Spline-based separable expansions for approximation, regression and classification

Nithin Govindarajan

Nico Vervliet, Lieven De Lathauwer

IPAM Workshop I: Tensor Methods and their Applications in the Physical and Data Sciences, UCLA, United States, April 1, 2021
What are we trying to accomplish?

Introduce a new technique for modeling functions in several variables:

- Regression tasks
- Classification tasks

Our recent submission to Frontiers:
*Regression and classification with spline-based separable expansions.*
N. Govindarajan, N. Vervliet, L. De Lathauwer.
The main challenge of approximating functions in high dimensions

**Curse-of-dimensionality** in approximation theory:

*In general, to approximate a $n$-times differentiable function in $D$ variables within $\varepsilon$-tolerance (measured in the uniform norm), one typically requires $M \gtrsim \left(\frac{1}{\varepsilon}\right)^{D/n}$ parameters*


caveat:

*Many high-dimensional functions in applications are inherently of “low complexity”*
Focus of this talk: exploiting low-rank structures through sums of separable functions

\[ f(x) = \sum_{r=1}^{R} \left( \prod_{d=1}^{D} \phi_{r,d}(x_d) \right) \]

Sums of separable functions = continuous analogs of polyadic decompositions
Revisiting this problem: are there any benefits of using splines over polynomials?

Past work (e.g., Mohlenkamp & Beylkin) mostly considered polynomials to approximate the component functions $\phi_{r,d}(\cdot)$, why not use piece-wise polynomials a.k.a. splines?
What to expect next?

Spline basics and splines in higher dimensions: exploiting low-rank structures
Performing regression and classification
A Gauss–Newton algorithm exploiting sparsity
Numerical examples (regression)
Numerical examples (classification)
Key take-aways and future work
The knot set and B-spline basis terms

Let $\mathcal{T} = \{t_i\}_{i=0}^{N+M}$ denote the set of knots:

$$a = t_0 = \ldots = t_{N-1} \leq t_N \leq t_{N+1} \leq \ldots \leq t_{M+1} = \ldots = t_{M+N} = b.$$ 

The B-spline basis terms $\{B_{m,N}\}_{m=0}^{M}$ are defined through the recursion formula

$$B_{m,N}(x) := \frac{x - t_m}{t_{m+N} - t_m} B_{m,N-1}(x) + \frac{t_{m+N+1} - x}{t_{m+N+1} - t_{m+1}} B_{m+1,N-1}(x),$$

where $B_{m,0}(x) := \begin{cases} 1 & x \in [t_m, t_{m+1}) \\ 0 & \text{otherwise} \end{cases}$.
The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!

$$B_{m,N}(x) = 0, \quad x \in (\infty, t_m) \cup [t_{m+N+1}, \infty).$$
The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!

\[ B_{m,N}(x) = 0, \quad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty). \]
The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!

\[ B_{m,N}(x) = 0, \quad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty). \]
The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!

\[ B_{m,N}(x) = 0, \quad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty). \]
The B-spline function

Any continuous function can be approximated arbitrarily well by

\[ S(x) = \begin{bmatrix} B_{0,N}(x) & \cdots & B_{M,N}(x) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_M \end{bmatrix} = B_{\mathcal{S},N}(x)c. \]

through either *increasing* the knot density and order of the spline.
Taking direct tensor products of splines leads to exponential blow-up of coefficients...

\[
\hat{f}(x; C) = \sum_{m_1=0}^{M_1} \cdots \sum_{m_D=0}^{M_D} c_{m_1 \cdots m_D} \prod_{d=1}^{D} B_{m_{d},N(d)}^{(d)}(x_d) = C \cdot \prod_{d=1}^{D} B_d(x_1) \cdots B_D(x_D)
\]

\[\prod_{d=1}^{D} (M_d + 1) \text{ parameters}\]
Exploit low-rank structure: $C(\Gamma_1, \ldots, \Gamma_D) = [\Gamma_1, \ldots, \Gamma_D]$, to alleviate this blow-up!

