Variable Selection in Local Regression Models via an Iterative LASSO

Diego Vidaurre
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
diego.vidaurre@fi.upm.es

Concha Bielza
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
mcbielza@fi.upm.es

Pedro Larrañaga
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid
pedro.larranaga@fi.upm.es

Abstract

Locally weighted regression is a technique that predicts the response for new cases from their neighbors in the training dataset. In this paper we propose to join modern regularization approaches to locally weighted regression. Specifically, the LASSO method is able to select relevant variables leading to sparse models. We present two algorithms that embed LASSO in an iterative procedure that incrementally discard or add variables, respectively, in such a way that a LASSO-wise regularization path is locally obtained. The algorithms are tested in two different datasets from the UCI repository, obtaining promising results.

1 Introduction

Let $\boldsymbol{\chi} = \{\chi_1, ..., \chi_p\}$ denote the set of covariates and Y the response variable. Linear regression is a widely used tool concerning the influence of $\boldsymbol{\chi}$ over Y. This relationship is modelled by a linear combination of some of the covariates, such that a least squares function is minimized. Let $\mathscr{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), ..., (\boldsymbol{x}^{(n)}, y^{(n)})\}$ be the dataset containing the set of n points in the space of covariates and the response, where $\boldsymbol{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_p^{(i)})$. Let $\boldsymbol{X}$ denote the $n \times p$ matrix whose rows are the p-vectors $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(n)}$ and let $\boldsymbol{y} = (y^{(1)}, y^{(2)}, ..., y^{(n)})$ the vector of responses. Assuming the data is centered, the common linear regression model assumes a relationship such that:

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

where $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)$ are the regression coefficients. Being $\boldsymbol{\Sigma}$ the $p \times p$ covariance matrix, the stochastic unobserved component $\boldsymbol{\epsilon}$ term is distributed:

$$\boldsymbol{\epsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma}). \tag{2}$$

Hence, there are p parameters to be determined, so that the sum of the squares of the distances from the response points to the line drawn by the linear equation is to be minimized. Typical hypothesis to be checked are linearity, normality, independence and variance homogeneity.

Since it bases its method on empirical loss minimization, linear regression may overfit the data. Regularization techniques add a penalization term to the usual regression preventing overfitting, reducing the variance of the estimates and giving rise to more interpretable models. Two widely used methods are *ridge* [1] and the *least absolute shrinkage regression and selection operator* [2]. We are focusing here on the second, commonly referred to as LASSO or l1-regularization. For a general review of LASSO, see [3]. A significant property of the LASSO is its ability to move many regression coefficients to zero, performing a variable selection (sparser models) at the same time than prediction. The LARS [4] algorithm is a variable selection and regression method that outperforms the classical forward stepwise regression algorithm [5], and solves the LASSO with a small modification in a very efficient way.

However, the response variable cannot be always predicted by means of a simple linear function of the covariates, and the results are not optimal from a statistical point of view. In this case some kind of nonlinear analysis may be required. In general, nonlinear regression procedures [6] intend to fit data to any selected equation, finding the values of the parameters that minimize the sum of the squares of the distances from the data points to the curve.

Sometimes, to perform a nonlinear analysis is not straightforward, and it is not possible to establish a unique function for the entire data space. In this case it is more convenient to use some form of local learning. A common method is the *locally weighted regression* (*LOESS*), built on classical least squares regression [7, 8, 9]. For each point in the covariate space, there is a neighborhood containing the point in which the regression surface is well approximated by a function from a parametric class. In this approach, instead of minimizing the residual sum of squares, a weighted sum of squares is minimized. The weights are provided by a function of the distances between the data and the point of interest, giving more importance to closer points. In [7] a second algorithm, called *robust locally weighted regression*, is proposed for providing robustness. In short, after firstly performing the LOESS procedure, the algorithm iteratively calculates new sets of weights basing on the residuals of the estimates $\hat{\mathbf{y}}$ regarding the real response $\mathbf{y}$, in such a way that large residuals correspond to small weights and vice versa. Regression and weights calculation are repeated until some stopping criterion is met. Also in the local fitting arena, in [10] the authors face nonlinearity by using a sum of smooth functions instead of a single parametric model for local learning.

A different form of local analysis is the spatial analysis. The *expansion method* [11] and the *geographically weighted regression* (GWR) [12] are well-known algorithms. Both assume that the influence of the covariates on the response might vary according to the spatial location of the data, typically 2D coordinates where the data are collected. In the expansion method, the regression coefficients at a specific location are the result of a function of the location itself and a set of parameters (constant for all cases) to be learnt from the dataset. In the GWR algorithm, weights are locally assigned to data, so that nearer data are given more importance than further data.

