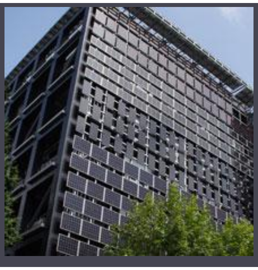
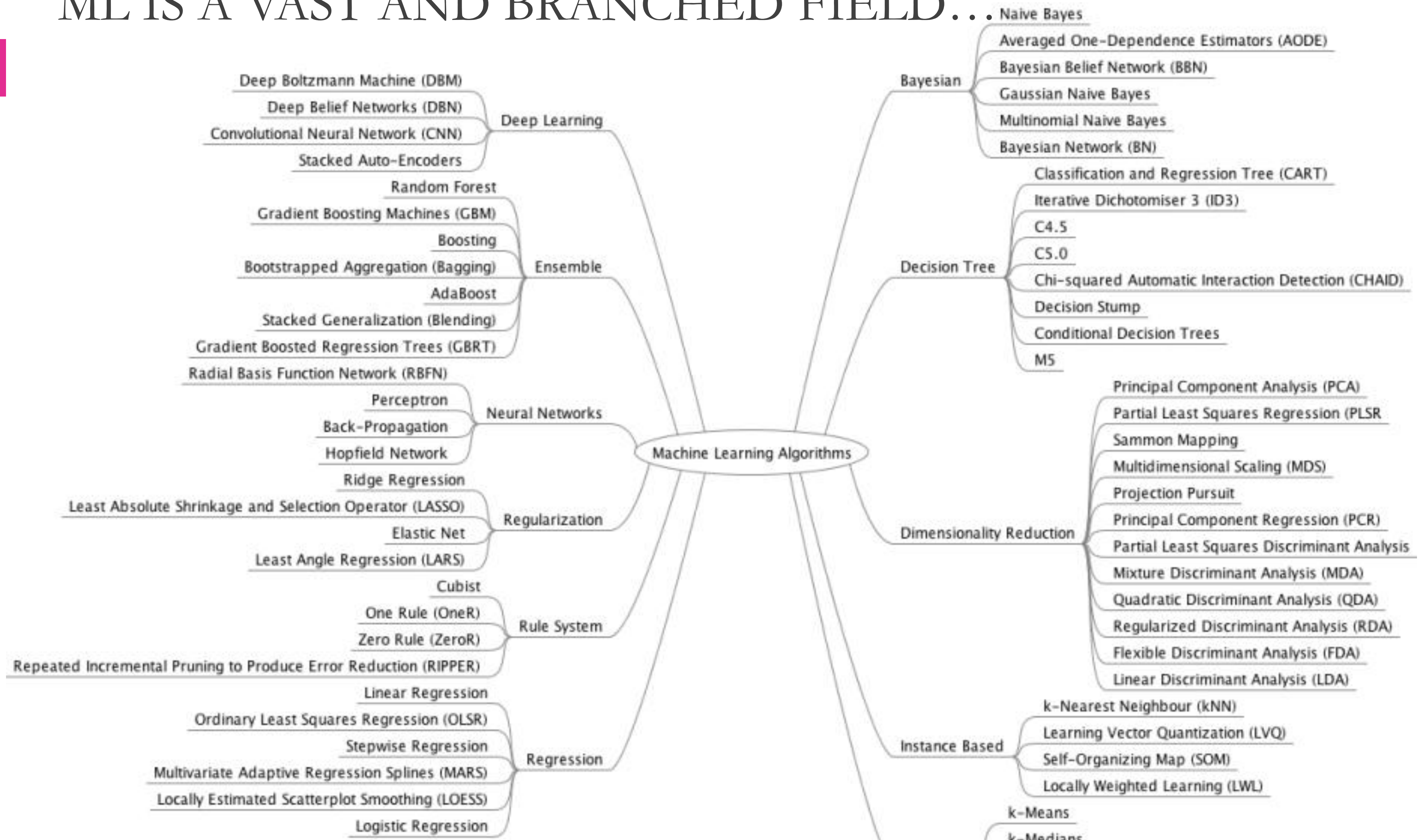
EFFICIENT MACHINE LEARNING FROM SPARSE DATA IN HIGH-*D* FEATURE SPACES

Sergei Manzhos

Ihara-Manzhos Laboratory
https://sites.google.com/site/sergeimanzhos/
http://www.chemeng.titech.ac.jp/~iharalab/
School of Materials and Chemical Technology



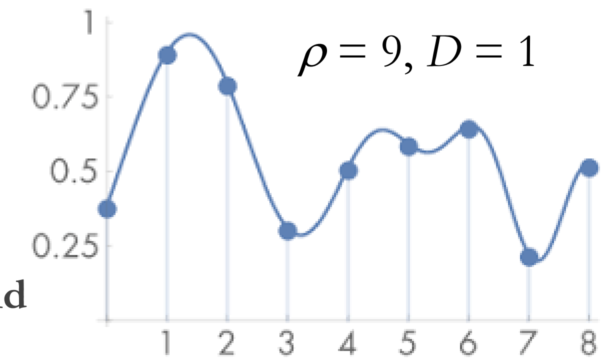
ML IS A VAST AND BRANCHED FIELD…



- We will focus on regression problems
- And methods ***based on*** neural networks and kernel methods

OFF-THE-SHELF ML METHODS WORK WELL IN MODERATELY HIGH-D WITH MODERATELY SPARSE DATA

- Curses of dimensionality
  - Exponentially difficult to fill space: $N_{\text{data}} = \rho^D$
  - Data are **always** sparse in multidimensional spaces (example: 10^6 points in 20D ~ 2 points/DOF, 10^7 ~ 2.2 points/DOF)
  - **Exponential growth of required no. of data and terms hold for direct product type representations (think a regular grid / Fourier expansion)**
    - **ML methods avoid them**
  - What is local in low-d may be non-local in high-D: 1d: 68% of quadrature within σ; 6d: 10% of quadrature



$$f(\boldsymbol{x}) = \prod_{i=1}^{D} (2\pi\sigma_i^2)^{-\frac{1}{2}} exp\left(\frac{(x_i - \bar{x}_i)^2}{2\sigma_i^2}\right)$$

Consequential for kernel regressions

- Possible blessings of dimensionality
  - Concentration of metrics:

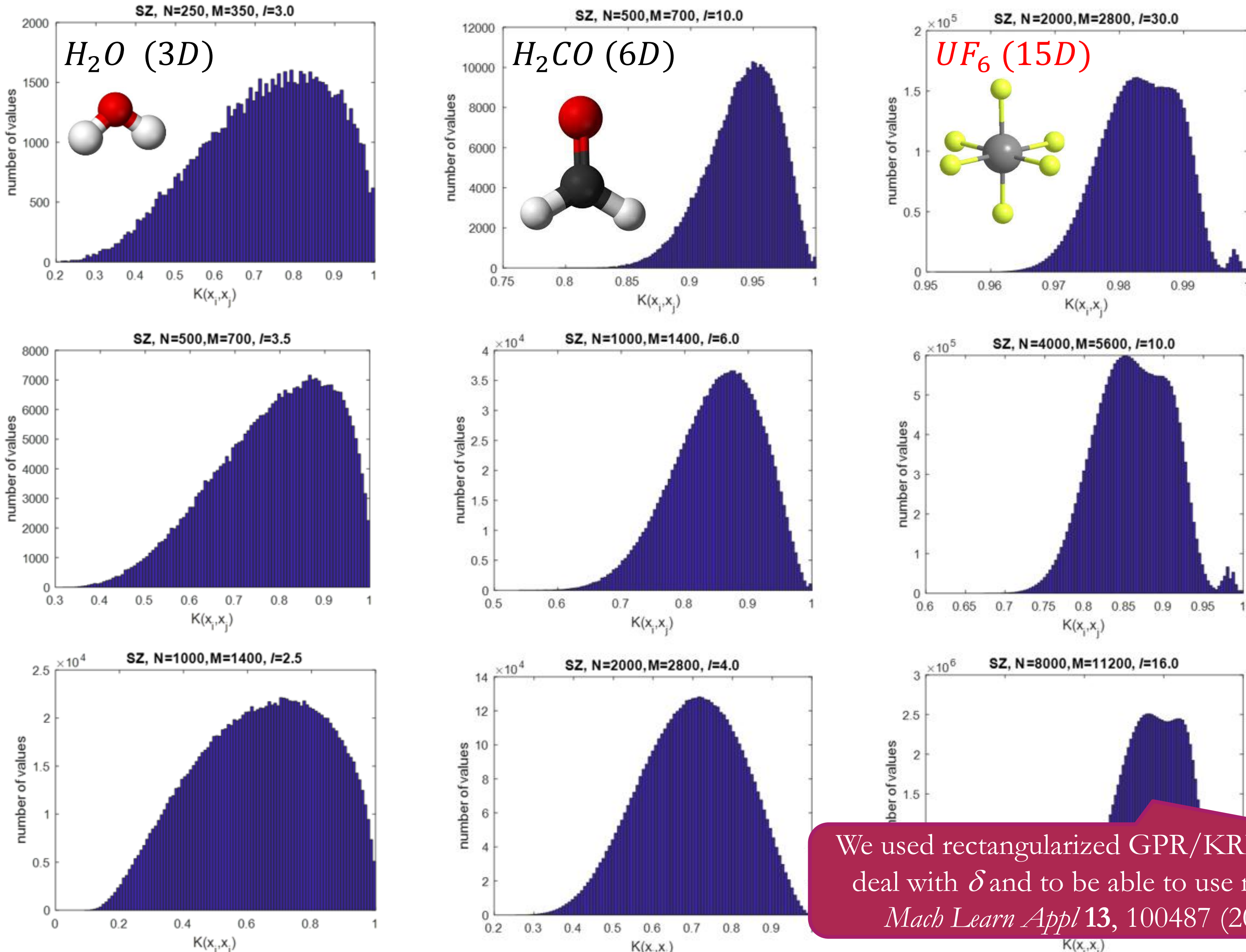
$$\lim_{D\to\infty} E\left(\frac{dist_{max}(D) - dist_{min}(D)}{dist_{min}(D)}\right) \to 0$$

  - Some ML methods work better in high than in low dimensionality (e.g. NN)

- **But in very high-D or with very low density of data specialized ML methods are desired**

