Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Incremental Regression Function Construction with Small Landmarks [⋆]

Gang Wang [*] **Shiyin Qin** [**] **and Pipei Huang** [**]

[*] *Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. (e-mail: wanggang@cse.ust.hk).*
[**] *School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. (e-mail: qsy@buaa.edu.cn, huangpipei@gmail.com)*

Abstract: In automatic control and its related applications, many problems can be formulated as the regression estimation problem. In this paper, we construct a nonlinear regression model by using kernels as basis functions in a dictionary and applying the L_1 norm as the regularizer. The regression function obtained from this model possesses the sparseness property where only a subset of points are used to represent the function. We call this subset of points as landmarks. It is a convex optimization problem. However, instead of using the standard optimization tools to solve a convex problem for a particular regularization value, we develop an efficient regularization path algorithm that can trace all solutions for all possible regularization parameter values. It overcomes the computational difficulty in model selection. Since the algorithm generally adds basis functions incrementally to improve the prediction accuracy, the regression function can be represented concisely with small landmarks.

1. INTRODUCTION

Regression analysis is one of the most fundamental techniques in many areas such as pattern recognition, automatic control, statistics, etc. In a typical regression problem, we are given a training set of n independent and identically distributed (i.i.d.) data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$, where $\mathbf{x}_i$ and y_i are the input and output, respectively, of the ith data point. The goal is to learn a function $f(\mathbf{x})$ that can predict as accurately as possible the output of the data points sampled according to the underlying data distribution.

Many regression models have been studied in the past, including ridge regression, neural networks, support vector regression (SVR) [Vapnik, 1995], boosting, etc. Among these methods, SVR is probably the most popular one in recent years due to its strong theoretical foundations as well as excellent empirical successes demonstrated. It encourages the sparse solutions, thus, offers both representational and computational advantages. The SVR model contains two hyperparameters, which are the error parameter and the regularization parameter. Typically the values of these two hyperparameters have to be chosen in advance by the users. However, there do not exist proper choices that are good under all circumstances. In practice, users usually adopt some default values for the hyperparameters even though they are by no means optimal choices. Extensive exploration of the optimal parameter values is seldom pursued since this requires re-training the model many times under different hyperparameter settings.

LASSO is another regression model proposed by [Tibshirani, 1996], where the acronym LASSO stands for *least absolute shrinkage and selection operator*. LASSO regularizes ordinary least square regression with an L_1 regularizer. Compared with SVR, LASSO not only possesses the sparseness property but also has only one hyperparameter to set. Having only one hyperparameter in the model makes parameter tuning much easier to achieve than if there exist multiple hyperparameters. In this paper, we apply the dictionary method to construct a nonlinear representation of the regression function by using kernels as the basis functions with the L_1 norm as the regularization term. Thus, this model can be considered as the nonlinear LASSO regression model. Instead of applying convex optimization techniques to obtain one solution for a single regularization value, we propose a simple and efficient algorithm to sequentially calculate all solutions corresponding to all values of the regularization parameter. Our algorithm belongs to a class of the so-called regularization path algorithms. Given a regularization parameter value and the optimal solution obtained for that value, the regularization path algorithm attempts to compute the next solution as the regularization parameter changes its value. As a result, the user does not have to choose multiple regularization parameter values in advance and to solve the optimization problem multiple times. Our algorithm traces the regularization path as a function of the regularization parameter, and the computational requirement is much lower than that of the traditional approach which requires training the model multiple times.

Rosset and Zhu [2003] firstly investigated the regularization paths and found that any model with L_1 regularization and a quadratic, piecewise quadratic, piecewise linear, or linear loss function has a piecewise linear regularization

⋆ This work was supported by hi-tech research and development (863) program of China (Grant No. 2006AA04Z207) and research fund for doctorial program of higher education of China (Grant No. 20060006018).

