Holonomic extended least angle regression

Marc Härkönen¹ · Tomonari Sei² · Yoshihiro Hirose³

Received: 30 August 2018 / Revised: 19 March 2020 / Accepted: 15 September 2020 /

Published online: 8 October 2020

© The Author(s) 2020

Abstract

One of the main problems studied in statistics is the fitting of models. Ideally, we would like to explain a large dataset with as few parameters as possible. There have been numerous attempts at automatizing this process. Most notably, the Least Angle Regression algorithm, or LARS, is a computationally efficient algorithm that ranks the covariates of a linear model. The algorithm is further extended to a class of distributions in the generalized linear model by using properties of the manifold of exponential families as dually flat manifolds. However this extension assumes that the normalizing constant of the joint distribution of observations is easy to compute. This is often not the case, for example the normalizing constant may contain a complicated integral. We circumvent this issue if the normalizing constant satisfies a holonomic system, a system of linear partial differential equations with a finite-dimensional space of solutions. In this paper we present a modification of the holonomic gradient method and add it to the extended LARS algorithm. We call this the holonomic extended least angle regression algorithm, or HELARS. The algorithm was implemented using the statistical software R, and was tested with real and simulated datasets.

Keywords Generalized linear model · Holonomic gradient method · Least angle regression

This work was partly supported by JSPS KAKENHI Grant Numbers JP17K00044, JP18K18008, JST CREST Grant Number JPMJCR1763, and the Vilho, Yrjö and Kalle Väisälä Foundation.

✉ Marc Härkönen
harkonen@gatech.edu

¹ School of Mathematics, Georgia Institute of Technology, 686 Cherry St NW, Atlanta, GA 30332, USA

² Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

³ Faculty of Information Science and Technology and Global Station for Big Data and Cybersecurity, Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan

1 Introduction

In model selection, one would ideally want to choose a statistical model that fits the data well, while still being simple enough to allow meaningful interpretations and explanatory power. In this paper we consider model simplification of linear and generalized linear models, where we want to choose a subset of covariates to include in the model.

In two decades, there have been many advances in sparse modeling. One of the most famous methods is L1-regularization: Least Absolute Shrinkage and Selection Operator (LASSO [23]). LASSO is defined only for the normal linear regression problem. However, the idea of LASSO has been applied to many other problems. For example, Park and Hastie [19] considered the generalized linear models and Yuan and Lin [25] treated Gaussian graphical models. Least Angle Regression (LARS, [6]) is an efficient algorithm for computing the LASSO solution paths. The LARS algorithm is described based on Euclidean geometry because LARS considers the normal linear regression problem. Hirose and Komaki [10] proposed the Extended LARS (ELARS) algorithm based on the information geometry of dually flat spaces. ELARS is an algorithm for estimating and selecting parameters in the generalized linear models. The idea of ELARS was applied to edge selection in the Gaussian graphical models [11] and the contingency table models [12]. Another version of geometrical extensions of LARS was given by Augugliaro, Mineo and Wit [4], and a geometrical approach to sparse modeling was also proposed in [26].

The ELARS algorithm by Hirose and Komaki [10] has a computational drawback in that it assumes that the potential function (i.e. the normalizing constant) of the underlying probability distribution function is easy to compute. This is often not the case, which motivates us to use the holonomic gradient method, a computationally efficient method for computing potential functions and their gradients, introduced by Nakayama et al. [17]. There are a few other methods that deal with non-normalized models in the literature. In [13] Hyvärinen proposed a method for estimating parameters of non-normalized models. This method makes use of a 2-local proper scoring rule, which can be computed without knowledge of the normalizing constant [20]. In the case of the ELARS algorithm, such methods are of limited use since we need to know the value of the normalizing constant and its derivatives at several points throughout the algorithm.

A system of linear partial differential equations is called a holonomic system if it has a finite-dimensional space of solutions. Refer to Sect. 3 for a more precise description. If the potential function satisfies a holonomic system, we can use a modification of the holonomic gradient method to keep track of its value at each step of the algorithm and update it when needed in a computationally efficient way. We call the combined algorithm the holonomic extended LARS algorithm, or HELARS.

The main result of the paper is an implementation in $\mathbb{R}$ of the HELARS. In one explicit example we consider the normal distribution restricted to positive values as the underlying distribution. This distribution is simple enough as an introductory example due to its similarities with the well-known normal distribution, especially with regards to e- and m-affine coordinate conversions. Despite the truncated normal having no closed form potential function, our implementation of the algorithm does not

use numerical integration to compute its value. Of course, the potential function of the truncated normal distribution is nothing but the Gaussian cumulative distribution function, which is implemented as a built-in function in almost all software packages. However, since the truncated normal model is a special case of more complicated models such as the exponential-polynomial distributions [8] and the multivariate truncated normal distributions [14], our result will become a prototype of the overall method.

We will also consider another more complicated example, which we will call the weighted truncated normal distribution. In this example the coordinate conversions become complicated. However, by exploiting the holonomic system we can recover relations between the normalizing constant and its derivatives. Since throughout the algorithm we keep track of the normalizing constant and derivatives, this will allow us to recover any coordinate at any point in the algorithm.

The paper is organized as follows. In Sect. 2 we review basic definitions and results concerning generalized linear models. In Sect. 3 we present the holonomic gradient method. Section 4 discusses the extended LARS algorithm [10], and we look at what necessary changes and additions are needed for the HELARS algorithm. In Sect. 5 we walk through the HELARS algorithm using the two examples mentioned before. We validate the algorithm using both real and simulated datasets in Sect. 6. Finally, we end with a discussion in Sect. 7.

2 Generalized linear models

In this section we will review some foundations of generalized linear models [15]. We will follow [1] in our exposition.

Definition 1 Consider a statistical model $\mathcal{P} = \{p_{\xi} \mid \xi \in \Xi\}$, where $\Xi \subseteq \mathbb{R}^d$. We say that $\mathcal{P}$ is an exponential family if for $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{R}^n$, $\xi = (\xi^1, \dots, \xi^d) \in \Xi$ we have

$$p_{\xi}(\mathbf{y}) = p(\mathbf{y} \mid \xi) = e^{C(\mathbf{y}) + \xi \cdot \mathbf{F}(\mathbf{y}) - \psi(\xi)}, \quad (1)$$

for some $\mathbf{F}: \mathbb{R}^n \rightarrow \mathbb{R}^d$ and $C: \mathbb{R}^n \rightarrow \mathbb{R}$, and where $\psi(\xi)$ is the logarithm of the normalizing constant, i.e.

$$\psi(\xi) = \log \int e^{C(\mathbf{y}) + \xi \cdot \mathbf{F}(\mathbf{y})} d\mathbf{y}$$

Example 1 The *normal distribution* is a member of the exponential family. It has the probability density function

$$\begin{aligned} p(y \mid \mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y - \mu)^2}{2\sigma^2} \right] \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{y^2 - 2y\mu + \mu^2}{2\sigma^2} \right] \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\exp(\mu^2/(2\sigma^2))\sqrt{2\pi\sigma^2}} \exp\left[-\frac{y^2 - 2y\mu}{2\sigma^2}\right] \\
&= \exp(C(\mathbf{y}) + \boldsymbol{\xi} \cdot \mathbf{F}(\mathbf{y}) - \psi(\boldsymbol{\xi})).
\end{aligned}$$

Note that here $C(\mathbf{y}) = 0$, the natural parameter is $\boldsymbol{\xi} = \left[-\frac{1}{2\sigma^2} \frac{\mu}{\sigma^2}\right]^T$, $\mathbf{F}(\mathbf{y}) = [y^2 \ y]^T$ and $\psi(\boldsymbol{\xi}) = \frac{(\boldsymbol{\xi}^2)^2}{4\xi^1} + \log \sqrt{-\pi/\xi^1}$.

Definition 2 The *Fisher information matrix* of a distribution $p(\mathbf{y} \mid \boldsymbol{\xi})$ at a point $\boldsymbol{\xi}$ is a $d \times d$ matrix $\mathbf{G}(\boldsymbol{\xi}) = (g_{i,j})$ with entries given by

$$g_{i,j} = E \left[\frac{\partial \log p(\mathbf{y} \mid \boldsymbol{\xi})}{\partial \xi^i} \frac{\partial \log p(\mathbf{y} \mid \boldsymbol{\xi})}{\partial \xi^j} \middle| \boldsymbol{\xi} \right]$$

Equivalently, we may write the elements of the Fisher information matrix as

$$g_{i,j} = -E \left[\frac{\partial^2}{\partial \xi^i \partial \xi^j} \log p(\mathbf{y} \mid \boldsymbol{\xi}) \middle| \boldsymbol{\xi} \right]$$

Next we introduce generalized linear models. Assume that we have n independent observations $y_1, y_2, \dots, y_n$. Each observation y_i is sampled from an exponential family with scalar parameter ξ^i , which will depend on a *covariate vector* $\mathbf{x}^i = (x_1^i, \dots, x_d^i)$. The $(n \times d)$ matrix $\mathbf{X} = (x_j^i)$ is called the *design matrix*. We assume that the covariate vector $\mathbf{x}^i$ influences the distribution of y_i only via the *linear predictor* ζ^i , defined as

$$\zeta^i := \sum_{j=1}^d \theta^j x_j^i,$$

or using matrices $\boldsymbol{\zeta} := \mathbf{X}\boldsymbol{\theta}$, for some vector $\boldsymbol{\theta} \in \mathbb{R}^d$. We can also add an intercept term by defining a new design matrix $\tilde{\mathbf{X}} := [\mathbf{1}_{n \times 1} \ \mathbf{X}]$ so that $\boldsymbol{\zeta} := \tilde{\mathbf{X}}\boldsymbol{\theta}$, for some vector $\boldsymbol{\theta} = (\theta^0, \theta^1, \dots, \theta^d)^T$. We will always use an intercept term throughout this paper.

The final piece of a generalized linear model, the *link function* $g(\mu)$, determines in which way the linear predictor influences the distribution by setting

$$\zeta^i = g(\mu_i),$$

where μ_i is the expectation of y_i . The *canonical link* is the link function g for which $\xi^i = \eta^i$ for all $i = 1, 2, \dots, n$.

The combination of an exponential family, design matrix and link function define a *generalized linear model* (GLM). The model has $d+1$ parameters $\theta^0, \theta^1, \dots, \theta^d$ that we want to estimate given the response $\mathbf{y}$ and design matrix $\mathbf{X}$. Fitting a GLM is usually more delicate than fitting a linear model. The standard goal is to find the parameters

$\theta^0, \theta^1, \dots, \theta^d$ which maximize the (log-)likelihood. The log-likelihood of the joint distribution of n observations is $L = \sum_{i=1}^n L_i$, where $L_i = \log p(y_i \mid \xi^i)$. This is maximized when all the partial derivatives $\frac{\partial L}{\partial \theta^k}$ vanish, assuming that the log-likelihood is strictly concave. In general, the partial derivatives will not be linear functions of θ , so we have to resort to numerical methods to compute the MLE.

One possible iterative method to find the maximum likelihood estimate $\hat{\theta}$ is the Newton–Raphson method. Note that we will use this method extensively along with the holonomic gradient method in our implementation (see Sect. 5) for both maximum likelihood estimation and other optimization tasks. We start with an initial guess $\theta_{(0)}$. For each $k \geq 0$, we approximate the function at the point $\theta_{(k)}$ with a polynomial of degree 2. Finding the extremum of the approximation is easy, and we set the point reaching the extremum as the next estimate $\theta_{(k+1)}$.

3 Holonomic gradient method

In this section we will describe the holonomic gradient method (HGM), first proposed by Nakayama et al. [17]. Consider first the “classical” gradient descent algorithm, which is used to find a local minimum of a function $F: \mathbb{R}^n \rightarrow \mathbb{R}$. Given a starting point (or initial guess) $\mathbf{x}^{(0)}$, we know that the value of the function decreases the fastest in the direction opposite to the gradient. In other words, we should choose the next point

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \gamma_1 \nabla F(\mathbf{x}^{(0)}), \quad (2)$$

for some stepsize $\gamma_1 > 0$. Given a suitably chosen stepsize γ_1 , we have $F(\mathbf{x}^{(1)}) < F(\mathbf{x}^{(0)})$. We then iterate

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \gamma_{k+1} \nabla F(\mathbf{x}^{(k)}), \quad (3)$$

while choosing a suitable stepsize $\gamma_k > 0$ at each iteration. We can terminate the algorithm when the gradient is small enough (i.e. when we are close to a local minimum), or when a certain number of iterations have elapsed. Details concerning the choice of step size and efficiency of this method will not be discussed here; see for example [3].

