Sparse group lasso and high dimensional multinomial classification

Martin Vincent *, Niels Richard Hansen

University of Copenhagen, Department of Mathematical Sciences, Universitetsparken 5, 2100 Copenhagen Ø, Denmark

ARTICLE INFO

ABSTRACT

The sparse group lasso optimization problem is solved using a coordinate gradient descent algorithm. The algorithm is applicable to a broad class of convex loss functions. Convergence of the algorithm is established, and the algorithm is used to investigate the performance of the multinomial sparse group lasso classifier. On three different real data examples the multinomial group lasso clearly outperforms multinomial lasso in terms of achieved classification error rate and in terms of including fewer features for the classification. An implementation of the multinomial sparse group lasso algorithm is available in the R package msgl. Its performance scales well with the problem size as illustrated by one of the examples considered—a 50 class classification problem with 10 k features, which amounts to estimating 500 k parameters.

1. Introduction

The sparse group lasso is a regularization method that combines the lasso (Tibshirani, 1994) and the group lasso (Meier et al., 2008). Friedman et al. (2010a) proposed a coordinate descent approach for the sparse group lasso optimization problem. Simon et al. (2013b) used a generalized gradient descent algorithm for the sparse group lasso and considered applications of this method to linear, logistic and Cox regressions. We present a sparse group lasso algorithm suitable for high dimensional problems. This algorithm is applicable to a broad class of convex loss functions. In the algorithm we combine three non-differentiable optimization methods: the coordinate gradient descent (Tseng and Yun, 2009), the block coordinate descent (Tseng, 2001) and a modified coordinate descent method.

Our main application is to multiclass classification based on the multinomial regression model. The lasso penalty has, for some time, been considered as a regularization approach for multinomial regression (Friedman et al., 2010b). The parameters in the multinomial model are, however, naturally structured, with multiple parameters corresponding to one feature, and the lasso penalty does not take this structure into account. To accommodate for this we suggest to add a group lasso term with the parameters corresponding to the same feature grouped together. The resulting penalty is known as the sparse group lasso penalty. We found that using the sparse group lasso penalty for multinomial regression generally improved the performance of the estimated classifier and reduced the number of features included in the model.

The formulation of an efficient and robust sparse group lasso algorithm is not straightforward due to non-differentiability of the penalty. First, the sparse group lasso penalty is not completely separable, which is problematic when using a standard coordinate descent scheme. To obtain a robust algorithm an adjustment is necessary. Our solution, which efficiently treats the singularity at zero that cannot be separated out, is a minor modification of the coordinate descent algorithm. Second, our algorithm is a Newton type algorithm, hence we sequentially optimize penalized quadratic approximations of the loss function. This approach raises another challenge: how to reduce the costs of computing the Hessian? In Section 3.6 we show

---

* Corresponding author. Tel.: +45 22860740.
E-mail addresses: vincent@math.ku.dk, martin.vincent.dk@gmail.com (M. Vincent).

that an upper bound on the Hessian is sufficient to determine whether the minimum over a block of coefficients is attained at zero. This approach enables us to update a large percentage of the blocks without computing the complete Hessian. In this way we reduce the run-time, provided that the upper bound of the Hessian can be computed efficiently. We found that this approach reduces the run-time on large data sets by a factor of more than 2.

Our focus is on applications of the multinomial sparse group lasso to problems with many classes. For this purpose we have investigated three multiclass classification problems. We found that multinomial group lasso and sparse group lasso perform well on these problems. The error rates were substantially lower than the best obtained with multinomial lasso, and the low error rates were achieved for models with fewer features having non-zero coefficients. For example, we consider a text classification problem consisting of Amazon reviews with 50 classes and 10 k textual features. This problem showed a large improvement in the error rates: from approximately 40% for the lasso to less than 20% for the group lasso.

We provide a generic implementation of the sparse group lasso algorithm in the form of a C++ template library. The implementation for multinomial and logistic sparse group lasso regressions is available as an R package. For our implementation the time to compute the sparse group lasso solution is of the same order of magnitude as the time required for the multinomial lasso algorithm as implemented in the R package glmnet. The computation time of our implementation scales well with the problem size.

1.1. Sparse group lasso

Consider a convex, bounded below and twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. We say that $\hat{\beta} \in \mathbb{R}^n$ is a *sparse group lasso minimizer* if it is a solution to the unconstrained convex optimization problem

$$\text{minimize } f + \lambda \Phi \tag{1}$$

where $\Phi : \mathbb{R}^n \to \mathbb{R}$ is the *sparse group lasso penalty* (defined below) and $\lambda > 0$.

Before defining the sparse group lasso penalty some notation is needed. We decompose the search space

$$\mathbb{R}^n = \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$$

into $m \in \mathbb{N}$ blocks having dimensions $n_i \in \mathbb{N}$ for $i = 1, \ldots, m$, hence $n = n_1 + \cdots + n_m$. For a vector $\beta \in \mathbb{R}^n$ we write $\beta = (\beta^{(1)}, \ldots, \beta^{(m)})$ where $\beta^{(1)} \in \mathbb{R}^{n_1}, \ldots, \beta^{(m)} \in \mathbb{R}^{n_m}$. For $J = 1, \ldots, m$ we call $\beta^{(J)}$ the J'th *block* of β. We use the notation $\beta_i^{(J)}$ to denote the i'th coordinate of the J'th block of β, whereas β_i is the i'th coordinate of β.

Definition 1 (*Sparse Group Lasso Penalty*)**.** The sparse group lasso penalty is defined as

$$\Phi(\beta) \stackrel{\text{def}}{=} (1 - \alpha) \sum_{J=1}^{m} \gamma_J \left\| \beta^{(J)} \right\|_2 + \alpha \sum_{i=1}^{n} \xi_i |\beta_i|$$

for $\alpha \in [0, 1]$, group weights $\gamma \in [0, \infty)^m$, and parameter weights $\xi = (\xi^{(1)}, \ldots, \xi^{(m)}) \in [0, \infty)^n$ where $\xi^{(1)} \in [0, \infty)^{n_1}, \ldots, \xi^{(m)} \in [0, \infty)^{n_m}$.

The sparse group lasso penalty includes the lasso penalty ($\alpha = 1$) and the group lasso penalty ($\alpha = 0$). Note also that for sufficiently large values of λ the solution of (1) is zero. The infimum of these, denoted $\lambda_{\max}$, is computable, see Section 3.2.

We emphasize that the sparse group lasso penalty is specified by

- a grouping of the parameters $\beta = (\beta^{(1)}, \ldots, \beta^{(m)})$,
- and the weights α, γ and ξ.

It is well known that the lasso penalty results in sparse solutions to (1), while the group lasso penalty results in groupwise sparse solutions (that is, the entire group of parameters is zero or non-zero). However group lasso does not give sparsity within groups — sparse group lasso does.

In the second part of the paper we develop an algorithm for solving the optimization problem (1). The convergence of the algorithm is established for any sparse group lasso penalty, regardless of how the parameters are grouped. For multinomial regression, as considered in the next section, we restrict attention to a specific grouping of the parameters that reflects the features. In the symmetric parametrization of the multinomial regression model with K classes there are K parameters per feature. Our suggestion is to group these K parameters together. Thus we do not group the features, only the parameters associated with each feature. For the examples we considered this particular grouping resulted in models with fewer features having non-zero parameters compared to ordinary lasso penalization. More importantly, the error rates were typically also smaller.

Our `msgl` R package supports the particular grouping for multinomial regression as well as additional groupings of the features, i.e. the number of parameters in each group is a multiple of K. The `sgl` C++ template library can be configured to handle any grouping.

Table 1
Summary of data sets and the associated classification problem.

Data set	Features	K	N	p
Cancer sites	microRNA expressions	18	162	217
Amazon reviews	Various textual features	50	1500	10 k
Muscle diseases	Gene expression	10	107	22 k

2. The multinomial sparse group lasso classifier

In this section we examine the characteristics of the multinomial sparse group lasso method. Our main interest is the application of the multinomial sparse group lasso classifier to problems with many classes. For this purpose we have chosen three classification problems based on three different data sets, with 10, 18 and 50 classes. In Lu et al. (2005) the microRNA expression profile of different types of primary cancer samples is studied. In Section 2.2.1 we consider the problem of classifying the primary site based on the microRNA profiles in this data set. The Amazon reviews author classification problem, presented in Liu et al. (2011), is studied in Section 2.2.2. The messenger RNA profile of different human muscle diseases is studied in Bakay et al. (2006). We consider, in Section 2.2.3, the problem of classifying the disease based on the messenger RNA profiles in this data set. Table 1 summarizes the dimensions and characteristics of the data sets and the associated classification problems. Finally, in Section 2.3, we examine the characteristics of the method applied to simulated data sets.

