Computational Statistics and Data Analysis 97 (2016) 60–70

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

Natural coordinate descent algorithm for L1-penalised regression in generalised linear models

Tom Michoel

The Roslin Institute, The University of Edinburgh, Easter Bush, Midlothian, EH25 9RG, Scotland, UK

ARTICLE INFO

Article history:
Received 15 July 2014
Received in revised form 25 August 2015
Accepted 18 November 2015
Available online 28 November 2015

Keywords:
Penalised regression
Generalised linear model
Coordinate descent algorithm
Logistic regression

ABSTRACT

The problem of finding the maximum likelihood estimates for the regression coefficients in generalised linear models with an ℓ_1 sparsity penalty is shown to be equivalent to minimising the unpenalised maximum log-likelihood function over a box with boundary defined by the ℓ_1-penalty parameter. In one-parameter models or when a single coefficient is estimated at a time, this result implies a generic soft-thresholding mechanism which leads to a novel coordinate descent algorithm for generalised linear models that is entirely described in terms of the natural formulation of the model and is guaranteed to converge to the true optimum. A prototype implementation for logistic regression tested on two large-scale cancer gene expression datasets shows that this algorithm is efficient, particularly so when a solution is computed at set values of the ℓ_1-penalty parameter as opposed to along a regularisation path. Source code and test data are available from http://tmichoel.github.io/glmnat/.

1. Introduction

In high-dimensional regression problems where the number of potential model parameters greatly exceeds the number of training samples, the use of an ℓ_1 penalty which augments standard objective functions with a term that sums the absolute effect sizes of all parameters in the model has emerged as a hugely successful and intensively studied variable selection technique, particularly for the ordinary least squares (OLS) problem (e.g. Tibshirani, 1996, Osborne et al., 2000a, Osborne et al., 2000b, Efron et al., 2004, Zou and Hastie, 2005, Johnstone and Titterington, 2009, Friedman et al., 2010, El Ghaoui et al., 2012, Tibshirani et al., 2012 and Tibshirani, 2013). Generalised linear models (GLMs) relax the implicit OLS assumption that the response variable is normally distributed and can be applied to, for instance, binomially distributed binary outcome data or Poisson distributed count data (Nelder and Wedderburn, 1972). However, the most popular and efficient algorithm for ℓ_1-penalised regression in GLMs uses a quadratic approximation to the log-likelihood function to map the problem back to an OLS problem and although it works well in practice, it is not guaranteed to converge to the optimal solution (Friedman et al., 2010). Here it is shown that calculating the maximum likelihood coefficient estimates for ℓ_1-penalised regression in generalised linear models can be done via a coordinate descent algorithm consisting of successive soft-thresholding operations on the *unpenalised* maximum log-likelihood function without requiring an intermediate OLS approximation. Because this algorithm can be expressed entirely in terms of the natural formulation of the GLM, it is proposed to call it the *natural coordinate descent algorithm*.

To make these statements precise, let us start by introducing a response variable $Y \in \mathbb{R}$ and predictor vector $X \in \mathbb{R}^p$. It is assumed that Y has a probability distribution from the exponential family, written in canonical form as

$$p(y \mid \eta, \phi) = h(y, \phi) \exp\Big(\alpha(\phi)\big\{y\eta - A(\eta)\big\}\Big),$$

E-mail address: tom.michoel@roslin.ed.ac.uk.

http://dx.doi.org/10.1016/j.csda.2015.11.009
0167-9473/

where $\eta \in \mathbb{R}$ is the natural parameter of the distribution, ϕ is a dispersion parameter and $h, \alpha > 0$ and A convex are known functions. The expectation value of Y is a function of the natural parameter, $E(Y) = A'(\eta)$, and linked to the predictor variables by the assumption of a linear relation $\eta = X^T\beta$, where $\beta \in \mathbb{R}^p$ is the vector of regression coefficients. It is tacitly assumed that $X_1 \equiv 1$ such that β_1 represents the intercept parameter. Suppose now that we have n observation pairs (x_i, y_i) (with $x_{i1} = 1$ fixed for all i). The minus log-likelihood of the observations for a given set of regression coefficients β under the GLM is given by

$$H(\beta) = \frac{1}{n}\sum_{i=1}^{n} A(x_i^T\beta) - y_i(x_i^T\beta) \equiv U(\beta) - w^T\beta, \tag{1}$$

where any terms not involving β have been omitted, $U(\beta) = \frac{1}{n}\sum_{i=1}^{n} A(x_i^T\beta)$ is a convex function, $w = \frac{1}{n}\sum_{i=1}^{n} y_i x_i \in \mathbb{R}^p$, and the dependence of U and w on the data (x_i, y_i) has been suppressed for notational simplicity. In the penalised regression setting, this cost function is augmented with ℓ_1 and ℓ_2 penalty terms to achieve regularity and sparsity of the minimum-energy solution, i.e. H is replaced by

$$H(\beta) = U(\beta) - w^T\beta + \lambda\|\beta\|_2^2 + \mu\|\beta\|_1, \tag{2}$$

where $\|\beta\|_2 = (\sum_{j=1}^{p} |\beta_j|^2)^{\frac{1}{2}}$ and $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$ are the ℓ_2 and ℓ_1 norm, respectively, and λ and μ are positive constants. The ℓ_2 term merely adds a quadratic function to U which serves to make its Hessian matrix non-singular and it will not need to be treated explicitly in our analysis. Furthermore a slight generalisation is made where instead of a fixed parameter μ, a vector of predictor-specific penalty parameters μ_j is used. This allows for instance to account for the usual situation where the intercept coefficient is unpenalised ($\mu_1 = 0$). The problem we are interested in is thus to find

$$\hat{\beta} = \operatorname*{argmin}_{\beta\in\mathbb{R}^p} H(\beta), \tag{3}$$

with H a function of the form

$$H(\beta) = U(\beta) - w^T\beta + \sum_{j=1}^{p} \mu_j|\beta_j|, \tag{4}$$

where $U : \mathbb{R}^p \to \mathbb{R}$ is a smooth convex function, $w \in \mathbb{R}^p$ is an arbitrary vector and $\mu \in \mathbb{R}^p$, $\mu \succcurlyeq 0$ is a vector of non-negative parameters. The notation $u \succcurlyeq v$ is used to indicate that $u_j \geq v_j$ for all j and likewise the notation $u \cdot v$ will be used to indicate elementwise multiplication, i.e. $(u \cdot v)_j = u_j v_j$. The maximum of the *unpenalised* log-likelihood, considered as a function of w, is of course the Legendre transform of the convex function U,

$$L(w) = \max_{\beta\in\mathbb{R}^p}\Big\{w^T\beta - U(\beta)\Big\},$$

and the unpenalised regression coefficients satisfy

$$\hat{\beta}_0(w) = \operatorname*{argmax}_{\beta\in\mathbb{R}^p}\Big\{w^T\beta - U(\beta)\Big\} = \nabla L(w),$$

where ∇ is the usual gradient operator (see Lemma 1 in Appendix A.1). This leads to the following key result:

