. CI. OMPUT.
. 31, . 3, . 1840 1857

MULTIVARIATE REGRESSION AND MACHINE LEARNING WITH SUMS OF SEPARABLE FUNCTIONS*

Before making any decisions on how to move your arm and hand, you need some estimate of the current position of your finger. Between your forehead and your finger are several joints, with (at least) $d = 10$ angles. One strategy is to use sensory input from your muscles and joints to determine these angles and then use geometry to calculate the distance (or vector) from your finger to your forehead. A second strategy,

We found in [1, 2, 23] that this extra freedom allows one to find good approximations

as measured by the decay of its Fourier transform. Fifth, note that Gaussians are separable, since

$$\exp\left(-c\ \mathbf{x}-\mathbf{z}\ ^2\right)=\prod_{i=1}^{d}\exp\left(-c(x_i-z_i)^2\right). \tag{1.9}$$

By expanding a radial function in Gaussians, one can obtain a separated representation for it. Several important operators, such as the inverse Laplacian, have radial kernels and, thus, can be represented using this technique (see, e.g., [2]).

The goals of this paper are to present algorithms to construct regression functions of the form (1.3) and to give preliminary numerical evidence that such representations are worth using.

First, in section 2, we construct and present the basic algorithm for constructing a regression function of the form (1.3). We also present several variants; in particular, we show how to handle vector-valued data and how to incorporate regularization to encourage smoothness and avoid overfitting. The algorithms are linear in both N and d and so are suitable for large data sets in high dimensions. The basic algorithm depends (quadratically) on r, and, thus, the central remaining issue is how large r must be in practice.

Second, in section 3, we demonstate by numerical experiments that interesting functions can indeed be well approximated in the form (1.3) with small r. One can of course construct functions where these methods would fail. However, our experiments confirm that the class of functions that we can approximate well is wide enough to include "naturally occurring" functions of many variables, and so these methods are useful in practice. In particular, we show that our method outperforms other methods on several benchmark data sets.

Remark 1.1. It may be appropriate to first transform the data locations using a dimensionality-reduction technique such as principle component analysis (see, e.g., [16]), the Johnson–Lindenstrauss lemma [19], or manifold learning (e.g., [6]) and then apply our method in the reduced coordinates. Thus, while our method may sometimes be used instead of these techniques, it can also be used in conjunction with them.

2. Description of the algorithm.

In this section we describe the basic algorithm and several variants. First, we give the core principles, which allow us to reduce to one-dimensional subproblems. Second, we describe how to solve this one-dimensional subproblem when using a linear function space and how to set up the problem in the nonlinear case. Third, we consider how to incorporate procedures to avoid overfitting. Finally, we extend the method to vector-valued data and functions.

2.1. Core principles.

2.1.26 Tc 5(1.)4.M(e)-385(.8(i)-a5cd)-2/F9 wT10a5(1.)4.i19(n)5.1n

evaluations at the data points, we may take inner products with our data, i.e.,

$$\langle D, g \rangle_D = \left\langle \{(\mathrm{x}_j, y_j)\}_{j=1}^N, g \right\rangle_D = \sum_j^N y_j g(\mathrm{x}_j). \tag{2.2}$$

Thus, we can treat the data as if it were some unknown function. In the associated pseudo norm the usual least-squares error is then simply given by

$$\left\| \{(\mathrm{x}_j, y_j)\}_{j=1}^N - \right. \tag{2.3}$$

2.1.4. Computational cost, so far. Although we have deferred the discussion on solving the one-dimensional subproblems, it is worthwhile at this point to account

each of which is a vector of length M_l. The matrix $\mathbb{A}$, thus, depends only on the data locations $\{x_j\}$

2.3. Avoiding overfitting. One issue to address for regression/learning algorithms is the avoidance of overfitting. As an extreme example of overfitting, one could use the function

$$g(\mathbf{x}) = \sum_{j=1}^{N} y_j \exp(-c\ \mathbf{x} - \mathbf{x_j}\ ^2) \tag{2.15}$$

to represent the data. For large enough c, this function would match the given data nearly exactly but be completely unreasonable for other locations. There are two

Table 3.1

*Mean squared error (MSE) for fitting the Friedman*1 *data set with no noise, using polynomials in each direction. The degree* 0

Table 3.2

MSE for fitting the Friedman2 data set with no noise, using polynomials in each direction. The degree 0 *entry estimates the variance in our realization of the data.*

	$r = 1$	$r = 2$	$r = 3$
degree 0	143303		
degree 1	177	51	51
degree 2	162	21	20
degree 3	155	11	9

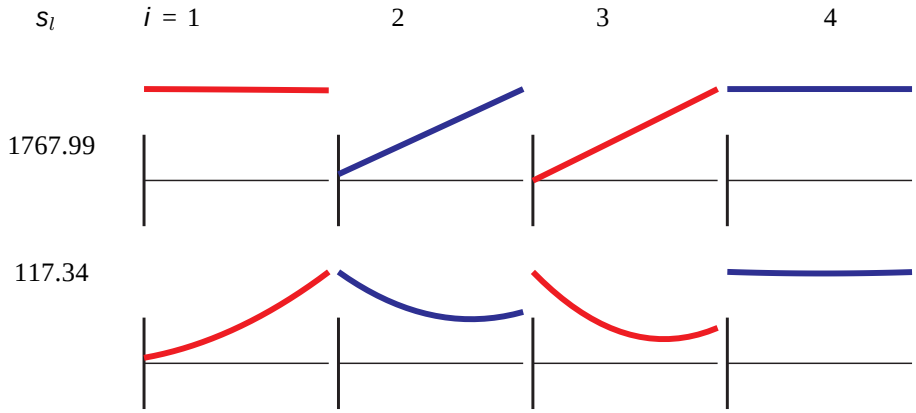

Fig. 3.2. *The function of the form* (1.3) *with* $r = 2$ *using polynomials of degree two that captures* 99.985% *of the variance of the Friedman2 function* (3.2). *Each subplot shows a* $g_i^l(x_i)$.