2.1. Setup

Consider a classification problem with K classes, N samples, and p features. Assume given a data set $(x_1, y_1), \ldots, (x_N, y_N)$ where, for all $i = 1, \ldots, N$, $x_i \in \mathbb{R}^p$ is the observed feature vector and $y_i \in \{1, \ldots, K\}$ is the categorical response. We organize the feature vectors in the $N \times p$ *design matrix*

$$X \stackrel{\text{def}}{=} (x_1 \cdots x_N)^T.$$

As in Friedman et al. (2010b) we use a symmetric parametrization of the multinomial model. With $h : \{1, \ldots, K\} \times \mathbb{R}^p \rightarrow \mathbb{R}$ given by

$$h(l, \eta) \stackrel{\text{def}}{=} \frac{\exp(\eta_l)}{\sum_{k=1}^{K} \exp(\eta_k)},$$

the multinomial model is specified by

$$P(y_i = l | x_i) = h(l, \beta^{(0)} + \beta x_i).$$

The model parameters are organized in the K-dimensional vector, $\beta^{(0)}$, of intercept parameters together with the $K \times p$ matrix

$$\beta \stackrel{\text{def}}{=} \left(\beta^{(1)} \cdots \beta^{(p)} \right), \tag{2}$$

where $\beta^{(i)} \in \mathbb{R}^K$ are the parameters associated with the i'th feature.

The log-likelihood is

$$\ell(\beta^{(0)}, \beta) = \sum_{i=1}^{N} \log h(y_i, \beta^{(0)} + \beta x_i). \tag{3}$$

Our interest is the sparse group lasso penalized maximum likelihood estimator. That is, $(\beta^{(0)}, \beta)$ is estimated as a minimizer of the sparse group lasso penalized negative-log-likelihood:

$$- \ell(\beta^{(0)}, \beta) + \lambda \left((1 - \alpha) \sum_{J=1}^{p} \gamma_J \left\| \beta^{(J)} \right\|_2 + \alpha \sum_{i=1}^{Kp} \xi_i |\beta_i| \right). \tag{4}$$

In our applications we let $\gamma_J = \sqrt{K}$ for all $J = 1, \ldots, p$ and $\xi_i = 1$ for all $i = 1, \ldots, Kp$, but other choices are possible in the implementation. Note that the parameter grouping, as part of the penalty specification, is given in terms of the columns in (2), i.e. $m = p$.

A common parametrization of the multinomial regression model singles out a reference class, and the probabilities of the other classes are then given relative to the reference class. As pointed out in Kim et al. (2006) this is problematic when lasso penalization is used for parameter estimation, and the symmetric parametrization introduced above, and used in Friedman et al. (2010b) as well, is preferred. It ensures that the resulting estimator is invariant to permutations of the classes. The

parameters in the symmetric parametrization are, however, not identifiable. If β_l denotes the *l*'th row of the matrix β, then for $l, k = 1, \ldots, K$

$$\frac{P(y_i = l|x)}{P(y_i = k|x)} = \exp(\beta_l^{(0)} - \beta_k^{(0)} + (\beta_l - \beta_k)x),$$

and it follows that the differences $\beta_l - \beta_k$ and $\beta_l^{(0)} - \beta_k^{(0)}$ are identifiable. In practice, as was also noted in Section 4.1 in Friedman et al. (2010b), the consequence of the penalization is that the estimated parameters minimize the sparse group lasso penalty among all equivalent parameters. If some parameters, like $\beta^{(0)}$, are not penalized, a procedure like mean centering suggested in Friedman et al. (2010b) can be used to numerically select one of the equivalent parameters.

2.2. Data examples

The data sets were preprocessed before applying the multinomial sparse group lasso estimator. Two preprocessing schemes were used: *normalization* and *standardization*. Normalization is sample centering and scaling in order to obtain a design matrix with row means 0 and row variances 1. Standardization is feature centering and scaling in order to obtain a design matrix with column means 0 and column variances 1. Note that the order in which normalization and standardization are applied matters.

The purpose of normalization is to remove technical (non-biological) variation. A range of different normalization procedures exist for biological data. Sample centering and scaling is one of the simpler procedures. We use this simple normalization procedure for the two biological data sets in this paper. Normalization is done before and independently of the sparse group lasso algorithm.

The purpose of standardization is to create a common scale for the features. This ensures that differences in scale will not influence the penalty and thus the variable selection. Standardization is an option for the sparse group lasso implementation, and it is applied as the last preprocessing step for all three example data sets.

We want to compare the performance of the multinomial sparse group lasso estimator for different values of the regularization parameter α. Applying the multinomial sparse group lasso estimator with a given $\alpha \in [0, 1]$ and λ-sequence, $\lambda_1, \ldots, \lambda_d > 0$, results in a sequence of estimated models with parameters $\{\hat{\beta}(\lambda_i, \alpha)\}_{i=1,\ldots,d}$. The generalization error can be estimated by cross validation (Hastie et al., 2001). For our applications we keep the sample ratio between classes in the cross validation subsets approximately fixed to that of the entire data set. Hence, we may compute a sequence, $\{\widehat{\text{Err}}(\lambda_i, \alpha)\}_{i=1,\ldots,d}$, of estimated expected generalization errors for the sequence of models. However, for given α_1 and α_2 we cannot simply compare $\widehat{\text{Err}}(\lambda_i, \alpha_1)$ and $\widehat{\text{Err}}(\lambda_i, \alpha_2)$, since the λ_i value is scaled differently for different values of α. We will instead compare the models with the same number of non-zero parameters and the same number of non-zero parameter groups, respectively. Define

$$\hat{\Theta}(\lambda, \alpha) \stackrel{\text{def}}{=} \sum_{J=1}^{p} I(\hat{\beta}^{(J)}(\lambda, \alpha) \neq 0)$$

with $\hat{\beta}(\lambda, \alpha)$ the estimator of β for the given values of λ and α. That is, $\hat{\Theta}(\lambda, \alpha)$ is the number of non-zero parameter blocks in the fitted model. Note that there is a one-to-one correspondence between parameter blocks and features in the design matrix. Furthermore, we define the total number of non-zero parameters as

$$\hat{\Pi}(\lambda, \alpha) \stackrel{\text{def}}{=} \sum_{i=1}^{n} I(\hat{\beta}_i(\lambda, \alpha) \neq 0).$$

In particular, we want to compare the fitted models with the same number of parameter blocks. There may, however, be more than one λ-value corresponding to a given value of $\hat{\Theta}$. Thus we compare the models on a subsequence of the λ-sequence. This subsequence is defined below. With $\theta_1 < \cdots < \theta_{d'}$ for $d' \leq d$ denoting the different elements of the set $\{\hat{\Theta}(\lambda_i, \alpha)\}_{i=1,\ldots,d}$ in increasing order we define

$$\tilde{\lambda}_i(\alpha) \stackrel{\text{def}}{=} \min \left\{ \lambda \ \middle| \ \hat{\Theta}(\lambda, \alpha) = \theta_i \right\}.$$

We then compare the characteristics of the multinomial sparse group lasso estimators for different α values by comparing the estimates

$$\left\{ \left(\widehat{\text{Err}}(\tilde{\lambda}_i(\alpha), \alpha), \hat{\Theta}(\tilde{\lambda}_i(\alpha)), \hat{\Pi}(\tilde{\lambda}_i(\alpha)) \right) \right\}_{i=1,\ldots,d'}.$$

2.2.1. Cancer sites

The data set consists of bead-based expression data for 217 microRNAs from normal and cancer tissue samples. The samples are divided into 11 normal classes, 16 tumor classes and 8 tumor cell line classes. For the purpose of this study we select the normal and tumor classes with more than 5 samples. This results in an 18 class data set with 162 samples. The data set is unbalanced, with the number of samples in each class ranging from 5 to 26 and with an average of 9 samples per
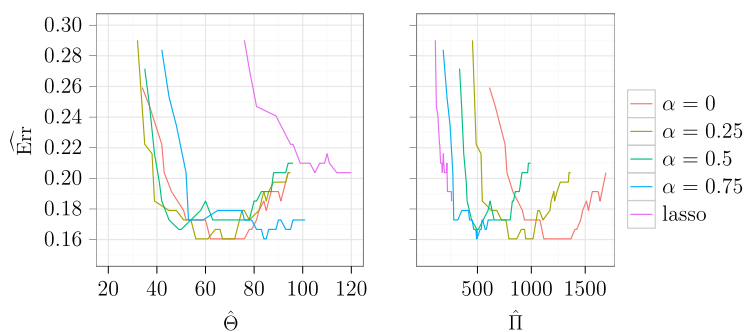
Fig. 1. Estimated expected generalization error, for different values of α, for the microRNA cancer site data set. The cross validation based estimate of the expected misclassification error is plotted against the number of non-zero parameter blocks in the model (left), and against the number of non-zero parameters in the model (right). The estimated standard error is approximately 0.03 for all models.

