The Group Lasso for Logistic Regression

Lukas Meier * Sara van de Geer † Peter Bühlmann ‡

Abstract

The Group Lasso is an extension of the Lasso to do variable selection on (predefined) groups of variables in linear regression models. The estimates have the attractive property of being invariant under groupwise orthogonal reparametrizations. We extend the Group Lasso to logistic regression models and present an efficient algorithm, especially suitable for high-dimensional problems, which can also be applied to generalized linear models to solve the corresponding convex optimization problem. The Group Lasso estimator for logistic regression is shown to be statistically consistent even if the number of predictors is much larger than sample size but with sparse true underlying structure. We further use a two-stage procedure which aims for sparser models than the Group Lasso, leading to improved prediction performance for some cases. Moreover, due to the two-stage nature, the estimates can be constructed to be hierarchical. The methods are used on simulated and real datasets about splice site detection in DNA sequences.

Keywords: Categorical data, Coordinate descent algorithm, DNA splice site, High-dimensional generalized linear model, Penalized likelihood, Group variable selection.

1 Introduction

The Lasso (Tibshirani, 1996), originally proposed for linear regression models, has become a popular model selection and shrinkage estimation method. In the usual linear regression setup we have a continuous response $\boldsymbol{Y} \in \mathbb{R}^n$, an $n \times p$ design matrix X and a parameter vector $\boldsymbol{\beta} \in \mathbb{R}^p$. The Lasso estimator is then defined as

$$\widehat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - X\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j| \,,$$

where $\|\boldsymbol{u}\|_2^2 = \sum_{i=1}^n u_i^2$ for a vector $\boldsymbol{u} \in \mathbb{R}^n$. For large values of the penalty parameter λ, some components of $\widehat{\boldsymbol{\beta}}_\lambda$ are set exactly to zero. The ℓ_1-type penalty of the Lasso can also be applied to other models as for example Cox regression (Tibshirani, 1997), logistic regression (Lokhorst, 1999; Roth, 2004; Shevade and Keerthi, 2003; Genkin *et al.*, 2004) or multinomial logistic regression (Krishnapuram *et al.*, 2005) by replacing the residual sum of squares by the corresponding negative log-likelihood function.

*Seminar für Statistik, ETH Zürich, Zürich, Switzerland. Address: Leonhardstrasse 27, CH-8092 Zürich, Switzerland. E-Mail: meier@stat.math.ethz.ch

†Seminar für Statistik, ETH Zürich, Zürich, Switzerland

‡Seminar für Statistik, ETH Zürich, Zürich, Switzerland

Already for the special case in linear regression when not only continuous but also categorical predictors (factors) are present, the Lasso solution is not satisfactory as it only selects individual dummy variables instead of whole factors. Moreover, the Lasso solution depends on how the dummy variables are encoded. Choosing different contrasts for a categorical predictor will produce different solutions in general. The Group Lasso (Yuan and Lin, 2006; Bakin, 1999; Cai, 2001; Antoniadis and Fan, 2001) overcomes these problems by introducing a suitable extension of the Lasso penalty. The estimator is defined as

$$\widehat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} \|Y - X\boldsymbol{\beta}\|_2^2 + \lambda \sum_{g=1}^{G} \|\boldsymbol{\beta}_{\mathcal{I}_g}\|_2,$$

where $\mathcal{I}_g$ is the index set belonging to the gth group of variables, $g = 1, \ldots, G$. This penalty can be viewed as an intermediate between the ℓ_1- and ℓ_2-type penalty. It has the attractive property that it does variable selection at the group level and is invariant under (groupwise) orthogonal transformations like Ridge regression (Yuan and Lin, 2006).

This article deals with the Group Lasso penalty for logistic regression models. The logistic case calls for new computational algorithms. Kim *et al.* (2006) first studied the Group Lasso for logistic regression models and proposed a gradient descent algorithm to solve the corresponding constrained problem. We present methods which allow us to work directly on the penalized problem and whose convergence property does not depend on unknown constants as in Kim *et al.* (2006). Our algorithms are efficient in the sense that they can handle problems where p and n are large. Furthermore, they are also applicable for generalized linear models, beyond the case of logistic regression. We do not aim for an (approximate) path-following algorithm (Rosset, 2005; Zhao and Yu, 2004; Park and Hastie, 2007, 2006) but our approaches are fast enough for computing a whole range of solutions for varying penalty parameters on a (fixed) grid. Our approach is related to Genkin *et al.* (2004) which presents an impressively fast implementation ("the fastest") for large-scale logistic regression with the Lasso; in fact, we can also deal with dimensionality p in the tenthousands, but now for the Group Lasso. Moreover, we present an asymptotic consistency theory for the Group Lasso in high-dimensional problems where the predictor dimension is much larger than sample size. This has neither been developed for linear nor logistic regression. High-dimensionality of the predictor space arises in many applications, in particular with higher-order interaction terms or basis expansions for logistic additive models where the groups correspond to the basis functions for individual continuous covariates. Our application about the detection of splice sites, the regions between coding (exons) and non-coding (introns) DNA segments involves the categorical predictor space $\{A, C, G, T\}^7$ which has cardinality 16'384.

The rest of this article is organized as follows: In Section 2 we restate in more detail the idea of the Group Lasso for logistic regression models, present two efficient algorithms which are proven to solve the corresponding convex optimization problem and compare them with other optimization methods. Furthermore, we show that the Group Lasso estimator is statistically consistent for high-dimensional, sparse problems. In Section 3 we outline a two-stage procedure which often produces more adequate models both in terms of model size and prediction performance. Simulations follow in Section 4 and an application of the modeling of functional DNA sites can be found in Section 5. Section 6 contains the discussion. All proofs are given in the Appendix.

2 Logistic Group Lasso

2.1 Model Setup

Assume we have i.i.d. observations $(\boldsymbol{x}_i, y_i), i = 1, \ldots, n$ of a p-dimensional vector $\boldsymbol{x}_i \in \mathbb{R}^p$ of G predictors and a binary response variable $y_i \in \{0, 1\}$. Both categorical and continuous predictors are allowed. We denote by df_g the degrees of freedom of the gth predictor and can thus rewrite $\boldsymbol{x}_i = (\boldsymbol{x}_{i,1}^T, \ldots, \boldsymbol{x}_{i,G}^T)^T$ with the group of variables $\boldsymbol{x}_{i,g} \in \mathbb{R}^{df_g}$, $g = 1, \ldots, G$. For example, the main effect of a factor with 4 levels has $df = 3$ while a continuous predictor involves $df = 1$ only.

Linear logistic regression models the conditional probability $p_{\boldsymbol{\beta}}(\boldsymbol{x}_i) = \mathbb{P}_{\boldsymbol{\beta}}[Y = 1 \,|\, \boldsymbol{x}_i]$ by

$$\log\left\{\frac{p_{\boldsymbol{\beta}}(\boldsymbol{x}_i)}{1 - p_{\boldsymbol{\beta}}(\boldsymbol{x}_i)}\right\} = \eta_{\boldsymbol{\beta}}(\boldsymbol{x}_i), \tag{2.1}$$

with

$$\eta_{\boldsymbol{\beta}}(\boldsymbol{x}_i) = \beta_0 + \sum_{g=1}^{G} \boldsymbol{x}_{i,g}^T \boldsymbol{\beta}_g,$$

where β_0 is the intercept and $\boldsymbol{\beta}_g \in \mathbb{R}^{df_g}$ is the parameter vector corresponding to the gth predictor. We denote by $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ the whole parameter vector, i.e. $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}_1^T, \ldots, \boldsymbol{\beta}_G^T)^T$. The Logistic Group Lasso estimator $\widehat{\boldsymbol{\beta}}_\lambda$ is given by the minimizer of the convex function

$$S_\lambda(\boldsymbol{\beta}) = -\ell(\boldsymbol{\beta}) + \lambda \sum_{g=1}^{G} s(df_g) \|\boldsymbol{\beta}_g\|_2, \tag{2.2}$$

where $\ell(\cdot)$ is the log-likelihood function, i.e.

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \eta_{\boldsymbol{\beta}}(\boldsymbol{x}_i) - \log[1 + \exp\{\eta_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\}].$$

The tuning parameter $\lambda \geq 0$ controls the amount of penalization. Note that we do not penalize the intercept. However, as shown in Lemma 2.1, the minimum in (2.2) is attained. The function $s(\cdot)$ is used to rescale the penalty with respect to the dimensionality of the parameter vector $\boldsymbol{\beta}_g$. Unless stated otherwise, we use $s(df_g) = \sqrt{df_g}$ to ensure that the penalty term is of the order of the number of parameters df_g. The same rescaling is used in Yuan and Lin (2006).

Lemma 2.1. *Assume that* $0 < \sum_{i=1}^{n} y_i < n$. *For* $\lambda > 0$ *and* $s(d) > 0$ *for all* $d \in \mathbb{N}$, *the minimum in optimization problem (2.2) is attained.*

The first condition in Lemma 2.1 is a minimal requirement for the observed data. If the design matrix X has full rank, the minimizer of $S_\lambda(\cdot)$ is unique. Otherwise, the set of minimizers is a convex set whose elements correspond to the same minimum value of $S_\lambda(\cdot)$.

