Random Feature Expansion for Surrogate Modelling

appliedAI Seminar

Maternus Herold

07.07.2022

INITIATIVE FOR
APPLIED ARTIFICIAL
INTELLIGENCE

Agenda

1. Motivation

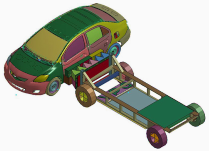
2. Random Feature Expansion

3. Surrogate Modellierung using Sparse Random Features

Motivation

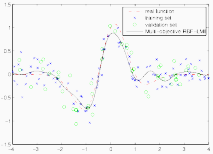
Why Surrogate Modelling is Useful

Basis Model



- e.g. Simulations, $\mathbf{y} = \mathcal{M}(\mathbf{X})$
- very accurate and complex
- long compute time per cycle
- hard to understand/analysis

Surrogate Model

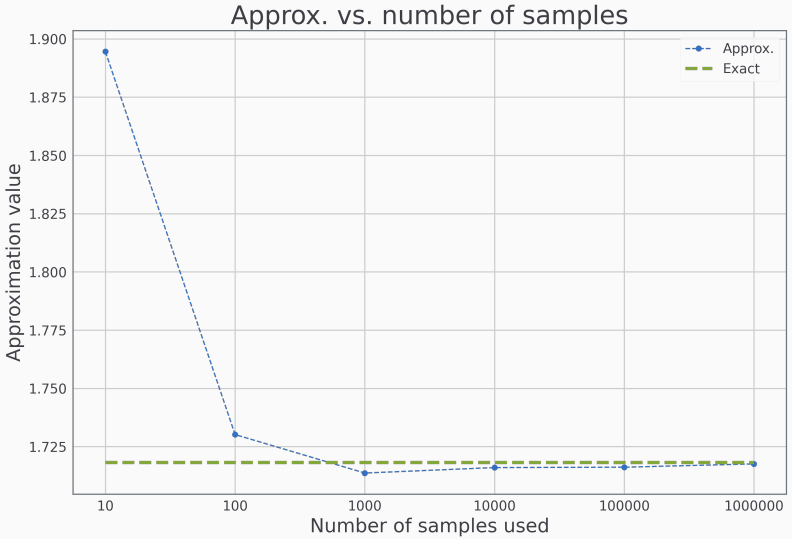


- Approx. of the model, $f_\theta(\mathbf{X}) \approx \mathcal{M}(\mathbf{X})$
- highly performant
- allows statistical analyses
- inaccuracies

Surrogate Modelling in High Dimensions is Difficult

- **Target:** Uncertainty Quantification or Sensitivity/Robustness Analysis
- Generating samples is expensive $\rightarrow$ limiting model complexity
- High dimensional domain
- $\rightarrow$ Curse of Dimensionality

Higher dimensions require more samples for a good estimation



Approaches for Surrogate Modelling in High Dimensions

Specialized Algorithms

- Regularized models
- Sparse Polynomial Chaos Expansion
- Small Neural Networks

Dimensionality Reduction

- Reduce input dimension and model in reduced space
- (Kernel) PCA, Encoder, ...
- But data is **unstructured**

Proposed self-supervised algorithm

Algorithm 1: Self-supervised projection of $\mathbf{X}$ into a lower dimension and fitting a surrogate model on that representation.

$\theta_0 \leftarrow \mathcal{U}[0,1]^d$
$k \in \mathbb{N}, \quad k \ll d$
iterations $\leftarrow \mathbb{N}$

for step : iterations **do**
    $d_{\text{red}} \leftarrow \phi(\theta_{\text{step}}, k, d)$, $d_{\text{red}} \in \mathbb{R}^k$
    $f \leftarrow \Psi(d_{\text{red}})$
    $\hat{\epsilon} \leftarrow \mathcal{L}(y, \hat{f})$
    $\theta_{\text{step}+1} \leftarrow \text{PSO}(\hat{\epsilon}, \theta_{\text{step}})$
end for

return $\theta, \hat{\epsilon}$

Proposed self-supervised algorithm

Algorithm 1: Self-supervised projection of $\mathbf{X}$ into a lower dimension and fitting a surrogate model on that representation.