Fig. 2. Estimated expected generalization error, for different values of α, for the Amazon reviews author classification problem. The cross validation based estimate of the expected misclassification error is plotted against the number of non-zero parameter blocks in the model (left), and against the number of non-zero parameters in the model (right). The estimated standard error is approximately 0.01 for all models.
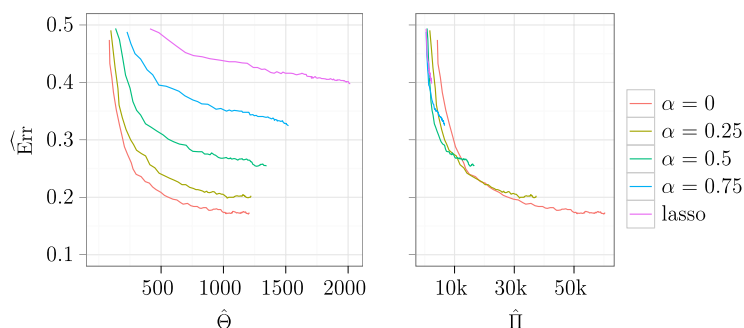
class. Data was normalized and then standardized before running the sparse group lasso algorithm. For more information about this data set see Lu et al. (2005). The data set is available from the Gene Expression Omnibus with accession number GSE2564.

Fig. 1 shows the result of a 10-fold cross validation for 5 different values of α, including the lasso and group lasso. The λ-sequence runs from $\lambda_{\max}$ to 10^{-4}, with $d = 200$. It is evident that the group lasso and sparse group lasso models achieve a lower expected error using fewer genes than the lasso model. However, models with a low α value have a larger number of non-zero parameters than models with a high α value. A reasonable compromise could be the model with $\alpha = 0.25$. This model does not only have a low estimated expected error, but the low error is also achieved with a lower estimated number of non-zero parameters, compared to group lasso.

2.2.2. Amazon reviews

The Amazon review data set consists of 10 k textual features (including lexical, syntactic, idiosyncratic and content features) extracted from 1500 customer reviews from the Amazon Commerce Website. The reviews were collected among the reviews from 50 authors with 50 reviews per author. The primary classification task is to identify the author based on the textual features. The data and feature set were presented in Liu et al. (2011) and can be found in the UCI machine learning repository (Frank and Asuncion, 2010). In Liu et al. (2011) a Synergetic Neural Network is used for author classification, and a 2 k feature based 10-fold CV accuracy of 0.805 is reported. The feature selection and training of the classifier were done separately.

We did 10-fold cross validation using multinomial sparse group lasso for five different values of α. The results are shown in Fig. 2. The λ-sequence runs from $\lambda_{\max}$ to 10^{-4}, with $d = 100$. The design matrix is sparse for this data set. Our implementation of the multinomial sparse group lasso algorithm utilizes the sparse design matrix to gain speed and for memory efficiency. No normalization was applied for this data set. Features were scaled to have variance 1, but were not centered. For this data set it is evident that lasso performs badly, and that the group lasso performs best − in fact much better than lasso. The group lasso achieves an accuracy of around 0.82 with a feature set of size ∼1 k. This outperforms the neural network in Liu et al. (2011).

2.2.3. Muscle diseases

This data set consists of messenger RNA array expression data of 119 muscle biopsies from patients with various muscle diseases. The samples are divided into 13 diagnostic groups. For this study we only consider classes with more than
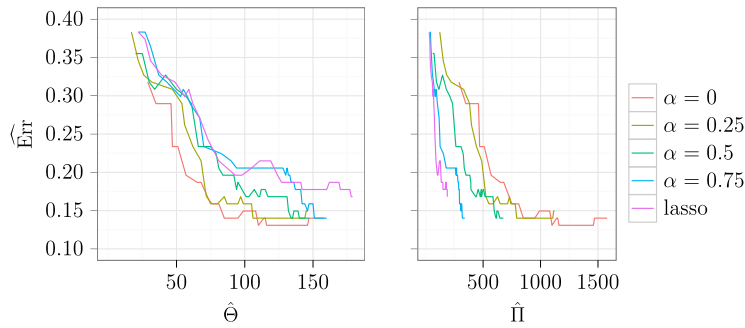
Fig. 3. Estimated expected generalization error, for different values of α, for the muscle disease classification problem. The cross validation based estimate of the expected misclassification error is plotted against the number of non-zero parameter blocks in the model (left), and against the number of non-zero parameters in the model (right). The estimated standard error is approximately 0.04 for all models.

5 samples. This results in a classification problem with 107 samples and 10 classes. The data set is unbalanced with class sizes ranging from 4 to 20 samples per class. Data was normalized and then standardized before running the sparse group lasso algorithm. For background information on this data set, see Bakay et al. (2006). The data set is available from the Gene Expression Omnibus with accession number GDS1956.

The results of a 10-fold cross validation are shown in Fig. 3. The λ-sequence runs from λ_{max} to 10^{-5}, with $d = 200$. We see the same trend as in the other two data examples. Again the group lasso models perform well, but not significantly better than the closest sparse group lasso models ($\alpha = 0.25$). The lasso models perform reasonably well on this data set, but they are still outperformed by the sparse group lasso models.

2.3. A simulation study

In this section we investigate the characteristics of the sparse group lasso estimator on simulated data sets. We are primarily interested in trends in the generalization error as α is varied and $\hat{\lambda}$ is selected by cross validation on a relatively small training set. We suspect that this trend will depend on the distribution of the data. We restrict our attention to multiclass data where the distribution of the features given the class is Gaussian. Loosely speaking, we suspect that if the differences in the data distributions are very sparse, i.e. the centers of the Gaussian distributions are mostly identical across classes, the lasso will produce models with the lowest generalization error. If the data distribution is sparse, but not very sparse, then the optimal α is in the interval $(0, 1)$. For a dense distribution, with center differences between all or most classes, we expect the group lasso to perform best. The simulation study confirms this.

The mathematical formulation is as follows. Let

$$\mu = (\mu_1 \ldots \mu_K)$$

where $\mu_i \in \mathbb{R}^p$ for $i = 1, \ldots, K$ and $p = p_a + p_b$. Denote by $\mathcal{D}_\mu$ a data set consisting of N samples for each of the K classes — each sampled from the Gaussian distribution with centers $\mu_1, \ldots, \mu_K$, respectively, and with a common covariance matrix Σ. Let $\hat{\lambda}$ be the smallest λ-value with the minimal estimated expected generalization error, as determined by cross validation on $\mathcal{D}_\mu$. Denote by $\text{Err}_\mu(\lambda, \alpha)$ the generalization error of the model $\hat{\beta}(\lambda, \alpha)$ that has been estimated from the training set $\mathcal{D}_\mu$, by the sparse group lasso, for the given values of λ and α. Then let

$$Z_\mu(\alpha) = \text{Err}_\mu(\hat{\lambda}, \alpha) - \text{Err}_{\text{Bayes}}(\mu)$$

where $\text{Err}_{\text{Bayes}}(\mu)$ is the Bayes rate. We are interested in trends in Z_μ, as a function of α, for different configurations of $\mu_1, \ldots, \mu_K$. To be specific, we will sample $\mu_1, \ldots, \mu_K$ from one of the following distributions:

- A *sparse* model distribution, where the first p_a entries of μ_i are i.i.d. with a distribution that is a mixture of the uniform distribution on $[-2, 2]$ and the degenerate distribution at 0 with point probability p_0.
- A *dense* model distribution, where the first p_a entries of μ_i are i.i.d. Laplace distributed with location 0 and scale b.

The last p_b entries are zero. We take $p_a = \lfloor 5/(1 - p_0) \rfloor$ throughout for the sparse model distribution. The within class covariance matrix Σ is constructed using features from the cancer site data set. Let Σ_0 be the empirical covariance matrix of p randomly chosen features. To avoid that the covariance matrix becomes singular we take

$$\Sigma = (1 - \delta)\Sigma_0 + \delta\mathbf{I}$$

for $\delta \in (0, 1)$.