The issue with this method is that it requires computing the gradient $\nabla F(\mathbf{x}^{(k)})$ at each step. In many statistical applications, the function we want to optimize will be a likelihood function, which will in some cases contain an integral that does not have a closed form expression, and has to be computed using numerical methods.

The holonomic gradient method takes a different approach to function minimization. The main idea is still the same in that we use the same iterative step as in the classical gradient descent in (3).

The difference is how we compute the gradient. We will construct a vector $\mathbf{Q}$ and a set of matrices $\mathbf{P}_i$ to form a *Pfaffian system*

$$\frac{\partial \mathbf{Q}}{\partial x_i} = \mathbf{P}_i \mathbf{Q}. \quad (4)$$

The vector $\mathbf{Q}$ will be chosen so that the gradient $\nabla F(\mathbf{x}^{(k)})$ is easily recoverable, typically $\nabla F(\mathbf{x}^{(k)}) = \mathbf{A}(\mathbf{x}^{(k)})\mathbf{Q}(\mathbf{x}^{(k)})$ for some matrix $\mathbf{A}$. With the gradient, we can determine the next point $\mathbf{x}^{(k+1)}$ using (3). Given $\mathbf{Q}(\mathbf{x}^{(k)})$, the value of $\mathbf{Q}$ in the previous step, we can compute its value in the next step $\mathbf{Q}(\mathbf{x}^{(k+1)})$ by solving the Pfaffian system (4) using standard numerical ODE solvers.

Observe that we can also implement the Newton–Raphson method using the holonomic gradient framework. The update step will be

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{Hess}(F(\mathbf{x}^{(k)}))^{-1} \nabla F(\mathbf{x}^{(k)}), \quad (5)$$

and we can also recover the Hessian easily from the Pfaffian system, since there is a matrix $\mathbf{B}(\mathbf{x}^{(k)})$ such that $\mathbf{Hess}(F)(\mathbf{x}^{(k)}) = \mathbf{B}(\mathbf{x}^{(k)})\mathbf{Q}(\mathbf{x}^{(k)})$.

3.1 Rings of differential operators

In order to construct Pfaffian systems, we will be looking at certain ideals in a ring of differential operators. Intuitively, it can be thought of as a regular polynomial ring whose “variables” are differential operators $\partial_1, \dots, \partial_n$, and coefficients are rational functions, i.e. functions of the form $f(x_1, \dots, x_n)/g(x_1, \dots, x_n)$, where f, g are both polynomials. The set of all rational functions (with complex coefficients in n variables) is a field denoted by $\mathbb{C}(x_1, \dots, x_n)$, in contrast to the regular polynomial ring’s $\mathbb{C}[x_1, \dots, x_n]$.

Definition 3 The ring R_n of differential operators with rational function coefficients is

$$R_n := \left\{ \sum_{\alpha} c_{\alpha}(x_1, \dots, x_n) \partial_1^{\alpha_1} \partial_2^{\alpha_2} \cdots \partial_n^{\alpha_n} \mid \alpha \in \mathbb{Z}_{\geq 0}^n, c_{\alpha} \in \mathbb{C}(x_1, \dots, x_n), \right. \\ \left. \text{a finite number of } c_{\alpha} \text{ are non-zero} \right\}$$

Note that the “multiplication” operation inside the ring is actually a composition of operators. Multiplication in this ring satisfies the following commutation rules

$$\begin{aligned} \partial_i x_i &= x_i \partial_i + 1 \\ \partial_i x_j &= x_j \partial_i, \text{ if } i \neq j \\ \partial_i \partial_j &= \partial_j \partial_i \\ x_i x_j &= x_j x_i \end{aligned}$$

Since every element in R_n is an operator, there is a natural action $\bullet$ on the set C^{∞} of smooth functions. If $f(x_1, \dots, x_n) \in C^{\infty}$, then

$$\begin{aligned} \partial_i \bullet f(x_1, \dots, x_n) &= \frac{\partial f}{\partial x_i} \\ x_i \bullet f(x_1, \dots, x_n) &= x_i f(x_1, \dots, x_n). \end{aligned}$$

We then note that $(\partial_i x_i) \bullet f = (x_i \partial_i + 1) \bullet f$, which motivates the definition of multiplication in the ring.

Because of these commutation rules, when multiplying two elements in R_n , we can always rearrange factors within a term so that the differential operators ∂_i are on the right of each term, as in Definition 3. For example, $\partial_1 x_1 x_2 \partial_2 = x_1 x_2 \partial_1 \partial_2 + x_2 \partial_2$.

Most standard theorems and algorithms from algebraic geometry in the regular (commutative) polynomial ring $k[x_1, \dots, x_n]$ over an arbitrary field k carry over to R_n with minor modifications. Because of this, and by slight abuse of notation, we will write R_n as

$$R_n = \mathbb{C}(x_1, \dots, x_n) \langle \partial_1, \dots, \partial_n \rangle,$$

where the coefficient field is $\mathbb{C}(x_1, \dots, x_n)$ and the angled brackets around the “variables” ∂_i emphasize the non-commutativity.

Remark 1 As R_n is not a commutative ring, we have to distinguish between left and right ideals. In the following and throughout the manuscript, any mention of an ideal will always be a left ideal, following the convention in [9].

In particular, Macaulay’s theorem will be useful in the next section. For $\alpha \in \mathbb{Z}_{\geq 0}^n$, let $\partial^\alpha := \partial_1^{\alpha_1} \partial_2^{\alpha_2} \dots \partial_n^{\alpha_n}$. Let $<$ be a term order¹ on the differential operators ∂_i . Let $f = \sum_{\alpha} c_{\alpha} \partial^{\alpha} \in R_n$. The leading term of f is the term $\text{LT}(f) = c_{\alpha'} \partial^{\alpha'}$ such that $\partial^{\alpha} < \partial^{\alpha'}$ for all $\alpha \neq \alpha'$ such that $c_{\alpha} \neq 0$. For an ideal $I \subset R_n$, we define $\text{LT}(I)$ as the ideal generated by all leading terms in I . We note that in general $\text{LT}(I)$ is not a subset of I nor is $\text{LT}(I)$ generated by the leading terms of the generators of I .

Theorem 1 (Macaulay’s theorem) *Let $I \subset R_n$ be an ideal. The set of standard monomials*

$$\{w \in R_n \mid w \text{ is a monomial, } w \notin \text{LT}(I)\}$$

is a basis of R_n/I as a vector space over R_n .

We say an ideal $I \subseteq R_n$ is 0-dimensional if there are finitely many standard monomials. A necessary and sufficient condition for I to be 0-dimensional is that for all i the following holds

$$I \cap \mathbb{C}(x_1, \dots, x_n) \langle \partial_i \rangle \neq \{0\}.$$

For more details on computations and algorithms in rings of differential operators, see [9].

Remark 2 Instead of the ring R_n , many algorithms actually use the Weyl algebra $D_n := \mathbb{C}\langle x_1, \dots, x_n, \partial_1, \dots, \partial_n \rangle$, which consists of elements of the form $\sum_{\alpha} c_{\alpha} \partial^{\alpha}$, where the c_{α} are polynomials, as opposed to rational functions. In D_n , the counterpart of a zero

¹ A total order on monomials ∂^{α} where $\partial^{\alpha} < \partial^{\beta}$ implies $\partial^{\gamma} \partial^{\alpha} < \partial^{\gamma} \partial^{\beta}$.

dimensional ideal of R_n is called a *holonomic ideal*. There is a natural homomorphism $D_n \hookrightarrow R_n$, which gives the following connection: the extension of a holonomic ideal is zero dimensional, and the restriction of a zero dimensional ideal is holonomic. For the sake of this manuscript, the aforementioned connection will suffice; for a more complete treatment of holonomic ideals, see [9].

3.2 Pfaffian systems

Let $f(x_1, \dots, x_n) \in C^\infty$ be a function, and $\ell \in R_n$. When $\ell \bullet f = 0$, we say that f is *annihilated* by ℓ . We say that f is annihilated by an ideal $I \subset R_n$ if f is annihilated by all $\ell \in I$. Observe that if $I = \langle \ell_1, \dots, \ell_s \rangle$, f is annihilated by I if and only if f is annihilated by all ℓ_i . Assume that I is 0-dimensional, and let $s_1 = 1, s_2, \dots, s_r$ be the standard monomials, which are the $\mathbb{C}(x_1, \dots, x_n)$ -vector space generators of R_n/I . For all $1 \leq i \leq n$ and $1 \leq j \leq r$, we can look at the image of the operator $\partial_i s_j$ in the quotient R_n/I (under the canonical map $p \mapsto p + I$), and write it as a $\mathbb{C}(x_1, \dots, x_n)$ -linear combination of the standard monomials

$$\partial_i s_j = \sum_{k=1}^r p_{jk}^i s_k,$$

where $p_{jk}^i \in \mathbb{C}(x_1, \dots, x_n)$ for all $1 \leq i \leq n$ and $1 \leq j, k \leq r$. Thus, if we define the vector $\mathbf{S} = (s_1, s_2, \dots, s_r)^T$, then for each i , there is a matrix $\mathbf{P}_i$ with entries in $\mathbb{C}(x_1, \dots, x_n)$ such that

$$\partial_i \mathbf{S} = \mathbf{P}_i \mathbf{S}, \quad (6)$$

with $(\mathbf{P}_i)_{jk} = p_{jk}^i$.

Define the vector $\mathbf{Q} = (s_1 \bullet f, s_2 \bullet f, \dots, s_r \bullet f)^T$. Since f is annihilated by all elements in I , equation (6) is true when we replace $\mathbf{S}$ by $\mathbf{Q}$. We get the following system of differential equations

$$\frac{\partial \mathbf{Q}}{\partial x_i} = \mathbf{P}_i \mathbf{Q}, \quad (7)$$

called the Pfaffian system. Because we chose $s_1 = 1$, the gradient of f can be recovered from the first elements of each equation in the Pfaffian system

$$\nabla f = \begin{bmatrix} (\mathbf{P}_1 \mathbf{Q})_1 \\ (\mathbf{P}_2 \mathbf{Q})_1 \\ \vdots \\ (\mathbf{P}_n \mathbf{Q})_1 \end{bmatrix}$$

By following the procedure above, one can construct a Pfaffian system given a 0-dimensional (or holonomic) ideal annihilating our function f . This is indeed desirable,

since finding a 0-dimensional annihilating ideal is often easier than finding a Pfaffian system from scratch. One noteworthy fact is that if f has a holonomic annihilating ideal in D_n , then its integral over one variable $\int f dx_i$ also has a holonomic annihilating ideal in D_{n-1} . This is extremely useful when using the holonomic gradient method, since the normalizing constant will usually contain an integral. Oaku [18] describes an algorithm for computing the (holonomic) annihilating ideal of an integral.

4 Holonomic extended LARS

In this section, we will describe the holonomic extended least angle regression algorithm. We will also compute explicit forms for the Fisher information matrix, divergence, and coordinate conversion functions between the e-affine and m-affine coordinates.