$\theta_0 \leftarrow \mathcal{U}[0,1]^d$
$k \in \mathbb{N}, \quad k \ll d$
iterations $\leftarrow \mathbb{N}$

for step : iterations **do**
    $d_{\text{red}} \leftarrow \phi(\theta_{\text{step}}, k, d)$, $d_{\text{red}} \in \mathbb{R}^k$
    $f \leftarrow \Psi(d_{\text{red}})$
    $\hat{\epsilon} \leftarrow \mathcal{L}(y, \hat{f})$
    $\theta_{\text{step}+1} \leftarrow \text{PSO}(\hat{\epsilon}, \theta_{\text{step}})$
end for

return $\theta, \hat{\epsilon}$

Random Feature Expansion

Motivation of Random Feature Expansions

Risk minimization problems are common in Machine Learning:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{N} \mathcal{L}(y_i, f_i(x_i)) + \lambda J(f), \tag{1}$$

and by the Representer Theorem [5]

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \tag{2}$$

Motivation of Random Feature Expansions

Risk minimization problems are common in Machine Learning:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{N} \mathcal{L}(y_i, f_i(x_i)) + \lambda J(f), \tag{1}$$

and by the Representer Theorem [5]

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \tag{2}$$

BUT kernel methods do not scale well as N gets large!

Theory on RKHS and Random Features

maybe next time ... 😉

Kernel Approximation using Random Features

Idea: randomized map $\mathbf{z} : \mathbb{R}^D \mapsto \mathbb{R}^R$

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}} \approx z(\mathbf{x})^T z(\mathbf{y}) \tag{3}$$

Kernel Approximation using Random Features

Idea: randomized map $\mathbf{z} : \mathbb{R}^D \mapsto \mathbb{R}^R$

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}} \approx z(\mathbf{x})^T z(\mathbf{y}) \tag{3}$$

The minimizer reads:

$$\begin{aligned} f^*(\mathbf{x}) &= \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) \\ &= \sum_{i=1}^{N} \alpha_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle_{\mathcal{H}} \\ &\approx \sum_{i=1}^{N} \alpha_i z(\mathbf{x})^T z(\mathbf{x}_i) \\ &= \mathbf{c}^T z(\mathbf{x}) \end{aligned} \tag{4}$$

How could such an approx. look like?

Take $h : \mathbf{x} \mapsto \exp(i\omega^T \mathbf{x}),\ \mathbf{x} \in \mathbb{R}^d$ where $\omega \sim \mathcal{N}_d(\mathbf{0}, \mathbf{I})$

How could such an approx. look like?

Take $h : \mathbf{x} \mapsto \exp(i\omega^T \mathbf{x}),\ \mathbf{x} \in \mathbb{R}^d$ where $\omega \sim \mathcal{N}_d(\mathbf{0}, \mathbf{I})$

$$\begin{aligned} \mathbb{E}_\omega \left[h(\mathbf{x})h(\mathbf{y})^*\right] &= \mathbb{E}_\omega \left[\exp(i\omega^T(\mathbf{x}-\mathbf{y}))\right] \\ &= \int_{\mathbb{R}} \rho(\omega) \exp(i\omega^T(\mathbf{x}-\mathbf{y}))d\omega \\ &= \exp(-\frac{1}{2}(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})) \end{aligned} \tag{5}$$

Generalisation using Bochner's Theorem

Theorem (Bochner's Theorem)

A continuous kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ on $\mathbb{R}^d$ is pd. iff $k(\delta)$ is the Fourier transform of a non-negative measure [4].

$$k(\delta) = \int_{\mathbb{R}^d} \rho(\omega) \exp(i\omega^T \delta) d\omega = \mathbb{E}_\omega \left[h(\mathbf{x}) h(\mathbf{y})^* \right] \tag{6}$$

Generalisation using Bochner's Theorem

Theorem (Bochner's Theorem)

A continuous kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ on $\mathbb{R}^d$ is pd. iff $k(\delta)$ is the Fourier transform of a non-negative measure [4].

$$k(\delta) = \int_{\mathbb{R}^d} \rho(\omega) \exp(i\omega^T \delta) d\omega = \mathbb{E}_\omega \left[h(\mathbf{x}) h(\mathbf{y})^* \right] \tag{6}$$

If $k(\delta)$ is properly scaled, then $\rho(\omega)$ is a probability measure. And $h(\cdot)h(\cdot)^*$ is an unbiased estimator of $k(\mathbf{x}, \mathbf{y})$.