The primary quantity of interest is

$$\text{err}(\alpha) \stackrel{\text{def}}{=} \text{E}\left(Z_\mu(\alpha)\right), \tag{5}$$
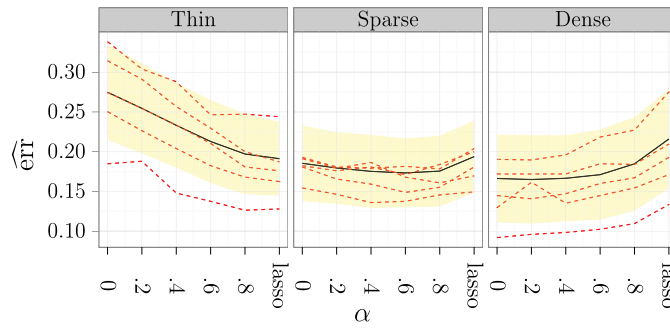
Fig. 4. The estimated expected error gap (solid black line) for the three configurations. The central 95% of the distribution of $Z_\mu(\alpha)$ is shown as the shaded area on the plot. The error gap for 5 randomly selected μ-configurations is shown (red dashed lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 5. The estimated expected true positive rate (solid black line) for the three configurations. The central 95% of the distribution of tpr is shown as the shaded area on the plot. The true positive rate for 5 randomly selected μ-configurations is shown (red dashed lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
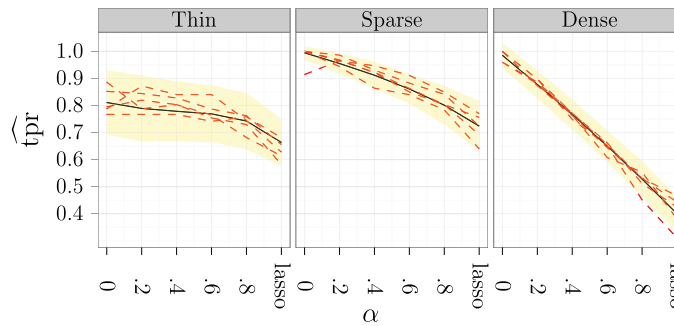
the expectation being over μ and the data set $\mathcal{D}_\mu$. We are also interested in how well we can estimate the non-zero patterns of the μ_i's. Consider this as Kp two class classification problems, one for each parameter, where we predict the μ_{ij} to be non-zero if $\hat{\beta}_{ij}$ is non-zero, and μ_{ij} to be zero otherwise. We calculate the number of false positives, true positives, false negatives and true negatives. The positive predictive value (ppv) and the true positive rate (tpr) are of particular interest. The true positive rate measures how sensitive a given method is at discovering non-zero entries. The positive predictive value measures the precision with which the method is selecting the non-zero entries. We consider the following two quantities

$$\text{tpr}(\alpha) \stackrel{\text{def}}{=} E\left[\text{tpr}\left(\hat{\beta}(\hat{\lambda}, \alpha)\right)\right] \quad \text{and} \quad \text{ppv}(\alpha) \stackrel{\text{def}}{=} E\left[\text{ppv}\left(\hat{\beta}(\hat{\lambda}, \alpha)\right)\right]. \tag{6}$$

In order to estimate the quantities (5) and (6) we sample M configurations of μ from one of the above distributions. For each configuration we sample a training and a test data set of sizes NK and $100K$, respectively. Using the training data set we fit the model $\hat{\beta}(\hat{\lambda}, \alpha)$ and estimate $Z_\mu(\alpha)$ using the test data set. Estimates $\widehat{\text{err}}(\alpha)$, $\widehat{\text{tpr}}(\alpha)$ and $\widehat{\text{ppv}}(\alpha)$ are the corresponding averages over the M configurations.

For this study we chose $M = 100, N = 15, K = 25, p_b = 50, \delta = 0.25$ and the following three configuration distributions:

- *Thin* configurations, where the centers are distributed according to the sparse model distribution with $p_0 = 0.95$, as defined above.
- *Sparse* configurations, where the centers are distributed according to the sparse model distribution with $p_0 = 0.80$.
- *Dense* configurations, where the centers are distributed according to the dense model distribution with scale $b = 0.2$ and $p_a = 25$.

In Fig. 4 we see that for thin configurations the lasso has the lowest estimated error gap, along with the sparse group lasso with $\alpha = 0.8$. For the sparse configurations the results indicate that the optimal choice of α is in the open interval $(0, 1)$, but in this case all choices of α result in a comparable error gap. For the dense configurations the group lasso is among the methods with the lowest error gap.

In Fig. 5 we plotted the true positive rate for the three configurations. Except for the thin configurations, the lasso is markedly less sensitive than the sparse group and group lasso methods. However, looking at Fig. 6 we see that the sparse group and group lasso methods have a lower precision than the lasso, except for the dense configurations. We note that the group lasso has the worst precision, except for the dense configurations.
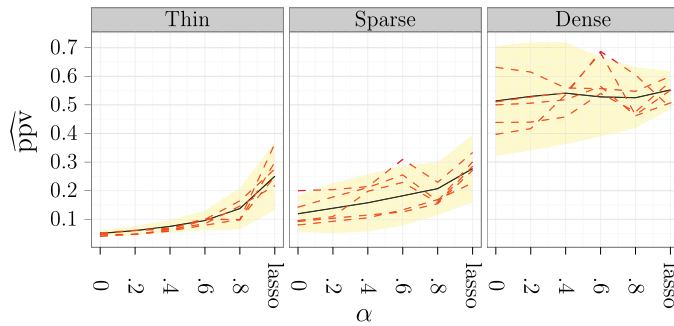
Fig. 6. The estimated expected positive predictive value (solid black line) for the three configurations. The central 95% of the distribution of ppv is shown as the shaded area on the plot. The positive predictive value for 5 randomly selected μ-configurations is shown (red dashed lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3. The sparse group lasso algorithm

In this section we present the sparse group lasso algorithm. The algorithm is applicable to a broad class of loss functions. Specifically, we require that the loss function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, twice continuously differentiable and bounded below. Additionally, we require that all quadratic approximations around a point in the sublevel set

$$\left\{ \beta \in \mathbb{R}^n \mid f(\beta) + \lambda \Phi(\beta) \leq f(\beta_0) + \lambda \Phi(\beta_0) \right\}$$

are bounded below, where $\beta_0 \in \mathbb{R}^n$ is the initial point. The last requirement will ensure that all subproblems are well defined.

The algorithm solves (1) for a decreasing sequence of λ values ranging from $\lambda_{\max}$ to a user specified $\lambda_{\min}$. The algorithm consists of four nested main loops:

- A numerical continuation loop, decreasing λ.
- An outer coordinate gradient descent loop (Algorithm 1).
- A middle block coordinate descent loop (Algorithm 2).
- An inner modified coordinate descent loop (Algorithm 3).

In Sections 3.3–3.5 we discuss the outer, middle and inner loop, respectively. In Section 3.6 we develop a method allowing us to bypass computations of large parts of the Hessian, hereby improving the performance of the middle loop. Section 4 provides a discussion of the available software solutions, as well as run-time performance of the current implementation.

Algorithms for solving the group lasso optimization problem have been around for some time, see, for example, Simil et al. (2007) for an interesting application to multi-response linear regression. The sparse group lasso optimization problem is, however, more complicated, and group lasso algorithms cannot be used to compute a solution to the sparse group lasso optimization problem. Coordinate descent methods still constitute the core of our algorithm, and we give a short review tailored to this paper in Appendix A. See also Tseng and Yun (2009) and Tseng (2001) for further details.

3.1. The sparse group lasso penalty

In this section we derive fundamental results regarding the sparse group lasso penalty.

We first observe that Φ is separable in the sense that if, for any group $J \in 1, \ldots, m$, we define the convex penalty $\Phi^{(J)} : \mathbb{R}^{n_J} \to \mathbb{R}$ by

$$\Phi^{(J)}(\hat{x}) \overset{\text{def}}{=} (1-\alpha)\gamma_J \left\| \hat{x} \right\|_2 + \alpha \sum_{i=1}^{n_J} \xi_i^{(J)} \left| \hat{x}_i \right|$$

then $\Phi(\beta) = \sum_{J=1}^m \Phi^{(J)}(\beta^{(J)})$. Separability of the penalty is required to ensure convergence of coordinate descent methods, see Tseng and Yun (2009), Tseng (2001), and see also Appendix A.

In a block coordinate descent scheme the primary minimization problem is solved by minimizing each block, one at a time, until convergence. We consider conditions ensuring that

$$0 \in \arg\min_{x \in \mathbb{R}^{n_J}} g(x) + \lambda \Phi^{(J)}(x) \tag{7}$$

for a given convex and twice continuously differentiable function $g : \mathbb{R}^{n_J} \to \mathbb{R}$. For $J = 1, \ldots, m$ a straightforward calculation shows that the subdifferential of $\Phi^{(J)}$ at zero is

$$\partial \Phi^{(J)}(0) = (1-\alpha)\gamma_J B^{n_J} + \alpha \operatorname{diag}(\xi^{(J)}) T^{n_J}$$

where $B^n \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \mid \|x\|_2 \le 1\}$, $T^n \stackrel{\text{def}}{=} [-1, 1]^n$ and where for $x \in \mathbb{R}^n$ diag(x) denotes the $n \times n$ diagonal matrix with diagonal x. For an introduction to the theory of subdifferentials see Chapter 4 in Bertsekas et al. (2003).