There have been few attempts to combine local learning and variable selection with regularization. Also in the field of spatial analysis, the *geographically weighted LASSO* (GWL) [13] introduces a LASSO-wise penalization on the GWR estimated coefficients. Regardless of spatial analysis, ridge regression (along with principal components regression and partial least squares regression) is applied to local linear prediction of chaotic time series [14]. However, to the best of our knowledge, a LASSO penalization scheme for locally weighted regression has never been proposed.

Our contribution is a method based on LASSO both for local prediction and local variable selection. The setting is a scenario where usual linear regression is not appropriate, and a local approach seems to be convenient. We have developed two algorithms based on LARS for this aim. Unlike GWL, the distances are calculated in the covariate space instead of from separate location coordinates. A naïve approximation could be to add a LASSO penalty to the locally weighted regression. However we are using LASSO because we expect a sparse solution, and the irrelevant covariates should not have been used in the weights estimation (distance calculation). The problem lies in that the distance calculation is previous to the regression, and hence previous to know what variables are irrelevant. To overcome this obstacle, we suggest a couple of iterative algorithms that alternate variable selection with distance computation. One proceeds forwardly from the empty solution where all variables are excluded from the model and starts adding variables, one by one, until all the variables are in the model. The other works in the opposite backwards way, steming from the model with all variables and removing one by one until the empty solution is reached. At each step, distances are recalculated from the current variables in the model, assigning weights to the data for the following regularized regression. LARS is used for selection and removal of variables. As we will explain below, both local algorithms produce a pathway of solutions, from where a unique solution might be selected by means of some selection criterion.

The organization of the paper is as follows: Section 2 describes local regression and the LARS/LASSO algorithms in detail. Section 3 states the novel algorithms, that we are calling *forward local l1 selector*, and *backward local LARS selector*. Section 4 outlines the set of experiments to test the algorithms. Finally, in Section 5 we round the paper off with conclusions and future work.

2 Foundations

2.1 Local regression

The local regression method was originally devised for time series, where one expects that events close in time share common patterns. We focus on the LOESS procedure discussed in [7]. Although locally weighted regression paradigm is not limited to local linear fitting, we will not work in this paper with functions other than linear. In [15] a bunch of mathematical properties of LOESS is discussed. [16] lists some advantages of using local regression.

Assuming the response to be centered, LOESS sets out the following local regression for $\boldsymbol{x}^{(k)}$:

$$\sum_{i=1}^{n} \left\{ \left(y^{(i)} - \sum_{j=1}^{p} x_j^{(i)} \beta_j \right)^2 g\left(\frac{d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(k)})}{\tau} \right) \right\}, \tag{3}$$

where $g(.)$ is a weight function, $d(.)$ is a distance function and τ is the bandwidth constant.

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

$$w_i^{(k)} = \sqrt{g\left(\frac{d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(k)})}{\tau} \right)}, \tag{4}$$

and let $\boldsymbol{W}$ be the diagonal matrix whose elements $W_{ii}^{(k)} = w_i^{(k)}$, the vector of coefficients can be estimated as:

$$\boldsymbol{\beta}^{(k)} = [\mathbf{X}^T \boldsymbol{W}^{(k)T} \boldsymbol{W}^{(k)} \mathbf{X}]^{-1} \mathbf{X}^T \boldsymbol{W}^{(k)T} \boldsymbol{W}^{(k)} \boldsymbol{y}. \tag{5}$$

When a new point $\boldsymbol{x}^{(k)}$ comes up, its response $y^{(k)}$ is predicted by using ad-hoc coefficients $\boldsymbol{\beta}^{(k)}$ locally to the point itself and calculated just at this moment. The distribution of $y^{(k)}$ is unknown. This method is known as *lazy regression* [17]. If the procedure turns out to be too demanding for the abundant affluence of new $\boldsymbol{x}^{(k)}$, or we need a ready-to-use closed model for any reason, a possibility is to run the algorithm for each pair $(\boldsymbol{x}^{(i)}, y^{(i)})$ and use for each new $\boldsymbol{x}^{(k)}$ the set of regression coefficients corresponding to the closer point in $\boldsymbol{X}$, say $\boldsymbol{x}^{(i1)}$. Depending on the distance from $\boldsymbol{x}^{(k)}$ to $\boldsymbol{x}^{(i1)}$, we can also decide either to calculate a new set of regression coefficients or to use $\boldsymbol{\beta}_{i1}$.

There are four relevant aspects when considering LOESS: the parametric family to be locally fitted, the fitting criterion, the weight function and the bandwidth [18].

