Parallel Fitting of Additive Models for Regression

Valeriy Khakhutskyy[1] and Markus Hegland[2]

[1] Institute for Advanced Study, Technische Universität Munchen,
Lichtenbergstrasse 2a, D-85748 Garching, Germany
khakhutv@in.tum.de
[2] Centre for Mathematics and Its Applications, Mathematical Sciences Institute
Australian National University, Canberra, ACT 0200, Australia

Abstract. To solve big data problems which occur in modern data mining applications, a comprehensive approach is required that combines a flexible model and an optimisation algorithm with fast convergence and a potential for efficient parallelisation both in the number of data points and the number of features.

In this paper we present an algorithm for fitting additive models based on the basis expansion principle. The classical backfitting algorithm that solves the underlying normal equations cannot be properly parallelised due to inherent data dependencies and leads to a limited error reduction under certain circumstances. Instead, we suggest a modified BiCGStab method adapted to suit the special block structure of the problem. The new method demonstrates superior convergence speed and promising parallel scalability.

We discuss the convergence properties of the method and investigate its convergence and scalability further using a set of benchmark problems.

Keywords: backfitting, additive models, parallelisation, regression.

1 Introduction

Approximation of high-dimensional functions from data is an important area of machine learning research. With increasing interest in data mining and increasing computational power more effort is spent on collecting and analysing data and, hence, data mining applications continuously grow in size and dimensionality.

Some algorithms, e.g. sparse grids [1], show linear complexity with respect to the amount of data points for storage and computation. This result is already optimal, as we cannot process data without looking at least once at every data point, although the multiplicative complexity constants for different models and algorithm implementations could lead to significant differences in actual runtime. Hence, further development is focused on minimising complexity constants by improving implementation efficiency, by parallelising, or by developing sub-sampling heuristics [2–4].

Dimensionality of the approximation problem poses a more difficult challenge. In 1961 Richard Bellman coined the term "curse of dimensionality", which refers

C. Lutz and M. Thielscher (Eds.): KI 2014, LNCS 8736, pp. 243–254, 2014.

to the observation that the costs for computation and storage of an approximation for a d-dimensional function and a pre-defined accuracy ε have the complexity $\mathcal{O}\left(\varepsilon^{-d/r}\right)$ [5]. The constant r depends on the properties of the problem, e.g. smoothness, separability, etc., as well as on the kind of approximation method and its implementation.

The exponential dependency of the cost on the dimensionality is inherent in the nature of the problem. Research in information-based complexity suggests that the curse of dimensionality can be avoided only if a problem possesses a special structure that algorithms exploit, e.g. sufficiently fast converging ANOVA decomposition [6–10].

To solve big data problems which occur in modern data mining applications, a comprehensive approach is required that combines a flexible model and an optimisation algorithm with fast convergence and a potential for efficient parallelisation both in data size and dimensionality. In this paper we discuss such a comprehensive approach based on additive models and a parallel BiCGStab algorithm for optimisation.

Additive models are well established in statistics and thoroughly studied in the literature [11, 12]. The concept with somewhat different model requirements, is popular in the machine learning community. For example, recent developments apply new optimisation methods [13] and a parallelisation paradigm [14].

Moreover, estimation of additive models is an integral part of generalised additive models – a more powerful but also more computationally demanding representation concept [11].

We begin with a revision of the original formulation of additive model estimation proposed by Buja, Hastie, and Tibshirani [12] and show how beside smoothing problems (Section 2), many current regression methods can be cast into this form (Section 3). In Section 4 we then suggest a Krylov-space method for solution of normal equations and show how the problem structure can be exploited for efficiency and parallelisation. We illustrate convergence and scalability of the new method using benchmark problems in Section 5. Finally, we conclude with a discussion of the results and provide an overview of future work in Section 6.

2 Theoretical Background

We consider a dataset of the form $(\mathbf{t}^{(1)}, y^{(1)}), \ldots, (\mathbf{t}^{(N)}, y^{(N)})$ with input variables $\mathbf{t}^{(i)}$ and target variables $y^{(i)}$ and make a basic assumption about the existence of an underlying function $f(\mathbf{t})$ that generates the data and some additive model error and measurement noise summarised in the term $\epsilon(\mathbf{t})$. Then the input-target relationship can be represented as

$$y = f(\mathbf{t}) + \epsilon(\mathbf{t}).$$

Different methods to find an approximation of f in a functional space V often lead to a penalised least squares problem

$$\min_{f \in V} \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{t}^{(i)}) - y^{(i)})^2 + \lambda \|Df\|_2^2 \tag{1}$$

with a positive regularisation parameter λ and some smoothness operator D.

