Sparse regularization for low-rank regression

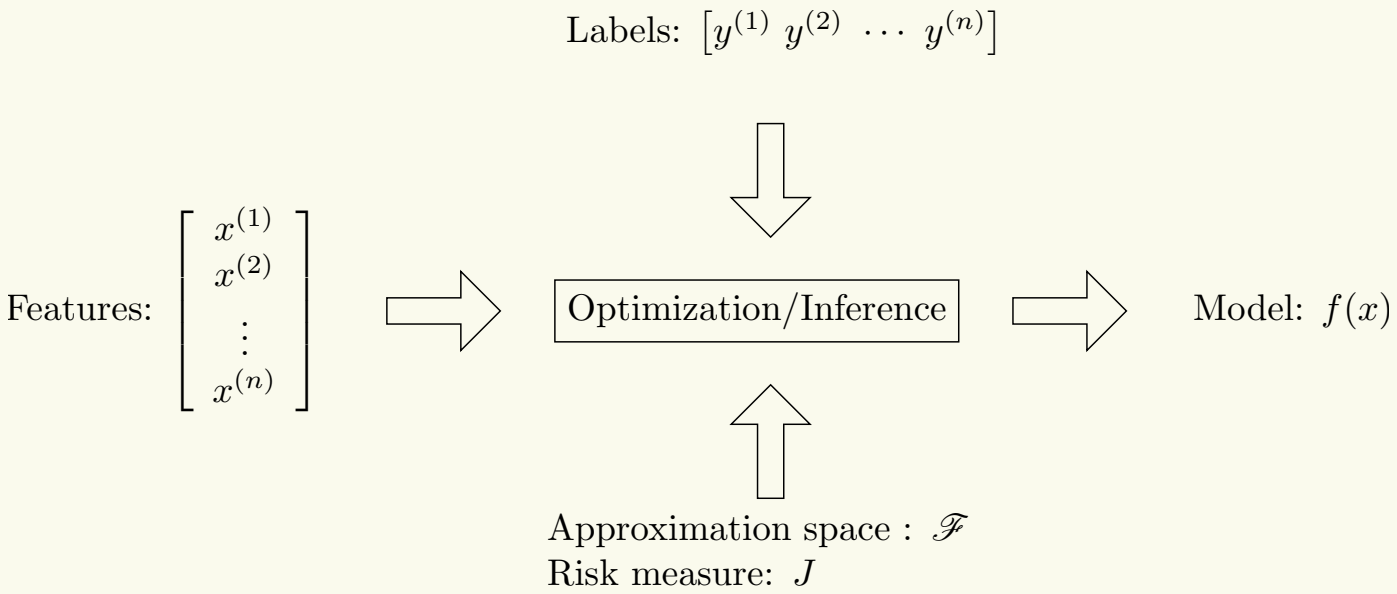
SIAM Conference on Uncertainty Quantification, Orange County, USA
April 15–19, 2018

Alex Gorodetsky[†] and John Jakeman[+]

[†] University of Michigan, [+] Sandia National Laboratories

Sandia
National
Laboratories

Problem: Supervised learning



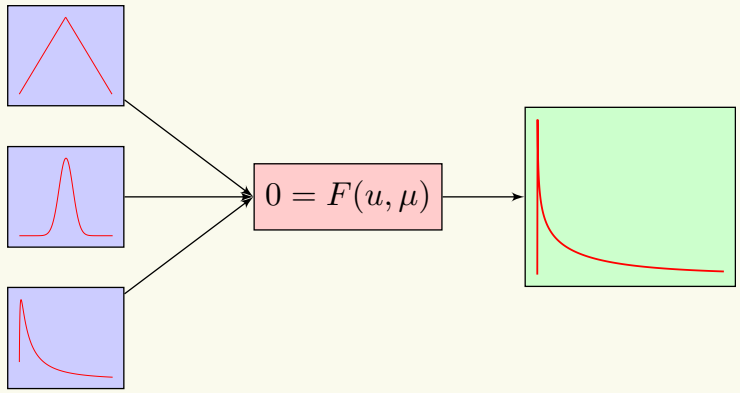
- Build data-fit models to (noisy) data
- Parametric approaches for $\mathscr{F}$:
  - Good: statistical properties, stability
  - Bad: curse-of-dimensionality on basis, expressivity
- Nonparametric approaches for $\mathscr{F}$:
  - Good: expressivity
  - Bad: excess risk scales exponentially with dimensionality, expensive

Supervised learning in UQ

This presentation

Build data-based model with no control over sampling

- Uses
  - Generating surrogates: replace model within analysis
  - Enhancing convergence, e.g., within MLMC
  - Representing random variables
- Challenges
  - Large dimensional input spaces
  - Small amounts of data
  - Capturing high-order behavior



Parametric approaches

- Approximation space: tensor-product basis expansions $\{\psi_i\}_{i=1}^{P}$

$$f(x) = \sum_{i=1}^{P} \boldsymbol{\theta}_i \psi_i(x) = \sum_{i_1=1}^{P_1} \dots \sum_{i_d=1}^{P_d} \boldsymbol{\theta}_{i_1 \dots i_d} \phi_{1 i_1}(x_1) \dots \phi_{d i_d}(x_d)$$

Parametric approaches

- Approximation space: tensor-product basis expansions $\{\psi_i\}_{i=1}^{P}$

$$f(x) = \sum_{i=1}^{P} \boldsymbol{\theta}_i \psi_i(x) = \sum_{i_1=1}^{P_1} \dots \sum_{i_d=1}^{P_d} \boldsymbol{\theta}_{i_1 \dots i_d} \phi_{1 i_1}(x_1) \dots \phi_{d i_d}(x_d)$$

- Many algorithmic setups
  - Ridge regression:
    $$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda \|\boldsymbol{\theta}\|_2$$
  - $L1$ regression:
    $$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda \|\boldsymbol{\theta}\|_1$$
  - $L0$ regression:
    $$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda \|\boldsymbol{\theta}\|_0$$
  - Low-rank regression: Assume reduced-order structure on $\boldsymbol{\theta}$

Parametric approaches

- Approximation space: tensor-product basis expansions $\{\psi_i\}_{i=1}^{P}$

$$f(x) = \sum_{i=1}^{P} \boldsymbol{\theta}_i \psi_i(x) = \sum_{i_1=1}^{P_1} \dots \sum_{i_d=1}^{P_d} \boldsymbol{\theta}_{i_1 \dots i_d} \phi_{1 i_1}(x_1) \dots \phi_{d i_d}(x_d)$$

