the slope coefficients and the standard deviation of (the shortened) *Scenario 1*. We increase and decrease the standard deviation in the new simulations by κ, that is we set $\sigma = \kappa\sigma_1$, with $\kappa \in \{0.1, 0.2, \ldots, 2\}$, and use σ to generate the data. Therefore, for each κ, the signal-to-noise ratio is

$$\mathrm{SNR} = \frac{\sqrt{\|X\beta_1\|_2^2/n}}{\sigma} = \frac{\sqrt{\|X\beta_1\|_2^2/n}}{\kappa\sigma_1} = \frac{\mathrm{SNR}_1}{\kappa}.$$

For $\kappa = 1$, we retain the same data generating process an in *Scenario 1*, just with a lower number of zero covariates. The signal-to-noise ratio decreases with κ, that is, for lower values of κ, the signal-to-noise ratio is higher than for higher values of κ.

To compare the full 3CoSE algorithm to the version of it using only one iteration, we compute the ratio of the prediction mean squared errors, for different values of κ:

$$\mathrm{PMSE\_ratio}(\kappa) = \frac{\mathrm{PMSE}_\kappa^{\mathrm{3CoSE}}}{\mathrm{PMSE}_\kappa^{\mathrm{one-step}}}. \tag{1}$$

A ratio smaller than 1 indicates that the 3CoSE algorithm has a smaller prediction error. We plot this ratio of the prediction mean squared errors as a function of κ in Figure 1.

The results reveal that the 3CoSE algorithm improves over a single estimation of the covariate coefficients after the initial connection sign estimation for sufficiently high values of the signal-to-noise ratio (corresponding to sufficiently low values of κ). When the signal-to-noise ratio decreases (when κ increases), the difference in prediction performance becomes smaller until it disappears (the full algorithm never leads to worse

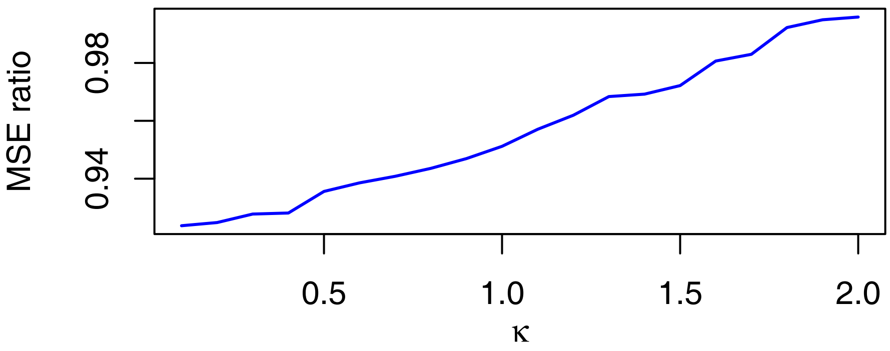

Figure 1. Prediction error ratio – the figure shows the prediction error ratios ratio_κ for a range of $\kappa \in \{0.1, 0.2, \ldots, 2\}$.

performance than the simple one-iteration version in the simulations that we conduct). When the signal-to-noise ratio is higher, the algorithm yields more accurate coefficient sign estimates, thereby improving the estimation of the covariate coefficients and thus obtains superior prediction performance.

4. Data application

We apply the method to time-to-event data from patients with diffuse large B-cell lymphoma [13]. The data contain 240 observations with 7399 microarray features, the number of events is 138. We restrict the analysis forecasting event times to the microarray features represented in regulatory and cancer pathways in the KEGG pathway database. This reduces the number of microarray features to 1281 (however, we do use the additional microarray features for inverse probability weighting, which we employ, because part of the data are censored or truncated). The regulatory network linking the 1281 microarray features contains in total 3645 connections. For details on pre-processing of the data, see Schumacher *et al.* [14] or Binder and Schumacher [1]. The approach we take is to use a linear model with inverse probability weighting to account for censored or truncated data (see, e.g. Wooldridge [22]). This is a straightforward approach that is easy to interpret and implement. Note, however, that this is not the only possible approach to deal with such data, an alternative would, for instance, be to use Cox regressions (for an example with a network penalty making use of Cox regressions, see Sun *et al.* [17]).

