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

KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

erc
European Research Council
Established by the European Commission

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 ϵ-tolerance (measured in the uniform norm), one typically requires $M \gtrsim (\frac{1}{\epsilon})^{D/n}$ parameters*

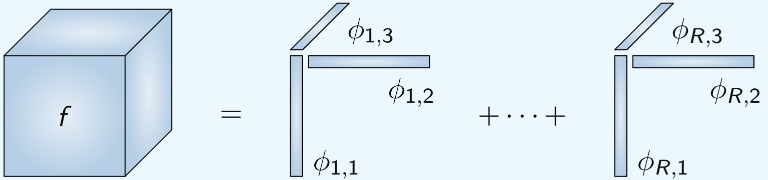
Optimal nonlinear approximation. DeVore et al., Manuscripta mathematica, 1989.

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

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



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

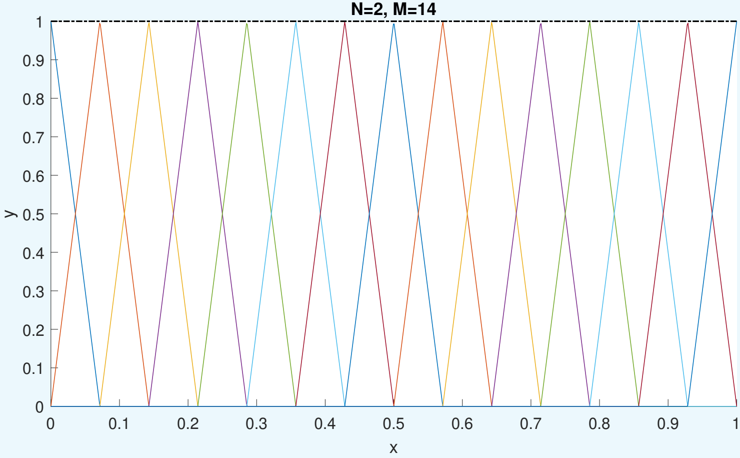
$$a = t_0 = \ldots = t_{N-1} \leq t_N \leq t_{N+1} \leq \ldots \leq t_{M+1} = \ldots = t_{M+N} = b.$$

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

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

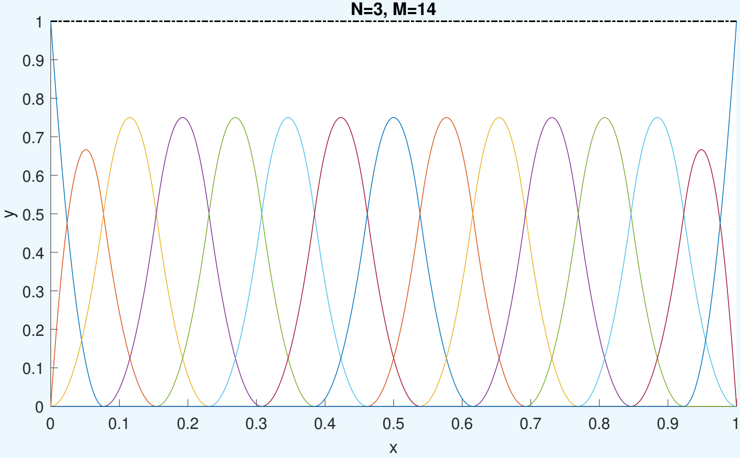
where $B_{m,0}(x) := \begin{cases} 1 & x \in [t_m, t_{m+1}) \\ 0 & \text{otherwise} \end{cases}.$

The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!



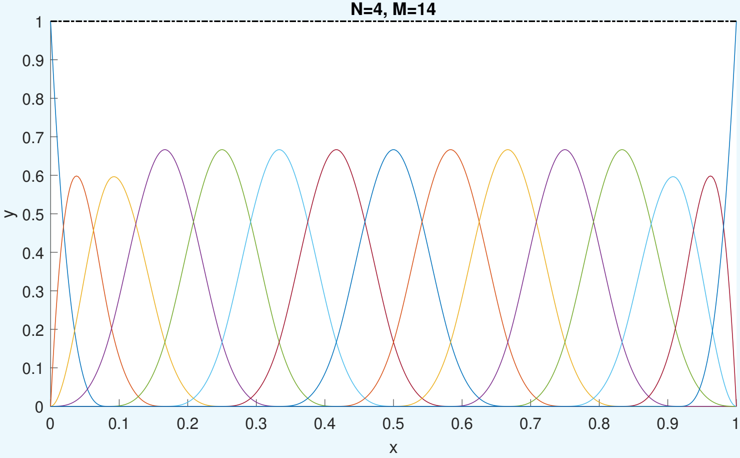
$$B_{m,N}(x) = 0, \qquad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty).$$

The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!