Theorem 1. *The solution $\hat{\beta}(w, \mu)$ of*

$$\hat{\beta}(w, \mu) = \operatorname*{argmin}_{\beta\in\mathbb{R}^p}\Big\{U(\beta) - w^T\beta + \sum_{j=1}^{p} \mu_j|\beta_j|\Big\} \tag{5}$$

is given by

$$\hat{\beta}(w, \mu) = \hat{\beta}_0\big(\hat{u}(w, \mu)\big) = \nabla L\big(\hat{u}(w, \mu)\big),$$

where $\hat{u}(w, \mu)$ is the solution of the constrained convex optimisation problem

$$\hat{u}(w, \mu) = \operatorname*{argmin}_{\{u\in\mathbb{R}^p\,:\,|u-w|\preccurlyeq\mu\}} L(u). \tag{6}$$

Furthermore the sparsity patterns of $\hat{\beta}$ and $\hat{u} - w + \operatorname{sgn}(\hat{\beta}) \cdot \mu$ are complementary,

$$\hat{\beta}_j(w, \mu) \neq 0 \Leftrightarrow \hat{u}_j(w, \mu) = w_j - \operatorname{sgn}(\hat{\beta}_j)\mu_j.$$

The proof of this Theorem consists of an application of Fenchel's duality theorem and is provided in Appendix A.1. Two special cases of Theorem 1 merit attention. Firstly, in the case of lasso or elastic net penalised linear regression, $U(\beta) = \frac{1}{2}\beta^T C\beta$ is a quadratic function of β, with $C \in \mathbb{R}^{p\times p}$ a positive definite matrix, such that $L(w) = \frac{1}{2}w^T C^{-1}w$ and $\nabla L(w) = C^{-1}w$. If furthermore C is diagonal, with diagonal elements $c_j > 0$, then Eq. (6) reduces to the p independent problems

$$\hat{u}_j = \operatorname*{argmin}_{\{u\in\mathbb{R}:\ |u-w_j|\le\mu_j\}} \frac{1}{c_j}u_j^2,$$

with solution

$$\hat{u}_j = \begin{cases} \operatorname{sgn}(w_j)(|w_j| - \mu_j) & \text{if } |w_j| > \mu_j \\ 0 & \text{otherwise} \end{cases}$$

and $\hat{\beta}_j(w,\mu) = \frac{1}{c_j}\hat{u}_j$. This is the well-known analytic solution of the lasso with uncorrelated predictors (Tibshirani, 1996), which forms the basis for numerically solving the case of arbitrary C as well (Friedman et al., 2010). Secondly, in the case of penalised covariance matrix estimation, $U(\Theta) = -\log\det\Theta$ for non-negative definite matrices $\Theta \in \mathbb{R}^{p\times p}$, and $L(W) = \log\det(-W)^{-1}$ for $W \in \mathbb{R}^{p\times p}$ negative definite (and $+\infty$ otherwise) (Boyd and Vandenberghe, 2004, Section 3.3.1). Eq. (6) is then exactly the dual problem studied by Banerjee et al. (2008).

2. Natural coordinate descent algorithm

2.1. Exact algorithm

It is well-known that a cyclic coordinate descent algorithm for the ℓ_1-penalised optimisation problem in Eq. (5) converges (Tseng, 2001). When only one variable is optimised at a time, keeping all others fixed, the equivalent variational problem in Eq. (6) reduces to a remarkably simple soft-thresholding mechanism illustrated in Fig. 1. More precisely, let $U(\beta)$ be a smooth convex function of a single variable $\beta \in \mathbb{R}$, $w_0 = \operatorname{argmin}_{u\in\mathbb{R}} L(u)$ and $\sigma = \operatorname{sgn}(w - w_0)$. The solution of the one-variable optimisation problem

$$\hat{u} = \operatorname*{argmin}_{\{u\in\mathbb{R}:\ |u-w|\le\mu\}} L(u),$$

with $\mu \ge 0$, can be expressed as follows. If $|w - w_0| \le \mu$ then $\hat{u} = w_0$ and hence $\hat{\beta}(w,\mu) = 0$. Otherwise we must have $\hat{u} = w - \sigma\mu$ and $\hat{\beta}(w,\mu) = L'(w - \sigma\mu) = \hat{\beta}_0(w - \sigma\mu)$. Hence the solution takes the form of a generalised 'soft-thresholding'

$$\hat{\beta}(w,\mu) = \begin{cases} \hat{\beta}_0(w - \sigma\mu) & |w - w_0| > \mu \\ 0 & |w - w_0| \le \mu, \end{cases}$$

see also Fig. 1. In other words, compared to the multivariate problem in Theorem 1 where there remains ambiguity about the signs $\operatorname{sgn}(\hat{\beta}_j)$, in the one-variable case the sign is uniquely determined by the relative position of w and w_0.

Numerically solving the unpenalised one-variable problem is usually straightforward. First note that by assumption, U is differentiable and therefore it is itself the Legendre transform of L. Hence

$$w_0 = \operatorname*{argmin}_{u\in\mathbb{R}} L(u) = \operatorname*{argmax}_{u\in\mathbb{R}} \big\{\beta u - L(u)\big\}\Big|_{\beta=0} = U'(0).$$

Likewise, and assuming there exists no analytic expression for L, $\hat{\beta}_0(w - \sigma\mu)$ can be found as the zero of the function

$$f(\beta) = U'(\beta) - w + \sigma\mu.$$

For U convex, this is a monotonically increasing function of β and conventional one-dimensional root-finding algorithms converge quickly.