- Many algorithmic setups
  - Ridge regression:
  $$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda \|\boldsymbol{\theta}\|_2$$
  - $L1$ regression:
  $$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda \|\boldsymbol{\theta}\|_1$$
  - $L0$ regression:
  $$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda \|\boldsymbol{\theta}\|_0$$
  - Low-rank regression: Assume reduced-order structure on $\boldsymbol{\theta}$
- Constructing Ψ is problematic in high-dimensions
  - Total order expansions: exponential growth of terms
  - Low order expansion: limited expressivity

Outline

1. Modeling formulation
   - Functional tensor-train decomposition
   - Relation to other low-rank functional decompositions
   - Discussion of sparsity
2. Optimization and rank adaptation
   - Derivation of efficient gradient
   - Rank adaptation
3. Numerical results
   - Synthetic problem
   - 22 data sets from UCI

Outline

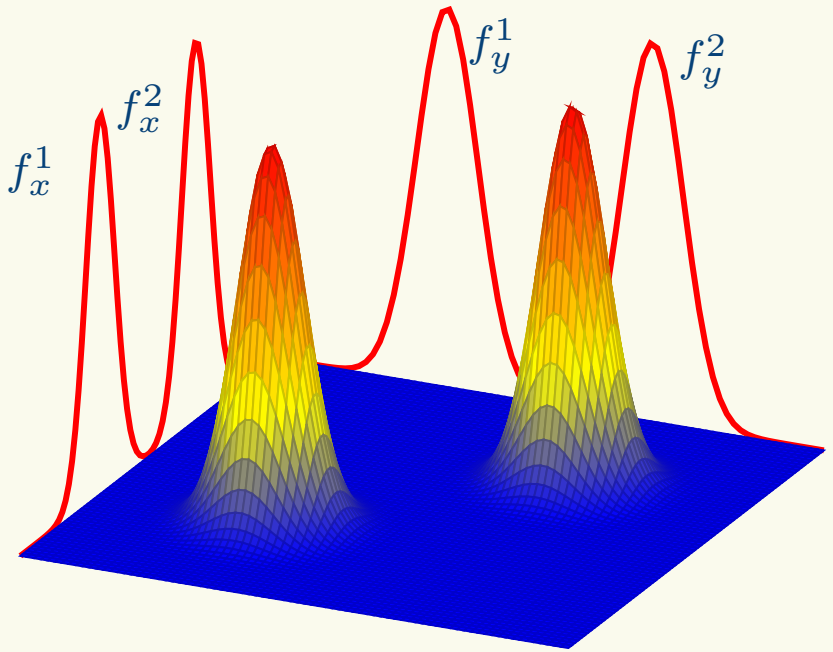
1. Modeling formulation
   - Functional tensor-train decomposition
   - Relation to other low-rank functional decompositions
   - Discussion of sparsity
2. Optimization and rank adaptation
   - Derivation of efficient gradient
   - Rank adaptation
3. Numerical results
   - Synthetic problem
   - 22 data sets from UCI

Takeaway: Formulating a low-rank regression problem with nonlinear least squares and sparse regularization and solving it with gradient-based methods is effective in high-dimensions in the face of large polynomial orders and ranks

Low-rank compression of functions

Main idea

Represent a few univariate functions instead of multivariate functions

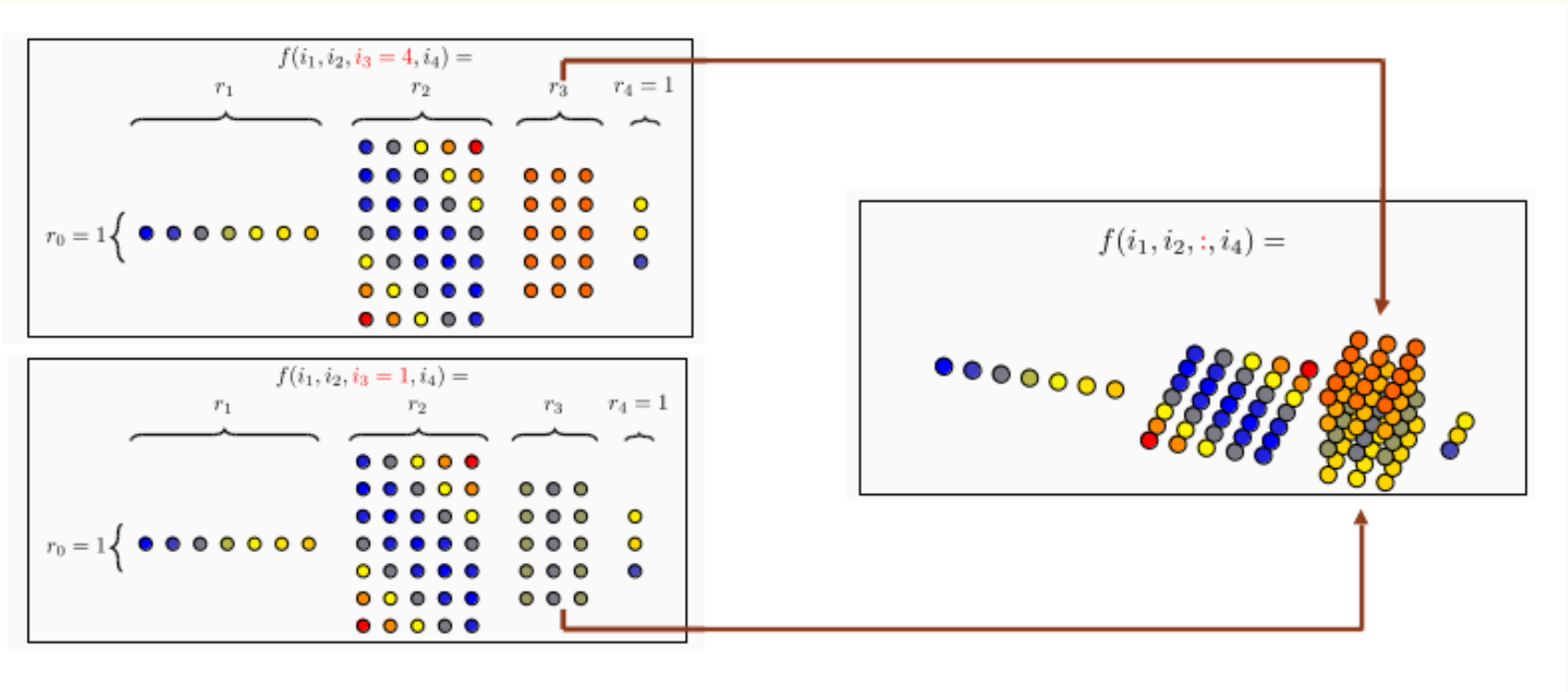


$$f(x,y) = f_x^1(x)f_y^1(y) + f_x^2(x)f_y^2(y)$$

Model format: tensor-train (Oseledets 2011)