$$\hat{f}(x; \Gamma_1, \ldots, \Gamma_D) = C(\Gamma_1, \ldots, \Gamma_D) \cdot 1 B_1(x_1) \cdots D B_D(x_D) = \sum_{r=1}^{R} \prod_{d=1}^{D} B_d(x_d) \gamma_{r,d}$$

\[ \prod_{d=1}^{D} (M_d + 1) \text{ parameters} \quad \text{=} \quad \gamma_{1,1} + \cdots + \gamma_{R,1} \quad \text{R} \left( \sum_{d=1}^{D} M_d + 1 \right) \text{ parameters} \]
Splines basics and splines in higher dimensions: exploiting low-rank structures

Performing regression and classification

A Gauss–Newton algorithm exploiting sparsity

Numerical examples (regression)

Numerical examples (classification)

Key take-aways and future work
Regression is performed with the quadratic objective function

Given samples $\{(x_i, y_i)\}_{i=1}^I \subset [0,1]^D \times \mathbb{R}$ from a underlying target function $f \in C([0,1]^D)$, we minimize:

$$Q(\Gamma_1, \ldots, \Gamma_D) := \frac{1}{2} \sum_{i=1}^I \left( \hat{f}(x_i; \Gamma_1, \ldots, \Gamma_D) - y_i \right)^2.$$
A level-set approach to modeling a binary classification function

Binary classification function $g : [0, 1]^D \rightarrow \{0, 1\}$ can be modeled by the function

$$g(x) = \begin{cases} 0 & f(x) \leq 0 \\ 1 & f(x) > 0 \end{cases}$$
Replace step function with the logistic function $\sigma_\alpha : t \mapsto 1/(\exp(-\alpha t) + 1)$

Replace $g$ with

$$g_\alpha(x) := (\sigma_\alpha \circ f)(x) = \sigma_\alpha(f(x)),$$

where $\alpha > 0$ controls sharpness of transition.
Logistic objective function

\( g_\alpha \) is replaced by the approximant

\[
\hat{g}_\alpha(x; \Gamma_1, \ldots, \Gamma_D) := \sigma_\alpha \circ \hat{f}(x; \Gamma_1, \ldots, \Gamma_D),
\]

Given a collection of labeled data \( \{(x_i, y_i)\}_{i=1}^I \subset [0, 1]^D \times \{0, 1\} \), the performance of \( \hat{g}_\alpha \) is optimized by maximizing the quantity

\[
0 \leq \prod_{y_i=0} (1 - \hat{g}_\alpha(x_i; \Gamma_1, \ldots, \Gamma_D)) \prod_{y_i=1} \hat{g}_\alpha(x_i; \Gamma_1, \ldots, \Gamma_D) \leq 1,
\]
Logistic objective function

$g_\alpha$ is replaced by the approximant

$$\hat{g}_\alpha(x; \Gamma_1, \ldots, \Gamma_D) := \sigma_\alpha \circ \hat{f}(x; \Gamma_1, \ldots, \Gamma_D),$$

Equivalent to minimizing the objective function

$$L_\alpha(\Gamma_1, \ldots, \Gamma_D) := -\sum_{i=1}^I y_i \log \hat{g}_\alpha(x_i; \Gamma_1, \ldots, \Gamma_D) + (1 - y_i) \log (1 - \hat{g}_\alpha(x_i; \Gamma_1, \ldots, \Gamma_D)).$$
Spline basics and splines in higher dimensions: exploiting low-rank structures
Performing regression and classification
A Gauss–Newton algorithm exploiting sparsity
Numerical examples (regression)
Numerical examples (classification)
Key take-aways and future work
Minimization of objective functions is effectively done with Gauss-Newton dogleg algorithm.

- Exploit multi-linear structure of the objective functions, see:

- Main computational burden:
  
  evaluating gradients and Grammian-vector products.
Benefit of compactly supported B-splines: significant speed-ups in Grammian and gradient by exploiting sparsity!

Gradient:

\[ g_{r,d} = A_d \left( \left( \sum_{k=1, k \neq d}^D A_k^T \gamma_{r,k} \right) \ast \eta \right). \]

Grammian (of the Jacobian) vector product

\[ w_{r,d} = A_d \left( \left( \sum_{\tilde{d}=1}^D \sum_{\tilde{r}=1}^R \left( \sum_{k=1, k \neq \tilde{d}}^D A_k^T \gamma_{\tilde{r},k} \right) \ast \sum_{\tilde{d}=1}^D A_d^T z_{\tilde{r}, \tilde{d}} \right) \right). \]
Benefit of compactly supported B-splines: significant speed-ups in Grammian and gradient by exploiting sparsity!

