Diffeomorphism-based feature learning using Poincaré inequalities

Clémentine PRIEUR @UGA @Inria
Joint work with Romain Verdière, Olivier Zahm @Inria

Aspects mathématiques et appliqués
des méthodes de réduction de complexité

Collège de France, June, 22-24, 2026

Romain Verdière
@INRIA,LJK

Olivier Zahm
@INRIA,LJK

Outline

Model approximation

Let u be a computational model defined on an open set $\mathcal{X} \subseteq \mathbb{R}^d$:

$$u : \left\{ \begin{array}{ccc} \mathcal{X} & \longrightarrow & \mathbb{R}^p \\ \boldsymbol{X} & \longmapsto & u(\boldsymbol{X}) \end{array} \right.$$

with $\boldsymbol{X}$ a random vector with probability distribution π and $d \gg 1$.

Context

- u is computationally expensive to evaluate,
- ∇u can be evaluated for the same computational cost as an evaluation of u using the adjoint method.

This context is the one in many scientific and engineering problems.

Goal

Given a set $\mathcal{D} = (x_k, u(x_k), \nabla u(x_k))_{1 \leq k \leq n_{\text{train}}}$, build a fast to evaluate approximation model $\tilde{u}$ such that

$$\mathbb{E}[\|u(\boldsymbol{X}) - \tilde{u}(\boldsymbol{X})\|^2]$$

is small, with $\|\cdot\|$ the euclidean norm on $\mathbb{R}^p$.

Need for dimension reduction

Due to the **curse of dimensionality**, usual approximation methods are failing for n_{train} too small with respect to $d \gg 1$.

Work around: reduce the input dimension of the model by finding intrinsic low dimensional structures.

Problem formulation

We aim at building a *feature map* $g : \mathcal{X} \to \mathbb{R}^m$ and a *profile function* $f : \mathbb{R}^m \to \mathbb{R}^p$ with $m \ll d$ such that

$$\mathbb{E}[\|u(\boldsymbol{X}) - f \circ g(\boldsymbol{X})\|^2]$$

is small, that is such that $f \circ g$ approximates accurately u.

Well-known linear dimension reduction methods are: Active Subspace, Sliced Inverse Regression...

Outline

Approximation strategy

Real-valued model

Let $\mathcal{X}$ be an open subset of $\mathbb{R}^d$ and a model $u : \left\{ \begin{array}{ccc} \mathcal{X} & \longrightarrow & \mathbb{R} \\ \boldsymbol{X} & \longmapsto & u(\boldsymbol{X}) \end{array} \right.$ with $\boldsymbol{X}$ a random vector and $d \gg 1$.

Naive strategy

Build f and g jointly by minimizing the reconstruction error (in $\mathbb{L}^2$ or $\mathbb{H}^1$ norm):

$$\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2],$$

$$\mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2] + \mathbb{E}[\|\nabla u(\boldsymbol{X}) - \nabla(f \circ g)(\boldsymbol{X})\|^2].$$

Our strategy

A two-step strategy:

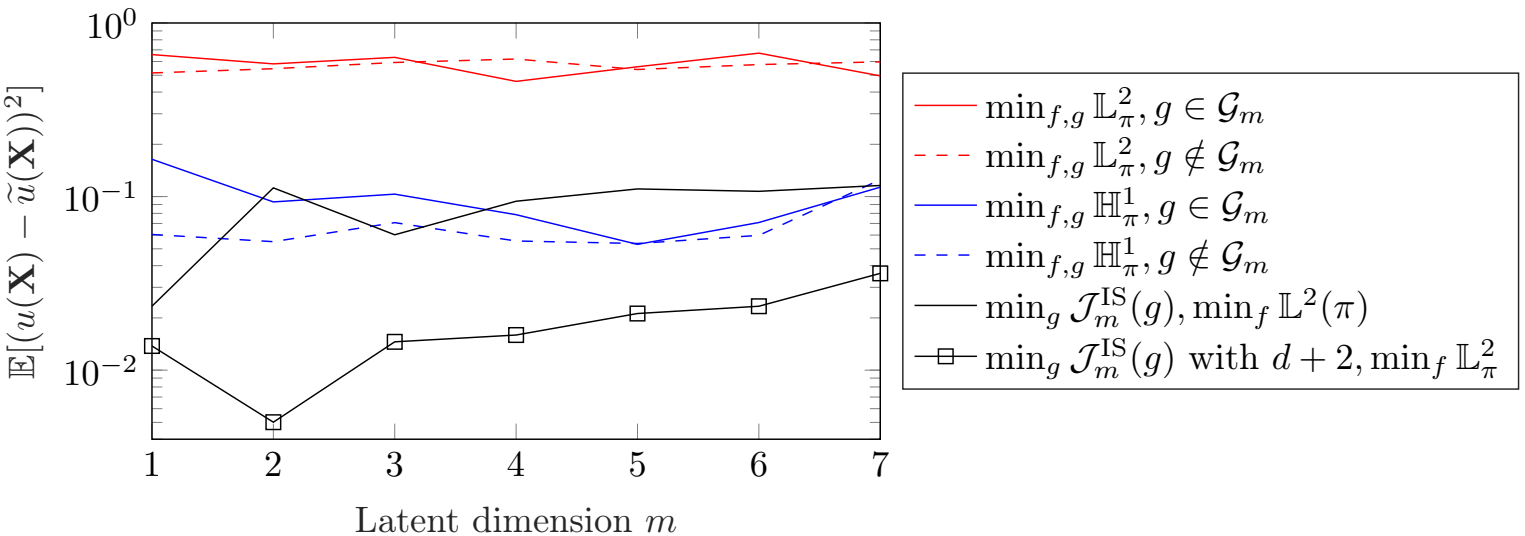
1. minimize an upper-bound of the approximation error to build g;
2. regress $u(\boldsymbol{X})$ on $\boldsymbol{Z}_m = g(\boldsymbol{X})$ to train f, where m is the latent dimension.