- TT decomposition provides compression for multiway arrays
  - Existence of best approximation guaranteed
  - Storage scales linearly with dimension and polynomially with rank
- TT-ranks are related to the ranks of *reshapings* of a tensor

$$r_k \leq \text{rank} f(\underbrace{i_1, \ldots, i_k}_{\boldsymbol{i}_{\leq k}}; \underbrace{i_{k+1}, \ldots, i_d}_{\boldsymbol{i}_{>k}})$$

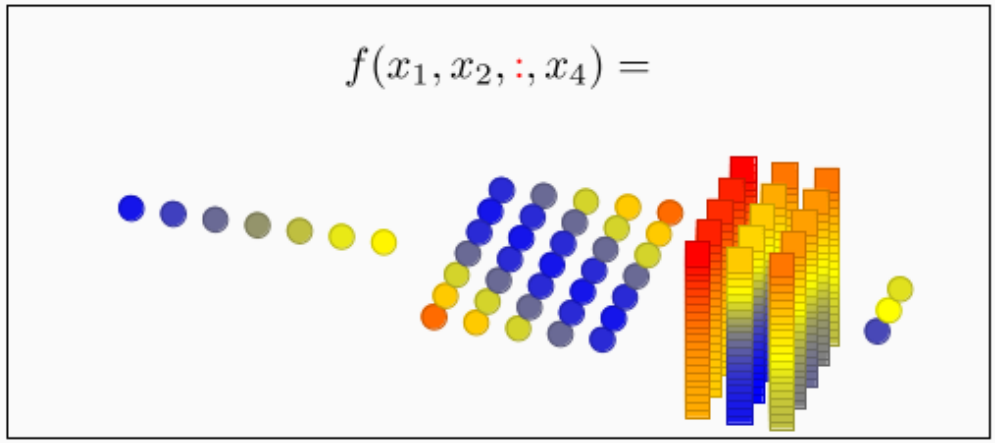


Functional tensor-train
Tensor-train as a functional ansatz

- Approximate multivariate functions instead of multiway arrays
- Adapt to local and global structure
- Evaluation through products of matrix-valued functions

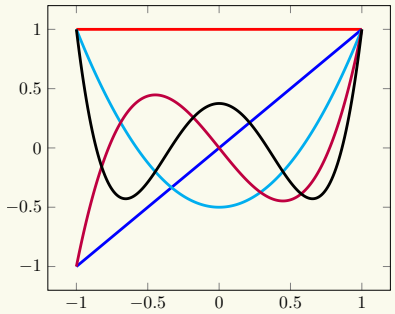
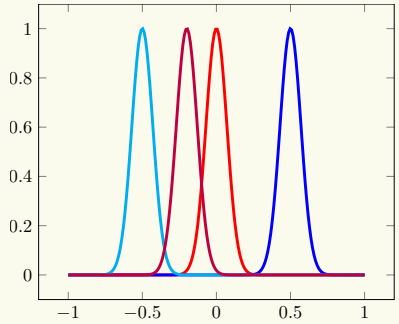
$$f(x_1, x_2, \ldots, x_d) = \sum_{i_0=1}^{r_0} \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} f_1^{(i_0 i_1)}(x_1) f_2^{(i_1 i_2)}(x_2) \ldots f_d^{(i_{d-1} i_d)}(x_d)$$
$$= \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d)$$

$$\underbrace{\begin{bmatrix} f_k^{(11)}(x_k) & \cdot & f_k^{(1 r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1} 1)}(x_k) & \cdot & f_k^{(r_{k-1} r_k)}(x_k) \end{bmatrix}}_{\mathcal{F}_k(x_k)}$$



Oseledets 2013, Gorodetsky, Karaman, Marzouk 2015

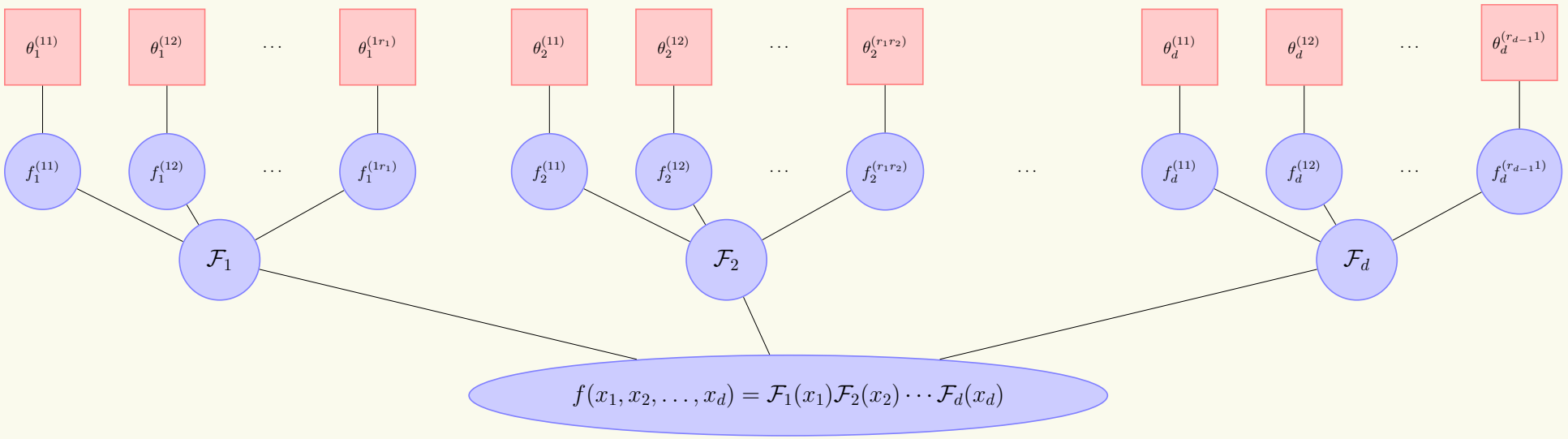
Linear and nonlinear parameterizaitons

	Linear	Nonlinear
		
Evaluation	$\sum_{\ell=1}^{p_{kij}} \theta_{kij\ell} \phi_{k\ell}^{(ij)}(x_k)$	$\sum_{\ell=1}^{p_{kij}} \exp\left((x_k - \theta_{kij\ell})^2\right)$
Gradient	$\phi_{k\ell}^{(ij)}(x_k)$	$-2\,(x_k - \theta_{kij\ell}) \exp\left((x_k - \theta_{kij\ell})^2\right)$

- Linear: gradient independent of parameter
  $\rightarrow$ faster optimization, lower expressivity
- Nonlinear: gradient dependent on parameters
  $\rightarrow$ slower optimization, higher expressivity

FT storage scheme

- Store parameters and univariate representation separately
- No implicit or explicit tensors
- Tensor-product structure is imposed on the **univariate functions** themselves.