Proposition 1 below gives a necessary and sufficient condition for (7) to hold. Before we state the proposition the following definition is needed.

Definition 2. For $n \in \mathbb{N}$ we define the map $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ by

$$\kappa(v, z)_i \stackrel{\text{def}}{=} \begin{cases} 0 & |z_i| \le v_i \\ z_i - \text{sgn}(z_i)v_i & \text{otherwise} \end{cases} \quad \text{for } i = 1, \dots, n$$

and the function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ by

$$K(v, z) \stackrel{\text{def}}{=} \|\kappa(v, z)\|_2^2 = \sum_{\{i \mid |z_i| > v_i\}} (z_i - \text{sgn}(z_i)v_i)^2 .$$

Proposition 1. *Assume given $a > 0$, $v, z \in \mathbb{R}^n$ and define the closed sets*

$$Y = z + \text{diag}(v)T_n \quad and \quad X = aB^n + Y.$$

Then the following hold:

a. $\kappa(v, z) = \arg\min_{y \in Y} \|y\|_2$.
b. $0 \in X$ *if and only if* $K(v, z) \le a^2$.
c. *If* $K(v, z) > a^2$ *then* $\arg\min_{x \in X} \|x\|_2 = \left(1 - a/\sqrt{K(v, z)}\right) \kappa(v, z)$.

The proof of Proposition 1 is given in Appendix C. Proposition 1 implies that (7) holds if and only if

$$\sqrt{K(\lambda\alpha\xi^{(J)}, \nabla g(0))} \le \lambda(1 - \alpha)\gamma_J.$$

The following observations will prove to be valuable. Note that we use $\preceq$ to denote coordinatewise ordering.

Lemma 1. *For any three vectors $v, z, z' \in \mathbb{R}^n$ the following hold:*

a. $K(v, z) = K(v, |z|)$.
b. $K(v, z) \le K(v, z')$ *when* $|z| \preceq |z'|$.

Proof. (a) is a simple calculation and (b) is a consequence of the definition and (a). $\square$

3.2. The λ-sequence

For sufficiently large λ values the only solution to (1) will be zero. We denote the infimum of these by $\lambda_{\max}$. By using the above observations it is clear that

$$\begin{aligned}
\lambda_{\max} &\stackrel{\text{def}}{=} \inf\left\{\lambda > 0 \mid \hat{\beta}(\lambda) = 0\right\} \\
&= \inf\left\{\lambda > 0 \mid \forall J = 1, \dots, m : \sqrt{K(\lambda\alpha\xi^{(J)}, \nabla f(0)^{(J)})} \le \lambda(1 - \alpha)\gamma_J\right\} \\
&= \max_{J=1,\dots,m} \inf\left\{\lambda > 0 \mid \sqrt{K(\lambda\alpha\xi^{(J)}, \nabla f(0)^{(J)})} \le \lambda(1 - \alpha)\gamma_J\right\}.
\end{aligned}$$

It is possible to compute

$$\inf\left\{\lambda > 0 \mid \sqrt{K(\lambda\alpha\xi^{(J)}, \nabla f(0)^{(J)})} \le \lambda(1 - \alpha)\gamma_J\right\}$$

by using the fact that the function $\lambda \to K(\lambda\alpha\xi^{(J)}, \nabla f(0)^{(J)})$ is piecewise quadratic and monotone.

3.3. Outer loop

In the outer loop a coordinate gradient descent scheme is used. In this paper we use the simplest form of this scheme. In this simple form the coordinate gradient descent method is similar to Newton's method; however the important difference is the way the non-differentiable penalty is handled. The convergence of the coordinate gradient descent method is not trivial and is established in Tseng and Yun (2009).

The algorithm is based on a quadratic approximation of the loss function f, at the current estimate of the minimizer. The difference, Δ, between the minimizer of the penalized quadratic approximation and the current estimate is then a descent direction. A new estimate of the minimizer of the objective is found by applying a line search in the direction of Δ. We repeat this until a stopping condition is met, see Algorithm 1. Note that a line search is necessary in order to ensure global convergence. For most iterations, however, $t = 1$ will give sufficient decrease in the objective. With $q = \nabla f(\beta)$ and $H = \nabla^2 f(\beta)$ the quadratic approximation of f around the current estimate, β, is

$$q^T(x - \beta) + \frac{1}{2}(x - \beta)^T H(x - \beta) = q^T x - q^T \beta + \frac{1}{2}x^T Hx - \frac{1}{2}\left(\beta^T Hx + x^T H\beta\right) + \frac{1}{2}\beta^T H\beta.$$

H is symmetric, thus it follows that the quadratic approximation of f around β equals

$$Q(x) - q^T\beta + \frac{1}{2}\beta^T H\beta,$$

where $Q : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$Q(x) \stackrel{\text{def}}{=} (q - H\beta)^T x + \frac{1}{2}x^T Hx.$$

We have reduced problem (1) to the following penalized quadratic optimization problem

$$\min_{x \in \mathbb{R}^n} Q(x) + \lambda \Phi(x). \tag{8}$$

Algorithm 1 Outer loop. Solve (1) by coordinate gradient descent.

Require: $\beta = \beta_0$
  repeat
    Let $q = \nabla f(\beta)$, $H = \nabla^2 f(\beta)$ and $Q(x) = (q - H\beta)^T x + \frac{1}{2}x^T Hx$.
    Compute $\hat{\beta} = \arg\min\limits_{x \in \mathbb{R}^n} Q(x) + \lambda \Phi(x)$.
    Compute step size t and set $\beta = \beta + t\Delta$, for $\Delta = \beta - \hat{\beta}$.
  until stopping condition is met.

The convergence of Algorithm 1 is implied by Theorem 1e in Tseng and Yun (2009). This implies:

Proposition 2. *Every cluster point of the sequence $\{\beta_k\}_{k \in \mathbb{N}}$ generated by Algorithm 1 is a solution of problem (1).*

Remark 1. The convergence of Algorithm 1 is ensured even if H is a (symmetric) positive definite matrix approximating $\nabla^2 f(\beta)$. For high dimensional problems it might be computationally beneficial to take H to be diagonal, e.g. as the diagonal of $\nabla^2 f(\beta)$.

3.4. Middle loop

In the middle loop the penalized quadratic optimization problem (8) is solved. The penalty Φ is block separable, i.e.

$$Q(x) + \lambda \Phi(x) = Q(x) + \lambda \sum_{J=0}^{p} \Phi^{(J)}(x^{(J)})$$

with $\Phi^{(J)}$ convex, and we can therefore use the block coordinate descent method over the blocks $x^{(1)}, \ldots, x^{(m)}$. The block coordinate descent method will converge to a minimizer even for non-differentiable objectives if the non-differentiable parts are block separable, see Tseng (2001). Since Φ is separable and Q is convex, twice continuously differentiable and bounded below, the block coordinate descent scheme converges to the minimizer of problem (8). Hence, our problem is reduced to the following collection of problems, one for each $J = 1, \ldots, m$,

$$\min_{\hat{x} \in \mathbb{R}^{n_J}} Q^{(J)}(\hat{x}) + \lambda \Phi^{(J)}(\hat{x}) \tag{9}$$

where $Q^{(J)} : \mathbb{R}^{n_J} \to \mathbb{R}$ is the quadratic function

$$\hat{x} \to Q(x^{(1)}, \ldots, x^{(J-1)}, \hat{x}, x^{(J+1)}, \ldots, x^{(m)})$$

up to an additive constant. We decompose an $n \times n$ matrix H into block matrices in the following way

$$H = \begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1m} \\ H_{21} & H_{22} & \cdots & H_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ H_{m1} & H_{m2} & \cdots & H_{mm} \end{pmatrix}$$

where H_{IJ} is an $n_I \times n_J$ matrix. By the symmetry of H it follows that

$$Q^{(J)}(\hat{x}) = \hat{x}^T (q - H\beta)^{(J)} + \frac{1}{2} \left(2 \sum_I \hat{x}^T H_{JI} x^{(I)} - \hat{x}^T H_{JJ} x^{(J)} + \hat{x}^T H_{JJ} \hat{x} \right)$$

$$= \hat{x}^T \left(q^{(J)} + [H(x - \beta)]^{(J)} - H_{JJ} x^{(J)} \right) + \frac{1}{2} \hat{x}^T H_{JJ} \hat{x}$$

up to an additive constant. We may, therefore, redefine

$$Q^{(J)}(\hat{x}) \overset{\text{def}}{=} \hat{x}^T g^{(J)} + \frac{1}{2} \hat{x}^T H_{JJ} \hat{x}$$

where the *block gradient* $g^{(J)}$ is defined by

$$g^{(J)} \overset{\text{def}}{=} q^{(J)} + [H(x - \beta)]^{(J)} - H_{JJ} x^{(J)}. \tag{10}$$

For the collection of problems given by (9) a considerable fraction of the minimizers will be zero in practice. By Lemma 1 this is the case if and only if

$$\sqrt{K(\lambda \alpha \xi^{(J)}, g^{(J)})} \leq \lambda (1 - \alpha) \gamma_J.$$

These considerations lead us to Algorithm 2.

