Lazy lasso for local regression

Diego Vidaurre · Concha Bielza · Pedro Larrañaga

Abstract Locally weighted regression is a technique that predicts the response for new data items from their neighbors in the training data set, where closer data items are assigned higher weights in the prediction. However, the original method may suffer from overfitting and fail to select the relevant variables. In this paper we propose combining a regularization approach with locally weighted regression to achieve sparse models. Specifically, the lasso is a shrinkage and selection method for linear regression. We present an algorithm that embeds lasso in an iterative procedure that alternatively computes weights and performs lasso-wise regression. The algorithm is tested on three synthetic scenarios and two real data sets. Results show that the proposed method outperforms linear and local models for several kinds of scenarios.

Keywords Lasso · l1-regularization · Variable selection · Loess · Locally weighted regression · Sparse models · Lazy lasso · Nonparametric variable selection

1 Introduction

Let $X_1, \ldots, X_p$ denote independent covariates and Y a response variable. Multiple linear regression is a widely used method for determining the influence of the covariates on the response. This influence is modelled by a linear combination of some of the covariates, chosen to minimize a least squares function.

D. Vidaurre (✉) · C. Bielza · P. Larrañaga
Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain
e-mail: diego.vidaurre@fi.upm.es

C. Bielza
e-mail: mcbielza@fi.upm.es

P. Larrañaga
e-mail: pedro.larranaga@fi.upm.es

Let $\mathscr{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots, (\boldsymbol{x}^{(n)}, y^{(n)})\}$ be a data set containing a set of n points in the covariates space and the response, where $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)})^T$.

Let $\boldsymbol{X}$ denote the $n \times p$ matrix whose ith row is the p-vector $\boldsymbol{x}^{(i)}$ and let $\boldsymbol{y} = \left(y^{(1)}, y^{(2)}, \ldots, y^{(n)}\right)^T$ be the vector of responses. Provided data are standardized, the common linear regression model assumes a relationship such that

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^T$ are the regression coefficients. It is assumed that the stochastic unobserved component $\boldsymbol{\epsilon}$ is distributed

$$\epsilon_i \sim \mathcal{N}\left(0, \sigma_i^2\right), \quad i = 1, \ldots, n. \tag{2}$$

Hence, there are p parameters $\beta_1, \ldots, \beta_p$ to be determined. Ordinary least squares (OLS) estimate such parameters by minimizing the sum of the squares of the distances from the true response to the fitted response:

$$\operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} \beta_j \right)^2. \tag{3}$$

Typical model assumptions are Gaussianity, independence and homogeneity of variance of the components of $\boldsymbol{\epsilon}$. Since OLS is based on empirical loss minimization, it might overfit the data. Regularization techniques add a penalization term to the usual regression. This prevents overfitting, reduces the variance of the estimates and gives rise to more interpretable models. Two widely used methods are *ridge* regression (Hoerl and Kennard 1970) and the *least absolute shrinkage and selection operator* (Tibshirani 1996). We focus here on the least absolute shrinkage and selection operator, commonly referred to as *lasso* or l_1-regularization. A significant property of the lasso is its ability to move many regression coefficients to zero, performing variable selection (sparse models) at the same time as prediction. Like ridge regression, the lasso has a better prediction stability than OLS. For a recent overview of the lasso see (Hesterberg et al. 2008). The LARS algorithm (Efron et al. 2004) is a variable selection and regression method that could be considered an efficient version of the forward stagewise regression algorithm (Weisberg 1980). With a slight modification, LARS very efficiently solves the lasso.

Thanks to the variable selection capability, l_1-regularization is widely used in practice, especially when p is much greater than n. These scenarios have become of increasing importance in the last decades. Problems related to computational biology, like genomics and proteomics (Larrañaga et al. 2006) or neuroscience (Kass et al. 2005), are of special interest. High variance and overfitting are major issues when dealing with this kind of data. L_1-regularization and variants have been shown to be particularly helpful in this setting, where very simple models are preferred; see, for example, (Ma et al. 2007; Grosenick et al. 2008). This is supported by a large amount of theoretical work; see for example (Donoho 2006; Meinshausen and Yu 2009).

However, the response variable cannot be always predicted by means of a simple linear function of the covariates. In this case, some kind of nonlinear analysis may be required. In general, nonlinear regression procedures (Seber and Wild 1989) intend to fit data to any selected equation, finding the values of the parameters that minimize the sum of the squared distances from the true response to the estimated curve.

Regression trees (RTs) (Breiman et al. 1984) are a particular kind of nonlinear regression model that also perform variable selection. Basically, RTs recursively divide the space into smaller regions, depending on the value of certain variables. At each leaf of the tree there is a simpler, linear regression, or just a single response.

Another effective alternative is to employ some form of local learning or nonparametric approach (Fan 1992; Ruppert and Wand 1994), which does not make any assumption on the form of the global function. A common method is *loess*, a *locally weighted regression* procedure built on classical least squares regression (Cleveland 1979; Cleveland and Devlin 1988). In loess, for each point in the covariate space, there is a neighborhood containing the point where the regression surface is well approximated by a function from a parametric class. A weighted residual sum of squares, instead of the residual sum of squares, is minimized. The weights are provided by a function of the distances between the data and the point of interest, attaching more importance to closer points. Locally weighted regression avoids the ϵ homoscedasticity assumption.

There have been some attempts to combine local learning and variable selection with regularization. A recent nonparametric procedure involving variable selection is the *regularization of derivative expectation operator*, referred to as *rodeo* (Lafferty and Wasserman 2008), which explicitly keeps one different bandwidth parameter for each variable, indicating the size of the neighborhood for this variable. This method performs simultaneous bandwidth and variable selection by computing the infinitesimal change in the estimation as a function of the bandwidths, and then thresholding these derivatives to achieve a sparse estimate. Following a greedy strategy, rodeo updates the bandwidths at each step, so that the bandwidths of relevant variables are shrunk more whereas the bandwidths of irrelevant variables remain larger. To compute this update, a $p \times p$ matrix inversion is required at each step. Although the rodeo is proved to have nice theoretical properties, it assumes that relevance of the variables depends just on how much they depart from the linear model. This can lead to a wrong estimation in specific scenarios.

A lasso penalty has been built into the varying coefficient model with local kernel estimation (Wang and Xia 2009). Each data item is associated with a univariate index variable, ranging between 0 and 1, so that data items with similar indexes will also have similar regression coefficients. All vectors of regression coefficients, one per data item, are jointly estimated. Some appealing theoretical properties of the proposed algorithm are shown. Since the closeness between data items is supposed to be known a priori, there is no need to calculate distances. Hence this approach evades the problems stated below.

A different form of local analysis is spatial analysis, where the influence of the covariates on the response follows different patterns according to the spatial location of the data, typically 2-dimensional coordinates representing the data items. The *expansion method* (Jones and Casetti 1992) is a local procedure allowing the parameters

to be functions of other attributes, such as location. Another well-known algorithm is *geographically weighted regression* (GWR) (Fotheringham et al. 2002). In GWR, weights are assigned to data so that nearer data items are given more importance than further data items. The main difference from loess is that the distances are computed on separate spatial coordinates rather than on the covariate space. In the field of spatial analysis, the recent *geographically weighted lasso* (GWL) (Wheeler 2009) introduces a lasso-wise penalization on the GWR-estimated coefficients.

Our contribution is a method based on lasso for both local prediction and local variable selection. Some of the estimated local regression coefficients will be exactly zero, making variable selection more explicit than for rodeo. The setting is a scenario where usual linear regression is not appropriate, and a local approach would appear to be better suited. The proposed algorithm makes use of LARS to solve the lasso. Unlike GWL (and just like loess), there is no explicit spatial coordinates for each data item, and the distances between data items are calculated in the covariate space.