Representing a shift-invariant kernel by a randomly parametrized map

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{y}) &= k(\delta) \\
&= \int_{\mathbb{R}^d} \rho(\omega) \exp(i\omega^T(\delta)) d\omega \\
&= \mathbb{E}_\omega \left[\exp(i\omega^T(\delta)) \right] \\
&\approx \frac{1}{R} \sum_{i=1}^{R} \exp(i\omega_i^T(\delta)) \\
&= \mathbf{h}(\mathbf{x})\mathbf{h}(\mathbf{y})^*
\end{aligned}
\tag{7}
$$

Deriving real-valued Random Fourier Features

Using $\exp(i\omega^T \mathbf{x}) \to \cos(\omega^T \mathbf{x}) \in \mathbb{R}$

Then take

$$
\begin{aligned}
\omega &\sim \rho(\omega) \\
b &\sim \mathcal{U}(0, 2\pi) \\
z_\omega(\mathbf{x}) &= \sqrt{2}\cos(\omega^T \mathbf{x} + b) \\
\mathbf{z}(\mathbf{x}) &= \left[\frac{1}{\sqrt{R}} z_{\omega_1}(\mathbf{x}), \ldots, \frac{1}{\sqrt{R}} z_{\omega_R}(\mathbf{x})\right]^T
\end{aligned}
$$

Example: Approximating the Kernel matrix

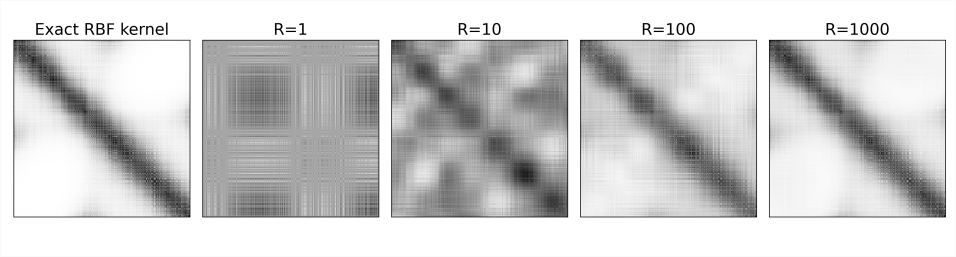


Abbildung 1: Approximating the Kernel matrix $\mathbf{K} \in \mathbb{R}^{1000 \times 1000}$ of the RBF kernel with $\gamma = \frac{1}{2}$ on the scikit-learn curves dataset[2]. The normalized deviations between $\mathbf{K}$ and its approximations: 2.54, 0.82, 0.37 and 0.09.

Low-order functions

Definition (Order-q Functions[1])

Given a linear function $f : \mathbb{R}^d \to \mathbb{C}$. f is said to be an order-q function of at most K terms, if there exist K such functions $g_1, \ldots, g_K : \mathbb{R}^q \to \mathbb{C}$, where $q \ll d, q \in \mathbb{N}$, such that

$$f(\mathbf{x}) = \sum_{j=1}^{K} g_j(\mathbf{x}|_{\mathcal{S}_j}) = \sum_{j=1}^{K} g_j(x_{1_j}, \ldots, x_{q_j}).$$

$\mathcal{S}_j$ is a subset of the index set $[d]$ and $\mathbf{x}|_{\mathcal{S}_j}$ is the restriction of the input vector onto the subset.

That is, the function f can be represented using K linear terms, each of which depends on q of the d variables.

[1] The definition is in accordance to Hashemi et al. [1] with minor adaption.

Sparse Random Feature Expansion

Algorithm 2 Sparse Random Feature Expansion with Sparse Feature Weights (SRFE-S)

1: **Input:** parametric basis function $\phi(\mathbf{x};\boldsymbol{\omega}) = \phi(\langle \mathbf{x}, \boldsymbol{\omega} \rangle)$, feature sparsity level q, probability density $\zeta : \mathbb{R}^q \to \mathbb{R}$, stability parameter η.

2: Draw m data points $\mathbf{x}_k \sim \mathcal{D}_x$ and observe outputs $y_k = f(\mathbf{x}_k) + e_k$ with $|e_k| \leq E$.

3: Draw a complete set of N q-sparse feature weights $\boldsymbol{\omega}_j \in \mathbb{R}^d$ sampled from density $\zeta : \mathbb{R}^q \to \mathbb{R}$ as defined in Definition 1.

4: Construct a random feature matrix $\mathbf{A} \in \mathbb{C}^{m \times N}$ such that $a_{kj} = \phi(\mathbf{x}_k; \boldsymbol{\omega}_j)$.

5: Solve

$$\mathbf{c}^\sharp = \arg\min_{\mathbf{c}} \quad \|\mathbf{c}\|_1 \qquad \text{s.t.} \qquad \|\mathbf{A}\mathbf{c} - \mathbf{y}\| \leq \eta\sqrt{m}.$$

6: **Output:** Form the approximation

$$f^\sharp(\mathbf{x}) = \sum_{j=1}^{N} c_j^\sharp \phi(\mathbf{x}; \boldsymbol{\omega}_j).$$

Surrogate Modellierung using Sparse Random Features

Proposed self-supervised algorithm

Algorithm 2: Self-supervised projection of $\mathbf{X}$ into a lower dimension and fitting a surrogate model on that representation.