Special case: recovering existing models
Or — converting to low-rank tensors

UNIVERSITY OF MICHIGAN

Assumption

- *linear parameterization of univariate functions*
- *identitical basis expansions within a core, i.e.,* $p_{kij} = p_k$ *and* $\phi_{k\ell}^{(ij)} = \phi_{k\ell}$

- Obtain identical formats to Doostan et al., 2013, Mathelin 2014, Chevreuil et al., 2015.
- Define a TT-core $\boldsymbol{\mathcal{F}}_k \in \mathbb{R}^{r_{k-1} \times p_k \times r_k}$ such that

$$\boldsymbol{\mathcal{F}}_k[:, \ell, :] = \begin{bmatrix} \theta_{k11\ell} & \cdot & \theta_{k1r_k\ell} \\ \vdots & \ddots & \vdots \\ \theta_{kr_{k-1}1\ell} & \cdot & \theta_{kr_{k-1}r_k\ell} \end{bmatrix}$$

- TT-core and FT-core related according to

$$\mathcal{F}_k(x_k) = \sum_{\ell=1}^{p_k} \boldsymbol{\mathcal{F}}_k[:, \ell, :] \phi_{k\ell}(x_k).$$

- TT-approximation of coefficients of a tensor-product of basis functions

$$f(x_1, \ldots, x_d) = \sum_{\ell_1=1}^{p_1} \cdots \sum_{\ell_d=1}^{p_d} \boldsymbol{\mathcal{F}}_1[:, \ell_1, :] \cdots \boldsymbol{\mathcal{F}}_d[:, \ell_d, :] \phi_{1\ell_1}(x_1) \cdots \phi_{d\ell_d}(x_d).$$

Sparsity in FT format

- Example: Rank-2 additive function

$$f(x_1, \ldots, x_d) = f_1(x_1) + \cdots + f_d(x_d)$$

- FT format:

$$f(x_1, x_2, \ldots, x_d) = [f_1(x_1)\ 1] \begin{bmatrix} 1 & 0 \\ f_2(x_2) & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 \\ f_d(x_d) \end{bmatrix}$$

- TT storage requirement: $4p(d-2)$ floats
- FT storage requirement: $d(p+3)-4$ floats
- In the limit, FT requires almost 4 times less storage
- Difference is more striking for higher order interactions

Example 2: Quadratic

- Quadratic $x^T \mathbf{A} x$, $a_{ij} = \mathbf{A}[i, j]$
- $\mathcal{O}(d^3)$ storage in TT format
- $\mathcal{O}(d^2)$ if sparsity taken into account

$$\mathcal{F}_k(x_k) = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ x_k & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & & & \ddots & \ddots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ a_{kk}x_k^2 & a_{kk+1}x_k & a_{kk+2}x_k & \cdots & a_{kd-1}x_k & a_{kd}x_k & 1 \end{bmatrix}$$

Variable ordering as a sparsity problem

$$f(x_1, x_2, x_3) = \sum_{j=1}^{P-1} x_1^j x_3^j + x_2$$

Variable ordering as a sparsity problem

$$f(x_1, x_2, x_3) = \sum_{j=1}^{P-1} x_1^j x_3^j + x_2$$

- Direct ranks = $[1,\ P,\ P,\ 1] \rightarrow P + P^2 + P$ elements

$$f(x_1, x_2, x_3) = \begin{bmatrix} x_1 & x_1^2 & \dots & x_1^{P-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \cdots & 0 & 0 \\ & & & \\ 0 & \ddots & 0 & \vdots \\ 0 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & x_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_3^2 \\ \vdots \\ x_3^{P-1} \\ 1 \end{bmatrix}$$

Variable ordering as a sparsity problem

$$f(x_1, x_2, x_3) = \sum_{j=1}^{P-1} x_1^j x_3^j + x_2$$

- Direct ranks = $[1,\ P,\ P,\ 1] \rightarrow P + P^2 + P$ elements

$$f(x_1, x_2, x_3) = \begin{bmatrix} x_1 & x_1^2 & \dots & x_1^{P-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \cdots & 0 & 0 \\ & & & \\ 0 & \ddots & 0 & \vdots \\ 0 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & x_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_3^2 \\ \vdots \\ x_3^{P-1} \\ 1 \end{bmatrix}$$

- Sparse storage = $P + P + P$ elements

Variable ordering as a sparsity problem

$$f(x_1, x_2, x_3) = \sum_{j=1}^{P-1} x_1^j x_3^j + x_2$$

- Direct ranks = $[1,\ P,\ P,\ 1] \rightarrow P + P^2 + P$ elements

$$f(x_1, x_2, x_3) = \begin{bmatrix} x_1 & x_1^2 & \dots & x_1^{P-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \cdots & 0 & 0 \\ & & & \\ 0 & \ddots & 0 & \vdots \\ 0 & \cdots & 1 & 0 \\ 0 & \cdots & 0 & x_2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_3^2 \\ \vdots \\ x_3^{P-1} \\ 1 \end{bmatrix}$$

- Sparse storage = $P + P + P$ elements
- Ordered ranks = $[1,\ P,\ 2,\ 1] \rightarrow P + 2P + 2$ elements

$$f(x_1, x_3, x_2) = \begin{bmatrix} x_1 & x_1^2 & \dots & x_1^{P-1} & 1 \end{bmatrix} \begin{bmatrix} x_3 & 0 \\ x_3^2 & 0 \\ \vdots & \vdots \\ x_3^{P-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \end{bmatrix}$$

Constrained least squares

$$f^* = \arg\min_{f \in \mathscr{F}_{\boldsymbol{r}}} \quad \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2$$

$$\mathscr{F}_{\boldsymbol{r}} = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \begin{array}{l} f = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d) \\ \mathcal{F}_1 : \mathcal{X}_1 \to \mathbb{R}^{1\times r_1} \\ \mathcal{F}_k : \mathcal{X}_k \to \mathbb{R}^{r_{k-1}\times r_k},\ 2 \le k < d \\ \mathcal{F}_d : \mathcal{X}_d \to \mathbb{R}^{r_{d-1}\times 1} \end{array} \right\}$$