As said above, we are focusing on the linear parametric family. Assuming $\mathbf{y}$ to be Gaussian with constant variance, least squares is a natural choice for the fitting criterion. If we cannot assure constant variance, some form of regularization may be used along with least squares. On the whole, the parametric family and the fitting criterion depend on the assumptions about the nature of the data and the distribution of the response. As we will detail in Section 3, we are using a penalized least squares favouring parsimony.

Regarding the weight function, any weight function that satisfies the properties listed in [7] may be used. The different choices we are using for the weight function are formulated in Section 3.

Finally, the choice of the bandwidth is a crucial parameter; the nature of the data, its cardinality and dimension, are relevant for a correct selection. On one hand, the bandwidth may be fixed beforehand or selected locally for each $\boldsymbol{x}^{(k)}$. The latter is particulary appropriate for online training [19] and yields some advantages in any case. Typically, it is done by a leave-one-out cross-validation, which can be solved recursively for an increased efficiency [20]. We have tested both fixed and variable bandwidth selection. However, in principle the recursive method of [20] cannot be applied here, as it is thought to solve the least squares problem by the classical estimation method. On the other hand, the *curse of dimensionality* states that as far as the dimension p is bigger, the points quickly become sparse. In this case it is a good idea to increment the bandwidth to compensate this effect.

A relevant issue related to the extent of p is the adecuacy of local regression for high dimensionality. First, the analyst must take into account that local methods are relatively computationally intensive. The expected computation time for a LOESS estimate is the same than for a least squares fit, $O(p^3 + np^2)$, plus the complexity for the weights calculation, $O(np)$. For a single $\boldsymbol{x}^{(k)}$, n distances have to be calculated. It could be demanding, specifically when p is high. Furthermore, in principle, the estimated $\boldsymbol{\beta}^{(k)}$ is only valid for this point. If n is very large, there has been little work done for local regression. In [21] the author presents some validation tests to test the adecuacy of smoothing in binary logistic regression. Although the method and the scenario are slightly different, the conclusions are valid for the LOESS. In short, his analysis shows that the results are still reliable for increments of p if n is large enough, although the inclusion of irrelevant variables has a quite negative effect in the smoothing process. This is just the point we are tackling in this paper.

2.2 LARS/LASSO

During the last years, the original reference for the LASSO algorithm [2] has received over 1600 cites according to Google Scholar by the time this paper is being written. The LASSO estimates are defined as

$$\boldsymbol{\beta}_\alpha = \text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_j^{(i)} \beta_j)^2, \tag{6}$$

subject to

$$\sum_{j=1}^{p} |\beta_j| \leq \alpha. \tag{7}$$

Unlike ordinary least squares and ridge regression, LASSO forces regression coefficients to become zero as we decrease the tunning parameter α. In this way it simultaneously performs variable selection and estimation. The complete solution of the LASSO for all values of α forms the *regularization path*. The regularization path usually starts with a small α and all coefficients equal to zero. One coefficient at a time is made different from zero, although from time to time any variable may also exit the model. For variable selection purposes, we only need to concern about a finite set of α values, specifically those that make the number of zero coefficients to change. Regarding estimation, we still need to pay attention only to this finite set, since the increments on the coefficients between two consecutive values of α are linear. This property makes the regularization path to be entitled as *piecewise constant*.

The LASSO is a quadratic programming problem with a linear inequality constraint. However, the LARS algorithm [4], designed for least angle regression, is able to calculate all possible LASSO estimates for a given problem in $O(p^3 + np^2)$ with a small modification. This is the same cost than for a usual least squares fit. In short, LARS is an iterative algorithm that starts with an empty set of active (non-zero) variables and adds at each step one variable χ_t to this set. This is the one whose correlation with the residuals is the largest. The vector of correlations is:

$$\boldsymbol{c} = \boldsymbol{X}^T(\boldsymbol{y} - \hat{\boldsymbol{y}}). \tag{8}$$

The coefficients of the variables in the active set are increased toward the direction of the least-squares fit based on such variables. So forth, a new variable gets active when its correlation with the residuals equals that of the active set.

Regarding the mathematical properties, there is an amount of theoretical work supporting the LASSO. For instance, in [22] consistency of LASSO is discussed and demonstrated under certain conditions. There have also been some variations of the original LASSO to improve such properties [23].

3 Local LARS/LASSO

In the discussion section, [8] comment the need of incorporating into the LOESS methodology a variable selection procedure when required, i.e, when we know of the presence of irrelevant variables. In this line of argument, we present two algorithms that combine l1-regularization with the usual locally weighted regression paradigm.