HOW HIGH-D MATERN KERNELS DIE: SPARSE DATA

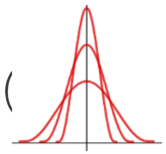
Number of training pts

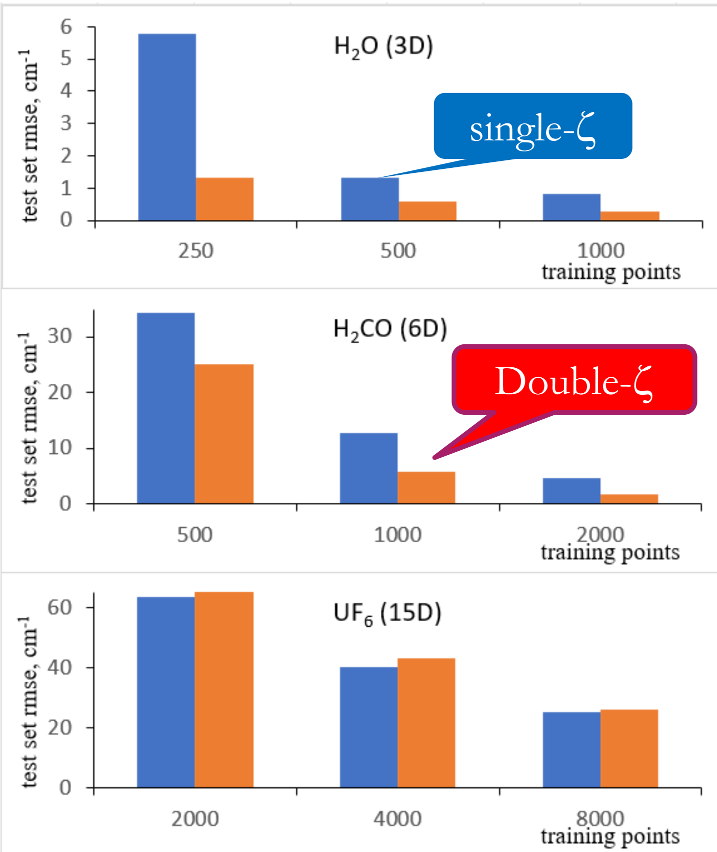


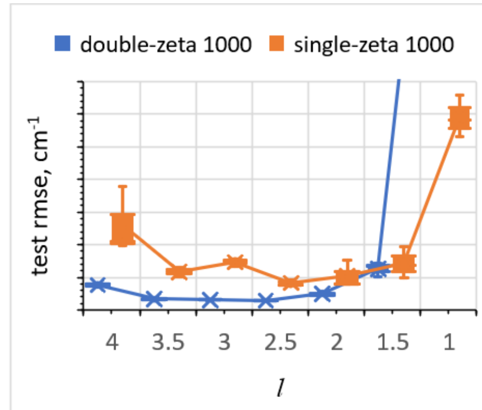
We used rectangularized GPR/KRR to not deal with δ and to be able to use multi-ζ *Mach Learn Appl* **13**, 100487 (2023)

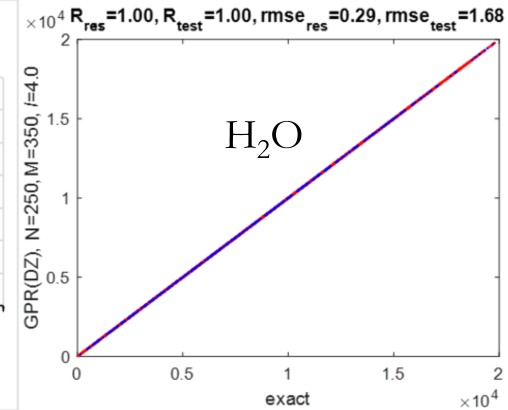
- **Matern type kernels lose the property of locality…**

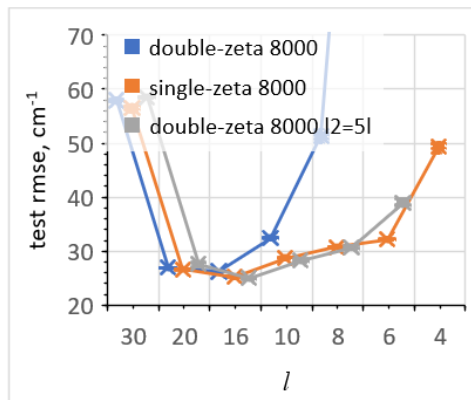
J. Chem. Phys. **158**, 044111 (2023)

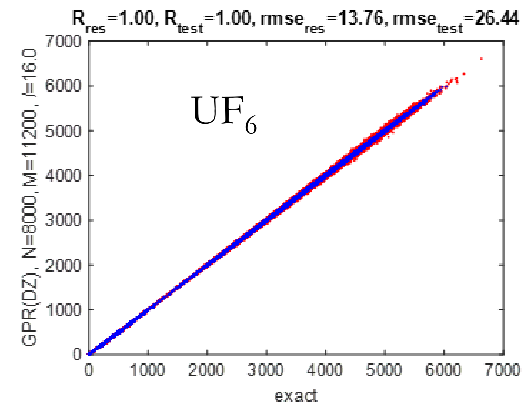
ADVANTAGE OF MULTI-ζ () DIES WHEN LOCALITY IS GONE











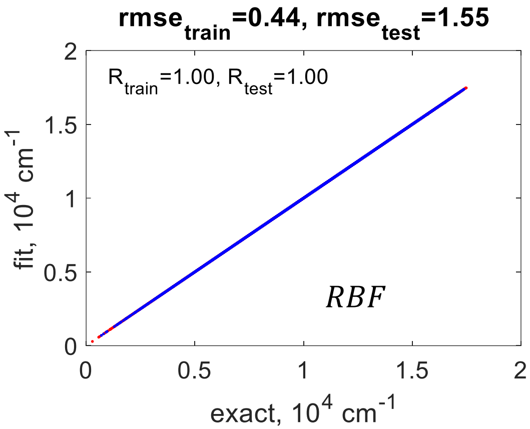
Error in, cm^{-1}	PES	ΔZPE	Lowest 50 transitions		Lowest 100 transitions	
			mae	rmse	mae	Rmse
Single-zeta kernel	4.60	0.077	0.138	0.166	0.139	0.165
Double-zeta kernel	1.67	-0.026	0.013	0.016	0.014	0.019

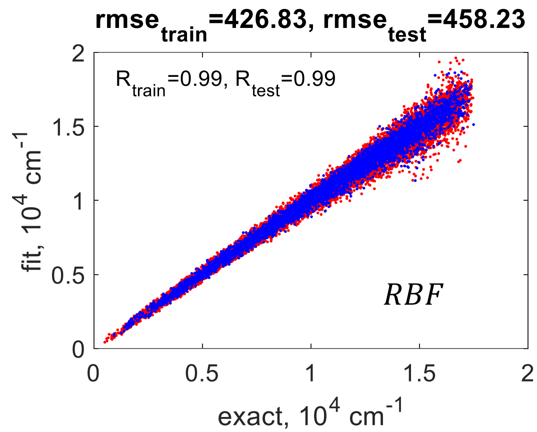
Double-ζ kernel PES representation useful for properties when it is meaningful (H_2CO, 2000 kernel centers)

J. Chem. Phys., **158**, 044111 (2023)

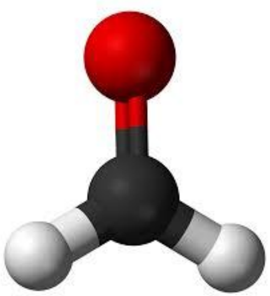
HOW HIGH-D KERNELS DIE: SPARSE DATA

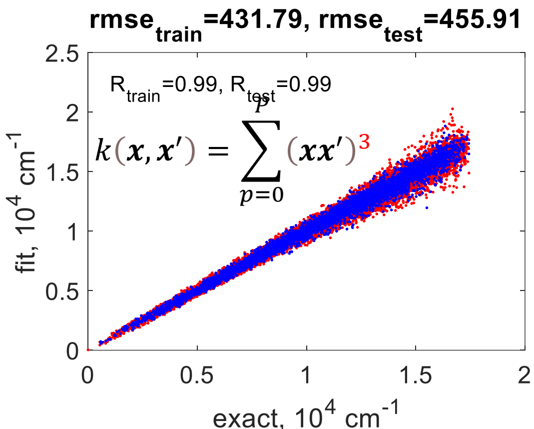
5,600 train pts
20,000 test pts
$l_{opt}\sim 5$

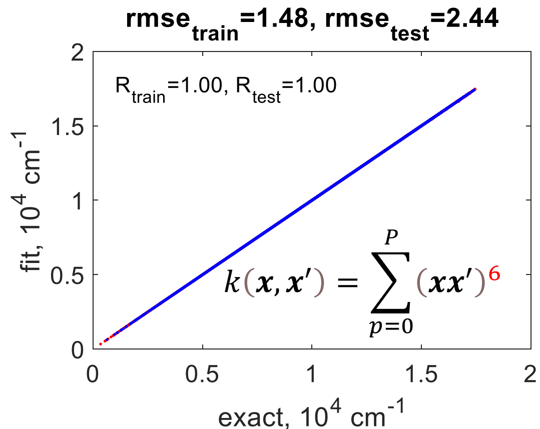




$l = 50$







Product kernel = polynomial regression

$$k(\boldsymbol{x},\boldsymbol{x}') = \chi + (\boldsymbol{x}\boldsymbol{x}')^2:$$

$$f(\boldsymbol{x}) = \left(\chi \sum_{n=1}^{N} c_n\right) + \sum_{d=1}^{D}\left(\sum_{n=1}^{N} c_n \left(x_d^{(n)}\right)^2\right) x_d^2 + \sum_{d\neq d'}^{D}\left(\sum_{n=1}^{N} c_n x_d^{(n)} x_{d'}^{(n)}\right) x_d x_{d'}$$

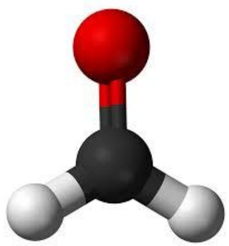
$$= b_0 + \sum_{d=1}^{D} b_d^{(2)} x_d^2 + \sum_{d\neq d'}^{D} b_{dd'} x_d x_{d'}$$

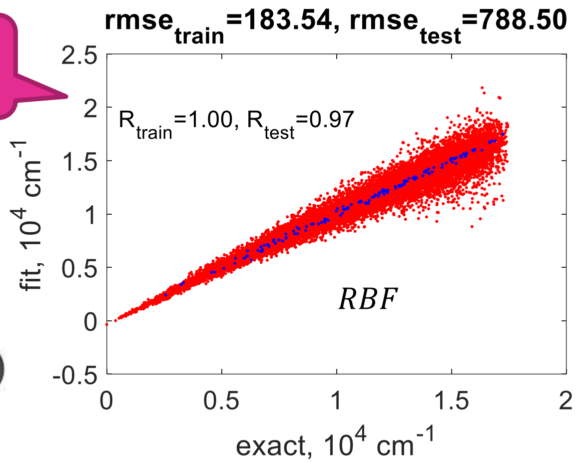
J. Chem. Phys., **160**, 012201(2024)