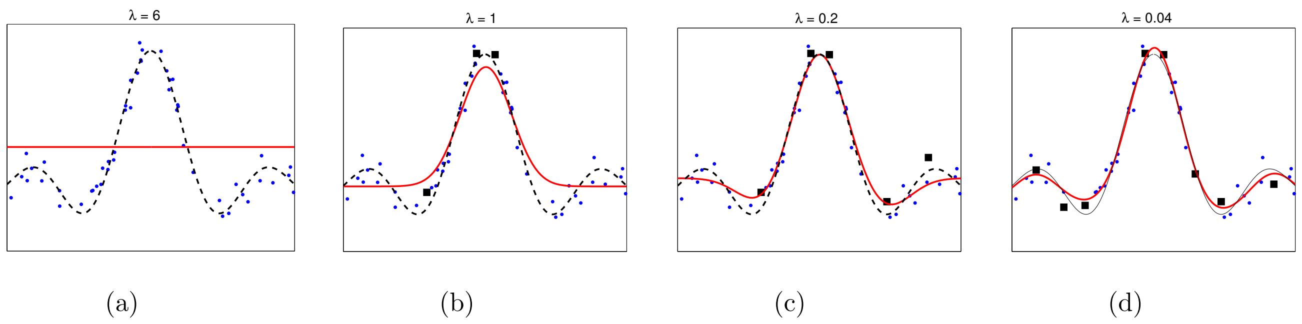

Fig. 1. Nonlinear LASSO regression results for different regularization values. The x-axis is the 1-dimensional input and y-axis is the output. There are 50 data points $\{(x_i, y_i)\}$ with $y_i = sinc(x_i) + e_i$, where e_i is a Gaussian noise with zero mean and a variance 0.1. The black dashed curve is the sinc function and the red curve is the regression function. The landmarks are shown as small squares. The Gaussian RBF kernel $K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\gamma)$ with $\gamma = 0.5$ is used.

path and hence the entire path can be computed efficiently. Zhu et al. [2003] proposed an algorithm to compute the entire regularization path for the L_1-norm support vector classification (SVC) and Hastie et al. [2004] proposed one for the standard L_2-norm SVC. They are again based on the property that the paths are piecewise linear. Rosset [2004] proposed a general path following algorithm to approximate the nonlinear regularization path. Besides the regularization parameter, Wang et al. [2006] showed that this approach can also be applied to explore the solution paths for some other model hyperparameters.

The regression function in our algorithm is formulated as a linear combination of basis functions, where each basis function is related to one input point. Since L_1 norm is used, the regression function is represented by a subset of points, which are called landmarks. Our algorithm finds landmarks incrementally to construct the regression function. Some results during the algorithm is shown in Figure 1. The algorithm begins by setting the regularization parameter to infinity, leading to an extremely simple initialization step without landmarks. (It corresponds to the case in Figure 1(a).) As the regularization parameter decreases in value, the number of landmarks will gradually increase and the regression function becomes to fit the data better. (The landmark number in Figure 1(b)-(d) is 3, 5 and 8, respectively.) We observe in Figure 1(c) that when the regression function is fitted well, only a small number of landmarks are used and it gives a concise representation of the regression function. Other points can be discarded since they will never be used for predicting new point. By estimating the generalization errors of the regression function during the algorithm, we can determine the optimal choice of the regularization parameter without having to train the model multiple times. Moreover, we may control the number of landmarks used to represent the function exactly by stopping to add more landmarks as soon as a certain level of accuracy is reached.

2. PROBLEM FORMULATION

The dictionary method is a general model building approach which constructs the regression function as a linear expansion of basis functions, such as the following:

$$f(\mathbf{x}) = \sum_{i=1}^{q} \beta_i h_i(\mathbf{x}) + \beta_0, \tag{1}$$

where $\mathcal{D} = \{h_1(\mathbf{x}), \ldots, h_q(\mathbf{x})\}$ is a dictionary of basis functions and $\boldsymbol{\beta} = (\beta_i)_{i=1}^q$ and β_0 are the coefficients of the regression function. To extend LASSO for nonlinear regression, it is natural to consider using kernels for the basis functions. Kernel methods [Schölkopf and Smola, 2002, Müller et al., 2001] have demonstrated great successes in solving many machine learning and pattern recognition problems. These methods implicitly map data points from the input space to some feature space where typically a linear method is applied. The implicit feature mapping is determined by a kernel function, which allows the inner product between two points in the feature space to be computed without having to know the explicit mapping from the input space to the feature space. Rather, it is simply a function of two points in the input space. Only functions that satisfy Mercer's condition are eligible candidates for such kernel functions. Some basic kernel functions and kernel construction rules have been proposed. The most commonly used kernel functions are probably the radial basis function (RBF) kernel or the polynomial kernel.

Through applying the kernels for the basis functions, the regression function $f(\mathbf{x})$ admits a representation of the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i K(\mathbf{x}_i, \mathbf{x}) + \beta_0, \tag{2}$$

where $K(\cdot,\cdot)$ is a kernel function defined on $\mathbb{R}^d \times \mathbb{R}^d$. Thus, each basis function $K(\mathbf{x}_i, \mathbf{x})$ is only related to one input point. Let $\mathbf{y} = (y_i)_{i=1}^n$ denote the output vector and $\mathbf{K} = [K(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^n$ the kernel matrix. The regularized empirical loss L that nonlinear LASSO aims to minimize is given by

$$L(\boldsymbol{\beta}, \beta_0) = \frac{1}{2}\|\mathbf{y} - \mathbf{K}\boldsymbol{\beta} - \beta_0\mathbf{1}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{3}$$

where $\mathbf{1}$ is a vector of ones and $\|\cdot\|_2$ denotes the L_2 norm of a vector. We refer to this nonlinear LASSO formulation as kernelized LASSO (kLASSO). Note that $f(\mathbf{x})$ is expressed as an expansion in terms of only a subset of the data points for which β_i is nonzero. Since L_1 norm is used, $f(\mathbf{x})$ tends to give a sparse representation. Apparently, the minimization problem in (3) is a convex optimization problem and hence it could be solved using QP optimization techniques. However, for reasons that will be made clear later, we choose to solve this problem via a completely different path in the next subsection.