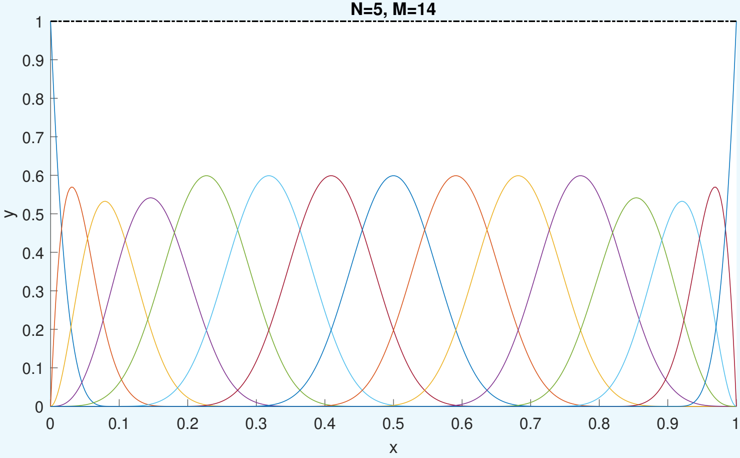
$$B_{m,N}(x) = 0, \qquad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty).$$

The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!



$$B_{m,N}(x) = 0, \qquad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty).$$

The B-spline basis elements $B_{m,N}(\cdot)$ are compactly supported!



$$B_{m,N}(x) = 0, \qquad x \in (-\infty, t_m) \cup [t_{m+N+1}, \infty).$$

The B-spline function

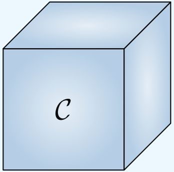
Any continuous function can be approximated arbitrarily well by

$$S(x) = \begin{bmatrix} B_{0,N}(x) & \cdots & B_{M,N}(x) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_M \end{bmatrix} = B_{\mathscr{T},N}(x)\boldsymbol{c}.$$

through either *increasing* the knot density and order of the spline.

Taking direct tensor products of splines leads to exponential blow-up of coefficients...

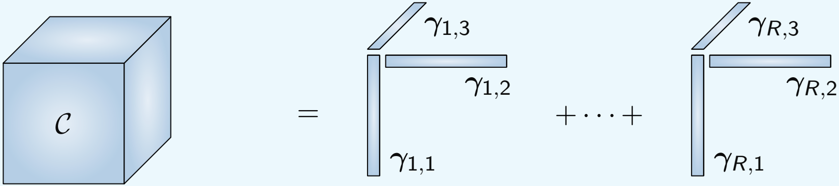
$$\hat{f}(x;\mathcal{C}) = \sum_{m_1=0}^{M_1} \cdots \sum_{m_D=0}^{M_D} c_{m_1 \cdots m_D} \prod_{d=1}^{D} B^{(d)}_{m_d, N^{(d)}}(x_d) = \mathcal{C} \bullet_1 B_d(x_1) \cdots \bullet_D B_D(x_D)$$



$\prod_{d=1}^{D}(M_d + 1)$ parameters

Exploit low-rank structure: $\mathcal{C}(\Gamma_1, \ldots, \Gamma_d) = [\![\Gamma_1, \ldots, \Gamma_D]\!]$, to alleviate this blow-up!

$$\hat{f}(x; \Gamma_1, \ldots, \Gamma_D) = \mathcal{C}(\Gamma_1, \ldots, \Gamma_D) \bullet_1 B_1(x_1) \cdots \bullet_D B_D(x_D) = \sum_{r=1}^{R} \prod_{d=1}^{D} B_d(x_d) \gamma_{r,d}$$