- Function space $\mathscr{F}_{\boldsymbol{r}}$ includes functions with ranks $< \boldsymbol{r}$.
  - Obtained by constraining certain blocks of $\mathcal{F}_k$ to be zero.
- Gradient based procedure for optimization
  - Alternating least squares
  - Quasi-Newton methods
  - Riemannian manifold

Constrained least squares

$$f^* = \arg\min_{f \in \mathscr{F}_{\boldsymbol{r}}} \quad \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2 + \lambda\Omega[f]$$

$$\mathscr{F}_{\boldsymbol{r}} = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \begin{array}{l} f = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d) \\ \mathcal{F}_1 : \mathcal{X}_1 \to \mathbb{R}^{1\times r_1} \\ \mathcal{F}_k : \mathcal{X}_k \to \mathbb{R}^{r_{k-1}\times r_k},\ 2 \le k < d \\ \mathcal{F}_d : \mathcal{X}_d \to \mathbb{R}^{r_{d-1}\times 1} \end{array} \right\}$$

- Function space $\mathscr{F}_{\boldsymbol{r}}$ includes functions with ranks $< \boldsymbol{r}$.
  - Obtained by constraining certain blocks of $\mathcal{F}_k$ to be zero.
- Gradient based procedure for optimization
  - Alternating least squares
  - Quasi-Newton methods
  - Riemannian manifold

Desired and actual sparsity regularization

- Structure of FT-cores admits natural grouping in terms of univariate functions
- Setting functions to zero lowers number of interactions
- Want to minimize number of nonzero functions: "L_0"-type penalty
- Instead, pose an "L_1"-type penalty, sum of magnitudes
- Measure magnitude of function with "L_2"

$$\Omega[f] = \sum_{k=1}^{d} \sum_{i=1}^{r_{k-1}} \sum_{j=1}^{r_k} \| f_k^{(ij)} \|^2$$

Similar to Mathelin 2014 and Rai 2014, where sparsity of the TT cores was sought in each step of ALS

Gradient computation for single core

UNIVERSITY OF MICHIGAN

Partial Derivative: $\frac{\partial f(x;\boldsymbol{\theta})}{\partial \theta_{\boldsymbol{i}}} = \mathcal{F}_{<k}(x_{<k}) \frac{\partial_{\boldsymbol{i}} \mathcal{F}_k(x_k)}{\partial \theta_{\boldsymbol{i}}} \mathcal{F}_{>k}(x_{>k})$

Evaluation of left cores: $\mathcal{F}_{<k}(x_{<k}) = \mathcal{F}_1(x_1) \cdots \mathcal{F}_{k-1}(x_{k-1})$

Evaluation of right cores: $\mathcal{F}_{>k}(x_{>k}) = \mathcal{F}_{k+1}(x_{k+1}) \cdots \mathcal{F}_d(x_d)$

Gradient computation for single core

UNIVERSITY OF MICHIGAN

Partial Derivative: $\frac{\partial f(x;\boldsymbol{\theta})}{\partial \theta_{\boldsymbol{i}}} = \mathcal{F}_{<k}(x_{<k}) \frac{\partial_{\boldsymbol{i}} \mathcal{F}_k(x_k)}{\partial \theta_{\boldsymbol{i}}} \mathcal{F}_{>k}(x_{>k})$

Evaluation of left cores: $\mathcal{F}_{<k}(x_{<k}) = \mathcal{F}_1(x_1) \cdots \mathcal{F}_{k-1}(x_{k-1})$

Evaluation of right cores: $\mathcal{F}_{>k}(x_{>k}) = \mathcal{F}_{k+1}(x_{k+1}) \cdots \mathcal{F}_d(x_d)$

Assumption

Each univariate function in each FT-core is independently parameterized.

Proposition: Partial derivative of core $\mathcal{F}_k$

The partial derivative of core $\mathcal{F}_k$ with respect to $\theta_{kij\ell}$ is a *sparse* matrix

$$\mathbf{G}[\alpha, \beta] = \begin{cases} \frac{\partial f_k^{(ij)}(x_k)}{\partial \theta_{kij\ell}} & \text{if } \alpha = i, \beta = j \\ 0 & \text{otherwise} \end{cases}$$

for $\alpha = 1, \ldots, r_{k-1}$, and $\beta = 1, \ldots r_k$.

Given $\mathcal{F}_{<k}(x_{<k})$, $\mathcal{F}_{>k}(x_{>k})$, and $\frac{\partial f_k^{(ij)}(x_k)}{\partial \theta_{kij\ell}}$, derivative requires $\rightarrow \mathcal{O}(1)$ ops.

Computational complexity for single core

- Let $\boldsymbol{a} = \mathcal{F}_{<k}(x_{<k})$ and $\boldsymbol{c} = \mathcal{F}_{>k}(x_{>k})$.
- Then $\partial_{\boldsymbol{i}} f(x;\boldsymbol{\theta}) = \boldsymbol{a}^T \partial_{\boldsymbol{i}} \mathcal{F}_k(x_k)\boldsymbol{c}$
- Let $G(p)$ denote the number of ops. to evaluate the gradient of $f_k^{(ij)}$ with respect to *all* parameters.

Complexity

Computing, $\partial_{\boldsymbol{i}} f(x;\boldsymbol{\theta})$ for all $\boldsymbol{i}$ within the kth core requires

$$\mathcal{O}\left(\left(G(p)+p\right) r_{k-1} r_k\right)$$

operations.

FT evaluation and gradient algorithm
Forward sweep

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\boldsymbol{a} = \mathbf{F}_1\mathcal{F}(x_1)$	$E(p)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{\theta}_{1ij\ell}}$	$G(p)r_1$	pr_1

FT evaluation and gradient algorithm
Forward sweep

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\boldsymbol{a} = \mathbf{F}_1\mathcal{F}(x_1)$	$E(p)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{\theta}_{1ij\ell}}$	$G(p)r_1$	pr_1
Stage 2	Eval	$\mathbf{F}_2 = \mathcal{F}(x_2)$	$E(p)r_1r_2$	r_1r_2
	Up. part. deriv	$\partial f_{2ij\ell} = \boldsymbol{a}[i]\frac{\partial f_2^{(ij)}}{\partial \boldsymbol{\theta}_{2ij\ell}}$	$G(p)r_1r_2$	pr_1r_2
	Update	$\boldsymbol{a} \leftarrow \boldsymbol{a}\mathbf{F}_2$	r_1r_2	-