Since β_0 only appears in the empirical loss term of L in equation (3) and the term is defined based on L_2 norm, L is differentiable with respect to β_0. By setting $\partial L/\partial\beta_0$ to zero, we obtain

$$\beta_0 = (\mathbf{y} - \mathbf{K}\boldsymbol{\beta})^T\mathbf{1}/n. \tag{4}$$

Thus, when the optimal value of $\boldsymbol{\beta}$ is available, β_0 can be calculated directly from $\boldsymbol{\beta}$. However, since L_1 norm is used in the regularization term, L is not differentiable with respect to $\boldsymbol{\beta}$ at the point $\boldsymbol{\beta} = 0$. This calls for a special method to handle the problem. Following Rosset and Zhu [2003], we represent $\boldsymbol{\beta} = \boldsymbol{\beta}^+ - \boldsymbol{\beta}^-$, where $\boldsymbol{\beta}^+ = (\beta_i^+)_{i=1}^n$ and $\boldsymbol{\beta}^- = (\beta_i^-)_{i=1}^n$, with the constraints $\beta_i^+, \beta_i^- \geq 0$ for $i = 1, \ldots, n$. The optimization problem based on minimizing L as defined in equation (3) can thus be rewritten as:

$$\min_{\boldsymbol{\beta}^+,\boldsymbol{\beta}^-} \frac{1}{2}\left\|\mathbf{y} - \mathbf{K}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) - \beta_0\mathbf{1}\right\|_2^2 + \lambda\sum_{i=1}^n(\beta_i^+ + \beta_i^-)$$

subject to $\beta_i^+, \beta_i^- \geq 0 \quad i = 1, \ldots, n$.

Introducing Lagrange multipliers $\boldsymbol{\mu}^+ = (\mu_i^+)$, $\boldsymbol{\mu}^- = (\mu_i^-)$, $i = 1, \ldots, n$ for the inequality constraints above, the corresponding Lagrangian dual function L_d is expressed as:

$$\begin{aligned} L_d(\boldsymbol{\beta}^+, \boldsymbol{\beta}^-, \boldsymbol{\mu}^+, \boldsymbol{\mu}^-) = &\frac{1}{2}\left\|\mathbf{y} - \mathbf{K}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) - \beta_0\mathbf{1}\right\|_2^2 \\ &+\lambda\sum_{i=1}^n(\beta_i^+ + \beta_i^-) \\ &-\sum_{i=1}^n \mu_i^+\beta_i^+ - \sum_{i=1}^n \mu_i^-\beta_i^-. \end{aligned} \tag{5}$$

Note that L_d is differentiable with respect to both $\boldsymbol{\beta}^+$ and $\boldsymbol{\beta}^-$. Thus we can state the optimality conditions as:

$$\frac{\partial L_d}{\partial \beta_i^+} = \big(-\mathbf{K}(\mathbf{y} - \mathbf{K}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) - \beta_0\mathbf{1})\big)_i + \lambda - \mu_i^+ = 0, \tag{6}$$

$$\frac{\partial L_d}{\partial \beta_i^-} = \big(\mathbf{K}(\mathbf{y} - \mathbf{K}(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) - \beta_0\mathbf{1})\big)_i + \lambda - \mu_i^- = 0. \tag{7}$$

From the Karush-Kuhn-Tucker (KKT) conditions, we also have:

$$\mu_i^+ \geq 0, \ \ \mu_i^- \geq 0, \tag{8}$$

$$\mu_i^+\beta_i^+ = 0, \mu_i^-\beta_i^- = 0. \tag{9}$$

Based on the conditions (6)–(9) above, we know that, for any fixed regularization value, the optimal solution $\hat{\boldsymbol{\beta}}$ $(= \hat{\boldsymbol{\beta}}^+ - \hat{\boldsymbol{\beta}}^-)$ should have the following properties:

- If $\lambda > 0$:
  - * $\hat{\beta}_i^+ > 0 \Rightarrow$ $\Big(\mathbf{K}(\mathbf{y} - \mathbf{K}(\hat{\boldsymbol{\beta}}^+ - \hat{\boldsymbol{\beta}}^-) - \hat{\beta}_0\mathbf{1})\Big)_i = \lambda, \ \beta_i^- = 0$
  - * $\hat{\beta}_i^- > 0 \Rightarrow$ $\Big(\mathbf{K}(\mathbf{y} - \mathbf{K}(\hat{\boldsymbol{\beta}}^+ - \hat{\boldsymbol{\beta}}^-) - \hat{\beta}_0\mathbf{1})\Big)_i = -\lambda, \hat{\beta}_i^+ = 0$
  - * $\hat{\beta}_i^+ = 0, \ \hat{\beta}_i^- = 0 \Rightarrow$ $-\lambda \leq \Big(\mathbf{K}(\mathbf{y} - \mathbf{K}(\hat{\boldsymbol{\beta}}^+ - \hat{\boldsymbol{\beta}}^-) - \hat{\beta}_0\mathbf{1})\Big)_i \leq \lambda$
- If $\lambda = 0$:
  - * $\mathbf{K}(\mathbf{y} - \mathbf{K}\hat{\boldsymbol{\beta}} - \hat{\beta}_0\mathbf{1}) = \mathbf{0}$ (non-regularized solution)