The "groupwise" ℓ_2-norm in (2.2) is an intermediate between the Lasso and the Ridge penalty function. It encourages that in general either $\widehat{\boldsymbol{\beta}}_g = \mathbf{0}$ or $\widehat{\beta}_{g,j} \neq 0$ for all $j \in \{1, \ldots, df_g\}$, where we have omitted the index λ for easier notation. A geometrical interpretation of this special sparsity property is given in Yuan and Lin (2006). An example

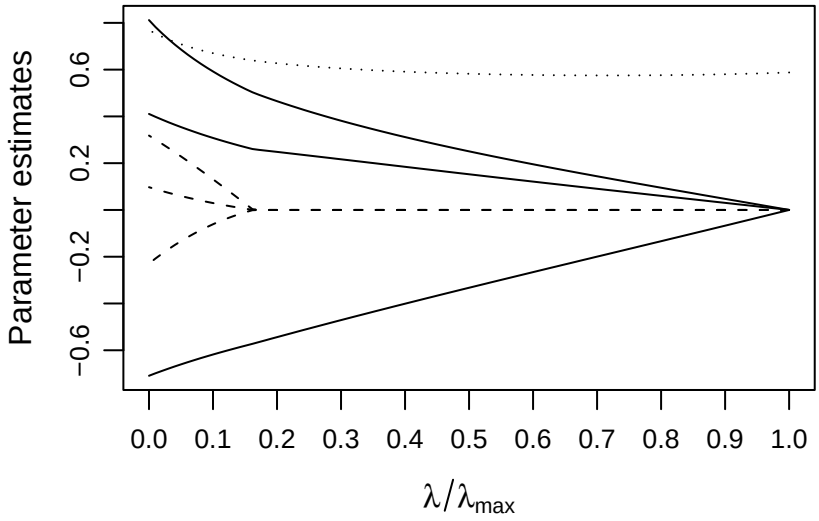

Figure 1: Solution path $\{\widehat{\boldsymbol{\beta}}_\lambda\}_{\lambda \geq 0}$ for a model consisting of an intercept (dotted line) and two factors having 3 degrees of freedom each (dashed and solid lines, respectively). λ_{max} is the value of the penalty parameter λ such that no penalized group is active in the model.

of a solution path $\{\widehat{\boldsymbol{\beta}}_\lambda\}_{\lambda \geq 0}$ for a model consisting of an intercept and two factors having 3 degrees of freedom each is depicted in figure 1.

Let the $n \times df_g$ matrix X_g be the columns of the design matrix corresponding to the gth predictor. If we assume that the block matrices X_g are of full rank, we can perform a (blockwise) orthornomalization – e.g. by a QR-decomposition – to get $X_g^T X_g = I_{df_g},\ g = 1, \ldots, G$. Using such a design matrix, the Group Lasso estimator does not depend on the encoding scheme of the dummy variables. We choose a rescaled version $X_g^T X_g = n \cdot I_{df_g}$ to ensure that the parameter estimates are on the same scale when varying the sample size n. After parameter estimation, the estimates have to be transformed back in order to correspond to the original encoding.

2.2 Algorithms for the Logistic Group Lasso

2.2.1 Block Coordinate Descent

Parameter estimation is computationally more demanding than for linear regression models. The algorithm presented in Yuan and Lin (2006) sequentially solves a system of (necessary and sufficient) non-linear equations which corresponds to a groupwise minimization of the penalized residual sum of squares. Hence, the algorithm is a special case of a block coordinate descent algorithm. No result on numerical convergence is given in Yuan and Lin (2006).

For the more difficult case of logistic regression, we can also use a block coordinate descent algorithm and we prove numerical convergence by using the results of Tseng (2001) as

Logistic Group Lasso Algorithm (Block Coordinate Descent)

(1) Let $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ be an initial parameter vector.
(2) $\beta_0 \leftarrow \arg\min_{\beta_0} S_\lambda(\boldsymbol{\beta})$
(3) **for** $g = 1, \ldots, G$
  if $\|X_g^T(\boldsymbol{y} - \boldsymbol{p}_{\boldsymbol{\beta}_{-g}})\|_2 \leq \lambda s(df_g)$
    $\boldsymbol{\beta}_g \leftarrow \mathbf{0}$
  else
    $\boldsymbol{\beta}_g \leftarrow \arg\min_{\boldsymbol{\beta}_g} S_\lambda(\boldsymbol{\beta})$
  end
end
(4) Repeat step (2)–(3) until some convergence criteria is met.

Table 1: Logistic Group Lasso Algorithm using Block Coordinate Descent Minimization.

shown in Proposition 2.1. The key lies in the separable structure of the non-differentiable part in S_λ. These properties of course also apply for the Group Lasso for linear and generalized linear regression models.

We cycle through the parameter groups and minimize the objective function S_λ, keeping all but the current parameter group fixed. This leads us to the algorithm presented in Table 1, where we denote by $\boldsymbol{\beta}_{-g}$ the parameter vector $\boldsymbol{\beta}$ when setting $\boldsymbol{\beta}_g$ to $\mathbf{0}$ while all other components remain unchanged. In step (3) we first check whether the minimum is at the non-differentiable point $\boldsymbol{\beta}_g = \mathbf{0}$. If not, we can use a standard numerical minimizer, e.g. a Newton type algorithm, to find the optimal solution with respect to $\boldsymbol{\beta}_g$. In such a case the values of the last iteration can be used as starting values to save computing time. If the group was not in the model in the last iteration, we first go a small step in the opposite direction of the gradient of the negative log-likelihood function to ensure that we start at a differentiable point.

Proposition 2.1. *Step (2) and (3) of the block coordinate descent algorithm perform groupwise minimizations of S_λ and are well defined in the sense that the corresponding minima are attained. Furthermore, if we denote by $\widehat{\boldsymbol{\beta}}^{(t)}$ the parameter vector after t block updates, then every limit point of the sequence $\{\widehat{\boldsymbol{\beta}}^{(t)}\}_{t\geq 0}$ is a minimum point of S_λ.*

Because the iterates can be shown to stay in a compact set, the existence of a limit point is guaranteed.

The main drawback of such an algorithm is that the blockwise minimizations of the active groups have to be performed numerically. However, for small and moderate sized problems in the dimension p and the group sizes df_g this turns out to be sufficiently fast. For large-scale applications it would be attractive to have a closed form solution for a block update as in Yuan and Lin (2006). This will be discussed in the next subsection.

2.2.2 Block Coordinate Gradient Descent

The key idea of the block coordinate gradient descent method of Tseng and Yun (2006) is to combine a quadratic approximation of the log-likelihood with an additional line

search. Using a second order Taylor expansion at $\widehat{\boldsymbol{\beta}}^{(t)}$ and replacing the Hessian of the log-likelihood function $\ell(\cdot)$ by a suitable matrix $H^{(t)}$ we define

$$\begin{aligned} M_\lambda^{(t)}(\boldsymbol{d}) &= -\left\{\ell(\widehat{\boldsymbol{\beta}}^{(t)}) + \boldsymbol{d}^T\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)}) + \frac{1}{2}\boldsymbol{d}^T H^{(t)}\boldsymbol{d}\right\} + \lambda\sum_{g=1}^{G} s(df_g)\|\widehat{\boldsymbol{\beta}}_g^{(t)} + \boldsymbol{d}_g\|_2 \quad (2.3) \\ &\approx S_\lambda(\widehat{\boldsymbol{\beta}}^{(t)} + \boldsymbol{d}), \end{aligned}$$

where $\boldsymbol{d} \in \mathbb{R}^{p+1}$. Now we consider the minimization of $M_\lambda^{(t)}(\cdot)$ with respect to the gth penalized parameter group. This means that we restrict ourselves to vectors $\boldsymbol{d}$ with $\boldsymbol{d}_k = \boldsymbol{0}$ for $k \neq g$. Moreover, we assume that the corresponding $df_g \times df_g$ submatrix $H_{gg}^{(t)}$ is of the form $H_{gg}^{(t)} = h_g^{(t)} \cdot I_{df_g}$ for some scalar $h_g^{(t)} \in \mathbb{R}$.