Approximation classes

- Approximation class for g:

$$\mathcal{G}_m(\mathcal{X}) = \left\{ \begin{array}{rccc} g: & \mathcal{X} & \to & \mathbb{R}^m \\ & x & \mapsto & (\varphi_1(x), \ldots, \varphi_m(x)) \end{array} \;\middle|\; \varphi \in \mathcal{D}(\mathcal{X}) \right\},$$

  with $\mathcal{D}(\mathcal{X})$ a class of $\mathcal{C}^1$-diffeomorphisms from $\mathcal{X}$ to $\mathbb{R}^d$.
- Approximation class for f: the class of fully connected neural networks.



Toy benchmark $u(\boldsymbol{X}) = \cos(\|\boldsymbol{X}\|^2)$ for $\boldsymbol{X} \sim \mathcal{U}([0,1]^d)$ with $d = 20$. $\mathbb{L}^2_\pi$ approximation error as a function of m. See [Verdière et al., 2023].

Input space upper bound

Note that, if $u(\mathbf{x}) = f \circ g(\mathbf{x})$, then

$$\nabla u(\mathbf{x}) = \underbrace{\nabla g(\mathbf{x})^T}_{\in \mathbb{R}^{d \times m}} \underbrace{\nabla f(g(\mathbf{x}))}_{\in \mathbb{R}^m} \Rightarrow \nabla u(\mathbf{x}) \in \text{range}(\nabla g(x)^T).$$

A natural choice for the upper bound is thus

$$\mathcal{J}_m^{\text{IS}}(g) := \mathbb{E}\left[\left\|\nabla u(\mathbf{X}) - \Pi_{\text{range}(\nabla g(\mathbf{X})^T)} \nabla u(\mathbf{X})\right\|^2\right].$$

Input space upper bound

Note that, if $u(\mathbf{x}) = f \circ g(\mathbf{x})$, then

$$\nabla u(\mathbf{x}) = \underbrace{\nabla g(\mathbf{x})^T}_{\in \mathbb{R}^{d \times m}} \underbrace{\nabla f(g(\mathbf{x}))}_{\in \mathbb{R}^m} \Rightarrow \nabla u(\mathbf{x}) \in \text{range}(\nabla g(x)^T).$$

A natural choice for the upper bound is thus

$$\mathcal{J}_m^{\text{IS}}(g) := \mathbb{E}\left[\left\|\nabla u(\mathbf{X}) - \Pi_{\text{range}(\nabla g(\mathbf{X})^T)} \nabla u(\mathbf{X})\right\|^2\right].$$

We have proven $u = f \circ g \Rightarrow \mathcal{J}_m^{\text{IS}}(g) = 0$. Is the reciprocal true?

Input space upper bound

Note that, if $u(\mathbf{x}) = f \circ g(\mathbf{x})$, then

$$\nabla u(\mathbf{x}) = \underbrace{\nabla g(\mathbf{x})^T}_{\in \mathbb{R}^{d \times m}} \underbrace{\nabla f(g(\mathbf{x}))}_{\in \mathbb{R}^m} \Rightarrow \nabla u(\mathbf{x}) \in \text{range}(\nabla g(x)^T).$$

A natural choice for the upper bound is thus

$$\mathcal{J}_m^{\text{IS}}(g) := \mathbb{E}\left[\left\|\nabla u(\mathbf{X}) - \Pi_{\text{range}(\nabla g(\mathbf{X})^T)} \nabla u(\mathbf{X})\right\|^2\right].$$

We have proven $u = f \circ g \Rightarrow \mathcal{J}_m^{\text{IS}}(g) = 0$. Is the reciprocal true?