For simplicity, we remove the ˆ notation for the optimal solution in the rest of this paper. Let $\mathbf{g} = (g_i)_{i=1}^n$ be an n-dimensional vector such that $\mathbf{g} = \mathbf{K}(\mathbf{y} - \mathbf{K}\boldsymbol{\beta} - \beta_0\mathbf{1})$. The above properties indicate that, for those nonzero coefficients β_i at the optimal solution, their corresponding elements g_i are equal to λ or $-\lambda$ whose sign is determined by the sign of the coefficient. Thus, we have a set of landmarks, denoted as $\mathcal{A} = \{i \in \{1, ..., n\} : \beta_i \neq 0\}$, such that

$$i \in \mathcal{A} \Rightarrow g_i = \mathrm{sgn}(\beta_i)\,\lambda, \tag{10}$$

$$i \notin \mathcal{A} \Rightarrow |g_i| < \lambda. \tag{11}$$

Thus, based on the points in $\mathcal{A}$, we can build a linear system with which the kLASSO regularization path algorithm can be devised.

3. REGULARIZATION PATH ALGORITHM

Initialization of the regularization path in kLASSO is quite straightforward. When λ is set to infinity initially, the optimal solution corresponds to $\boldsymbol{\beta} = \mathbf{0}$ and $\beta_0 = \mathbf{y}^T\mathbf{1}/n$. Thus $\mathbf{g} = \mathbf{K}(\mathbf{y} - \beta_0\mathbf{1})$. At this moment, there is no landmarks and hence the condition (11) holds for all elements in $\mathbf{g}$. In other words, we have $|g_i| < \lambda \ \forall i$. Thus the regression function is β_0 for any input and hence it is flat.

As λ decreases, $\boldsymbol{\beta}$ will remain zero until λ reaches a certain value when one data point has its $|g_i|$ value equal to λ. This data point which is the first to be added to $\mathcal{A}$ is [1]

$$j = \arg\max_i |g_i|. \tag{12}$$

At this time, the regularization parameter value is $\lambda = |g_j|$. From condition (10), we know that the sign of each coefficient in $\mathcal{A}$ is the same as the sign of its corresponding entry in $\mathbf{g}$. As λ further decreases, the direction of changing g_j is determined as follows:

- β_j will increase in the positive direction if $g_j = \lambda$;
- β_j will decrease in the negative direction if $g_j = -\lambda$.

Thus, we label the sign of the newly added coefficient as

$$\mathrm{sgn}(\beta_j) = \mathrm{sgn}(g_j) \tag{13}$$

As λ continues to decrease, the coefficient β_j will become nonzero to make the condition $|g_j| = \lambda$ hold. Generally, given a regularization parameter value and the optimal solution obtained for that value, the updating direction of the next optimal solution can be calculated directly from condition (10) when $\mathcal{A}$ is fixed. However, other indices may be added to $\mathcal{A}$ as λ decreases, and the updating direction needs to be re-calculated. We say an event occurs when a new point is added to be a landmark. Let $\boldsymbol{\beta}^l$, β_0^l, $\mathcal{A}^l$, $\mathbf{g}^l$ and λ^l denote the optimal solution and the values of other related parameters right after the lth event has occurred. Let $\mathcal{A}^l$ contain m indices represented as an m-tuple $\big(\mathcal{A}^l(1), ..., \mathcal{A}^l(m)\big)$, such that $\mathcal{A}^l(i) < \mathcal{A}^l(j)$ for $i < j$. We attempt to trace the regularization path of the

[1] If there are more than one point satisfying this condition, all these points are added to $\mathcal{A}$. In this case, the algorithm can still proceed without difficulty.

coefficients $\boldsymbol{\beta}_{\mathcal{A}^l}$ while other coefficients $\boldsymbol{\beta}_{\bar{\mathcal{A}}^l}$ remain zero. [2] For λ values such that $\lambda^{l+1} < \lambda < \lambda^l$, we have the following equation:

$$\Big(\mathbf{KK}(\boldsymbol{\beta}-\boldsymbol{\beta}^l)+(\beta_0-\beta_0^l)\mathbf{K1}\Big)_{\mathcal{A}^l} = (\lambda^l-\lambda)\,\mathrm{sgn}(\boldsymbol{\beta}^l_{\mathcal{A}^l}) \tag{14}$$