If $\|\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)})_g - h_g^{(t)}\widehat{\boldsymbol{\beta}}_g^{(t)}\|_2 \leq \lambda s(df_g)$, the minimizer of (2.3) is

$$\boldsymbol{d}_g^{(t)} = -\widehat{\boldsymbol{\beta}}_g^{(t)}.$$

Otherwise

$$\boldsymbol{d}_g^{(t)} = -\frac{1}{h_g^{(t)}}\left\{\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)})_g - \lambda s(df_g)\frac{\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)})_g - h_g^{(t)}\widehat{\boldsymbol{\beta}}_g^{(t)}}{\|\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)})_g - h_g^{(t)}\widehat{\boldsymbol{\beta}}_g^{(t)}\|_2}\right\}.$$

If $\boldsymbol{d}^{(t)} \neq \boldsymbol{0}$, an inexact line search using the Armijo rule has to be performed: Let $\alpha^{(t)}$ be the largest value in $\{\alpha_0\delta^l\}_{l\geq 0}$ such that

$$S_\lambda(\widehat{\boldsymbol{\beta}}^{(t)} + \alpha^{(t)}\boldsymbol{d}^{(t)}) - S_\lambda(\widehat{\boldsymbol{\beta}}^{(t)}) \leq \alpha^{(t)}\sigma\Delta^{(t)},$$

where $0 < \delta < 1$, $0 < \sigma < 1$, $\alpha_0 > 0$, and $\Delta^{(t)}$ is the improvement in the objective function S_λ when using a linear approximation for the log-likelihood, i.e.

$$\Delta^{(t)} = -(\boldsymbol{d}^{(t)})^T\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)}) + \lambda s(df_g)\|\widehat{\boldsymbol{\beta}}_g^{(t)} + \boldsymbol{d}_g^{(t)}\|_2 - \lambda s(df_g)\|\widehat{\boldsymbol{\beta}}_g^{(t)}\|_2.$$

Finally, we define

$$\widehat{\boldsymbol{\beta}}^{(t+1)} = \widehat{\boldsymbol{\beta}}^{(t)} + \alpha^{(t)}\boldsymbol{d}^{(t)}.$$

The algorithm is outlined in Table 2. When minimizing $M_\lambda^{(t)}$ with respect to a penalized group, we first have to check whether the minimum is at a non-differentiable point as outlined above. For the (unpenalized) intercept this is not necessary and the solution can be directly computed

$$d_0^{(t)} = -\frac{1}{h_0^{(t)}}\nabla\ell(\widehat{\boldsymbol{\beta}}^{(t)})_0.$$

For a general matrix $H^{(t)}$ the minimization with respect to the gth parameter group depends on $H^{(t)}$ only through the corresponding submatrix $H_{gg}^{(t)}$. To ensure a reasonable quadratic approximation in (2.3), $H_{gg}^{(t)}$ is ideally chosen to be close to the corresponding submatrix of the Hessian of the log-likelihood function. Restricting ourselves to matrices of the form $H_{gg}^{(t)} = h_g^{(t)} \cdot I_{df_g}$, a possible choice is (Tseng and Yun, 2006)

$$h_g^{(t)} = -\max\left[\operatorname{diag}\left\{-\nabla^2\ell(\widehat{\boldsymbol{\beta}}^{(t)})_{gg}\right\}, c_*\right], \quad (2.4)$$

where $c_* > 0$ is a lower bound to ensure convergence (see Proposition 2.2). The matrix $H^{(t)}$ does not necessarily have to be recomputed in each iteration. Under some mild

Logistic Group Lasso Algorithm (Block Coordinate Gradient Descent)

(1) Let $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ be an initial parameter vector.
(2) **for** $g = 0, \ldots, G$
  $H_{gg} \leftarrow h_g(\boldsymbol{\beta}) \cdot I_{df_g}$
  $\boldsymbol{d} \leftarrow \underset{\boldsymbol{d} \,|\, \boldsymbol{d}_k = \boldsymbol{0}, k \neq g}{\arg\min} \; M_\lambda(\boldsymbol{d})$
  if $\boldsymbol{d} \neq \boldsymbol{0}$
    $\alpha \leftarrow$ Line search
    $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha \cdot \boldsymbol{d}$
  end
end
(3) Repeat step (2) until some convergence criteria is met.

Table 2: Logistic Group Lasso Algorithm using Block Coordinate Gradient Descent Minimization.

conditions on $H^{(t)}$ convergence of the algorithm is assured as can be seen from Tseng and Yun (2006) and from the proof of Proposition 2.2.

Standard choices for the tuning parameters are for example $\alpha_0 = 1$, $\delta = 0.5$, $\sigma = 0.1$ (Bertsekas, 2003; Tseng and Yun, 2006). Other definitions of $\Delta^{(t)}$ as for example to include the quadratic part of the improvement are also possible. We refer the reader to Tseng and Yun (2006) for more details and proofs that $\Delta^{(t)} < 0$ for $\boldsymbol{d}^{(t)} \neq \boldsymbol{0}$ and that the line search can always be performed.

Proposition 2.2. *If $H_{gg}^{(t)}$ is chosen according to (2.4), then every limit point of the sequence $\{\widehat{\boldsymbol{\beta}}^{(t)}\}_{t \geq 0}$ is a minimum point of S_λ.*

Remark 2.1. *When cycling through the coordinate blocks, we could restrict ourselves to the current active set and visit the remaining blocks e.g. every 10th iteration to update the active set. This is especially useful for very high-dimensional settings and it easily allows for $p \approx 10^4 - 10^5$. For the high-dimensional example in Section 2.3, this modification decreases the computational times by about 40% of what is reported in Figure 2. Moreover, it is also possible to update the coordinate blocks in a non-cyclic manner or all at the same time which would allow for a parallelizable approach with the convergence result still holding.*

Remark 2.2. *The block coordinate gradient descent algorithm can also be applied to the Group Lasso in other models. For example, any generalized linear model where the response y has a distribution from the exponential family falls into this class. This is available in our R-package* `grplasso`.

A related algorithm is found in Krishnapuram *et al.* (2005), where a global upper bound on the Hessian is used to solve the Lasso problem for multinomial logistic regression. This approach can also be used with the Group Lasso penalty resulting in a closed form solution for a block update. However, the upper bound is not tight enough for moderate and small values of λ which leads to too slow convergence in general. Genkin *et al.* (2004) overcomes this problem by working with an updated local bound on the second derivative and by restricting the change of the current parameter to a local neighbourhood.

For linear models, the LARS-algorithm (Efron *et al.*, 2004; Osborne *et al.*, 2000) is very efficient for computing the path of Lasso solutions $\{\widehat{\boldsymbol{\beta}}_\lambda\}_{\lambda \geq 0}$. For logistic regression, approximate path following algorithms have been proposed (Rosset, 2005; Zhao and Yu, 2004; Park and Hastie, 2007). But with the Group Lasso penalty, some of them are not applicable (Rosset, 2005) or do not necessarily converge to a minimum point of S_λ (Zhao and Yu, 2004), and all of them do not seem to be computationally faster than working iteratively on a fixed grid of penalty parameters λ. The latter has been observed as well by Genkin *et al.* (2004) for logistic regression with the Lasso in large-scale applications.

To calculate the solutions $\widehat{\boldsymbol{\beta}}_\lambda$ on a grid of the penalty parameter $0 \leq \lambda_K < \ldots < \lambda_1 \leq \lambda_{max}$ we can for example start at

$$\lambda_{max} = \max_{g \in \{1,\ldots,G\}} \frac{1}{s(df_g)} \|X_g^T(\boldsymbol{y} - \bar{\boldsymbol{y}})\|_2,$$

where only the intercept is in the model. We then use $\widehat{\boldsymbol{\beta}}_{\lambda_k}$ as a starting value for $\widehat{\boldsymbol{\beta}}_{\lambda_{k+1}}$ and proceed iteratively until $\widehat{\boldsymbol{\beta}}_{\lambda_K}$ with λ_K equal or close to zero. Instead of updating the approximation of the Hessian $H^{(t)}$ in each iteration, we can use a constant matrix based on the previous parameter estimates $\widehat{\boldsymbol{\beta}}_{\lambda_k}$ to save computing time, i.e.

$$H_{gg}^{(t)} = h_g(\widehat{\boldsymbol{\beta}}_{\lambda_k}) I_{df_g},$$

for the estimation of $\widehat{\boldsymbol{\beta}}_{\lambda_{k+1}}$. Some cross-validation can then be used for choosing the parameter λ. Most often, we aim for minimal test-sample negative log-likelihood score.

2.3 Comparison with Other Algorithms

In this subsection we compare the Block Coordinate Gradient Descent algorithm (BCGD) with the algorithm of Kim *et al.* (2006) (BSR, standing for Blockwise Sparse Regression). After an earlier version of this manuscript, Park and Hastie (2006) (PH) also applied their methodology of Park and Hastie (2007) to Group Lasso models which we also include in our comparison.

We emphasize that BSR is a method which requires the specification of an algorithmic tuning parameter, denoted by s. It is shown in Kim *et al.* (2006) that numerical convergence of BSR only holds if s is chosen sufficiently small (depending on the unknown Lipschitz constant of the gradient). Moreover, a small parameter s slows down the computational speed of BSR, and vice-versa for a large s. Thus, we are in a situation of trading-off numerical convergence versus computational speed. Our BCGD method does not require the specification of an algorithmic tuning parameter to ensure convergence, and we view this as a very substantial advantage for practical use.