$\prod_{d=1}^{D}(M_d + 1)$ parameters

$R(\sum_{d=1}^{D} M_d + 1)$ parameters

Regression is performed by minimizing the quadratic objective function

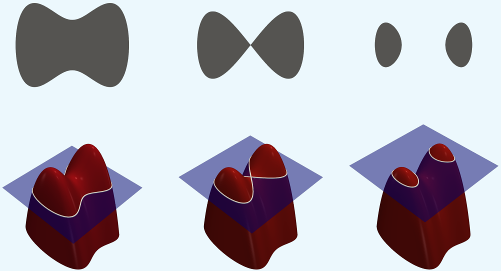
Given samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{I} \subset [0,1]^D \times \mathbb{R}$ from a underlying target function $f \in C([0,1]^D)$, we minimize:

$$Q(\Gamma_1, \ldots, \Gamma_D) := \frac{1}{2} \sum_{i=1}^{I} \left(\hat{f}(x_i; \Gamma_1, \ldots, \Gamma_D) - y_i \right)^2 .$$

A level-set approach to modeling a binary classification function

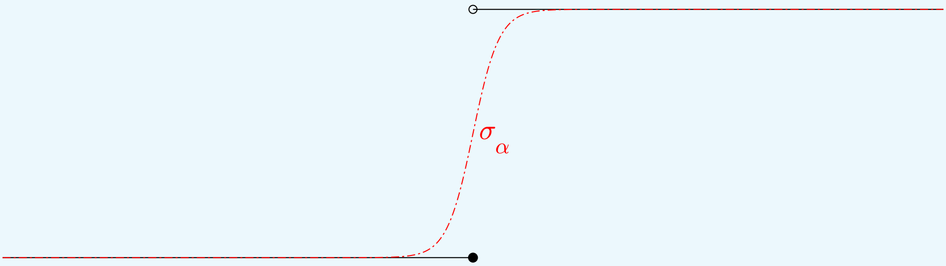
Binary classification function $g : [0,1]^D \to \{0,1\}$ can be modeled by the function

$$g(x) = \begin{cases} 0 & f(x) \leq 0 \\ 1 & f(x) > 0 \end{cases}$$



(copyright wikimedia)

Replace step function with the logistic function $\sigma_\alpha : t \mapsto 1/(\exp(-\alpha t) + 1)$

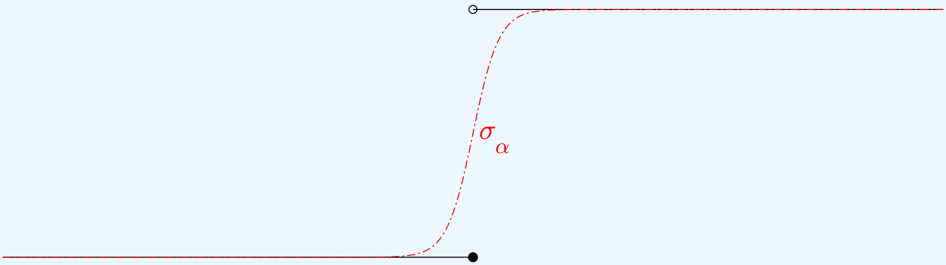


Replace g with

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

where $\alpha > 0$ controls sharpness of transition.

Replace step function with the logistic function $\sigma_\alpha : t \mapsto 1/(\exp(-\alpha t) + 1)$



g_α is further replaced by the approximant

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

Classification is performed by minimizing the Logistic objective function

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 when both

$$\prod_{y_i=0} (1 - \hat{g}_\alpha(x_i; \Gamma_1, \ldots, \Gamma_D)) \quad \text{and} \quad \prod_{y_i=1} \hat{g}_\alpha(x_i; \Gamma_1, \ldots, \Gamma_D)$$

is maximized as much as possible.