Let $\mathbf{k}_i$ be the ith column of the kernel matrix $\mathbf{K}$. Then $\mathbf{K}_{\mathcal{A}^l} = [\mathbf{k}_{\mathcal{A}^l(1)},...,\mathbf{k}_{\mathcal{A}^l(m)}]$ is an $n\times m$ matrix. Moreover, $\mathbf{H} = \mathbf{KK}$ and $\mathbf{H}_{\mathcal{A}^l} = [(\mathbf{H})_{ij}]_{i,j=\mathcal{A}^l(1)}^{\mathcal{A}^l(m)}$ is an $m\times m$ sub-matrix of $\mathbf{H}$. Since the coefficients $\boldsymbol{\beta}_{\bar{\mathcal{A}}^l}$ are zero, equation (14) is simplified to

$$\mathbf{H}_{\mathcal{A}^l}(\boldsymbol{\beta}_{\mathcal{A}^l}-\boldsymbol{\beta}^l_{\mathcal{A}^l})+(\beta_0-\beta_0^l)\mathbf{K}^T_{\mathcal{A}^l}\mathbf{1} = (\lambda^l-\lambda)\,\mathrm{sgn}(\boldsymbol{\beta}^l_{\mathcal{A}^l}). \tag{15}$$

From equation (4), we also have

$$\mathbf{1}^T\mathbf{K}_{\mathcal{A}^l}(\boldsymbol{\beta}_{\mathcal{A}^l}-\boldsymbol{\beta}^l_{\mathcal{A}^l})/n+(\beta_0-\beta_0^l)=0. \tag{16}$$

Hence, (15) and (16) constitute $m+1$ linear equations. We denote

$$\triangle\boldsymbol{\beta}^a = \begin{bmatrix}\beta_0-\beta_0^l\\ \boldsymbol{\beta}_{\mathcal{A}^l}-\boldsymbol{\beta}^l_{\mathcal{A}^l}\end{bmatrix},\quad \mathbf{D} = \begin{bmatrix}1 & \mathbf{1}^T\mathbf{K}_{\mathcal{A}^l}/n\\ \mathbf{K}^T_{\mathcal{A}^l}\mathbf{1} & \mathbf{H}_{\mathcal{A}^l}\end{bmatrix},$$
$$\mathbf{c}^a = \begin{bmatrix}0\\ \mathrm{sgn}(\boldsymbol{\beta}^l_{\mathcal{A}^l})\end{bmatrix}.$$

The $m+1$ linear equations can be represented in matrix form as:

$$\mathbf{D}\triangle\boldsymbol{\beta}^a = (\lambda^l-\lambda)\mathbf{c}^a. \tag{17}$$

If $\mathbf{D}$ has full rank, then $\mathbf{D}^{-1}$ exists and the *moving rate* of the optimal solution is

$$\mathbf{r}^a = \begin{bmatrix}r_0\\ \mathbf{r}_{\mathcal{A}^l}\end{bmatrix} = \mathbf{D}^{-1}\mathbf{c}^a. \tag{18}$$

As λ deceases from λ^l by a certain distance $\triangle\lambda$, the optimal solution will be updated as:

$$\begin{bmatrix}\beta_0\\ \boldsymbol{\beta}_{\mathcal{A}^l}\end{bmatrix} = \begin{bmatrix}\beta_0^l\\ \boldsymbol{\beta}^l_{\mathcal{A}^l}\end{bmatrix} + \triangle\lambda\begin{bmatrix}r_0\\ \mathbf{r}_{\mathcal{A}^l}\end{bmatrix}. \tag{19}$$

Equation (19) gives the direction to update the landmark coefficients in $\mathcal{A}$ while other coefficients remain zero. It is clear that the landmark coefficients $\beta_i (i\in\mathcal{A})$ proceed linearly in λ. Given a regularization value and the solution obtained for that value, the solutions of the neighborhood regularization values can be calculated directly. However, the algorithm needs to monitor the occurrence of any of the following events as λ decreases:

- A point $i\in\mathcal{A}^l$ leaves $\mathcal{A}^l$, i.e., $\beta_{\mathcal{A}^l(i)}=0$;
- A point $i\in\bar{\mathcal{A}}^l$ joins $\mathcal{A}^l$, i.e., $|g_i|=\lambda$.

When either of these two events occurs, the optimality conditions will no longer hold if λ is further decreased. The landmark set $\mathcal{A}$ needs to be updated and the new direction for the moving rate is calculated. By monitoring the occurrence of these events, we compute the largest $\lambda < \lambda^l$ for which an event occurs. This λ value is a breakpoint and is denoted as λ^{l+1}. From equation (19), it is easy to estimate the largest λ for an event of the first type where a landmark coefficient reaches zero. To estimate the largest λ for an event of the second type, we assume the jth ($j\in\bar{\mathcal{A}}^l$) point is the first point to become the landmark and this happens when λ has decreased by a distance d_j. Thus we have

$$\Big(\mathbf{Ky}-\mathbf{H}(\boldsymbol{\beta}^l_{\mathcal{A}^l}+d_j\mathbf{r}_{\mathcal{A}^l})-(\beta_0+d_jr_0)\mathbf{K1}\Big)_j = (\lambda^l-d_j), \tag{20}$$
or
$$\Big(\mathbf{Ky}-\mathbf{H}(\boldsymbol{\beta}^l_{\mathcal{A}^l}+d_j\mathbf{r}_{\mathcal{A}^l})-(\beta_0+d_jr_0)\mathbf{K1}\Big)_j = -(\lambda^l-d_j). \tag{21}$$

