L₁ Penalized Estimation in the Cox Proportional Hazards Model

Jelle J. Goeman

Department of Medical Statistics and Bioinformatics, Leiden University Medical Center, Leiden, The Netherlands

Received 29 January 2009, revised 28 May 2009, accepted 10 June 2009

This article presents a novel algorithm that efficiently computes L₁ penalized (lasso) estimates of parameters in high-dimensional models. The lasso has the property that it simultaneously performs variable selection and shrinkage, which makes it very useful for finding interpretable prediction rules in high-dimensional data. The new algorithm is based on a combination of gradient ascent optimization with the Newton–Raphson algorithm. It is described for a general likelihood function and can be applied in generalized linear models and other models with an L₁ penalty. The algorithm is demonstrated in the Cox proportional hazards model, predicting survival of breast cancer patients using gene expression data, and its performance is compared with competing approaches. An R package, penalized, that implements the method, is available on CRAN.

Key words: Gradient ascent; Lasso; Penalty; Survival.

1 Introduction

The lasso (Tibshirani, 1996, 1997) is a regularized estimation approach for regression models that constrains the L₁ norm of the regression coefficients. There are two alternative definitions. The first defines the lasso estimates \( \hat{\beta} \) of the vector of regression coefficients \( \beta \) in terms of constrained likelihood optimization as

\[
\hat{\beta} = \text{argmax}_{\beta} \ell(\beta), \quad \text{subject to } ||\beta||_1 \leq s,
\]

(1)

where \( \ell \) is the log likelihood of the model, and \( || \cdot ||_1 \) is the L₁ norm. The second definition defines \( \hat{\beta} \) in terms of penalized likelihood optimization as

\[
\hat{\beta} = \text{argmax}_{\beta} \{ \ell(\beta) - \lambda ||\beta||_1 \}.
\]

(2)

For a given likelihood function these two definitions are equivalent, as Eq. (2) can be constructed as the Lagrange multiplier version of the optimization problem (1). The definition (2) has an additional Bayesian interpretation as the mode of the posterior distribution of coefficients \( \beta \) if a priori each coefficient was independently double exponentially distributed with common parameter \( \lambda \).

The lasso is attractive as a regularization method because it simultaneously performs variable selection and shrinkage. It shrinks all regression coefficients toward zero and automatically sets many of them exactly to zero, depending on the amount of regularization employed. This can be especially useful in high-dimensional data, in which there are more regression coefficients than observations. In this case strong variable selection is desirable in order to obtain an interpretable prediction rule, and shrinkage is desirable to prevent overfit.

Practical application of the lasso in high-dimensional applications requires the availability of efficient algorithms to solve either Eq. (1) or Eq. (2). Computation time is especially important when the tuning parameters are to be chosen via cross-validation. For linear models, Tibshirani (1996)
originally proposed quadratic programming to solve Eq. (1). Osborne, Presnell, and Turlach (2000) described an algorithm based on solving the dual optimization problem, which could also handle high-dimensional situations. Efron et al. (2004) proposed the LARS algorithm, which simultaneously solves Eqs. (1) and (2) for all values of the tuning parameters s and λ, by tracing the solution of \( \hat{\beta} \) as a function of λ and/or s.

Optimizing Eq. (1) or Eq. (2) for the Cox proportional hazards model or for other generalized linear models is computationally much more demanding. Tibshirani (1997) originally proposed a method to solve Eq. (1) by alternating weighted least squares to find the unconstrained estimates and linear programming to adapt to the constraint. This alternating scheme was refined by Gui and Li (2005), who suggested to use an iteratively reweighted LARS algorithm. Both of these algorithms were computationally too demanding to be used in high-dimensional data, as was pointed out by Segal (2006).

Two competing approaches seem more promising. The coordinatewise gradient approach, represented by Shevade and Keerthi (2003), Kim and Kim (2004) and Genkin, Lewis, and Madigan (2007), is to optimize Eq. (2) directly from a starting value of \( \beta \) by cycling through the coordinates of \( \beta \) and sequentially updating them on the basis of the gradient of the penalized likelihood. The path approach, on the other hand, by Keerthi and Shevade (2007) and Park and Hastie (2007), directly generalizes the LARS algorithm (Efron et al., 2004) to generalized linear models by following the solution path of \( \hat{\beta} \) as a function of λ.

This article presents a novel full gradient algorithm for maximizing the lasso-penalized likelihood (2). Like the coordinatewise gradient approaches of Shevade and Keerthi (2003), Kim and Kim (2004) and Genkin et al. (2007), it follows the gradient of the likelihood from a given starting value of \( \beta \). However, unlike these approaches, it does not update a single coordinate at a time, but it uses the full gradient at each step. Furthermore, the algorithm can automatically switch to a Newton–Raphson algorithm when it gets close to the optimum, thus avoiding the tendency to slow convergence of gradient ascent algorithms (Kim and Kim, 2004). Although a gradient approach, the algorithm described in this article can easily be combined with the philosophy of the path-based approaches by sequentially calculating \( \hat{\beta} \) for successive values of λ. A great advantage here over the pure path algorithms is the complete freedom to choose the step size.