As commented above, a first possible approach is equivalent to the GWL algorithm in [13], that is, to directly apply a set of weights to the data set. The weights would be obtained from some transformation of the Euclidean or Mahalanobis distances to the point of interest. Whereas for GWL these distances come from separate coordinates, in locally weighted regression the distances are calculated in the space of covariates. Although simple and easy to implement, irrelevant variables are getting involved in the distance calculation task. Therefore, we state that this method is naive and ineffective, and it is expected to lead to incorrect predictions and incorrect feature selections. This effect will get more marked as the number of irrelevant variables increases. To simplify the terminology, we will call this method as *naive local selector*.

To minimize this risk we propose an iterative algorithm that calculates distances just on the active set of variables at each step. Two versions are presented: a forward algorithm and a backward algorithm.

The forward algorithm, that we will call *forward local l1 selector*, starts with an empty set of variables. It initializes a set of n weights on the distances over all variables. After appropriately weighting the data with them, it runs a LARS algorithm, stopping at the first iteration and keeping the first variable coming up. This variable, say indexed by j, will be the first member of the active set V. Then, weights are recalculated, but using only $V = \{\chi_j\}$, and LARS is run again over weighted data, that will stop when a variable not included in V appears. This variable is then included in V. It recalculates weights again, basing on the active set, and iterates like that until the active set contains all variables or any other stopping criterion is met. Note that at each step of the algorithm, LARS starts from zero variables and makes an arbitrary number of iterations, adding variables before reaching a variable not included in V. However, it is expected that, previous to get a variable not in V, LARS will pass through most of the variables in V at that moment.

Finally, when the algorithm completes all p iterations, some selection criterion is needed. Since the LARS method has run several times starting from zero variables, there are some solutions available with one variable, some solutions with two variables, etc. We will just select the best solution of each variable cardinality by the criterion applied to a separate test database. For example, as the algorithm iterates p times, and run LARS p times, there will be exactly p solutions with one variable. In this work we are using a minimum absolute error l1-penalized on the (weighted) test database. The pseudocode in Algorithm 1 roughly schematizes the procedure.

Algorithm 1 forward local l1 selector

```
Input: training data set $\boldsymbol{X}, \boldsymbol{y}$ with $p$ variables and $n$ cases,
Input: testing data set $\boldsymbol{X}', \boldsymbol{y}'$ with $p$ variables and $n'$ cases,
Input: bandwidth $\tau$ of the neighborhood,
Input: weight function $g(.)$ and distance function $d(.)$,
Input: point $\boldsymbol{x}^{(k)}$, whose response is to be predicted,
Output: set of coefficients $\boldsymbol{\beta}^{(k)}$
Calculate distances $\boldsymbol{d} = (d_1, d_2, ..., d_n)$ to $\boldsymbol{x}^{(k)}$ over all variables
$\boldsymbol{w} := g(\boldsymbol{d}, \tau)$ (vector of weights)
$\boldsymbol{W} := diagonal(\boldsymbol{w})$
$V := \{\}$ (active set)
$t := 0$
repeat
    $\boldsymbol{Xw} := \boldsymbol{W} * \boldsymbol{X}$ (weighted covariates)
    $\boldsymbol{yw} := \boldsymbol{W} * \boldsymbol{y}$ (weighted response)
    $\boldsymbol{paths}(t) := LARS(\boldsymbol{Xw}, \boldsymbol{yw})$ (stopping when a variable $\notin V$ appears)
    $V := V \cup \chi_j \quad | \quad \boldsymbol{path}_j(t) \neq 0$
    Calculate distances $\boldsymbol{d}$ to $\boldsymbol{x}^{(k)}$ using variables in $V$
    $\boldsymbol{w} := g(\boldsymbol{d}, \tau)$
    $\boldsymbol{W} := diagonal(\boldsymbol{w})$
    $t := t + 1$
until $|V| = p$
for $j := 1$ to $p$ do
    $\boldsymbol{\beta}(j) = bestsolution(\boldsymbol{X}', \boldsymbol{y}', \boldsymbol{paths}, j)$,
    the best solution among those with $j$ coefficients different from zero
end for
```

The second algorithm is a backward version of the forward local l1 selector, with some differences that we show straightaway. We call it the *backward local l1 selector*. The algorithm starts with all the p covariates, $V = \{\chi_1, \chi_2, ..., \chi_p\}$, calculates the weights and uses LASSO to discard one variable, say indexed by j. In a second step, it calculates the distances again on $V = V \backslash \chi_j$, and performs another LASSO regression with the new weights, discarding another variable. The algorithm keeps alternating variable selection with weights calculation until some stopping criterion is met, or until there are no variables left. Two possible stopping criteria are the similarity of the predictions and the similarity of the weights across iterations. In this work we are running the algorithm until all variables are run out. Note that a test database is not needed anymore, because LARS is run p times and only the last solution before stopping is kept. That is, there is already only one solution for each variable cardinality. Hence, the entire dataset can be used for the training. We show the pseudocode for the backward local l1 selector in Algorithm 2.