Classification is performed by minimizing the Logistic objective function

This is equivalent to minimizing the objective function

$$L_\alpha(\Gamma_1, \dots, \Gamma_D) := -\sum_{i=1}^{I} y_i \log \hat{g}_\alpha(x_i; \Gamma_1, \dots, \Gamma_D) + (1 - y_i) \log \left(1 - \hat{g}_\alpha(x_i; \Gamma_1, \dots, \Gamma_D)\right).$$

Minimization of objective functions is effectively done with Gauss-Newton dogleg algorithm

- Exploit multi-linear structure of the objective functions, see:
  - *Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization.* Sorber et al., SIAM J. Optim., 2013.
  - *Numerical optimization-based algorithms for data fusion.* Vervliet et al., Data Handling in Science and Technology, 2019.
- 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!

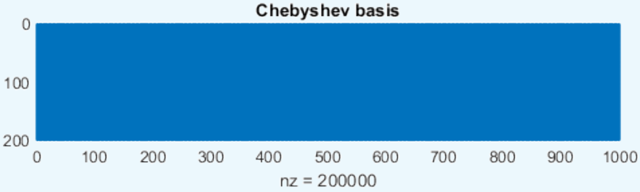
Gradient:

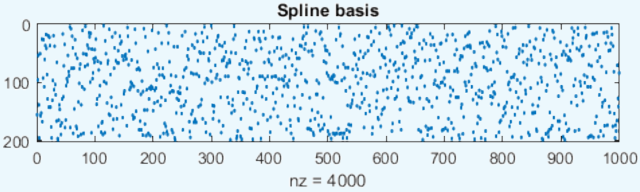
$$g_{r,d} = A_d \left(\left(\mathop{\Huge *}_{k=1, k \neq d}^{D} A_k^\top \gamma_{r,k} \right) * \boldsymbol{\eta} \right).$$

Grammian (of the Jacobian) vector product

$$w_{r,d} = A_d \left(\left(\mathop{\Huge *}_{k=1, k \neq d}^{D} A_k^\top \gamma_{r,k} \right) * \boldsymbol{\xi} * \left(\sum_{\tilde{d}=1}^{D} \sum_{\tilde{r}=1}^{R} \left(\mathop{\Huge *}_{k=1, k \neq d}^{D} A_k^\top \gamma_{\tilde{r},k} \right) * A_{\tilde{d}}^\top z_{\tilde{r},\tilde{d}} \right) \right).$$

Benefit of compactly supported B-splines: significant speed-ups in Grammian and gradient by exploiting sparsity!

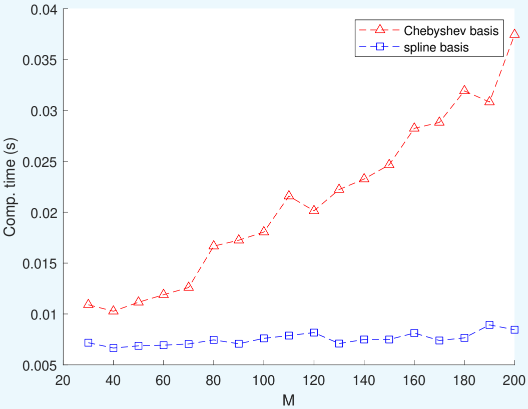




If the order of the B-spline is kept low:

$$\mathcal{O}\left(DIMR\right) \quad \rightarrow \quad \mathcal{O}\left(DIR\right) \text{ flops}$$

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.

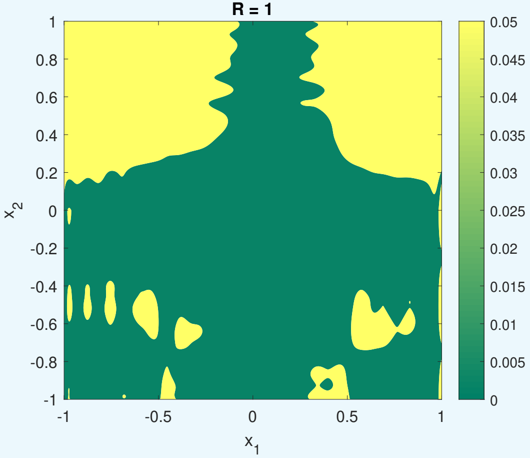
$N = 4,\ R = 3,\ I = 1000.$

A $R = 3$ separable function

Consider the following example

$$f(\boldsymbol{x}) = \underbrace{|x_1||x_2|}_{\text{non-smooth term}} + \sin(2\pi x_1)\cos(2\pi x_2) + x_1^2 x_2, \quad \boldsymbol{x} \in [-1,1] \times [-1,1].$$