The core of the algorithm will be described in Section 2 for the general situation of optimizing Eq. (2) for any concave and twice-differentiable likelihood function \( \ell(\beta) \). The refinement of automatic switching to the Newton–Raphson algorithm is explained in Section 3, and in Section 4 the application of the algorithm is illustrated on a data set of gene expression measurement of breast cancer patients. Next, in Section 5, the issue of gradient versus path algorithms is discussed in the context of efficient cross-validation. Section 6 compares the performance of the algorithm with that of the path algorithm of Park and Hastie (2007). Finally, Section 7 discusses extensions of the algorithm to the elastic net (Zou and Hastie, 2005) and to situations in which coefficients are constrained to be positive.

## 2 Gradient Ascent

To understand why gradient ascent, which is generally considered an inefficient optimization method, may be the method of choice for lasso estimation, it is worthwhile to look more closely into the penalized likelihood function that is to be optimized.

Writing \( \beta = (\beta_1, \ldots, \beta_p)^T \), the target function

\[
\ell_{\text{pen}}(\beta) = \ell(\beta) - \lambda \sum_{i=1}^{p} |\beta_i|
\]


© 2009 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim www.biometrical-journal.com
is given by a sum of two terms. The first term is the log likelihood \( \ell(\beta) \). In the models in which we are interested this is a highly regular function: concave and everywhere at least twice differentiable. The second term, the penalty \( P(\beta) = -\lambda \sum_{j=1}^{p} |\beta_j| \) is less well behaved: it is concave and continuous, but is only differentiable at points with \( |\beta_j| \neq 0 \) for all \( j \).

It should first be noted that the penalized log likelihood function, being the sum of two concave functions, is itself a convex function. It is not generally strictly concave, however, as there may be regions in which both the log likelihood and the penalty are not strictly concave. In some cases this may result in a function \( \ell_{\text{pen}}(\beta) \) with a “flat top”: a contiguous optimum consisting of more than a single point. This may happen in general when any covariate is a convex combination of the others, the classical example being a duplicated covariate. The penalized likelihood function may also display very weak concavity near the optimum, especially if \( \lambda \) is small, which can be a major convergence problem for lasso algorithms in general (see also Section 6).

Second, it should be noted that the penalized likelihood is not differentiable everywhere due to the lack of differentiability of the penalty function. Still, it is possible to define a directional derivative

\[
\ell'_{\text{pen}}(\beta; \mathbf{v}) = \lim_{t \to 0} \frac{1}{t} \left[ \ell_{\text{pen}}(\beta + t \mathbf{v}) - \ell_{\text{pen}}(\beta) \right]
\]

for every point \( \beta \) in every direction \( \mathbf{v} \in \mathbb{R}^p \). The gradient can then be defined for every \( \beta \) as the scaled direction of steepest ascent. Let \( \mathbf{v}_{\text{opt}} \) be the direction that maximizes \( \ell'_{\text{pen}}(\beta; \mathbf{v}) \) among all \( \mathbf{v} \) with \( \|\mathbf{v}\| = 1 \), then the gradient can be defined as

\[
g(\beta) = \begin{cases} 
\ell'_{\text{pen}}(\beta; \mathbf{v}_{\text{opt}}) \cdot \mathbf{v}_{\text{opt}} & \text{if } \ell'_{\text{pen}}(\beta; \mathbf{v}_{\text{opt}}) \geq 0 \\
0 & \text{otherwise},
\end{cases}
\]

where \( \mathbf{0} \) is a \( p \)-vector of zeros. Note that due to the concavity of \( \ell_{\text{pen}} \) there is at most one \( \beta \) with maximal directional derivative \( \ell'_{\text{pen}}(\beta; \mathbf{v}_{\text{opt}}) < 0 \), although there may be a contiguous area of points which have \( \ell'_{\text{pen}}(\beta; \mathbf{v}_{\text{opt}}) = 0 \) if the target function is not strictly concave at the optimum.

The gradient \( g(\beta) = (g_1(\beta), \ldots, g_p(\beta))^t \) can be calculated from the unpenalized log likelihood gradient \( h(\beta) = \partial \ell(\beta)/\partial \beta = (h_1(\beta), \ldots, h_p(\beta))^t \) as

\[
g(\beta) = \begin{cases} 
h(\beta) - \lambda \cdot \text{sign}(\beta) & \text{if } \beta_i \neq 0 \\
h(\beta) - \lambda \cdot \text{sign}(h(\beta)) & \text{if } \beta_i = 0 \text{ and } |h(\beta)| > \lambda
\end{cases}
\]

otherwise,

where

\[
\text{sign}(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x = 0 \\
-1 & \text{if } x < 0.
\end{cases}
\]

This gradient is discontinuous at every point at which the penalized log likelihood is not differentiable, i.e. at every point with \( \beta_i = 0 \) for some \( i \).