As mentioned above, (1) suffers from the curse of dimensionality so that only the problems with moderate dimensionality can be handled. Approaches that do not face the curse, e.g. additive models or ANOVA [15], are based on decomposition of the space V into a sum of simpler function spaces

$$V = V_1 + \ldots + V_n. \tag{2}$$

For example, in the context of ANOVA the decomposition of a function f with a d-dimensional input has the form

$$f(\mathbf{t}) = f_0 + \sum_{j=1}^{d} f_j(t_j) + \sum_{1 \le i < j \le d} f_{i,j}(t_i, t_j) + \sum_{1 \le i < j < k \le d} f_{i,j,k}(t_i, t_j, t_k) + \ldots ,$$

where t_j stands for the j-th component of the data point $\mathbf{t}$.

For tractability we drop the higher-order interaction terms. This corresponds to an assumption that the structure of the problem admits reasonably accurate representation which utilises only low-order interaction terms. In the following we consider only the 1-dimensional terms f_j. However, the extension to first-order interaction terms $f_{i,j}$ is straightforward.

While it is not assumed that the V_j are linearly independent, we assume that the penalty term is consistent with the decomposition of V, such that $f_j, j = 0, \ldots, d$, solve the optimisation problem

$$\min_{f_0 \in \mathbb{R}, f_1 \in V_1, \ldots, f_d \in V_d} \frac{1}{N} \sum_{i=1}^{N} \left(f_0 + \sum_{j=1}^{d} f_j(t_j^{(i)}) - y^{(i)} \right)^2 + \sum_{j=1}^{d} \lambda_j \|D_j f_j\|_2^2. \tag{3}$$

As discussed in the next section, many models for representation of f_j can be cast in terms of linear algebra. Denote $\mathbf{x}$ as the vector of components of $f(\mathbf{t})$ with respect to some generating system, e.g. coefficients of a polynomial model. Furthermore, let $\mathbf{Ax}$ be a vector of function values $f(\mathbf{t}^{(i)})$ and $\mathbf{y}$ the vector of data values $y^{(i)}$. Taking into account the residual $\mathbf{r}$ we obtain

$$\underbrace{[\mathbf{A}_1 \ldots \mathbf{A}_d]}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_d \end{bmatrix}}_{\mathbf{x}} = \mathbf{y} - \mathbf{r} \tag{4}$$

with $\mathbf{x}_j \in \mathbb{R}^{m_j}$, $\mathbf{A}_j \in \mathbb{R}^{N \times m_j}$, $\mathbf{y}, \mathbf{r} \in \mathbb{R}^N$, $\mathbf{x} \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{N \times m}$, and $m := m_1 + \ldots + m_d$. The problem (3) can now be written as

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{y}\|_2^2 + \lambda \mathbf{x}^T \mathbf{D} \mathbf{x} \tag{5}$$

with $\mathbf{D}$ a block diagonal matrix, which can be partitioned such that its block structure is compatible with that of $\mathbf{x}$. Often $\mathbf{D}$ will just be an identity matrix.

In order to minimise (5) one needs to solve the normal equations

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D})\mathbf{x} = \mathbf{A}^T\mathbf{y} \tag{6}$$

If we substitute (4) into (6) we obtain an $m \times m$ system

$$\begin{bmatrix} \mathbf{A}_1^T\mathbf{A}_1 + \lambda\mathbf{D}_1 & \mathbf{A}_1^T\mathbf{A}_2 & \cdots & \mathbf{A}_1^T\mathbf{A}_d \\ \mathbf{A}_2^T\mathbf{A}_1 & \mathbf{A}_2^T\mathbf{A}_2 + \lambda\mathbf{D}_2 & \cdots & \mathbf{A}_2^T\mathbf{A}_d \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^T\mathbf{A}_1 & \mathbf{A}_d^T\mathbf{A}_2 & \cdots & \mathbf{A}_d^T\mathbf{A}_d + \lambda\mathbf{D}_d \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_d \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1^T\mathbf{y} \\ \mathbf{A}_2^T\mathbf{y} \\ \vdots \\ \mathbf{A}_d^T\mathbf{y} \end{bmatrix}. \tag{7}$$