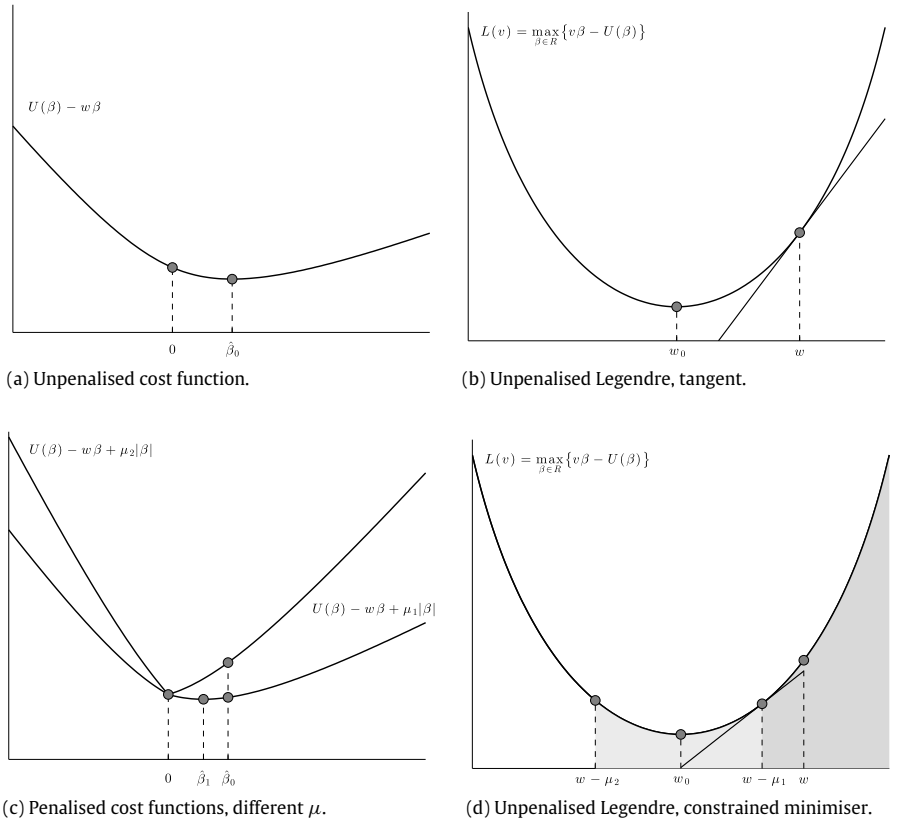
The p-dimensional natural coordinate descent algorithm simply consists of iteratively applying the above procedure to the one-dimensional functions

$$U_j(\beta_j) = U(\hat{\beta}_1, \ldots, \hat{\beta}_{j-1}, \beta_j, \hat{\beta}_{j+1}, \ldots, \hat{\beta}_p),$$

where $\hat{\beta} \in \mathbb{R}^p$ are the current coefficient estimates, i.e.

$$\hat{\beta}_j^{(\text{new})} = \begin{cases} \operatorname{argmax}_{\beta\in\mathbb{R}}\{(w_j - \sigma_j\mu_j)\beta - U_j(\beta)\} & \text{if } \big|w_j - w_{0,j}\big| > \mu_j \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $w_{0,j} = U_j'(0)$ and $\sigma_j = \operatorname{sgn}(w_j - w_{0,j})$.



(a) Unpenalised cost function.

(b) Unpenalised Legendre, tangent.

(c) Penalised cost functions, different μ.

(d) Unpenalised Legendre, constrained minimiser.

Fig. 1. Illustration of Theorem 1 in one dimension. a. The unpenalised cost function $U(\beta) - w\beta$ is a convex function of β; the maximum-likelihood estimate $\hat{\beta}_0$ is its unique minimiser. b. The maximum-likelihood estimate is also equal to the slope of the tangent to the Legendre transform of U at w. c. Every value of the ℓ_1 penalty parameter μ leads to a different cost function; for $\mu = \mu_1$ sufficiently small, the maximum-likelihood estimate $\hat{\beta}_1 < \hat{\beta}_0$ is non-zero while for sufficiently large $\mu = \mu_2$ it is exactly zero. d. The penalised problem can also be solved by minimising the *unpenalised* Legendre transform over the interval $[w - \mu, w + \mu]$; for $w > w_0$ and $\mu_1 < w - w_0$ the absolute minimiser of L is not included in this interval such that the constrained minimiser is the boundary value $w - \mu_1$ and the maximum-likelihood estimate $\hat{\beta}_1$ equals the slope of the tangent at $w - \mu_1$, while for $\mu_2 \geq w - w_0 > 0$, the constrained minimiser is always the absolute minimiser which has a tangent with slope zero. Note that because L is convex, the slope at $w - \mu_1$ is always smaller than the slope at w (i.e. $\hat{\beta}_1 < \hat{\beta}_0$). Similar reasoning applies when $w < w_0$.

Standard techniques can be used to make the algorithm more efficient by organising the calculations around the set of non-zero coefficients (Friedman et al., 2010), that is, after every complete cycle through all coordinates, the current set of non-zero coefficients is updated until convergence before another complete cycle is run (see pseudocode in Appendix C).

2.2. Quadratic approximation algorithm with exact thresholding

An alternative method for updating $\hat{\beta}_j$ in the preceding algorithm is to use a quadratic approximation to $U_j(\beta_j)$ around the current estimate of $\hat{\beta}_j$ in Eq. (7). This leads to a linear approximation for $\hat{\beta}_j^{(\text{new})}$, i.e. if $|w_j - w_{0,j}| > \mu_j$, then

$$\begin{aligned}\hat{\beta}_j^{(\text{new})} &= \operatorname*{argmax}_{\beta\in\mathbb{R}}\{(w_j - \sigma_j\mu_j)\beta - U_j(\beta)\} \\ &\approx \operatorname*{argmax}_{\beta\in\mathbb{R}}\left\{(w_j - \sigma_j\mu_j)\beta - U_j(\hat{\beta}_j) - U_j'(\hat{\beta}_j)(\beta - \hat{\beta}_j) - \frac{1}{2}U_j''(\hat{\beta}_j)(\beta - \hat{\beta}_j)^2\right\} \\ &= \hat{\beta}_j + \frac{w_j - \sigma_j\mu_j - U_j'(\hat{\beta}_j)}{U_j''(\hat{\beta}_j)}. \end{aligned} \tag{8}$$

This approximation differs from the standard quadratic approximation (Friedman et al., 2010) by the fact that it still uses the *exact* thresholding rule from (7). To be precise, given current estimates $\hat{\beta}$, the standard approximation updates the jth coordinate by minimising the approximate quadratic cost function

$$\frac{1}{2}U_j''(\hat{\beta}_j)\beta_j^2 - \left[w_j - U_j'(\hat{\beta}_j) + U_j''(\hat{\beta}_j)\hat{\beta}_j\right]\beta_j + \mu_j|\beta_j|,$$

which has the solution

$$\hat{\beta}_j^{(\text{new})} = \begin{cases} \hat{\beta}_j + \dfrac{w_j - U_j'(\hat{\beta}_j) - \sigma_j'\mu_j}{U_j''(\hat{\beta}_j)} & \text{if } \left|w_j - U_j'(\hat{\beta}_j) + U_j''(\hat{\beta}_j)\hat{\beta}_j\right| > \mu_j \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where $\tilde{\sigma}_j = \text{sgn}\,(w_j - U_j'(\hat{\beta}_j) + U_j''(\hat{\beta}_j)\hat{\beta}_j)$.