Consider a set of observed data $\{y_a, \mathbf{x}^a = (x_1^a, \dots, x_d^a) \mid a = 1, 2, \dots, n\}$, where $\mathbf{y} = (y_1, \dots, y_n)^T$ is the response vector, and $\mathbf{X} = (x_j^a)$ is the design matrix, which has dimensions $(n \times d)$. We will also add an intercept term to the model, so the design matrix becomes $\tilde{\mathbf{X}} = [\mathbf{1}_{n \times 1} \mid \mathbf{X}]$. We will consider exponential families of the form

$$p(\mathbf{y} \mid \boldsymbol{\xi}) = \exp \left(\sum_{a=1}^n y_a \xi^a + \sum_{b=1}^r u_b(\mathbf{y}) \xi^{b+n} - \psi^*(\boldsymbol{\xi}) \right). \quad (8)$$

We will define some notation. Let $\boldsymbol{\xi}$ be the natural parameter, a $n + r$ sized vector containing elements ξ^i . We can split $\boldsymbol{\xi}$ into two subvectors: we call $\boldsymbol{\xi}'$ the subvector containing the first n elements, i.e. $\boldsymbol{\xi}' = (\xi^a)_{a=1}^n$, and we call $\boldsymbol{\xi}''$ the subvector containing the last r elements, i.e. $\boldsymbol{\xi}'' = (\xi^b)_{b=n+1}^{n+r}$. Hence $\boldsymbol{\xi} = (\boldsymbol{\xi}', \boldsymbol{\xi}'')^T$. The function $\psi^*(\boldsymbol{\xi})$ is the potential function of $\boldsymbol{\xi}$, and it is equal to the logarithm of the normalizing constant of the distribution

$$\psi^*(\boldsymbol{\xi}) = \log \int \exp \left(\sum_{a=1}^n y_a \xi^a + \sum_{b=1}^r u_b(\mathbf{y}) \xi^{b+n} \right) d\mathbf{y}$$

In a generalized linear model with canonical link function, the natural parameter is related to linear predictor by $\boldsymbol{\xi}' = \tilde{\mathbf{X}}\boldsymbol{\theta}'$, where $\boldsymbol{\theta}' = (\theta^0, \theta^1, \dots, \theta^d)^T$ is a parameter vector. In addition, as in [10], we require r additional parameters that are equal to $\boldsymbol{\xi}''$. Hence we can write $\boldsymbol{\theta}'' = \boldsymbol{\xi}''$, and define $\boldsymbol{\theta} = (\boldsymbol{\theta}', \boldsymbol{\theta}'')$. Equation (8) thus becomes

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \exp \left(\mathbf{y}^T \tilde{\mathbf{X}}\boldsymbol{\theta}' + \sum_{b=1}^r u_b(\mathbf{y}) \theta^{b+d} - \psi(\boldsymbol{\theta}) \right),$$

where the potential function of $\boldsymbol{\theta}$ is $\psi(\boldsymbol{\theta}) = \psi^*(\tilde{\mathbf{X}}\boldsymbol{\theta}', \boldsymbol{\theta}'')$. Alternatively, define the sufficient statistic

$$\mathbf{Y} = (y_1, \dots, y_n, u_1(\mathbf{y}), \dots, u_r(\mathbf{y}))^T,$$

and an $(n + r) \times (d + r + 1)$ block-diagonal matrix

$$\mathbf{X}_B = \begin{bmatrix} \tilde{\mathbf{X}} & \mathbf{0}_{n \times r} \\ \mathbf{0}_{r \times (d+1)} & \mathbf{I}_{r \times r} \end{bmatrix}$$

where $\mathbf{0}_{n \times m}$ and $\mathbf{I}_{n \times n}$ are respectively the $(n \times m)$ zero matrix and the $(n \times n)$ identity matrix. Then we have the identities

$$\begin{aligned} \xi &= \mathbf{X}_B \theta \\ \psi(\theta) &= \psi^*(\mathbf{X}_B \theta) \end{aligned} \quad (9)$$

and the probability density function becomes

$$p(\mathbf{y} \mid \theta) = \exp(\mathbf{Y}^T \mathbf{X}_B \theta - \psi(\theta)).$$

We also note that when $\mathbf{X}_B$ has full rank and ξ is (approximately) in the column space, we can recover θ using least-squares.

The ξ coordinate, being the natural parameter of an exponential family, is the e-affine coordinate of the model manifold. The corresponding m-affine coordinate μ is the expectation parameter $\mu = \mathbb{E}[\mathbf{Y}]$ and its potential function is defined as $\phi^*(\mu) = \xi \cdot \mu - \psi^*(\xi)$. The model manifold defined by the coordinates θ is a submanifold of the model defined by the ξ coordinates. The e-affine coordinate of this submanifold is θ , and there is a dual m-affine coordinate η which is related to μ by

$$\eta = \mathbb{E}[(\mathbf{Y}^T \mathbf{X}_B)^T] = \mathbf{X}_B^T \mathbb{E}[\mathbf{Y}] = \mathbf{X}_B^T \mu. \quad (10)$$

Note that the Fisher information matrix of the model in (8) is equal to the Hessian of the potential function $\psi^*(\xi)$, denoted $\mathbf{G}^* = (g_{i,j}^*)$. Similarly, denote the Hessian of the potential function of the m-affine coordinates $\phi^*(\mu)$ as the matrix $\mathbf{G}_* = (g_{*}^{i,j})$. Since ξ and μ are dual coordinates, the matrices $\mathbf{G}^*$ and $\mathbf{G}_*$ are inverses of each other.

The Hessian $\mathbf{G}(\theta) = (g_{i,j})$ of the potential function $\psi(\theta) (= \psi^*(\tilde{\mathbf{X}}\theta', \theta''))$ can be recovered using the chain rule:

$$g_{i,j} = \frac{\partial^2 \psi(\theta)}{\partial \theta^i \partial \theta^j} = \frac{\partial \eta_i}{\partial \theta^j} = \sum_{a=0}^{n+d} \sum_{b=0}^{n+d} \frac{\partial \eta_i}{\partial \mu_a} \frac{\partial \mu_a}{\partial \xi^b} \frac{\partial \xi^b}{\partial \theta^j}. \quad (11)$$

Using the identities in (9) and (10), we see that

$$\begin{aligned} \frac{\partial \eta_i}{\partial \mu_a} &= \frac{\partial (\mathbf{X}_B^T \mu)_i}{\partial \mu_a} = (\mathbf{X}_B^T)_{i,a} \\ \frac{\partial \xi^b}{\partial \theta^j} &= \frac{\partial (\mathbf{X}_B \theta)_b}{\partial \theta^j} = (\mathbf{X}_B)_{b,j}. \end{aligned}$$

Thus (11) becomes

$$g_{i,j} = \sum_{a=1}^n \sum_{b=1}^n (\mathbf{X}_B^T)_{i,a} g_{a,b}^* (\mathbf{X}_B)_{b,j},$$

which implies that $\mathbf{G} = \mathbf{X}_B^T \mathbf{G}^* \mathbf{X}_B$. We denote elements of its inverse with superscripts: $\mathbf{G}^{-1} = (g^{i,j})$. Similar to the previous case, we have

$$\begin{aligned} (\mathbf{G})_{i,j} &= \frac{\partial \eta_i}{\partial \theta^j} = g_{i,j} \\ (\mathbf{G}^{-1})_{i,j} &= \frac{\partial \theta^i}{\partial \eta_j} = g^{i,j} \end{aligned} \quad (12)$$

Remark 3 In subsequent sections we will make extensive use of matrix and vector differentiation. The convention in [16] will be used: the shape of $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ depends either on the shape of $\mathbf{f}$ or the shape of $\mathbf{x}^T$. For example, differentiating a scalar by a length n column vector yields a length n row vector

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_n} \right]$$

Differentiating a length m column vector by a scalar yields a length m column vector

$$\frac{\partial \mathbf{f}}{\partial x} = \left[\frac{\partial f_1}{\partial x} \quad \frac{\partial f_2}{\partial x} \quad \dots \quad \frac{\partial f_m}{\partial x} \right]^T$$

Finally, differentiating a length m column vector by a length n column vector yields an $(m \times n)$ matrix, the Jacobian.

$$\mathbf{Jac}(\mathbf{f}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left(\frac{\partial f_i}{\partial x_j} \right)_{i,j}$$

This notation allows the natural use of the chain rule for derivatives $\frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}} = \frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{x}))}{\partial \mathbf{g}(\mathbf{x})} \frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}$ with the usual matrix multiplication between the two terms on the right hand side. Furthermore, we can express the Hessian of a scalar valued function $f(\mathbf{x})$ as $\mathbf{Hess}(f) = \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}^T}$

4.1 Mixed coordinate conversion

Next, consider a point P on the dually flat manifold

$$S = \{p(\mathbf{y} \mid \boldsymbol{\theta}) = \exp(\mathbf{Y}^T \mathbf{X}_B \boldsymbol{\theta} - \psi(\boldsymbol{\theta})) \mid \boldsymbol{\theta} \in \mathbb{R}^{d+r+1}\}.$$

It is characterized by the e- and m-affine coordinates $\boldsymbol{\theta} = (\theta^0, \theta^1, \dots, \theta^{d+r})$ and $\boldsymbol{\eta} = (\eta_0, \eta_1, \dots, \eta_{d+r})$. Alternatively, we may use *mixed coordinates*, i.e. for some

$J \subset \{0, 1, \dots, d+r\}$ we represent P as $(\eta_J, \theta^{\bar{J}})$, where $\eta_J = \{\eta_j \mid j \in J\}$ is the subvector of η containing only elements which have indices in J , and $\theta^{\bar{J}} = \{\theta^j \mid j \notin J\}$ is the subvector of θ containing elements with indices not in J .

Let $J \subseteq \{0, 1, 2, \dots, d+r\}$, $\bar{J} = \{0, 1, 2, \dots, d+r\} \setminus J$ and let $P = (\eta_J, \theta^{\bar{J}})$ denote a mixed coordinate. Converting mixed coordinates to unmixed coordinates means recovering $\eta_{\bar{J}}$ and θ^J given η_J and $\theta^{\bar{J}}$. Let $\tilde{\theta}(\theta^J) = (\theta^J, \theta^{\bar{J}})$, a function $\mathbb{R}^{|J|} \rightarrow \mathbb{R}^{d+r+1}$ obtained by mixing the fixed $\theta^{\bar{J}}$ and the unknown θ^J coordinates in the positions defined by J . Thus, the function $\tilde{\theta}$ outputs the full θ coordinates, where $\theta^{\bar{J}}$ are always constant, and θ^J are allowed to vary. Similarly, let $\tilde{\eta}(\eta_{\bar{J}}) = (\eta_J, \eta_{\bar{J}})$ be the same function for the η coordinates.

Depending on the implementation, we will have two alternative methods to compute the mixed-to-unmixed coordinate conversion. If a closed form expression for the conversion from μ to ξ (and vice-versa) is available, we can use Newton's method to find the root of the function

$$\mathbf{F}(\theta^J, \eta_{\bar{J}}) = \tilde{\eta}(\eta_{\bar{J}}) - \eta(\tilde{\theta}(\theta^J)). \quad (13)$$

Proposition 1 *The Jacobian $\mathbf{Jac}(F)$ of F in (13), has columns*

$$(\mathbf{Jac}(F))_i = \begin{cases} -(\mathbf{G}(\tilde{\theta}))_i & \text{if } i \in J \\ \mathbf{e}_i & \text{if } i \in \bar{J} \end{cases},$$

where $\mathbf{G}(\tilde{\theta})_i$ is the i th column of $\mathbf{G}(\tilde{\theta}) = \frac{\partial^2 \psi(\theta)}{\partial \theta \partial \theta^T} \Big|_{\theta=\tilde{\theta}}$ (see (12)), and the vector $\mathbf{e}_i$ is the i th standard basis of $\mathbb{R}^{n+d+1}$.

Proof Let $i \in \bar{J}$. Then

$$\frac{\partial F}{\partial \eta_i} = \frac{\partial}{\partial \eta_i} \tilde{\eta}(\eta_{\bar{J}}) = 0.$$

Since the i th element of $\tilde{\eta}(\eta_{\bar{J}})$ is simply η_i and none of the other elements depend on η_i , we have $\frac{\partial F}{\partial \eta_i} = \mathbf{e}_i$.

Next let $i \in J$. Using the chain rule we get

$$\begin{aligned} \frac{\partial F}{\partial \theta^i} &= 0 - \frac{\partial}{\partial \theta^i} (\eta(\tilde{\theta}(\theta^J))) \\ &= -\frac{\partial \eta(\tilde{\theta})}{\partial \tilde{\theta}} \cdot \frac{\partial \tilde{\theta}(\theta^J)}{\partial \theta^i} \end{aligned}$$

$$\begin{aligned} &= -(\mathbf{G}(\tilde{\boldsymbol{\theta}})) \cdot \mathbf{e}_i \\ &= -(\mathbf{G}(\tilde{\boldsymbol{\theta}}))_i \end{aligned}$$

□

Newton's method will iteratively output a vector $(\boldsymbol{\theta}^J, \boldsymbol{\eta}_{\bar{J}})$ with the following update step

$$(\boldsymbol{\theta}^J, \boldsymbol{\eta}_{\bar{J}})^{(k+1)} = (\boldsymbol{\theta}^J, \boldsymbol{\eta}_{\bar{J}})^{(k)} - \mathbf{Jac}(\mathbf{F})^{-1} \mathbf{F},$$

where the Jacobian and $\mathbf{F}$ are evaluated at $(\boldsymbol{\theta}^J, \boldsymbol{\eta}_{\bar{J}})^{(k)}$. Given a suitable initial guess, the method converges very quickly.