For comparing the different algorithms, we use a random design matrix where the predictors are simulated according to a centered multivariate normal distribution with covariance matrix $\Sigma_{i,j} = \rho^{|i-j|}$. If not stated otherwise, $\rho = 0.5$ is used. For the penalty parameter λ multiplicative grids between λ_{max} and $\lambda_{max}/100$ are used.

For BCGD we use the R package `grplasso` and for BSR our own implementation in R. As BSR works with a constraint instead of a penalty, we use the result of BCGD as constraint value. We use an equivalent stopping criteria as in package `grplasso`, i.e. the relative function improvement and the relative change of the parameter vector have to

be small enough. Although this slowed down the algorithms, it is necessary in order to identify the correct active set of the solution. For both algorithms we make use of the preceding solution of the path as starting value for the next grid point. For BCGD we update the Hessian at each 5th grid point and we use an "ordinary" cycling through the coordinate blocks. For the path-following algorithm PH we use the corresponding Matlab implementation available at `http://www.stanford.edu/~mypark/glasso.htm`. As recommended, the step length on the λ-scale is chosen adaptively. However, we were able to run PH with reasonable computing time on very small datasets only.

One of them is motivated by the user guide of PH. It consists of $n = 200$ observations of $G = 3$ groups each having $df = 3$, i.e. $p = 10$ (with intercept). For the design matrix we use $\rho = 0$ and the whole parameter vector is set to zero, i.e. there is no signal. 20 grid points are used for λ. The corresponding cpu times (in seconds) based on 20 simulation runs are 0.093 (0.01), 0.041 (0.0054), 5.96 (1.23) for BCGD, BSR and PH respectively. Standard deviations are given in parantheses. We used the tuning parameter $s = 0.01$ for BSR. Already for such a simple, low-dimensional problem, BCGD and BSR were substantially faster than PH. As mentioned above, we could not run PH for larger problems (this is probably due to implementation, but we also think that an optimized implementation of PH, involving potentially large active sets, would be slower than BCGD or BSR).

As a second example, we use a higher-dimensional setting with $n = 100$ and $G = 250$ groups each having $df = 4$ ($p = 1001$). The first 10 groups are active with coefficient 0.2, resulting in a Bayes risk of approximately 0.2. The computing times based on 20 simulation runs are depicted in Figure 2, where we have used 100 grid points for λ. The boxplot for $s_0 = 0.025$ is dashed because the probability for numerical convergence was only 20%. BSR with s suitably chosen is not faster in this example. The success for

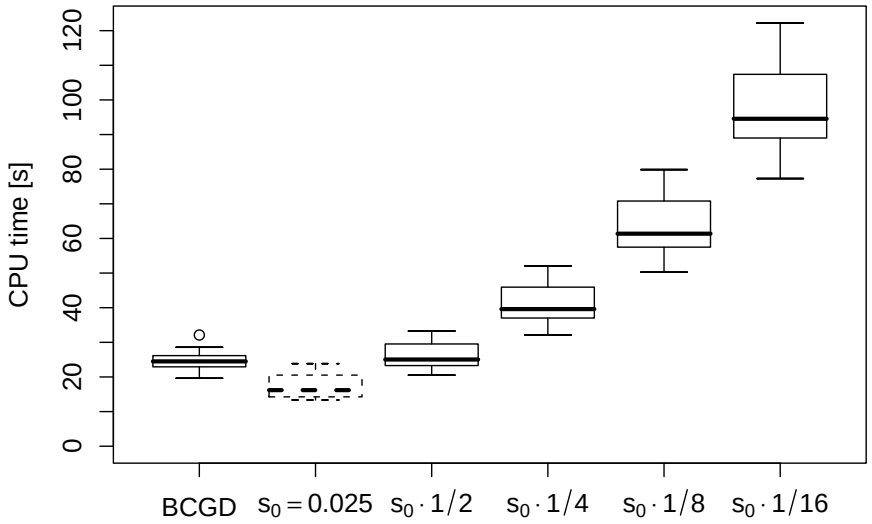

Figure 2: CPU times for BCGD (left) and for BSR with different values of the parameter s. See text for more details.

numerical convergence depends heavily on the choice of s and additional time is needed

to find an optimal value for s. For some single λ values, BSR sometimes turned out to be faster, but the difference between the computing times when calculating the whole path is much smaller due to good starting values and the fact that BSR slows down for small values of λ.

We also applied BSR to the splice site dataset in Section 5. The running times are reported in Table 3. We summarize that BCGD is often almost as fast or even faster (as for the

BCGD	$s = 5 \cdot 10^{-4}$	$s = 2.5 \cdot 10^{-4}$	$s = 1.25 \cdot 10^{-4}$	$s = 6.125 \cdot 10^{-5}$
948	2737	4273	6688	10581

Table 3: CPU times [s] on the splice site dataset for BCGD (left) and for BSR with different values for the parameter s. The algorithm did not converge for larger values of s.

real splice site dataset) as BSR with the optimal algorithmic tuning parameter s. This tuning parameter varies very much from problem to problem, and it is highly unrealistic to have reasonable a-priori knowledge about a good parameter. Thus, the user needs to do some trial-and-error first which can be very unpleasant. In contrast, BCGD runs fully automatic and is proved to converge, as described in Section 2.2.2.

Due to implementational issues it can be difficult to compare different algorithms. But the fact that coordinate-wise approaches for sparse models are efficient for high-dimensional data has also been noticed by Genkin *et al.* (2004) or Balakrishnan and Madigan (2006). They have successfully applied related algorithms for the Lasso even when the number of variables was in the hundreds of thousands. For the coordinate-wise approaches in general, already after a few sweeps through all variables, both the objective function and the number of selected variables is close to the optimal solution.

2.4 Consistency

A reasonable choice of the tuning parameter λ will depend on the sample size n, as well as on the number of groups G, and the degrees of freedom within each group. Assuming the degrees of freedom per group is kept fixed, the smoothing parameter λ can be taken of order $\log(G)$. Then the Group Lasso can be shown to be globally consistent under some further regularity and sparseness conditions. This section gives more details on this asymptotic result.

Let us consider the data (before rescaling) $(\boldsymbol{x}_i, y_i)$ as independent copies of the population variable $(\boldsymbol{x}, y)$. The negative log-likelihood function is used as loss function which we denote for easier notation by

$$\gamma_{\boldsymbol{\beta}}(\boldsymbol{x}, y) = -\left(y\eta_{\boldsymbol{\beta}}(\boldsymbol{x}) - \log[1 + \exp\{\eta_{\boldsymbol{\beta}}(\boldsymbol{x})\}]\right).$$

The theoretical risk is defined as

$$R(\boldsymbol{\beta}) = \mathbb{E}\left[\gamma_{\boldsymbol{\beta}}(\boldsymbol{x}, y)\right],$$

and the empirical counterpart as

$$R_n(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \gamma_{\boldsymbol{\beta}}(\boldsymbol{x}_i, y_i).$$

With this notation, the Logistic Group Lasso estimator $\widehat{\boldsymbol{\beta}}_\lambda$ is the minimizer of

$$\frac{S_\lambda(\boldsymbol{\beta})}{n} = R_n(\boldsymbol{\beta}) + \frac{\lambda}{n}\sum_{g=1}^{G} s(df_g)\|\boldsymbol{\beta}_g\|_2.$$

Let us consider a minimizer

$$\boldsymbol{\beta}^0 \in \arg\min_{\boldsymbol{\beta}} R(\boldsymbol{\beta}).$$

Note that if the model is well-specified it holds that

$$\mathbb{E}\left[y \,|\, \boldsymbol{x}\right] = p_{\boldsymbol{\beta}^0}(\boldsymbol{x}).$$

There are various ways to measure the quality of the estimation procedure. We will use the global measure

$$d^2(\eta_{\widehat{\boldsymbol{\beta}}_\lambda}, \eta_{\boldsymbol{\beta}^0}) = \mathbb{E}\left[\left|\eta_{\widehat{\boldsymbol{\beta}}_\lambda}(\boldsymbol{x}) - \eta_{\boldsymbol{\beta}^0}(\boldsymbol{x})\right|^2\right].$$

The following assumptions are made:

(A1) We will suppose that for some constant $0 < \epsilon \leq 1/2$

$$\epsilon \leq p_{\boldsymbol{\beta}^0}(\boldsymbol{x}) \leq 1 - \epsilon$$

for all $\boldsymbol{x}$.

(A2) We will require that the matrix

$$\Sigma = \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right]$$

is non-singular. We denote the smallest eigenvalue of Σ by ν^2.

(A3) Let $\boldsymbol{x}_g$ denote the gth predictor in $\boldsymbol{x}$. We normalize $\boldsymbol{x}_g$ such that it has identity innerproduct matrix $\mathbb{E}\left[\boldsymbol{x}_g\boldsymbol{x}_g^T\right] = I_{df_g}$. With this normalization, we assume in addition that for some constant L_n,

$$\max_{\boldsymbol{x}} \max_{g} \boldsymbol{x}_g^T\boldsymbol{x}_g \leq nL_n^2.$$

The smallest possible order for L_n^2 is $L_n^2 = O(1/n)$, since we use the normalization $\mathbb{E}\left[\boldsymbol{x}_g\boldsymbol{x}_g^T\right] = I_{df_g}$. For categorical predictors, $L_n^2 = O(1/n)$ corresponds to the balanced case where in each category the probability of finding an individual in that category is bounded away from zero and one.