Analogous to the directional derivative we can define the directional second derivative

\[
\ell''_{\text{pen}}(\beta; \mathbf{v}) = \lim_{t \to 0} \frac{1}{t^2} \left[ \ell''_{\text{pen}}(\beta + t \mathbf{v}; \mathbf{v}) - \ell''_{\text{pen}}(\beta; \mathbf{v}) \right],
\]

even where the Hessian matrix is not defined. For the penalized log likelihood the directional second derivative is given for every \( \beta \) and \( \mathbf{v} \) by

\[
\ell''_{\text{pen}}(\beta; \mathbf{v}) = \mathbf{v}^t \frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^t} \mathbf{v}.
\]

In practice it is hardly ever necessary to calculate the full \( p \times p \) Hessian matrix of \( \ell(\beta) \) to calculate the directional second derivative, as the direction \( \mathbf{v} \) of interest, which is the direction of the gradient, will typically have many zeros. Furthermore, in the Cox proportional hazards model, as well as in
generalized linear models with a canonical link function, the Hessian matrix is of the form

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = X^T WX$$

where $X$ is an $n \times p$ design matrix and $W$ an $n \times n$ weights matrix. This structure of the Hessian matrix allows the algorithm to avoid construction of the full $p \times p$ Hessian when calculating $\ell''_{\text{pen}}(\beta; v) = (Xv)^T W (Xv)$ in such generalized linear models.

The discontinuities of the gradient divide the domain of $\ell_{\text{pen}}$ into $3^p$ subdomains on which the gradient is continuous. The gradient and the directional second derivative only describe the local behavior of $\ell_{\text{pen}}$ within such a subdomain. Therefore, the information in the gradient can only be used in a series of highly local optimization steps. This makes the gradient information suitable for use in a gradient ascent algorithm, but not for many other optimization algorithms.

The gradient ascent algorithm uses a series of Taylor approximations. At each step it approximates $\ell_{\text{pen}}$ locally from $\beta$ in the direction of the gradient by a directional second order Taylor approximation, given by

$$\ell_{\text{pen}}(\beta + tg(\beta)) \approx \ell_{\text{pen}}(\beta) + t\ell'(\beta; g(\beta)) + \frac{1}{2} t^2 \ell''_{\text{pen}}(\beta; g(\beta)).$$

This approximation is meaningful only within a single subdomain of gradient continuity, i.e. for $0 < t < t_{\text{edge}}$, with

$$t_{\text{edge}} = \min_i \left\{ -\frac{\beta_i}{g_i(\beta)} : \text{sign}(\beta_i) = -\text{sign}(g_i(\beta)) \neq 0 \right\}.$$

The optimum of the Taylor approximation in the subdomain is at

$$t_{\text{opt}} = -\frac{\ell'(\beta; g(\beta))}{\ell''_{\text{pen}}(\beta; g(\beta))}$$

provided $t_{\text{opt}} < t_{\text{edge}}$, otherwise it is at $t_{\text{edge}}$. The algorithm proceeds in every next step with a new directional Taylor approximation from the optimum found in the previous one (see Box 1). Convergence occurs when $g(\beta) = 0$. If there is not a unique optimum, the algorithm will converge to a point in the optimal area.

Box 1. A simple gradient ascent algorithm for $L_1$ penalized likelihood.

**Start with some $\beta^{(0)}$. For steps $i = 0, 1, 2, \ldots$ of the algorithm, iterate**

$$\beta^{(i+1)} = \beta^{(i)} + \min(t_{\text{opt}}, t_{\text{edge}}) g(\beta^{(i)})$$

**until convergence.**

### 3 Newton–Raphson Algorithm

An important advantage of the algorithm described in the previous section is its great computational simplicity. Once the gradient and Hessian of the unpenalized log likelihood are known for $\beta^{(0)}$, finding $\beta^{(i+1)}$ involves only a few of the most basic calculations. In particular, it involves no matrix inversion or other computationally expensive matrix operations. However, the algorithm in the simple form presented in Box 1 tends to require a large number of steps to converge. Part of these steps can be avoided by giving the algorithm the option of switching to the Newton–Raphson algorithm.

It is well known that the gradient ascent algorithm converges much slower than the Newton–Raphson algorithm in situations in which both can be applied, which is when the target function is concave and twice differentiable. We can exploit this knowledge if we view the basic gradient ascent algorithm of Box 1 as a series of optimizations, each constrained to a single
subdomain of gradient continuity. There are 3\(^p\) such subdomains, each defined by constant
\[ \text{sign}(\mathbf{b}) = (\text{sign}(\mathbf{b}_1), \ldots, \text{sign}(\mathbf{b}_p))^T, \]
in which the gradient is continuous as a function of \( \mathbf{b} \). Each optimization step of the gradient ascent algorithm takes place within the subdomain
\[ \text{sign}(\mathbf{b}_i) = \lim_{\epsilon \to 0} \text{sign}(\mathbf{b} + \epsilon \mathbf{g}(\mathbf{b})). \]
Constrained to a single subdomain we can view the target function \( \ell_{\text{pen}} \) as an \( m \)-dimensional function, where \( m = 2J \leq p \), and \( J = \{ j : \text{sign}(\mathbf{b}_j) \neq 0 \} \) is the index set of the “active” covariates.

At most one of the 3\(^p\) subdomains has an interior optimum. Locally, in a neighborhood around this optimum, the target function \( \ell_{\text{pen}} \) is nicely twice differentiable and concave. This means that locally around the optimum the conditions for applying the Newton–Raphson algorithm are met. In view of the efficiency of the Newton–Raphson algorithm, it may, therefore, be worthwhile to switch to the Newton–Raphson algorithm once the gradient ascent has brought the current solution close to the true optimum. A good indication of proximity to the true optimum is that \( t_{\text{opt}} < t_{\text{edge}} \).