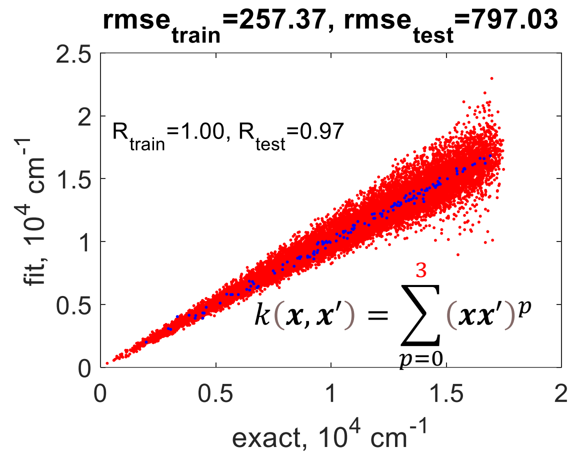
- **…and may degenerate into a low-order polynomial basis**

HOW HIGH-D MATERN KERNELS DIE: SPARSE DATA

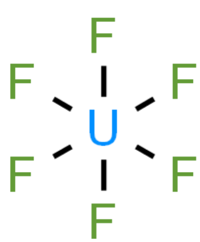
150 train pts
l_{opt}~15

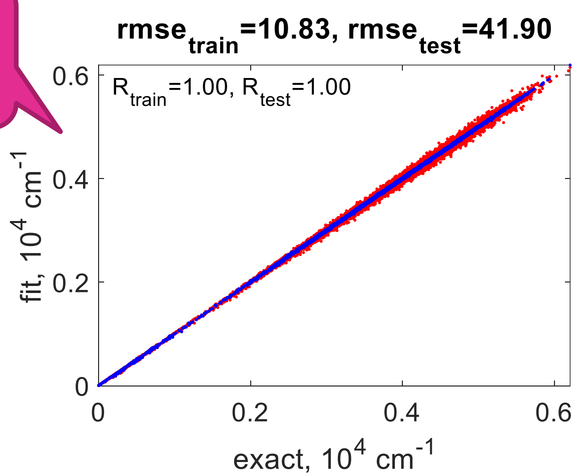


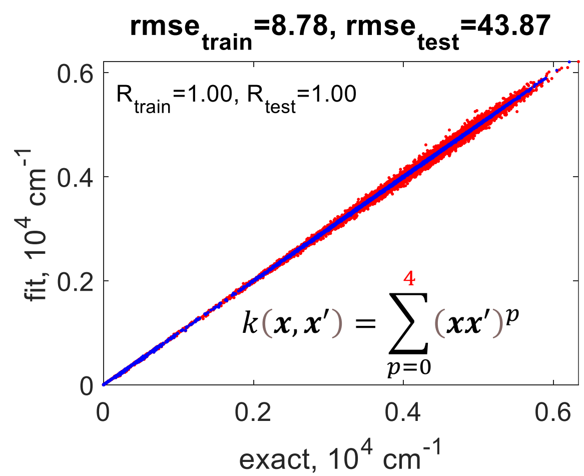




15D
5600 train pts
l_{opt}~30







- **Higher D → lower data density → higher l_{opt}**

J. Chem. Phys., **160**, 012201(2024)

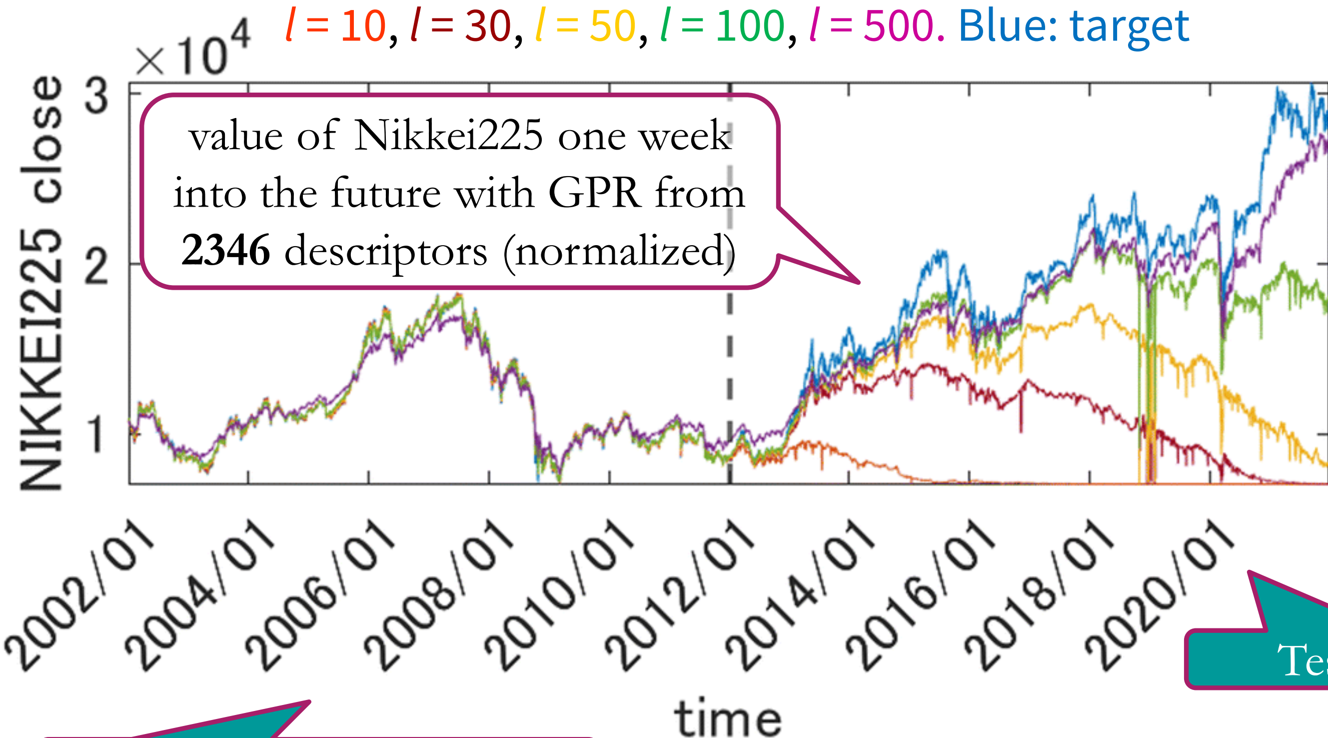
FAILURE OR COLLAPSE OF GPR WITH MATERN KERNELS IN VERY HIGH-D

EXAMPLE FROM QUANTITATIVE FINANCE

$$y(t+\Delta t) = f\left(x_1(t), x_2(t), \dots x_D(t)\right)$$

$$\exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{x}'|^2}{l^2}\right) = \prod_d^D \exp\left(-\left(\frac{x_d - x'_d}{l}\right)^2\right)$$

A product of very many functions each of value <1



Test: 2012-2020

Training: 2002-2011

GPR collapses unless l is so large it loses any advantage over simple linear regression

Phys. Chem. Chem. Phys. **25**, 1546 (2023)

HIGH-DIMENSIONAL MODEL REPRESENTATION (HDMR)

1st order additive

$$x \in R^{D};\ f(\boldsymbol{x}) \approx f_0 + \sum_{i=1}^{D} f_i(x_i) + \sum_{\substack{i=1,\\ j=i+1}}^{D} f_{ij}(x_i, x_j) + \cdots$$

$$+ \sum_{i_1, i_2, \ldots, i_d}^{D} f_{i_1, i_2, \ldots, i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d}) + \cdots$$

$C_d(D)$ component functions at each d

In some form dates back 100 years (ANOVA)

HDMR is formalized in a series of papers by Rabitz's group at Princeton

J Math Chem A **1999,** *25*, 197 etc

Particular cases: physics – MBE, …

$$E = \sum_{I=1}^{N_b} E_I + \sum_{I}^{N_b} \sum_{J>I}^{N_b} \Delta E_{IJ} + \sum_{I}^{N_b} \sum_{J>I}^{N_b} \sum_{K>J}^{N_b} \Delta E_{IJK} + \cdots$$

… computational spectroscopy: N-mode approximation

Can consider only highest-d terms:

$$f(\boldsymbol{x}) \approx \sum_{i_1, i_2, \ldots, i_d}^{D} f_{i_1, i_2, \ldots, i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d})$$

Comput Phys Commun **2009,** *180*, 2002
J Phys Chem A **2020,** *124*, 7598
Comput Phys Commun **2022,** *271*, 108220

- Formalization of **representation with lower-dimensional functions (many-body, N-mode)**
  - Easier to build: component functions are lower-dimensional, could be built with fewer data w/out overfitting (e.g. *J Math Chem* **61** (2023) 7)
    - Fitting method can work in its comfort zone
  - Easier to use (integrals etc)
  - Provides elements of insight (see e.g. *Comput Phys Commun* **2022,** *271*, 108220)

HDMR-ML: A POWERFUL TOOL TO WORK WITH SPARSE DATA

Review: *Artificial Intelligence Chemistry* **1**, 100008 (2023)

$$f(\boldsymbol{x}) \approx f_0 + \sum_{i=1}^{D} f_i(x_i) + \sum_{\substack{i=1,\\ j=i+1}}^{D} f_{ij}(x_i, x_j) + \cdots + \sum_{i_1,i_2,\ldots,i_d}^{D} f_{i_1,i_2,\ldots,i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d})$$

$$f_{k_1 k_2 \ldots k_d}(x_{k_1}, x_{k_2}, \ldots, x_{k_d}) = f(\boldsymbol{x}) - \sum_{\substack{\{i_1 i_2 \ldots i_d\} \in \{12 \ldots D\} \\ \{i_1 i_2 \ldots i_d\} \neq \{k_1 k_2 \ldots k_3\}}}^{D} NN_{i_1 i_2 \ldots i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d})$$

Cycle through all functions until convergence

Comput Phys Commun **180** (2009) 2002
Chem Rev **121** (2021) 10187 & refs therein

$$f_{k_1 k_2 \ldots k_d}(x_{k_1}, x_{k_2}, \ldots, x_{k_d}) = f(\boldsymbol{x}) - \sum_{\substack{\{i_1 i_2 \ldots i_d\} \in \{12 \ldots D\} \\ \{i_1 i_2 \ldots i_d\} \neq \{k_1 k_2 \ldots k_3\}}}^{D} GPR_{i_1 i_2 \ldots i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d})$$