A possible naïve approximation would be to add an l_1-penalty to the locally weighted regression so as to reach a sparse solution (some regression coefficients equal to zero). This implies that we have to calculate the weights (distances) before performing variable selection. However, ideally, only relevant covariates should be involved in the weights calculation. If the solution is sparse, there will be several irrelevant covariates involved in the weights calculation, yielding an incorrect weighting scheme and a rather inaccurate prediction. The problem is that distances are calculated prior to the regression, and hence before we know what variables are relevant for prediction.

To overcome this obstacle, we suggest an iterative algorithm that alternates variable selection and distance computation. At each step, distances are computed using the current variables in the model, and weights are assigned to data for the next l_1-regression. We use a single overall bandwidth parameter, and the effective bandwidth of each variable is adaptively adjusted in the distance calculation stage. Besides, since, at each step, only a subset of the variables is involved, unlike rodeo, we mostly avoid large matrix inversions. Also, we devise a validation procedure that implies no additional cost.

In the framework of functional data analysis, the stepwise algorithm proposed by Ferraty et al. (2010) is related to ours. Whereas Ramsay and Silverman (2005) have popularized the functional data analysis field, the first nonparametric contributions are described by Ferraty and Vieu (2006) and more recently by Ferraty et al. (2010). Now, we have an infinite (or very high) dimensional functional variable and a scalar response. The goal is to reduce the very high set of predictors to a set of highly predictive points, called design points. This method performs a greedy, forward addition of variables guided by the cross-validated error. At each step, the variable that, together with the variables currently in the model, most improves the accuracy is selected. A backward deletion process is subsequently applied.

There are, however, substantial differences between this method and ours. First, Ferraty et al's method is not a lazy approach, that is, it is not focused on a certain point of interest and considers the whole data set. Hence, the goals are different, although both algorithms could be adapted to pursue any objective. Second, the estimation procedure and the search strategy also differ. Ferraty et al. (2010) take the weighted

least squares estimation at each step. Instead, we obtain a weighted l_1-regularized estimation, choosing the extent of regularization that most improves the model. Given that the whole l_1-regularization path can be obtained at the same cost that a least squares fit, and since their method performs one least squares fit per candidate variable at each step, our method is computationally more efficient when the number of variables is high. This is supported also for a cheap validation procedure. Furthermore, whereas their method adds one variable at a time (and deletes one variable at a time afterwards), our method can add and delete several variables at each step, possibly enlarging the number of different visited models. Finally, we compute the adaptive bandwidths directly as a function of the importance of each variable. Ferraty et al. (2010), on the other hand, perform a heuristic search for this purpose.

Our method is appropriate when the influence of the covariates on the response is sparse and nonlinear. Since the algorithm is computationally more expensive linear, simpler methods, the analyst should first check that it is adequate. For example, data nonlinearity could be examined by charting the response against some of the variables separately, for example by a scatter plot matrix with all the pair-wise scatter plots of the variables in a matrix format. If p is high, such a procedure could be tedious. In this case, there is a need for a sparser estimation, which is the other goal of the algorithm.

The paper is organized as follows. Section 2 describes local regression and LARS/lasso in detail. Section 3 states the novel algorithm, called the *lazy lasso*. Section 4 includes the set of experiments used to test the algorithm. Finally, in Sect. 5 we sum up the paper with conclusions and future work.

2 Foundations

2.1 Local regression

The local regression method was originally devised for time series, where events that are close in time are expected to share common patterns. Although the locally weighted regression paradigm is not limited to local linear fitting, we will work in this paper with linear functions only. The loess procedure (Cleveland 1979) is a popular locally weighted regression technique. Devlin (1986) discussed a number of mathematical properties of loess. Hastie and Loader (1993) listed some advantages of using local regression. For example, local regression overcomes the biasing problems of other methods, generalizes easily to high-order polynomials fitting, and is relatively insensitive to data design.

Assuming standardized data, loess estimates the regression coefficients for $\boldsymbol{x}^{(l)}$, $l \notin \{1, \ldots, n\}$ by minimizing

$$RSS^{(l)}(\boldsymbol{\beta}^{(l)}) = \sum_{i=1}^{n} \left(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} \beta_j^{(l)} \right)^2 \cdot g_\tau \left(d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(l)}) \right), \tag{4}$$

where $\boldsymbol{\beta}^{(l)} = (\beta_1^{(l)}, \beta_2^{(l)}, \ldots, \beta_p^{(l)})^T$ are the regression cofficients for the point of interest $\boldsymbol{x}^{(l)}$, $\tau \in (0, 1]$ is the bandwidth constant that determines the size of the

neighborhood of $\boldsymbol{x}^{(l)}$ to be included in the regression, $g_\tau(\cdot)$ is a weight function and $d(\cdot)$ is a distance function.

Let $\boldsymbol{w}^{(l)} = (w_1^{(l)}, \ldots, w_n^{(l)})^T$ be the vector of weights, with components

$$w_i^{(l)} = \sqrt{g_\tau\left(d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(l)})\right)}, \quad i = 1, \ldots, n, \tag{5}$$

and let $\boldsymbol{W}^{(l)}$ be the diagonal matrix related to the local regression for $\boldsymbol{x}^{(l)}$, whose elements are $w_i^{(l)}$. Then, the vector of coefficients can be estimated as

$$\hat{\boldsymbol{\beta}}^{(l)} = \left(\boldsymbol{X}^T \left(\boldsymbol{W}^{(l)}\right)^T \boldsymbol{W}^{(l)} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \left(\boldsymbol{W}^{(l)}\right)^T \boldsymbol{W}^{(l)} \boldsymbol{y} = \left(\boldsymbol{Z}^T \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^T \boldsymbol{v}, \tag{6}$$

where $\boldsymbol{Z} = \boldsymbol{W}^{(l)}\boldsymbol{X}$ and $\boldsymbol{v} = \boldsymbol{W}^{(l)}\boldsymbol{y}$ are the weighted covariate matrix and the weighted response vector, respectively.

There are four key aspects when considering loess: the parametric family to be locally fitted, the fitting criterion, the weight function and the bandwidth (Cleveland and Loader 1996).

As mentioned above, we are focusing on the linear parametric family. Assuming the experimental errors (Eq. (2)) to be uncorrelated and Gaussian-distributed, least squares is a natural choice for the fitting criterion. On the whole, the parametric family and the fitting criterion depend on assumptions concerning the nature of the data and the distribution of the response.

Regarding the weight function, any weight function that satisfies the properties listed in (Cleveland 1979) may be used. Specifically, $g_\tau(\cdot)$ must be a nonincreasing, symmetric and positive function defined in $\mathbb{R}^+$.

Finally, the choice of the bandwidth is crucial. This parameter controls how narrow or wide the neighborhood used to make the estimation is. Nature of the data, cardinality and dimension are important for correct bandwidth selection. For instance, the *curse of dimensionality* states that as the dimension p increases, the points quickly become sparse. In this case it is a good idea to increment the bandwidth to offset this effect. The bandwidth may be set beforehand at a constant value, chosen as a function of the kth nearest neighbor (Cleveland 1979) (k needs to be selected) or adaptively selected for each new data item (Fan and Gijbels 1992). Adaptive selection is particularly appropriate for online training (Cleveland and Loader 1996) and has some advantages in any case.