4.1. *Inverse probability weighting*

Inverse probability weighting is a method to reweight the data, which can be used, for example, to take into account that data may be censored or truncated (it is thus in spirit similar to using a sample pre-processed with matching techniques, as for example proposed in Ho *et al.* [7]). Inverse probability weighting consists of two parts. First, for each observation, the probability that an event is reported (i.e. that it is neither censored nor truncated) is estimated. Second, the data is restricted to those observations with a reported event, and each observation receives the weight of the inverse estimated probability that an event is reported for this observation (thus observations for which it is likely that relatively many similar observations are censored or truncated receive a higher weight).

We employ the 6119 covariates not used for forecasting to estimate the probability that an event is reported for an observation. This estimation (the first part described in the previous paragraph) consists of two steps. Because the problem is high dimensional, we first perform variable selection. Therefore, we use a Lasso logit regression. In this regression, the dependent variable is the binary variable indicating whether an event was reported for observation or not ($1 =$ yes, $0 =$ no). We use 10-fold cross-validation to determine the penalty parameter that leads to the best prediction. This yields a selection of 24 covariates. However, we do not use the Lasso logit estimates, because of their bias. Instead, we use the selected 24 covariates in the second step to estimate the probabilities with a regular logit regression. We denote the estimated probability (from the second step) that an event is reported for observation i by $\hat{p}_i$.

We then use the inverse probability weights to reweight the data (the second part described in the previous paragraph). Therefore, we restrict the data to those observations for which an event is reported. We reweight each observation i by $\frac{\hat{p}_i^{-1}}{\sum_{k=1}^{N} \hat{p}_k^{-1}}$, where N is the number of observations with an event and $\sum_{k=1}^{N} \hat{p}_k^{-1}$ is a normalization factor.

4.2. Application of the method to the reweighted data

We can now adequately forecast (log) event times with the reweighted data. To be precise, reweighting the data means for the estimation of the penalized log-likelihood that the part of the log-likelihood or the RSS, which reads without any reweighting $(y - X\beta)^{\mathrm{T}}(y - X\beta)$, or equivalently $\sum_{i=1}^{N} (y_i - x_i^T \beta)^2$, becomes

$$\sum_{i=1}^{N} \frac{\hat{p}_i^{-1}}{\sum_{k=1}^{N} \hat{p}_k^{-1}} (y_i - x_i^T \beta)^2.$$

This expression equals $(z - U\beta)^{\mathrm{T}}(z - U\beta)$, where z and U are the variables arising from y and X by multiplying each element of observation i by the square root of the weight, that is by $\sqrt{\hat{p}_i^{-1}/\sum_{k=1}^{N} \hat{p}_k^{-1}}$, for $i = 1, \dots, N$. Thus, with y denoting the vector of log event times of the unweighted data of all observations with an event, the dependent variable that can be used as input in the regular version of the regression or algorithm software z equals $(\sqrt{\hat{p}_1^{-1}/\sum_{k=1}^{N} \hat{p}_k^{-1}}\, y_1, \dots, \sqrt{\hat{p}_N^{-1}/\sum_{k=1}^{N} \hat{p}_k^{-1}}\, y_N)^{\mathrm{T}}$. Similarly, with X denoting the covariate matrix of the unweighted data of all observations with an event, the covariate matrix that can be used as input in the regular version of the regression or algorithm software, U, is constructed by multiplying all elements in row i of X by $\sqrt{\hat{p}_i^{-1}/\sum_{k=1}^{N} \hat{p}_k^{-1}}$, $i = 1, \dots, N$.

Thus, the inverse probability weighting can be implemented with this pre-multiplication of the data, so that z and U can (after standardization) be used as straightforward inputs to the software. It is, therefore, in general not necessary to use different versions of code for the statistical methods we compare.