J Phys Chem A **124** (2020) 7598
Comput Phys Commun **271** (2022) 108220 (with example of KED)

- Component functions are easier to build from fewer data: HDMR is helpful when data are sparse, *which they always are in high-D* (full D-dimensional coupling terms may not be recoverable)
- Using ML for $f_{i_1,i_2,\ldots,i_d}$ simplifies greatly their calculation (avoids debilitating $D-d$ dimensional integrals which formally express $f_{i_1,i_2,\ldots,i_d}$)

HDMR REPRESENTATION ACHIEVED VIA GPR KERNEL DESIGN

$$k(\boldsymbol{x},\boldsymbol{x}') = \sum_{\{i_1 i_2 \dots i_d\}\in\{12\dots D\}} A_{i_1 i_2 \dots i_d} k_{i_1 i_2 \dots i_d}\left(\boldsymbol{x}_{i_1 i_2 \dots i_d}, \boldsymbol{x}'_{i_1 i_2 \dots i_d}\right)$$

Amplitudes of individual terms are unimportant and can be omitted

$$f(\boldsymbol{x}) = \sum_{\{i_1 i_2 \dots i_d\}\in\{12\dots D\}} A_{i_1 i_2 \dots i_d} \sum_{n=1}^{M} k_{i_1 i_2 \dots i_d}\left(\boldsymbol{x}_{i_1 i_2 \dots i_d}, \boldsymbol{x}^{(n)}_{i_1 i_2 \dots i_d}\right) c_n$$

$$\equiv \sum_{\{i_1 i_2 \dots i_d\}\in\{12\dots D\}} B_{i_1 i_2 \dots i_d} \tilde{f}_{i_1 i_2 \dots i_d}\left(\boldsymbol{x}_{i_1 i_2 \dots i_d}\right)$$

$$f_{k_1 k_2 \dots k_d}\left(x_{k_1}, x_{k_2}, \dots, x_{k_d}\right) = \boldsymbol{K}^{*}_{i_1 i_2 \dots i_d}\boldsymbol{c}$$

In the simplest case $k(\boldsymbol{x},\boldsymbol{x}') = \sum_{i=1}^{D} k_i(x_i, x_i{}')$

$$f(\boldsymbol{x}) \approx \sum_{i=1}^{D} f_i(x_i) = \left(\sum_{i=1}^{D} \boldsymbol{K}^{*}_{i}\right)\boldsymbol{c}'$$

$$\boldsymbol{c}' = \left(\sum_{i=1}^{D} \boldsymbol{K}_i\right)^{-1}\boldsymbol{t}$$

$\boldsymbol{K}^{*}_{i}$ is a row vector with elements $k_i\left(x_i, x_i^{(n)}\right)$

Duvenaud et al. *Adv Neural Inf Process Syst* (2011) 226–234
Manzhos et al. *Machin Learn Sci Technol* **2** (2022) 01LT02
Rassmussen GPR books also talks about additive kernels

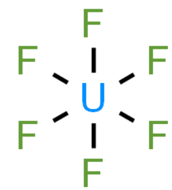
If the kernel is in HDMR form the final function representation also is

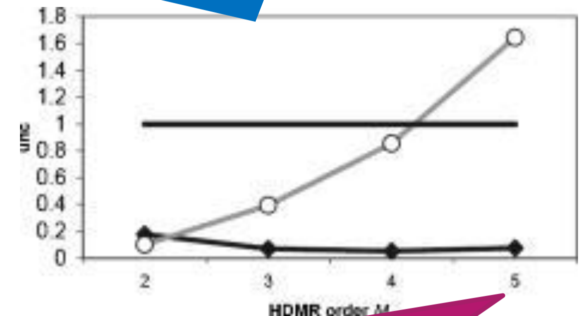
USE OF HDMR TO GENERATE SYNTHETIC DATA TO HELP TUNE HYPERPARAMETERS

$$f(\boldsymbol{x}) \approx f_0 + \sum_{i=1}^{D} f_i(x_i) + \sum_{\substack{i=1,\\ j=i+1}}^{D} f_{ij}(x_i, x_j) + \sum_{i_1, i_2, \ldots, i_d}^{D} f_{i_1, i_2, \ldots, i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d}) + \cdots$$

Relative uncertainties of NN parameters in an HDMR-NN fit of H_2O_2 PES (*J. Chem. Phys.* **125** (2006) 084109)

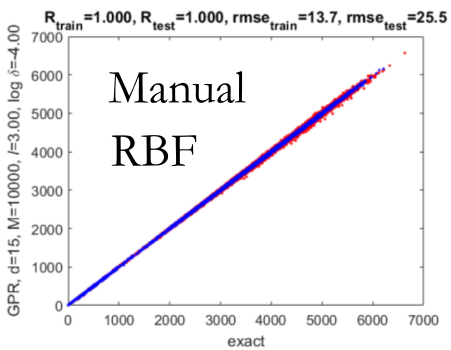
Low-order terms can be reliably determined from few data

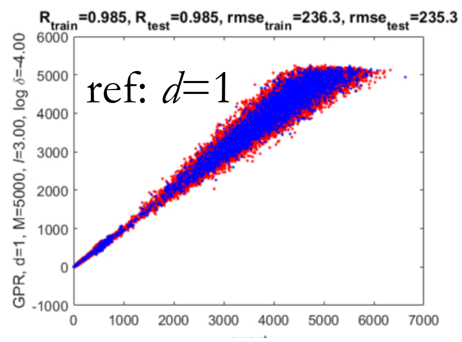


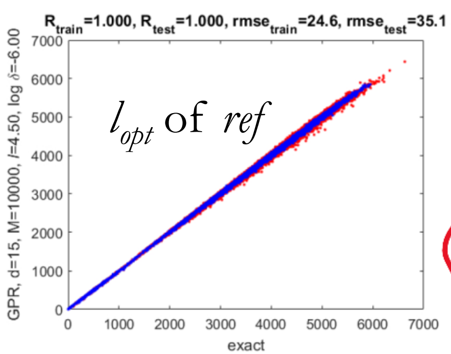


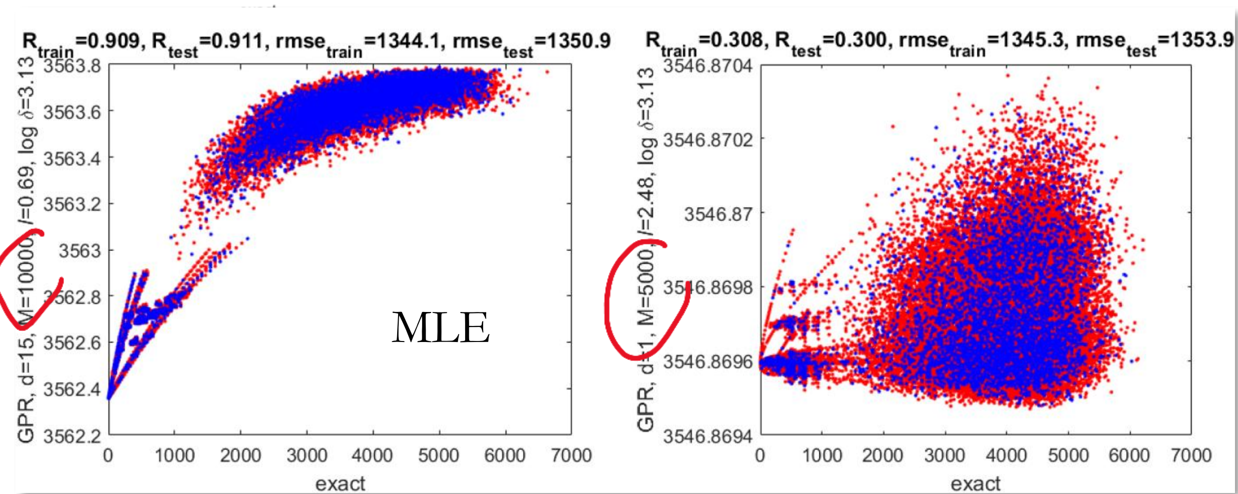
Higher order terms may not be recoverable

- Achievable quality of fit is limited by the density of sampling









J. Math. Chem., **61**, 7 (2023)

WITH SPARSE DATA ONLY LOW-ORDER TERMS ARE RECOVERABLE : EXAMPLE OF UF6 INTERATOMIC POTENTIAL

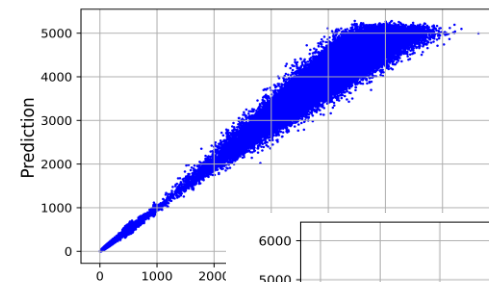
$$V^{UF_6}(\boldsymbol{x}) \approx \sum_{i=i}^{N} f_i^{GPR}\left(x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}\right)$$

$$\boldsymbol{x}: 15(=D)\ modes\ \ of\ vibration$$

Test set of 50,000 pts

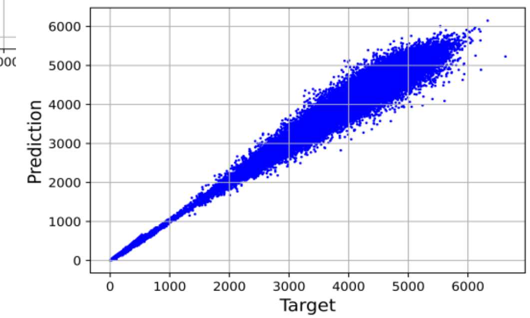
rmse, cm^{-1}	N	5,000	3,000	**2,000**
Full-D ($d = D$)	1	42.2	75.4	**106.7**
$d = 1$	15	234.6	236.4	**237.3**
$d = 2$	105	168.1	178.6	**190.3**
$d = 3$	455	65.6	78.0	**97.4**

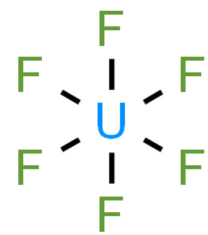
- With 2000 points in 15D space
  - **1.66 data per degree of freedom**
  - rmse(3*d*-HDMR-GPR) < RMSE(full-D GPR)
- A finite dataset in D-dimensional space is not a D-dimensional object