Then, the distance d_j is computed as

$$d_j = \min\left\{\frac{\lambda^l-g_j^l}{1-\mathbf{e}_j^T\mathbf{r}_{\mathcal{A}^l}-r_0(\mathbf{K1})_j},\ \frac{-\lambda^l-g_j^l}{-1-\mathbf{e}_j^T\mathbf{r}_{\mathcal{A}^l}-r_0(\mathbf{K1})_j}\right\} \tag{22}$$

where $\mathbf{e}_j = \big(\mathbf{H}_{j\mathcal{A}^l(i)}\big)_{i=1}^m$ is a vector of size m. The reason for using $+/-$ instead of $\mathrm{sgn}(g_j)$ in equations (20) and (21) is because $\mathrm{sgn}(g_j)$ for $j\in\bar{\mathcal{A}}$ may be flipped as λ decreases. Thus, it is necessary to verify the condition $g_i=|\lambda|$ for either the positive or the negative possibility. The algorithm applies equation (22) to all non-landmark points and chooses a point with the smallest distance, i.e., $d=\arg\min_{j\in\bar{\mathcal{A}}} d_j$. If this point becomes to be the landmark earlier than any landmark coefficient being zero, the parameters are updated as follows:

$$\lambda^{l+1} = \lambda^l - d, \tag{23}$$
$$\boldsymbol{\beta}^{l+1}_{\mathcal{A}^l} = \boldsymbol{\beta}^l_{\mathcal{A}^l} + d\mathbf{r}_{\mathcal{A}^l}, \tag{24}$$
$$\beta_0^{l+1} = \beta_0^l + dr_0. \tag{25}$$

This is the $(l+1)$st event. We then update $\mathcal{A}$ and continue until the algorithm terminates. Algorithm 1 shows the pseudocode of our proposed regularization path algorithm for kLASSO. It requires user to set a stopping criterion in advance to terminate the algorithm. The stopping criterion is defined depending on user's preference, where the landmark number, the prediction accuracy, the regularization value, etc can be used.

Algorithm 1 The path algorithm for kLASSO

```
Input: data $\{(\mathbf{x}_i, y_i)\}$.

set $\boldsymbol{\beta}=\mathbf{0}$ and $\beta_0=\mathbf{y}^T\mathbf{1}/n$;
create $\mathcal{A}$ through Eqn (12);
repeat
  calculate the moving rate through Eqn (18);
  find the next breakpoint of regularization value;
  update the solution $(\boldsymbol{\beta}, \beta_0)$ through Eqn (23)-(25) and
  update the active set $\mathcal{A}$;
until the stopping criterion

Output: a sequence of solutions and corresponding
breakpoints.
```

Instead of applying QP optimization multiple times to verify a limit number of solutions, our proposed algorithm can sequentially compute all kLASSO solutions for all possible regularization values. Since the regularization path proceeds in a piecewise linear fashion, any solution can be computed from two solutions at the breakpoints it is between. Hence, it is efficient to explore the regularization path by monitoring the breakpoints and remembering the solutions at these breakpoints only. Figure 2 gives an example of some coefficient curves during the algorithm.

[2] $\bar{\mathcal{A}}$ is the complement of $\mathcal{A}$, i.e., $\bar{\mathcal{A}} = \{1,\ldots,n\}\backslash\mathcal{A}$.

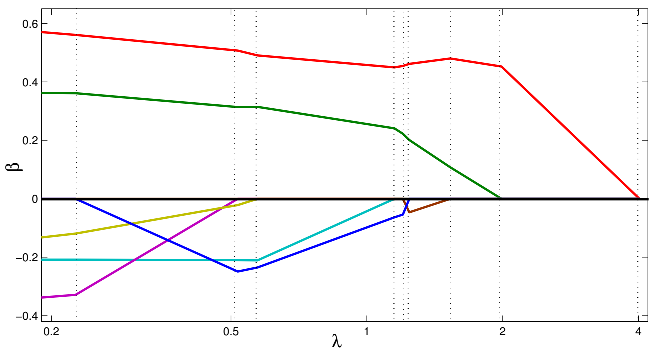

Fig. 2. An example of regularization path in kLASSO. Every curve corresponds to a coefficient path for different regularization values. The vertical lines indicate the breakpoints in the coefficient paths. The horizontal axis is in logarithm scale

The algorithm initially sets λ to infinity and all coefficients are zero. As λ decreases, more points become to be landmarks and are used to represent the regression function. It is possible for a point to enter or leave the landmark set multiple times. However, as λ decreases, less weight is put on the L_1 regularizer and the number of landmarks tends to increase. Therefore, the algorithm constructs the kLASSO regression function through adding the landmarks incrementally. From experimental results, we see that the regression function always fits the data well where only small landmarks are used.