Algorithm 2 Middle loop. Solve (8) by block coordinate descent.

---

repeat
    Choose next block index J according to the cyclic rule.
    Compute the block gradient $g^{(J)}$.
    if $\sqrt{K(\lambda \alpha \xi^{(J)}, g^{(J)})} \leq \lambda (1 - \alpha) \gamma_J$ **then**
        Let $x^{(J)} = 0$.
    else
        Let $x^{(J)} = \underset{\hat{x} \in \mathbb{R}^{n_J}}{\arg \min} \, Q^{(J)}(\hat{x}) + \lambda \Phi^{(J)}(\hat{x})$.
    end if
until stopping condition is met.

3.5. Inner loop

Finally we need to determine the minimizer of (9), i.e. the minimizer of

$$\underbrace{Q^{(J)}(\hat{x}) + \lambda (1 - \alpha) \gamma_J \left\| \hat{x} \right\|_2}_{\text{loss}} + \underbrace{\lambda \alpha \sum_{i=0}^{n_J} \xi_i^{(J)} \left| \hat{x}_i \right|}_{\text{penalty}}. \tag{11}$$

The two first terms of (11) are considered the loss function and the last term is the penalty. Note that the loss is not differentiable at zero (due to the L_2-norm), thus we cannot completely separate out the non-differentiable parts. This implies that ordinary block coordinate descent is not guaranteed to converge to a minimizer. Algorithm 3 adjusts for this problem, and we have the following proposition.

Proposition 3. *For any $\epsilon > 0$ the cluster points of the sequence $\{\hat{x}_k\}_{k \in \mathbb{N}}$ generated by Algorithm 3 are minimizers of* (11).

Proof. Since $Q^{(J)}(0) + \lambda \Phi^{(J)}(0) = 0$ Algorithm 3 is a modified block coordinate descent scheme. Furthermore J is chosen such that (11) is not optimal at 0. We can therefore apply Lemma 4 in Appendix B, from which the claim follows directly.   □

Hence, for a given block $J = 1, \ldots, m$ we need to solve the following two problems:

I. For each $j = 1, \ldots, n_J$, compute a minimizer for the function

$$\mathbb{R} \ni \hat{x} \to Q^{(J)}(x^{(J)}, \ldots, x_{j-1}^{(J)}, \hat{x}, x_{j+1}^{(J)}, \ldots, x_{n_J}^{(J)}) + \lambda \Phi^{(J)}(x^{(J)}, \ldots, x_{j-1}^{(J)}, \hat{x}, x_{j+1}^{(J)}, \ldots, x_{n_J}^{(J)}).$$

II. Compute a descent direction at zero for (11).

Regarding I. Writing out the equation we see that in the j'th iteration we need to find the minimizer of the function $\omega : \mathbb{R} \to \mathbb{R}$ given by

$$\omega(\hat{x}) \overset{\text{def}}{=} c\hat{x} + \frac{1}{2}h\hat{x}^2 + \gamma\sqrt{\hat{x}^2 + r} + \xi\left|\hat{x}\right| \tag{12}$$

with $c = g_j^{(J)} + \sum_{i \neq j}(H_{JJ})_{ji}x_i$, $\gamma = \lambda(1-\alpha)\gamma_J$, $\xi = \lambda\alpha\xi_j^{(J)}$, $r = \sum_{i \neq j}x_i^2$, and where h is the j'th diagonal of the Hessian block H_{JJ}.

By convexity of f we conclude that $h \geq 0$. Lemma 2 below deals with the case $h > 0$. Since the quadratic approximation Q is bounded below the case $h = 0$ implies that $c = 0$, hence for $h = 0$ we have $\hat{x} = 0$.

Lemma 2. *If $h > 0$ then the minimizer $\hat{x}$ of ω is given as follows:*

a. *If $r = 0$ or $\gamma = 0$ then*

$$\hat{x} = \begin{cases} \dfrac{\xi + \gamma - c}{h} & \text{if } c > \xi + \gamma \\ 0 & \text{if } |c| \leq \xi + \gamma \\ \dfrac{-\xi - \gamma - c}{h} & \text{if } c < -\xi - \gamma. \end{cases}$$

b. *If $r > 0$, $\gamma > 0$ then $\hat{x} = 0$ if $|c| \leq \xi$ and otherwise the solution to*

$$c + \text{sgn}(\xi - c)\xi + h\hat{x} + \gamma\frac{\hat{x}}{\sqrt{\hat{x}^2 + r}} = 0.$$

Proof. Simple calculations will show the results. □

For case (b) in the above lemma we solve the equation by applying a standard root finding method.

Regarding II. For a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and a point $x \in \mathbb{R}^n$, the vector

$$\Delta = -\underset{\hat{x} \in \partial f(x)}{\arg\min}\left\|\hat{x}\right\|_2$$

is a descent direction at x provided f is not optimal at x, see Bertsekas et al. (2003, Section 8.4). We may use this fact to compute a descent direction at zero for the function (11). By Proposition 1 it follows that $\Delta \in \mathbb{R}^n$ defined by

$$\Delta_i \overset{\text{def}}{=} -\begin{cases} 0 & \left|g_i^{(J)}\right| \leq \lambda\alpha\xi_i^{(J)} \\ g_i^{(J)} - \lambda\alpha\xi_i^{(J)}\text{sgn}(g_i^{(J)}) & \text{otherwise} \end{cases}$$

is a descent direction at zero for the function (11).

Algorithm 3 Inner loop. Compute the minimizer of (11) by a modified coordinate descent scheme.

  repeat
    Choose next parameter index j according to the cyclic rule.
    Compute
$$x_j^{(J)} = \underset{\hat{x} \in \mathbb{R}}{\arg\min}\ Q^{(J)}(x_1^{(J)}, \ldots, x_{j-1}^{(J)}, \hat{x}, x_{j+1}^{(J)}, \ldots, x_{n_J}^{(J)})$$
$$+ \lambda\Phi^{(J)}(x^{(J)}, \ldots, x_{j-1}^{(J)}, \hat{x}, x_{j+1}^{(J)}, \ldots, x_{n_J}^{(J)})$$

    if $\left\|x^{(J)}\right\|_2 < \epsilon$ and $Q^{(J)}(x^{(J)}) + \lambda\Phi^{(J)}(x^{(J)}) \geq 0$ **then**
      Compute a descent direction, Δ, at zero for (11).
      Use line search to find t such that $Q^{(J)}(t\Delta) + \lambda\Phi^{(J)}(t\Delta) < 0$.
      Let $x^{(J)} = t\Delta$
    end if
  until stopping condition is met.

3.6. Hessian upper bound optimization

In this section we present a way of reducing the number of blocks for which the block gradient needs to be computed. The aim is to reduce the computational costs of the algorithm.

Table 2
Timing the Hessian bound optimization scheme.