If the order of the B-spline is kept low:

\[ O(DIMR) \rightarrow O(DIR) \text{ flops} \]
Benefit of compactly supported B-splines: significant speed-ups in Grammian and gradient by exploiting sparsity!

average required computation time to pass through one cycle of the GN algorithm. \( N = 4, R = 3, l = 1000. \)
Spline basics and splines in higher dimensions: exploiting low-rank structures
Performing regression and classification
A Gauss–Newton algorithm exploiting sparsity
Numerical examples (regression)
Numerical examples (classification)
Key take-aways and future work
A $R = 3$ separable function

Consider the following example

$$f(x) = |x_1||x_2| + \sin(2\pi x_1) \cos(2\pi x_2) + x_1^2 x_2, \quad x \in [-1, 1] \times [-1, 1].$$

non-smooth term
As expected... an $R = 3$ is sufficient for a good approximation

(Knots are uniformly distributed on the approximation domain)
As expected... an $R = 3$ is sufficient for a good approximation

(Knots are uniformly distributed on the approximation domain)
As expected... an $R = 3$ is sufficient for a good approximation

(Knots are uniformly distributed on the approximation domain)
As expected... an $R = 3$ is sufficient for a good approximation.

(Knots are uniformly distributed on the approximation domain)
Unlike for splines, Runge’s phenomenon can adversely affect quality of approximation.

No of separable terms $R = 3$.

(Knots are uniformly distributed on the approximation domain)
Taming Runge’s phenomenon with splines: *keep order low and increase knots*

Runge’s phenomenon can adversely contribute to the overfitting problem
Low-rank structures in real life datasets - an example

NASA dataset from the UCI machine learning repository:

- **Independent variables:**
  - frequency,
  - angle of attack,
  - chord length,
  - free-stream velocity,
  - suction-side displacement thickness.

- **Dependent variable:** *self-noise generated by airfoil.*

- Randomly split data into a training (1202 samples) and a test (301 samples) sets.
An $R = 5$ separable function is sufficient to model the NASA dataset.
Spline basics and splines in higher dimensions: exploiting low-rank structures

Performing regression and classification

A Gauss–Newton algorithm exploiting sparsity

Numerical examples (regression)

Numerical examples (classification)

Key take-aways and future work
The separable rank can be increased to account for complexity of the classification sets

Consider the labeled dataset:
The separable rank can be increased to account for complexity of the classification sets.
The separable rank can be increased to account for complexity of the classification sets.
The separable rank can be increased to account for complexity of the classification sets.
The separable rank can be increased to account for complexity of the classification sets.
The separable rank can be increased to account for complexity of the classification sets.
Our method compared with well-established techniques for classification

<table>
<thead>
<tr>
<th>no. of training samples</th>
<th>FiC training (%)</th>
<th>no. of training samples</th>
<th>FiC test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

CPD spline with R=7, M=16
SVM with RBF kernel
SVM with order 9 polynomial kernel
Patternnet with 50 nodes
CPU time for training grows more moderately with dataset size.
Spline basics and splines in higher dimensions: exploiting low-rank structures
Performing regression and classification
A Gauss–Newton algorithm exploiting sparsity
Numerical examples (regression)
Numerical examples (classification)
Key take-aways and future work
Key take-aways and future work

Important take-aways:

- With B-splines, sparsity can be exploited to further accelerate GN algorithm
- Runge phenomenon effects are easily suppressed by keeping order of the spline low
- Low-rank structures do appear in practice!
- A new promising technique for (binary) classification

Future work:

- Extend to other decompositions, e.g., Hierarchical Tucker, Tensor Train,
- multi-class classification,
- knot optimization
Spline-based separable expansions for approximation, regression and classification

Nithin Govindarajan

Nico Vervliet, Lieven De Lathauwer

IPAM Workshop I: Tensor Methods and their Applications in the Physical and Data Sciences, UCLA, United States, April 1, 2021