This observation leads to the following modification of the algorithm. Let \( \mathbf{\tilde{b}} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)^T \), and let \( \mathbf{g}(\mathbf{b}) = (g_1(\mathbf{b}), \ldots, g_m(\mathbf{b}))^T \) be the gradient in the constrained domain and \( \mathbf{H} \) the \( m \times m \) Hessian of the constrained optimization, given by

\[
\mathbf{H}_{ijk}(\mathbf{b}) = \frac{\partial^2 \ell(\mathbf{b})}{\partial \mathbf{b}_i \partial \mathbf{b}_j} \quad k = 1, \ldots, m, \ l = 1, \ldots, m.
\]

A step of the Newton–Raphson algorithm in the current subdomain would propose

\[
\mathbf{\tilde{b}}^{(i+1)} = \mathbf{\tilde{b}}^{(i)} - \mathbf{H}(\mathbf{\tilde{b}}^{(i)})^{-1}\mathbf{g}(\mathbf{\tilde{b}}^{(i)}).
\]

This proposal can be mapped back to a \( p \)-vector \( \mathbf{p}_{\text{NR}}^{(i+1)} \) by augmenting \( \mathbf{\tilde{b}}^{(i+1)} \) with zeros for all inactive covariates. As the Taylor approximation underlying such a Newton–Raphson step is valid only within the current subdomain, we should accept \( \mathbf{p}_{\text{NR}}^{(i+1)} \) only when \( \text{sign}(\mathbf{p}_{\text{NR}}^{(i+1)}) = \text{sign}(\mathbf{\tilde{b}}^{(i)}) \).

Incorporation of the Newton–Raphson extension leads to the algorithm given in Box 2.

### Box 2. A gradient ascent algorithm for \( L_1 \) penalized likelihood with Newton–Raphson steps.

Start with some \( \mathbf{p}^{(0)} \). For steps \( i = 0, 1, 2, \ldots \) of the algorithm, iterate

\[
\mathbf{p}^{(i+1)} = \begin{cases} 
\mathbf{p}^{(i)} + t_{\text{edge}} \mathbf{g}(\mathbf{p}^{(i)}) & \text{if } t_{\text{opt}} \geq t_{\text{edge}} \\
\mathbf{p}^{(i+1)}_{\text{NR}} & \text{if } t_{\text{opt}} < t_{\text{edge}} \text{ and } \text{sign}(\mathbf{p}_{\text{NR}}^{(i+1)}) = \text{sign}(\mathbf{p}^{(i)}) \\
\mathbf{p}^{(i)} + t_{\text{opt}} \mathbf{g}(\mathbf{p}^{(i)}) & \text{otherwise}
\end{cases}
\]

until convergence.

Some minor amendments may be still made to the basic algorithm of Box 2. Practical experience has indicated that it is not recommended to try a Newton–Raphson step in step \( i = 0 \), especially if \( \mathbf{p}^{(0)} = \mathbf{0} \), nor, after a Newton–Raphson step has failed, to try another Newton–Raphson step before the active set has changed. As trying a Newton–Raphson step can be computationally expensive, it is often better to avoid the attempt whenever it is likely to fail. However, once the algorithm gets close to the optimum, the Newton–Raphson steps will greatly speed up convergence.

The algorithm can also be easily amended to the situation in which some parameters should not be penalized or in which some parameters should be penalized more than others. In this case, we
may write the penalized likelihood as

$$
\ell_{\text{pen}}(\beta) = \ell(\beta) - \sum_{i=1}^{p} \lambda_i |\beta_i|,
$$

so that each parameter has its own individual value of $\lambda_i$. Unpenalized parameters can get $\lambda_i = 0$.

## 4 Application

As our primary interest is in survival analysis, the above algorithm was applied to maximization of the penalized partial likelihood in the Cox proportional hazards model. We analyze the publicly available breast cancer gene expression data set from the article of Van de Vijver et al. (2002), which is a follow-up to the well-known article by Van’t Veer et al. (2002) in which a 70-gene signature for prediction of metastasis-free survival in breast cancer was proposed. An alternative predictor was constructed by Van Houwelingen et al. (2006) using ridge regression in a Cox proportional hazards model. A preliminary test of the global null hypothesis of no predictive ability of the gene expression measurements was strongly rejected ($p < 0.00001$, Goeman et al., 2005), indicating potential for predicting survival with penalized methods in this data set.

The data set consists of cDNA expression profiles of 295 tumor samples of breast cancer patients diagnosed between 1984 and 1995 at the Netherlands Cancer Institute, and aged 52 years or younger at the time of diagnosis. The median follow-up was 6.7 years (range 0.05–18.3). Following Van Houwelingen et al. (2006), we define survival (i.e. time until death) as the event of interest in the Cox proportional hazards model. The data set has 79 events and 216 censored observations. Of the 24,885 genes on the cDNA chip, 4919 were selected on the basis of quality criteria as in Van’t Veer et al. (2002). The gene expression measurements were normalized with the Rosetta error model as in Van de Vijver et al. (2002).