Proposition [Verdière et al., 2023]

Let $u \in \mathcal{C}^1(\mathcal{X}; \mathbb{R})$ and $g \in \mathcal{G}_m(\mathcal{X})$. Then, $\mathcal{J}_m^{\text{IS}}(g) = 0$ if and only if there exists $f \in \mathcal{C}^1(\mathbb{R}^m; \mathbb{R})$ such that $u = f \circ g$.

This result does not generalize to any approximation class for g (see [Bigoni et al., 2022]).

Input space upper bound

Note that, if $u(\mathbf{x}) = f \circ g(\mathbf{x})$, then

$$\nabla u(\mathbf{x}) = \underbrace{\nabla g(\mathbf{x})^T}_{\in \mathbb{R}^{d \times m}} \underbrace{\nabla f(g(\mathbf{x}))}_{\in \mathbb{R}^m} \Rightarrow \nabla u(\mathbf{x}) \in \text{range}(\nabla g(x)^T).$$

A natural choice for the upper bound is thus

$$\mathcal{J}_m^{\text{IS}}(g) := \mathbb{E}\left[\left\|\nabla u(\mathbf{X}) - \Pi_{\text{range}(\nabla g(\mathbf{X})^T)} \nabla u(\mathbf{X})\right\|^2\right].$$

We have proven $u = f \circ g \Rightarrow \mathcal{J}_m^{\text{IS}}(g) = 0$. Is the reciprocal true?

Proposition [Verdière et al., 2023]

Let $u \in \mathcal{C}^1(\mathcal{X}; \mathbb{R})$ and $g \in \mathcal{G}_m(\mathcal{X})$. Then, $\mathcal{J}_m^{\text{IS}}(g) = 0$ if and only if there exists $f \in \mathcal{C}^1(\mathbb{R}^m; \mathbb{R})$ such that $u = f \circ g$.

This result does not generalize to any approximation class for g (see [Bigoni et al., 2022]).

Can we prove $u \approx f \circ g \Rightarrow \mathcal{J}_m^{\text{IS}}(g) \approx 0$?

Input space upper bound

Poincaré inequality

For $\boldsymbol{Y}$ a random vector in $\mathbb{R}^d$, the Poincaré constant $\mathbb{C}(\boldsymbol{Y})$ is defined as the smallest constant such that

$$\mathbb{E}[(h(\boldsymbol{Y}) - \mathbb{E}h(\boldsymbol{Y}))^2] \leq \mathbb{C}(\boldsymbol{Y})\mathbb{E}[\|\nabla h(\boldsymbol{Y})\|^2]$$

holds for any continuously differentiable function $h : supp(\boldsymbol{Y}) \to \mathbb{R}$. $\boldsymbol{Y}$ satisfies Poincaré inequality if it satisfies the above inequality with $\mathbb{C}(\boldsymbol{Y}) < +\infty$.

Input space upper bound

Poincaré inequality

For $\boldsymbol{Y}$ a random vector in $\mathbb{R}^d$, the Poincaré constant $\mathbb{C}(\boldsymbol{Y})$ is defined as the smallest constant such that

$$\mathbb{E}[(h(\boldsymbol{Y}) - \mathbb{E}h(\boldsymbol{Y}))^2] \leq \mathbb{C}(\boldsymbol{Y})\mathbb{E}[\|\nabla h(\boldsymbol{Y})\|^2]$$

holds for any continuously differentiable function $h : supp(\boldsymbol{Y}) \to \mathbb{R}$. $\boldsymbol{Y}$ satisfies Poincaré inequality if it satisfies the above inequality with $\mathbb{C}(\boldsymbol{Y}) < +\infty$.

Proposition [Verdière et al., 2023]

If $\mathbb{C}^{\mathrm{IS}}(\boldsymbol{X}|\mathcal{G}_m(\mathcal{X})) := \sup_{g\in\mathcal{G}_m(\mathcal{X})} \sup_{z_m\in\mathbb{R}^m} \mathbb{C}(\boldsymbol{X}|g(\boldsymbol{X}) = z_m) < +\infty$, then the reconstruction error satisfies

$$\min_{f:\mathbb{R}^m\to\mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f\circ g(\boldsymbol{X}))^2] \leq \mathbb{C}^{\mathrm{IS}}(\boldsymbol{X}|\mathcal{G}_m(\mathcal{X}))\mathcal{J}_m^{\mathrm{IS}}(g).$$

We propose to build g as the solution to: $\min_{g\in\mathcal{G}_m(\mathcal{X})} \mathcal{J}_m^{\mathrm{IS}}(g)$.