In some cases (such as Example 3), there is no obvious closed form expression for the conversion from $\boldsymbol{\mu}$ to $\boldsymbol{\xi}$, which is required to compute the function F . In these cases we will exploit the Pfaffian system to recover $\boldsymbol{\xi}$ from the vector $\mathbf{Q}$ in equation (7). Details for this case will be deferred to Sect. 5.2.

4.2 Extended least angle regression algorithm

We describe shortly the algorithm by Hirose and Komaki [10]. Let I be the set containing the indices of covariates present in the model. We first start with the model containing all covariates, that is $I = \{1, 2, \dots, d\}$, and compute a suitable initial value. In the examples in this manuscript and the pseudo-code below, we use the maximum likelihood estimate $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ as the initial point. Thus we will require that $n \geq d$. In addition, we also compute the maximum likelihood estimate $\hat{\boldsymbol{\theta}}^\emptyset$ of the empty model, i.e. the model where $\theta^1, \dots, \theta^d = 0$. The corresponding $\boldsymbol{\eta}$ -coordinates are denoted $\hat{\boldsymbol{\eta}}^\emptyset$. We will work in the d dimensional submanifold of S

$$M = \left\{ \boldsymbol{\eta} \mid \eta_0 = \hat{\eta}_0^\emptyset, \eta_{d+1} = \hat{\eta}_{d+1}^\emptyset, \dots, \eta_{d+r} = \hat{\eta}_{d+r}^\emptyset \right\}, \quad (14)$$

and set $\hat{\boldsymbol{\theta}}_{(0)} := \hat{\boldsymbol{\theta}}_{\text{MLE}}$ and $k = 1$. The intuitive idea is that we start with all covariates, and sequentially remove one covariate at a time by following the diagonal of a hypercube whose edges are m-geodesics. This will give us a piecewise linear path from $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ to $\hat{\boldsymbol{\theta}}^\emptyset$, and we will be able to infer the relative importance of the covariates based on the order in which the covariates are eliminated.

For each $i \in I$, let $\bar{\boldsymbol{\theta}}_i$ be the m-projection of the current point $\hat{\boldsymbol{\theta}}_{(k)}$ to the e-flat submanifold corresponding to $\theta^i = 0$. Let i^* be the coordinate which has smallest divergence between the point $\hat{\boldsymbol{\theta}}_{(k)}$ and its m-projection $\bar{\boldsymbol{\theta}}_i$, and let this divergence be t^* . Now for each $i \in I$, look at the m-geodesic connecting $\hat{\boldsymbol{\theta}}_{(k)}$ and $\bar{\boldsymbol{\theta}}_i$, and find the point $\boldsymbol{\theta}_i^*$ along that geodesic that has divergence t^* from $\hat{\boldsymbol{\theta}}_{(k)}$. The estimate for the next step $\hat{\boldsymbol{\theta}}_{(k+1)}$ is constructed as follows: for all $i \in I$, set the i th coordinate of $\hat{\boldsymbol{\theta}}_{(k+1)}$ to the i th coordinate of $\boldsymbol{\theta}_i^*$, and for all $j \notin I$, set the j th coordinate of $\hat{\boldsymbol{\theta}}_{(k+1)}$ to 0. Notice that by construction the i^* th coordinate will also be 0. We now remove i^* from the

list of “active” covariates I , and restart at the beginning of the paragraph, this time in the submanifold

$$M_I = \left\{ \boldsymbol{\eta} \mid \eta_0 = \hat{\eta}_0^\emptyset, \eta_{d+1} = \hat{\eta}_{d+1}^\emptyset, \dots, \eta_{d+r} = \hat{\eta}_{d+r}^\emptyset, \theta^j = 0 \forall j \notin I \right\}.$$

We quit the algorithm after d steps, when no covariates are left. The divergence function used in the submanifold M_I , which we will denote by $D^{[I]}$, is the restriction of the KL-divergence on M onto the submanifold. We compute it as follows: denote the coordinates in M_I as $\boldsymbol{\theta}_I = (\theta^i)_{i \in I}$ and $\boldsymbol{\eta}_I = (\eta_i)_{i \in I}$. The potential functions become $\psi_I(\boldsymbol{\theta}_I) = \psi(\boldsymbol{\theta}_I, \mathbf{0}_j)$ and $\phi_I(\boldsymbol{\eta}_I) = \boldsymbol{\eta}_I \cdot \boldsymbol{\theta}_I - \psi_I(\boldsymbol{\theta}_I)$. The divergence is then

$$D^{[I]}(p, q) = \phi_I(\boldsymbol{\eta}_I(p)) + \psi_I(\boldsymbol{\theta}_I(q)) - \boldsymbol{\eta}_I(p) \cdot \boldsymbol{\theta}_I(q).$$

The algorithm starts from the full model and proceeds step by step towards the empty model, which is the opposite direction compared to the LARS. Other than that, the geometric idea of the algorithm is the same as in LARS: at each step k , we move the current estimate $\hat{\boldsymbol{\theta}}_{(k)}$ towards the origin, in a direction that bisects the m-geodesics corresponding to each m-projection. We hit the next estimate $\hat{\boldsymbol{\theta}}_{(k+1)}$ exactly when one of the coordinates $i \in I$ of the vector $\hat{\boldsymbol{\theta}}_{(k)}$ hits 0.

The following pseudocode describes the algorithm more precisely. The algorithm is described in the submanifold M of (14), so we do not write down coordinates $0, d+1, \dots, d+r$ explicitly. We input the data (observations and design matrix) and an underlying distribution (essentially the functions $u_1(\mathbf{y}), \dots, u_r(\mathbf{y})$ in (8)), and we get as an output a sequence of estimators $(\hat{\boldsymbol{\theta}}_{(0)}, \dots, \hat{\boldsymbol{\theta}}_{(d)})$, where the estimator obtained in the k th step corresponds to a model with k covariates removed.

1. Let $I = \{1, 2, \dots, d\}$, $\hat{\boldsymbol{\theta}}_{(0)} := \hat{\boldsymbol{\theta}}_{\text{MLE}}$, and $k = 0$.
2. For all $i \in I$, let $M(i, 0, I) = \{\boldsymbol{\theta} \mid \theta^i = 0, \theta^j = 0 (j \notin I)\} = M(I \setminus \{i\})$ and calculate the m-projection $\bar{\boldsymbol{\theta}}(i, I)$ of $\hat{\boldsymbol{\theta}}_{(k)}$ on $M(i, 0, I)$.
3. Let $t^* = \min_{i \in I} D^{[I]}(\hat{\boldsymbol{\theta}}_{(k)}, \bar{\boldsymbol{\theta}}(i, I))$ and $i^* = \arg \min_{i \in I} D^{[I]}(\hat{\boldsymbol{\theta}}_{(k)}, \bar{\boldsymbol{\theta}}(i, I))$.
4. For every $\alpha^i \in \mathbb{R}, i \in I$, let $M(i, \alpha^i, I) = \{\boldsymbol{\theta} \mid \theta^i = \alpha^i, \theta^j = 0 (j \notin I)\}$.
For every $i \in I$, compute α^i such that the m-projection $\bar{\boldsymbol{\theta}}'(i, \alpha^i, I)$ of $\hat{\boldsymbol{\theta}}_{(k)}$ on $M(i, \alpha^i, I)$ satisfies $t^* = D^{[I]}(\hat{\boldsymbol{\theta}}_{(k)}, \bar{\boldsymbol{\theta}}'(i, \alpha^i, I))$.
5. Let $\hat{\theta}_{(k+1)}^i = \alpha^i (i \in I)$ and $\hat{\theta}_{(k+1)}^j = 0 (j \notin I)$.
6. If $k = d - 1$, then go to step 7. If $k < d - 1$, then go to step 2 with $k := k + 1$, $I := I \setminus \{i^*\}$.
7. Let $\hat{\boldsymbol{\theta}}_{(d)} = \mathbf{0}$. Output $\hat{\boldsymbol{\theta}}_{(0)}, \dots, \hat{\boldsymbol{\theta}}_{(d)}$ and quit the algorithm.

We can now rank the covariates in order of importance by looking at the output of the algorithm. The zeroth estimator $\hat{\boldsymbol{\theta}}_{(0)}$ was defined as the maximum likelihood estimate of the full model containing every covariate, and at each subsequent estimator, one of the components will vanish, i.e. the k th estimator $\hat{\boldsymbol{\theta}}_{(k)}$ will have exactly k of its elements equal to zero. The element that vanishes corresponds to the covariate that is deemed the least impactful at that particular step. Thus by looking at the order in which the covariates vanish in the sequence $\hat{\boldsymbol{\theta}}_{(0)}, \dots, \hat{\boldsymbol{\theta}}_{(d)}$, we can order the covariates from least to most important.

4.3 Adding holonomicity

We will focus our attention to the potential function $\psi(\xi)$ in (8), which can be written as

$$\psi^*(\xi) = \log \int \exp \left(\sum_{a=1}^n y_a \xi^a + \sum_{b=1}^r u_b(\mathbf{y}) \xi^{b+n} \right) d\mathbf{y}.$$

Whether or not $\psi^*(\xi)$ has a closed form representation depends on the underlying distribution. We could use numerical integration if no closed form expression for $\psi^*(\xi)$ exists, but such an approach is computationally inefficient.

Instead, we assume that all observations are independent. Then we can write the potential function as a sum

$$\psi^*(\xi) = \sum_{a=1}^n \log \int \exp \left(y_a \xi^a + \sum_b u_b^a(y_a) \xi^{b+n} \right) dy_a \quad (15)$$

$$=: \sum_{a=1}^n \log f_a(\xi), \quad (16)$$

where $u_b(\mathbf{y}) = \sum_a u_b^a(y_a)$, and $f_a(\xi) := \int \exp(y \xi^a + \sum_b u_b^a(y) \xi^{b+n}) dy$.

We will further assume that for each $a = 1, \dots, n$, $b = 1, 2, \dots, n+r$, the function f_a has a zero-dimensional annihilating ideal in $\mathbb{C}(\xi^a, \xi^{n+1}, \dots, \xi^{n+r}) \langle \partial_{\xi^a}, \partial_{\xi^{n+1}}, \dots, \partial_{\xi^{n+r}} \rangle$. As was discussed in the end of Sect. 3, such an ideal can be found by using Oaku's integration algorithm [18], or alternatively, as in Examples 2 and 3, we can also derive the annihilating ideal manually (e.g. finding relations between derivatives using integration by parts).

Following the construction in Sect. 3.2, for each $a = 1, \dots, n$ we have a Pfaffian system

$$\frac{\partial \mathbf{Q}_a}{\partial \xi^b} = \mathbf{P}_a^b \mathbf{Q}_a,$$

where f_a is the first entry of the vector $\mathbf{Q}_a$.

The key observation here is that we can now write the gradient and Hessian of the potential function $\psi^*(\xi)$ using only the entries of all vectors $\mathbf{Q}_a$. Hence, throughout the ELARS algorithm, instead of recomputing the f_a each time using numerical integration, we do it only once at some initial point ξ_0 to construct the vectors $\mathbf{Q}_a(\xi_0)$. As the point ξ changes, we update the vectors $\mathbf{Q}_i$ using the holonomic update procedure described in Sect. 4.3.1.