A key issue is the adequacy of local regression for high dimensionality settings. Fowlkes (1987) presented some validation tests to rate the adequacy of smoothing in binary logistic regression for high dimensions. This is equivalent to the loess smoothing procedure. In short, his analysis shows that the results are still reliable for high values of p if n is large enough, although the inclusion of irrelevant variables has quite a negative effect on the smoothing process. This is precisely the point we tackle in this paper.

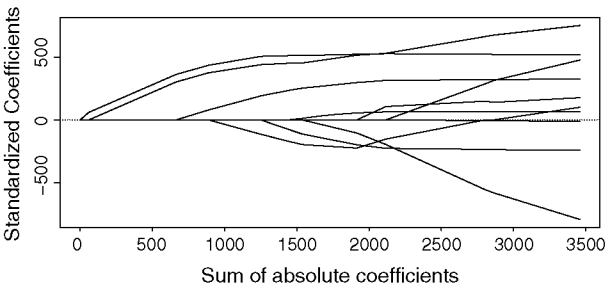

Fig. 1 Regularization path for the *Diabetes* data set

2.2 LARS/lasso

Nowadays, lasso is meeting with great acceptance in the machine learning research community. At the time of writing, the original reference (Tibshirani 1996) had received over 3,500 cites according to Google Scholar. The lasso estimate is

$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{7}$$

Unlike ridge regression, the lasso forces regression coefficients to become exactly zero as the tuning parameter λ is increased. In this way it simultaneously performs variable selection and parameter estimation. The complete solution of the lasso for all values of λ forms the *regularization path*. Figure 1 depicts the regularization path for the *Diabetes* data set (Efron et al. 2004). The Y-axis represents the magnitude for the regression coefficients and the X-axis represents the sum of absolute coefficients. Each coefficient is represented by a different line.

The regularization path usually starts with a large λ, where all coefficients are equal to zero. As λ decreases, one coefficient at a time is made different from zero, although, from time to time, any non-zero variable may also exit the model. For variable selection purposes, we are concerned with a finite set of λ values only, specifically the λ values that lead to changes in the number of non-zero coefficients. The increments on the coefficients between two consecutive values of λ in such a set are linear. This property means that the regularization path is referred to as a *piecewise linear* path.

The lasso is a quadratic programming problem with a linear inequality constraint. With slight modifications, however, the LARS algorithm (Efron et al. 2004) is able to calculate the whole lasso regularization path for a given problem with the same cost as OLS. In short, LARS is an iterative algorithm that starts with an empty set of selected variables (non-zero regression coefficients) and adds one variable to this set at each step. This is the variable with the highest absolute value of the correlation with the current residuals. The vector of correlations is

$$\boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{y}}), \tag{8}$$

where $\hat{\boldsymbol{y}}$ is the predicted response based on the covariates of the set of selected variables. The coefficients of the variables in the set of selected variables are increased in the direction of the OLS fit based on these variables. A new variable is selected when its correlation with the residuals equals that of the elements of the current set of selected variables. The whole set of selected variables has the same correlation with the residuals.

Regarding the mathematical properties, there is a great deal of theoretical work supporting the lasso. For example, Knight and Fu (2000) demonstrated the consistency of the lasso for fixed p and $n \to \infty$, that is, they show that the lasso selects the true sparse model under these conditions. Zhao and Yu (2006) discussed the consistency of the lasso under certain conditions in the large p setting. This is important, because the lasso is specially suitable when n is not big compared to p. There are also some variations on the original lasso that improves its properties, e.g. (Zou 2006; Foster et al. 2008).

3 The lazy lasso

3.1 The algorithm

Cleveland and Devlin (1988) discussed the need to incorporate a variable selection procedure into the loess methodology if required, i.e. if we suspect the presence of irrelevant variables. Taking up this argument, we present an algorithm that combines l_1-regularization with the usual locally weighted regression paradigm.

A first possible approach is equivalent to the GWL algorithm (Wheeler 2009), that is, directly applying a set of weights to the data set and then launching LARS. LARS thus solves a weighted lasso problem, simultaneously performing variable selection and local-level regression. The weights would be obtained from some transformation of the distances of each data item to the point of interest $\boldsymbol{x}^{(l)}$. As in loess, the distances are calculated in the covariate space. Although this is simple and easy to implement, the distance calculation involves irrelevant variables. Therefore, we claim that this method is naïve and ineffective, and it is expected to lead to incorrect predictions and incorrect feature selections. This effect will be more pronounced for a large number of irrelevant variables. We will call this method the *naïve lazy lasso*.

We assume the hypothesis of local homoscedasticity for $\boldsymbol{\epsilon}$ in (2), that is, σ_i is supposed to be constant within a certain neighborhood. We are interested in the set of local regression coefficients $\hat{\boldsymbol{\beta}}^{(l)}$ minimizing

$$\sum_{i=1}^{n} \left(w_i^{(l)} y^{(i)} - \sum_{j=1}^{p} w_i^{(l)} x_j^{(i)} \beta_j^{(l)} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{9}$$

where

$$w_i^{(l)} = \sqrt{g_\tau \left(d_{\boldsymbol{\delta}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(l)}) \right)}, \quad i = 1, \ldots, n. \tag{10}$$

We define the distance function as

$$d_{\boldsymbol{\delta}}\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(l)}\right) = \sqrt{\sum_{j=1}^{p} \delta_j \left(x_j^{(i)} - x_j^{(l)}\right)^2}, \quad i = 1, \ldots, n, \tag{11}$$

where vector $\boldsymbol{\delta}$ is defined so that the distance calculation attaches more importance to relevant variables. The simplest choice is to set $\delta_j = 1$ if X_j is relevant for the prediction of the response, and $\delta_j = 0$ otherwise. A more convenient definition sets δ_j as a smooth function of β_j:

$$\delta_j = p \frac{|\beta_j|}{\sum_{j'=1}^{p} |\beta_{j'}|}, \tag{12}$$

so that $\sum_{j=1}^{p} \delta_j = p$. In both cases, $\beta_j = 0$ leads to $\delta_j = 0$, and irrelevant variables are not included in the distance calculation. $\boldsymbol{\delta}$ can be considered a vector of adaptive bandwidths.

Since this problem cannot be solved analytically, we propose an iterative procedure, the *lazy lasso*, that calculates distances based on the current $\boldsymbol{\delta}$ vector at each step.

In the first iteration, we let $\delta_j = 1$, $\forall j \in \{1, \ldots, p\}$. We calculate distances from Eq. (11) and weights from Eq. (10). As in loess, we weight the data set. Then, the LARS algorithm is run on this weighted data set to solve the minimization in (9). From the resulting LARS regularization path, we select the best vector of regression coefficients according to some criterion (see below for details). We update $\boldsymbol{\delta}$ according to this vector of regression coefficients using (12). Distances and weights are again recalculated using the new $\boldsymbol{\delta}$ vector, and the data set is weighted. Subsequently, LARS is run again over this weighted data set. The algorithm alternates LARS and weights calculation until some stopping criterion is met. Here, we stop the process when there is no improvement in the best score for a given number of iterations.

The pseudocode in Algorithm 1 roughly outlines the method. Here, $\mathbf{d}$ is a vector of distances, ***path*** is the LARS regularization path and $\boldsymbol{\beta}^*$ is the best set of regression coefficients at each iteration. The $evaluate(\cdot)$ and $best(\cdot)$ functions are based on the validation procedures that we detail in the next section. In the pseudocode, we obtain $\boldsymbol{\delta}$ from Eq. (12). The algorithm terminates if there is no improvement in the best score for κ iterations.