Input space upper bound

Poincaré inequality

For $\boldsymbol{Y}$ a random vector in $\mathbb{R}^d$, the Poincaré constant $\mathbb{C}(\boldsymbol{Y})$ is defined as the smallest constant such that

$$\mathbb{E}[(h(\boldsymbol{Y}) - \mathbb{E}h(\boldsymbol{Y}))^2] \leq \mathbb{C}(\boldsymbol{Y})\mathbb{E}[\|\nabla h(\boldsymbol{Y})\|^2]$$

holds for any continuously differentiable function $h : supp(\boldsymbol{Y}) \to \mathbb{R}$. $\boldsymbol{Y}$ satisfies Poincaré inequality if it satisfies the above inequality with $\mathbb{C}(\boldsymbol{Y}) < +\infty$.

Proposition [Verdière et al., 2023]

If $\mathbb{C}^{\mathrm{IS}}(\boldsymbol{X}|\mathcal{G}_m(\mathcal{X})) := \sup_{g\in\mathcal{G}_m(\mathcal{X})} \sup_{z_m\in\mathbb{R}^m} \mathbb{C}(\boldsymbol{X}|g(\boldsymbol{X}) = z_m) < +\infty$, then the reconstruction error satisfies

$$\min_{f:\mathbb{R}^m\to\mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f\circ g(\boldsymbol{X}))^2] \leq \mathbb{C}^{\mathrm{IS}}(\boldsymbol{X}|\mathcal{G}_m(\mathcal{X}))\mathcal{J}_m^{\mathrm{IS}}(g).$$

We propose to build g as the solution to: $\min_{g\in\mathcal{G}_m(\mathcal{X})} \mathcal{J}_m^{\mathrm{IS}}(g)$.

Remark Controlling $\mathbb{C}^{\mathrm{IS}}(\boldsymbol{X}|\mathcal{G}_m(\mathcal{X}))$ is a difficult task in general and is an active research field [Bonnefont and Joulin, 2022, Cattiaux and Guillin, 2022].

Feature space upper bound

Take avantage of the approximation class chosen for g to apply Poincaré Inequality in the feature space.

Choosing $g(x) = (\varphi_1(x), \ldots, \varphi_m(x))$, with φ a diffeomorphism, is meaningful if for all $z_m \in \mathbb{R}^m$, the function

$$F : \left\{ \begin{array}{lcl} \mathbb{R}^{d-m} & \to & \mathbb{R} \\ z_\perp & \mapsto & (u \circ \varphi^{-1})(z_m, z_\perp), \end{array} \right.$$

is almost constant.
To achieve this, we can minimize:

$$\mathcal{J}_m^{\mathrm{FS}}(g) := \mathbb{E}[\|\nabla F\|^2] = \mathbb{E}[\|(\nabla \varphi(x)^{-\top} \nabla u(x))_\perp\|^2].$$

Remark A similar cost function is used in [Zhang et al., 2019a] but without theoretical justification.

Feature space upper bound

Take avantage of the approximation class chosen for g to apply Poincaré Inequality in the feature space.

Choosing $g(x) = (\varphi_1(x), \dots, \varphi_m(x))$, with φ a diffeomorphism, is meaningful if for all $z_m \in \mathbb{R}^m$, the function

$$F : \left\{ \begin{array}{ccc} \mathbb{R}^{d-m} & \to & \mathbb{R} \\ z_\perp & \mapsto & (u \circ \varphi^{-1})(z_m, z_\perp), \end{array} \right.$$

is almost constant.
To achieve this, we can minimize:

$$\mathcal{J}_m^{\mathrm{FS}}(g) := \mathbb{E}[\|\nabla F\|^2] = \mathbb{E}[\|(\nabla \varphi(x)^{-\top} \nabla u(x))_\perp\|^2].$$

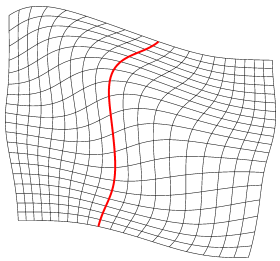
Remark A similar cost function is used in [Zhang et al., 2019a] but without theoretical justification.