4.3. Results

We analyze the performance of the different methods employing the 0.632 -bootstrap [4] with 50 bootstrap samples (that is, we draw $0.632n$ observations per sample, which is the

Table 7. Forecast errors (.632 bootstrap), diffuse B-cell lymphoma data.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Error .632 ($\times$100)	0.7302	0.7327	0.7309	0.9116
Bootstrap SEs ($\times$100)	(0.2115)	(0.2119)	(0.2116)	(0.2185)

expected number of unique observations when drawing with replacement). Over the 50 bootstrap samples, the 3CoSE algorithm with the normalized Laplacian selects on average 91 microarray features (out of 1281), 3CoSE-CL selects 202, Net (Li Li) selects 82, while a pure Lasso regression selects only 9 (penalty parameters are again selected via 10-fold cross validation).[7]

Table 7 reports the forecast errors (0.632 -bootstrap estimates of prediction errors), multiplied by 100 for convenience (standard errors are computed over the bootstrap samples and also multiplied by 100). We can see that the three methods taking into account network information improve considerably over the standard Lasso with forecast errors that are about 20 % lower. However, there are hardly any differences between 3CoSE, 3CoSE-CL, and Net (Li Li). 3CoSE has the lowest forecast error, followed by Net (Li Li), but differences between the three network methods are minimal.

On average, both 3CoSE and 3CoSE-CL estimate about 70% of network connections to be positive and 30% to be negative. 3CoSE estimates 2553 connections to be positive and 1092 negative, 3CoSE-CL estimates 2561 positive and 1084 negative connection signs. However, the number of negative connection signs of connections between two covariates that are estimated to be non-zero is much lower. 3CoSE estimates that on average about 4 connections between selected covariates are negative, while 3CoSE-CL estimates that no connections between non-zero covariates are negative.[8]

This may explain why the forecasting performance is so similar between 3CoSE, 3CoSE-CL, and Net (Li Li). While a considerable number of connection signs are estimated to be negative, the regression coefficients of the negatively connected covariates are usually estimated to be zero. The (relatively few) connected selected covariates have almost exclusively connection signs that are estimated to be positive. In that sense, the data resembles the simulations with only positive connections, and we see again that also in such a case the new algorithm performs as well as the method already assuming positive connections in terms of predictive power (while the newly gained information about connection signs can be of interest also for connections between covariates that have an estimated coefficient of zero).

5. Summary

This paper shows that incorporating network information into regularized regression via the introduced 3CoSE algorithm can lead to improved prediction performance. This algorithm can furthermore contribute to discovering signs of network connections when these are unknown. This can be helpful in a variety of fields where networks play an important role, including biology and biomedicine (we have borrowed the motivation of the conducted simulation studies and the microarray data to which we applied the method from these fields), economics, finance, and computer science. We make the accompanying R-package LassoNet [16] publicly available, in the hope that this method will be more widely applied in different fields. The availability of the R-package may also facilitate refining the method to adapt it to different settings.

Notes

1. Other work with double penalty functions includes the elastic net [24] and the sparse-group lasso [15]. There are also more general structured-sparsity inducing penalty functions, see, for instance, [2] and the references therein. See [19] for a comprehensive treatment of structured sparsity regression problems.
2. We assume that a penalty matrix M is symmetric and that we can write $\beta^T M\beta = \sum_{u\sim v}(a(u,v)\beta_u + b(u,v)\beta_v)^2$, with $a(u,v)$ and $b(u,v)$ real numbers. The normalized and combinatorial Laplacians are of such a form.
3. To be precise, for $\beta_j < 0$, we obtain $\beta_j = \frac{\sum_{i=1}^n 2x_{ij}(y_i - \tilde{y}_i^{(j)}) - 2\lambda_2 \sum_{j\neq v} M_{jv}\tilde{\beta}_v + \lambda_1}{2n + 2\lambda_2 M_{jj}}$.
4. The proof makes use of a theorem from [18] stating that the coordinate descent algorithm converges to the global minimum in cases where the function f that shall be minimized is of the form $f(\beta_1,\ldots,\beta_p) = g(\beta_1,\ldots,\beta_p) + \sum_{j=1}^p h_j(\beta_j)$, with g differentiable and convex and $h_j, j = 1,\ldots,p$ convex.
   With a network penalty, the penalized RSS can be written as $\mathrm{RSS}(\beta) = g(\beta) + \sum_{j=1}^p h_j(\beta_j)$, with $g(\beta) = (y - X\beta)^{\mathrm{T}}(y - X\beta) + \lambda_2\beta^T M\beta$ and $h_j(\beta_j) = \lambda_1|\beta_j|$. Then g is a sum of differentiable and convex functions and thus again differentiable and convex, while the h_j are obviously convex.
   In short, if $\beta^T M\beta$ is convex, the coordinate wise descent algorithm is sure to converge to the global minimum. That is in particular the case for all M which allow $\beta^T M\beta$ to be written as a sum of squared terms.
5. One may argue that gene regulation could be depicted by a directed graph. However, we do not consider it problematic if network information that could be represented by a directed graph is entered into the regression setting via an undirected graph (as similarly done in Li and Li [9], among others). Information on the directions (where available) is simply not used in the method, because the second part of the penalty term only penalizes differences between regression coefficients of two connected covariates (sometimes the negative thereof), independently of whether one influences the other or vice versa. Naturally, the method thus cannot be used to discover directions in a covariate network, should the covariates be connected on a directed graph, but the method can be used to improve prediction performance, to select covariates, or to find out about connection signs.
6. We do not apply any double-shrinkage correction, which is mentioned in Li and Li [9]. Unlike in the elastic net [24], where one may indeed talk of double shrinkage, the additional network penalty pulls different coefficients toward each other and not toward zero.
7. The grid of penalty parameters consists of the combinations of $\lambda_1 = \{\underbrace{300,\ldots,100}_{\text{in steps of 25}}, \underbrace{99,\ldots,1}_{\text{in steps of 6}}\}$ and $\lambda_2 = \{\underbrace{0,\ldots,400}_{\text{in steps of 10}}\}$.
8. The selected tuning parameters can again be found in the appendix.