To keep the local homoscedasticity assumption for the weighted data set, we have chosen $g_\tau(\cdot)$ as a k-nearest neighborhood function. This function assigns $w_i = 1$ for the τn data items closest to $\boldsymbol{x}^{(l)}$ and $w_i = 0$ otherwise. Hence, the weighted data set is just a subset of the original data set with constant σ_i for because they are the closest points. As discussed below, local homoscedasticity is a requirement for the validation procedure. Although other functions, like the *Epanechnikov* function or the *tricube* function, are much more frequent in local regression, the weight function mainly affects the visual quality of the regression curve and does not significantly influence the prediction accuracy (Loader 1999). However, both the *Epanechnikov* and the *tricube* functions alter the data set so that we cannot assume σ_i to be homogeneous within a certain neighborhood.

Algorithm 1 lazy lasso

Input: training data set $\mathscr{D}$ with p variables and n data items
Input: bandwidth τ and stopping criterion parameter κ
Input: weight function $g(\cdot)$ and distance function $d(\cdot)$
Input: point $\boldsymbol{x}^{(l)}$, whose response is to be predicted
Output: set of coefficients $\hat{\boldsymbol{\beta}}^{(l)}$ and estimated response $\hat{y}^{(l)}$
Initialization:
$\delta_j := 1$, for $j = 1, \ldots, p$
$overallBest := \infty$; $toStop := 0$
repeat
  Calculate all distances $\mathrm{d}_i := d_{\boldsymbol{\delta}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(l)})$, for $i = 1, \ldots, n$
  $\boldsymbol{w}^{(l)} := \sqrt{g_\tau(\mathbf{d})}$
  $\boldsymbol{W}^{(l)} := n \times n$ diagonal matrix, $W_{ii}^{(l)} = w_i^{(l)}$, for $i = 1, \ldots, n$
  $\boldsymbol{Z} := \boldsymbol{W}^{(l)}\boldsymbol{X}$
  $\boldsymbol{v} := \boldsymbol{W}^{(l)}\boldsymbol{y}$
  $\boldsymbol{path} := LARS(\mathbf{Z}, \boldsymbol{v})$
  $\boldsymbol{\beta}^* := best(\boldsymbol{path}; \mathbf{Z}, \boldsymbol{v})$
  $\delta_j := p|\beta_j| / \sum_{j'=1}^{p} |\beta_{j'}|$, for $i = 1, \ldots, n$
  $score := evaluate(\boldsymbol{\beta}^*; \mathbf{Z}, \boldsymbol{v})$
  if $score \geq overallBest$ **then**
    $toStop := toStop + 1$
  else
    $toStop := 0$
    $overallBest := score$
    $\hat{\boldsymbol{\beta}}^{(l)} := \boldsymbol{\beta}^*$
  end if
until $toStop = \kappa$
$\hat{y}^{(l)} := \boldsymbol{x}^{(l)^T} \hat{\boldsymbol{\beta}}^{(l)}$

3.2 Validation procedures

Validation plays a crucial role in the lazy lasso. On the one hand, a specific point of the regularization path must be selected from each LARS run. On the other hand, a final solution should be selected from the final lazy lasso sequence. Hence, the number of solutions for evaluation can be considerably large. An efficient evaluation method is thus required. In addition, we do not know in advance the proper bandwidth τ for the incoming point $\boldsymbol{x}^{(l)}$. The procedure recommended for finding a specific τ value for $\boldsymbol{x}^{(l)}$ should be data-driven and adaptive.

We first deal with model selection along the LARS regularization path. Since we assume local homoscedasticity and we have used the k-nearest neighborhood function to weight the data set, we can now reasonably assume σ_i to be constant for the weighted data set (that is, within this neighborhood of $\boldsymbol{x}^{(l)}$).

The Mallows' C_p statistic (Mallows 1973), which needs $\sigma_i = \sigma$ for all i, is defined as

$$C_p = \frac{RSS(\hat{\boldsymbol{\beta}})}{\sigma^2} - n + 2v, \tag{13}$$

where $RSS(\hat{\boldsymbol{\beta}})$ is the residual sum of squares and ν is the effective degrees of freedom of the model. Since we are interested only in $\hat{\boldsymbol{\beta}}^{(l)}$, we use the usual C_p naturally adapted for local fitting at point $\boldsymbol{x}^{(l)}$ (Cleveland and Loader 1996):

$$C_p^{(l)} = \frac{RSS^{(l)}\left(\hat{\boldsymbol{\beta}}^{(l)}\right)}{\sigma^{(l)^2}} - tr\left(\boldsymbol{W}^{(l)}\right) + 2\nu, \tag{14}$$

where $tr(\boldsymbol{W}^{(l)}) = \sum_{i=1}^{n} w_i^{(l)}$. A reasonable estimator for the (constant) local noise variance $\sigma^{(l)^2}$ is

$$\hat{\sigma}^{(l)^2} = \frac{RSS_0^{(l)}\left(\hat{\boldsymbol{\beta}}^{(l)}\right)}{tr\left(\boldsymbol{W}^{(l)}\right) - \nu}, \tag{15}$$

where $RSS_0^{(l)}(\hat{\boldsymbol{\beta}}^{(l)})$ corresponds to the $\lambda = 0$ (OLS) fit of the LARS regularization path. Finally, an unbiased estimation $\hat{\nu}$ for the lasso is the number of non-zero predictors in the model (Zou et al. 2007). Note that the k-nearest neighborhood weighting scheme also simplifies the estimation of ν. Therefore, the $C_p^{(l)}$ assessment requires no additional computations, since $\hat{\sigma}^{(l)^2}$, $\hat{\nu}$ and $RSS_0^{(l)}(\hat{\boldsymbol{\beta}}^{(l)})$ are LARS products.

From the above, we can evaluate the solutions that LARS outputs for a given weighted data set. This corresponds to the $best(\cdot)$ function in Algorithm 1. Unfortunately, this procedure does not work for comparing solutions from different LARS runs. This is because we do not have a universal $\sigma^{(l)^2}$ estimation for different weighting schemes.

Leave-one-out cross-validation through a local version of the prediction sum of squares (PRESS) procedure (Allen 1974) is a common and computationally efficient choice for validation in local learning; see (Cleveland and Loader 1996; Loader 1999) for details. It does not need a $\sigma^{(l)^2}$ estimation. The PRESS statistic is defined as

$$PRESS^{(l)} = \frac{1}{tr(\boldsymbol{W}^{(l)})} \sum_{i=1}^{n} \left(\frac{w_i^{(l)}\left(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} \beta_j^{(l)}\right)}{1 - H_{ii}} \right)^2, \tag{16}$$

where $\boldsymbol{H}$ is the hat matrix, such that $\hat{\boldsymbol{v}} = \boldsymbol{H}\boldsymbol{v} = \boldsymbol{H}\boldsymbol{W}^{(l)}\boldsymbol{y}$. Diagonal elements H_{ii} quantify the influence of the observed response on the fitted response for each data item. $\boldsymbol{H}$ has no direct closed form for the lasso but can be derived from a linear approximation to the lasso fit (Tibshirani 1996). Transforming the lasso penalty into a Lagrangian penalty $\sum_j \beta_j^2/|\beta_j|$, $\boldsymbol{H}$ becomes

$$\boldsymbol{H} = \boldsymbol{Z}(\boldsymbol{Z}^T\boldsymbol{Z} + \lambda \boldsymbol{B}^-)^{-1}\boldsymbol{Z}^T, \tag{17}$$

where $\boldsymbol{B}$ is a diagonal matrix such that $B_{jj} = |\hat{\beta}_j|$ and $\boldsymbol{B}^-$ is the $\boldsymbol{B}$ pseudoinverse. Note that, in the evaluation of (14), we could calculate ν as $tr(\boldsymbol{H})$. However, this is

computationally expensive and, as noted by Efron et al. (2004), the accuracy gain is often negligible.