Hence, compared to the exact coordinate update rule (7), the standard algorithm not only uses a quadratic approximation to the cost function, but also a linear approximation

$$w_{0,j} = U_j'(0) \approx U_j'(\hat{\beta}_j) - U_j''(\hat{\beta}_j)\hat{\beta}_j.$$

The following result shows that, under certain conditions, the approximate and exact thresholding will return the same result:

Proposition 1. *Let $U(\beta)$ be a smooth convex function of a single variable $\beta \in \mathbb{R}$, and let $\hat{\beta}$ be the solution of*

$$\hat{\beta} = \underset{\beta\in\mathbb{R}}{\text{argmin}}\big\{U(\beta) - w\beta + \mu|\beta|\big\},$$

with $w \in \mathbb{R}$ and $\mu > 0$. Denote $w_0 = U'(0)$ and $\tilde{w}_0 = U'(\hat{\beta}) - U''(\hat{\beta})\hat{\beta}$. Then

$$|w - w_0| > \mu \Leftrightarrow |w - \tilde{w}_0| > \mu.$$

The proof of this proposition can be found in Appendix A.2. Note that in the coordinate descent algorithms the single-coordinate functions U change from step to step, that $\tilde{w}_0$ is calculated on the *current* instead of the *new* solution, and that, in the quadratic approximation algorithm, both the current and new solutions are only approximate minimisers. Hence this result only shows that if all these errors are sufficiently small, then both thresholding rules will agree.

3. Numerical experiments

I implemented the natural coordinate descent algorithm for logistic regression in C with a Matlab interface (source code available from http://tmichoel.github.io/glmnat/). The penalised cost function for $\beta \in \mathbb{R}^p$ in this case is given by

$$H(\beta) = U(\beta) - w^T\beta + \mu\sum_{j=2}^{p}|\beta_j|,$$

where

$$U(\beta) = \frac{1}{n}\sum_{i=1}^{n}\log\big(1 + e^{x_i^T\beta}\big), \qquad w = \frac{1}{n}\sum_{i=1}^{n}y_i x_i^T,$$

and $(x_i \in \mathbb{R}^p, y_i \in \{0, 1\})$, $i = 1, \ldots, n$ are the observations. Recall from Section 1 that β_1 is regarded as the (unpenalised) intercept parameter and therefore a fixed value of one ($x_{i1} = 1$) is added to every observation. As convergence criterion I used $\max_{j=1,\ldots,p}|\hat{\beta}_j^{(\text{new})} - \hat{\beta}_j^{(\text{old})}| < \epsilon$, where $\epsilon > 0$ is a fixed parameter. The difference is calculated at every iteration step when a single coefficient is updated and the maximum is taken over a full iteration after all, resp. all active, coefficients have been updated once.

To test the algorithm I used gene expression levels for 17,814 genes in 540 breast cancer samples (BRCA dataset) (The Cancer Genome Atlas Network, 2012b) and 20,531 genes in 266 colon and rectal cancer samples (COAD dataset) (The Cancer Genome Atlas Network, 2012a) as predictors for oestrogen receptor status (BRCA) and early–late tumour stage (COAD), respectively (see Appendix B for details, processed data available from http://tmichoel.github.io/glmnat/). I compared the implementation of the natural coordinate descent algorithm against `glmnet` (version dated 30 Aug 2013) (Qian et al., 2013), a Fortran-based implementation for Matlab of the coordinate descent algorithm for penalised regression in generalised linear models proposed by Friedman et al. (2010), which was found to be the most efficient in a comparison to various other softwares by the original authors (Friedman et al., 2010) as well as in an independent study (Yuan et al., 2010). All analyses were run on a laptop with 2.7 GHz processor and 8 GB RAM using Matlab v8.2.0.701 (R2013b).

Following Friedman et al. (2010), I considered a geometric path of regularisation parameters

$$\mu^{(k)} = \frac{\max_{j=2,\ldots,p}|w_j|}{m^{\frac{k-1}{m-1}}}, \tag{10}$$

where $m = 100$, and $k = 1, \ldots, m$, corresponding to the default choice in `glmnet`. Note that $\mu^{(1)}$ is the smallest penalty that yields a solution where only the intercept parameter is non-zero. Such a path of parameters evenly spaced on log-scale typically corresponds to models with a linearly increasing number of non-zero coefficients (Friedman et al., 2010). To compare the output of two different algorithms over the entire regularisation path, I considered the maximum relative score difference

$$\max_{k=1,\ldots,m} \frac{H(\hat{\beta}^{(1,k)}) - H(\hat{\beta}^{(2,k)})}{H(\hat{\beta}^{(1,k)})},$$

where $\hat{\beta}^{(1,k)}$ and $\hat{\beta}^{(2,k)}$ are the coefficient estimates obtained by the respective algorithms for the kth penalty parameter.

A critical issue when comparing algorithm runtimes is to match convergence threshold settings. Fig. 2(a) shows the runtimes of the exact natural coordinate descent algorithm (using Eq. (7)) and its quadratic approximation (using Eq. (8)), and their maximum relative score difference for a range of values of the convergence threshold ϵ. The quadratic approximation algorithm is about twice as fast as the exact algorithm and, as expected, both return numerically identical results within the accepted tolerance levels. For subsequent analyses only the quadratic approximation algorithm was used. Because `glmnet` uses a different convergence criterion than the one used here, I ran the natural coordinate descent algorithm with a range of values for ϵ and calculated the maximum relative score difference over the entire regularisation path with respect to the output of `glmnet` with default settings. Fig. 2(b) shows that there is a dataset-dependent value for ϵ where this difference is minimised and that the minimum difference is within the range observed when running `glmnet` with randomly permuted order of predictors. These minimising values $\epsilon_{\text{BRCA}} = 6.3 \times 10^{-4}$ and $\epsilon_{\text{COAD}} = 2.0 \times 10^{-3}$ were used for the subsequent comparisons.

First, I compared the natural coordinate descent algorithms with exact and approximate thresholding rules (cf. Eqs. (7) and (9)). For both datasets and all penalty parameter values, no differences were found between the two rules during the entire course of the algorithm, indicating that the error terms discussed after Proposition 1 are indeed sufficiently small in practice. Since there is as yet no analytical proof extending Proposition 1 to the algorithmic setting, the exact thresholding rule was used for all subsequent analyses.