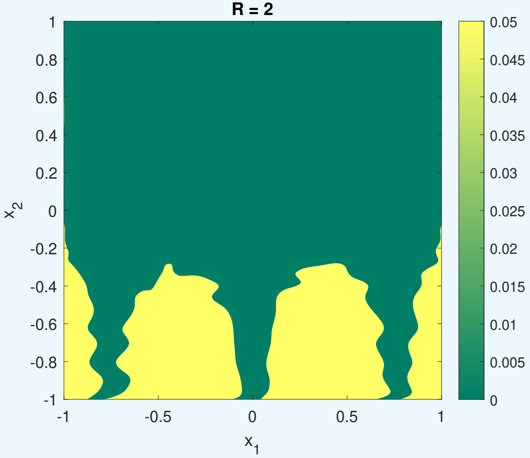
As expected... an $R = 3$ is sufficient for a good approximation



Absolute error

(Knots are uniformly distributed on the approximation domain)

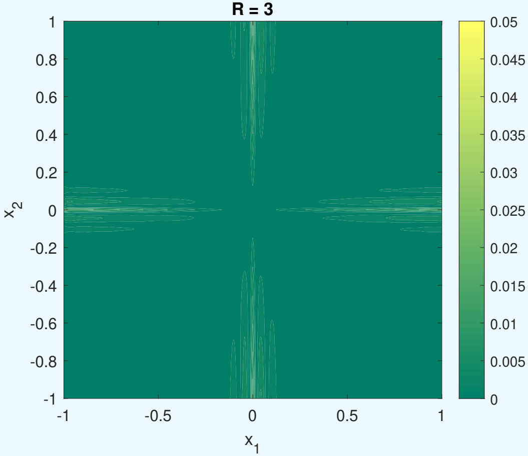
As expected... an $R = 3$ is sufficient for a good approximation



Absolute error

(Knots are uniformly distributed on the approximation domain)

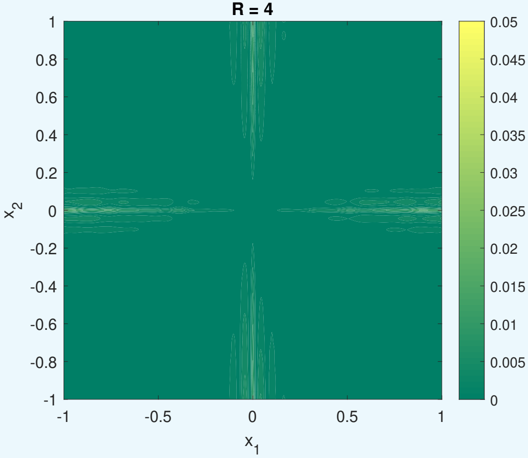
As expected... an $R = 3$ is sufficient for a good approximation



Absolute error

(Knots are uniformly distributed on the approximation domain)

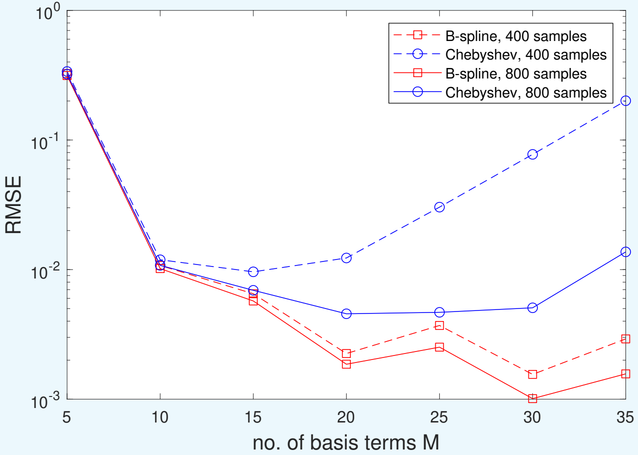
As expected... an $R = 3$ is sufficient for a good approximation



Absolute error

(Knots are uniformly distributed on the approximation domain)