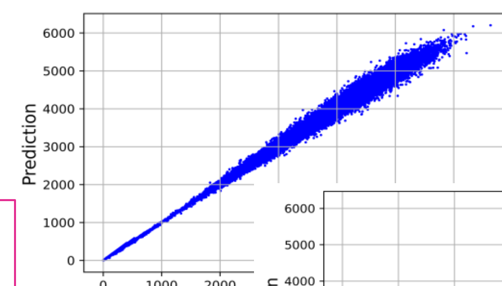


d=1

d=2

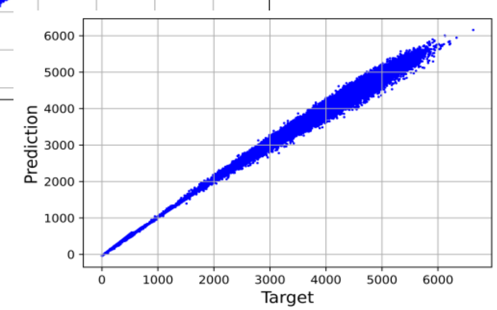






d=3

d=15



J Phys Chem A **2020**, *124*, 7598

HOW TO DEAL WITH OVERFITTING AND NON-LINEAR OPTIMIZATION? - BEYOND TRADITIONAL NN

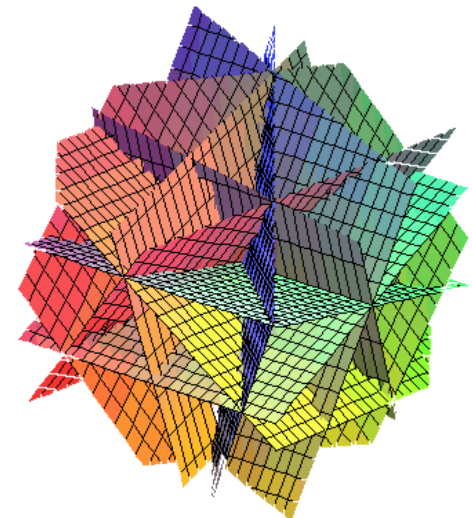
traditional NN: $\boldsymbol{w}$, b define hyperplanes that are optimized for given data, $\sigma(x)$, and N_{neurons}

$$\sigma(\boldsymbol{w_n x} + b_n)$$

$$f(\boldsymbol{x}) = \sum_{n=0}^{N} c_n \sigma_n(\boldsymbol{w_n x} + b_n)$$

We will instead define $\boldsymbol{w}$, b with rule and then optimize $\sigma(x)$ for given data and N_{neurons}

- $\boldsymbol{w}$ define planes' orientations – sample them
- b define distance from origin – we will not need to deal with them



"ridge NN"

- Avoid non-linear optimization
- No particular advantage using the same activation function for all neurons if parameters are not optimized
  - Can optimize neuron activation functions for a particular problem
  - This makes sense if we can build cheaply… which we can

NN AS AN ADDITIVE MODEL IN REDUNDANT COORDINATES

The only extra cost vs standard GPR is summation in the kernel (parallelizable)

$$k(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{N} k_i(y_i, y_i') \qquad k_i(y_i, y_i') = exp\left(-\frac{(y_i - y_i')^2}{2l^2}\right)$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \sigma_i(\boldsymbol{w_i x} + b_i) \equiv \sum_{m=1}^{M} b_m k\left(\boldsymbol{y}, \boldsymbol{y}^{(m)}\right) =$$

$$\sum_{i=1}^{N} \sum_{m=1}^{M} b_m k_i\left(y_i, y_i^{(m)}\right) = \sum_{i=1}^{N} f_i(y_i)$$

NN in $\boldsymbol{x}$

1d-GPR in $\boldsymbol{y}$

$$y_n = \boldsymbol{x}^T \boldsymbol{s}_n$$

Elements of Sobol sequence
$\boldsymbol{w_i}$ define "directions" of y_i -
distribute them pseudorandomly

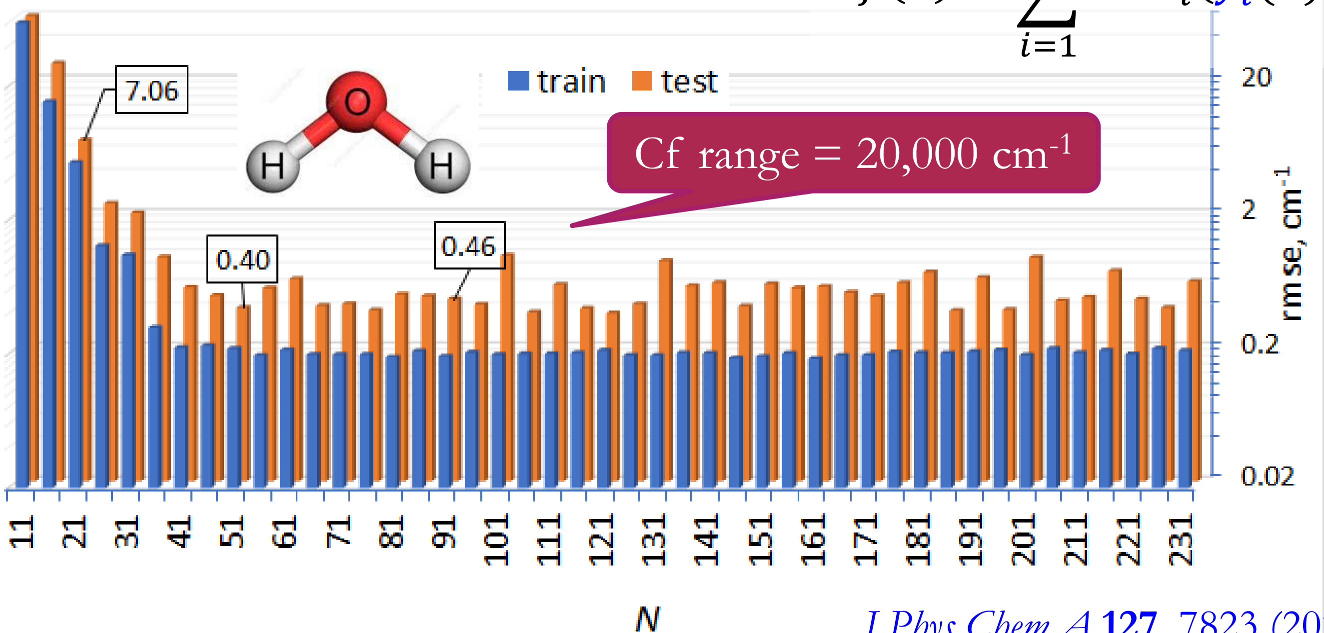
- A first order additive model in redundant coordinates $\boldsymbol{y}$
- f_i can be obtained from 1st order HDMR-GPR in one go with an additive kernel… for given **W**
- Can reduce *N* by using different (optimal) neuron activation functions for different neurons

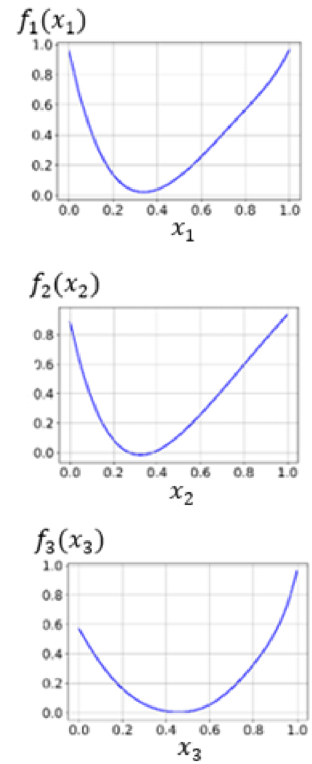
J Phys Chem A **127**, 7823 (2023)

- Avoid nonlinear parameter optimization with rule-based NN weights
  - With no nonlinear optimization, no advantage for all neuron activation functions to be the same
- Without nonlinear parameter optimization… what happens to overfitting as the no. of neurons is increased beyond optimum?

NN WITH OPTIMAL NN NEURONS BUILT WITH 1ST ORDER HDMR-GPR

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} GPR_i(y_i(\boldsymbol{x}))$$





J Phys Chem A **127**, 7823 (2023)

- H_2O PES, training on 1000 samples, large test set of 9000 samples
- **no overfitting** as no. of neurons grows due to absence of nonlinear optimization
- Combines expressive power of NN with robustness of linear regression

- See Huixin Liu (刘辉鑫)'s poster for use of GPR-NN for nuclear mass prediction
- *See J. Phys. Chem. Lett.* 15, 6974 (2024) for use in studies of neuromorphic computing

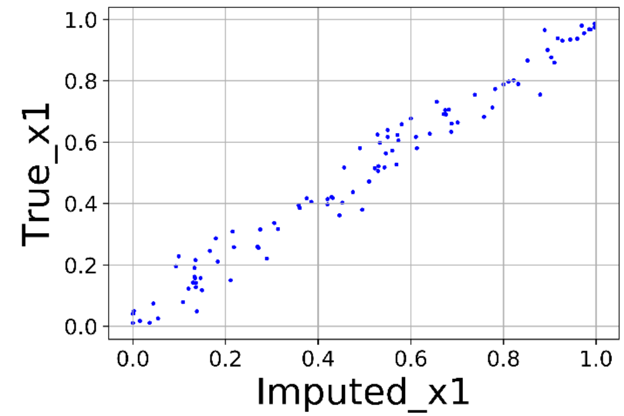
OTHER ADVANTAGES OF HMDR-GPR: HELPS IMPUTE MISSING DATA…

$$f(\boldsymbol{x}) \approx f_0 + \sum_{i=1}^{D} f_i(x_i)$$

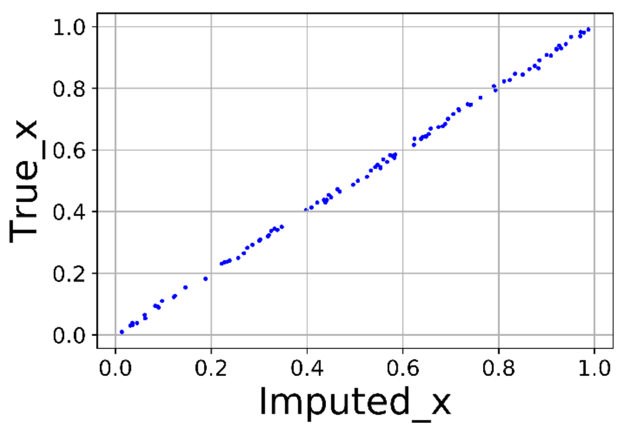
$$f_i(x_i^m) = f(\boldsymbol{x_m}) - f_0 - \sum_{\substack{j=1,\\ j\neq i}}^{D} f_j(x_j^m)$$