Next, I compared the natural coordinate descent algorithm to `glmnet` considering both "cold" and "warm" starts. For the cold starts, the solution for the kth penalty parameter value, $\hat{\beta}(\mu^{(k)})$, was calculated starting from the initial vector $\hat{\beta} = 0$. For the warm starts, $\hat{\beta}(\mu^{(k)})$ was calculated along the path of penalty parameters $\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(k)}$, each time using $\hat{\beta}(\mu^{(l)})$ as the initial vector for the calculation of $\hat{\beta}(\mu^{(l+1)})$. This scenario was designed to answer the question: If a solution $\hat{\beta}(\mu)$ is sought for some fixed value of $\mu < \mu^{(1)}$, is it best to run the coordinate descent algorithm once starting from the initial vector $\hat{\beta} = 0$ (cold start), or to run the coordinate descent algorithm multiple times along a regularisation path, each time with an initial vector that should be close to the next solution (warm start)? Clearly, if a solution is needed for all values of a regularisation path it is always better to run through the path once using warm starts at each step.

For `glmnet`, there is a clear advantage to using warm starts and, as also observed by Friedman et al. (2010), for smaller values of μ, it can be faster to compute along a regularisation path down to μ than to compute the solution at μ directly (Fig. 2(e), (f)). In contrast, the natural coordinate descent algorithm is much less sensitive to the use of warm starts (i.e. to the choice of initial vector for $\hat{\beta}$) and it is considerably faster than `glmnet` when calculating solutions at single penalty parameter values (Fig. 2(c), (d)).

To investigate whether this qualitative difference between both algorithms is a general property, the following process was repeated 1000 times: a gene was randomly selected from the BRCA dataset, a binary response variable was defined from the sign of its expression level, and penalised logistic regression with penalty parameter $\mu^{(90)}$ (cf. Eq. (10)) was performed using 5000 randomly selected genes as predictors, using both cold start (with initial vector $\hat{\beta} = 0$) and warm start (along the regularisation path $\mu^{(1)}, \ldots, \mu^{(90)}$); the response gene was constrained to have at least 30% samples of either sign and the predictor genes were constrained to not contain the response gene. This scheme ensured that datasets with sufficient variability in the correlation structure among the predictor variables and between the predictor and the response variables were represented among the test cases. As expected, total runtime correlated well with the size of the model, defined here as the number of predictors with $|\beta_j| > 10^{-3}$, more strongly so for the natural coordinate descent algorithm (Pearson's $\rho_{\text{cold}} = 0.92$, $\rho_{\text{warm}} = 0.91$) than for `glmnet` ($\rho_{\text{cold}} = 0.79$, $\rho_{\text{warm}} = 0.84$). Consistent with these high linear correlations the difference in speed (runtime^{-1}) between cold and warm start was inversely proportional to model size (Spearman's $\rho = -0.90$; Fig. 3(a)). Furthermore, cold start outperformed warm start ($v_{\text{cold}} > v_{\text{warm}}$) in all 1000 datasets. For `glmnet` the opposite was true: warm start always outperformed cold start. However the speed difference ($v_{\text{cold}} - v_{\text{warm}}$) did not correlate as strongly with model size (Spearman's $\rho = 0.53$; Fig. 3(b); note that the opposite sign of the correlation coefficient is merely due to the opposite sign of the speed differences).

This consistent qualitative difference between both algorithms with respect to the choice of initial vector was unexpected in view of the results in Section 2.2. Upon closer inspection, it was revealed that the natural coordinate descent algorithm uses a scheme whereby the parameters of the quadratic approximation for coordinate j (i.e., $U_j'(\hat{\beta}_j)$ and $U_j''(\hat{\beta}_j)$) are updated whenever there is a change in $\hat{\beta}_{j'}$ for some $j' \neq j$. In contrast, `glmnet` uses two separate loops, called "middle" and "inner" loop in Friedman et al. (2010). In the middle loop, the quadratic approximation to U at the current solution $\hat{\beta}^*$ is calculated,