$\theta_0 \leftarrow \mathcal{U}[0,1]^d$
$k \in \mathbb{N}, \quad k \ll d$
iterations $\leftarrow \mathbb{N}$

for step : iterations **do**
    $d_{\text{red}} \leftarrow \phi(\theta_{\text{step}}, k, d)$, $d_{\text{red}} \in \mathbb{R}^k$
    $f \leftarrow \Psi(d_{\text{red}})$
    $\hat{\epsilon} \leftarrow \mathcal{L}(y, \hat{f})$
    $\theta_{\text{step}+1} \leftarrow \text{PSO}(\hat{\epsilon}, \theta_{\text{step}})$
end for

return $\theta, \hat{\epsilon}$

Experiment Setup

Data generation via Sobol function:

$$y_i = \prod_j^d \frac{|4x_i - 2| + c_i}{1 + c_i}, \ \mathbf{c} = \{1, 2, 5, 10, 20, 100, 5 \cdot 10^3, \ldots, 5 \cdot 10^3\}$$

Data split:

$$\mathbf{X}_{\text{train}} \in \mathbb{R}^{800 \times 20}, \ \mathbf{X}_{\text{val}} \in \mathbb{R}^{1000 \times 20}$$

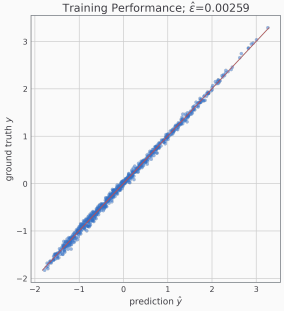
Methods:

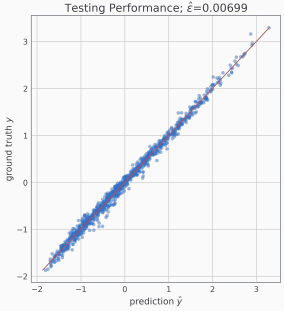
ϕ_θ : Kernel PCA, anisotropic kernel

f_ω : Polynomial Chaos Expansion, $f_\omega(\mathbf{x}) = \sum_{a \in A} c_a \psi_a(\mathbf{x})$

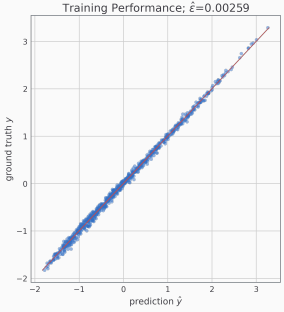
f_ω : Random Feature Expansion, $f_\omega^\# = \sum_j^N c_j^\# \phi(\mathbf{x}, \omega_j)$

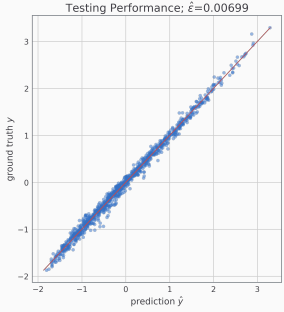
Results

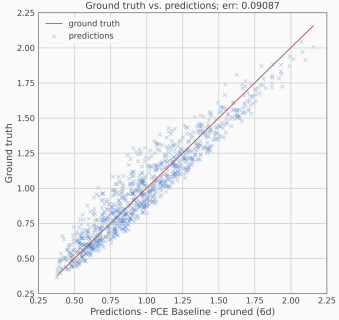

Training Performance; $\hat{\epsilon}$=0.00259
ground truth y
prediction $\hat{y}$
−2
−1
0
1
2
3


Testing Performance; $\hat{\epsilon}$=0.00699
ground truth y
prediction $\hat{y}$
−2
−1
0
1
2
3

Results


Training Performance; $\hat{\epsilon}$=0.00259
ground truth y
prediction $\hat{y}$


Testing Performance; $\hat{\epsilon}$=0.00699
ground truth y
prediction $\hat{y}$


Ground truth vs. predictions; err: 0.09087
ground truth
predictions
Ground truth
Predictions - PCE Baseline - pruned (6d)

What else can be done with Random Features

Random Fourier Attention [3]