Proposition [Verdière et al., 2023]

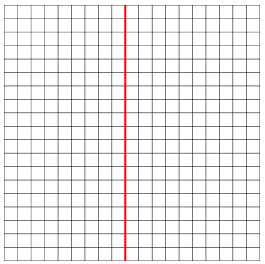
If $\mathbb{C}_m^{\mathrm{FS}}(\boldsymbol{X}|\mathcal{D}(\mathcal{X})) := \sup_{\varphi \in \mathcal{D}(\mathcal{X})} \sup_{z_m \in \mathbb{R}^m} \mathbb{C}(\varphi(\boldsymbol{X})|g(\boldsymbol{X}) = z_m) < +\infty$, then the reconstruction error satisfies

$$\min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}))^2] \leq \mathbb{C}_m^{\mathrm{FS}}(\boldsymbol{X}|\mathcal{D}(\mathcal{X}))\mathcal{J}_m^{\mathrm{FS}}(\varphi).$$

Feature space upper bound



input space



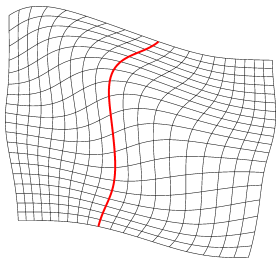
feature space

Remark

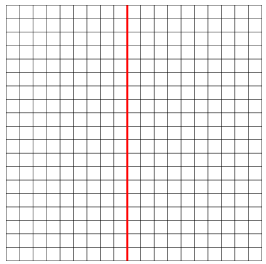
Note that $\varphi(\boldsymbol{X}) \sim \mathcal{N}(0, I_d)$ implies $\mathbb{C}_m^{\mathrm{FS}}(\boldsymbol{X}|\mathcal{D}(\mathcal{X})) = 1$. We penalize the cost function adequaltely: we build $g(x) = (\varphi_1(x), \ldots, \varphi_m(x))$ as the solution to

$$\min_{\varphi \in \mathcal{D}(\mathcal{X})} \mathcal{J}_{m,\lambda}^{\mathrm{FS}}(\varphi) := \mathbb{E}[\|(\nabla\varphi(\boldsymbol{X})^{-\top}\nabla u(\boldsymbol{X}))_{\perp}\|^2] + \lambda \underbrace{\mathbb{E}\left[\frac{\|\varphi(\boldsymbol{X})\|^2}{2} - \log|\det \nabla\varphi(\boldsymbol{X})|\right]}_{D_{\mathrm{KL}}(\varphi(\boldsymbol{X})\|\boldsymbol{Z}) + \textit{constant} \text{ with } \boldsymbol{Z}\sim\mathcal{N}(0,I_d)}.$$

Feature space upper bound



input space



feature space

Remark

Note that $\varphi(\boldsymbol{X}) \sim \mathcal{N}(0, I_d)$ implies $\mathbb{C}_m^{\mathrm{FS}}(\boldsymbol{X}|\mathcal{D}(\mathcal{X})) = 1$. We penalize the cost function adequaltely: we build $g(x) = (\varphi_1(x), \ldots, \varphi_m(x))$ as the solution to

$$\min_{\varphi \in \mathcal{D}(\mathcal{X})} \mathcal{J}_{m,\lambda}^{\mathrm{FS}}(\varphi) := \mathbb{E}[\|(\nabla\varphi(\boldsymbol{X})^{-\top}\nabla u(\boldsymbol{X}))_{\perp}\|^2] + \lambda \underbrace{\mathbb{E}\left[\frac{\|\varphi(\boldsymbol{X})\|^2}{2} - \log|\det \nabla\varphi(\boldsymbol{X})|\right]}_{D_{\mathrm{KL}}(\varphi(\boldsymbol{X})\|\boldsymbol{Z}) + \textit{constant} \text{ with } \boldsymbol{Z}\sim\mathcal{N}(0,I_d)}.$$

Note however that:

- there is no guarantees that $D_{\mathrm{KL}}(\varphi(\boldsymbol{X})\|\boldsymbol{Z}) \approx 0$, with $\boldsymbol{Z} \sim \mathcal{N}(0, I_d)$, implies $\mathbb{C}_m^{\mathrm{FS}}(\boldsymbol{X}|\mathcal{D}(\mathcal{X})) \approx 1$,
- $\mathcal{J}_{m,\lambda}^{\mathrm{FS}}$ involves hyperparameter λ that needs tuning and makes the training process more difficult.

Outline

Dimension augmentation trick

Augmenting the dimension permits to facilitate the construction of diffeomorphisms. See [Dupont et al., 2019, Zhang et al., 2019b] for neural ODEs or [Lyu et al., 2022, Huang et al., 2020] for normalizing flows.

Not all functions $g : \mathcal{X} \to \mathbb{R}^m$ are the first components of a diffeomorphism.