$$x_i^m = f_i^{-1}(y_i^m)$$

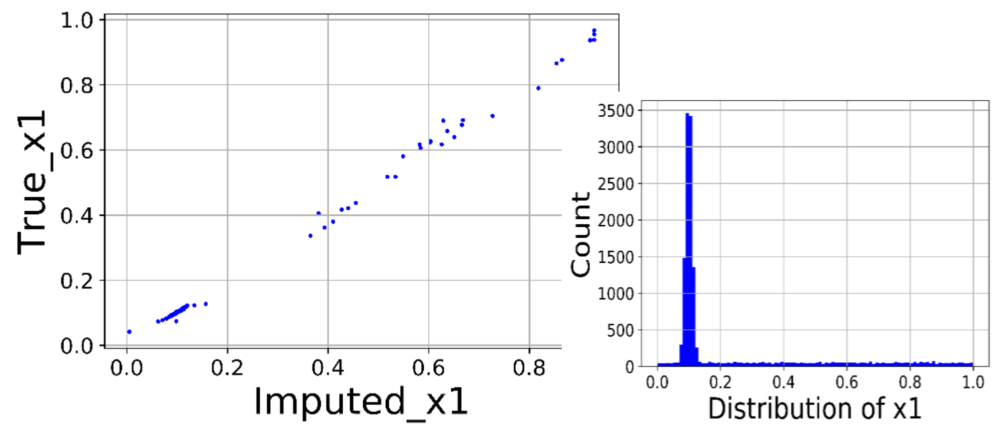
$f(x, y, z) = x + y + z$+noise



$f(x, y, z) = x + y + z$



$f(x, y, z) = x + y + z$ with bad distribution



Comput Phys Commun, **271**, 108220 (2022)
Code: https://github.com/owen-ren0003/rshdmrgpr

… and helps find optimal hyperparameters when data are sparse: *J Math Chem* **61**, 7 (2023)

ML-TO-FORMULA: ML-GUIDED KEF CONSTRUCTION

Kinetic energy in Orbital−free DFT

$$E_{kin} = \int \tau(\boldsymbol{r}) d\boldsymbol{r}$$

$$\boldsymbol{x} = \left(\overline{\tau_{TF}}, \overline{\tau_{TF}p}, \overline{\tau_{TF}p^2}, \overline{\tau_{TF}pq}, \overline{\tau_{TF}q^2}, \overline{\rho V_{eff}}\right)$$

$$\bar{\tau}(\boldsymbol{x}) = \sum_{i=1}^{N} GPR_i(y_i(\boldsymbol{x}))$$

$$E_{kin} = V\bar{\tau}$$

Scaled gradient expansion (4th order)

$$\tau_{GE4}(\boldsymbol{r}) = \tau_{TF}(\boldsymbol{r})\left(1 + \frac{5}{27}p(\boldsymbol{r}) + \frac{20}{9}q(\boldsymbol{r}) + \frac{8}{81}q(\boldsymbol{r})^2 - \frac{1}{9}p(\boldsymbol{r})q(\boldsymbol{r}) + \frac{8}{243}p(\boldsymbol{r})^2\right)$$

$$p = \frac{|\nabla\rho|^2}{4(3\pi^2)^{2/3}\rho^{8/3}}, \qquad q = \frac{\Delta\rho}{4(3\pi^2)^{2/3}\rho^{5/3}}$$

Mach. Learn. Sci. Technol. **6**, 035002 (2025)

"insightful" ML
See *Digital Discovery* **3**, 1967 (2024), *J. Mater. Inform.* **5**, 38 (2025) for uses in materials informatics

Additive model, feature importance

Coupling terms

$$\bar{\tau}(\boldsymbol{x}) = \sum_{i=1}^{D} f(x_i) + \sum_{i=D+1}^{N} f_i(y_i(\boldsymbol{x}))$$

Functional form

ML-guided analytic KEF

$$\bar{\tau} = \sum_{n=1}^{6}\sum_{p=1}^{P_n} a_{np}(x'_n)^p$$

GPR-NN WITH OPTIMIZED REDUNDANT COORDINATES COULD OBVIATE DEEP NN IN SOME APPLICATIONS

Cf. kernel regression $f(\boldsymbol{x}) = \sum_{i=1}^{M} c_i k(\xi\,|\boldsymbol{x}, \boldsymbol{x}^{(i)})$

$$f(\boldsymbol{x}) = \sum_{k_n=0}^{N_n} w_{l,k_n}^{(n)} \sigma_{n,k_n}\left(\sum_{k_{n-1}=0}^{N_{n-1}} w_{k_n,k_{n-1}}^{(n-1)} \sigma_{n-1,k_{n-1}}\left(\dots \sum_{k_1=0}^{N_1} w_{k_2k_1}^{(2)} \sigma_{1,k_1}\left(\sum_{i=0}^{d} w_{k_1 i}^{(1)} x_i\right)\right)\right)$$

$$\equiv \sum_{i=0}^{N_n} c_i\, \theta_i(\boldsymbol{W}|\boldsymbol{x})$$

$$\theta_i(\boldsymbol{x}) = \theta_i(\boldsymbol{x}_0) + \nabla\theta_i(\boldsymbol{x}_0)\Delta\boldsymbol{x} + \frac{1}{2}(\Delta\boldsymbol{x})^T \boldsymbol{H}\,\Delta\boldsymbol{x} + \cdots$$

$$f_{GPRNN}(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \sigma_i(\boldsymbol{w_i}\boldsymbol{x} + b_i) \equiv \sum_{m=1}^{M} b_m k\left(\boldsymbol{y}, \boldsymbol{y}^{(m)}\right) =$$

$$\sum_{i=1}^{N}\sum_{m=1}^{M} b_m k_i\left(y_i, y_i^{(m)}\right) = \sum_{i=1}^{N} f_i(y_i)$$

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

Approaches expressive power of deep NN in diadic approximation

$$f_n(y_n(\boldsymbol{x})) = f_n(y_n(\boldsymbol{x}_0)) + \left(\boldsymbol{w}_n \left.\frac{df_n}{dy_n}\right|_{y_{n0}}\right)\Delta\boldsymbol{x} + \frac{1}{2}(\Delta\boldsymbol{x})^T\left(\boldsymbol{w}_n\boldsymbol{w}_n^T \left.\frac{d^2 f_n}{dy_n^2}\right|_{y_{n0}}\right)\Delta\boldsymbol{x} + \cdots$$

arXiv:2509.08457v1

EASE OF BUILDING AN ORDERS-OF-COUPLING (HDMR) REPRESENTATION

0.5	0.5	0.5	0	0
0.75	0.25	0.75	0	0
0.25	0.75	0.25	0	0
0.375	0.375	0.625	0	0
0	0.5	0.5	0.5	0
0	0.75	0.25	0.75	0
0	0.25	0.75	0.25	0
0	0.375	0.375	0.625	0
0	0	0.5	0.5	0.5
0	0	0.75	0.25	0.75
0	0	0.25	0.75	0.25
0	0	0.375	0.375	0.625
0	0.5	0	0.5	0.5
0	0.75	0	0.25	0.75
0	0.25	0	0.75	0.25
0	0.375	0	0.375	0.625
0.5	0	0	0.5	0.5
0.75	0	0	0.25	0.75
0.25	0	0	0.75	0.25
0.375	0	0	0.375	0.625

…

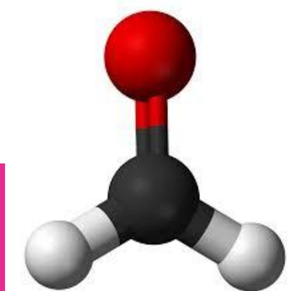
$$f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \sigma_i(\boldsymbol{w_i x} + b_i) = \sum_{i=1}^{N} f_i(y_i)$$

$$y_n = \boldsymbol{x}^T \boldsymbol{s}_n$$

D-dimensional vector with

- *d* elements of *d-dimensional* Sobol sequence
- *D* - *d* elements are zero

Order of coupling d	No. of coupling terms	Test rmse, cm^{-1} (this work)	Test rmse, cm^{-1} (Ref. [44])
1	6	1302.7	1315.6
2	15	385.5	410.0
3	20	20.4	29.1
4	**15**	**14.6**	**16.1**
5	**6**	**16.5**	**14.4**
6	1	19.6	23.8



When data density is low, full-D terms may not be recoverable

Artificial Intelligence Chemistry, **1**, 100013 (2023)

CONCLUSIONS

- ML methods are useful in high-dimensional problems as they avoid direct product representations, but when data density is very low or dimensionality is high, off-the-shelf methods may fail
  - Many non-linear parameters in NNs that also scale with D
  - Data are always sparse in high-D and high-order coupling terms (full-D function) may not be recoverable - Cannot be palliated by just adding more data
- GPR/KRR are attractive as they combine high expressive power of nonlinear kernel and robustness of linear regression
  - Can collapse in very high D
  - Loss of kernel resolution / property of locality of Matern kernels, loss of advantage of multi-ζ bases
  - Loss of advantage vs low-order polynomial regression
- One can stabilize the solution by using orders of coupling representations
  - Low-order (low-*d*) terms are easier to build (fit) and to use
  - It is convenient to use machine-learned component functions: HDMR-NN, HDMR-GPR
  - Low-order HDMR helps impute missing data and optimize hyperparameters
- It is possible to combine advantages of NN and GPR:
  - GPRNN: NN in original feature space, additive GPR in redundant coordinates
    - No nonlinear optimization
      - The shapes of all neuron activation functions are gotten in one go from 1d-HDMR-GPR
      - Optimal for given data, weights, and NN size
    - Provides an easy way to build orders-of coupling representations
    - With optimized redundant coordinates conceptually approaches expressive power of deep NN
- We provided examples from computational chemistry but **these approaches are general**

THANKS

Select collaborators, group members

Manabu Ihara
Johann Lüder
Tucker Carrington
Dmitry Voytsekhovsky
Owen Ren
Pavlo Golub
Eita Sasaki
Shunsaku Tsuda
Hyojae Lee
Heejoo Yoon
Mohamed Boussaidi

Funders

NSERC
Mitacs
Ministry of Education of Singapore
JST
Dlab
Digital Research Alliance of Canada