We are particularly interested in problems where (7) is used to determine the predicted values $\hat{\mathbf{f}} = \mathbf{A}\mathbf{x} = \mathbf{y} - \mathbf{r}$, which in view of (4) can be written as

$$\hat{\mathbf{f}} := \mathbf{f}_1 + \ldots + \mathbf{f}_d, \text{ with } \mathbf{f}_i = \mathbf{A}_i\mathbf{x}_i, i = 1, \ldots, d. \tag{8}$$

If we multiply every row block i of (7) by $\mathbf{A}_i(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{D}_i)^{-1}$ from the left and introduce

$$\mathbf{S}_i := \mathbf{A}_i(\mathbf{A}_i^T\mathbf{A}_i + \lambda\mathbf{D}_i)^{-1}\mathbf{A}_i^T \quad (i = 1, \ldots, d), \tag{9}$$

we obtain equations of the form

$$\underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{S}_1 & \mathbf{S}_1 & \ldots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{I} & \mathbf{S}_2 & \ldots & \mathbf{S}_2 \\ \mathbf{S}_3 & \mathbf{S}_3 & \mathbf{I} & \ldots & \mathbf{S}_3 \\ \vdots & \vdots & & \ddots & \vdots \\ \mathbf{S}_d & \mathbf{S}_d & \mathbf{S}_d & \ldots & \mathbf{I} \end{bmatrix}}_{\mathbf{S}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_3 \\ \vdots \\ \mathbf{f}_d \end{bmatrix}}_{\mathbf{f}} = \begin{bmatrix} \mathbf{S}_1\mathbf{y} \\ \mathbf{S}_2\mathbf{y} \\ \mathbf{S}_3\mathbf{y} \\ \vdots \\ \mathbf{S}_d\mathbf{y} \end{bmatrix}. \tag{10}$$

Following the tradition from statistics we call matrices $\mathbf{S}_i$ *smoothing matrices*. The normal equation (10) is the one we are interested in solving. In Section 5 we discuss why we prefer it over (7).

3 Regression Models

Problem (7) arises naturally in a number of regression models which utilise the basis expansion [16]

$$f_j(t_j) = \sum_{l=1}^{m_j} \beta_l \phi_l(t_j) \tag{11}$$

with functions ϕ_l being a basis of a function space V_j and β_l being model parameters.

The matrix $\mathbf{A}_j$ has the components

$$\{\mathbf{A}_j\}_{kl} = \phi_l(t_j^{(k)}), k = 1, \ldots, N, l = 1, \ldots, m_j.$$

We illustrate this on the example of regression splines. The derivation for other basis expansion models, e.g. linear and polynomial regression or sparse grids, is analogous.

Regression splines form piecewise-polynomials of order M (for cubic splines $M = 4$) with knots ξ_j, $j = 1, \ldots, K$. The general form for the truncated-power basis set would be [16]

$$\phi_j(t) = t^{j-1}, j = 1, \ldots, M,$$
$$\phi_{M+l}(t) = \max\{0, (X - \xi_l)^{M-1}\}, l = 1, \ldots, K.$$

The cubic spline regression solves the problem (5) with $\mathbf{D}$ having the components $\{\mathbf{D}\}_{kl} = \int \phi_k^{''}(t)\phi_l^{''}(t)dt$.

Buja et al. have shown the existence of at least one solution for the normal equations (10) for symmetric smoothers with eigenvalues in [0,1] and so this result also applies for other additive models that fit the requirements [12]. Furthermore, the Gauss-Seidel and related procedures would always converge to some solution of (10).

It can be shown that for any Hermitian positive definite matrix $\mathbf{D}_j$ and $\mathbf{A}_j$ resulting from our basis expansion the corresponding smoothing matrix $\mathbf{S}_j$ has its spectrum in $[0, 1)$ as well and hence all convergence results apply.

Problem (10) arises naturally from the models based on reproducing kernel Hilbert spaces, such as Nadaraya-Watson kernel regression.

Nadaraya-Watson kernel regression is another popular non-parametric regression method from statistics. The estimator function describes the conditional expectation of the target variable relative to the input variable:

$$f(t) = \frac{\sum_{i=1}^{N} \phi_h(t - t^{(i)})y^{(i)}}{\sum_{i=1}^{N} \phi_h(t - t^{(i)})}, \tag{12}$$

where ϕ is a kernel function with a bandwidth h.