However, for any $h \in \mathcal{C}^1(\mathcal{X}, \mathbb{R}^m)$, feature map $g(x, \xi) = h(x) + \xi$ corresponds to the m first components of diffeomorphism $\varphi : \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}^m \times \mathcal{X}$ defined as:

$$\varphi(x, \xi) = \begin{pmatrix} h(x) + \xi \\ x \end{pmatrix} \quad \text{with} \quad \varphi^{-1}(z_1, z_2) = \begin{pmatrix} z_2 \\ z_1 - h(z_2) \end{pmatrix}.$$

Dimension augmentation trick

Augmenting the dimension permits to facilitate the construction of diffeomorphisms. See [Dupont et al., 2019, Zhang et al., 2019b] for neural ODEs or [Lyu et al., 2022, Huang et al., 2020] for normalizing flows.

Not all functions $g : \mathcal{X} \to \mathbb{R}^m$ are the first components of a diffeomorphism.

However, for any $h \in \mathcal{C}^1(\mathcal{X}, \mathbb{R}^m)$, feature map $g(x, \xi) = h(x) + \xi$ corresponds to the m first components of diffeomorphism $\varphi : \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}^m \times \mathcal{X}$ defined as:

$$\varphi(x, \xi) = \begin{pmatrix} h(x) + \xi \\ x \end{pmatrix} \quad \text{with} \quad \varphi^{-1}(z_1, z_2) = \begin{pmatrix} z_2 \\ z_1 - h(z_2) \end{pmatrix}.$$

Dimension augmentation

Apply the above methodology to the dimension-augmented function $\overline{u} : \mathcal{X} \times \Xi \to \mathbb{R}$ defined by

$$\overline{u}(\boldsymbol{X}, \boldsymbol{\Xi}) = u(\boldsymbol{X}),$$

where $\Xi \subseteq \mathbb{R}^k$ for some $k \geq 1$ and where $\boldsymbol{\Xi}$ is a random vector with values in Ξ, centered and independent from $\boldsymbol{X}$.

Dimension augmentation trick

For $m \leq d$, we consider the following set of feature maps

$$\mathcal{G}_m\left(\mathcal{X} \times \mathbb{R}^k\right) = \left\{ \begin{array}{rccc} g: & \mathcal{X} \times \Xi & \to & \mathbb{R}^m \\ & (x, \xi) & \mapsto & (\varphi_1(x, \xi), \ldots, \varphi_m(x, \xi)) \end{array} \middle| \, \varphi \in \mathcal{D}\left(\mathcal{X} \times R^k\right) \right\}$$

where $\mathcal{D}\left(\mathcal{X} \times \mathbb{R}^k\right)$ is a set of $\mathcal{C}^1$-diffeomorphisms from $\mathcal{X} \times \Xi$ to $\mathbb{R}^{d+k}$. For $g \in \mathcal{G}_m\left(\mathcal{X} \times \mathbb{R}^k\right)$, applying the methodology described previously, we obtain:

$$\min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(\overline{u}(\boldsymbol{X}, \boldsymbol{\Xi}) - f \circ g(\boldsymbol{X}, \boldsymbol{\Xi}))^2] \leq \mathbb{C}^{\mathrm{IS}}\left((\boldsymbol{X}, \boldsymbol{\Xi}) | \mathcal{G}_m\left(\mathcal{X} \times \mathbb{R}^k\right)\right) \mathcal{J}^{\mathrm{IS}}_{k,m}(g)$$

$$\min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(\overline{u}(\boldsymbol{X}, \boldsymbol{\Xi}) - f \circ g(\boldsymbol{X}, \boldsymbol{\Xi}))^2] \leq \mathbb{C}^{\mathrm{FS}}_m\left((\boldsymbol{X}, \boldsymbol{\Xi}) | \mathcal{D}\left(\mathcal{X} \times \mathbb{R}^k\right)\right) \mathcal{J}^{\mathrm{FS}}_{k,m}(\varphi)$$

with

$$\mathcal{J}^{\mathrm{IS}}_{k,m}(g) = \mathbb{E}\left[\left\|\nabla \overline{u}(\boldsymbol{X}, \boldsymbol{\Xi}) - \Pi_{\mathrm{range}(\nabla g(\boldsymbol{X}, \boldsymbol{\Xi})^\top)} \nabla \overline{u}(\boldsymbol{X}, \boldsymbol{\Xi})\right\|^2\right],$$

$$\mathcal{J}^{\mathrm{FS}}_{k,m}(\varphi) = \mathbb{E}\left[\left\|\left(\nabla \varphi(\boldsymbol{X}, \boldsymbol{\Xi})^{-\top} \nabla \overline{u}(\boldsymbol{X}, \boldsymbol{\Xi})\right)_\perp\right\|^2\right].$$