Note that Eq. (16) includes a weighted residual sum of squares for all data items $\boldsymbol{x}^{(i)}$. In principle, it should involve calculating $\hat{\boldsymbol{\beta}}^{(i)}$ for each i. However, the PRESS statistic can make efficient use of $\hat{\boldsymbol{\beta}}^{(l)}$ instead of $\hat{\boldsymbol{\beta}}^{(i)}$ for each data item. In this paper, moreover, we use the k-nearest neighborhood function, so weights are either 1 or 0. Therefore, the numerator in (16) is just the usual residual sum of squares within some neighborhood of $\boldsymbol{x}^{(l)}$. Equation (16) is the $evaluate(\cdot)$ function in Algorithm 1.

4 Experiments

In this section we will describe some experimental results on synthetic and real data sets that illustrate the behaviour of the lazy lasso and the naïve lazy lasso algorithms, compared to the lasso, loess, regression tree (RT) and rodeo.

We perform leave-one-out validation. For each data set (with n instances), we have built n models; for each model, the point of interest is one different data item, whose response is unknown, and the $n - 1$ remaining data items make up the data set itself.

4.1 Synthetic data sets

The algorithms have been tested on several data sets, generated from three different nonlinear controlled models: *m1*, *m2* and *m3*. Model *m1* represents the scenario of a single sparse nonlinear function. The sparse condition is expected to be detrimental for loess, which calculates the distances over all variables including the irrelevant ones. Models *m2* and *m3* are a more complex case, where the function generating the response and the sparsity pattern vary across the data set. Roughly speaking, the response may be obtained either from a single function for the entire data set (*m1*) or from different functions for different locations in the covariate space (*m2* and *m3*).

From each model, we have simulated 50 data sets. All generated data sets have $n = 2{,}000$ samples.

The three models have $p = 100$ covariates, but only some covariates are relevant. Whereas for *m1* the subset of relevant covariates is constant for the whole data set, this subset varies across the data set in *m2* and *m3*. For *m1*, all covariates are sampled from a Gaussian distribution with mean $\mu = 0$ and standard deviations $\sigma = 1$. For *m2* and *m3*, whereas irrelevant covariates are sampled from a Gaussian distribution with mean $\mu = 0$, relevant covariates have been sampled from a Gaussian distribution with non-zero mean (see below). Standard deviation is equal to 1 for all covariates in *m2* and *m3*.

The prediction function for model *m1* is

$$y^{(i)} = x_1^{(i)2} + 5 \sin x_2^{(i)} + x_3^{(i)} x_4^{(i)} + \epsilon_i, \quad i = 1, \ldots, n,$$

where $\epsilon_i \sim \mathcal{N}(0, 0.25)$ for all data data items. Relevant covariates in *m1* are thus X_j, $j \in \{1, 2, 3, 4\}$, and irrelevant covariates are X_j, $j \notin \{1, 2, 3, 4\}$.

Models *m2* and *m3* have four different prediction functions, indexed, say, by $z = 1, 2, 3, 4$. These prediction functions have an equal probability of 1/4 of generating responses. For model *m2* such prediction functions are linear:

$$y^{(i)} = 2x^{(i)}_{3(z-1)+1} + 0.5x^{(i)}_{3(z-1)+2} - x^{(i)}_{3(z-1)+3} + \epsilon_i, \quad z = 1, 2, 3, 4,$$

whereas for model *m3*, they are nonlinear:

$$y^{(i)} = x^{(i)2}_{3(z-1)+1} + 5 \sin x^{(i)}_{3(z-1)+2} + x^{(i)}_{3(z-1)+3} + \epsilon_i, \quad z = 1, 2, 3, 4.$$

Hence, there are three relevant covariates for each data item, and covariates X_j, $j \in \{13, \ldots, 100\}$ are always irrelevant.

As mentioned earlier, for models *m2* and *m3*, relevant covariates are sampled from a Gaussian distribution with non-zero mean μ_j; specifically, when $z = 1$, $\mu_j = -3$ $(j = 1, 2, 3)$; when $z = 2$, $\mu_j = -1$ $(j = 4, 5, 6)$; when $z = 3$, $\mu_j = 1$ $(j = 7, 8, 9)$; and when $z = 4$, $\mu_j = 3$ $(j = 10, 11, 12)$. Regarding the noise term ϵ_i, we set $\sigma_i = 0.2$ for $z = 1, 3$ and $\sigma_i = 0.4$ for $z = 2, 4$.

Firstly, we ran a set of tests using constant bandwidths for all the data items in each data set. We experimented with values ranging from $2p/n$ $(= 0.1)$ to $8p/n$ $(= 0.4)$. Figure 2 summarizes the results over the 50 data sets. Rows correspond, respectively, to models *m1*, *m2* and *m3*. The charts in the left column illustrate the mean error against the bandwidth. The charts in the right column illustrate the mean number of selected variables against the bandwidth. We display the output of lasso and RT as a reference. Rodeo is not considered here because the bandwidth selection is always adaptive.

The proposed iterative algorithm outperforms the naïve approach and the other algorithms in most cases. Excepting *m1*, where RT error is lower than lazy lasso error for bandwidths over 0.28, lazy lasso accuracy is always the best. The improvement over loess is specially remarkable. The difference between lazy lasso and naïve lazy lasso accuracies is also significant. This is more marked for *m3*, which turns out to be the most difficult data set. On the other hand, the number of selected variables is similar for the lazy lasso and the naïve lazy lasso, and much lower than for the lasso. Interestingly, the number of selected covariates for the lazy lasso and the naïve lazy lasso approximates that of RT when the bandwidth moves up from the lowest values. Although not shown in the Fig. 2 for space reasons, the number of correctly selected variables does not vary much for different bandwidths.

Figure 3 shows the boxplot of the error and number of selected variables for an adaptive bandwidth. Table 1 shows the mean number (and standard deviation) of correctly selected variables. Even though lasso and RTs do not need a bandwidth parameter, both have been included for comparison purposes.

Regarding prediction performance, the lazy lasso achieves by far the lowest mean error and the lowest standard deviation in all cases. This can be interpreted as a measure of robustness. Furthermore, the lazy lasso is shown to have a good variable selection ability. It selects a higher number of correct variables than the other methods, excepting rodeo, at the expense, however, of selecting more irrelevant variables than the naïve lazy lasso and RT. Although rodeo selects always all the relevant variables,