The smoothing matrix in this case can be expressed explicitly as

$$\{\mathbf{S}_j\}_{kl} = \frac{\phi_h(\mathbf{t}_j^{(k)} - \mathbf{t}_j^{(l)})}{\sum_{i=1}^{N} \phi_h(\mathbf{t}_j^{(k)} - \mathbf{t}_j^{(i)})}, k = 1, \ldots, N, l = 1, \ldots, N \tag{13}$$

Convergence theory for this type of models is currently an area of active research [17].

4 Fitting Methods

As mentioned in the previous section, the convergence of many algorithms for solving (10) depends on properties of the spectrum of $\mathbf{S}_j$. However, while the convergence can be established, the convergence rate of the algorithms would depend on the magnitude and distribution of the system matrix eigenvalues.

A popular method to solve (10) is the Backfitting Algorithm 1. The main principle of the algorithm is a blocked Gauss-Seidel method.

Buja et al. show that the convergence rate of the backfitting algorithm heavily depends on the magnitude of the eigenvalues that are significantly smaller than

Algorithm 1 Backfitting Algorithm

Input: $\mathbf{S}_1, \ldots, \mathbf{S}_d$ smoothing matrices, $\mathbf{y}$ target vector
Output: $\mathbf{f}^T := (\mathbf{f}_1^T, \ldots, \mathbf{f}_d^T)^T$ predictions of the additive models
while not converged **do**
  for $j = 1$ to d **do**
    $\mathbf{f}_j = \mathbf{S}_j \left(\mathbf{y} - \sum_{k \neq j} \mathbf{f}_k \right)$
  end for
end while

1. This means that the error terms corresponding to the eigenvectors with eigenvalues near 1 (e.g. constant, linear and low-frequency) would not be eliminated at all by the algorithm.

Figure 1 illustrates a typical distribution of the eigenvalues of the matrix $\mathbf{S}$. The matrix has a single large eigenvalue that is equal to the d, followed by a cluster of eigenvalues in the range $[0, 1.37)$.

At this point we suggest to use a BiCGStab-based Krylov method [18] instead of Gauss-Seidel iteration for solving Equation (10) to improve the convergence. Not only is it better suited for system matrices with clustered eigenvalues, it also eliminates the error components that are problematic for backfitting and have to be handled separately, e.g. using modified backfitting [12].

Algorithm 2 shows the original formulation of the BiCGStab algorithm without preconditioning. It is not well suited for parallelisation as it requires synchronisation in lines 9, 10, 13, and 14.

Algorithm 2 BiCGStab Algorithm

1. **Input:** $\mathbf{S}, \mathbf{S}_1, \ldots, \mathbf{S}_d$ smoothing matrices, $\mathbf{y}$ target vector
2. **Output:** $\mathbf{f}$ predictions of the additive models
3. $\mathbf{r}^T = (\mathbf{r}^0)^T = (\mathbf{S}_1\mathbf{y}, \mathbf{S}_2\mathbf{y}, \ldots, \mathbf{S}_d\mathbf{y})$
4. $\mathbf{f} = \mathbf{t} = \mathbf{v} = \mathbf{s} = \mathbf{0}$
5. $\alpha = \omega = 1$; $\rho^{\text{old}} = \rho^{\text{new}} = \beta = \mathbf{r}^T\mathbf{r}$
6. **while** not converged **do**
7. $\quad \beta = \rho^{\text{new}}/\rho^{\text{old}} \cdot \alpha/\omega$; $\rho^{\text{old}} = \rho^{\text{new}}$
8. $\quad \mathbf{p} = \beta(\mathbf{p} - \omega\mathbf{v}) + \mathbf{r}$
9. $\quad \mathbf{v} = \mathbf{S}\mathbf{p}$ {synchronisation}
10. $\quad \alpha = \rho^{\text{old}}/\mathbf{v}^T\mathbf{r}_0$ {synchronisation}
11. $\quad \mathbf{s} = \mathbf{r} - \alpha\mathbf{v}$
12. $\quad$ check convergence
13. $\quad \mathbf{t} = \mathbf{S}\mathbf{s}$ {synchronisation}
14. $\quad \omega = \frac{\mathbf{t}^T\mathbf{s}}{\mathbf{t}^T\mathbf{t}}$; $\rho^{\text{new}} = -\omega\mathbf{t}^T\mathbf{r}_0$ {synchronisation}
15. $\quad \mathbf{r} = \mathbf{s} - \omega\mathbf{t}$
16. $\quad \mathbf{f} = \mathbf{f} + \omega\mathbf{s} + \alpha\mathbf{p}$
17. $\quad$ check convergence
18. **end while**