Algorithm 2 backward local l1 selector

```
Input: training data set $\boldsymbol{X}, \boldsymbol{y}$ with $p$ variables and $n$ cases,
Input: bandwidth $\tau$ of the neighborhood,
Input: weighting function $g(.)$ and distance function $d(.)$,
Input: point $\boldsymbol{x}^{(k)}$, whose response is to be predicted,
Output: set of coefficients $\boldsymbol{\beta}^{(k)}$
$V = \{\chi_1, ..., \chi_p\}$ (active set)
Calculate distances $\boldsymbol{d}$ to $\boldsymbol{x}^{(k)}$ using variables in $V$
$\boldsymbol{w} := g(\boldsymbol{d}, \tau)$ (vector of weights)
$\boldsymbol{W} := diagonal(\boldsymbol{w})$
$\boldsymbol{Xw} := \boldsymbol{W} * \boldsymbol{X}$ (weighted covariates)
$\boldsymbol{yw} := \boldsymbol{W} * \boldsymbol{y}$ (weighted response)
$\boldsymbol{paths}(0) := LARS(\boldsymbol{Xw}, \boldsymbol{yw})$, taking the last solution, with all $\boldsymbol{\beta} \neq 0$
$t := 1$
repeat
  Calculate distances $\boldsymbol{d}$ to $\boldsymbol{x}^{(k)}$ using variables in $V$
  $\boldsymbol{w} := g(\boldsymbol{d}, \tau)$
  $\boldsymbol{W} := diagonal(\boldsymbol{w})$
  $\boldsymbol{Xw} := \boldsymbol{W} * \boldsymbol{Xv}$ (let $\boldsymbol{Xv} = \boldsymbol{X}$ but including only variables in $V$)
  $\boldsymbol{yw} := \boldsymbol{W} * \boldsymbol{y}$
  $\boldsymbol{paths}(t) := LARS(\boldsymbol{Xw}, \boldsymbol{yw})$, taking the last but one, with one $\beta_j = 0$
  $V = V \setminus \chi_j \quad | \quad \boldsymbol{paths}(t)_j = 0$
  $t := t + 1$
until $|V| = 0$
```

For both algorithms, we have used Euclidean distances, and for weighting we have employed the well-known tricube function used in [8]. Let $\boldsymbol{x}^{(k)}$ be the point that the local procedure concerns and $\boldsymbol{x}^{(i)}$ any other point. The tricube function establishes

$$w_i = \begin{cases} \left(1 - \left(\frac{d(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(k)})}{d(\boldsymbol{x}^{(q)}, \boldsymbol{x}^{(k)})}\right)^3\right)^3 & \text{if } \boldsymbol{x}^{(i)} \in C_\tau \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

where $d(.)$ is a distance function, τ is the bandwidth, C_τ contains the τn closer points to $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(q)}$ is the furthest point to $\boldsymbol{x}^{(k)}$ in C_τ.

Regarding the computation complexity of the algorithms, in principle we are running p times a LARS algorithm with complexity $O(p^3 + np^2)$, which would yield a complexity $O(p^4 + np^3)$ plus the distances calculation ($O(np)$). However, it is remarkable that for the backward version, as long as the algorithms iterates, LARS has to deal with fewer variables. Therefore the complexity is

$$\sum_{j=1}^{p} O(j^3 + nj^2) = \sum_{j=1}^{p} O(j^3) + \sum_{j=1}^{p} O(nj^2) = O\left(\left(\frac{j(j+1)}{2}\right)^2 + n\frac{j(j+1)(2j+1)}{6}\right). \tag{10}$$

For the forward version, LARS stops earlier, specially in the first iterations. The worst complexity is the same than for the backward algorithm, although the mean complexity is smaller (at iteration t, LARS will loop at most t times, but sometimes it stops before). Naive algorithm has the same complexity than LARS.