An update is executed whenever the landmark set changes. Two matrices, $\mathbf{K}$, $\mathbf{H}$, and one vector, $\mathbf{K1}$, are used repeatedly in each update of the regularization path algorithm. As such, it is more cost-effective to calculate them in advance as a preprocessing step, with time complexity $O(n^3)$ and space complexity $O(n^2)$. In each update based on equation (19), a set of linear equations is solved to calculate the moving rate, with $O(m^3)$ time complexity. The size of the linear system m is typically much less than the number of input points n. To find next breakpoint with the smallest decreasing distance d requires that every coefficient be scanned, requiring a time complexity of $O(n)$. Thus, the computations for all coefficients based on equation (23) require $O(mn)$ complexity. Since setting λ to infinity has a simple solution, the path starts from a large λ value and extends in the direction of decreasing λ. After the algorithm starts, more points become to be landmarks, thus, the prediction accuracy of the regression function increases. As the algorithm proceeds, the regression function evolves to fit the input points better. However, the prediction accuracy is not always improved as λ decreases, since overfitting may occurs for small λ. Therefore, it is not necessary to explore the entire regularization path. As long as further decreasing λ cannot give a clear improvement in the prediction accuracy, the algorithm should be terminated in order to have a concise representation. From our extensive experiments, we observed that an optimal solution is always obtained after a very small number of iterations T, which is much less than n. This suggests that the overall computational cost is $O\left(n^3 + T(mn + m^3)\right)$.

The complexity result in the above is obtained from the preliminary complexity analysis. However, there are some optimization techniques which can be incorporated into the algorithm to accelerate the execution of the algorithm. For example, it takes $O(n^3)$ computational cost to compute the matrix $\mathbf{H}$. However, the algorithm only uses a small part of entries in $\mathbf{H}$. In equation (18), $\mathbf{H}_{\mathcal{A}}$ is used to calculate the moving rate. To estimate the value of the next breakpoint for the second type event, those entries $\mathbf{H}_{ij}$ for $i = \bar{\mathcal{A}}, j = 1, ..., n$ are needed. Hence it is not necessary to compute the whole matrix $\mathbf{H}$ in advance. We can compute those required entries in $\mathbf{H}$ as long as the algorithm needs them to proceed and they have not been computed before. Thus, the time complexity for $\mathbf{H}$ is reduced to $O(mn^2)$ from $O(n^3)$. Another computational improvement is from the observation that there is typically one element difference between $\mathcal{A}^l$ and $\mathcal{A}^{l+1}$. From the Sherman-Morrison-Woodbury formula for block matrix inversion, inverse updating/downdating in equation (18) can reduce the computations to $O(m^2)$.

4. EXPERIMENTAL RESULTS

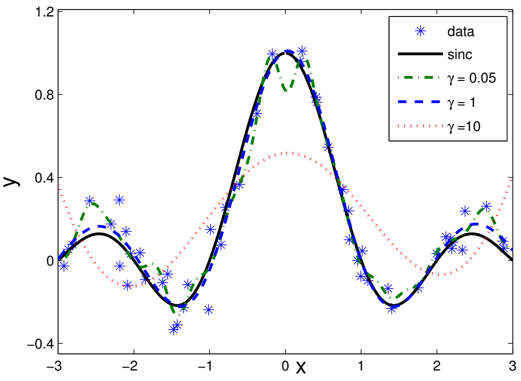

Fig. 3. Based on the kLASSO regularization path algorithm with $\gamma = 0.05, 1, 10$, the optimal solution for each path in terms of the mean square error on the validation set is plotted.

We randomly generate a set of 100 data points $\{(x_i, y_i)\}$ with x_i drawn uniformly from $[-3, 3]$ and $y_i = \frac{\sin(\pi x_i)}{\pi x_i} + e_i$, where e_i is a Gaussian noise term with mean zero and standard deviation 0.1. We randomly partition the dataset into a training set of 50 points and a validation set of 50 points. The Gaussian RBF kernel $K(x_i, x_j) = \exp(\|x_i - x_j\|^2/\gamma)$ is used with three different γ values, 0.05, 1 and 10. Since input points in this synthetic dataset are from one dimensional space, we can easily illustrate the results of the regression function in figures. In Figure 1, we choose four kLASSO solutions along the regularization path and plot corresponding regression functions and the landmarks.