Example 2 (Truncated normal) We consider a distribution where each observation is independently distributed according to a normal distribution restricted to positive values only. We will call this the *truncated normal distribution*. We get a potential

function of the form

$$\psi^*(\xi) = \sum_{a=1}^n \log \int_0^\infty \exp(y\xi^a + y^2\xi^{n+1}) \, dy$$

Let $f_a = \int_0^\infty \exp(y\xi^a + y^2\xi^{n+1}) \, dy$. We first note that $\partial_{\xi^{n+1}} f_a = \partial_{\xi^n}^2 f_a$, and using integration by parts, we have

$$\begin{aligned} f_a &= \left[\frac{1}{\xi^a} \exp(\xi^a y + \xi^{n+1} y^2) \right]_0^\infty - \frac{2\xi^{n+1}}{\xi^a} \int_0^\infty y \exp(\xi^a y + \xi^{n+1} y^2) \, dy \\ &= -\frac{1}{\xi^a} - \frac{2\xi^{n+1}}{\xi^a} \partial_{\xi^a} f_a, \end{aligned}$$

which gives the partial differential equation $(\xi^a + 2\xi^{n+1}\partial_{\xi^a}) \bullet f_a = -1$. From this we see that

$$\begin{aligned} \partial_{\xi^a} f_a &= -\frac{\xi^a}{2\xi^{n+1}} f_a - \frac{1}{2\xi^{n+1}} \\ \partial_{\xi^a}^2 f_a &= -\frac{\xi^a}{2\xi^{n+1}} \partial_{\xi^a} f_a - \frac{1}{2\xi^{n+1}} f_a \\ &= \frac{1}{2\xi^{n+1}} \left(\frac{(\xi^a)^2}{2\xi^{n+1}} - 1 \right) f_a + \frac{\xi^a}{(2\xi^{n+1})^2}. \end{aligned}$$

In this example we would choose $\mathbf{Q}_a = (f_a, 1)^T$, so that we get the Pfaffian system

$$\begin{aligned} \frac{\partial \mathbf{Q}_a}{\partial \xi^a} &= -\frac{1}{2\xi^{n+1}} \begin{bmatrix} \xi^a & 1 \\ 0 & 0 \end{bmatrix} \mathbf{Q}_a \\ \frac{\partial \mathbf{Q}_a}{\partial \xi^{n+1}} &= \frac{1}{(2\xi^{n+1})^2} \begin{bmatrix} (\xi^{na})^2 - 2\xi^{n+1} & \xi^a \\ 0 & 0 \end{bmatrix} \mathbf{Q}_a \end{aligned}$$

We can now compute the gradient of the potential function. Recall that $\psi^*(\xi) = \sum_a \log f_a$, so

$$\begin{aligned} \frac{\partial \psi^*}{\partial \xi^a} &= \frac{1}{f_a} \cdot \frac{\partial f_a}{\partial \xi^a} \\ \frac{\partial \psi^*}{\partial \xi^{n+1}} &= \sum_a \frac{1}{f_a} \cdot \frac{\partial f_a}{\partial \xi^{n+1}} \end{aligned}$$

when $a = 1, 2, \dots, n$. Hence if all $\mathbf{Q}_a$ are known, the gradient of ψ^* can be computed using elementary operations, since f_a and its derivatives can be recovered from $\mathbf{Q}_a$ and the Pfaffian systems.

The Hessian of ψ^* can be computed in a similar manner. We have

$$\begin{aligned}\frac{\partial^2 \psi^*}{\partial \xi^a{}^2} &= \frac{1}{f_a} \cdot \frac{\partial^2 f_a}{\partial \xi^a{}^2} - \frac{1}{f_a^2} \cdot \left(\frac{\partial f_a}{\partial \xi^a} \right)^2 \\ \frac{\partial^2 \psi^*}{\partial \xi^a \partial \xi^{n+1}} &= \frac{1}{f_a} \cdot \frac{\partial^2 f_a}{\partial \xi^a \partial \xi^{n+1}} - \frac{1}{f_a^2} \cdot \frac{\partial f_a}{\partial \xi^a} \cdot \frac{\partial f_a}{\partial \xi^{n+1}} \\ \frac{\partial^2 \psi^*}{\partial \xi^{n+1}{}^2} &= \sum_{a=1}^n \left(\frac{1}{f_a} \cdot \frac{\partial^2 f_a}{\partial \xi^{n+1}{}^2} - \frac{1}{f_a^2} \cdot \left(\frac{\partial f_a}{\partial \xi^{n+1}} \right)^2 \right) \\ \frac{\partial^2 \psi^*}{\partial \xi^a \partial \xi^b} &= 0 \quad \text{when } a \neq b = 1, 2, \dots, n,\end{aligned}$$

and we remark that all derivatives and second derivatives of f_a can be computed from the Pfaffian system.

Example 3 (Weighted truncated normal) Consider a set of n independent observations, where each is distributed according to

$$p(y; \xi^a, \xi^{n+1}) = \frac{y^c \exp(y\xi^a + y^2\xi^{n+1})}{f_a},$$

when $y \geq 0$ and for some fixed $c > 0$ and where $a = 1, 2, \dots, n$. The function f_a is defined as the normalizing constant

$$f_a := \int_0^\infty y^c \exp(y\xi^a + y^2\xi^{n+1}) \, dy.$$

We will derive a Pfaffian system for f_a . To simplify notation, we will write ∂_a instead of $\frac{\partial}{\partial \xi^a}$. Note that

$$\partial_a^n f_a = \int_0^\infty y^{c+n} \exp(\xi^a y + \xi^{n+1} y^2) \, dy,$$

so in particular $\partial_{n+1} f_a = \partial_a^2 f_a$. Integration by parts yields

$$\begin{aligned}f_a &= -\frac{1}{c+1} \int_0^\infty (\xi^a + 2\xi^{n+1}y) y^{c+1} \exp(\xi^a y + \xi^{n+1}y^2) \, dy \\ &= -\frac{1}{c+1} (\xi^a \partial_a f_a + 2\xi^{n+1} \partial_a^2 f_a)\end{aligned}$$

so f_a is annihilated by the holonomic ideal $\langle \partial_{n+1} - \partial_a^2, \frac{2\xi^{n+1}}{c+1}\partial_a^2 + \frac{\xi^a}{c+1}\partial_a + 1 \rangle$. If we define the vector $\mathbf{Q}_a = (f_a, \partial_a f_a, \partial_a^2 f_a, \partial_a^3 f_a)^T$, we get the Pfaffian system

$$\begin{aligned} \frac{\partial \mathbf{Q}_a}{\partial \xi^a} &= \begin{bmatrix} \partial_a^1 f_a \\ \partial_a^2 f_a \\ \partial_a^3 f_a \\ \partial_a^4 f_a \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{(c+1)(c+2)}{(2\xi^{n+1})^2} & 0 & \frac{(\xi^a)^2 - (4c+10)\xi^{n+1}}{(2\xi^{n+1})^2} & 0 \end{bmatrix} \mathbf{Q}_a := \mathbf{P}_a^a \mathbf{Q}_a, \\ \frac{\partial \mathbf{Q}_a}{\partial \xi^{n+1}} &= \begin{bmatrix} \partial_a^2 f_a \\ \partial_a^3 f_a \\ \partial_a^4 f_a \\ \partial_a^5 f_a \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{(c+1)(c+2)}{(2\xi^{n+1})^2} & 0 & \frac{(\xi^a)^2 - (4c+10)\xi^{n+1}}{(2\xi^{n+1})^2} & 0 \\ 0 & -\frac{(c+1)(c+2)}{(2\xi^{n+1})^2} & \frac{2\xi^a}{(2\xi^{n+1})^2} & \frac{(\xi^a)^2 - (4c+10)\xi^{n+1}}{(2\xi^{n+1})^2} \end{bmatrix} \mathbf{Q}_a := \mathbf{P}_a^{n+1} \mathbf{Q}_a. \end{aligned} \quad (17)$$

As in the previous example, we can now compute any derivative of the potential function using the vectors $\mathbf{Q}_a$ and matrices $\mathbf{P}_a^a, \mathbf{P}_a^{b+1}$.

4.3.1 Holonomic update of the vectors $\mathbf{Q}_a$

Nearly every step of the algorithm requires the knowledge of the vectors $\mathbf{Q}_a(\xi)$ for all $a = 1, 2, \dots, n$ at some point P with coordinates ξ . For example in the case of the truncated normal distribution in Example 2, computing the vectors $\mathbf{Q}_a$ would require a total of n separate numerical integrations. Computing $\mathbf{Q}_a(\xi)$ from scratch at every step is computationally costly. The Pfaffian systems however gives a family of ODE systems that are satisfied by the vectors $\mathbf{Q}_a$. Given a point ξ_{old} where $\mathbf{Q}_a(\xi_{\text{old}})$ are known, we can use standard ODE solvers such as Runge-Kutta to find the value of $\mathbf{Q}_a(\xi)$ at some other point ξ . In our implementation, we use the R package `hgm` [22] by Takayama et al., which uses the RK4(5)7 method from Dormand and Prince [5].

We can also find the value of $\mathbf{Q}_a$ after a change in θ coordinates. Since $\xi = \mathbf{X}_B \theta$ we have

$$\frac{\partial \mathbf{Q}_a(\xi(\theta))}{\partial \theta} = \frac{\partial \mathbf{Q}_a(\xi)}{\partial \xi} \frac{\partial \mathbf{X}_B \theta}{\partial \theta} = \frac{\partial \mathbf{Q}_a(\xi)}{\partial \xi} \cdot \mathbf{X}_B$$

Again, if both θ_{old} and $\mathbf{Q}_a(\theta_{\text{old}})$ are known, then we can use numerical ODE solvers to obtain $\mathbf{Q}_a(\theta)$.

There are also cases where we need to conduct the holonomic update step with respect to mixed coordinates. Let $J \subset \{0, 1, \dots, d+r\}$ and assume $P_{\text{old}} = (\theta_{\text{old}}^J, \eta_{J_{\text{old}}})$ and $\mathbf{Q}_a(P_{\text{old}})$ are known. In order to obtain $\mathbf{Q}_a(P)$ for some other $P = (\theta^J, \eta_J)$, we will need to find a Pfaffian system for $\mathbf{Q}_a$ in terms of the mixed coordinates

Theorem 2 Let $\emptyset \neq J \subsetneq \{0, 1, \dots, d+r\}$ be a nonempty, strict subset and let $\rho = (\theta^J, \eta_J)$ denote a mixed coordinate. For a vector ρ with $d+r+1$ elements, let

ρ_J denote the subvector $(\rho_j)_{j \in J}$, and similarly $\rho_{\bar{J}} = (\rho_j)_{j \notin J}$. Let

$$\begin{aligned} \theta^* \mathbb{R}^{d+r+1} &\longrightarrow \mathbb{R}^{d+r+1} \\ \rho &\longmapsto \theta \end{aligned}$$

be the function that maps the mixed coordinates ρ to the θ coordinate. Then

$$\frac{\partial \mathbf{Q}_a}{\partial \rho} = \frac{\partial \mathbf{Q}_a}{\partial \theta} \frac{\partial \theta^*}{\partial \rho},$$

where

$$\begin{aligned} \frac{\partial \theta_J^*}{\partial \rho_J} &= \left(\frac{\partial \eta_J}{\partial \theta^J} \right)^{-1} & \frac{\partial \theta_J^*}{\partial \rho_{\bar{J}}} &= \left(\frac{\partial \eta_J}{\partial \theta^J} \right)^{-1} \cdot \left(-\frac{\partial \eta_J}{\partial \theta^{\bar{J}}} \right) \\ \frac{\partial \theta_{\bar{J}}^*}{\partial \rho_J} &= \mathbf{0}_{|\bar{J}| \times |J|} & \frac{\partial \theta_{\bar{J}}^*}{\partial \rho_{\bar{J}}} &= \mathbf{I}_{|\bar{J}| \times |\bar{J}|} \end{aligned}$$

Furthermore, $\frac{\partial \mathbf{Q}_a}{\partial \rho}$ is a function of ρ and $\mathbf{Q}_a(\rho)$.

Proof We wish to find the derivative of $\mathbf{Q}_a$ in terms of some mixed coordinates ρ . We will first convert the mixed coordinates ρ to θ coordinates, and then evaluate the derivative. By the chain rule, we obtain the first part of the theorem

$$\frac{\partial \mathbf{Q}_a(\theta^*(\rho))}{\partial \rho} = \frac{\partial \mathbf{Q}_a}{\partial \theta} \frac{\partial \theta^*}{\partial \rho}.$$

By definition, $\rho_J = \eta_J$ and $\rho_{\bar{J}} = \theta^{\bar{J}}$, and the function θ^* satisfies the following identities

$$\theta^*(\rho)_{\bar{J}} = \theta^{\bar{J}} \quad (18)$$

$$\eta(\theta^*(\rho))_J = \eta_J \quad (19)$$

Let $i \in \bar{J}$ and $j \in J$. Then clearly

$$\frac{\partial \theta_i^*}{\partial \rho_j} = \frac{\partial \theta^i}{\partial \eta_j} = 0,$$

since the components of ρ do not depend on each other. Likewise, if both $i, j \in \bar{J}$, then

$$\frac{\partial \theta_i^*}{\partial \rho_j} = \frac{\partial \theta^i}{\partial \theta^j} = \delta_i^j.$$

Hence we get $\frac{\partial \theta_{\bar{J}}^*}{\partial \rho_J} = \mathbf{0}_{|\bar{J}| \times |J|}$ and $\frac{\partial \theta_{\bar{J}}^*}{\partial \rho_{\bar{J}}} = \mathbf{I}_{\bar{J} \times \bar{J}}$.