We used cross-validation to calculate the optimal value of the tuning parameter $\lambda$. The reason to prefer cross-validation over less computationally intensive alternatives such as AIC, BIC or generalized cross-validation is that, unlike the latter methods, cross-validation does not require the estimation of the effective dimension of the model. Although useful proposals have been forwarded (Zou, Hastie, and Tibshirani 2007), the applicability and interpretation of the effective dimension in the lasso context are far from straightforward.

In Fig. 1 the left-hand panel shows the effect of the choice of the tuning parameter $\lambda$ on the cross-validated partial likelihood (Verweij and Van Houwelingen, 1993), calculated on the basis of leave-one-out cross-validation. The function shows a clear optimum at $\lambda = 7.702$, indicating that there is predictive potential in the gene expression data. The right-hand panel of Fig. 1 gives the number of estimated non-zero regression coefficients as a function of $\lambda$. At the optimal cross-validated $\lambda$, the model is quite sparse with only 16 active predictors.

The two graphs of Fig. 1 also point to important caveats of fitting $L_1$ penalized models. In the first place, the plot of the cross-validated partial likelihood has several local modes. This is a typical feature of lasso fits, which occurs with all optimality criteria. Therefore, it is important always to check that an optimal $\lambda$ found is in fact the global and not a local optimum. Second, the plot of the number of non-zero regression coefficients shows that, although the severity of the penalization increases with $\lambda$ by definition, the number of non-zero regression coefficients is not necessarily monotone as a function of $\lambda$.

It is interesting to note that the optimal cross-validated partial likelihood of $-479.45$ is not as high as the optimal cross-validated partial likelihood of $-476.17$ found for Cox ridge regression on this data set by Van Houwelingen et al. (2006). This may indicate that the “true” prediction rule for this data is not as sparse as the lasso model would have it, and that the sparseness imposed on the prediction rule by the $L_1$ penalization can come at a price of reduced predictive performance. However, a ridge regression predictor with thousands of non-zero regression coefficients can be difficult to interpret and awkward to work with. A small loss of predictive ability can be a price worth paying for an important gain in simplicity. A similar
conclusion was reached by Bøvelstad et al. (2007) in an excellent review of survival prediction methods on several breast cancer gene expression data sets. They showed an overall best predictive performance of ridge regression, but a best performance of the lasso among prediction methods with a variable selection component. Similar results were obtained by van Wieringen et al. (2009).

The sixteen predictive genes with non-zero regression coefficients are summarized in Table 1. Interestingly, no less than three of these sixteen genes are also part of the 70-gene signature found by Van’t Veer et al. (2002) on a subset of the data set used here. Several of the genes found have known functions in cancer-related pathways, whereas others have not been identified as predictive for survival in breast cancer before.

Although the lasso estimates are certainly more interpretable than their ridge regression counterparts, the interpretation of lasso coefficients should still be approached with great care, especially in high-dimensional data such as these. Lasso estimates are highly biased by construction: the lasso favors high-variance predictors over low-variance ones; it favors predictors that are hardly correlated with other predictors over ones that are highly correlated. As a consequence, both the selected set of covariates and the magnitude of their estimated effects are subject to the interpretation difficulties: it is often impossible to disentangle the effect of the bias (i.e. the penalty) on the fitted coefficients from the effect of the data (i.e. the likelihood).

In Fig. 2 the left-hand panel gives survival curves as predicted by the fitted model for various percentiles of the prognostic index \( X^b \). For comparison, and to check for problems in the fit of the model, the right-hand panel gives Kaplan–Meier curves for four subgroups of the 295 patients. The subgroups have been chosen by the percentiles of the double cross-validated 5-year survival probability, i.e. the survival probability predicted for each patient from a model fitted on the data set reduced to the other 294 patients, using a value of the tuning parameter that was determined using cross-validated likelihood on only these 294 patients. The Kaplan–Meier for the full data is added for reference. The plots give no clear indication of overfit or of violation of important model assumptions such a proportionality of the hazards.

5 Gradient, Path and Cross-validation

The algorithm described above has been designed to find estimates of regression coefficients for a single fixed value of \( \lambda \). In some situations, however, it can be interesting to fit the model for a range

![Figure 1](image-url)
of values of λ, for example to optimize λ over a suitable criterion such as cross-validated likelihood or classification error. In such situations an approach that finds the solution path of \( \hat{\beta} \) as a function of \( \lambda \) may be preferable to a direct algorithm, as argued by Keerthi and Shevade (2007) and Park and Hastie (2007). This section compares path-based and gradient-based algorithms in the context of the discussion in the literature.