4 Experiments

In this section we face the algorithms with two real databases: *Housing* and *Forest Fires*. Both can be found in the UCI Repository [1]. Housing dataset deals with prices of housing in the suburbs of Boston. Besides the response variable (the price) it has 14 variables (integer and real), and 506 instances. Forest fires dataset, thoroughly described in [24], has 13 real attributes and 517 instances. It concerns the occurence of forest fires in the Montensinho natural park, Portugal. A logarithm function $ln(y_i+1)$ has been applied on the response. Since among the variables we have the fires location coordinates, it is very suitable the use of some way of spatial analysis. However, we will abstain from including such analysis as it is not the concern of this work. Thus, we put the coordinates values into the independent variables set. The comparisons have been done with LASSO, usual LOESS and *Regression trees* (DT) [25].

Firstly, to compare local approaches, we have run a set of tests using constant bandwidths, experimenting with 12 values between 0.15 and 0.8. To choose the best solution of the pathway for the proposed algorithms, we have cross-validated with 1/4 of each dataset. For space reasons, we only show the results for the Housing dataset (see Figure 1). Tables 1 and 2 show also some useful statistics for Housing database. An equivalent table has not been shown for Forest fires database because the many small values of the responses (small prediction errors) are dominated by the few big values (big prediction errors). We have taken one by one all points in the datasets, we have predicted their responses and compared to the true response. For each bandwidth and each algorithm, we show mean prediction errors and the mean numbers of variables. The best solution of each pathway has been selected with a l1-penalized score, basing on a separate test proportion of the database. Figure 1 shows that for small bandwidths the performance of the proposed algorithms is better than LOESS. However, LASSO turns out to be more accurate excepting for big bandwidths of LOESS. It reveals that the dataset can be up to a point linearly approximated. The good news is that the local approach needs some less variables to make the prediction.

Besides a competitive performance, the proposed algorithms seem to behave more robustly. It can be observed in the worst case (maximum error) and standard deviation for LOESS, LASSO and Regression Tree, significantly bigger than for the other algorithms.

As mentioned above, in local regression it is common to use an ad-hoc bandwidth for each point to be predicted. To analyze this method, in Figure 2 split prediction errors and number of variables in 15 intervals and plot histograms for each algorithm, for the Forest fires dataset. Again, we have used cross-validation with 1/4 of the dataset. We have cut off errors above 3.0. This is done to make

[1]`http://archive.ics.uci.edu/ml/`

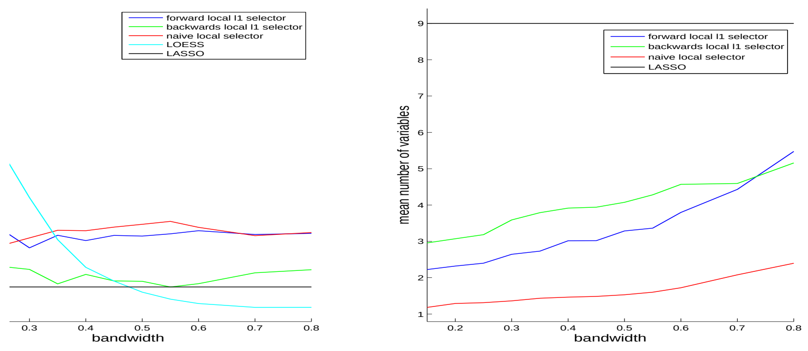

Figure 1: Evolution of the mean error and the mean number of variables for an increasing bandwidth, for Housing dataset. For LOESS, the number of variables is always the maximum, so it has been omitted.

	Ff	Bf	Nf	LRf	Fv	Bv	Nv	LRv	L	RT
mean	3.8	3.6	3.9	4.3	4.3	4.2	4.0	3.4	3.5	2.9
median	2.5	2.4	2.7	2.0	2.8	2.7	2.5	1.9	2.3	1.9
std dv.	4.4	3.9	4.1	7.3	4.9	4.7	4.4	5.7	3.8	3.6
max.	27.9	23.8	26.8	93.3	27.4	27.2	27.5	61.5	30.8	30.7

Table 1: Some statistics for estimation error, for Housing dataset, and algorithms: forward local l1 selector and fixed bandwidth (Ff), backward local l1 selector and fixed bandwidth (Bf), naive local l1 selector and fixed bandwidth (Nf), LOESS and fixed bandwidth (LRf), forward local l1 selector and adaptive bandwidth (Fv), backward local l1 selector and adaptive bandwidth (Bv), naive local l1 selector and adaptive bandwidth (Nv), LOESS and adaptive bandwidth (LRv), classical LASSO (L) and Regression tree (RT). For fixed bandwidth cases, a value of 0.3 has been taken.