FT evaluation and gradient algorithm
Forward sweep

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\boldsymbol{a} = \mathbf{F}_1\mathcal{F}(x_1)$	$E(p)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{\theta}_{1ij\ell}}$	$G(p)r_1$	pr_1
Stage 2	Eval	$\mathbf{F}_2 = \mathcal{F}(x_2)$	$E(p)r_1r_2$	r_1r_2
	Up. part. deriv	$\partial f_{2ij\ell} = \boldsymbol{a}[i]\frac{\partial f_2^{(ij)}}{\partial \boldsymbol{\theta}_{2ij\ell}}$	$G(p)r_1r_2$	pr_1r_2
	Update	$\boldsymbol{a} \leftarrow \boldsymbol{a}\mathbf{F}_2$	r_1r_2	-
Stage k	Eval	$\mathbf{F}_k = \mathcal{F}(x_k)$	$E(p)r_{k-1}r_k$	$r_{k-1}r_k$
	Up. part. deriv	$\partial f_{kij\ell} = \boldsymbol{a}[i]\frac{\partial f_k^{(ij)}}{\partial \boldsymbol{\theta}_{kij\ell}}$	$G(p)r_{k-1}r_k$	$pr_{k-1}r_k$
	Update	$\boldsymbol{a} \leftarrow \boldsymbol{a}\mathbf{F}_k$	$r_{k-1}r_k$	-

FT evaluation and gradient algorithm
Forward sweep

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\boldsymbol{a} = \mathbf{F}_1\mathcal{F}(x_1)$	$E(p)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{\theta}_{1ij\ell}}$	$G(p)r_1$	pr_1
Stage 2	Eval	$\mathbf{F}_2 = \mathcal{F}(x_2)$	$E(p)r_1r_2$	r_1r_2
	Up. part. deriv	$\partial f_{2ij\ell} = \boldsymbol{a}[i]\frac{\partial f_2^{(ij)}}{\partial \boldsymbol{\theta}_{2ij\ell}}$	$G(p)r_1r_2$	pr_1r_2
	Update	$\boldsymbol{a} \leftarrow \boldsymbol{a}\mathbf{F}_2$	r_1r_2	-
Stage k	Eval	$\mathbf{F}_k = \mathcal{F}(x_k)$	$E(p)r_{k-1}r_k$	$r_{k-1}r_k$
	Up. part. deriv	$\partial f_{kij\ell} = \boldsymbol{a}[i]\frac{\partial f_k^{(ij)}}{\partial \boldsymbol{\theta}_{kij\ell}}$	$G(p)r_{k-1}r_k$	$pr_{k-1}r_k$
	Update	$\boldsymbol{a} \leftarrow \boldsymbol{a}\mathbf{F}_k$	$r_{k-1}r_k$	-
Stage d	Eval	$\mathbf{F}_d = \mathcal{F}(x_d)$	$E(p)r_{d-1}$	r_{d-1}
	Part. deriv	$\partial f_{dij\ell} = \boldsymbol{a}[i]\frac{\partial f_d^{(ij)}}{\partial \boldsymbol{\theta}_{dij\ell}}$	$G(p)r_{d-1}$	pr_{d-1}
	Final evaluation	$f(x) = \boldsymbol{a}\mathbf{F}_k$	r_{d-1}	-

FT evaluation and gradient algorithm
Backward sweep

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage $d-1$	Initialize	$\boldsymbol{c} = \mathbf{F}_d$	-	-
	Part deriv	$\partial f_{1ij\ell} = \partial f_{1ij\ell}\boldsymbol{c}[j]$	r_{d-1}	-

FT evaluation and gradient algorithm
Backward sweep

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage $d-1$	Initialize	$\boldsymbol{c} = \mathbf{F}_d$	-	-
	Part deriv	$\partial f_{1ij\ell} = \partial f_{1ij\ell} \boldsymbol{c}[j]$	r_{d-1}	-
Stage k	Up. right product	$\boldsymbol{c} \leftarrow \mathbf{F}_{k+1} \boldsymbol{c}$	$r_k r_{k+1}$	-
	Up. part. deriv	$\partial f_{kij\ell} = \partial f_{kij\ell} \boldsymbol{c}[j]$	$r_{k-1} r_k$	-

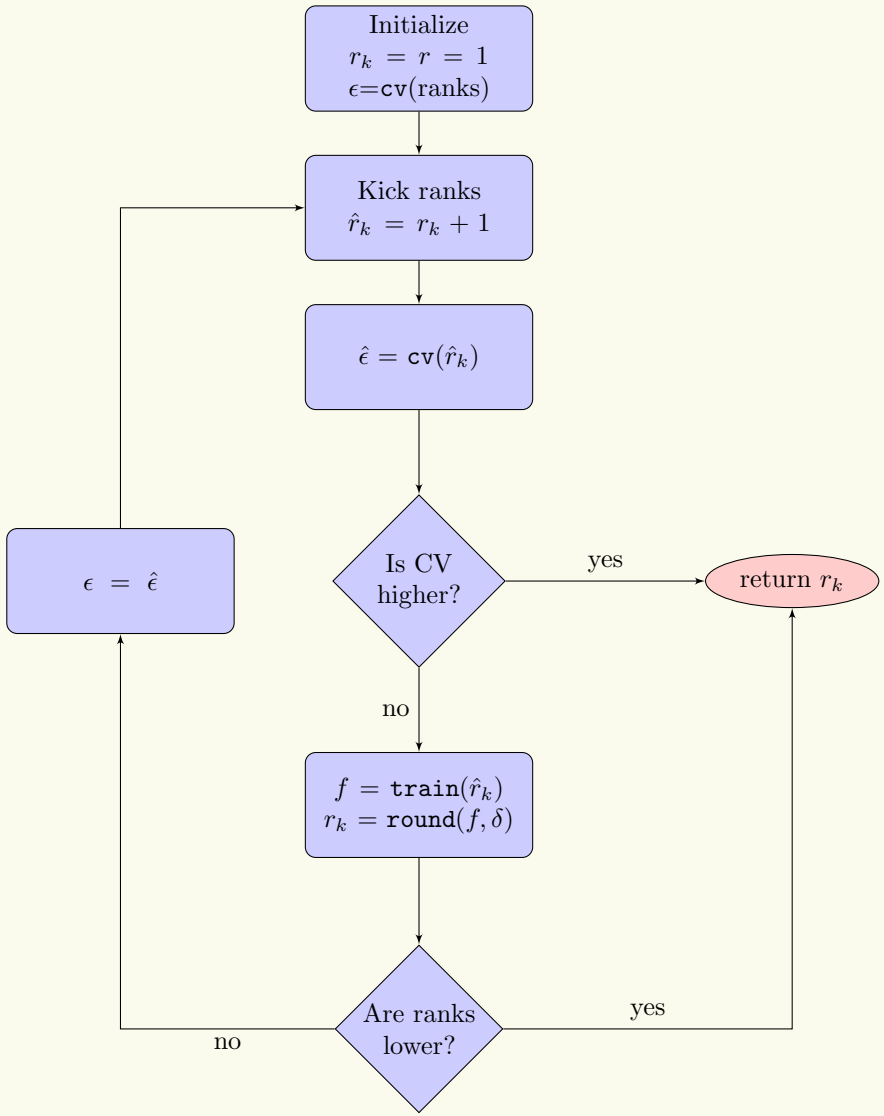
FT evaluation and gradient algorithm
Backward sweep