Differentiating both sides of (19) by ρ_J , we have

$$\frac{\partial \eta(\theta^*(\rho))_J}{\partial \rho_J} = \frac{\partial \eta_J}{\partial \rho_J},$$

where the right-hand side becomes $\frac{\partial \eta_J}{\partial \eta_J} = \mathbf{I}$, and the left-hand side becomes

$$\frac{\partial \eta(\theta^*(\rho))_J}{\partial \rho_J} = \frac{\partial \eta(\theta^*(\rho))_J}{\partial \theta^*(\rho)} \frac{\partial \theta^*(\rho)}{\partial \rho_J} = \frac{\partial \eta_J}{\partial \theta^J} \frac{\partial \theta^*(\rho)_J}{\partial \rho_J},$$

since $\frac{\partial \theta^*_J}{\partial \rho_J} = \mathbf{0}$. Thus

$$\frac{\partial \eta_J}{\partial \theta^J} \frac{\partial \theta^*(\rho)_J}{\partial \rho_J} = \mathbf{I} \implies \frac{\partial \theta^*(\rho)_J}{\partial \rho_J} = \left(\frac{\partial \eta_J}{\partial \theta^J} \right)^{-1}$$

Finally, differentiate both sides of (19) by $\rho_{\bar{J}}$ to get

$$\frac{\partial \eta(\theta^*(\rho))_J}{\partial \rho_{\bar{J}}} = \frac{\partial \eta_J}{\partial \rho_{\bar{J}}}.$$

The right hand side is equal to $\frac{\partial \eta_J}{\partial \theta^J} = 0$, since once again the elements of ρ do not depend on each other. The left-hand side becomes

$$\begin{aligned} \frac{\partial \eta(\theta^*(\rho))_J}{\partial \theta^*(\rho)} \frac{\partial \theta^*(\rho)}{\partial \rho_{\bar{J}}} &= \frac{\partial \eta(\theta^*(\rho))_J}{\partial \theta^*(\rho)_J} \frac{\partial \theta^*(\rho)_J}{\partial \rho_{\bar{J}}} + \frac{\partial \eta(\theta^*(\rho))_J}{\partial \theta^*(\rho)_{\bar{J}}} \frac{\partial \theta^*(\rho)_{\bar{J}}}{\partial \rho_{\bar{J}}} \\ &= \frac{\partial \eta_J}{\partial \theta_J} \frac{\partial \theta^*(\rho)_J}{\partial \rho_{\bar{J}}} + \frac{\partial \eta_J}{\partial \theta_{\bar{J}}} \frac{\partial \theta^J}{\partial \theta^{\bar{J}}} \\ &= \frac{\partial \eta_J}{\partial \theta_J} \frac{\partial \theta^*(\rho)_J}{\partial \rho_{\bar{J}}} + \frac{\partial \eta_J}{\partial \theta_{\bar{J}}} \mathbf{I}. \end{aligned}$$

Hence

$$\frac{\partial \eta_J}{\partial \theta_J} \frac{\partial \theta^*(\rho)_J}{\partial \rho_{\bar{J}}} + \frac{\partial \eta_J}{\partial \theta_{\bar{J}}} = 0 \implies \frac{\partial \theta^*(\rho)_J}{\partial \rho_{\bar{J}}} = \left(\frac{\partial \eta_J}{\partial \theta_J} \right)^{-1} \left(-\frac{\partial \eta_J}{\partial \theta_{\bar{J}}} \right)$$

Finally, $\frac{\partial \mathbf{Q}_a}{\partial \rho}$ is indeed a function of ρ and $\mathbf{Q}_a(\rho)$, since $\frac{\partial \mathbf{Q}_a}{\partial \theta}, \frac{\partial \eta}{\partial \theta} = \frac{\partial^2 \psi(\theta)}{\partial \theta \partial \theta^T}$ is a function of ρ and $\mathbf{Q}_a(\rho)$ ² based on the discussion in the beginning of Sect. 4.3. $\square$

² after appropriate coordinate conversions.

4.3.2 Holonomic m-projections

Using the results of Theorem 2 we can now carry out m-projections and recover the vector $\mathbf{Q}_a$ at the projected point given the value of $\mathbf{Q}_a$ at the previous point. Let $\emptyset \neq I \subset \{0, 1, \dots, d+r\}$, $i \notin I$ and $\alpha \in \mathbb{R}$. In the algorithm, all of the m-projections will be to the submanifold $M(i, \alpha, I) = \{\theta \mid \theta^i = \alpha, \theta^j = 0 \ (j \notin I)\}$. Let a point P have the dual coordinates θ and η . The m-projection of P onto $M(i, \alpha, I)$ will have the mixed coordinates $\rho = (\theta^I, \theta^i = \alpha, \eta_{I \setminus \{i\}})$. In other words, we first convert the point P to mixed coordinates according to the set $I \cup \{i\}$ to get $\rho_0 = (\theta^{I \cup \{i\}}, \eta_{\overline{I \cup \{i\}}})$, and then send the element θ^i to α to get ρ . Given $\mathbf{Q}_a(\rho_0) (= \mathbf{Q}_a(P))$ and the Pfaffian system, we may now use Theorem 2 to obtain $\mathbf{Q}_a(\rho)$, and thus recover the full θ coordinates from the mixed coordinates.

4.4 Holonomic extended LARS algorithm

The holonomic extended LARS algorithm is our main result. The algorithm works exactly as the extended LARS algorithm described in Sect. 4.2, but now we have also to specify a Pfaffian system for the $\mathbf{Q}_a(\xi)$ as an input in addition to the data (response $\mathbf{y}$ and design matrix $\mathbf{X}$) and the underlying distribution $(u_1(\mathbf{y}), \dots, u_r(\mathbf{y}))$. Again, we describe the algorithm in the submanifold $M \subset S$ (see (14)), so we will mostly ignore the coordinates indexed by $0, d+1, \dots, d+r$ in the vectors μ and θ . We will only compute them at the end of step 5, because they are needed for the initial guesses of the numerical solvers.

We get as an output a sequence of estimators $\hat{\theta}_{(0)}, \dots, \hat{\theta}_{(d)}$, where the k th estimator $\hat{\theta}_{(k)}$ corresponds to a model with $d-k$ covariates. The holonomic extended LARS algorithm thus looks as follows

1. Let $I = \{1, 2, \dots, d\}$, $\hat{\theta}_{(0)} := \hat{\theta}_{\text{MLE}}$, and $k = 0$. Compute $\mathbf{Q}_a(\hat{\theta}_{(0)})$ for all $a = 1, 2, \dots, n$.
2. For all $i \in I$, let $M(i, 0, I) = \{\theta \mid \theta^i = 0, \theta^j = 0 \ (j \notin I)\} = M(I \setminus \{i\})$ and calculate the holonomic m-projection $\bar{\theta}(i, I)$ of $\hat{\theta}_{(k)}$ on $M(i, 0, I)$ and obtain the vectors $\mathbf{Q}_a(\bar{\theta}(i, I))$.
3. Let $t^* = \min_{i \in I} D^{[I]}(\hat{\theta}_{(k)}, \bar{\theta}(i, I))$ and $i^* = \arg \min_{i \in I} D^{[I]}(\hat{\theta}_{(k)}, \bar{\theta}(i, I))$.
4. For any $\alpha^i \in \mathbb{R}$, $i \in I$, let $M(i, \alpha^i, I) = \{\theta \mid \theta^i = \alpha^i, \theta^j = 0 \ (j \notin I)\}$. For every $i \in I$, compute α^i such that the m-projection $\bar{\theta}'(i, \alpha^i, I)$ of $\hat{\theta}_{(k)}$ on $M(i, \alpha^i, I)$ satisfies $t^* = D^{[I]}(\hat{\theta}_{(k)}, \bar{\theta}'(i, \alpha^i, I))$.
5. Let $\hat{\theta}_{(k+1)}^i = \alpha^i \ (i \in I)$ and $\hat{\theta}_{(k+1)}^j = 0 \ (j \notin I)$.
The $(k+1)$ th estimate will have mixed coordinates $\hat{\rho}_{(k+1)} = (\hat{\eta}_0^\theta, \hat{\theta}_{(k+1)}^1, \dots, \hat{\theta}_{(k+1)}^d, \hat{\eta}_{d+1}^\theta, \dots, \hat{\eta}_{d+r}^\theta)$. Use the holonomic update to compute $\mathbf{Q}_a(\hat{\rho}_{(k+1)})$ using the value $\mathbf{Q}_a(\hat{\rho}_{(k)})$, and then use this to obtain the remaining coordinates $\hat{\theta}_{(k+1)}^0, \hat{\theta}_{(k+1)}^{d+1}, \dots, \hat{\theta}_{(k+1)}^{d+r}$.
6. If $k = d-1$, then go to step 7. If $k < d-1$, then go to step 2 with $k := k+1$, $I := I \setminus \{i^*\}$.
7. Let $\hat{\theta}_{(d)} = 0$. Output $\hat{\theta}_{(0)}, \dots, \hat{\theta}_{(d)}$ and quit the algorithm.

As in Sect. 4.2, by looking at the order in which the covariates vanish in the sequence $\hat{\theta}_{(0)}, \dots, \hat{\theta}_{(d)}$, we can determine the order of importance of the covariates.

5 Worked out examples

In this section we will work out the implementation of the Holonomic Extended Least Angle Regression algorithm on two examples. The algorithm was implemented in the R programming language [21], and the code can be found in [7].

5.1 Truncated normal

As in Example 2, we consider a model where each observation comes from a normal distribution restricted to positive values. Its probability density function is

$$p(y \mid \mu, \sigma^2) = \frac{e^{-(y-\mu)^2/(2\sigma^2)}}{\int_0^\infty e^{-(y-\mu)^2/(2\sigma^2)} dy},$$

where $y \in (0, \infty)$, $\mu \in \mathbb{R}$ and $\sigma \in (0, \infty)$. By expanding, we can write the probability density function as a function of natural parameters

$$p(y \mid \xi_1, \xi_2) = \frac{e^{\xi_1 y + \xi_2 y^2}}{\int_0^\infty e^{\xi_1 y + \xi_2 y^2} dy}, \quad (20)$$

where $\xi_1 = \frac{\mu}{\sigma^2}$ and $\xi_2 = -\frac{1}{2\sigma^2}$. From the form above we see that the truncated normal distribution belongs to the exponential family. Note also that the normalizing constant $f(\xi_1, \xi_2) = \int_0^\infty e^{\xi_1 y + \xi_2 y^2} dy$ does not have a closed form expression in general, and converges if and only if $\xi_2 < 0$. A generalization of the truncated normal distribution are the *exponential-polynomial distributions*, of the form

$$f(y \mid \xi_1, \dots, \xi_d) = \frac{\exp(\xi_n y^n + \dots + \xi_1 y)}{\int_0^\infty \exp(\xi_n y^n + \dots + \xi_1 y) dy}$$

for $y > 0$ and $\xi_n < 0$. This family of distributions and their usage with the holonomic gradient method has been studied in Hayakawa and Takemura [8].

We can naturally construct a generalized linear model using the canonical link where each observation is distributed according to the truncated normal distribution. Given a sample $\mathbf{y} = (y_1, \dots, y_n)$, assume that each y_a is independent and distributed according to a truncated normal distribution with a unique mean parameter μ_a and a common variance parameter σ^2 . Hence, using the notation in equation (20) each observation has their own ξ_1 parameter, and ξ_2 is the same in each observation. To make the notation consistent with [10], for each $a = 1, \dots, n$, the “ ξ_1 parameter” of observation a will be called ξ^a and the common “ ξ_2 parameter” will be called ξ^{n+1} .

With this notation, each observation will have the distribution

$$p(y_a \mid \xi^a, \xi^{n+1}) = \frac{e^{\xi^a y_a + \xi^{n+1} y_a^2}}{f(\xi^a, \xi^{n+1})},$$

and since every observation is independent, the joint distribution of $\mathbf{y}$ is

$$p(y_1, \dots, y_n \mid \xi^1, \dots, \xi^n, \xi^{n+1}) = \frac{e^{\sum_{a=1}^n \xi^a y_a + (\sum_{a=1}^n y_a^2) \xi^{n+1}}}{\prod_{a=1}^n f_a(\xi^1, \dots, \xi^{n+1})},$$

where $f_a(\xi^1, \dots, \xi^{n+1}) := f(\xi^a, \xi^{n+1})$.