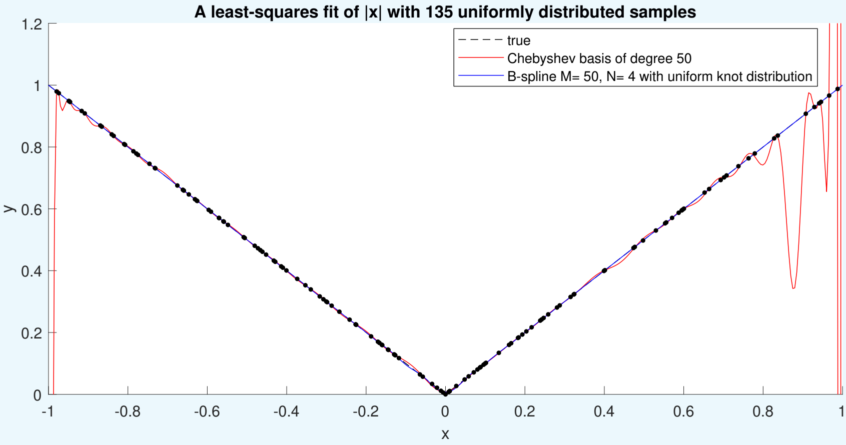
Unlike for splines, Runge's phenomenon can adversely affect quality of approximation



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*



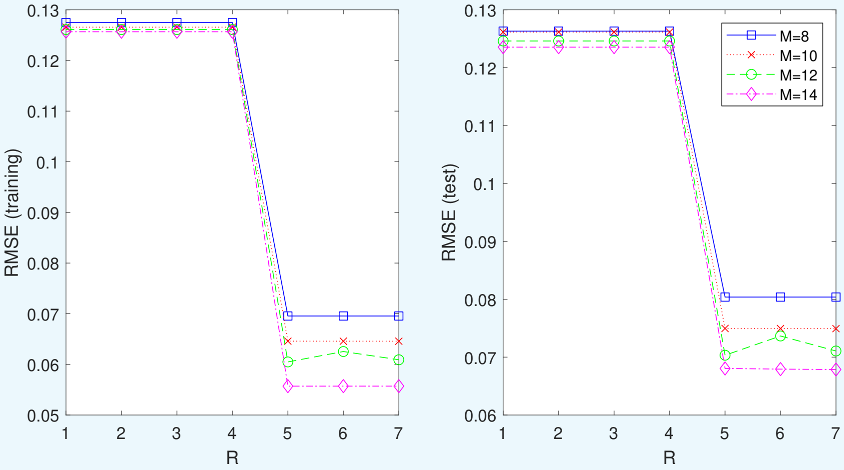
Runge's phenomenon can adversely contribute to the overfitting problem

Low-rank structures in real life datasets - an example

NASA dataset from the UCI machine learning repository:

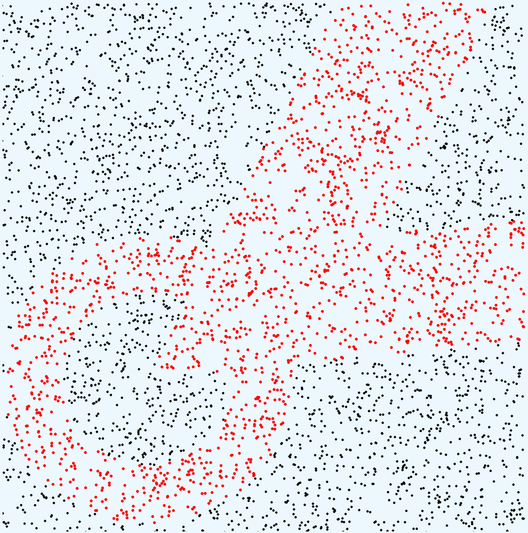
- independent variables:
  - *frequency*,
  - *angle of attack*,
  - *chord length*,
  - *free-stream velocity*,
  - *suction-side displacement thickness*.
- dependent variable: *self-noise generated by airfoil*.
- randomly split data into a training (1202 samples) and a test (301 samples) sets.