Data set	n	m	Ratio
Cancer	3.9 k	217	1.14
Amazon	500 k	10 k	1.76
Muscle	220 k	22 k	2.47

In the middle loop, Algorithm 2, the block gradient (10) is computed for all m blocks. We shall demonstrate that it is not necessary to compute the block gradient in order to determine if a block is zero, but that an upper bound of the block gradient is sufficient. Since the gradient, q, is already computed we focus on the term involving the Hessian. That is, for $J = 1, \ldots, m$, we compute a $b_J \in \mathbb{R}$ such that

$$\left| [H(x - \beta)]^{(J)} \right| \preceq b_J D_{n_J}$$

where $D_n \overset{\text{def}}{=} (1, 1, \ldots, 1) \in \mathbb{R}^n$. We define

$$t_J \overset{\text{def}}{=} \sup \left\{ x \geq 0 \ \middle| \ \sqrt{K_J \left(\lambda \alpha \xi^{(J)}, \left| q^{(J)} \right| + x D_{n_J} \right)} \leq \lambda(1 - \alpha)\gamma_J \right\}$$

when $\sqrt{K_J \left(\lambda \alpha \xi^{(J)}, \left| q^{(J)} \right| \right)} \leq \lambda(1 - \alpha)\gamma_J$ and otherwise let $t_J = 0$. When $b_J < t_J$ it follows by Lemma 1 that

$$K_J(\lambda \alpha \xi^{(J)}, g^{(J)}) = K_J \left(\lambda \alpha \xi^{(J)}, \left| g^{(J)} \right| \right)$$

$$\leq K_J \left(\lambda \alpha \xi^{(J)}, \left| q^{(J)} \right| + b_J D_{n_J} \right)$$

$$\leq \lambda^2 (1 - \alpha)^2 \gamma_J^2$$

and by Proposition 1 this implies that the block J is zero. The above considerations lead us to Algorithm 4. Note that it is possible to compute the t_J's by using the fact that function

$$\mathbb{R} \ni x \rightarrow K_J \left(\lambda \alpha \xi^{(J)}, \left| q^{(J)} \right| + x D_{n_J} \right)$$

is monotone and piecewise quadratic.

Algorithm 4 Middle loop with Hessian bound optimization.

---

repeat
    Choose next block index J according to the cyclic rule.
    Compute upper bound b_J.
    if $b_J < t_J$ **then**
        Let $x^{(J)} = 0$.
    else
        Compute $g^{(J)}$ and compute new $x^{(J)}$ (see Algorithm 2).
    end if
until stopping condition is met.

In Algorithm 4 it is unnecessary to compute the block gradient for all blocks, but only for those where $x^{(J)} \neq 0$ or when $b_j < t_J$. This will only be beneficial if we can efficiently compute a sufficiently good bound b_J. For a broad class of loss functions this can be done using the Cauchy–Schwarz inequality.

To assess the performance of the Hessian bound scheme we used our multinomial sparse group lasso implementation with and without bound optimization (and with $\alpha = 0.5$). Table 2 lists the ratio of the run-time without using bound optimization to the run-time with bound optimization, on the three different data sets. The Hessian bound scheme decreases the run-time for the multinomial loss function, and the ratio increases with the number of blocks m in the data set. The same trend can be seen for other values of α.

4. Software

We provide two software solutions in relation to the current paper. An R package, `msgl`, with a relatively simple interface to our multinomial and logistic sparse group lasso regression routines. In addition, a C++ template library, `sgl`, is provided. The `sgl` template library gives access to the generic sparse group lasso routines. The R package relies on this library. The `sgl` template library relies on several external libraries. We use the Armadillo C++ library (Sanderson, 2010) as our primary linear algebra engine. Armadillo is a C++ template library using expression template techniques to optimize the performance of matrix expressions, see Eddelbuettel and Sanderson (2013). Furthermore we utilize several Boost libraries (Boost, 2012).

Table 3
Times for computing the multinomial sparse group lasso regression solutions for a lambda sequence of length 100, on a 2.20 GHz Intel Core i7 processor (using one thread). In all cases the sequence runs from $\lambda_{\max}$ to 0.002. The number of samples in the data sets Cancer, Muscle and Amazon are respectively 162, 107 and 1500. See also Table 1 and the discussions in Sections 2.2.1, 2.2.3 and 2.2.2 respectively.

Data set	n (k)	m	Lasso (s)	Sparse group lasso		Group lasso (s)
				$\alpha = 0.75$ (s)	$\alpha = 0.25$ (s)	
Cancer	3.9	217	5.9	4.8	6.3	6.0
Muscle	220	22 k	25.0	25.8	37.7	36.7
Amazon	500	10 k	331.6	246.7	480.4	285.1

Boost is a collection of free peer-reviewed C++ libraries, many of which are template libraries. For an introduction to these libraries see for example Demming and Duffy (2010). Use of multiple processors for cross validation and subsampling is supported through OpenMP (Openmp, 2011).

The `msgl` R package is available from CRAN. The `sgl` library is available upon request.

4.1. Run-time performance

Table 3 lists run-times of the current multinomial sparse group lasso implementation for three real data examples. For comparison, the `glmnet` uses 5.2 s, 8.3 s and 137.0 s, respectively, to fit the lasso path for the three data sets in Table 3. The `glmnet` is a fast implementation of the coordinate descent algorithm for fitting generalized linear models with the lasso penalty or the elastic net penalty (Friedman et al., 2010b). Recently, support for multinomial group lasso has been added to `glmnet`, see Simon et al. (2013a). However, `glmnet` cannot be used to fit models with the sparse group lasso penalty.

5. Conclusion

We developed an algorithm for solving the sparse group lasso optimization problem with a general convex loss function. Furthermore, convergence of the algorithm was established in a general framework. This framework includes the sparse group lasso penalized negative-log-likelihood for the multinomial model, which is of primary interest for multiclass classification problems.

We implemented the algorithm as a C++ template library. An R package is available for the multinomial and the logistic regression loss functions. We presented applications to multiclass classification problems using three real data examples. The multinomial group lasso solution achieved optimal performance in all three examples in terms of estimated expected misclassification error. In one example some sparse group lasso solutions achieved comparable performance based on fewer features. If there is a cost associated with the acquisition of each feature, this could be beneficial if we want to minimize the cost while optimizing the classification performance. In general, the sparse group lasso solutions provide more sparse solutions than the group lasso. Sparsity is generally of interest for model selection purposes and for interpretation of the model.

Appendix A. Block coordinate descent methods

In this section we review the theoretical basis of the optimization methods that we apply in the sparse group lasso algorithm. We use three slightly different methods: a coordinate gradient descent, a block coordinate descent and a modified block coordinate descent.

We are interested in unconstrained optimization problems on $\mathbb{R}^n$ where the coordinates are naturally divided into $m \in \mathbb{N}$ blocks with dimensions $n_i \in \mathbb{N}$ for $i = 1, \ldots, m$. We decompose the search space

$$\mathbb{R}^n = \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m}$$

and denote by P_i the orthogonal projection onto the i'th block. For a vector $x \in \mathbb{R}^n$ we write $x = (x^{(1)}, \ldots, x^{(m)})$ where $x^{(1)} \in \mathbb{R}^{n_1}, \ldots, x^{(m)} \in \mathbb{R}^{n_m}$. For $i = 1, \ldots, m$ we call $x^{(i)}$ the i'th *block* of x. We assume that the objective function $F : \mathbb{R}^n \to \mathbb{R}$ is bounded below and of the form

$$F(x) = f(x) + \sum_{i=1}^{m} h_i(x^{(i)})$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and each $h_i : \mathbb{R}^{n_i} \to \mathbb{R}$, for $i = 1, \ldots, m$ are convex. Furthermore, we assume that for any $i = 1, \ldots, m$ and any $x_0 = (x_0^{(1)}, \ldots, x_0^{(m)})$ the function

$$\mathbb{R}^{n_i} \ni \hat{x} \to F(x_0^{(1)}, \ldots, x_0^{(i-1)}, \hat{x}, x_0^{(i+1)}, \ldots, x_0^{(m)})$$

is hemivariate. A function is said to be *hemivariate* if it is not constant on any line segment of its domain.

Algorithm 5 Coordinate gradient descent scheme.

---

repeat

    Compute quadratic approximation Q of f around the current point x.

    Compute search direction

$$x^{new} = \underset{\hat{x} \in \mathbb{R}^n}{\arg \min} \, Q(\hat{x}) + \sum_{i=1}^{m} h_i\left(\hat{x}^{(i)}\right).$$

    Let $\Delta = x - x^{new}$ and compute step size t using the Armijo rule and let $x \leftarrow x + t\Delta$.

until stopping condition is met.

Algorithm 6 Armijo rule.

Require: $a \in (0, 0.5)$ and $b \in (0, 1)$

    Let $\delta = \nabla f(x)^T \Delta + \sum_{i=1}^{m} (h_i(x_i + \Delta_i) - h_i(x_i))$.

    while $F(x + t\Delta) > F(x) + ta\delta$ **do**

        $t \leftarrow bt$.

    end while

A.1. Coordinate gradient descent

For this scheme we make the additional assumption that f is twice continuously differentiable everywhere. The scheme is outlined in Algorithm 5, where the step size is chosen by the Armijo rule outlined in Algorithm 6. Theorem 1e in Tseng and Yun (2009) implies the following:

Corollary 1. *If f is twice continuously differentiable then every cluster point of the sequence $\{x_k\}_{k \in \mathbb{N}}$ generated by Algorithm 5 is a minimizer of F.*

A.2. Block coordinate descent

Algorithm 7 Block coordinate descent.

---

repeat

    Choose next block index i according to the cyclic rule.