To reduce the communication we postpone the calculation of the scalar products until the aggregation of the results of matrix-vector multiplications is necessary. We can do this because the system matrix has a very specific structure and its application on a vector requires only minimal communication:

$$\mathbf{v}_j = \mathbf{p}_j + \mathbf{S}_j(\sum_{i \neq j} \mathbf{p}_j) = \mathbf{p}_j + \mathbf{S}_j(\sum_{i=1}^{d} \mathbf{p}_i - \mathbf{p}_j),$$

$$\mathbf{t}_j = \mathbf{s}_j + \mathbf{S}_j(\sum_{i \neq j} \mathbf{s}_j) = \mathbf{s}_j + \mathbf{S}_j(\sum_{i=1}^{d} \mathbf{s}_i - \mathbf{s}_j)$$

with $\mathbf{v}_j, \mathbf{p}_j, \mathbf{t}_j, \mathbf{s}_j$ denoting the subvectors, similar to $\mathbf{x}_j$.

Furthermore, to reduce communication we found it more convenient to aggregate the sum of vectors $\mathbf{t}_i$ and $\mathbf{v}_i$ instead of $\mathbf{p}_i$ and $\mathbf{s}_i$

Algorithm 3 presents the final algorithm. Vectors with the subscript "Σ", e.g. $\mathbf{t}_\Sigma$, represent the results of the sum of individual vectors, e.g. $\sum_i \mathbf{t}_i$. The scalar products in the second argument of the `Allreduce` functions in lines 10 and 20 stand for variables containing the corresponding scalar products. In these calls a vector and scalar products are joined into a single memory segment to reduce communication.

We illustrate the scalability of the algorithm in the next section.

5 Results

To motivate the use of the smoothing matrix formulation and normal equations of the form (10) instead of (7), we begin by illustrating the convergence speed of the same problem in these two formulations.

A synthetic dataset used for this experiment was generated from a linear model with random coefficients. This dataset has 100-dimensions whereas only 10 dimensions are informative. The targets were obfuscated by additive noise term with $\mathcal{N}(0, 0.01)$:

$$y = \beta_0 + \beta_1 t_1 + \ldots + \beta_{10} t_{10} + 0 t_{11} + \ldots + 0 t_{100} + \varepsilon,$$
$$\beta_0, \beta_1, \ldots, \beta_{10} \sim \mathcal{U}(0, 100), \varepsilon \sim \mathcal{N}(0, 0.01). \tag{14}$$

Altogether 1000 points were used to generate the results. To model the function we used regression cubic splines described in Section 3 with 10 basis functions per dimension and regression parameter $\lambda = 10^{-7}$.

Figure 2 illustrates the norm of the residual and the prediction error. We normalise the measurements by the norm of the residual/error at iteration 0, so that at the beginning the residual/error norm is always 1. Equation (7) is solved using MINRES method [19], while Equation (10) is solved once using backfitting and once using BiCGStab method.

One can easily see the superiority of the problem formulation (10) both by using the classical backfitting algorithm and by the BiCGStab method. While

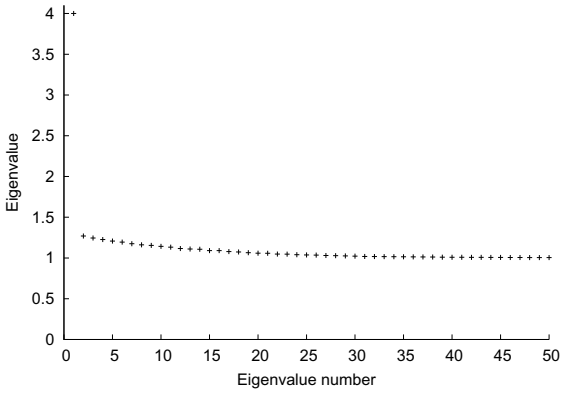
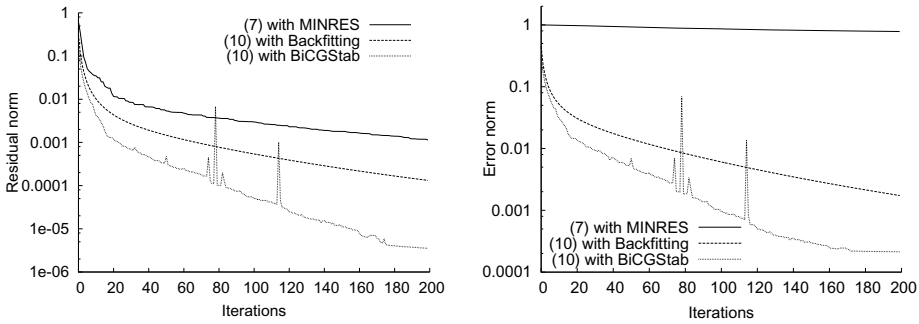