Acknowledgements

We thank seminar participants at the University of Freiburg and the Bank of Lithuania for comments and suggestions. Simulations were conducted on the supercomputer from the High Performance Computing Center of the Lithuanian National Center of Physical and Technology Sciences at Vilnius University. Jonas Striaukas is a Research Fellow of the Fonds de la Recherche Scientifique – FNRS, Belgium. This version supersedes older versions available online as unpublished preprints [20,21].

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by Fonds de la recherche scientifique – FNRS [grant number FC 23188].

References

[1] H. Binder and M. Schumacher, *Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models*, BMC Bioinform. 9 (2008), pp. 14.
[2] X. Chen, Q. Lin, S. Kim, J.G. Carbonell, and E.P. Xing, *Smoothing proximal gradient method for general structured sparse regression*, Ann. Appl. Stat. 6 (2012), pp. 719–752.
[3] F. Chung, *Spectral Graph Theory*, CBMS Regional Conferences Series, Vol. 92, American Mathematical Society, Providence, 1997.
[4] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*, Chapman and Hall, New York, 1993.
[5] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, *Pathwise coordinate optimization*, Ann. Appl. Stat. 1 (2007), pp. 302–332.
[6] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, J. Stat. Softw. 33 (2010), pp. 1.
[7] D.E. Ho, K. Imai, G. King, and E.A. Stuart, *Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference*, Polit. Anal. 15 (2007), pp. 199–236.
[8] L. Jacob, G. Obozinski, and J.P. Vert, *Group lasso with overlap and graph lasso*, in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal, 2009, pp. 433–440.
[9] C. Li and H. Li, *Network-constrained regularization and variable selection for analysis of genomic data*, Bioinformatics 24 (2008), pp. 1175–1182.
[10] C. Li and H. Li, *Variable selection and regression analysis for graph-structured covariates with an application to genomics*, Ann. Appl. Stat. 4 (2010), pp. 1498.
[11] C. Luo, W. Pan, and X. Shen, *A two-step penalized regression method with networked predictors*, Stat. Biosci. 4 (2012), pp. 27–46.
[12] W. Pan, B. Xie, and X. Shen, *Incorporating predictor network in penalized regression with application to microarray data*, Biometrics 66 (2010), pp. 474–484.
[13] A. Rosenwald, G. Wright, W. Chan, J. Connors, E. Campo, R. Fisher, R. Gascoyne, H. Muller-Hermelink, E. Smeland, and L. Staudt, *The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma*, N. Engl. J. Med. 346 (2002), pp. 1937–1947.
[14] M. Schumacher, H. Binder, and T. Gerds, *Assessment of survival prediction models based on microarray data*, Bioinformatics 23 (2007), pp. 1768–1774.
[15] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, *A sparse-group lasso*, J. Comput. Graph. Stat. 22 (2013), pp. 231–245.
[16] J. Striaukas and M. Weber, Lassonet: 3CoSE algorithm, R-package, 2020. Available at https://cran.r-project.org/web/packages/LassoNet/index.html.
[17] H. Sun, W. Lin, R. Feng, and H. Li, *Network-regularized high-dimensional cox regression for analysis of genomic data*, Stat. Sin. 24 (2014), pp. 1433.
[18] P. Tseng, *Coordinate ascent for maximizing nondifferentiable concave functions*, Tech. Rep., Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, 1988.
[19] S.A. van de Geer, *Estimation and Testing Under Sparsity*, Springer, Cham, 2016.
[20] M. Weber, M. Schumacher, H. Binder, *Regularized regression incorporating network information: Simultaneous estimation of covariate coefficients and connection signs*, Tinbergen Institute Discussion Paper 14-089/1, 2014.