One can then show consistency in the following sense. Let $\boldsymbol{\beta}_g^0$ denote the elements in the vector $\boldsymbol{\beta}^0$ corresponding to the gth group. Let N_0 be the number of non-zero group effects, i.e. the number of vectors $\boldsymbol{\beta}_g^0$ satisfying $\|\boldsymbol{\beta}_g^0\|_2 \neq 0$. Then there exist universal constants C_1, C_2, C_3 and C_4, and constants c_1 and c_2 depending on ϵ, ν and $\max_g df_g$, such that whenever

(A4) $C_1(1 + N_0^2)L_n^2 \log G \leq c_1$ and $C_1 \log G \leq \lambda \leq \frac{c_1}{(1+N_0^2)L_n^2}$

the following probability inequality holds:

$$\mathbb{P}\left[d^2(\eta_{\widehat{\boldsymbol{\beta}}_\lambda}, \eta_{\boldsymbol{\beta}^0}) \geq c_2 \frac{(1+N_0)\lambda}{n}\right] \leq C_2 \left\{\log(n) \exp\left(-\frac{\lambda}{C_3}\right) + \exp\left(-\frac{1}{C_4 L_n^2}\right)\right\}.$$

This result follows from arguments similar to the ones used in van de Geer (2003) and Tarigan and van de Geer (2006). An outline of the proof is given in the appendix.

For the asymptotic implications, let us assume that ϵ, ν and $\max_g df_g$ are kept fixed as $n \to \infty$ and that $G \gg \log n$. Take $\lambda \asymp \log G$, i.e. λ is of the order $\log G$. When for example the number of non-zero group effects N_0 satisfies $N_0 = O(1)$, and when $L_n^2 = O(1/\log G)$, then we find the, almost parametric, rate

$$d^2(\eta_{\widehat{\boldsymbol{\beta}}_\lambda}, \eta_{\boldsymbol{\beta}^0}) = O_P\left(\frac{\log G}{n}\right).$$

With $L_n^2 = O(1/n)$, the maximal rate of growth for N_0 is $N_0 = O(\sqrt{n/\log G})$, and when N_0 is exactly of this order we arrive at the rate

$$d^2(\eta_{\widehat{\boldsymbol{\beta}}_\lambda}, \eta_{\boldsymbol{\beta}^0}) = O_P\left(\sqrt{\frac{\log G}{n}}\right).$$

Remark 2.3. *We may improve the result by replacing N_0 by the number of non-zero coefficients of "the best" approximation of $\eta_{\boldsymbol{\beta}^0}$, which is the approximation that balances estimation error and approximation error.*

Remark 2.4. *A similar consistency result can be obtained for the Group Lasso for Gaussian regression.*

3 Logistic Group Lasso/Ridge Hybrid

3.1 General Case

As can be observed in the simulation study in Yuan and Lin (2006), the models selected by the Group Lasso are large compared to the underlying true models. For the ordinary Lasso, smaller models with good prediction performance can be obtained using Lasso with relaxation (Meinshausen, 2007). This idea can also be incorporated into the (logistic) Group Lasso approach and our proposal will also allow to fit hierarchical models.

Denote by $\widehat{\mathcal{I}}_\lambda \subseteq \{0, \ldots, G\}$ the index set of predictors selected by the Group Lasso with penalty parameter λ and by $\widehat{\mathcal{M}}_\lambda = \{\boldsymbol{\beta} \in \mathbb{R}^{p+1} \,|\, \boldsymbol{\beta}_g = 0 \text{ for } g \notin \widehat{\mathcal{I}}_\lambda\}$ the set of possible parameter vectors of the corresponding submodel. The Group Lasso Ridge Hybrid estimator is defined as

$$\widehat{\boldsymbol{\beta}}_{\lambda,\kappa} = \underset{\boldsymbol{\beta} \in \widehat{\mathcal{M}}_\lambda}{\arg\min} -\ell(\boldsymbol{\beta}) + \kappa \sum_{g=1}^{G} \frac{s(df_g)}{\sqrt{df_g}} \|\boldsymbol{\beta}_g\|_2^2 \tag{3.5}$$

for λ, $\kappa \geq 0$. The penalty in (3.5) is rescaled with $1/\sqrt{df_g}$ to ensure that it is of the same order as the Group Lasso penalty. The special case $\kappa = 0$ is analogous to the LARS/OLS hybrid in Efron *et al.* (2004) and is denoted by Group Lasso/MLE hybrid. In this case, we only need the Group Lasso to select a candidate model $\widehat{\mathcal{M}}_\lambda$ and estimate its parameters with the (unconstrained) maximum likelihood estimator. Optimization problem (3.5) can be solved with a Newton type algorithm. For large scale applications coordinatewise approaches as used in Genkin *et al.* (2004) may be more appropriate. The reason why we choose a Ridge type penalty follows in the next subsection.

3.2 Restriction to Hierarchical Models

When working with interactions between predictors (e.g. factors), the Group Lasso solutions are not necessarily hierarchical. An interaction may be present even though (some) corresponding main-effects are missing. In most applications hierarchical models are preferred because of their interpretability. The above two stage procedure (3.5) can be used to produce hierarchical models by expanding the model class $\widehat{\mathcal{M}}_\lambda$ to $\widehat{\mathcal{M}}_\lambda^{hier}$, where $\widehat{\mathcal{M}}_\lambda^{hier}$ is the hierarchical model class induced by $\widehat{\mathcal{M}}_\lambda$: $\widehat{\boldsymbol{\beta}}_{\lambda,\kappa}^{hier}$ is then defined as in (3.5), but with the minimum taken over $\widehat{\mathcal{M}}_\lambda^{hier}$.

Instead of using a Ridge type penalty, we could have also used again the Group Lasso penalty and proceed exactly as in Meinshausen (2007), using a "relaxed" penalty $\kappa \leq \lambda$ in the second stage. While this works well for the general (non-hierarchical) case, there are problems if we restrict ourselves to hierarchical models. Even if we choose $\kappa \leq \lambda$ the solutions may not be hierarchical due to the expansion of $\widehat{\mathcal{M}}_\lambda$ to $\widehat{\mathcal{M}}_\lambda^{hier}$. In other words: some variable selection may happen in addition in the second stage. Using a Ridge type penalty, we prevent any further model selection and just do shrinkage.

4 Simulation

We use a simulation scheme similar to that of Yuan and Lin (2006) but with larger models. In each simulation run we first sample n_{train} instances of a 9 dimensional multivariate normal distribution $(T_1, \ldots, T_9)^T$ with mean vector $\mathbf{0}$ and covariance matrix $\Sigma_{i,j} = \rho^{|i-j|}$. Each component T_k, $k = 1, \ldots, 9$ is subsequently transformed into a four-valued categorical random variable using the quartiles of the standard normal distribution. For the main effects, this results in (non-orthogonalized) predictors $\boldsymbol{x}_i = (\boldsymbol{x}_{i,1}^T, \ldots, \boldsymbol{x}_{i,9}^T)^T \in \mathbb{R}^{9\times 3}$, $i = 1, \ldots, n_{train}$. We use the sum-constraint as encoding scheme for the dummy variables, i.e. the coefficients have to add up to zero. The entire predictor space has dimension $4^9 = 262'144$. The corresponding responses y_i are simulated according to a Bernoulli distribution with model based probabilities.

The parameter vector $\boldsymbol{\beta}_g$ of a predictor with df_g degrees of freedom is set up as follows to conform to the encoding scheme: We simulate $df_g + 1$ independent standard normal distributions resulting in $\widetilde{\beta}_{g,1}, \ldots, \widetilde{\beta}_{g,df_g+1}$ and define

$$\beta_{g,j} = \widetilde{\beta}_{g,j} - \frac{1}{df_g + 1} \sum_{k=1}^{df_g+1} \widetilde{\beta}_{g,k}$$

for $j \in \{1, \ldots, df_g\}$. The intercept is set to zero. The whole parameter vector $\boldsymbol{\beta}$ is finally rescaled to adjust the empirical Bayes risk r at the desired level, where

$$r = \frac{1}{n} \sum_{i=1}^{n} \min\{p_{\boldsymbol{\beta}}(\boldsymbol{x}_i),\, 1 - p_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\}$$

for some large n. For all simulation runs for a given setting of ρ and r, the same parameter vector is re-used.

The four different cases studied are:

(A) The main effects and the 2-way interaction between the first two factors $\boldsymbol{x}_{i,1}$, $\boldsymbol{x}_{i,2}$ build the true model which has a total of 4 terms or 16 parameters. $n_{train} = 500$ observations are used in each simulation run.

(B) The underlying true model consists of all main effects and 2-way interactions between the first 5 factors $\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,5}$ resulting in 16 terms or 106 parameters. $n_{train} = 500$.