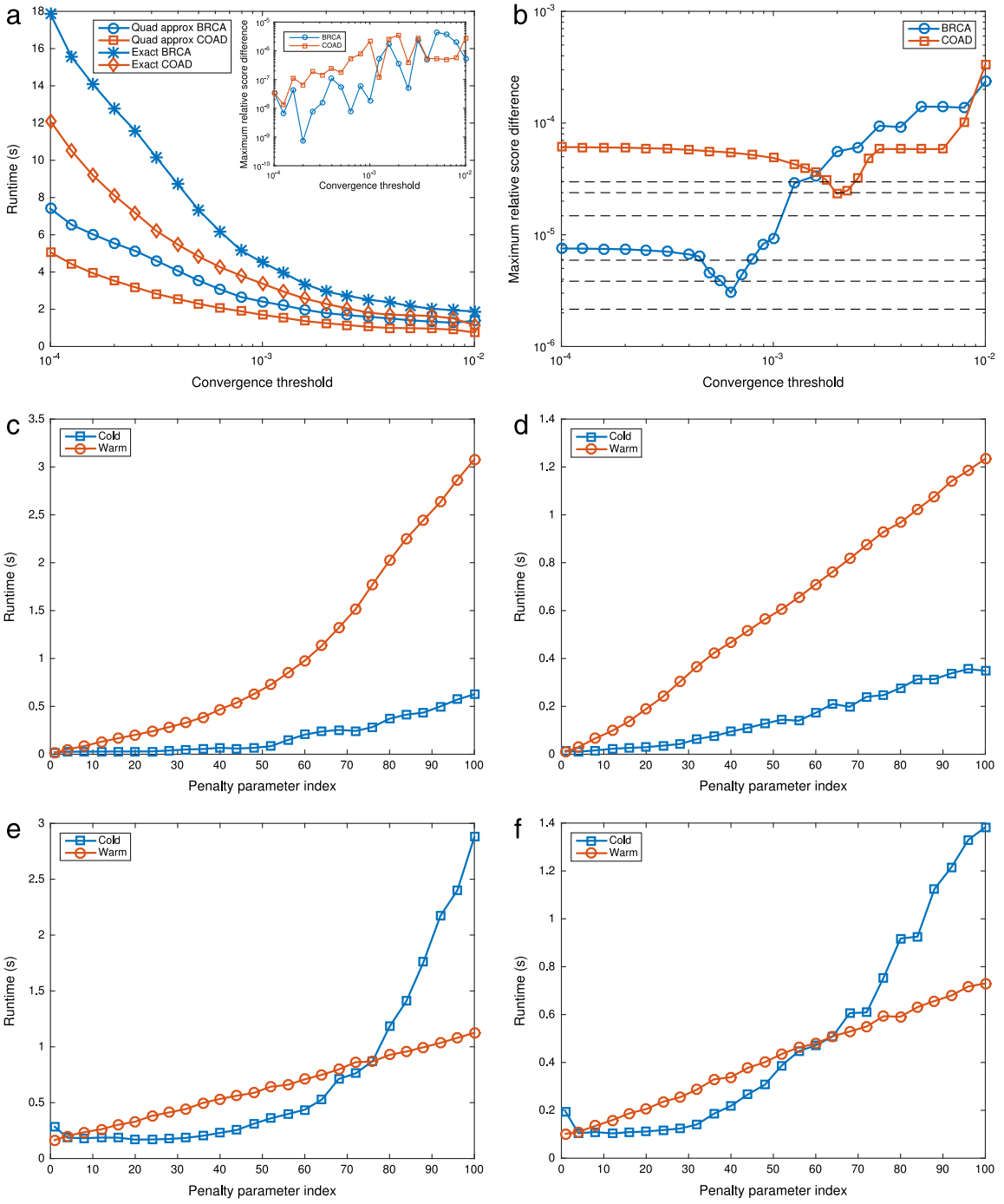


Fig. 2. Numerical evaluation of the natural coordinate descent algorithm. a. Runtime in seconds of the quadratic approximation algorithm on the BRCA (blue circles) and COAD (red squares) dataset and of the exact algorithm on the BRCA (blue asterisks) and COAD (red diamonds) dataset vs. convergence threshold parameter ϵ. The inset shows the maximum relative score difference between both algorithms for the same convergence thresholds. b. Maximum relative score difference between the natural coordinate descent algorithm and `glmnet` vs. convergence threshold parameter ϵ for the BRCA (blue circles) and COAD (red squares) dataset. The horizontal lines indicate the minimum, mean and maximum of the relative score difference over 10 comparisons between the original `glmnet` result and `glmnet` applied to data with randomly permuted order of predictors. c, d. Runtime in seconds of the natural coordinate descent algorithm with cold (blue squares) and warm (red circles) starts on the BRCA (c) and COAD (d) dataset vs. index k of the penalty parameter vector. e, f. Runtime in seconds of `glmnet` with cold (blue squares) and warm (red circles) starts on the BRCA (e) and COAD (f) dataset vs. index k of the penalty parameter vector. See main text for details. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

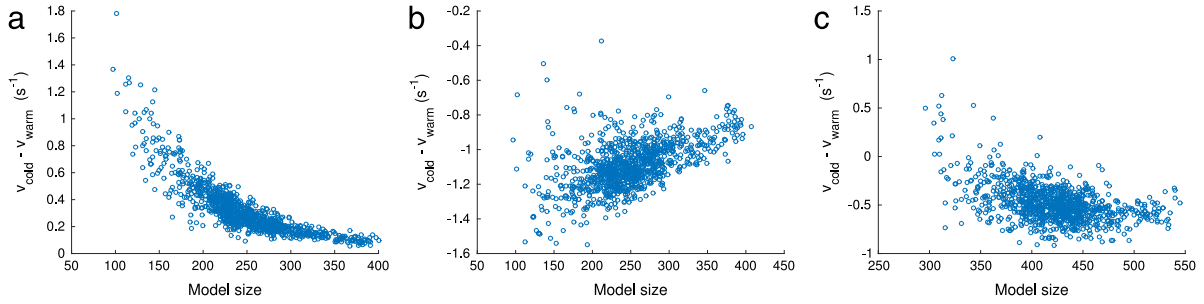


Fig. 3. Difference in speed (runtime^{-1}) vs. model size between cold start (v_{cold}) and warm start (v_{warm}) on sub-sampled datasets for the natural coordinate descent algorithm (a), logistic regression using `glmnet` (b) and linear regression using `glmnet` (c). See main text for details.

i.e.

$$U(\beta) \approx U(\hat{\beta}^*) + \sum_{j=1}^{p} G_j(\beta_j - \hat{\beta}_j^*) + \sum_{j,k=1}^{p} H_{jk}(\beta_j - \hat{\beta}_j^*)(\beta_k - \hat{\beta}_k^*), \tag{11}$$

where

$$G_j = \left.\frac{\partial U}{\partial \beta_j}\right|_{\hat{\beta}^*} \quad \text{and} \quad H_{jk} = \left.\frac{\partial^2 U}{\partial \beta_j \partial \beta_k}\right|_{\hat{\beta}^*}.$$

In the inner loop, a penalised least squares coordinate descent algorithm is run until convergence using the approximation (11), i.e. keeping the values of G and H fixed. A poor choice of initial vector will therefore result in values of G and H that are far from optimal, and running a coordinate descent algorithm until convergence without updating these values would therefore result in a loss of efficiency. It is therefore plausible that the continuous updating of the quadratic approximation parameters in the natural coordinate descent algorithm explains its robustness with respect to the choice of initial vector.

If this reasoning is correct, then the warm-start advantage should not be present if `glmnet` is used to solve penalised least squares problems, since in this case there is no middle loop to be performed. To test this hypothesis, I performed penalised linear regression (lasso) with `glmnet` on the same sub-sampled datasets, using the same binary response variable and the same penalty parameter values as in the previous logistic regressions. Although the speed difference $v_{\text{cold}} - v_{\text{warm}}$ between cold and warm start now indeed followed a similar pattern as the natural coordinate descent algorithm (Spearman's $\rho = -0.43$; Fig. 3(c)), in all but 20 datasets, warm start still outperformed cold start. This suggests that not updating the quadratic approximation at every step during logistic regression in `glmnet` may explain in part why it is more sensitive to the choice of initial vector, but additional, undocumented optimisations of the code must be in place to explain its warm-start advantage.

4. Conclusions

The popularity of ℓ_1-penalised regression as a variable selection technique owes a great deal to the availability of highly efficient coordinate descent algorithms. For generalised linear models, the best existing algorithm uses a quadratic least squares approximation where the coordinate update step can be solved analytically as a linear soft-thresholding operation. This analytic solution has been understood primarily as a consequence of the quadratic nature of the problem. Here it has been shown however that in the dual picture where the penalised optimisation problem is expressed in terms of its Legendre transform, this soft-thresholding mechanism is generic and a direct consequence of the presence of an ℓ_1-penalty term. Incorporating this analytic result in a standard coordinate descent algorithm leads to a method that is not only theoretically attractive and easy to implement, but also appears to offer practical advantages compared to the existing implementations of the quadratic-approximation algorithm. In particular it is more robust to the choice of starting vector and therefore considerably more efficient when it is cold-started, i.e. when a solution is computed at set values of the ℓ_1-penalty parameter as opposed to along a regularisation path of descending ℓ_1-penalties. This can be exploited for instance in situations where prior knowledge or other constraints dictate the choice of ℓ_1-penalty parameter or in data-intensive problems where distributing the computations for sweeping the ℓ_1-penalty parameter space over multiple processors can lead to significant gains in computing time. Future work will focus on developing such parallelized implementations of the natural coordinate descent algorithm and providing implementations for additional commonly used generalised linear models.