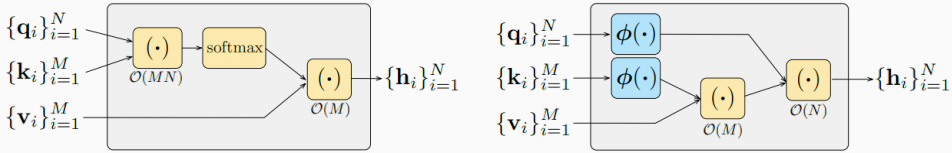


Abbildung 2: Approximating Softmax Attention

Random Feature Expansion for Surrogate Modelling

appliedAI Seminar

Maternus Herold

07.07.2022

INITIATIVE FOR
APPLIED ARTIFICIAL
INTELLIGENCE

Backup

Sparse Random Feature Expansion

Algorithm 2 Sparse Random Feature Expansion with Sparse Feature Weights (SRFE-S)

1: **Input:** parametric basis function $\phi(\mathbf{x};\boldsymbol{\omega}) = \phi(\langle \mathbf{x}, \boldsymbol{\omega} \rangle)$, feature sparsity level q, probability density $\zeta : \mathbb{R}^q \to \mathbb{R}$, stability parameter η.

2: Draw m data points $\mathbf{x}_k \sim \mathcal{D}_x$ and observe outputs $y_k = f(\mathbf{x}_k) + e_k$ with $|e_k| \leq E$.

3: Draw a complete set of N q-sparse feature weights $\boldsymbol{\omega}_j \in \mathbb{R}^d$ sampled from density $\zeta : \mathbb{R}^q \to \mathbb{R}$ as defined in Definition 1.

4: Construct a random feature matrix $\mathbf{A} \in \mathbb{C}^{m \times N}$ such that $a_{kj} = \phi(\mathbf{x}_k; \boldsymbol{\omega}_j)$.

5: Solve

$$\mathbf{c}^\sharp = \arg\min_{\mathbf{c}} \quad \|\mathbf{c}\|_1 \qquad \text{s.t.} \qquad \|\mathbf{A}\mathbf{c} - \mathbf{y}\| \leq \eta\sqrt{m}.$$

6: **Output:** Form the approximation

$$f^\sharp(\mathbf{x}) = \sum_{j=1}^{N} c_j^\sharp \phi(\mathbf{x}; \boldsymbol{\omega}_j).$$

Sparse Random Feature Expansion

Given f be a q-order function with all $g_l \in \mathcal{F}(\phi, \rho)$, $l \in \{1, 2, \ldots, K\}$ where $\phi = \exp(\langle \mathbf{x}, \omega \rangle i)$. [..] Let $\omega_1, \ldots, \omega_N \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_q)$ and $\mathbf{x}_1, \ldots, \mathbf{x}_m \sim \mathcal{N}(\mathbf{0}, \gamma^2 \mathbf{I}_d)$. [..] Then with the following inequalities

1. ...
2. $N \geq \frac{4}{\epsilon^2} \left(1 + 4R\sigma\sqrt{q} + \sqrt{\frac{q}{2}\log(\frac{d}{\delta})}\right)^2$
3. $m \geq 4\left(2\gamma^2\sigma^2 + 1\right)^{\max(2q-d,0)} \left(\gamma^2\sigma^2 + 1\right)^{\min(2q,2d-2q)} \log \frac{N^2}{\delta}$
4. ...

it holds with probability at least $1 - 4\delta$:

$$\sup_{\mathbf{x} \in \mathbb{R}^d} \|f(\mathbf{x}) - f^{\#}(\mathbf{x})\|_2 \leq \epsilon \|f\|_\rho + C' \kappa_{s,1}(\tilde{\mathbf{c}}^*) + C\eta\sqrt{s} \tag{8}$$

Complexity

Sampling $\mathcal{O}(\sqrt{n}\log n)$ centers from the dataset allows error bounds of $\mathcal{O}(1/\sqrt{n})$.

Literatur

[1] Abolfazl Hashemi u. a. *Generalization Bounds for Sparse Random Feature Expansions*. 2021. arXiv: 2103.03191 [stat.ML].

[2] F. Pedregosa u. a. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[3] Hao Peng u. a. *Random Feature Attention*. 2021. URL: https://arxiv.org/abs/2103.02143.

References ii

[4] Ali Rahimi und Benjamin Recht. “Random Features for Large-Scale Kernel Machines”. In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. NIPS’07. Vancouver, British Columbia, Canada: Curran Associates Inc., 2007, S. 1177–1184. ISBN: 9781605603520.

[5] Bernhard Schölkopf, Ralf Herbrich und Alex J. Smola. “A Generalized Representer Theorem”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, S. 416–426. DOI: 10.1007/3-540-44581-1_27.