We run the kLASSO regularization path algorithm until either 50 iterations are executed or λ is less than 5e-4. Further decreasing λ will make less contribution in improving the prediction accuracy. For each regularization path, we compute the mean squared error (MSE) on the validation set for all solutions at the breakpoints of the path. The solution that minimizes the MSE is then chosen and the corresponding regression function is plotted in Figure 3. We can see that three kernel values give three different regression functions: the function overfits the data when $\gamma = 0.05$ but underfits the data when $\gamma = 10$. On the other hand, it fits the data well when $\gamma = 1$.

In Figure 4(a), we plot the landmark number $|\mathcal{A}|$ as a function of the number of iterations for different γ values. When $\gamma = 0.05$, the function is very elastic. The landmark number increases greatly as the algorithm proceeds. During this process, more and more points move into the landmark set and then settle down there. The regression function is thus sensitive to many points, leading to overfitting of the data. When $\gamma = 10$, on the other hand, the number of landmarks always remains small. Since the function is not flexible enough, many points are not likely to stay at the landmark set simultaneously. This leads to underfitting of the data. It indicates that we could estimate whether a function fit the data well through observing the change in the number of landmarks without using the validation set. Figure 4(b) shows that λ decreases rapidly during the first few iterations of the regularization path algorithm. Afterwards, the rate of decrease in λ slows down significantly. There is an inflexion point after which λ has an imperceptible change between two consecutive steps. Then executing more updating steps of the algorithm slightly changes the solution. We next examine the relationship between the MSE and the iteration number in Figure 4(c) and 4(d). Since setting $\gamma = 10$ corresponds to the underfitting case, the MSEs on both training set and validation set are very large. When $\gamma = 0.05$, the MSE on training can always decrease, however, the MSE on validation does not show the same phenomenon. Instead, it slightly increases after further deceasing λ. Although the MSE on training for $\gamma = 0.5$ is larger than that for $\gamma = 0.05$, we can see the

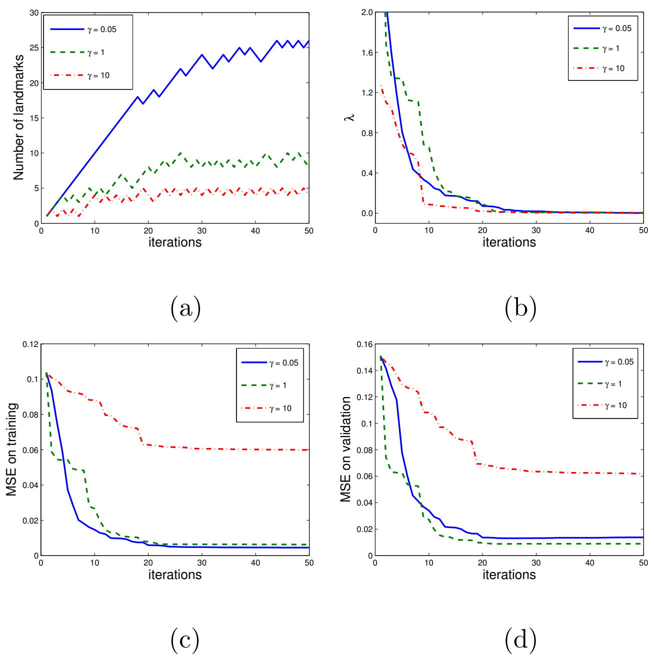

Fig. 4. The number of iterations versus the landmark number, the λ value, MSE on the training dataset and MSE on the validation dataset for the synthetic dataset generated from sinc function.

former has better generalization performance. Similar to Figure 4(b), MSEs decreases rapidly during the first few steps. The MSE on validation is minimized at around the position where the inflexion point occurs. At this time, only a small number of iterations have been executed. Further executing the regularization path cannot lead to continued improvement in the generalization ability. Instead, the resulting regression function becomes more redundant and is likely to lead to overfitting. Moreover, it incurs unnecessarily computational cost. Consequently, it is not necessary to explore the all solutions along the regularization path. The optimal solution preserving the sparseness property can be obtained very efficiently.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we propose a nonlinear LASSO formulation to the regression analysis. This formulation not only gives solutions with the sparseness property, but also has only one tuning hyperparameter which alleviates much user's efforts for model selection. We also develop a regularization path algorithm which can efficiently explore a sequence of solutions with different regularization values. Since the algorithm adds the landmark to construct the regression function, the number of landmarks for representing the regression function can be easily controlled as soon as a certain level of accuracy is reached. Therefore, it provides both representational and computational advantages for many potential applications.

REFERENCES

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transaction on Neural Network*, 12(2):181–202, 2001.

S. Rosset. Following curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, 2004.

S. Rosset and J. Zhu. Piecewise linear regularized solution paths. Technical report, Stanford University, 2003.

B. Schölkopf and A.J. Smola. *Learning with kernels.* MIT Press, 2002.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B*, 58:267–288, 1996.

V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

G. Wang, D.Y. Yeung, and F. Lochovsky. Two-dimensional solution path for support vector regression. In *Proceedings of the 23th International Conference on Machine Learning (ICML-06)*, 2006.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 2003.