In the generalized linear model, each observation y_i is explained by a set of d explanatory variables $x_1^i, \dots, x_d^i$. With the canonical link function in particular, the natural parameter is simply an affine combination of the explanatory variables, i.e. for some real numbers $\theta^0, \theta^1, \dots, \theta^d$, we have $\xi^i = \theta^0 + \theta^1 x_1^i + \dots + \theta^d x_d^i$.

We will now define some notation. Let $\mathbf{X} = (x_j^i)$ be the $(n \times d)$ design matrix, $\boldsymbol{\theta}' = (\theta^0, \theta^1, \dots, \theta^d)^T$ and $\boldsymbol{\xi}' = (\xi^1, \dots, \xi^n)^T$. If $\tilde{\mathbf{X}} = [\mathbf{1}_n \mathbf{X}]$, where $\mathbf{1}_n$ is a column vector of size n where each element is 1, then we have $\boldsymbol{\xi}' = \tilde{\mathbf{X}} \boldsymbol{\theta}'$. We can also define a block-diagonal matrix $\mathbf{X}_B$ and vector $\mathbf{Y}$ as

$$\mathbf{X}_B = \begin{bmatrix} \tilde{\mathbf{X}} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ \sum_{a=1}^n y_a^2 \end{bmatrix}$$

As we defined in Sect. 4, we have $\theta^{n+1} = \xi^{n+1}$. If we set $\boldsymbol{\theta} = (\theta^0, \dots, \theta^{d+1})^T$ and $\boldsymbol{\xi} = (\xi^1, \dots, \xi^{n+1})^T$ we have $\boldsymbol{\xi} = \mathbf{X}_B \boldsymbol{\theta}$, and we can write the pdf of the model as

$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \frac{\exp(\mathbf{Y}^T \mathbf{X}_B \boldsymbol{\theta})}{\prod_{a=1}^n f_a(\mathbf{X}_B \boldsymbol{\theta})} = \exp(\mathbf{Y}^T \mathbf{X}_B \boldsymbol{\theta} - \psi(\boldsymbol{\theta})), \quad \text{when } \theta^{d+1} < 0 \quad (21)$$

where $\psi(\boldsymbol{\theta}) = \psi^*(\mathbf{X}_B \boldsymbol{\theta}) = \sum_{a=1}^n \log f_a(\mathbf{X}_B \boldsymbol{\theta})$ is the potential function.

5.1.1 Maximum likelihood estimation

Next we will discuss details regarding maximum likelihood estimation of the model in equation (21). The log-likelihood is easily obtained from equation (21)

$$\ell(\boldsymbol{\theta} \mid \mathbf{y}) = \mathbf{Y}^T \mathbf{X}_B \boldsymbol{\theta} - \psi(\boldsymbol{\theta}) \quad (22)$$

We will use the Holonomic Gradient Method to find the maximum likelihood estimate. Since the Hessian matrix of $\ell(\boldsymbol{\theta} \mid \mathbf{y})$ is easily obtained, we will use the Newton–Raphson method. Since $\psi(\boldsymbol{\theta}) = \psi^*(\mathbf{X}_B \boldsymbol{\theta})$, we can use matrix calculus to obtain

the Hessian and gradient of the log-likelihood function. Indeed, since the gradient is $\frac{\partial \psi}{\partial \theta} = \frac{\partial \psi^*}{\partial \xi} \cdot \mathbf{X}_B$ and the Hessian is $\frac{\partial^2 \psi}{\partial \theta \partial \theta^T} = \mathbf{X}_B^T \cdot \frac{\partial^2 \psi^*}{\partial \xi \partial \xi^T} \cdot \mathbf{X}_B$, we get the gradient and Hessian of the log-likelihood function as follows

$$\begin{aligned} (\nabla \ell)^T &= \frac{\partial \ell}{\partial \theta} = \mathbf{Y}^T \mathbf{X}_B - \frac{\partial \psi^*}{\partial \xi} \cdot \mathbf{X}_B \\ \mathbf{H}_\ell &= \frac{\partial^2 \ell}{\partial \theta \partial \theta^T} = -\mathbf{X}_B^T \cdot \frac{\partial^2 \psi^*}{\partial \xi \partial \xi^T} \cdot \mathbf{X}_B. \end{aligned}$$

The terms $\frac{\partial \psi^*}{\partial \xi}$ and $\partial \xi \partial \xi^T$ are respectively the gradient and Hessian of the potential function, which can be computed using the Pfaffian systems as in Example 2.

There are some numerical issues to consider when using the method outlined above for maximum likelihood estimation. Let $\theta^{(k)}$ be the approximation of the maximum likelihood estimate at the k th iteration of the Newton–Raphson method. The next estimate is expressed as $\theta^{(k+1)} = \theta^{(k)} + \Delta$, and the difference Δ is obtained by solving the linear system

$$\mathbf{H}_\ell(\theta^{(k)})\Delta = -\nabla \ell(\theta^{(k)}).$$

However, there are times where the Newton–Raphson method is “too violent”, and yields a Δ of large magnitude, meaning that $\theta^{(k)}$ and $\theta^{(k+1)}$ are relatively far apart. This in turn increases the error in the holonomic update. Furthermore, there are cases where the Newton–Raphson method yields an iterate which does not belong to the model, i.e. when $\theta_{d+1}^{(k+1)} \geq 0$. In our implementation we solve the problem by shortening the step size γ when the Newton–Raphson method yields an estimate that is either too far from the previous estimate, or an estimate not belonging to the model.

5.1.2 Coordinate conversions

As described in Sect. 4, we have two sets of e-affine coordinates, ξ and θ , and m-affine coordinates, μ and η , along with their potential functions, respectively $\psi^*(\xi)$, $\psi(\theta)$, $\phi^*(\mu)$, and $\phi(\eta)$. The two sets of coordinates are related with

$$\begin{aligned} \xi &= \mathbf{X}_B \theta & \eta &= \mathbf{X}_B^T \mu \\ \psi(\theta) &= \psi^*(\mathbf{X}_B \theta) & \phi(\mathbf{X}_B^T \mu) &= \phi^*(\mu) \end{aligned} \quad (23)$$

Let P be a point on the manifold (21), and assume the vectors $\mathbf{Q}_a(P)$ are known. Given the ξ coordinates of P , we can recover its μ coordinates from the Pfaffian systems in Example 2 since $\mu_i = \frac{\partial \psi^*}{\partial \xi^i}$. If we expand the expressions in the Pfaffian,

we obtain

$$\begin{aligned}\mu_a &= -\frac{1}{2\xi^{n+1}} \left(\frac{1}{f_a} + \xi^a \right) \\ \mu_{n+1} &= -\frac{1}{2\xi^{n+1}} \sum_{a=1}^n (1 + \xi^a \mu_a).\end{aligned}\tag{24}$$

We can also invert (24) to get the coordinate conversion from μ to ξ

$$\begin{aligned}\xi^{n+1} &= -\frac{1}{2(\mu_{n+1} - \sum_{a=1}^n \mu_a^2)} \sum_{a=1}^n \left(1 - \frac{\mu_a}{f_a} \right) \\ \xi^a &= -2\xi^{n+1} \mu_a - \frac{1}{e^{f_a}}\end{aligned}\tag{25}$$

The conversion θ to η is also simple, since we can just compose the transformations in (23) and (24), i.e. $\eta(\theta) = \mathbf{X}_B^T \mu(\mathbf{X}_B \theta)$.

Next we will tackle mixed coordinate conversions. As in Sect. 4.1, let $J \subseteq \{0, 1, 2, \dots, d+r\}$, $\bar{J} = \{0, 1, 2, \dots, d+r\} \setminus J$ and let $P = (\eta_J, \theta^{\bar{J}})$ denote a mixed coordinate. Additionally, assume that the value of the vector $\mathbf{Q}_a$ is known at point P for all $a = 1, 2, \dots, n$.

Newton's method applied to the function $\mathbf{F}$ in (13) will output $\eta_{\bar{J}}$ and θ^J at the same time, thus allowing us to recover the full θ and η simultaneously. With the truncated normal distribution, using Newton's method to convert mixed coordinates converges very quickly given a suitable initial guess. Depending on the situation where mixed coordinate conversion is needed, there are a few suitable initial guesses that work well. Mixed coordinate conversion is needed in three different situations in the algorithm described in Sect. 4.4:

1. m-projections (steps 2, 4). Use the point before the projection as an initial guess.
2. updating $\mathbf{Q}_a$ (steps 2, 4, 5). Use the point before the update as the initial guess.
3. the “wrap-up step” (step 5). Use the estimate $\hat{\theta}_{(k)}$ of the current iteration as the initial guess.

We note again that there are cases where Newton's method outputs a point $(\theta^J, \eta_{\bar{J}})^{(k+1)} = (\theta^J, \eta_{\bar{J}})^{(k)} + \Delta^{(k)}$ that does not belong to the model.³ In our implementation, we simply iteratively half the step $\Delta^{(k)}$ until the resulting point $(\theta^J, \eta_{\bar{J}})^{(k+1)}$ is satisfactory. For the truncated normal distribution, such a scaling of the Newton step is required if $d+1 \in J$ and the element $(\theta^{d+1})^{(k+1)}$ in $(\theta^J, \eta_{\bar{J}})^{(k+1)}$ becomes positive. More precisely, in this case the next iterate becomes

$$(\theta^J, \eta_{\bar{J}})^{(k+1)} = (\theta^J, \eta_{\bar{J}})^{(k)} + \left(\frac{1}{2} \right)^\alpha \Delta^{(k)},$$

³ In Newton's method, $\Delta^{(k)} = (\text{Jac}(F))^{-1} F$, where F is the same as in (13), and both F and $\text{Jac}(F)$ are evaluated at $(\theta^J, \eta_{\bar{J}})^{(k)}$

$$\text{where } \alpha = \left\lceil -\frac{\log\left(-\frac{(\theta^{d+1})^{(k)}}{(\Delta^{d+1})^{(k)}}\right)}{\log 2} \right\rceil.$$

5.2 Weighted truncated normal distribution

For some fixed $c > 0$, consider the distribution given by the unnormalized density function

$$p(y \mid a, b) = y^c \exp(ay + by^2),$$

where $0 \leq y \leq \infty$, and $b < 0$. Its normalizing constant is given by

$$f = \int_0^\infty y^c \exp(ay + by^2) dy.$$

The annihilating ideal and Pfaffian system for f is derived in example 3.

For this example, writing closed form expressions for the coordinate conversions in terms of the f_a and ξ is difficult. We will circumvent this problem by making the vectors $\mathbf{Q}_a$ larger so that ξ can be recovered from the $\mathbf{Q}_a$ alone. As before, let $f_a = \int_0^\infty y^c \exp(\xi^a y + \xi^{n+1} y^2) dy$. We will write $\partial_a f_a$ instead of $\partial f_a / \partial \xi^a$ to simplify notation.

Recall from example 3 that the annihilating ideal of f_a is $\langle \partial_{\xi^{n+1}} - \partial_{\xi^a}^2, \frac{2\xi^{n+1}}{c+1} \partial_{\xi^a}^2 + \frac{\xi^a}{c+1} \partial_{\xi^a} + 1 \rangle$. By looking at the annihilating ideal of f_a we see that

$$\begin{aligned} 2\xi^{n+1} \partial_a^2 f_a + \xi^a \partial_a f_a + (c+1) f_a &= 0 \\ 2\xi^{n+1} \partial_a^3 f_a + \xi^a \partial_a^2 f_a + (c+2) \partial_a f_a &= 0. \end{aligned}$$

Hence if we define $\mathbf{Q}_a = (f_a, \partial_a f_a, \partial_a^2 f_a, \partial_a^3 f_a)^T$, we can recover the full vector ξ from the $\mathbf{Q}_a$ alone by solving the linear system above. Thus the vectors $\mathbf{Q}_a$ allow the recovery of every coordinate: for example, the μ coordinates are given by

$$\begin{aligned} \mu_a = E[Y_a] &= \frac{\int_0^\infty y^{c+1} \exp(y\xi^a + y^2\xi^{n+1}) dy}{\int_0^\infty y^c \exp(y\xi^a + y^2\xi^{n+1})} = \frac{\partial_a f_a}{f_a}, \quad a = 1, 2, \dots, n \\ \mu_{n+1} = E[Y_{n+1}] &= \sum_{a=1}^n E[Y_a^2] = \sum_{a=1}^n \frac{\partial_a^2 f_a}{f_a}. \end{aligned}$$

Mixed coordinate conversion is also greatly simplified; there is no need to use Newton's method. For example, assume we want to find the θ coordinates corresponding to the mixed coordinate ρ . It suffices to know the value of $\mathbf{Q}_a$ at some other point with mixed coordinates $\rho^{(\text{old})}$. Then we can use the holonomic update to compute $\mathbf{Q}_a$ at ρ , and use the system above to recover ξ , from which we can recover θ by using least-squares.