	Ff	Bf	Nf	Fv	Bv	Nv	L
mean	2.6	3.6	1.4	2.1	2.8	1.2	9.0
median	2.0	3.0	1.0	2.0	3.0	1.0	9.0
std dv.	1.8	1.9	1.0	1.4	1.5	0.6	0.0
max.	10.0	12.0	7.0	8.0	7.0	6.0	9.0

Table 2: Same setting than for Table 1, for average number of variables. Local Regression algorithms and Regression tree have been removed.

intervals more specific and informative, ignoring outliers. This is fair play anyway with regard to our comparisons, since most big errors were obtained with LASSO and Regression trees. As observed, results are slightly better for the proposed algorithms than for LOESS and LASSO. Specifically, for LASSO most errors concentrate around 1, whereas for the other algorithms many error are smaller. It is remarkable the good behaviour of the Regression tree in this case, although it produces more big errors than the proposed algorithms. Note that the best solution for the naive version and LASSO approach often yields all regression coefficients equal zero, which obviously generalizes poorly. This is so because there are many zero responses in this dataset. However, the forward and backward versions usually recover at least two variables.

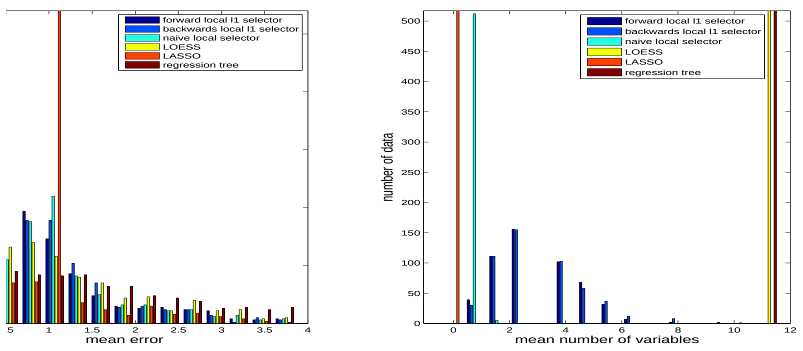

Figure 2: Histogram of 15 intervals of error and number of variables, for Forest fires dataset.

[24] emphasizes the importance of the prediction of small fires, which are the great majority. In Figure 5 we show a scatter plot of the response against the error. We exclude backward and naive approaches because of its similarity with the forward approach. All local algorithms were run with an adaptable bandwidth. The proposed algorithms are the ones which best predicts small fires, although all the methods have certain difficulties with responses equal to zero.

5 Discussion

In this work, we propose two variable selection and shrinkage iterative methods that lean on traditional locally weighted regression paradigm and l1-regularization. We prove its usefulness in two real datasets from the UCI repository, but we feel that better results are possible. The nature of the data is important to decide the adecuacy of the proposed methods. Specifically, the methods would stand out when the relation between covariates and response is sparse and nonlinear.

From the regularization side, we are providing an alternative for dealing with nonlinear data. From the local regression side, we supply the variable selection functionality. Moreover, using regularization techniques we can overcome the

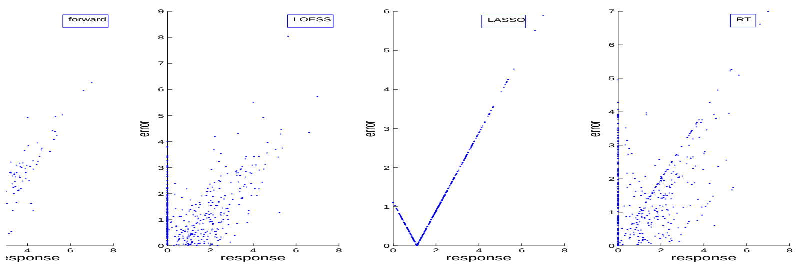

Figure 3: Scatter plots for Forest fires dataset, for forward local l1 selector, LOESS, LASSO and Regression tree, respectively. Responses are not centered

$p >> n$ case, that is, when the data matrix is not invertible As a derivation of least squares regression, locally weighted regression needs the cardinality of the dataset to be greater or equal than the number of variables.

Our approach is lazy in the sense that we lack an overall model valid for all future cases. Hence, as happens with locally weighted regression, we need to run the whole algorithm each time a new case is presented. Flexibility against nonlinearity and best performance of prediction are the advantages gained in exchange for a more expensive computation if compared with linear techniques. Although the way we are proceeding here is lazy, if computation time is a main concern, the analysis can draw the regression coefficients for some or all the cases in the training dataset, and extrapolate the new case to the closest points in the dataset in some way (for example giving a weighted mean of the "closest" responses).