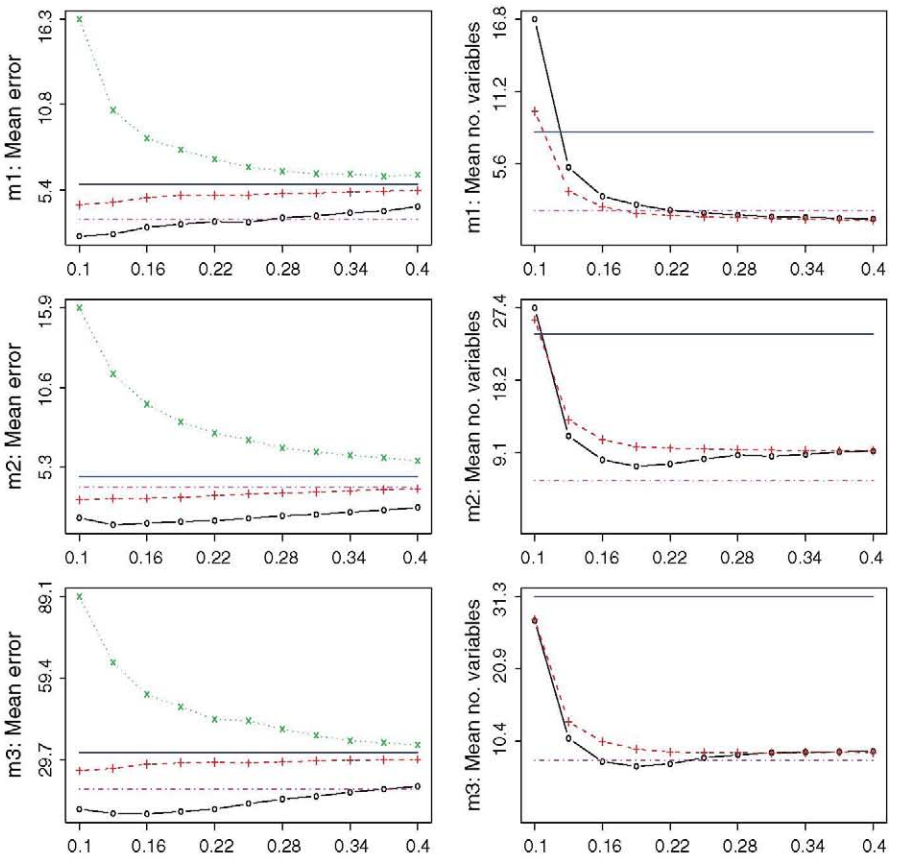

Fig. 2 Evolution of the mean error (*left*) and the mean number of selected variables (*right*) for an increasing bandwidth, for *m1* (*top*), *m2* (*middle*) and *m3* (*bottom*). *Solid lines with open circle* represent the lazy lasso, *dashed lines with plus symbol* represent the naïve lazy lasso, *dotted lines with multi symbol* represent loess, *solid straight lines* represent the lasso and *dashed-dotted straight lines* represent RT. Loess is not in the right-hand plots because it does not select variables

it clearly selects, with lasso, the highest total number of variables. The differences are statistically significant.

From this synthetic setting, we conclude that the devised lazy lasso algorithm can outperform other nonlinear methods like loess, RT, rodeo or the naïve lazy lasso. Given that the lasso is a linear method, its performance is, as expected, worse for the presented scenarios.

4.2 *Pumadyn* data set

Now, we test the algorithm on the *Pumadyn* data set, a realistic simulation of the dynamics of a Puma 560 robot arm. It is available from the Delve Repository.[1]

[1] http://www.cs.toronto.edu/~delve/data/datasets.html.

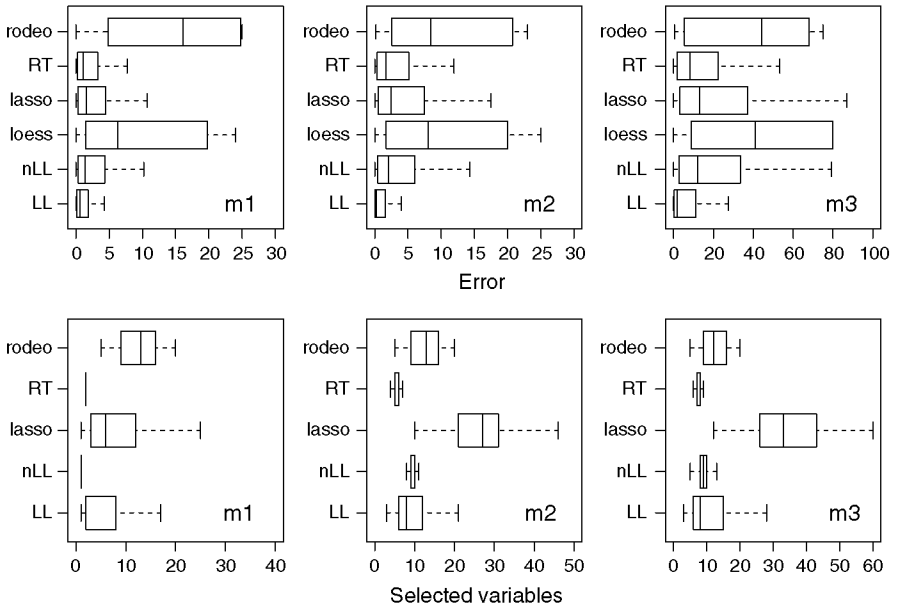

Fig. 3 Boxplots of the error (*top*) and total number of selected variables (*bottom*), for models *m1*, *m2* and *m3*. Tested algorithms are the lazy lasso with adaptive bandwidth (LL), the naïve lazy lasso with adaptive bandwidth (nLL), the lasso, RT and rodeo

Table 1 Mean and standard deviations of the number of correctly selected variables (out of four) for models *m1*, *m2* and *m3*

	LL	Naïve LL	Lasso	RT	Rodeo
m1	3.3 (±0.9)	1.2 (±0.4)	1.5 (±0.7)	1.9 (±0.2)	**4.0** (±0.0)*
m2	**2.8** (±0.4)*	2.0 (±0.3)	2.7 (±0.4)	1.2 (±0.5)	**2.8** (±0.34)*
m3	**2.8** (±0.5)*	2.2 (±0.6)	2.7 (±0.5)	1.6 (±0.6)	**2.8** (±0.34)*

The best result for each row is highlighted in bold. The symbol * is added when the difference to the second best method is statistically significant with a significance level of 0.05

Pumadyn is a family of data sets rather than a single data set. The number of covariates may be eight or 32. The data may be either linear or non-linear. Finally, the amount of noise in the output can be set to moderate or high. All combinations of these three parameters are possible, but we confine our study to the non-linear option. However, we introduce a new parameter: the number of incorporated irrelevant variables, i.e., randomly generated variables not related to the response. Let p_0 be the number of variables of the original data set (eight or 32). We have generated new data sets by adding p_0, $2p_0$ and $3p_0$ irrelevant variables to each original data set. All data sets have $n = 8,192$ data items.

Figure 4 shows the result of the experiments. The bandwidth is selected adaptively for the local algorithms. We ran a leave-one-out validation scheme. We do not show the number of correctly selected variables here because it is not clear which variables