In the linear regression model the solution path \( \hat{\beta}(\lambda) \) is piecewise linear. For that specific model, the LARS algorithm (Efron et al., 2004) traces this path exactly, simultaneously solving the lasso optimization problem for all values of \( \lambda \). In general, however, the solution path \( \hat{\beta}(\lambda) \) is not piecewise

<table>
<thead>
<tr>
<th>Gene identifier</th>
<th>Symbol</th>
<th>( \hat{\beta} )</th>
<th>Among 70</th>
<th>Known functions (Gene Ontology)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM 000909</td>
<td>NPY1R</td>
<td>(-0.103)</td>
<td>Signal transduction, locomotory behavior, feeding behavior, perception of pain and regulation of body size</td>
<td></td>
</tr>
<tr>
<td>NM 006198</td>
<td>PCP4</td>
<td>0.046</td>
<td>Central nervous system development</td>
<td></td>
</tr>
<tr>
<td>Contig 48328 RC</td>
<td>ZNF533</td>
<td>(-0.113)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NM 020974</td>
<td>SCUBE2</td>
<td>(-0.217)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NM 006398</td>
<td>UBD</td>
<td>(-0.200)</td>
<td>Proteolysis</td>
<td></td>
</tr>
<tr>
<td>NM 006419</td>
<td>CXCL13</td>
<td>(-0.117)</td>
<td>Cell-cell signaling, inflammatory response</td>
<td></td>
</tr>
<tr>
<td>Contig 55725 RC</td>
<td>CDCA7</td>
<td>0.305</td>
<td>Yes</td>
<td>Cell proliferation, transcription</td>
</tr>
<tr>
<td>Contig 56390 RC</td>
<td>SUSD3</td>
<td>(-0.220)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NM 000353</td>
<td>TAT</td>
<td>(-0.198)</td>
<td>Amino acid metabolic process</td>
<td></td>
</tr>
<tr>
<td>NM 001085</td>
<td>SERPINA3</td>
<td>(-0.161)</td>
<td>Inflammatory response, lipid metabolism</td>
<td></td>
</tr>
<tr>
<td>L27560</td>
<td>IGFBP5</td>
<td>0.050</td>
<td>Regulation of cell growth, signal transduction</td>
<td></td>
</tr>
<tr>
<td>NM 001124</td>
<td>ADM</td>
<td>0.107</td>
<td>Signal transduction, pregnancy, cell proliferation and response to wounding</td>
<td></td>
</tr>
<tr>
<td>NM 001168</td>
<td>BIRC5</td>
<td>0.573</td>
<td>Cell cycle, Apoptosis</td>
<td></td>
</tr>
<tr>
<td>NM 017852</td>
<td>NLRP2</td>
<td>(-0.124)</td>
<td>Apoptosis</td>
<td></td>
</tr>
<tr>
<td>NM 001394</td>
<td>DUSP4</td>
<td>(-0.020)</td>
<td>Cell cycle</td>
<td></td>
</tr>
</tbody>
</table>

a) This table gives gene identifier, gene symbol (HUGO), estimated regression coefficient, presence in the 70 gene signature of Van ’t Veer et al. (2002) and known gene functions according to Gene Ontology.
linear. Path approaches (Keerthi and Shevade, 2007; Park and Hastie, 2007) approximate the solution path by a piecewise linear approximation, linearly interpolating the solution between a series of \(\lambda\)-values for which \(\hat{\beta}(\lambda)\) is exactly calculated. It is fairly straightforward to turn an algorithm for a single \(\lambda\)-value into an algorithm that tracks the solution path. Starting from any \(\lambda_0 = \max_i |\beta_i(0)|\), which yields the estimate \(\hat{\beta}(\lambda_0) = 0\), one can calculate \(\hat{\beta}(\lambda)\) for successively decreasing values \(\lambda_0 > \lambda_1 > \ldots > \lambda_z \geq 0\), each time using the solution \(\hat{\beta}(\lambda_i)\) for \(\lambda_i\) as the starting value for the algorithm to calculate \(\hat{\beta}(\lambda_{i+1})\). If the successive values of \(\lambda\) are close together, the starting values are usually close enough to the solution to ensure very quick convergence.

This path-wise gradient approach has also been proposed for the coordinatewise gradient approaches (Keerthi and Shevade, 2007). It can be slow for these methods, however, due to the slow convergence of these methods near the optimum (Kim and Kim, 2004), which is encountered for each value of \(\lambda\). This slow convergence is avoided in the full gradient approach described in this article because of the incorporation of Newton–Raphson steps.

In fact, if the successive values of \(\lambda\) are chosen in such a way that each \(\lambda\) is an approximate breakpoint at which the active set changes, in the manner of Park and Hastie (2007), the pathwise gradient algorithm becomes almost equivalent to the path algorithm of Park and Hastie (2007). The main advantage of Park and Hastie’s path algorithm is that it improves upon the starting value \(\hat{\beta}(\lambda_i)\) by also using information in the derivative \(\partial \hat{\beta}/\partial \lambda\) at \(\lambda_i\). This use of the derivative guarantees convergence with only Newton–Raphson steps. However, due to its discontinuous nature, the derivative \(\partial \hat{\beta}/\partial \lambda\) is only useable locally in a small neighborhood of \(\lambda_i\), which often forces the path algorithm to make many very small \(\lambda\)-steps. An important advantage of the full gradient algorithm is, therefore, that it can calculate a solution path with any desired step size.