An $R = 5$ separable function is sufficient to model the NASA dataset

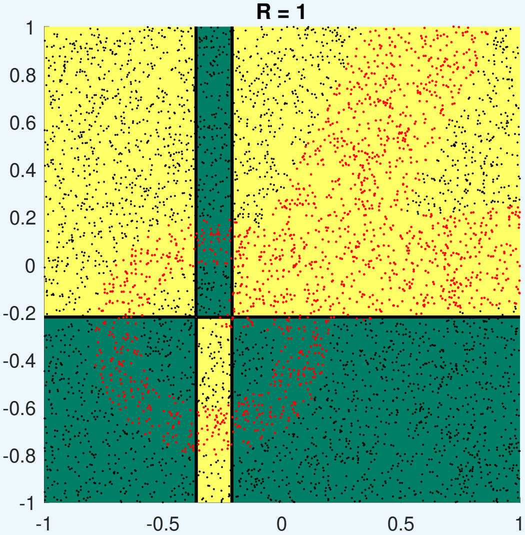


The separable rank can be increased to account for complexity of the classification sets

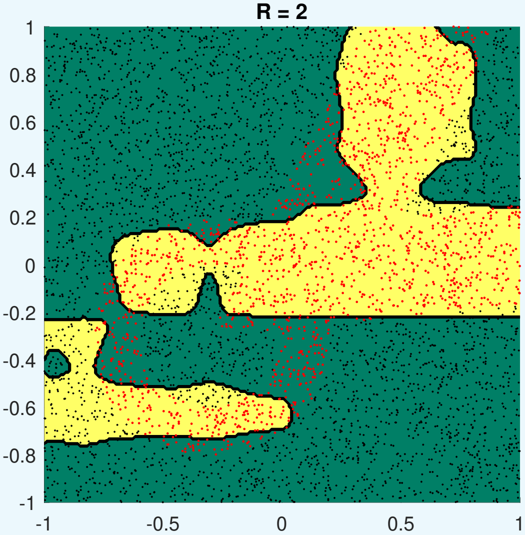
Consider the labeled dataset:



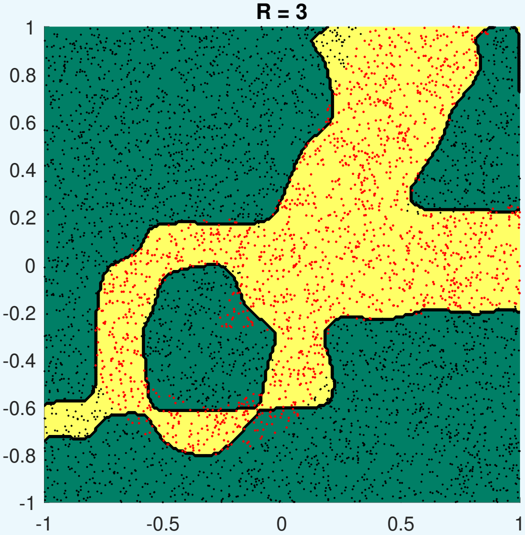
The separable rank can be increased to account for complexity of the classification sets



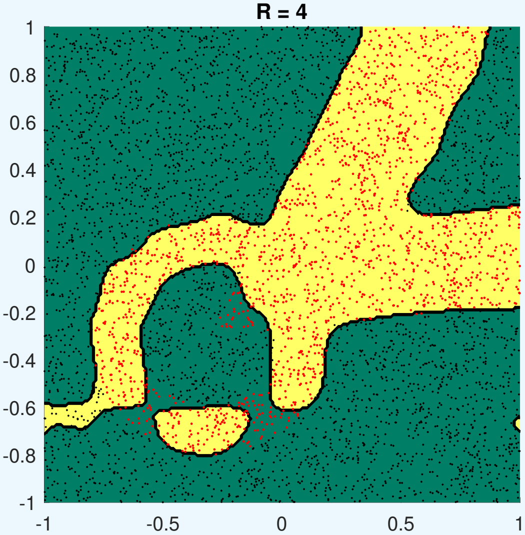
The separable rank can be increased to account for complexity of the classification sets