Table 3.3

MSE for fitting the Friedman3 data set with no noise, using polynomials in each direction. The degree

s_l	$i = 1$	2	3	4
0.1523				
2.2228				
2.0519				
0.9515				

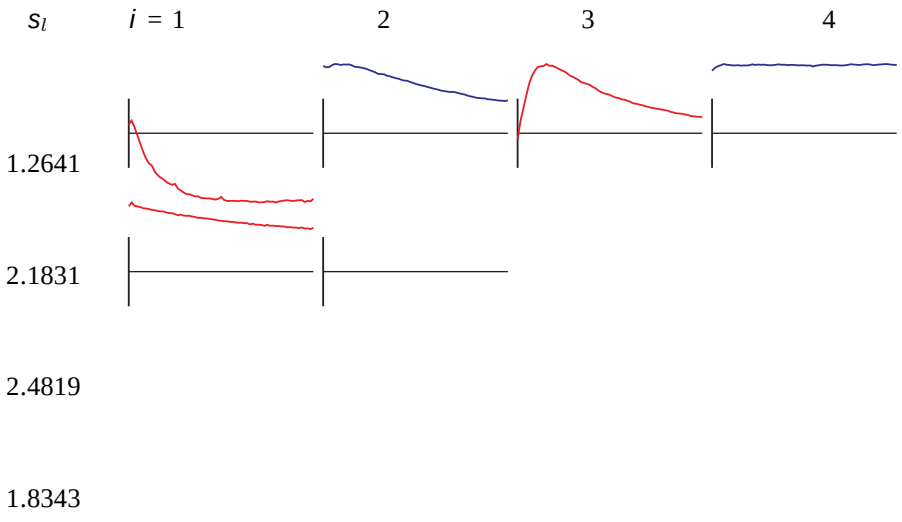

Table 3.6

Summary of the results on synthetic data as described in sections 3.1.1, 3.1.2, *and* 3.1.3. *We give the mean (with standard deviation) and median (with interquartile range) of the MSE for our approach, the best results from* [22], *and our rank in comparison to all approaches used in* [22].

		Best result from [22]		Our approach	
Data set	Criteria	Method	Error	Error	Rank
Friedman1	mean	Bruto	3.22	2.44 (0.18)	1/11
	median	Bruto	3.20 (0.14)	2.34 (0.26)	1/11
Friedman2	mean	svm	18130	16897 (520)	1/11
	median	svm	17990 (760)	16658 (422)	1/11
Friedman3	mean	nnet	0.01812	0.02172 (0.00119)	2/11
	median	nnet	0.01625 (0.00129)	0.02048 (0.00174)	2/11

On the Friedman1 data set, the variance of the noise is set to 1. We chose a basis of monomials, penalized them by their degree, and scaled the penalties by the overall factor λ.

On the Friedman2 data set, the variance of the noise is set to $125^2 = 15625$, which gives a signal-to-noise ratio of 3:1. We noticed that the simple setting of polynomials of degree one and separation rank one was often chosen in the parameter search. Since the MSE is already close to the variance of the noise, using higher degree or separation rank more likely results in overfitting.

On the Friedman3 data set, the variance of the noise is $0.1^2 = 0.01$, again resulting in a signal-to-noise ratio of 3:1 (based on the reported signal variance). As discussed in section 3.1.3, polynomials are a poor choice, so we use the multilevel basis. We chose a penalty of 0 for the constant term, 1 for the x term, and then doubling at each level, and scaled the penalties by the overall factor λ.

For the first two Friedman data sets we achieve the best performance in comparison to the benchmark study [22], and on the third data set only a highly nonlinear approach achieves better results.

3.2.2. Real data sets. We now compare with some of the real data sets used in [22]. Preprocessing of the data consists of omitting missing values, like in [22], and scaling all data to $[0, 1]^d$. We did not use two of the data sets since they were dominated by categorical attributes. In data sets with only two or three categorical values we use the usual binary coding scheme or choose a basis of vectors; see Remark 2.3. In our tests these two approaches gave similar numerical results.

Table 3.8

Warsaw, Poland, September 2007, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladen, and A. Skowron, eds., Lecture Notes in Artificial Intelligence 4701, Springer, Berlin, 2007, pp. 42–53.

[4] R. Bro, *Parafac. tutorial & applications*, in Chemom. Intell. Lab. Syst., 38 (1997), pp. 149–171; also available online from http://www.models.kvl.dk/users/rasmus/presentations/parafac_tutorial/paraf.htm.

[5] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.

[6] R. R. Coifman and S. Lafon, *Di usion maps*, Appl. Comput. Harmon. Anal., 21 (2006), pp. 5–30.

[7]