The most important advantage of a gradient-based approach over a path-based approach is its flexibility: it can start from any starting value \(\beta\). This flexibility can be exploited during cross-validation by first fitting the model on the full data, and using the estimated coefficients as starting values for the fits on the partial data sets. This often reduces computation time in tenfold cross-validation to only three to four times the computation time of a single fit. The gain can be even larger in leave-one-out cross-validation. Alternatively, if cross-validation is repeated for multiple values of \(\lambda\), the estimates found for values of \(\lambda\) nearby may be used as input for the next choice of \(\lambda\),

---

![Survival curves for the Van de Vijver breast cancer data.](image)

**Figure 2** Survival curves for the Van de Vijver breast cancer data. Left: predicted survival curves from the fitted model for percentiles of the prognostic index \(X\hat{\beta}\). Right: Kaplan–Meier curves for subgroups of patients, divided according to their cross-validated 5-year survival probability.
as in the path-based approach described above. A combination of both techniques can be used when finding the value of $\lambda$ that optimizes the cross-validated likelihood.

6 Computation Time

In the end, the choice among competing algorithms is often made on the basis of computation time. Given the wide diversity of models and data sets, it is unlikely that any algorithm is uniformly faster than all others. This probable fact was already illustrated by Keerthi and Shevade (2007), who compared the computation time of their path-based algorithm with that of the coordinatewise gradient approach of Genkin et al. (2007) on four different data sets, showing better performance of their own algorithm in three of the data sets, but not in the fourth one.

![Graphs showing computation time comparison between full gradient approach and path-based approach for different data sets.](image)

**Figure 3** Computation time comparison between the full gradient approach of this article (penalized package) and the path-based approach of the glmpath package (Park and Hastie, 2007) for four large survival gene expression data sets. Computation times are given for a single fit at a given value of $\lambda$ and for tenfold cross-validation at a single $\lambda$-value.
In this article we compare the full gradient algorithm with the path approach of Park and Hastie (2007). We chose their method for comparison because our interest is mostly in survival data, and only Park and Hastie have extended their method to the Cox proportional hazards model. Another reason for comparing with Park and Hastie (2007) was that, like the method of this article, the method of Park and Hastie (2007) has been implemented as an R package (glmpath), which made an unbiased comparison of computation times much easier.

Figure 3 shows computation time comparisons between the method of Park and Hastie (2007), implemented in the R package glmpath (version 0.93) and the method of the current article, implemented in the R package penalized (version 0.9–11). Each figure gives computation time (on a 3.0 GHz PC) for a single fit on several fixed values of $\lambda$. We used four large and well-known gene expression data sets with a survival response: Van de Vijver (Van de Vijver et al., 2002, 295

Figure 4  Computation time comparison between the path-based variant of the full gradient approach of this article (penalized package) and the path-based approach of the glmpath package (Park and Hastie, 2007) for three large survival gene expression data sets. Computation time is a function of the number of steps in the gradient-based approach, whereas the number of steps is fixed in the path-based approach. Computations are calculated at the optimal cross-validated value of $\lambda$ for the Van de Vijver, Rosenwald and Wang data sets, and at $\lambda = 2$ for the Beer data set.
subjects, 79 events and 4919 covariates), Rosenwald (Rosenwald et al., 2002, 240 subjects, 138 events and 7399 covariates), Beer (Beer et al., 2002, 86 subjects, 24 events and 3171 covariates) and Wang (Wang et al., 2005, 286 subjects, 107 events and 22 283 covariates). The Wang data set, unfortunately, had to be reduced to 8000 randomly selected genes due to memory problems in the glmpath package. All convergence criteria and other settings were set equal in the two packages. There were no discrepancies between the estimates found by the two methods.

Note that computation time depends enormously on $\lambda$ for both algorithms and that one of the best ways to ensure quick computation is to avoid estimation at small values of $\lambda$ whenever possible. The slow computation at small $\lambda$ is a consequence of the flatness of the top of the target function and is difficult to avoid with any algorithm. One practical solution (Park and Hastie, 2007) is to change the optimization problem by adding a small $L_2$ penalty to the likelihood, forcing strict unimodality.

Figure 3 shows that the full gradient algorithm can be quicker, and in some cases much quicker, than a path-based algorithm. It is not uniformly quicker, however, as the Beer data set illustrates. Our experience with these and other data sets so far is that the gradient-based algorithm tends to be quicker when the sample size is large, but may be slower for smaller sample sizes. An explanation for this phenomenon may be that the fitted prediction rule tends to be less sparse when the sample size increases, so that the path-based algorithms tend to have to make more steps, each of which is itself more costly due to the increased sample size.

The comparison of Fig. 3 is not completely fair, as it only considers the computation time of a single fit, whereas path-based algorithms do not only return a single fit, but also calculate a sequence of additional fits for higher values of $\lambda$. As an additional comparison, we compared the computation time of the path algorithm Park and Hastie (2007) with the path variant of the gradient-based approach of this article. Figure 4 shows computation times of the gradient algorithm using a number of equidistant intermediate steps (as explained in Section 5). We let the path run up to the optimal value with the largest cross-validated likelihood for three of the data sets (Van de Vijver, Rosenwald and Wang). As the optimal value of $\lambda$ for the Beer data set was infinite, we used $\lambda = 2$ for that data set. A single triangle marks the fixed number of steps taken by the path algorithm of Park and Hastie (2007) and its associated computation time.