5.3 Computational details

In order to not end up with nearly singular matrices in the algorithm, we will sometimes have to rescale both the design matrix and the response vector. We center and rescale each covariate such that the mean becomes 0 and the standard deviation becomes 1. In other words, if $\mathbf{x}^i$ is the i th column of the design matrix $\mathbf{X}$, the scaling maps

$$x_j^i \mapsto \frac{x_j^i - \bar{\mathbf{x}}^i}{\sigma_i},$$

where $\bar{\mathbf{x}}^i = \frac{1}{n} \sum_{j=1}^n x_j^i$ is the mean, and $\sigma^i = \sqrt{\sum_{j=1}^n (x_j^i - \bar{\mathbf{x}}^i)^2 / (n - 1)}$. Note that as in [10], scaling and centering and scaling the design matrix will not affect the result of the algorithm. In addition, we will scale the response vector $\mathbf{y}$ such that the sample standard deviation equals 1

$$y_i \mapsto \frac{y_i}{\sigma_y} = \frac{y_i}{\sqrt{\sum_{i=1}^n (y_i - \bar{\mathbf{y}})^2 / (n - 1)}}.$$

These scaling operations allow us to keep the orders of magnitude of the elements in the ξ and μ coordinates roughly equal, which in turn make the orders of magnitude of the elements in the θ and η coordinates roughly equal. This is often necessary when doing actual computations, since otherwise many operations involving mixed coordinates (for example the matrix $\mathbf{Jac}(F)$ in Proposition 1) will end up nearly singular, with certain columns several orders of magnitude larger than others.

6 Results

6.1 Simulations

We test the algorithm by first looking at simulated datasets. In all of the simulations we will assume the weighted truncated normal distribution from Sect. 5.2 as the underlying distribution with $c = 0.5$.

First, we will consider a sample of $n = 100$ observations and a design matrix with entries chosen uniformly at random from the range $[0, 1]$. We will give equal weight to every covariate with no intercept term: the true parameter vector θ will be chosen such that $\theta_1 = \dots = \theta_d = 1/d$, $\theta_0 = 0$, and $\theta_{d+1} = -1$. With this setup, we expect that the ordering of the covariates depends only on the magnitude of the coefficient found in the maximum likelihood estimation step at the beginning. The results with varying d are shown in Fig. 1. The algorithm starts on the right of each plot, where the value of each parameter is equal to the maximum likelihood estimate of the full model. At each iteration, we compute the divergence of the current parameters compared to the empty model, and we plot the value of each parameter. As expected, in all cases the path towards zero of each covariate seems unaffected by other covariates.

Fig. 1 Simulation of 100 observations with uncorrelated covariates

Next we will introduce some correlation structures between the covariates. In Fig. 2 we have 100 samples with 5 covariates, where the first four columns of the design matrix are chosen uniformly at random from the range $[0, 1]$. The fifth column will be equal to $\mathbf{X}_5 = \mathbf{X}_4 + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$, and σ is allowed to vary. We will choose θ such that each covariate has equal weight, as in the previous paragraph. We observe that as σ gets smaller, the paths for X_4 and X_5 become much more correlated, with a steeper slope than the other three covariates. Once either X_4 or X_5 gets eliminated, the slope becomes more like X_1, X_2, X_3 .

In Figs. 3 and 4 we simulate different kinds of correlation structures. For both examples, we still have $n = 100$ observations, and $\theta = 1/d$ unless otherwise specified. In Fig. 3, we set $\theta_1 = 0$ and $X_5 = X_4 \cdot X_3$. We note that the covariate X_1 is quickly eliminated since its value does not affect the observations at all.

In Fig. 4, we set covariates X_4 and X_5 equal to X_3 plus Gaussian noise with $\sigma = 0.05$. The covariates X_3, X_4, X_5 contain redundant information, so X_4 and X_5 get eliminated early.

Fig. 2 Simulation where $X_5 = X_4 + \epsilon$, where ϵ is Gaussian noise with different values of σ

Fig. 3 $X_5 = X_4 \cdot X_3$ and $\theta_1 = 0$

Fig. 4 $X_5 = X_3 + \epsilon_1$ and $X_4 = X_3 + \epsilon_2$

Table 1 Sum of square errors (SSE) for different submodels of the concrete dataset

Subset	SSE
(a) All covariates	72.8
(b) Water, fine aggr., slag	74.4
(c) Water, fly ash, slag	74.5
(d) Water, SP, fly ash	79.0

In (a), we consider the model with all variables as a baseline. Rows (b), (c) and (d) contains the top 3 subset given by our algorithm, [2] and [24] respectively

6.2 Real dataset

As a real life dataset, we will use a dataset on slump flow of concrete from [24]. The set has 103 data points with 7 input variables representing proportions of the following components in concrete: cement, blast furnace slag, fly ash, water, superplasticizer (SP), coarse aggregate and fine aggregate. The response variable is the slump flow of the concrete (in cm), which by definition cannot be negative. This motivates the use of the normal distribution truncated to only positive values. The resulting plot is shown in Fig. 5. The results in [24] assert that water, superplasticizer and fly ash have strong effects. This is in contrast to the results shown by our algorithm, which show that the three parameters with the most predictive power are water, fine aggregate and slag, and the effect of fly ash and superplasticizer is relatively weak. In [2], Antoniadis et al. studied different variable selection algorithms on the same concrete dataset. Their results also show that the effects of fly ash and superplasticizer in [24] have probably been overestimated, and their methods show that slag and water have the highest influence, with one method choosing fly ash as well. We compared the sum of square errors for these subsets in Table 1, and observe that for a subset of size 3, our ranking achieves the lowest SSE value.

Fig. 5 HELARS algorithm run on concrete dataset. The covariates are numbered as follows: 1. cement, 2. slag, 3. fly ash, 4. water, 5. SP, 6. coarse aggr., 7. fine aggr

7 Discussion

In this manuscript, we presented the holonomic extended LARS algorithm, and successfully implemented in R. The algorithm was tested on both simulated and real datasets, and the output was shown to be consistent with our expectations.

We conclude that the dually flat structure of the manifold of exponential families is still useful even when the potential function is not easy to compute. The HELARS implementation is slower than the ELARS implementation due to the overhead caused by keeping track of the $\mathbf{Q}_a$ and constantly updating it using the holonomic gradient method. The benefits of using the holonomic gradient method are most visible when the potential function does not have a closed form expression. Then we can either find a Pfaffian system for the potential function by hand, as in our example, or use the theory of D -modules to construct the Pfaffian system from a holonomic ideal annihilating the potential function. Since in exponential families the potential function is the integral of an exponential function, finding the annihilating ideal is relatively easy in many cases. If the resulting ideal is holonomic, we can then use the integration algorithm [18] to get the annihilating ideal of the integral. In cases where the coordinate conversions are not obvious, such as the weighted normal distribution, we can exploit the annihilating ideal to derive relations between the derivatives which allow the recovery of coordinates from $\mathbf{Q}_a$ alone. Since we keep track of $\mathbf{Q}_a$ throughout the algorithm, this observation makes coordinate conversions trivial, but may make the vectors $\mathbf{Q}_a$ larger than necessary.

In its current state, since the algorithm can only handle a certain class of generalized linear models using the canonical link function, a natural next step would be to extend it to an arbitrary generalized linear model.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Agresti, A.: Foundations of Linear and Generalized Linear Models. Wiley Series in Probability and Statistics. Wiley, Hoboken (2015)
2. Antoniadis, A., Gijbels, I., Verhasselt, A.: Variable selection in additive models using p-splines. *Technometrics* **54**(4), 425–438 (2012)
3. Ascher, U.M., Greif, C.: A first course in numerical methods. In: Computational Science and Engineering, vol. 7. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (2011). <https://doi.org/10.1137/1.9780898719987>
4. Augugliaro, L., Mineo, A.M., Wit, E.C.: Differential geometric least angle regression: a differential geometric approach to sparse generalized linear models. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **75**(3), 471–498 (2013). <https://doi.org/10.1111/rssb.12000>
5. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Math.* **6**(1), 19–26 (1980). [https://doi.org/10.1016/0771-050X\(80\)90013-3](https://doi.org/10.1016/0771-050X(80)90013-3)
6. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. *Ann. Stat.* **32**(2), 407–499 (2004). <https://doi.org/10.1214/009053604000000067>
7. Härkönen, M.: Holonomic extended LARS R implementation. <https://github.com/haerski/HELARS> (2019)
8. Hayakawa, J., Takemura, A.: Estimation of exponential-polynomial distribution by holonomic gradient descent. *Commun. Stat. Theory Methods* **45**(23), 6860–6882 (2016). <https://doi.org/10.1080/03610926.2014.968735>
9. Hibi, T. (ed.): Gröbner Bases: Statistics and Software Systems. Springer, Tokyo (2013). <https://doi.org/10.1007/978-4-431-54574-3>
10. Hirose, Y., Komaki, F.: An extension of least angle regression based on the information geometry of dually flat spaces. *J. Comput. Graph. Stat.* **19**(4), 1007–1023 (2010). <https://doi.org/10.1198/jcgs.2010.09064>. Supplementary materials available online
11. Hirose, Y., Komaki, F.: Edge selection based on the geometry of dually flat spaces for Gaussian graphical models. *Stat. Comput.* **23**(6), 793–800 (2013). <https://doi.org/10.1007/s11222-012-9347-3>
12. Hirose, Y., Komaki, F.: An estimation procedure for contingency table models based on nested geometry. *J. Jpn. Stat. Soc.* **45**(1), 57–75 (2015). <https://doi.org/10.14490/jjss.45.57>
13. Hyvärinen, A.: Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.* **6**, 695–709 (2005)
14. Koyama, T., Takemura, A.: Calculation of orthant probabilities by the holonomic gradient method. *Jpn. J. Ind. Appl. Math.* **32**, 187–204 (2015). <https://doi.org/10.1007/s13160-015-0166-8>

15. McCullagh, P., Nelder, J.A.: Generalized linear models. Monographs on Statistics and Applied Probability. Chapman and Hall, London (1989). <https://doi.org/10.1007/978-1-4899-3242-6>. Second edition [of MR0727836]
16. Minka, T.: Old and new matrix algebra useful for statistics (1997). <https://www.microsoft.com/en-us/research/publication/old-new-matrix-algebra-useful-statistics/>
17. Nakayama, H., Nishiyama, K., Noro, M., Ohara, K., Sei, T., Takayama, N., Takemura, A.: Holonomic gradient descent and its application to the Fisher–Bingham integral. *Adv. Appl. Math.* **47**(3), 639–658 (2011). <https://doi.org/10.1016/j.aam.2011.03.001>
18. Oaku, T.: Algorithms for b -functions, restrictions, and algebraic local cohomology groups of D -modules. *Adv. Appl. Math.* **19**(1), 61–105 (1997). <https://doi.org/10.1006/aama.1997.0527>
19. Park, M.Y., Hastie, T.: L_1 -regularization path algorithm for generalized linear models. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **69**(4), 659–677 (2007). <https://doi.org/10.1111/j.1467-9868.2007.00607.x>
20. Parry, M., Dawid, A.P., Lauritzen, S.: Proper local scoring rules. *Ann. Stat.* **40**(1), 561–592 (2012). <https://doi.org/10.1214/12-AOS971>
21. R Core Team: R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna (2017). <https://www.R-project.org/>
22. Takayama, N., Koyama, T., Sei, T., Nakayama, H., Nishiyama, K.: hgm: holonomic gradient method and gradient descent (2017). <https://CRAN.R-project.org/package=hgm>. R package version 1.17
23. Tibshirani, R.: Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B (Methodol.)* **58**(1), 267–288 (1996). <http://www.jstor.org/stable/2346178>
24. Yeh, I.C.: Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement Concrete Composites* **29**(6), 474–480 (2007)
25. Yuan, M., Lin, Y.: Model selection and estimation in the Gaussian graphical model. *Biometrika* **94**(1), 19–35 (2007). <https://doi.org/10.1093/biomet/asm018>
26. Yukawa, M., Amari, S.I.: ℓ_p -regularized least squares ($0 < p < 1$) and critical path. *IEEE Trans. Inform. Theory* **62**(1), 488–502 (2016). <https://doi.org/10.1109/TIT.2015.2501362>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.