Future work will revolve around the adaptation of the algorithms to multiresponse problems, applications to challeging data, use of recent variations of LASSO, and improvements over the algorithms. Specifically, we expect to develop a more sophisticated method for the selection phase of the forward algorithm. We consider this algorithm the most sensible and promising, but it needs to set aside a piece of the dataset to sieve solutions. This is a disadvantage against backward and naive algorithms that claims to be solved. Robustness is also an important concern. There are robust versions that prevent the bad effects of outliers both for LOESS [7] and for LASSO [26]. Methods that make the proposed algorithms more robust need to be investigated.

References

[1] A. Hoerl and R. Kennard, "Ridge regression: Biased estimates for nonorthogonal problems," *Technometrics*, vol. 12, pp. 55–67, 1970.

[2] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society. Series B*, vol. 58, pp. 267–288, 1996.

[3] T. Hesterberg, N. M. Choi, L. Meier, and C. Fraley, "Least angle and l_1 penalized regression: A review," *Statistics Surveys*, vol. 2, pp. 61–93, 2008.

[4] B. Efron, I. Johnstone, T. Hastie, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32(2), pp. 407–499, 2004.

[5] S. Weisberg, *Applied Linear Regression.* Wiley, New York, 1980.

[6] G. Seber and C. Wild, *Nonlinear Regression.* Wiley, New York, 1989.

[7] W. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74(368), pp. 829–836, 1979.

[8] W. Cleveland and S. Devlin, "Locally weighted regression: An approach to regression analysis by local fitting," *Journal of the American Statistical Association*, vol. 83(403), pp. 596–610, 1988.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Predictions.* Springer Verlag, New York, 2001.

[10] T. Hastie and R. Tibshirani, "Generalized additive models," *Statistical Science*, vol. 1(3), pp. 297–310, 1986.

[11] J. Jones and E. Casetti, *Applications of the Expansion Method.* Routledge, 1992.

[12] A. Fotheringham, C. Brunsdon, and M. Charlton, *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships.* Wiley, 2002.

[13] D. C. Wheeler, "Simultaneous coefficient penalization and model selection in geographically weighted regression: The geographically weighted Lasso," *Environment and Planning A*, vol. 41, pp. 722–742, 2009.

[14] D. Kugiumtzis, O. C. Lingjaerde, and N. Christophersen, "Regularized local linear prediction of chaotic time series," *Physica D: Nonlinear Phenomena*, vol. 112, pp. 344–360, 1998.

[15] S. Devlin, "Locally-weighted multiple regression: Statistical properties and its use to test for linearity," Bell Communications Research, Piscataway, NJ, Tech. Rep., 1986.

[16] T. Hastie and C. Loader, "Local regression: Automatic kernel carpentry," *Statistical Science*, vol. 8(2), pp. 120–143, 1993.

[17] D. Aha, "Special issue on lazy learning," *Artificial Intelligence Review*, vol. 11(1-5), pp. 1–6, 1997.

[18] W. Cleveland and C. Loader, "Smoothing by local regression: Principles and methods," *Statistical Theory and Computational Aspects of Smoothing*, pp. 10–49, 1996.

[19] G. Bontempi, M. Birattari, and H. Bersini, "Lazy learning for local modelling and control design," *International Journal of Control*, vol. 72(7), pp. 643–658, 1999.

[20] M. Birattari, G. Bontempi, and H. Bersini, "Lazy learning meets the recursive least squares algorithm," in *Advances in Neural Information Processing Systems 11.* MIT Press, 1999, pp. 375–381.

[21] E. Fowlkes, "Some diagnostics for binary logistic regression via smoothing," *Biometrika*, vol. 74(3), pp. 503–515, 1987.

[22] P. Zhao and B. Yu, "On model selection consistency of Lasso," *Machine Learning Research*, vol. 7, pp. 2541–2567, 2006.

[23] H. Zou, "The adaptive Lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101(12), pp. 1418–1429, 2006.

[24] P. Cortez and A. Morais, "A data mining approach to predict forest fires using meteorological data," in *Portuguese Conference on Artificial Intelligence*, 2007, pp. 512–526.

[25] L. Breiman, J. Friedman, R. Ohlsen, and C. Stone, *Classification and Regression Trees.* Wadsworth, Monterey, CA, 1984.

[26] J. Khan, S. V. Aelst, and R. Zamar, "Robust linear model selection based on least angle regression," *Journal of the American Statistical Association*, vol. 102(480), pp. 1289–1299, 2007.

Title: Proceedings of WUPES 2009

Publisher: University of Economics, Prague

Editors: Tomáš Kroupa, Jiřina Vejnarová

Cover design: Jiří Přibil

ISBN 978-80-245-1543-4