    $x^{(i)} \leftarrow \underset{\hat{x} \in \mathbb{R}^{n_i}}{\arg \min} \, F(\hat{x} \oplus P_i^{\perp} x)$.

until some stopping condition is met.

The block coordinate descent scheme is outlined in Algorithm 7. By Corollary 2 below the block coordinate descent method converges to a *coordinatewise minimum*.

Definition 3. A point $p \in \mathbb{R}^n$ is said to be a coordinatewise minimizer of F if for each block $i = 1, \ldots, m$ it holds that

$$F(p + (0, \ldots, 0, d_i, 0, \ldots, 0)) \geq F(p) \quad \text{for all } d_i \in \mathbb{R}^{n_i}.$$

If f is differentiable then by Lemma 3 the block coordinate descent method converges to a minimizer. Lemma 3 below is a simple consequence of the separability of F.

Lemma 3. *Let $p \in \mathbb{R}^n$ be a coordinatewise minimizer of F. If f is differentiable at p then p is a stationary point of F.*

Proposition 5.1 in Tseng (2001) implies the following:

Corollary 2. *For the sequence $\{x_k\}_{k \in \mathbb{N}}$ generated by the block coordinate descent algorithm (Algorithm 7) it holds that every cluster point of $\{x_k\}_{k \in \mathbb{N}}$ is a coordinatewise minimizer of F.*

Appendix B. Modified block coordinate descent

For this last scheme we make the additional assumption that f is twice continuously differentiable everywhere except at a given non-optimal point $p \in \mathbb{R}^n$. In this case the block coordinate descent method is no longer guaranteed to be globally convergent, as it may get stuck at p. One immediate solution to this is to compute a descent direction at p, then use a line search to find a starting point x_0 with $F(x_0) < F(p)$. Since f is differentiable on the sublevel set $\{x \in \mathbb{R}^n \mid F(x) < F(p)\}$ it follows by the results above that the cluster points of the generated sequence are stationary points of F. This procedure is not efficient since it discards a carefully chosen starting point. We apply the modified coordinate descent loop, outlined in Algorithm 8, instead.

Algorithm 8 Modified coordinate descent loop.

---

   repeat
      Let $i \leftarrow i + 1 \mod m$.
      $x^{(i)} \leftarrow \underset{\hat{x} \in \mathbb{R}^{n_i}}{\arg\min} F(\hat{x} \oplus P_i^{\perp} x)$.
      if $\|x - p\|_2 < \epsilon$ and $F(x) \geq F(p)$ **then**
         Compute descent direction Δ at p for F.
         Use line search to find t such that $F(p + t\Delta) < F(p)$.
         Let $x^{(i)} \leftarrow p + t\Delta$.
      end if
   until stopping condition is met.

Lemma 4. *Assume that f is differentiable everywhere except at $p \in \mathbb{R}^n$, and that F is not optimal at p. Then for any $\epsilon > 0$ the cluster points of the sequence $\{x^k\}_{k \in \mathbb{N}}$ generated by Algorithm 8 are minimizers of F.*

Proof. Let z be a cluster point of $\{x^k\}$. By Corollary 2, z is a coordinatewise minimizer of F. Then Lemma 3 implies that z is either p or a stationary point of F. We shall show by contradiction that p is not a cluster point of $\{x^k\}_{k \in \mathbb{N}}$, thus assume otherwise. The sequence $\{F(x^k)\}_{k \in \mathbb{N}}$ is decreasing; hence, if we can find a $k' \in \mathbb{N}$ such that $F(x^{k'}) < F(p)$ we reach a contradiction (since this would conflict with the continuity of F). Choose k' such that $\left\| x^{k'} - p \right\|_2 < \epsilon$. Since we may assume that $F(x^{k'}) \geq F(p)$ it follows by the definition of Algorithm 8 that $F(x^{k'+1}) < F(p)$. $\quad\square$

Appendix C. Proof of Proposition 1

   (a) Straightforward.
   (b) If $\|\kappa(v, z)\|_2 \leq a$ then $-\kappa(v, z) \in aB^n$ hence $0 \in X$. For the other implication simply choose $y_0 \in Y$ such that $-y_0 \in aB^n$ and note that $\|\kappa(v, z)\|_2 \leq \|y_0\|_2 \leq a$.
   (c) Assume $\|\kappa(v, z)\|_2 > a$, and let $x^* = (1 - a/\|\kappa(v, z)\|_2)\kappa(v, z)$. Then $x^* \in X$ and $\|x^*\|_2 = \|\kappa(v, z)\|_2 - a$. The point x^* is in fact a minimizer. To see this let $x' \in X$, that is we have

$$x' = z + as + \text{diag}(v)t$$

for some $s \in B^n$ and $t \in T_n$. It follows, by the triangle inequality and (a), that

$$\left\| x' \right\|_2 + a \geq \left\| x' - as \right\|_2 = \|z + \text{diag}(v)t\|_2 \geq \|\kappa(v, z)\|_2 .$$

So $\left\| x' \right\|_2 \geq \|\kappa(v, z)\|_2 - a = \|x^*\|_2$ and since X is convex and $x \to \|x\|_2$ is strictly convex the found minimizer x^* is the unique minimizer.

References

Bakay, M., Wang, Z., Melcon, G., Schiltz, L., Xuan, J., Zhao, P., Sartorelli, V., Seo, J., Pegoraro, E., Angelini, C., et al., 2006. Nuclear envelope dystrophies show a transcriptional fingerprint suggesting disruption of rb–myod pathways in muscle regeneration. Brain 129 (4), 996.
Bertsekas, D., Nedić, A., Ozdaglar, A., 2003. Convex Analysis and Optimization. In: Athena Scientific Optimization and Computation Series, Athena Scientific.
Boost C++ libraries, version 1.51.0, 2012. URL http://www.boost.org.
Demming, R., Duffy, D., 2010. Introduction to The Boost C++ Libraries. Datasim Education.
Eddelbuettel, D., Sanderson, C., 2013. Rcpparmadillo : accelerating *r* with high-performance C++ linear algebra. Computational Statistics & Data Analysis URL http://eprints.qut.edu.au/57210/.
Frank, A., Asuncion, A., 2010. UCI machine learning repository. URL http://archive.ics.uci.edu/ml.
Friedman, J., Hastie, T., Tibshirani, R., A note on the group lasso and a sparse group lasso, Tech. Rep., Comments: 8 pages, 3 figs (Jan 2010a), arXiv:1001.0736.
Friedman, J., Hastie, T., Tibshirani, R., 2010b. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33 (1), 1–22. URL http://www.jstatsoft.org/v33/i01/.
Hastie, T., Tibshirani, R., Friedman, J.H., 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction: With 200 Full-Color Illustrations. Springer-Verlag, New York.
Kim, Y., Kwon, S., Heun Song, S., 2006. Multiclass sparse logistic regression for classification of multiple cancer types using gene expression data. Computational Statistics & Data Analysis 51 (3), 1643–1655. http://dx.doi.org/10.1016/j.csda.2006.06.007. URL http://dx.doi.org/10.1016/j.csda.2006.06.007.
Liu, S., Liu, Z., Sun, J., Liu, L., 2011. Application of synergetic neural network in online writeprint identification. International Journal of Digital Content Technology and its Applications 5 (3), 126–135.
Lu, J., Getz, G., Miska, E., Alvarez-Saavedra, E., Lamb, J., Peck, D., Sweet-Cordero, A., Ebert, B., Mak, R., Ferrando, A., et al., 2005. Microrna expression profiles classify human cancers. Nature 435 (7043), 834–838.
Meier, L., Geer, S.V.D., Bhlmann, P., Zrich, E.T.H., 2008. The group lasso for logistic regression. Journal of the Royal Statistical Society, Series B.
The openmp api specification for parallel programming, version 3.1, 2011. URL http://www.openmp.org.
Sanderson, C., Armadillo: an open source C++ linear algebra library for fast prototyping and computationally intensive experiments, Tech. rep., NICTA (October 2010).
Simil, T., Selection, J.T.I., In, S., Simil, T., Tikka, J., 2007. Input selection and shrinkage in multiresponse linear regression. Computational Statistics and Data Analysis 52, 406–422.
Simon, N., Friedman, J., Hastie, T., 2013a. A Blockwise Descent Algorithm for Group-penalized Multiresponse and Multinomial Regression.
Simon, N., Friedman, J., Hastie, T., Tibshirani, R., 2013b. A sparse group lasso. Journal of Computational and Graphical Statistics.
Tibshirani, R., 1994. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B 58, 267–288.
Tseng, P., 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. Journal of optimization theory and applications 109 (3), 475–494.
Tseng, P., Yun, S., 2009. A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming 117 (1), 387–423.