(C) As case (B) but with $n_{train} = 1000$.

(D) All main effects and 2-way interactions between $\boldsymbol{x}_{i,k}$ and $\boldsymbol{x}_{i,l}$, $|k - l| = 1$ are active. In addition the 2-way interaction between $\boldsymbol{x}_{i,1}$, $\boldsymbol{x}_{i,5}$ and $\boldsymbol{x}_{i,3}$, $\boldsymbol{x}_{i,9}$ are present. This makes a total of 20 terms or 118 parameters. $n_{train} = 1000$.

For estimation, the candidate models used for the Logistic Group Lasso and its variants consist always of all main effects and all 2-way interactions from the 9 factors, which makes a total of 46 terms or $p = 352$ parameters. We use the restriction for hierarchical model fitting for the two-stage procedures.

In each simulation run, all models are fitted on a training dataset of size n_{train}. A multiplicative grid of the form $\{\lambda_{max}, 0.96 \cdot \lambda_{max}, \ldots, 0.96^{148} \cdot \lambda_{max}, 0\}$ is used for the penalty parameter λ of the Logistic Group Lasso. For the Logistic Group Lasso/Ridge hybrid, we consider values $\kappa \in \{1.5^{11}, 1.5^{10}, \ldots, 1.5^{-5}, 0\}$. The penalty parameters λ and κ are selected according to the (unpenalized) log-likelihood score on an independent validation set of size $n_{train}/2$. Finally, the models are evaluated on an additional test set of size n_{train}.

Table 4 reports the average test-set negative log-likelihood and the average number of selected terms based on 100 simulation runs for each setting. The corresponding standard deviations are given in parentheses. The Group Lasso produces the largest models followed by Group Lasso/Ridge hybrid and Group Lasso/MLE hybrid. Compared to the underlying true models, Group Lasso seems to select unnecessarily large models with many noise variables resulting in a low true discovery rate (not shown), which is defined as the ratio of the number of correctly selected terms and the total number of selected terms. On the other side, Group Lasso/MLE hybrid is very conservative in selecting terms resulting in a large true discovery rate at the cost of a low true positive rate (not shown). Group Lasso/Ridge hybrid seems to be the best compromise.

The prediction performance measured in terms of the (unpenalized) log-likelihood score on the test set is in most cases best for the Group Lasso/Ridge hybrid, followed by the Group Lasso and the Group Lasso/MLE hybrid. Group Lasso/Ridge hybrid seems to be able to benefit from the good prediction performance of the Group Lasso with the advantage of producing reasonably sized models.

5 Application to Splice Site Detection

The prediction of short DNA motifs plays an important role in many areas of computational biology. Gene finding algorithms such as GENIE (Burge and Karlin, 1997) often rely on the prediction of splice sites. Splice sites are the regions between coding (exons) and non-coding (introns) DNA segments. The 5' end of an intron is called a donor splice site and the 3' end an acceptor splice site. A donor site whose first two intron positions are the letters "GT" is called canonical, whereas an acceptor site is called canonical if the

Table 4: Average test-set negative log-likelihood and average number of selected terms based on 100 simulation runs. Standard deviations are given in parentheses. Methods: Group Lasso (GL), Group Lasso/Ridge hybrid (GL/R) and Group Lasso/MLE hybrid (GL/MLE).

			Test-set negative log-likelihood			Number of terms		
	ρ	r	GL	GL/R	GL/MLE	GL	GL/R	GL/MLE
Case A	0.00	0.15	185.57	185.76	212.74	20.73	4.11	3.96
			(10.85)	(26.42)	(37.28)	(8.02)	(0.40)	(0.49)
		0.25	273.57	269.35	278.76	16.39	4.30	4.01
			(8.28)	(19.63)	(29.57)	(6.74)	(0.83)	(1.01)
	0.20	0.15	185.85	182.31	207.24	20.08	4.28	3.90
			(11.62)	(20.82)	(37.63)	(7.15)	(1.05)	(0.56)
		0.25	274.92	269.34	273.89	15.21	4.17	4.03
			(8.48)	(17.63)	(22.11)	(6.22)	(0.53)	(0.58)
	0.50	0.15	194.67	191.90	207.16	18.73	4.17	3.79
			(12.49)	(24.68)	(28.25)	(7.61)	(0.73)	(0.50)
		0.25	279.59	275.35	283.90	14.34	4.26	3.83
			(9.37)	(13.34)	(16.78)	(6.38)	(0.80)	(0.88)
Case B	0.00	0.15	233.84	229.69	291.39	35.90	16.86	8.93
			(12.76)	(15.15)	(27.06)	(3.98)	(3.37)	(3.22)
		0.25	300.08	306.07	325.53	30.95	15.78	6.02
			(9.50)	(10.92)	(18.61)	(5.77)	(6.85)	(3.39)
	0.20	0.15	235.29	232.13	283.32	36.20	16.65	9.19
			(12.71)	(16.29)	(21.19)	(3.94)	(3.60)	(2.83)
		0.25	299.65	303.82	322.22	30.93	16.37	6.85
			(8.94)	(11.78)	(11.53)	(5.27)	(7.75)	(3.36)
	0.50	0.15	232.63	230.48	285.17	35.11	15.67	8.47
			(11.35)	(15.07)	(29.88)	(3.73)	(4.02)	(2.82)
		0.25	296.56	298.68	321.80	29.62	14.11	6.61
			(7.78)	(11.09)	(23.76)	(5.88)	(5.55)	(3.14)
Case C	0.00	0.15	408.63	385.03	423.05	40.95	15.78	14.72
			(16.55)	(19.17)	(34.11)	(2.52)	(1.06)	(1.13)
		0.25	569.56	557.99	581.94	36.34	15.41	13.63
			(14.13)	(17.31)	(24.89)	(3.83)	(2.27)	(1.85)
	0.20	0.15	408.27	385.64	424.42	40.54	15.89	14.49
			(18.53)	(20.43)	(30.87)	(2.79)	(1.56)	(1.18)
		0.25	566.46	556.70	584.23	36.65	15.92	13.25
			(15.17)	(19.44)	(28.48)	(4.11)	(2.75)	(2.01)
	0.50	0.15	397.35	376.20	421.71	39.90	15.74	14.29
			(19.83)	(24.79)	(41.32)	(2.85)	(1.49)	(1.16)
		0.25	559.28	550.82	579.71	35.86	15.85	13.09
			(15.57)	(18.55)	(27.67)	(4.00)	(2.77)	(1.98)
Case D	0.00	0.15	419.61	394.37	445.84	42.74	20.17	19.01
			(18.02)	(21.81)	(39.15)	(2.19)	(0.82)	(1.18)
		0.25	574.58	566.77	600.09	39.43	19.59	17.36
			(13.78)	(14.90)	(20.92)	(3.07)	(2.32)	(2.31)
	0.20	0.15	418.67	394.51	447.09	42.46	20.26	18.94
			(17.91)	(20.54)	(34.93)	(2.26)	(0.79)	(1.31)
		0.25	574.10	565.83	601.39	38.88	19.91	16.58
			(14.50)	(17.68)	(27.49)	(3.47)	(2.41)	(2.48)
	0.50	0.15	416.96	394.84	448.87	42.50	20.28	18.78
			(18.02)	(21.28)	(38.49)	(2.60)	(1.29)	(1.49)
		0.25	568.63	562.49	599.59	38.65	20.08	15.78
			(15.09)	(17.46)	(29.26)	(3.38)	(2.39)	(3.28)

corresponding intron ends with "AG". An overview of the splicing process and of some models used for detecting splice sites can be found in Burge (1998).

5.1 Dataset

MEMset Donor. This dataset consists of a training set of 8'415 true (encoded as $Y = 1$) and 179'438 false (encoded as $Y = 0$) human donor sites. An additional test set contains 4'208 true and 89'717 false donor sites. A sequence of a real splice site consists of the last 3 bases of the exon and the first 6 bases of the intron. False splice sites are sequences on the DNA which match the consensus sequence at position four and five. Removing the consensus "GT" results in a sequence length of 7 with values in $\{A, C, G, T\}^7$: thus, the predictor variables are 7 factors, each having 4 levels. The data is available at `http://genes.mit.edu/burgelab/maxent/ssdata/`. A more detailed description can be found in Yeo and Burge (2004).

The original training dataset is used to build a smaller balanced training dataset (5'610 true, 5'610 false donor sites) and an unbalanced validation set (2'805 true, 59'804 false donor sites). All sites are chosen randomly without replacement such that the two sets are disjoint. The additional test set remains unchanged. Note that the ratio of true to false sites are equal for the validation and the test set.

5.2 Procedure

All models are fitted on the balanced training dataset. As the ratio of true to false splice sites strongly differs from the training to the validation and the test set, the intercept is corrected as follows (King and Zeng, 2001):

$$\widehat{\beta}_0^{corr} = \widehat{\beta}_0 - \log\left(\frac{\bar{y}}{1-\bar{y}}\right) + \log\left(\frac{\pi}{1-\pi}\right),$$

where π is the proportion of true sites in the validation set.