Fig. 1. Distribution of the 50 largest eigenvalues of the matrix $\mathbf{S}$ from a 4-dimensional Friedman 2 dataset with cubic spline regressors

Algorithm 3 Parallel BiCGStab Algorithm: code for processor $j, 0 \leq j \leq n-1$

Input: $\mathbf{S}_j$ smoothing matrix, $\mathbf{y}$ target vector
Output: $\mathbf{f}_j$ predictions of the function $f_j(x)$ at the data points
$\mathbf{r}_j^0 = S_j\mathbf{y}; \mathbf{r}_j = \mathbf{r}_j^0$
`Allreduce`$([\mathbf{r}_j, \mathbf{r}_j^T\mathbf{r}_j], [\mathbf{r}_\Sigma, \rho^{\text{new}}], \text{SUM})$
$\alpha = \omega = 1$; $\rho^{\text{old}} = \beta = \rho^{\text{new}}$
$\mathbf{f}_j = \mathbf{t}_j = \mathbf{v}_j = \mathbf{v}_\Sigma = \mathbf{p}_j = \mathbf{p}_\Sigma = \mathbf{s}_j = \mathbf{s}_\Sigma = \mathbf{0}$
while not converged **do**
  if iteration > 0 **then**
    $\rho^{\text{old}} = \rho^{\text{new}}$
    `Allreduce`$([\mathbf{s}_j^T\mathbf{r}_j^0, \mathbf{t}_j^T\mathbf{s}_j, \mathbf{t}_j^T\mathbf{t}_j, \mathbf{t}_j^T\mathbf{r}_j^0, \mathbf{t}_j], [\mathbf{s}^T\mathbf{r}^0, \mathbf{t}^T\mathbf{s}, \mathbf{t}^T\mathbf{t}, \mathbf{t}^T\mathbf{r}^0, \mathbf{t}_\Sigma], \text{SUM})$
    $\omega = \frac{\mathbf{t}^T\mathbf{s}}{\mathbf{t}^T\mathbf{t}}; \rho^{\text{new}} = \mathbf{s}^T\mathbf{r}^0 - \omega\mathbf{t}^T\mathbf{r}^0; \beta = \frac{\rho^{\text{new}}}{\rho^{\text{old}}} \cdot \frac{\alpha}{\omega}$
    $\rho^{\text{old}} = \rho^{\text{new}}$
    $\mathbf{r}_j = \mathbf{s}_j - \omega\mathbf{t}_j; \mathbf{r}_\Sigma = \mathbf{s}_\Sigma - \omega\mathbf{t}_\Sigma$
    $\mathbf{f}_j = \mathbf{f}_j + \omega\mathbf{s}_j$
    check convergence on $\mathbf{r}$
  end if
  $\mathbf{p}_j = \beta(\mathbf{p}_j - \omega\mathbf{v}_j) + \mathbf{r}_j$
  $\mathbf{p}_\Sigma = \beta(\mathbf{p}_\Sigma - \omega\mathbf{v}_\Sigma) + \mathbf{r}_\Sigma$
  $\mathbf{v}_j = \mathbf{p}_j + S_j(\mathbf{p}_\Sigma - \mathbf{p}_j)$
  `Allreduce`$\big([\mathbf{v}_j^T\mathbf{r}_j^0, \mathbf{v}_j], [\mathbf{v}^T\mathbf{r}^0, \mathbf{v}_\Sigma], \text{SUM}\big)$
  $\alpha = \rho^{\text{old}}/\mathbf{v}^T\mathbf{r}^0$
  $\mathbf{s}_j = \mathbf{r}_j - \alpha\mathbf{v}_j; \mathbf{s}_\Sigma = \mathbf{r}_\Sigma - \alpha\mathbf{v}_\Sigma$
  $\mathbf{f}_j = \mathbf{f}_j + \alpha\mathbf{p}_j$
  check convergence on $\mathbf{s}$
  $\mathbf{t}_j = \mathbf{s}_j + S_j(\mathbf{s}_\Sigma - \mathbf{s}_j)$
end while

the residual in the formulation (7) decreases, it does not seem to significantly reduce the error of the resulting additive model which is our main goal.