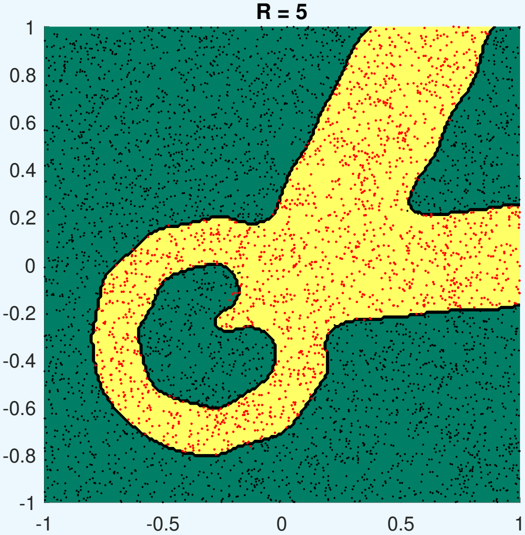
The separable rank can be increased to account for complexity of the classification sets



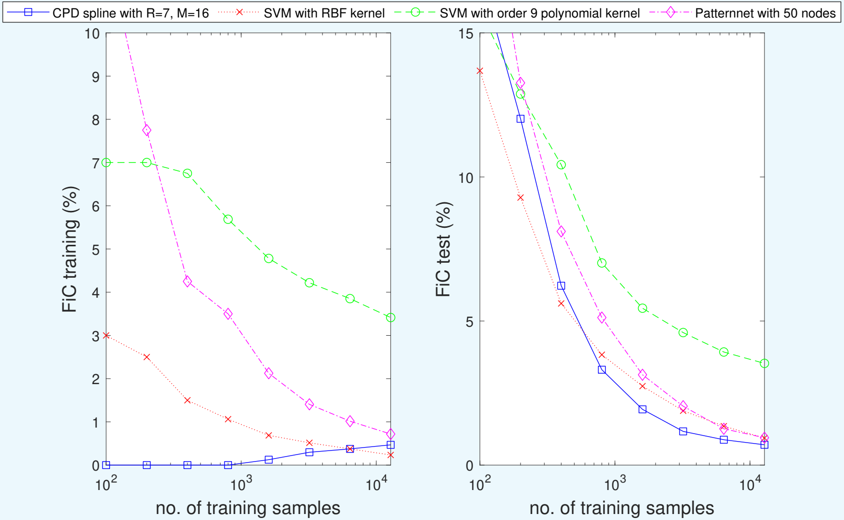
The separable rank can be increased to account for complexity of the classification sets



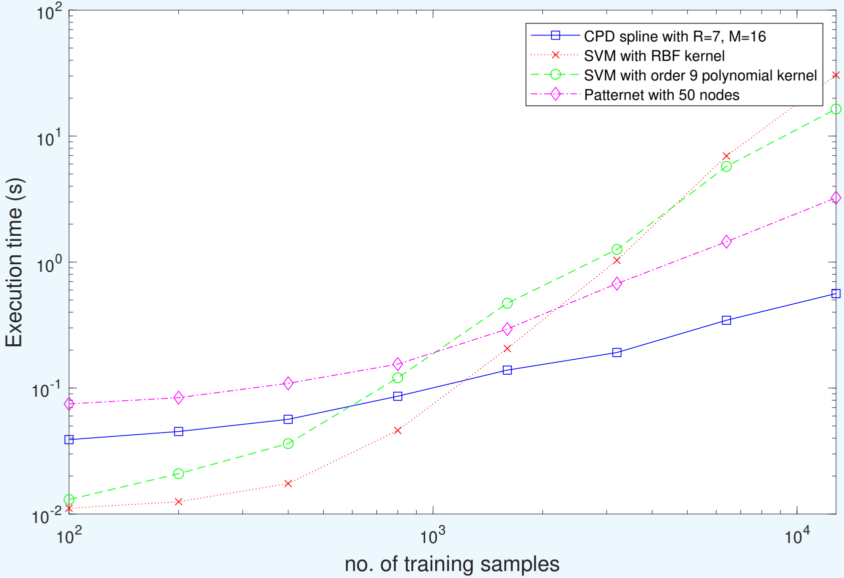
The separable rank can be increased to account for complexity of the classification sets



Our method compared with well-established techniques for classification



CPU time for training grows more moderately with dataset size



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