more

Outline

Let $u : \mathcal{X} \to \mathbb{R}^q$ with output dimension $q \geq 1$. As before, our goal is to build $g : \mathcal{X} \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}^q$ with $m << d$ such that

$$\mathbb{E}[\|u(\mathbf{X}) - f \circ g(\mathbf{X})\|^2]$$

is small, where $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^q$. Following the methodology proposed in [Zahm et al., 2020], we first decompose the reconstruction error as follows

$$\mathcal{E}(g) = \min_{f:\mathbb{R}^m \to \mathbb{R}^q} \mathbb{E}[\|u(\mathbf{X}) - f \circ g(\mathbf{X})\|^2] = \sum_{i=1}^{q} \min_{f_i:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u_i(\mathbf{X}) - f_i \circ g(\mathbf{X}))^2]$$

and then we bound each term $\mathcal{E}_i(g) = \min_{f_i} \mathbb{E}[(u_i(\mathbf{X}) - f_i \circ g(\mathbf{X}))^2]$ using Poincaré Inequalities.

Remark

Note that $\nabla u(x) \in \mathbb{R}^{q \times d}$ denotes now the Jacobian of u:

$$\nabla u(x) = \begin{pmatrix} \partial_1 u_1(x) & \dots & \partial_d u_1(x) \\ \vdots & \ddots & \vdots \\ \partial_1 u_q(x) & \dots & \partial_d u_q(x) \end{pmatrix}.$$

Previous results remain true with

$$\mathcal{J}_m^{\mathrm{IS}}(g) = \mathbb{E}[\|\nabla u(\mathbf{X})^\top - \Pi_{\mathrm{range}(\nabla g(\mathbf{X})^\top)} \nabla u(\mathbf{X})^\top\|_F^2],$$

$$\mathcal{J}_m^{\mathrm{FS}}(\varphi) = \mathbb{E}[\|U_\perp^\top \nabla \varphi(\mathbf{X})^{-\top} \nabla u(\mathbf{X})^\top\|_F^2],$$

$$\mathcal{J}_{m,\lambda}^{\mathrm{FS}}(\varphi) = \mathbb{E}[\|U_\perp^\top \nabla \varphi(\mathbf{X})^{-\top} \nabla u(\mathbf{X})^\top\|_F^2] + \lambda \mathbb{E}\left[\frac{\|\varphi(\mathbf{X})\|^2}{2} - \log|\det \nabla \varphi(\mathbf{X})|\right],$$

with $\|\cdot\|_F$ the Frobenius norm ($\|A\|_F^2 = \sum_{ij} A_{ij}^2$) and $U_\perp \in \mathbb{R}^{d\times(d-m)}$ the orthogonal projection matrix into the last $d-m$ coordinates.

Outline

An interesting application is to consider $u : \begin{cases} \mathcal{X} \to \mathbb{R}^d \\ x \mapsto x \end{cases}$ with the aim to learn $\mathbf{X}$ from samples $\mathbf{X}^{(i)}$, $1 \leq i \leq n_{\text{train}}$. It is an *unsupervised* learning task.

Autoencoder

Assume $\mathbf{X}$ takes values in an unknown m-dimensional manifold $\mathcal{M} \subseteq \mathbb{R}^d$. Then

- the feature map $g : \mathcal{X} \to \mathbb{R}^m$ can be interpreted as an *encoder* which extracts the *latent variable* $\mathbf{Z}_m = g(\mathbf{X})$,
- the profile function $f : \mathbb{R}^m \to \mathbb{R}^d$ corresponds to the *decoder* such that $f(\mathbf{Z}_m)$ yields an approximation to $\mathbf{X}$.

In this framework, for any encoder $g \in \mathcal{G}_m$, the optimal decoder f_g^* admits a closed form expression:

$$f_g^*(z_m) = \mathbb{E}[\varphi^{-1}(z_m, \mathbf{Z}_\perp) | \mathbf{Z}_m = z_m],$$

where $(\mathbf{Z}_m, \mathbf{Z}_\perp) = \varphi(\mathbf{X})$. The cost functions are then:

$$\mathcal{J}_m^{\text{IS}}(g) = \mathbb{E}[\| I_d - \Pi_{\text{range}(\nabla g(\mathbf{X})^\top)} \|_F^2] = \mathbb{E}[d - \text{trace}(\Pi_{\text{range}(\nabla g(\mathbf{X}))})] = d - m,$$

$$\mathcal{J}_{m,\lambda}^{\text{FS}}(\varphi) = \mathbb{E}[\| U_\perp^\top \nabla \varphi(\mathbf{X})^{-\top} \|_F^2] + \lambda \mathbb{E}\left[\frac{\|\varphi(\mathbf{X})\|^2}{2} - \log |\det \nabla \varphi(\mathbf{X})| \right].$$