Figure 4 illustrates the point made in Section 5 that the path-based variant of the gradient ascent algorithm is highly comparable to the path-based algorithm. In terms of computation time, the algorithm of Park and Hastie (2007) seems approximately as fast as the gradient-based algorithm that uses the same number of intermediate steps.

## 7 Extensions

The algorithm has been described above for $L_1$ penalized methods. With minor modifications it can also be used for different types of constraints and penalties. Two specific cases are discussed in this section: additional positivity constraints on the regression coefficients and an additional $L_2$ penalty.

Some models require all or some of the coefficients $\beta$ to be positive. Such constraints are encountered, for example, in genetic linkage models or in variance components analysis, as well as in other fields (De Boer, Den Besten and Ter Braak 2002). In this situation, the algorithm of Box 2 can almost directly be applied in the constrained domain, by only considering the subset of the $3^p$ areas of gradient continuity that are part of the constrained domain. For example, if all coefficients $\beta$ are constrained to be positive, the gradient (3) becomes

$$
\frac{\partial}{\partial \beta_i} \ell (\beta) = \begin{cases} 
  h_i(\beta) - \lambda \, \text{sign}(\beta_i) & \text{if } \beta_i > 0 \\
  h_i(\beta) - \lambda \, \text{sign}(h_i(\beta)) & \text{if } \beta_i = 0 \text{ and } h_i(\beta) > 0 \\
  0 & \text{otherwise.}
\end{cases}
$$

The resulting algorithm can also be applied with $\lambda = 0$, i.e. with only a positivity constraint.
In other cases the solution of the lasso may be considered too sparse, as e.g. in regression models the lasso tends to select only one from a set of correlated predictors. An alternative to the “pure” lasso in that case is to use a combination of $L_1$ and $L_2$ (ridge) penalties. This is the (naive) elastic net of Zou and Hastie (2005), defined by

$$\hat{\beta} = \arg \max \{ \ell(\beta) - \lambda_1 ||\beta||_1 - \lambda_2 ||\beta||_2^2 \},$$

where $||\cdot||_2$ is the $L_2$ norm. The elastic net estimates can be found with the algorithm of Box 2 by viewing the function to be optimized as the sum of a twice-differentiable function $\ell(\beta) - \lambda_2 ||\beta||_2^2$ and an $L_1$ penalty.

If the $L_2$ penalty is relatively large compared with the $L_1$ penalty, the set of active coefficients may grow quite large, potentially leading to the inversion of very large matrices in the Newton–Raphson steps of the algorithm. In the special case of the Cox proportional hazards model and in other generalized linear models, however, if $n \ll p$, such inversion of large matrices may be avoided by employing a local reparameterization. This reparameterization is possible because of the special form of the gradient of $\ell(\beta)$ in such models, which is given by

$$\frac{\partial \ell(\beta)}{\partial \beta} = X^T r$$

for some $n \times p$ design matrix $X$ and some $n$-vector $r$ of residuals. In subdomain of gradient continuity in which a Newton–Raphson step is tried, the estimating equations are given by

$$X^T r - \lambda_1 \text{sign}(\beta) - \lambda_2 \beta = 0.$$

As $\text{sign}(\beta)$ is constant within the relevant subdomain, it follows immediately from the estimating equations that the solution $\hat{\beta}$ must lie in the $(n+1)$-dimensional column span of the matrix $[X^T, \text{sign}(\beta)]$, leading naturally to a reparameterization of $\beta$ as an $n+1$-dimensional vector. Using this reparameterization, the Newton–Raphson steps never have to invert matrices larger than $(n+1) \times (n+1)$.

The gradient ascent algorithm may also be extended quite easily to other variants of the lasso that have recently been proposed, which often have comparable differentiability properties to the regular lasso. A highly comparable gradient ascent algorithm may, for example, be formulated for the group lasso (Yuan and Lin, 2006; Meier, van de Geer, and Bühlmann, 2008).

### 8 Discussion

This article has developed a simple and efficient algorithm for $L_1$ penalized estimation that follows the gradient to the maximum of the penalized likelihood, using a series of directional Taylor approximations.

The algorithm has been designed for generalized linear models and for the Cox proportional hazards model. In these models it improves conceptually over earlier gradient-based algorithms, which update the coefficients one covariate at a time, because the new algorithm uses the information in the full gradient at each step and updates all coefficients simultaneously. The Newton–Raphson step of the algorithm has links to path-based algorithms that follow the solution path of the estimated coefficients as a function of the tuning parameter. However, the gradient approach is more flexible than a path-based approach, because unlike a path-based approach it does not have to follow the solution path from $\beta = 0$, but it can start from any suitable starting value. This makes the algorithm highly suitable for situations in which many fits are to be calculated on similar data, for example when continuously updating a fit on changing data or when performing cross-validation. Furthermore, the gradient-based algorithm can be used as a path-based algorithm, in which case it allows the user to choose the number of steps freely.
The algorithm has been formulated in a very general way as an algorithm for optimizing any function, which is the sum of a twice-differentiable function and an \( L_1 \) penalty. Therefore, it can easily be extended to different models and different penalty structures, as for example in the elastic net, the group lasso, or in other contexts in which sparsity is to be imposed on the coefficients of a model.

Conflict of Interest:

The authors have declared no conflict of interest.

References