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage $d-1$	Initialize	$\boldsymbol{c} = \mathbf{F}_d$	-	-
	Part deriv	$\partial f_{1ij\ell} = \partial f_{1ij\ell}\boldsymbol{c}[j]$	r_{d-1}	-
Stage k	Up. right product	$\boldsymbol{c} \leftarrow \mathbf{F}_{k+1}\boldsymbol{c}$	$r_k r_{k+1}$	-
	Up. part. deriv	$\partial f_{kij\ell} = \partial f_{kij\ell}\boldsymbol{c}[j]$	$r_{k-1} r_k$	-
Stage 1	Up. part. deriv	$\partial f_{1ij\ell} = \partial f_{1ij\ell}\boldsymbol{c}[j]$	r_1	-

Alg. complexity: $\mathcal{O}\left(dr^2\left(G(p) + E(p)\right)\right)$

Rank adaptation

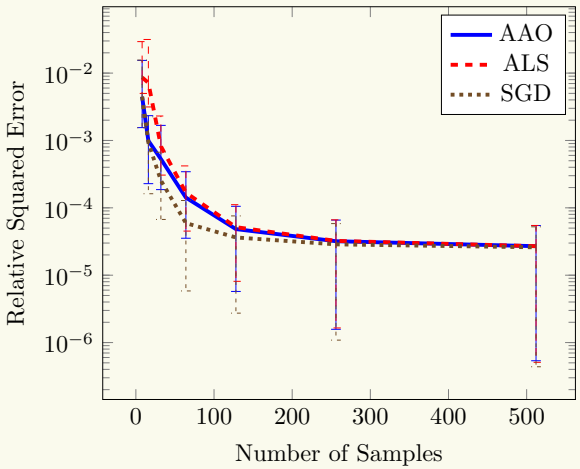
- Use FT rounding and CV
- Increase ranks until either
  - Rounding lowers all ranks
    $\rightarrow$ data not informative enough
  - CV error increases
    $\rightarrow$ avoid overfitting
- Rounding threshold is a parameter
  - Similar to regularization
  - Relation to other approaches?



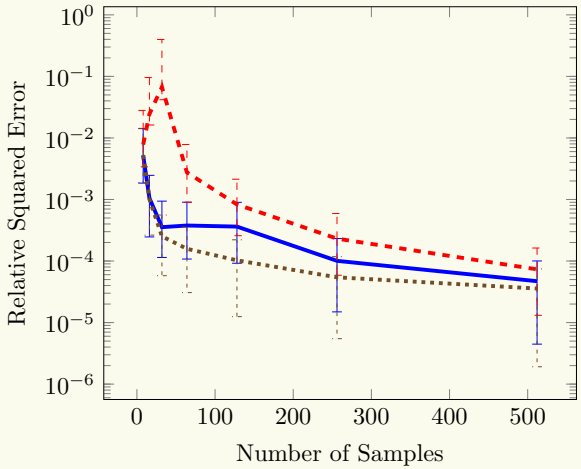
OTL Circuit function (6 dimensions)

- Compare gradient-based optimization vs ALS
- Quasi-Newton (AAO) and SGD for gradients
- Pre-specified ranks and polynomial orders

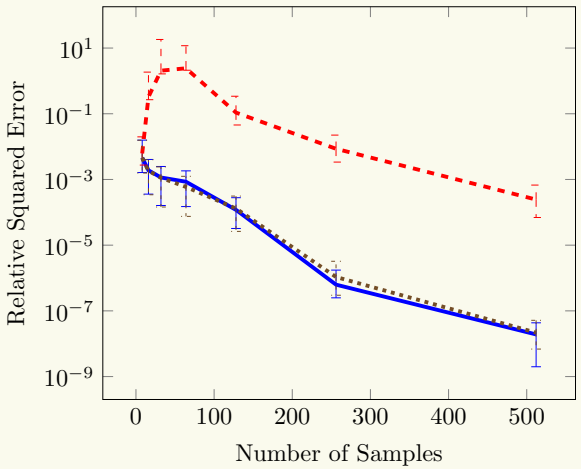
$$f(R_{b1}, R_{b2}, R_f, R_{c1}, R_{c2}, \beta) = \frac{(V_{b1} + 0.74)\beta(R_{c2} + 9)}{\beta(R_{c2} + 9) + R_f} + \frac{11.35R_f}{\beta(R_{c2} + 9) + R_f} + \frac{0.74R_f\beta(R_{c2} + 9)}{(\beta(R_{c2} + 9) + R_f)R_{c1}},$$