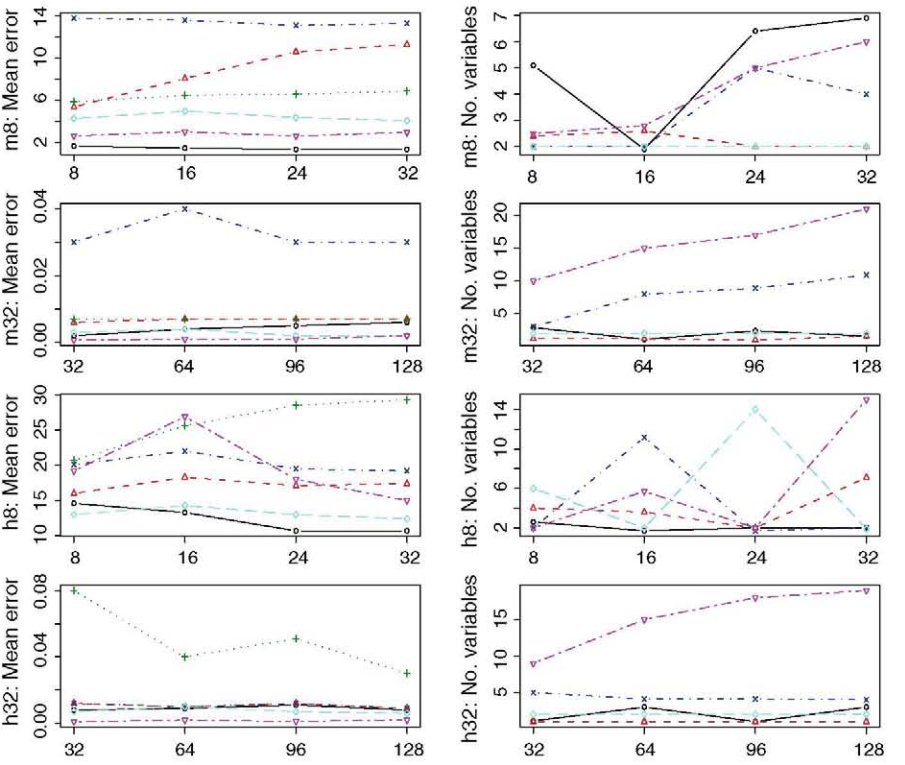

Fig. 4 Evolution of the mean error (*left*) and the mean number of selected variables (*right*) for different amounts of artificially added variables in *Pumadyn* data sets. Each row corresponds to some amount of noise (moderate (m) or high (h)) and some number of variables in the original data set (8 or 32). The *X-axis* represents the total number of variables in the data set, including those artificially added. *Solid lines with open circle* represent the lazy lasso, *dashed lines with triangle* represent the naïve lazy lasso, *dotted lines with plus symbol* represent loess, *dashed-dotted-lines with multi symbol* represent the lasso, *dashed lines with Diamond* represent RT and *dashed-dotted-lines with inverted triangle* lines represent rodeo

from the original set are really relevant. In general, all the algorithms have mostly discarded the added irrelevant variables.

As observed, the proposed method generally produces lower estimation errors than loess, the lasso and the naïve approximation. The results for RT are also very competitive, and rodeo, although selects more variables than the others, performs very well in the 32 variables data sets. Note that loess performances are better than the lasso for the moderate (m) noise data sets, whereas the lasso is better for the high (h) noise case. As expected, the more flexible the model is, the more likely it is to be affected by noise. Interestingly, the lazy lasso outperforms the lasso and loess in both cases.

Regarding the number of selected variables, there is not a dominant method. RT and lazy lasso select a reasonable amount of variables in most data sets. Rodeo often selects more variables than the other approaches. On average, the lasso appears to select more variables than the proposed local methods. Loess does not perform variable selection.

4.3 *Starplus* data set

In this section we report algorithm performance on the *StarPlus* data set,[2] extracted from the neuroscience field. This is functional magnetic resonance imaging (fMRI) data collected at Carnegie Mellon University.

Experiments are conducted on six subjects and forty trials per subject. For each trial, the subject is shown a picture for 4 s and a sentence for 4 s. The objective is to discriminate between these two mental states: "picture" or "sentence" Each data item matches a unique 3-dimensional image. Images are captured every 0.5 s. Hence, each trial has 16 useful images. Briefly, there are six data sets, one per subject, and they all have $n = 40 \times 16 = 640$ data items. On the other hand, each image has a number of voxels, split into 25 localized regions of interest (ROIs). In this paper, instead of considering each individual voxel, we will use the mean activation of voxels at each ROI. Therefore, our data set has $p = 25$ covariates.

The brain's inherent complexity moves us to consider nonlinear models. This is possible thanks to the dimensionality reduction resulting from the use of ROIs instead of individual voxels. In addition, a sparse model is helpful to identifying which brain regions are involved in this task. These factors make the lazy lasso adequate for modeling this kind of data.

Table 2 presents the results. For each data set, we performed n tests, each excluding a different image from the training set. We model the response as either -1 for the "picture" state or 1 for the "sentence" state. This way, classification is based on the sign of the response. For each data item, the error is 0 if it is correctly classified and 1 otherwise. The ratio of incorrectly classified images and the mean and standard deviation of the total number of selected ROIs is reported for each subject. We do not show the number of correctly selected variables because it is not clear which are relevant beforehand. We tested the statistical significance of the best prediction performance and best number of selected ROIs with regard to the second best method. The significance level was set to 0.05.

As observed, lazy lasso accuracy is not on average better than lasso accuracy. The lazy lasso achieves a better accuracy for three subjects, whereas the lasso is better for the other three subjects. Furthermore, these differences are not statistically significant. However, both algorithms are slightly better than loess and RT, while rodeo obtains the best accuracies. The naïve lazy lasso accuracy is definitely poorer. Although not shown in Table 2, the difference of accuracy between the lazy lasso and the naïve lazy lasso is significant in four out of six cases (*04847*, *05675*, *05680* and *05710*).

The big difference between the lazy lasso and loess, lasso, RT and rodeo is the number of selected ROIs. The lazy lasso selects much fewer ROIs than the lasso, RT and rodeo. Loess does not perform variable selection at all. Hence, at the cost of an insignificant loss of accuracy compared to the lasso, the lazy lasso exhibits a finer variable selection ability. The naïve lazy lasso selects the lowest number of ROIs, but its accuracy is poor.

[2] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-81/www/.

Table 2 Ratio of incorrectly classified images (*top*) and the total number of selected variables (*bottom*) for the *StarPlus* data set

Subject	LL	Naïve LL	Loess	Lasso	RT	Rodeo
Error						
04799	0.49	0.5	0.47	**0.43**	0.44	0.44
04820	0.47	0.5	0.45	0.44	0.46	**0.30***
04847	0.35	0.45	0.39	0.4	0.46	**0.31**
05675	0.39	0.46	0.43	**0.38**	0.42	0.41
05680	**0.32**	0.39	0.37	0.35	0.34	0.35
05710	0.39	0.52	0.45	0.4	0.41	**0.35**
Number of selected variables						
04799	4 (±6)	**1.4** (±2.7)*	–	19.8 (±0.7)	18.9 (±1.3)	8.0 (±1.6)
04820	4.9 (±5.7)	**1.4** (±2.3)*	–	9 (±0.3)	18.3 (±1)	17.5 (±0.5)
04847	7.2 (±7.5)	**2.5** (±4)*	–	24.9 (±0.3)	17.5 (±1.3)	9.2 (±0.4)
05675	7.6 (±7.2)	**2.2** (±3.9)*	–	11.4 (±0.8)	19.2 (±0.7)	10.1 (±1.5)
05680	5.3 (±4.4)	**2.9** (±3.1)*	–	22.7 (±0.6)	15 (±1.7)	6.5 (±4.0)
05710	8 (±7.2)	**2.1** (±3.7)*	–	15.4 (±1.5)	19.8 (±1.1)	10.1 (±6.7)

Tested algorithms are the lazy lasso with adaptive bandwidth (LL), the naïve lazy lasso with adaptive bandwidth (Naïve LL), loess with adaptive bandwidth, the lasso, RT and rodeo. The best result for each row is highlighted in bold. The symbol * is added when the difference to the second best method is statistically significant with a significance level of 0.05. Loess has been omitted from the variable selection report because it does not perform variable selection

Summing up, the lazy lasso has also been proven to work well in real environments. It often outperforms the naïve lazy lasso, the lasso, loess and RT, especially in complex scenarios.

5 Discussion

In this paper, we propose an iterative lazy variable selection and shrinkage method that relies on a traditional locally weighted regression paradigm and l_1-regularization. We prove the usefulness of the procedure on three synthetic scenarios, several data sets derived from the *Pumadyn* real data set and the *StarPlus* data set. The lazy lasso is particularly appealing for sparse data sets.