Acknowledgement

This research was supported by Roslin Institute Strategic Grant funding from the BBSRC (BB/J004235/1).

Appendix A. Technical proofs

A.1. Proof of Theorem 1

With the notations introduced in Section 1, let $F(\beta) = U(\beta) - w^T\beta$ and $G(\beta) = \sum_{j=1}^p \mu_j|\beta_j|$. F and G are convex functions on $\mathbb{R}^p$ which satisfy Fenchel's duality theorem (Rockafellar, 1970)

$$\min_{\beta\in\mathbb{R}^p}\big\{F(\beta) + G(\beta)\big\} = \max_{u\in\mathbb{R}^p}\big\{-F^*(u) - G^*(-u)\big\}, \tag{A.1}$$

where F^* and G^* are the Legendre transforms of F and G respectively. We have $F^*(u) = \max_{\beta\in\mathbb{R}^p}\{u^T\beta - F(\beta)\} = \max_{\beta\in\mathbb{R}^p}\{(u+w)^T\beta - U(\beta)\} = L(u+w)$ and $G^*(u) = \max_{\beta\in\mathbb{R}^p}\{u^T\beta - G(\beta)\} = \sum_{j=1}^p \max_{b\in\mathbb{R}}\{u_jb - \mu_j|b|\}$. If $|u_j| \le \mu_j$ then the jth term in this sum is 0, otherwise it is ∞, i.e. $G^*(u) = 0$ if $|u| \preccurlyeq \mu$ and $G^*(u) = \infty$ otherwise. It follows that

$$\min_{\beta\in\mathbb{R}^p} H(\beta) = -\min_{\{u\in\mathbb{R}^p:\,|u-w|\preccurlyeq\mu\}} L(u) = -L(\hat{u}) = \min_{\beta\in\mathbb{R}^p}\big\{U(\beta) - \hat{u}^T\beta\big\}, \tag{A.2}$$

where $\hat{u} = \operatorname{argmin}_{\{u\in\mathbb{R}^p:\,|u-w|\preccurlyeq\mu\}} L(u)$. Denoting $\mathcal{S} = \{u \in \mathbb{R}^p\colon |u-w| \preccurlyeq \mu\}$, the minimiser $\hat{u}$ must satisfy the optimality conditions (Boyd and Vandenberghe, 2004, Section 4.2.3): $\hat{u} \in \mathcal{S}$ and

$$(v-\hat{u})^T\nabla L(\hat{u}) \ge 0 \quad \text{for all } v \in \mathcal{S}. \tag{A.3}$$

For any index j, choose $v_j \in \mathcal{S}_j = \{u \in \mathbb{R}\colon |w_j - u| \le \mu_j\}$ arbitrary and set $v_k = \hat{u}_k$ for $k \ne j$. Then $v \in \mathcal{S}$ and by Eq. (A.3),

$$(v_j - \hat{u}_j)\frac{\partial L}{\partial u_j}(\hat{u}) \ge 0. \tag{A.4}$$

Assume $\frac{\partial L}{\partial u_j}(\hat{u}) \ne 0$ and $\hat{u}_j \ne w_j - \sigma_j\mu_j$, where $\sigma_j = \operatorname{sgn}\left(\frac{\partial L}{\partial u_j}(\hat{u})\right)$. Then there exists $\epsilon > 0$ such that $v_j = \hat{u}_j - \epsilon\sigma_j \in \mathcal{S}_j$, but this contradicts Eq. (A.4). By Lemma 1, if $\hat{\beta}_0 = \operatorname{argmin}_{\beta\in\mathbb{R}^p}\big\{U(\beta) - \hat{u}^T\beta\big\}$, then $\hat{\beta}_0 = \nabla L(\hat{u})$. Hence we have shown that

$$\hat{\beta}_{0,j} \ne 0 \Leftrightarrow \hat{u}_j = w_j - \operatorname{sgn}(\hat{\beta}_{0,j})\mu_j.$$

Denote $I = \{j\colon \hat{\beta}_{0,j} \ne 0\}$. We find

$$\hat{u}^T\hat{\beta}_0 = \sum_{j\in I}\hat{u}_j\hat{\beta}_{0,j} = \sum_{j\in I}\big[w_j - \operatorname{sgn}(\hat{\beta}_{0,j})\mu_j\big]\hat{\beta}_{0,j} = w^T\hat{\beta}_0 - \sum_{j=1}^p \mu_j|\hat{\beta}_{0,j}|,$$

and hence by Eq. (A.2),

$$\min_{\beta\in\mathbb{R}^p} H(\beta) = U(\hat{\beta}_0) - \hat{u}^T\hat{\beta}_0 = H(\hat{\beta}_0),$$

i.e. $\hat{\beta}_0$ is also the unique minimiser of the penalised cost function H. This concludes the proof of Theorem 1. □

Lemma 1. *For all $w \in \mathbb{R}^p$, we have*

$$\hat{\beta}_0(w) = \operatorname*{argmax}_{\beta\in\mathbb{R}^p}\Big\{w^T\beta - U(\beta)\Big\} = \nabla L(w).$$

Proof. For a given $\hat{w} \in \mathbb{R}^p$, let

$$\hat{\beta} = \operatorname*{argmax}_{\beta\in\mathbb{R}^p}\Big\{\hat{w}^T\beta - U(\beta)\Big\},$$

or

$$L(\hat{w}) = \max_{\beta\in\mathbb{R}^p}\Big\{\hat{w}^T\beta - U(\beta)\Big\} = \hat{w}^T\hat{\beta} - U(\hat{\beta}).$$

From Fenchel's inequality ($v^T\beta \le U(\beta) + L(v)$ for all $v, \beta \in \mathbb{R}^p$, cf. Boyd and Vandenberghe, 2004, Section 3.3.2) it follows that

$$v^T\hat{\beta} - L(v) \le U(\hat{\beta}) = \hat{w}^T\hat{\beta} - L(\hat{w}),$$

i.e.

$$\hat{w} = \operatorname*{argmax}_{w\in\mathbb{R}^p}\Big\{\hat{\beta}^Tw - L(w)\Big\}.$$