[21] M. Weber, J. Striaukas, M. Schumacher, and H. Binder, *Network-constrained covariate coefficient and connection sign estimation*, University of St. Gallen, School of Finance Research Paper 2020/01, 2020.
[22] J.M. Wooldridge, *Inverse probability weighted estimation for general missing data problems*, J. Econom. 141 (2007), pp. 1281–1301.
[23] Y. Zhu, X. Shen, and W. Pan, *Simultaneous grouping pursuit and feature selection over an undirected graph*, J. Am. Stat. Assoc. 108 (2013), pp. 713–725.
[24] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, J. R. Stat. Soc. Ser. B 67 (2005), pp. 301–320.

Appendix. Selected tuning parameters

A.1 Simulations with 1100 covariates

Tables A1 and A2 show the parameters that were selected via cross-validation in the different scenarios for the different methods in the simulations with 1100 covariates. Average selected λ_1 parameters for the first three methods (3CoSE, 3CoSE-CL and Net(Li Li)) are between 83 and 135, while for the lasso it is always 400. The selected network penalty parameter λ_2 varies across methods and simulations in a range from 0 to 71.

A.2 Simulations with 2200 covariates

Tables A3 and A4 similarly show the parameters that were selected via cross-validation in the different scenarios for the different methods in the simulations with 2200 covariates.

A.3 Application to diffuse large B-cell Lymphoma data

Table A5 shows the average selected penalty parameters in the application to real data. The penalty parameters selected by 3CoSE, 3CoSE-CL, and Net (Li Li) are of similar magnitude with a somewhat lower L_1 penalty parameter for 3CoSE-CL but a bit higher network penalty parameter. Pure Lasso

Table A1. Average selected λ_1 values.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Scenario 1	84.36	90.40	99.06	400
Scenario 2	108.30	118.62	118.62	400
Scenario 3	83.30	85.36	89.70	400
Scenario 4	128.60	134.08	109.86	400

Table A2. Average selected λ_2 values.

	3CoSE	3CoSE-CL	Net (Li Li)
Scenario 1	21.30	8.50	3.60
Scenario 2	1.60	0.00	0.00
Scenario 3	27.30	14.70	70.60
Scenario 4	1.70	0.70	19.60

Table A3. Average selected λ_1 values, 2200 covariates.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
Scenario 1	111.82	112.24	146.34	400.00
Scenario 2	109.76	114.88	115.48	400.00
Scenario 3	91.80	119.16	86.18	400.00
Scenario 4	103.52	109.60	91.52	400.00

Table A4. Average selected λ_2 values, 2200 covariates.

	3CoSE	3CoSE-CL	Net (Li Li)
Scenario 1	19.40	6.40	1.80
Scenario 2	1.20	0.10	0.10
Scenario 3	14.60	8.10	56.70
Scenario 4	0.90	0.00	15.30

Table A5. Average selected λ_1 and λ_2 values in the application to diffuse large B-cell lymphoma data.

	3CoSE	3CoSE-CL	Net (Li Li)	Lasso
λ_1	61.96	45.4	67.44	250
λ_2	105.8	113.2	88.4	

has a higher L_1 penalty parameter than the other methods, in line with the observation that Lasso selects fewer covariates.