(a) Residual (b) Error

Fig. 2. Comparison of problem formulations (7) solved with MINRES and (10) solved with backfitting and BiCGStab for minimisation of residual and error of a synthetic 100-dimensional dataset from model (14)

Figure 2 also gives an impression of the convergence comparison between backfitting and BiCGStab. Backfitting shows more stable, but slower convergence even on this simple example.

We extend the numerical comparison by a number of benchmark datasets (see Table 1 for description). Tables 2 and 3 compare the final normalised residual and error for a number of problems. As our main focus is the solution of Equation (10) we do not discuss the selection of optimal learning parameters and confine to setting the same parameters for the backfitting and BiCGStab.

Table 1. Summary description of the benchmark datasets

Dataset	Samples	Dimensions	Source
Boston Housing	506	13	[20]
DR 5	10000	6	[21]
Diabetes	442	10	[22]
Spam	3064	16	[16]
Friedman 1	2000	10	[23]
Friedman 2	1000	4	[23]
Friedman 3	1000	4	[23]

The BiCGStab method is at least as good and in many cases clearly superior to the backfitting algorithm. We also observed that, depending on the problem, backfitting would catch up with more iterations allowed or stagnate with a higher error.

Table 2. Results for cubic spline regression

Dataset	m_j	λ	Iter.	BiCGStab res.	BiCGStab error	BF res.	BF err.
Boston Housing	20	1E-4	6	1.87E-2	3.08E-1	3.48E-2	3.17E-1
DR 5	50	1E-6	11	1.77E-2	4.60E-1	4.74E-2	5.40E-1
Diabetes	25	1E-10	5	7.75E-2	5.46E-1	1.20E-1	5.85E-1
Spam	100	1E-10	6	2.09E-2	4.26E-1	3.27E-2	4.31E-1
Friedman 1	12	1E-7	3	3.44E-4	3.25E-1	1.04E-3	3.25E-1
Friedman 2	8	1E-7	2	9.49E-4	4.76E-1	7.17E-3	4.76E-1
Friedman 3	8	1E-7	2	6.48E-4	4.49E-1	6.98E-3	4.49E-1

Table 3. Results for Nadaraya-Watson regressor with Gaussian kernel

Dataset	Kernel Width	Iter.	BiCGStab res.	BiCGStab err.	BF. res.	BF. err.
Boston Housing	10	6	9.33E-3	2.81E-1	3.03E-2	2.88E-1
DR 5	10	12	1.61E-2	4.21E-1	4.27E-2	4.96E-1
Diabetes	10	4	1.00E-2	5.45E-1	5.53E-2	5.48E-1
Spam	10	5	1.43E-2	3.91E-1	4.00E-2	3.97E-1
Friedman 1	10	5	5.95E-3	2.53E-1	6.90E-3	2.53E-1
Friedman 2	10	3	2.27E-3	4.40E-1	7.67E-3	4.40E-1
Friedman 3	10	3	2.21E-3	4.23E-1	7.39E-3	4.28E-1

To study the scalability of the Algorithm 3 we implemented it in C++ using `MPI_Allreduce` function for data aggregation. Every processor j is responsible for one $\mathbf{S}_j$ smoothing matrix so that increasing number of processors corresponds to fitting an additive model approximation problem with more dimensions or interaction terms. We focus on the study of weak scaling of the Algorithm 3 since our primary goal is parallelisation as a mean to solve higher-dimensional problems.

In our experiment every processor j generates a random full-rank symmetric positive definite matrix of a given size with spectrum in $[0, 1]$ to use as the matrix $\mathbf{S}_j$. As, obviously, the number of synchronisation steps would usually increase if we solve a higher-dimensional problem, we limit the runs to 50 iterations in every case. Figure 3 illustrates the weak scaling results. The total time slightly increases as the communication time grows logarithmically with the number of processors. This growth however is marginal.

It is straightforward to parallelise matrix-vector and scalar products on a shared memory (MKL, OpenBLAS) and distributed memory (PBLAS) architectures for solving problems with more data. This would allow one to study strong scaling properties and is a topic for future work.