By assumption U is differentiable and hence so is L. It follows that (see Boyd and Vandenberghe, 2004, Section 3.3.2)

$$\hat{\beta} = \nabla L(\hat{w}). \quad \square$$

A.2. Proof of Proposition 1

First, assume $|w - w_0| \leq \mu$. By Theorem 1 we have $\hat{\beta} = 0$, and hence $\tilde{w}_0 = w_0$ and $|w - \tilde{w}_0| = |w - w_0| \leq \mu$. This establishes the direction $|w - \tilde{w}_0| > \mu \Rightarrow |w - w_0| > \mu$. If $|w - w_0| > \mu$, then again by Theorem 1 and using the notation from Section 2,

$$\hat{\beta} = \hat{\beta}_0(w - \sigma\mu) = \operatorname*{argmin}_{\beta \in \mathbb{R}} \{U(\beta) - (w - \sigma\mu)\beta\}.$$

Hence $U'(\hat{\beta}) = w - \sigma\mu$ and $\tilde{w}_0 = w - \sigma\mu - U''(\hat{\beta})\hat{\beta}$. If $\sigma = 1$, $\hat{\beta} > 0$ and, by convexity of U,

$$w - \tilde{w}_0 = \mu + U''(\hat{\beta})\hat{\beta} > \mu.$$

Similarly, if $\sigma = -1$, we have $w - \tilde{w}_0 < -\mu$. This establishes the direction $|w - w_0| > \mu \Rightarrow |w - \tilde{w}_0| > \mu$. □

Appendix B. The Cancer Genome Atlas data processing details

B.1. Breast cancer data (BRCA)

Processed data files were obtained from https://tcga-data.nci.nih.gov/docs/publications/brca_2012/:

- Normalised expression data for 17,814 genes in 547 breast cancer samples (file `BRCA.exp.547.med.txt`).
- Clinical data for 850 breast cancer samples (file `BRCA_Clinical.tar.gz`).

540 samples common to both files had an oestrogen receptor status reported as positive or negative in the clinical data. Oestrogen receptor status was used as the binary response data $Y \in \mathbb{R}^n$, $n = 540$, and gene expression for all genes (+ one constant predictor) was used as predictor data $X \in \mathbb{R}^{n \times p}$, $p = 17{,}815$.

B.2. Colon and rectal cancer data (COAD)

Processed data files were obtained from https://tcga-data.nci.nih.gov/docs/publications/coadread_2012/:

- Normalised expression data for 20,531 genes in 270 colon and rectal cancer samples (file `crc_270_gene_rpkm_datac.txt`).
- Clinical data for 276 colon and rectal cancer samples (file `crc_clinical_sheet.txt`).

266 samples common to both files had a tumour stage (from I to IV) reported in the clinical data. Early (I–II) and late (III–IV) stages were grouped and used as the binary response data $Y \in \mathbb{R}^n$, $n = 266$, and gene expression for all genes (+ one constant predictor) was used as predictor data $X \in \mathbb{R}^{n \times p}$, $p = 20{,}532$.

Appendix C. Natural coordinate descent algorithm pseudocode

See Algorithms 1–5.

Algorithm 1 Main loop

```
Initialise β̂ = 0.
CompleteCycle
while not converged do
    while not converged do
        ActiveSetCycle
    end while
    CompleteCycle
end while
```

Algorithm 2 Complete coordinate descent cycle

```
procedure CompleteCycle
    for j = 1, ..., p do
        CoordinateUpdate(j)
    end for
end procedure
```

Algorithm 3 Active set coordinate descent cycle

procedure ACTIVESETCYCLE
  for $j = 1, \ldots, p$ **do**
    if $\hat{\beta}_j \neq 0$ **then**
      COORDINATEUPDATE(j)
    end if
  end for
end procedure

Algorithm 4 Exact coordinate update

procedure COORDINATEUPDATE(j)
  Update $w_{0,j} = U_j'(0)$.
  if $|w_j - w_{0,j}| > \mu_j$ **then**
    $\hat{\beta}_j \leftarrow$ zero of $U_j'(\cdot) - w_j + \text{sgn}\,(w_j - w_{0,j})\mu_j$.
  else
    $\hat{\beta}_j \leftarrow 0$.
  end if
end procedure

Algorithm 5 Linear approximation for coordinate update

procedure COORDINATEUPDATELINEAR(j)
  Update $w_{0,j} = U_j'(0)$.
  if $|w_j - w_{0,j}| > \mu_j$ **then**
    $\hat{\beta}_j \leftarrow \hat{\beta}_j + \frac{w_j - \text{sgn}\,(w_j - w_{0,j})\mu_j - U_j'(\hat{\beta}_j)}{U_j''(\hat{\beta}_j)}$
  else
    $\hat{\beta}_j \leftarrow 0$.
  end if
end procedure

Appendix D. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.csda.2015.11.009.

References

Banerjee, O., El Ghaoui, L., d'Aspremont, A., 2008. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. J. Mach. Learn. Res. 9, 485–516.
Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press.
Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., 2004. Least angle regression. Ann. Statist. 32 (2), 407–499.
El Ghaoui, L., Viallon, V., Rabbani, T., 2012. Safe feature elimination for the lasso and sparse supervised learning problems. Pac. J. Optim. 8, 667–698.
Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. 33 (1), 1.
Johnstone, I., Titterington, D., 2009. Statistical challenges of high-dimensional data. Phil. Trans. R. Soc. A 367 (1906), 4237–4253.
Nelder, J., Wedderburn, R., 1972. Generalized linear models. J. R. Stat. Soc. Ser. A, Gen. 370–384.
Osborne, M., Presnell, B., Turlach, B., 2000a. On the lasso and its dual. J. Comput. Graph. Statist. 9 (2), 319–337.
Osborne, M.R., Presnell, B., Turlach, B.A., 2000b. A new approach to variable selection in least squares problems. IMA J. Numer. Anal. 20 (3), 389–403.
Qian, J., Hastie, T., Friedman, J., Tibshirani, R., Simon, N., 2013. Glmnet for Matlab. http://www.stanford.edu/~hastie/glmnet_matlab/.
Rockafellar, R.T., 1970. Convex Analysis. Princeton University Press.
The Cancer Genome Atlas Network, 2012a. Comprehensive molecular characterization of human colon and rectal cancer. Nature 487, 330–337.
The Cancer Genome Atlas Network, 2012b. Comprehensive molecular portraits of human breast tumours. Nature 490, 61–70.
Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B Stat. Methodol. 267–288.
Tibshirani, R., 2013. The lasso problem and uniqueness. Electron. J. Stat. 7, 1456–1490.
Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., Tibshirani, R., 2012. Strong rules for discarding predictors in lasso-type problems. J. R. Stat. Soc. Ser. B Stat. Methodol. 74 (2), 245–266.
Tseng, P., 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. 109 (3), 475–494.
Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., Lin, C.-J., 2010. A comparison of optimization methods and software for large-scale L1-regularized linear classification. J. Mach. Learn. Res. 9999, 3183–3234.
Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B Stat. Methodol. 67 (2), 301–320.