(a) $r = 2, p = 3$



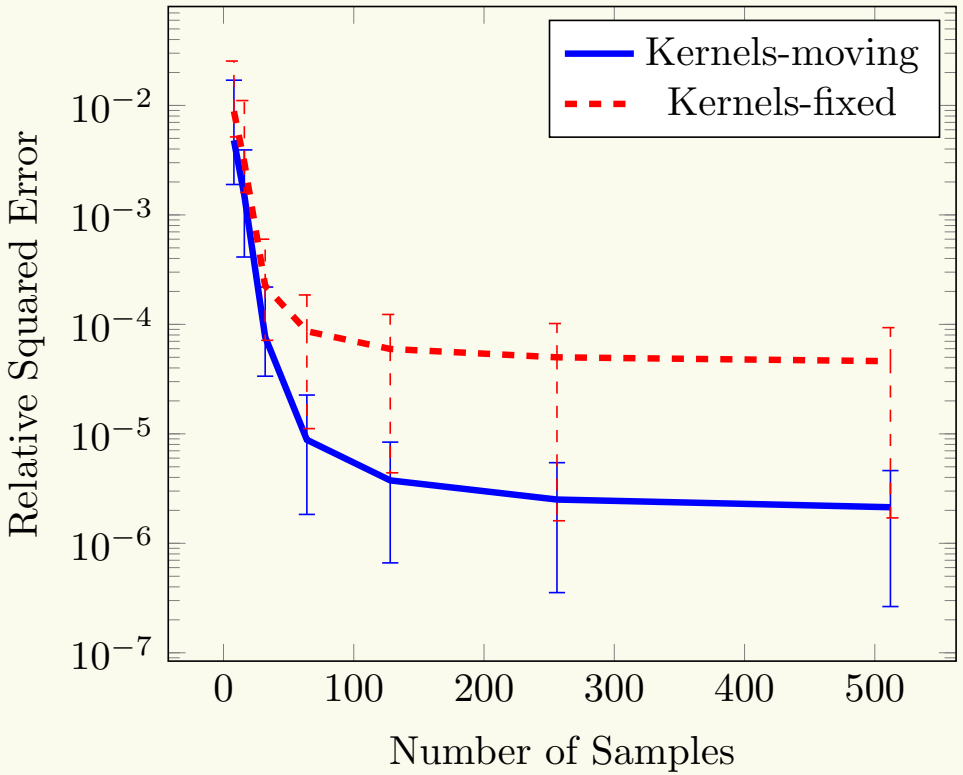
(b) $r = 4, p = 3$



(c) $r = 4, p = 9$

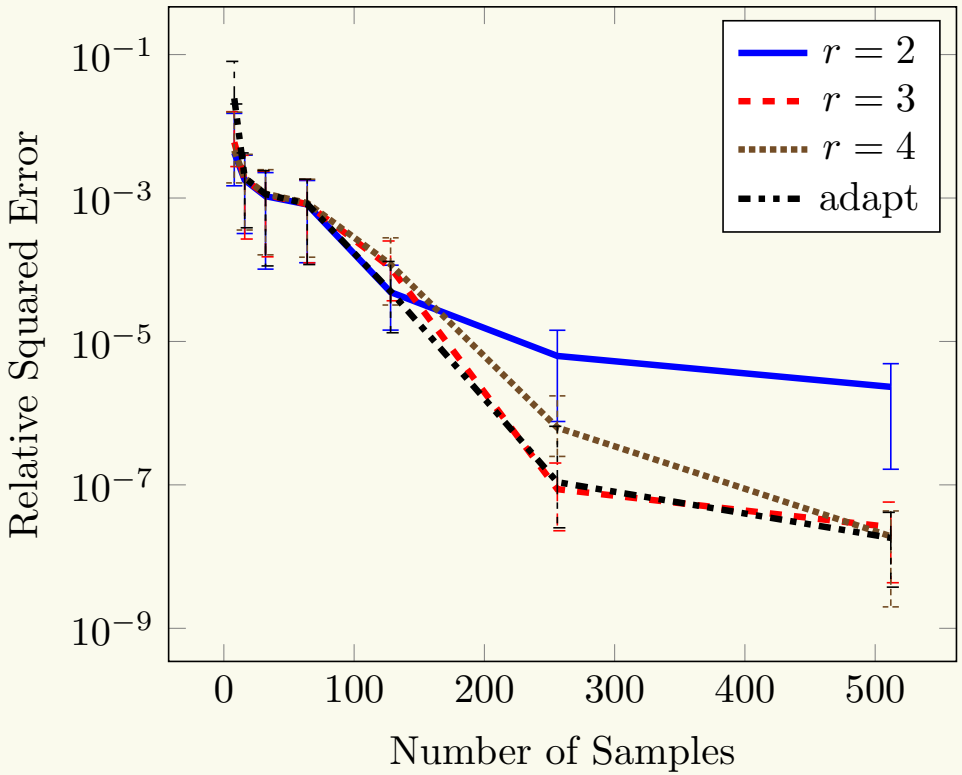
Linear vs. nonlinear parameterizations

- Switch to kernels
- Compare 8 kernels with known centers and 4 kernels with optimized centers



Rank adaptation

- Back to polynomials
- Fix to eight order
- Rank adaptation tracks best rank approximation



Real data

- Kernel expansion for each univariate function
- CV over regularization
- CV over number of basis functions
- Compare with other parametric approaches
- Data and results for non-FT algs from Kandasamy and Yu (2016).
  - 22 algorithms: both parametric and non parametric
  - 10 datasets from UCI database

Results

- Mean squared error on validation data for a subset of regression algorithms and data sets from Kandasamu 2016
- Only the algorithms which scored the best in at least one data set and the CV-based LASSO are shown
- **The FT proved the best parametric and second best overall**

Dataset (d, n)	SALSA	nSVR	RT	GBRT	MARS	FT	CVLASSO
Housing (12,256)	**0.26241**	0.38600	1.06015	0.42951	0.42379	0.32798	0.35218
Galaxy (20,2000)	**0.00014**	0.15798	0.02293	0.01405	0.00163	0.00056	0.00249
Skillcraft (18,1700)	0.54695	0.66311	1.08047	0.57273	0.54595	**0.54434**	0.56879
CCPP (59,2000)	0.06782	0.09449	1.04527	**0.06181**	0.08189	0.06631	0.06466
Speech (21,520)	0.02246	0.06994	0.05430	0.03515	**0.01647**	0.02684	0.03131
Music (90,1000)	0.62512	**0.59399**	1.45983	0.66652	0.88779	0.72094	0.64485
Telemonit (19,1000)	0.03473	0.05246	**0.01375**	0.04371	0.02400	0.04040	0.06487
Propulsion (15,200)	0.00881	0.00910	0.02341	0.00061	0.01290	**0.00009**	0.02660
Airfoil (40,750)	0.51756	0.55118	0.45249	**0.34461**	0.54552	0.46920	0.46444
Forestfires (10,211)	0.35301	0.43142	0.41531	**0.26162**	0.33891	0.39465	0.44175

Conclusions

- Takeaways:
  - Low-rank regression can be extremely flexible if formulated generally
  - Gradient-based optimization can be leveraged with more than just ALS
  - Attractive alternative to sparse regression
  - Sparse penalities on FT cores are promising, but needs more investigation
- Code:
  - Available at: `github.com/goroda/Compressed-Continuous-Computation`
  - Includes:
    - General purpose utilities for computing with low-rank functions
    - Optimal sampling and cross approximation for converting functions into FTs
- Preprint: A.A. Gorodetsky and J.D. Jakeman. *Gradient-based Optimization for Regression in the Functional Tensor-Train Format* (Under Review)

Acknowledgements

This research was funded by the Defense Advanced Research Projects Agency (DARPA) under the EQUiPS program and by the John von Neumann Postdoctoral Research Fellowship supported by the DOE Applied Mathematics program in the Office of Advanced Scientific Computing Research.