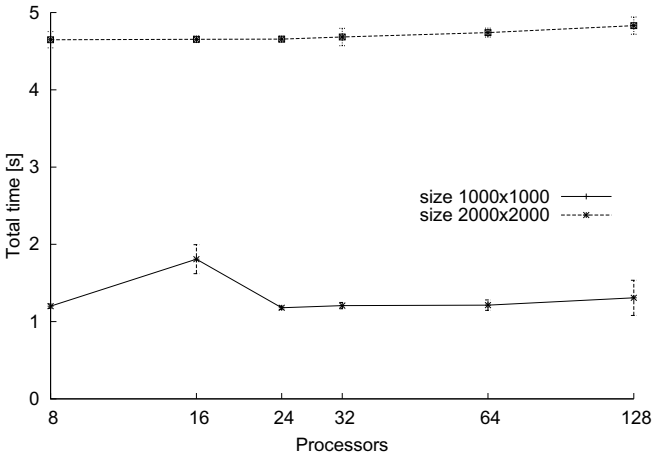


Fig. 3. Weak scaling results for Algorithm 3. Time measurements were performed after 50 iterations with individual matrices $\mathbf{S}_j$ of sizes 1000×1000 and 2000×2000. Median of 7 trials was used in the diagram, while bars indicate the standard deviation.

6 Conclusion

We presented a new approach for fitting additive models using BiCGStab algorithm that, besides smoothing, can be used for larger regression problems. This approach overcomes the shortcomings of the classical backfitting algorithm.

While the convergence of the BiCGStab method cannot be proved theoretically, it usually works well in practice. It converges fast and can be efficiently parallelised for distributed computer architectures.

Our future work will include parallelisation of the data-intensive operations as well as development of preconditioning methods to stabilise and accelerate the convergence.

References

1. Garcke, J., Griebel, M., Thess, M.: Data mining with sparse grids. Computing 67(3), 225–253 (2001)
2. Pflüger, D.: Spatially Adaptive Sparse Grids for High-Dimensional Problems. Verlag Dr. Hut, München (2010)
3. Heinecke, A., Pflüger, D.: Emerging architectures enable to boost massively parallel data mining using adaptive sparse grids. International Journal of Parallel Programming, 1–43 (July 2012)
4. Xu, W.: Towards optimal one pass large scale learning with averaged stochastic gradient descent. CoRR, abs/1107.2490 (2011)
5. Bellman, R., Bellman, R.: Adaptive Control Processes: A Guided Tour. Rand Corporation. Research studies, Princeton University Press (1961)

6. Novak, E., Wozniakowski, H.: Tractability of Multivariate Problems, Volume I: Linear Information. European Mathematical Society, Zürich (2008)
7. Novak, E.: Tractability of multivariate problems, Volume II: Standard Information for Functionals. European Mathematical Society, Zürich (2010)
8. Novak, E., Wozniakowski, H.: Tractability of Multivariate Problems, Volume III: Standard Information for Operators. European Mathematical Society, Zürich (2012)
9. Novak, E., Woźniakowski, H.: Approximation of infinitely differentiable multivariate functions is intractable. Journal of Complexity 25, 398–404 (2009)
10. Hegland, M., Wasilkowski, G.W.: On tractability of approximation in special function spaces. J. Complex. 29, 76–91 (2013)
11. Hastie, T., Tibshirani, R.: Generalized Additive Models. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis (1990)
12. Buja, A., Hastie, T., Tibshirani, R.: Linear smoothers and additive models. The Annals of Statistics 17, 453–510 (1989)
13. Chu, E., Keshavarz, A., Boyd, S.: A distributed algorithm for fitting generalized additive models. Optimization and Engineering 14, 213–224 (2013)
14. Hsu, D., Karampatziakis, N., Langford, J., Smola, A.: Parallel online learning, ch. 14. Cambridge University Press (2011)
15. Stone, C.J.: The dimensionality reduction principle for generalized additive models. The Annals of Statistics 14, 590–606 (1986)
16. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer Series in Statistics. Springer (2011)
17. Xia, Y.: A note on the backfitting estimation of additive models. Bernoulli 15, 1148–1153 (2009)
18. van der Vorst, H.: Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing 13(2), 631–644 (1992)
19. Paige, C., Saunders, M.: Solution of sparse indefinite systems of linear equations. SIAM Journal on Numerical Analysis 12(4), 617–629 (1975)
20. Harrison, D.J., Rubinfeld, D.L.: Hedonic housing prices and the demand for clean air. Journal of Environmental Economics and Management 5(1), 81–102 (1978)
21. Adelman-McCarthy, J.K., et al.: The fifth data release of the sloan digital sky survey. The Astrophysical Journal Supplement Series 172(2), 634 (2007)
22. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. The Annals of Statistics 32, 407–499 (2004)
23. Friedman, J.H.: Multivariate adaptive regression splines. The Annals of Statistics 19, 1–67 (1991)