On the regularization side, we provide a method for dealing with nonlinear data. Although LARS can be extended to tackle nonlinear functions, higher-order terms have to be defined in advance. On the local regression side, we provide a variable selection functionality. Local regression is known to be less useful in high-dimensional settings, due to the curse of dimensionality. Bias and variance cannot be kept at low and reasonable levels, respectively, when the number of data items in the local neighborhood is small compared to the number of variables. By reducing the dimension, our approach makes local regression more applicable in these cases.

Our approach is lazy in the sense that there is no overall model valid for all future data items. Hence, as happens with locally weighted regression, we need to run the

whole algorithm each time a new data item is presented. Flexibility for dealing with nonlinearity and better prediction and variable selection performance are the advantages gained in exchange for a more expensive computation when compared with non-lazy techniques. Although this procedure is lazy, if computation time is a primary concern, the analyst can somehow extrapolate the incoming data items to the closest data items in the data set, whose regression coefficients have already been estimated. If these data items are close enough, they are likely to share the same set of relevant variables.

Note also that the nature of the data is an important concern for deciding the adequacy of the proposed methods. Specifically, the methods would excel when the relation between covariates and response is sparse and nonlinear. Some preliminary tests should be run to check the adequacy of the method for a particular data set.

The lazy lasso has potential applications in the context of functional data analysis, where predictors are points on the continuum (Ramsay and Silverman 2005; Ferraty and Vieu 2006; Ferraty and Romain 2010). In this field, the objective is rather a global model that selects a set of highly predictive design points than a local estimation. As shown by Barrientos-Marin et al. (2010), however, the functional data estimation can be considerably benefited from some form of local analysis. To take advantage of this fact, for example, one could obtain a local model with the lazy lasso for each train data instance or for some selected subset. Then, those points (predictors) that have not been selected in any model, or have been selected in few models, could be discarded. Note that an adequate distance function had to be defined for dealing with functional data. To cope with the computational burden, we could apply some early stopping in the lazy lasso process and bound the number of selected variables at each step.

More future work will revolve around the adaptation of the algorithm to multi-response regression, the use of recent variations of the lasso and any improvements on the current algorithm. Robustness is also a major concern. There are robust versions that prevent the harmful effects of outliers for both loess (Cleveland 1979) and the lasso (Khan et al. 2007). Methods that make the proposed algorithm more robust need to be investigated.

Acknowledgments Research partially supported by the Spanish Ministry of Science and Innovation project TIN2010-20900-C04-04, Consolider Ingenio 2010-CSD2007-00018 and Cajal Blue Brain. We thank the anonymous referees and the associated editor for valuable comments about nonparametric variable selection and functional data analysis, which have definitely contributed to the improvement of this paper.

References

Allen DM (1974) The relationship between variable selection and data augmentation and a method for prediction. Technometrics 16:125–127

Barrientos-Marin J, Ferraty F, Vieu P (2010) Locally modelled regression and functional data. J Nonparametr Stat 22:617–632

Breiman L, Friedman J, Ohlsen R, Stone C (1984) Classification and regression trees. Wadsworth, Monterey

Cleveland WS (1979) Robust locally weighted regression and smoothing scatterplots. J Am Stat Assoc 74:829–836

Cleveland WS, Devlin SJ (1988) Locally weighted regression: an approach to regression analysis by local fitting. J Am Stat Assoc 83:596–610

Cleveland WS, Loader C (1996) Smoothing by local regression: principles and methods. In: Statistical theory and computational aspects of smoothing. Physica-Verlag, Heidelberg, pp 10–49
Devlin SJ (1986) Locally-weighted multiple regression: Statistical properties and its use to test for linearity. Tech. rep., Bell Communications Research, Piscataway
Donoho D (2006) For most large underdetermined systems of equations, the minimal l_1-norm solution is the sparsest solution. commun Pure Appl Math 59:797–829
Efron B, Johnstone I, Hastie T, Tibshirani R (2004) Least angle regression. Ann Stat 32:407–499
Fan J (1992) Design-adaptive nonparametric regression. J Am Stat Assoc 87:998–1004
Fan J, Gijbels I (1992) Variable bandwidth and local linear regression smoothers. Ann Stat 20:2008–2036
Ferraty F, Romain Y (2010) The oxford handbook of functional data analysis. Oxford university press, Dordrecht
Ferraty F, Vieu P (2006) Nonparametric functional data analysis: theory and practice. Springer, Berlin
Ferraty F, Hall P, Vieu P (2010) Most-predictive design points for functional data predictors. Biometrika 97:807–824
Foster SD, Verbyla AP, Pitchford WS (2008) A random model approach for the lasso. Comput Stat 23:217–233
Fotheringham AS, Brunsdon C, Charlton M (2002) Geographically weighted regression: the Analysis of spatially varying relationships. Wiley, Chichester
Fowlkes EB (1987) Some diagnostics for binary logistic regression via smoothing. Biometrika 74:503–515
Grosenick L, Greer S, Knutson B (2008) Interpretable classifiers for fmri improve prediction of purchases. IEEE Trans Neural Syst Rehabil Eng 16:539–548
Hastie T, Loader C (1993) Local regression: automatic kernel carpentry. Stat Sci 8:120–143
Hesterberg T, Choi NM, Meier L, Fraley C (2008) Least angle and l_1 penalized regression: a review. Stat Surv 2:61–93
Hoerl A, Kennard R (1970) Ridge regression: biased estimates for nonorthogonal problems. Technometrics 12:55–67
Jones JP, Casetti E (1992) Applications of the expansion method. Routledge, London
Kass RE, Ventura V, Brown EN (2005) Statistical issues in the analysis of neuronal data. J Neurophysiol 94:8–25
Khan JA, Aelst SV, Zamar RH (2007) Robust linear model selection based on least angle regression. J Am Stat Assoc 102:1289–1299
Knight K, Fu W (2000) Asymptotics for lasso-type estimators. Ann Stat 28:1356–1378
Lafferty J, Wasserman L (2008) Rodeo: Sparse, greedy nonparametric regression. Ann Stat 36:28–63
Larrañaga P, Calvo B, Santana R, Bielza C, Galdiano J, Inza I, Lozano J, Armañanzas R, Santafé G, Pérez A, Robles V (2006) Machine learning in bioinformatics. Brief Bioinform 7:86–112
Loader C (1999) Local regression and likelihood. Springer, Berlin
Ma S, Song X, Huang J (2007) Supervised group lasso with applications to microarray data analysis. BMC Bioinformatics 8:1–17
Mallows CL (1973) Some comments on C_p. Technometrics 15:661–675
Meinshausen N, Yu B (2009) Lasso-type recovery of sparse representations for high-dimensional data. Ann Stat 37(1):246–270
Ramsay J, Silverman BW (2005) Functional data analysis. Springer, Berlin
Ruppert D, Wand M (1994) Multivariate locally weighted least squares regression. Ann Stat 22:1346–1370
Seber GAF, Wild CJ (1989) Nonlinear regression. Wiley, New York
Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Series B 58:267–288
Wang H, Xia Y (2009) Shrinkage estimation of the varying coefficient model. J Am Stat Assoc 104:747–757
Weisberg S (1980) Applied Linear Regression. Wiley, New York
Wheeler DC (2009) Simultaneous coefficient penalization and model selection in geographically weighted regression: the geographically weighted lasso. Environ Plan A 41:722–742
Zhao P, Yu B (2006) On model selection consistency of lasso. J Mach Learn Res 7:2541–2567
Zou H (2006) The adaptive lasso and its oracle properties. J Am Stat Assoc 101:1418–1429
Zou H, Hastie T, Tibshirani R (2007) On the degrees of freedom of the lasso. Ann Stat 35:2173–2192