APPENDIX

RECAP OF SOME MAJOR ML METHODS: NEURAL NETWORKS

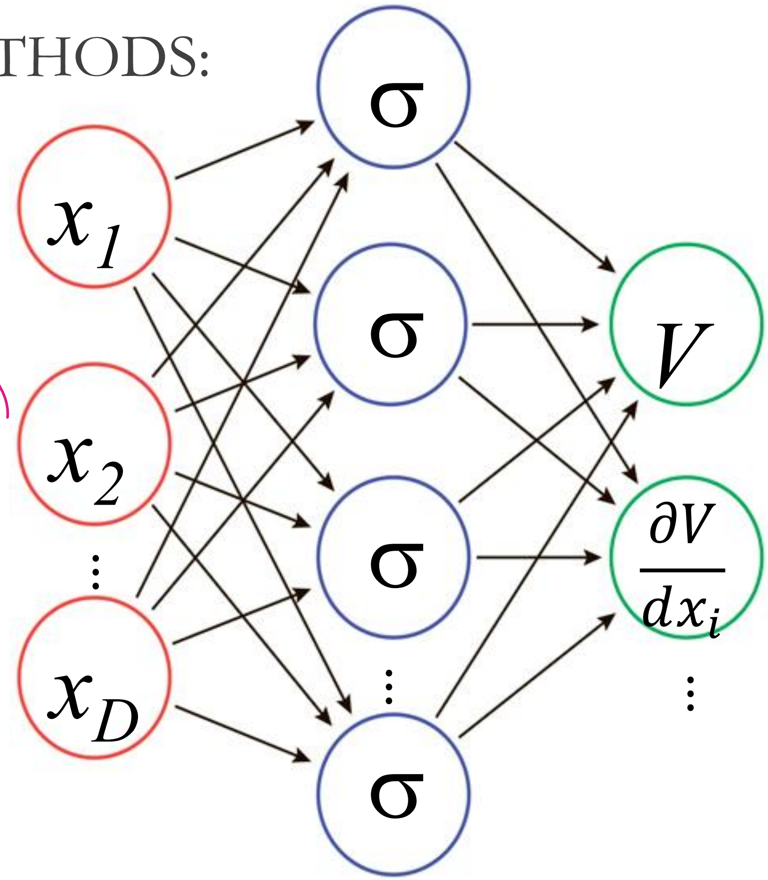
weights

biases

$$f_k(\boldsymbol{x}) = \sum_{n=0}^{N} c_{nk}\,\sigma_n(\boldsymbol{w}_n \boldsymbol{x} + b_n)$$

Training set

$$\{f^{(j)}, \boldsymbol{x}^{(j)}\}, j = 1, \dots, M; \;\; \boldsymbol{x} \in R^D$$

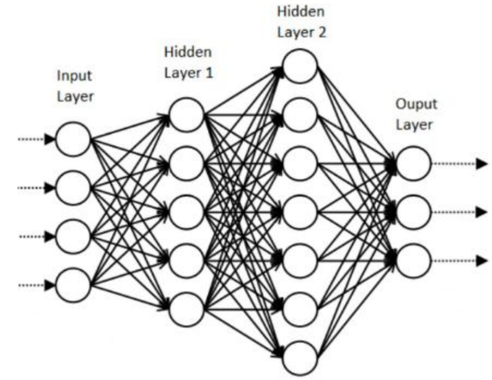


Often $\sigma(x) = (e^x - e^{-x})/(e^x + e^{-x})$
but can be **any smooth nonlinear** function

Key works by *Kolmogorov, Specher, Hornik, Gorban*

- An analogy with a biologic neural network is often made
- As a mathematical object used for non-linear regression, a view of a representation in a **non-direct product basis** is useful
- A single hidden layer NN is a universal approximator
  - Think carefully if you actually need a "deep" NN as it comes at a price of a large no. of non-linear parameters

PROS AND CONS OF FFNN



Int J Quant Chem **2015,** *115,* 1012; *Chem Rev* **2021,** *121,* 10187

"Deep" NN:

Non-direct product basis

$$f_l = \sigma_{out}\left(\sum_{k_n=0}^{N_n} w_{l,k_n}^{(n)} \sigma_{n,k_n}\left(\sum_{k_{n-1}=0}^{N_{n-1}} w_{k_n,k_{n-1}}^{(n-1)} \sigma_{n-1,k_{n-1}}\left(\dots \sum_{k_1=0}^{N_1} w_{k_2 k_1}^{(2)} \sigma_{1,k_1}\left(\sum_{i=0}^{d} w_{k_1 i}^{(1)} x_i\right)\right)\right)\right)$$

$$\sigma = \exp(\quad): \; f(\boldsymbol{x}) = \sum_{i=0}^{M} w_i^{(2)} \prod_{k=0}^{d} e^{w_{ik}^{(1)} x_k}$$

$$\int f(\boldsymbol{x}) d\boldsymbol{x} = \sum \prod \int dx_i \, f_i(x_i) dx_i$$

J Chem Phys **2006,** *125*, 194105

Refs in color are ours

$$f(\boldsymbol{x}) = \mu_1^{(2)} + \sum_{k_1=1}^{n_1} w_{1,k_1}^{(2)} \prod_{k_0=1}^{J} erf\left(\mu_{k_1 k_0}^{(1)} + w_{k_1 1 k_o}^{(1)} x_{k_0}\right)$$

Neural Computation **2001,** *14*, 241

Refs in black are literature

- A single hidden layer NN is a universal approximator
  - Think carefully if you actually need a deep NN as it comes at a price of a **large no. of non-linear parameters**
- Easy to achieve sum-of-products
  - Important for multidimensional integration

$$\forall \delta > 0, \exists N < \infty: \quad \left| t(\boldsymbol{x}) - \sum_{n=0}^{N} c_n \sigma(\boldsymbol{w_n x} + b_n) \right| < \delta$$

Universal approximator theorems do not consider the data aspect

GPR/KRR:
LINEAR REGRESSION WITH A NONLINEAR BASIS

what is the expected value of f at $\boldsymbol{x}$ given the set $\{f^{(n)}, \boldsymbol{x}^{(n)}\}$?

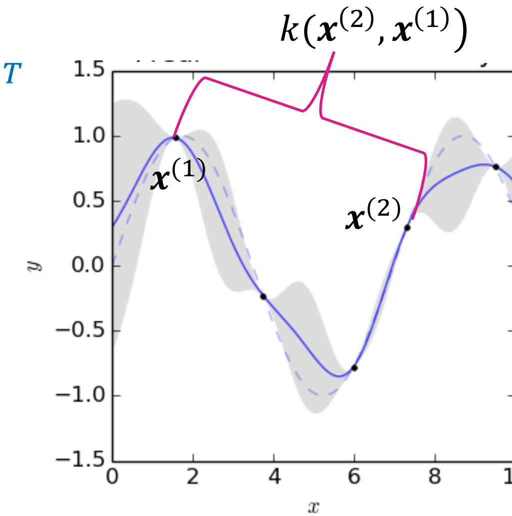
$$f(\boldsymbol{x}) = \boldsymbol{K}^*(\boldsymbol{K}^{-1}\boldsymbol{f}) = \sum_{n=1}^{M} b_n k\left(\boldsymbol{x}, \boldsymbol{x}^{(n)}\right)$$

$$var\left(f(\boldsymbol{x})\right) = K^{**} - \boldsymbol{K}^*\boldsymbol{K}^{-1}\boldsymbol{K}^{*T}$$

Matrix $\boldsymbol{K}$ can describe how correlated is each pair of data points

$$\boldsymbol{K} = \begin{pmatrix} k\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}\right) + \delta & k\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) & \cdots & k\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(M)}\right) \\ k\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right) & k\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)}\right) + \delta & & k\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(M)}\right) \\ \vdots & & \ddots & \vdots \\ k\left(\boldsymbol{x}^{(M)}, \boldsymbol{x}^{(1)}\right) & k\left(\boldsymbol{x}^{(M)}, \boldsymbol{x}^{(2)}\right) & \cdots & k\left(\boldsymbol{x}^{(M)}, \boldsymbol{x}^{(M)}\right) + \delta \end{pmatrix}$$

$$\boldsymbol{K}^* = \left(k\left(\boldsymbol{x}, \boldsymbol{x}^{(1)}\right) \quad k\left(\boldsymbol{x}, \boldsymbol{x}^{(2)}\right) \quad \ldots \quad k\left(\boldsymbol{x}, \boldsymbol{x}^{(M)}\right)\right) \qquad \boldsymbol{K}^{**} = k(\boldsymbol{x}, \boldsymbol{x})$$



Matern family of functions:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{|\boldsymbol{x} - \boldsymbol{x}'|}{l}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{|\boldsymbol{x} - \boldsymbol{x}'|}{l}\right)$$

particular cases: exp, Gaussian

https://en.wikipedia.org/wiki/Matern_covariance_function

hyperparameters

Optimized l_i informs on relevance of the i-th variable (ARD, automated relevance determination)

$$\nu = \infty\text{: } (RBF)\text{: } \sigma^2 exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{x}'|^2}{2l^2}\right)$$

$$\nu = \frac{1}{2}\text{: } \sigma^2 exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{x}'|}{l}\right)$$