Penalty parameters λ and κ are selected according to the (unpenalized) log-likelihood score on the validation set using the corrected intercept estimate.

For a threshold $\tau \in (0, 1)$ we assign observation i to class 1 if $p_{\widehat{\boldsymbol{\beta}}}(\boldsymbol{x}_i) > \tau$ and to class 0 otherwise. Note that the class assignment can also be constructed without intercept correction by using a different threshold.

The correlation coefficient ρ_τ corresponding to a threshold τ is defined as the pearson correlation between the binary random variable of the true class membership and the binary random variable of the predicted class membership. In Yeo and Burge (2004) the maximal correlation coefficient

$$\rho_{max} = \max\{\rho_\tau \,|\, \tau \in (0, 1)\}$$

is used as a goodness of fit statistics on the test set.

The candidate model used for the Logistic Group Lasso consists of all 3-way and lower order interactions involving 64 terms or $p = 1156$ parameters. In addition to the standard Logistic Group Lasso estimator, the hierarchical Group Lasso/Ridge hybrid and Group Lasso/MLE hybrid estimators are considered.

5.3 Results

The best model with respect to the log-likelihood score on the validation set is the Group Lasso estimator. It is followed by the Group Lasso/Ridge hybrid and the Group Lasso/MLE hybrid. The corresponding values of ρ_{max} on the test set are 0.6593, 0.6569 and 0.6541, respectively. They are all competitive with the results from Yeo and Burge (2004) whose best ρ_{max} equals 0.6589. While the Group Lasso solution has some active 3-way interactions, the Group Lasso/Ridge hybrid and the Group Lasso/MLE hybrid only contain 2-way interactions. Figure 3 shows the ℓ_2-norm of each parameter group for the three estimators. The 3-way interactions of the Group Lasso solution seem to be very weak. Considering also the non-hierarchical models for the two-stage procedures yields the same selected terms. Decreasing the candidate model size to only contain 2-way interactions gives similar results.

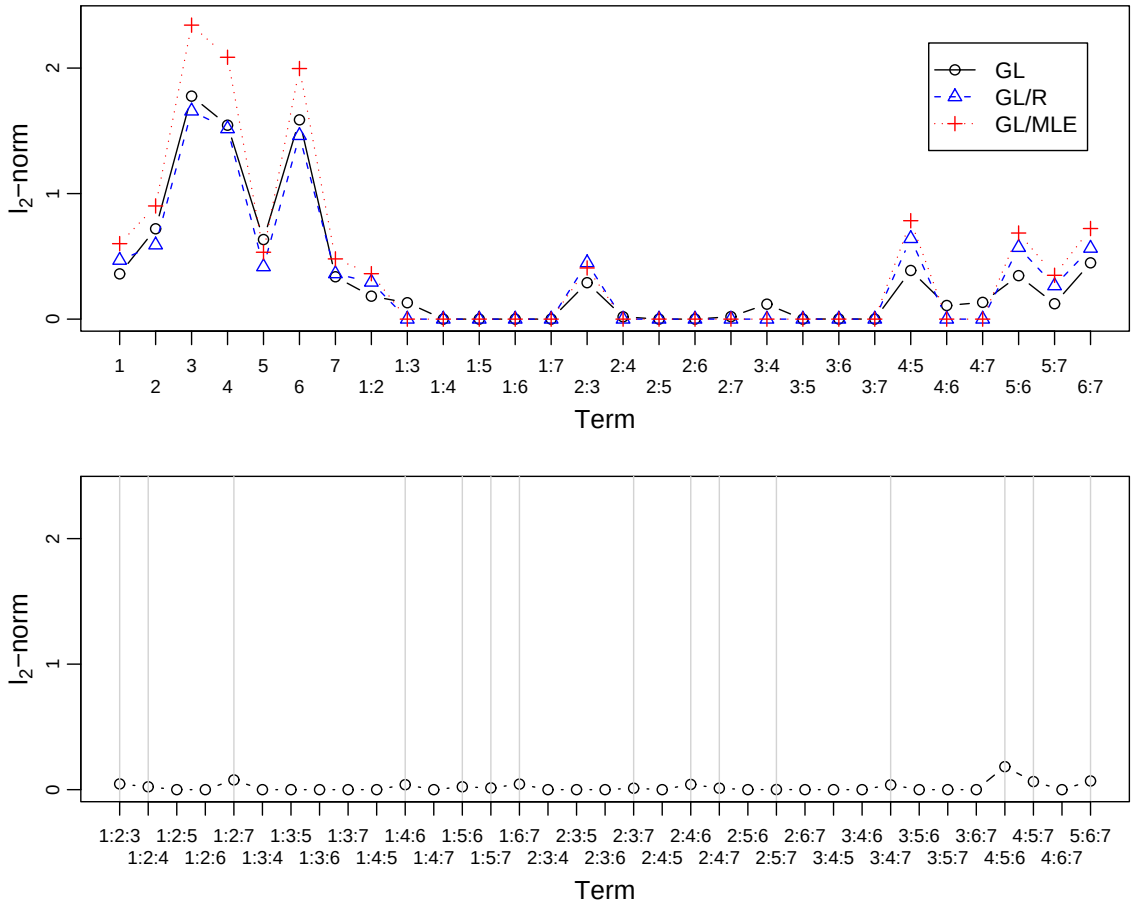

Figure 3: ℓ_2-norms $\|\widehat{\boldsymbol{\beta}}_g\|_2$, $g \in \{1, \ldots, G\}$ of the parameter groups with respect to the blockwise orthonormalized design matrix when using a candidate model with all 3-way interactions. $i:j:k$ denotes the 3-way interaction between the ith, jth and kth sequence position. The same scheme applies to the 2-way interactions and the main effects. Active 3-way interactions are additionally marked with vertical lines.

In summary, the prediction performance of the Group Lasso and its variants is competitive with the Maximum Entropy models used in Yeo and Burge (2004) which have been viewed

as (among) the best for short motif modeling and splice site prediction. Advantages of the Group Lasso and variants thereof include selection of terms. In addition, other (possibly continuous) predictor variables as for example global sequence information could be naturally included in the Group Lasso approach to improve the rather low correlation coefficients (Yeo and Burge, 2004).

6 Discussion

We study the Group Lasso for logistic regression. We present efficient algorithms, especially suitable for very high-dimensional problems, for solving the corresponding convex optimization problem which is inherently more difficult than ℓ_1-penalized logistic regression. The algorithms rely on recent theory and developments for block coordinate and block coordinate gradient descent (Tseng, 2001; Tseng and Yun, 2006). In contrast to the algorithm in Kim *et al.* (2006), our procedure is fully automatic and does not require the specification of an algorithmic tuning parameter to ensure convergence. Moreover, our algorithm is much faster than the recent proposal from Park and Hastie (2006). An implementation can be found in our R-package `grplasso`. Additionally, we present a statistical consistency theory for the setting where the predictor dimension is potentially much larger than sample size but assuming the true underlying logistic regression model is sparse. The algorithms with the supporting mathematical optimization theory as well as the statistical consistency theory also apply directly to the Group Lasso in other generalized linear models.

Furthermore, we propose the Group Lasso/Ridge hybrid method which often yields better predictions and better variable selection than the Group Lasso. In addition, our Group Lasso/Ridge hybrid allows for hierarchical model fitting.

Finally, we apply the Group Lasso and its variants to short DNA motif modeling and splice site detection. Our general methodology performs very well in comparison to the maximum entropy method which is considered to be among the best for this task.

Acknowledgement

We thank two anonymous referees and Sylvain Sardy for constructive comments.

A Appendix

A.1 Proof of Lemma 2.1

We will first eliminate the intercept. Let $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_G$ be fixed. To get the estimate for the intercept we have to minimize a function of the form

$$g(\beta_0) = -\sum_{i=1}^{n} \left[y_i \{\beta_0 + c_i\} - \log\{1 + \exp(\beta_0 + c_i)\} \right]$$

with derivative

$$g'(\beta_0) = -\sum_{i=1}^{n} \left\{ y_i - \frac{\exp(\beta_0 + c_i)}{1 + \exp(\beta_0 + c_i)} \right\},$$

where $c_i = \sum_{g=1}^{G} \boldsymbol{x}_{i,g}^T \boldsymbol{\beta}_g$ is a constant. It holds that $\lim_{\beta_0 \to \infty} g'(\beta_0) = n - \sum_{i=1}^{n} y_i > 0$ and $\lim_{\beta_0 \to -\infty} g'(\beta_0) = -\sum_{i=1}^{n} y_i < 0$. Furthermore $g'(\cdot)$ is continuous and strictly increasing. Therefore there exists a unique $\beta_0^* \in \mathbb{R}$ such that $g'(\beta_0^*) = 0$. By the implicit function theorem the corresponding function $\beta_0^*(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_G)$ is continuously differentiable. By replacing β_0 in $S_\lambda(\boldsymbol{\beta})$ by the function $\beta_0^*(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_G)$ and using duality theory, we can rewrite (2.2) as an optimization problem under the constraint $\sum_{g=1}^{G} \|\boldsymbol{\beta}_g\|_2 \leq t$ for some $t > 0$. This is an optimization problem of a continuous function over a compact set, hence the minimum is attained.

A.2 Proof of Proposition 2.1

We first show that the groupwise minima of S_λ are attained. For $g = 0$ this follows from the proof of Lemma 2.1. The case $g \geq 1$ corresponds to a minimization of a continuous function over a compact set, hence the minimum is attained. We now show that step (3) minimizes the convex function $S_\lambda(\boldsymbol{\beta}_g)$ for $g \geq 1$. Since $S_\lambda(\boldsymbol{\beta}_g)$ is not differentiable everywhere, we invoke subdifferential calculus (Bertsekas, 2003). The subdifferential of S_λ with respect to $\boldsymbol{\beta}_g$ is the set $\partial S_\lambda(\boldsymbol{\beta}_g) = \{-X_g^T(\boldsymbol{y} - \boldsymbol{p}_{\boldsymbol{\beta}}) + \lambda \boldsymbol{e}, \boldsymbol{e} \in E(\boldsymbol{\beta}_g)\}$, $E(\boldsymbol{\beta}_g) = \{\boldsymbol{e} \in \mathbb{R}^{df_g} : \boldsymbol{e}_g = s(df_g)\frac{\boldsymbol{\beta}_g}{\|\boldsymbol{\beta}_g\|_2}$ if $\boldsymbol{\beta}_g \neq \mathbf{0}$ and $\|\boldsymbol{e}_g\|_2 \leq s(df_g)$ if $\boldsymbol{\beta}_g = \mathbf{0}\}$. The parameter vector $\boldsymbol{\beta}_g$ minimizes $S_\lambda(\boldsymbol{\beta}_g)$ if and only if $\mathbf{0} \in \partial S_\lambda(\boldsymbol{\beta}_g)$ which is equivalent to the formulation of step 3. Furthermore conditions (A1), (B1) - (B3) and (C2) in Tseng (2001) hold. By Lemma 3.1 and Proposition 5.1 in Tseng (2001) every limit point of the sequence $\{\widehat{\boldsymbol{\beta}}^{(t)}\}_{t \geq 0}$ is a stationary point of the convex function S_λ, hence a minimum point.

A.3 Proof of Proposition 2.2

The proposition directly follows from Theorem 4.1(e) of Tseng and Yun (2006). We have to show that $-H^{(t)}$ is bounded by above and away from zero. The Hessian of the negative log-likelihood function is $N = \sum_{i=1}^{n} p_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\{1 - p_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\}\boldsymbol{x}_i \boldsymbol{x}_i^T \preceq \frac{1}{4} X^T X$ in the sense that $N - \frac{1}{4} X^T X$ is negative semidefinite. For the blockmatrix N_{gg} corresponding to the gth predictor it follows from the blockwise orthonormalization that $N_{gg} \preceq \frac{n}{4} I_{df_g}$ and hence $\max\{\text{diag}(N_{gg})\} \leq \frac{n}{4}$. An upper bound on $-H^{(t)}$ is therefore always guaranteed. The lower bound is enforced by the choice of $H_{gg}^{(t)}$. By the choice of the line search we ensure that $\alpha^{(t)}$ is bounded by above and therefore Theorem 4.1(e) of Tseng and Yun (2006) can be applied.

A.4 Outline of the Proof of the Consistency Result

The proof follows the arguments used in Tarigan and van de Geer (2006). The latter consider hinge loss instead of logistic loss, but, as they point out, a large part of their results can be easily extended because only the Lipschitz property of the loss is used there. Furthermore, under (A1), logistic loss has the usual "quadratic" behaviour near its overall minimum. This means that it does not share the problem of unknown margin behaviour with hinge loss, i.e. the situation is in that respect simpler than in Tarigan and van de Geer (2006).

The Group Lasso reduces to the ℓ_1 penalty (the usual Lasso) when there is only one degree of freedom in each group. The extension of consistency results to more degrees of freedom is straightforward, provided $\max_g df_g$ does not depend on n. We furthermore note that the Group Lasso uses a normalization involving the design matrix of the observed predictors. For the consistency result, one needs to prove that this empirical normalization is uniformly close to the theoretical one. This boils down to showing that empirical and theoretical eigenvalues of the design matrix per group cannot be too far away from each other, uniformly over the groups. Here, we invoke that assumption (A4) implies that L_n^2 is no larger than $c_1/(C_1 \log G)$. We then apply (A3) in Bernstein's inequality to bound the difference in eigenvalues.

A technical extension as compared to Tarigan and van de Geer (2006) is that we do not assume an a priori bound on the functions $\eta_{\boldsymbol{\beta}}(\cdot)$. This is now handled by using convexity arguments (similar to van de Geer (2003)), and again part of the assumption (A4), namely that λ is smaller than $c_1/((1+N_0^2)L_n^2)$. This assumption ensures that for all n, with high probability, the difference between $\eta_{\boldsymbol{\beta}^0}$ and the estimated regression $\eta_{\widehat{\boldsymbol{\beta}}_\lambda}$ is bounded by a constant independent of n.

References

Antoniadis, A. and Fan, J. (2001) Regularization of wavelet approximations (with discussion). *Journal of the American Statistical Assocation*, **96**, 939 – 967.

Bakin, S. (1999) *Adaptive Regression and Model Selection in Data Mining Problems*. Ph.D. thesis, Australian National University.

Balakrishnan, S. and Madigan, D. (2006) Algorithms for sparse linear classifiers in the massive data setting. Available at `http://www.stat.rutgers.edu/~madigan/PAPERS/`.

Bertsekas, D. P. (2003) *Nonlinear Programming*. Athena Scientific.

Burge, C. (1998) Modeling dependencies in pre-mrna splicing signals. In *Computatational Methods in Molecular Biology* (eds. S. Salzberg, D. Searls and S. Kasif), chap. 8, 129–164. Elsevier Science.

Burge, C. and Karlin, S. (1997) Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, **268**, 78–94.

Cai, T. T. (2001) Discussion of "Regularization of wavelet approximations" (auths. Antoniadis, A. and Fan, J.). *Journal of the American Statistical Assocation*, **96**, 960–962.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004) Least angle regression. *The Annals of Statistics*, **32**, 407–499.

Genkin, A., Lewis, D. D. and Madigan, D. (2004) Large-scale bayesian logistic regression for text categorization. Available at `http://www.stat.rutgers.edu/~madigan/PAPERS/`.

Kim, Y., Kim, J. and Kim, Y. (2006) Blockwise sparse regression. *Statistica Sinica*, **16**.

King, G. and Zeng, L. (2001) Logistic regression in rare events data. *Political Analysis*, **9**, 137–163.

Krishnapuram, B., Carin, L., Figueiredo, M. A. and Hartemink, A. J. (2005) Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**, 957–968.

Lokhorst, J. (1999) The lasso and generalised linear models. Honors Project, The University of Adelaide, Australia.

Meinshausen, N. (2007) Lasso with relaxation. *Computational Statistics and Data Analysis*. To appear.

Osborne, M., Presnell, B. and Turlach, B. (2000) A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, **20**, 389–403.

Park, M.-Y. and Hastie, T. (2006) Regularization path algorithms for detecting gene interactions. Available at `http://www-stat.stanford.edu/~hastie/pub.htm`.

Park, M.-Y. and Hastie, T. (2007) An l1 regularization-path algorithm for generalized linear models. *Journal of the Royal Statistical Society Series B*. To appear.

Rosset, S. (2005) Following curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems 17* (eds. L. K. Saul, Y. Weiss and L. Bottou), 1153–1160. Cambridge, MA: MIT Press.

Roth, V. (2004) The generalized lasso. *IEEE Transactions on Neural Networks*, **15**, 16–28.

Shevade, S. and Keerthi, S. (2003) A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, **19**, 2246–2253.

Tarigan, B. and van de Geer, S. (2006) Classifiers of support vector machine type with ℓ_1 complexity regularization. *Bernoulli*, **12**, 1045–1076.

Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, **58**, 267–288.

Tibshirani, R. (1997) The lasso method for variable selection in the cox model. *Statistics in Medicine*, **16**, 385–395.

Tseng, P. (2001) Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, **109**, 475–494.

Tseng, P. and Yun, S. (2006) A coordinate gradient descent method for nonsmooth separable minimization. Preprint.

van de Geer, S. (2003) Adaptive quantile regression. In *Recent Advances and Trends in Nonparametric Statistics* (eds. M. Akritas and D. Politis), 235–250. Elsevier.

Yeo, G. W. and Burge, C. B. (2004) Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *Journal of Computational Biology*, **11**, 475–494.

Yuan, M. and Lin, Y. (2006) Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B*, **68**, 49–67.

Zhao, P. and Yu, B. (2004) Boosted lasso. Tech. rep., University of California, Berkeley.