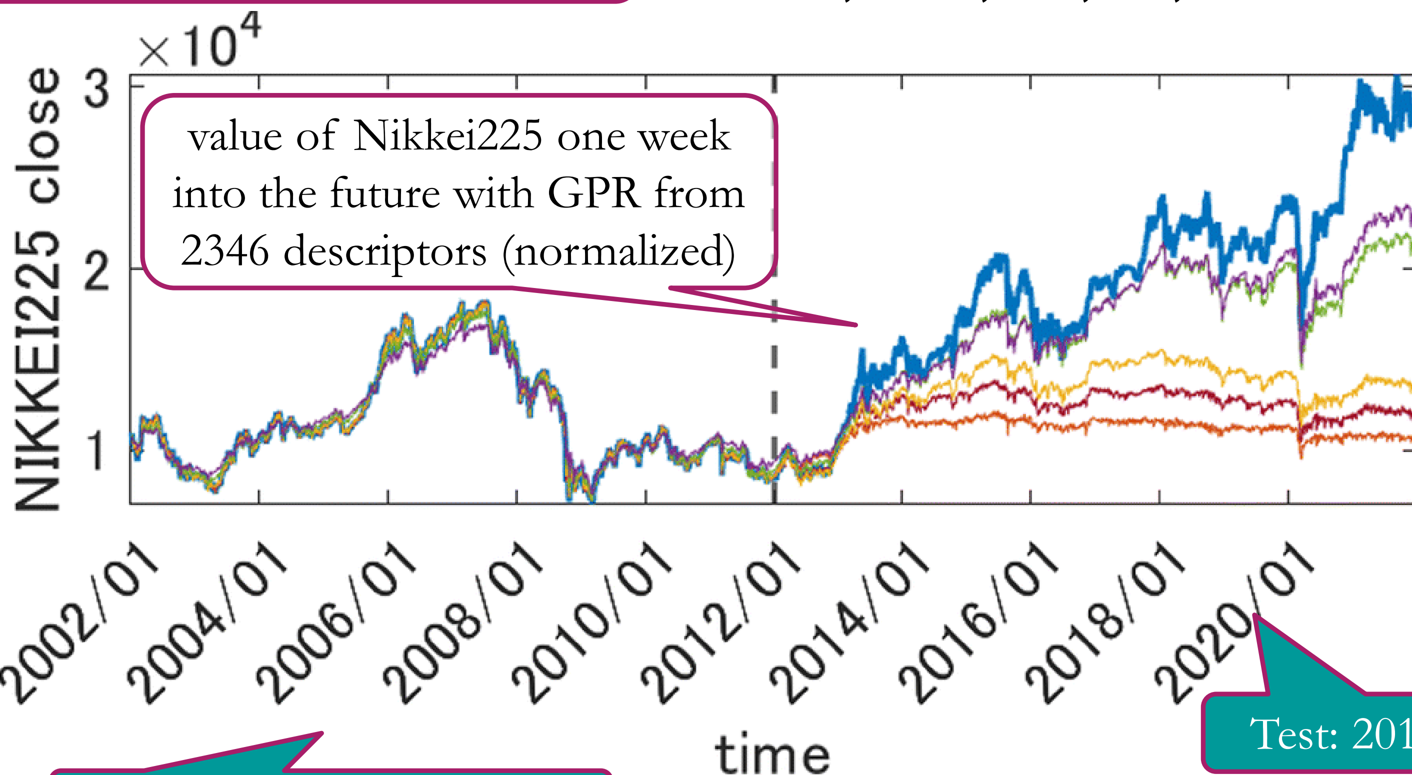
- Robust (**a type of linear regression**) but
  - Hard to apply to large datasets
  - Defining eqs do not scale with D but can collapse in very high-D (*Phys Chem Chem Phys* **25** (2023) 1546)

Can be palliated with additive GPR (HDMR)

HDMR AVOIDS COLLAPSE OF GPR IN VERY HIGH-D

kernel length parameters small enough to preserve the nonlinear nature of the model

$l = 0.1$, $l = 0.5$, $l = 1$, $l = 5$, $l = 10$. Blue: target



Training: 2002-2011

Test: 2012-2020

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \exp\left(-\left(\frac{x_d - x'_d}{l}\right)^2\right)$$

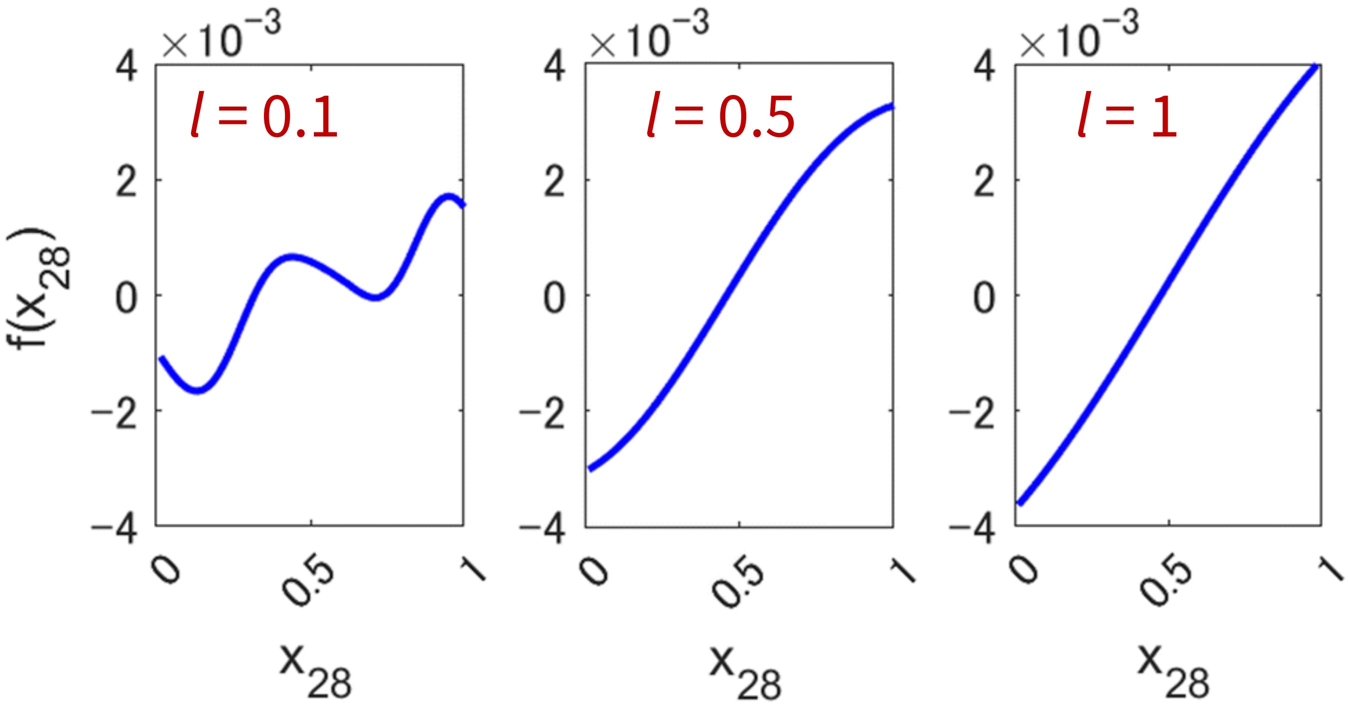
Phys. Chem. Chem. Phys. **25**, 1546 (2023)

HDMR ABLE TO RECOVER THE VALUE OF NONLINEARITY

$f_i(x_i)$ **in the example of N225 forecasting**

$$f(\boldsymbol{x}^*) = f_1(x_1^*) + \cdots + f_D(x_D^*)$$

compare to linear regression using a basis: $f(\boldsymbol{x}) = \sum_{i=1}^{D} c_i \theta_i(x_i)$

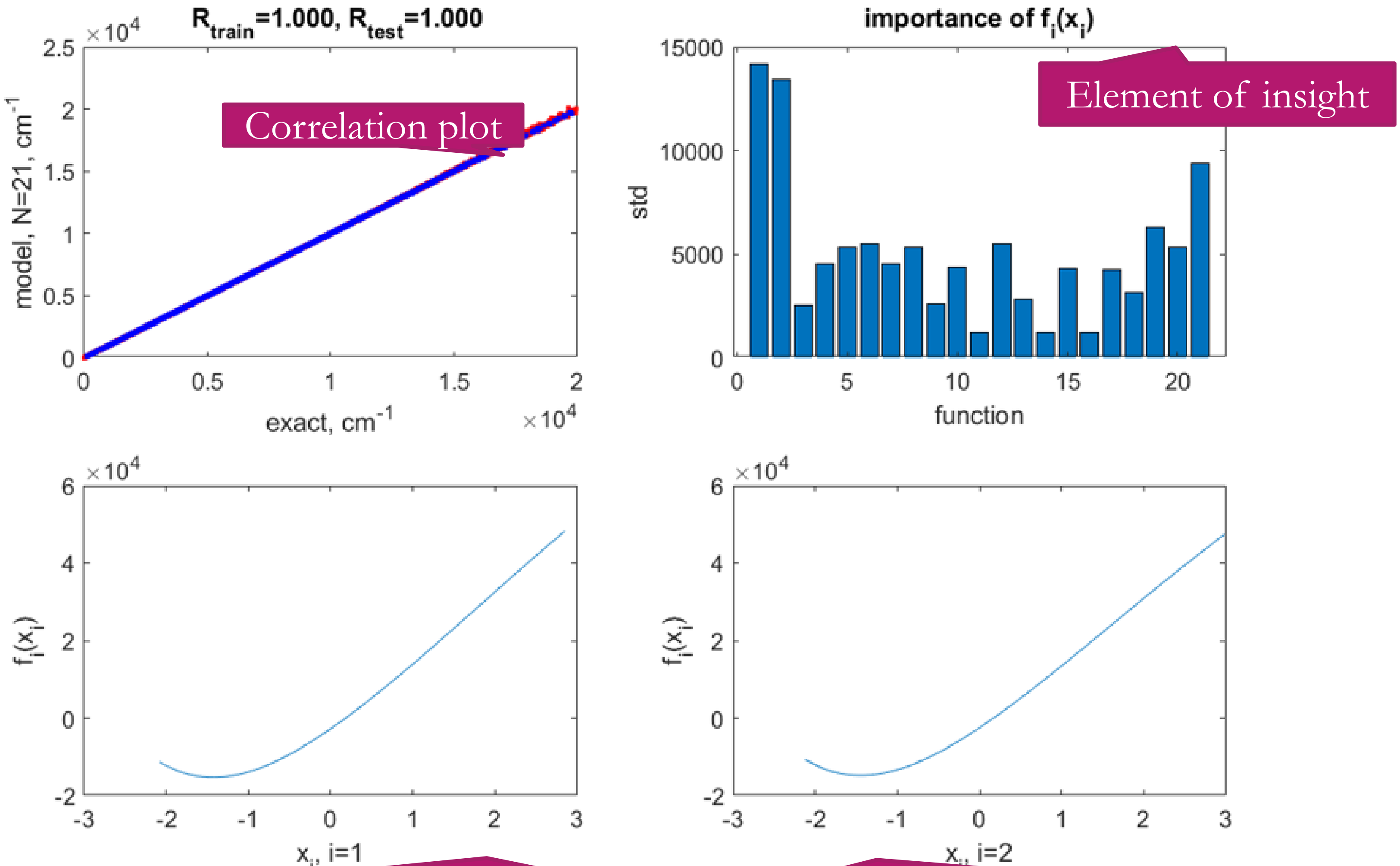


With $D = 2346$ even 2nd order terms ($f_{ij}(x_i, x_j)$) are hard

$$\sum_{i_1, i_2, \ldots, i_d}^{D} f_{i_1, i_2, \ldots, i_d}(x_{i_1}, x_{i_2}, \ldots, x_{i_d})$$

… preserving a higher expressive power of a nonlinear model

NN WITH OPTIMAL NN NEURONS BUILT WITH 1ST ORDER HDMR-GPR



Two neuron activation functions for terms with the largest magnitude

J Phys Chem A **127**, 7823 (2023)