Link with [Nguyen et al., 2019]

They also parametrize the encoder as $g = (\varphi_1, \ldots, \varphi_m)$, but the decoder is the sub-optimal decoder defined as $f(z_m) = \varphi^{-1}(z_m, 0_\perp)$, where $0_\perp = (0, \ldots, 0)$ is the zero vector in $\mathbb{R}^{d-m}$. The diffeomorphism φ is trained by minimizing directly the reconstruction error

$$\mathcal{L}_m^0(\varphi) = \mathbb{E}[\|\mathbf{X} - \varphi^{-1}(g(\mathbf{X}), 0_\perp)\|^2].$$

Outline

A class of invertible neural networks

See [Teshima et al., 2020, Ishikawa et al., 2022, Dinh et al., 2015] for details.

Bloc Affine Coupling Flows (BACF)

Let $d \geq 2$. For two functions $s, t : \mathbb{R}^{\lfloor d/2 \rfloor} \to \mathbb{R}^{d-\lfloor d/2 \rfloor}$, the block affine coupling flow (BACF) $\Psi_{s,t} : \mathbb{R}^d \to \mathbb{R}^d$ is defined by

$$\Psi_{s,t}(x) = \begin{pmatrix} x_{\leq \lfloor d/2 \rfloor} \\ x_{> \lfloor d/2 \rfloor} \odot \exp(s(x_{\leq \lfloor d/2 \rfloor})) + t(x_{\leq \lfloor d/2 \rfloor}) \end{pmatrix},$$

where $\odot$ is the element-wise product and $\exp(\cdot)$ is applied element-wise.

ℓ-layered BACF

For $d, \ell \in \mathbb{N}^*$, let

$$\mathcal{D}^{d,\ell}_{\mathrm{BACF}} = \left\{ (P \circ \Psi_{s_\ell,t_\ell}) \circ \ldots \circ (P \circ \Psi_{s_1,t_1}) \;\middle|\; s_i, t_i \in \mathcal{C}^1(\mathbb{R}^{\lfloor \frac{d}{2} \rfloor}; \mathbb{R}^{d-\lfloor \frac{d}{2} \rfloor}) \right\},$$

with the block-coordinate permutation $P : x \in \mathbb{R}^d \mapsto P(x) = (x_{> \lfloor \frac{d}{2} \rfloor}, x_{\leq \lfloor \frac{d}{2} \rfloor})$.

Remark

In practice, if $\mathcal{X} \subsetneq \mathbb{R}^d$, we can always consider diffeomorphism φ^0 that maps $\mathbb{R}^d$ to $\mathcal{X}$ and replace $u : \mathcal{X} \to \mathbb{R}$ with $u \circ \varphi^0 : \mathbb{R}^d \to \mathbb{R}$.

Outline

High dimensional function approximation

From a training set $\{(x^i, u(x^i), \nabla u(x^i))\}_{i=1}^{n_{\text{train}}}$:

1. we learn g by optimizing one of the upper-bounds,
2. we learn f as a fully connected neural network with 3 hidden layers of 50 neurons each and the sigmoid activation function by minimizing the empirical $\mathbb{L}^2_\pi$ error.

On a random testing set of size n_{test}, we compute:

$$\text{MSE} = \mathbb{E}((u(\mathbf{X}) - f \circ g(\mathbf{X}))^2), \qquad \text{NRMSE} = \frac{\sqrt{\mathbb{E}((u(\mathbf{X}) - f \circ g(\mathbf{X}))^2)}}{\max\limits_{1 \leq i \leq n_{\text{test}}} u(x^i) - \min\limits_{1 \leq i \leq n_{\text{test}}} u(x^i)},$$

$$\text{RL}_2 = \sqrt{\frac{\mathbb{E}((u(\mathbf{X}) - f \circ g(\mathbf{X}))^2)}{\mathbb{E}(u(\mathbf{X})^2)}}, \qquad \text{RL}_1 = \frac{\mathbb{E}(|u(\mathbf{X}) - f \circ g(\mathbf{X})|)}{\mathbb{E}(|u(\mathbf{X})|)}.$$

Cost functions are minimized with ADAM and a learning rate fixed to 0.01. For g, a multi-start procedure is applied with 8 starting points. After 300 steps, the best candidate is selected and optimized for 3000 additional steps. For f, single-start is used and ADAM is stopped after 5000 steps or when the loss function reduces to 10^{-8}.

High dimensional function approximation

Compared procedures

$\mathcal{J}_m^{\mathrm{IS}}$, $\mathcal{J}_m^{\mathrm{FS}}$, and $\mathcal{J}_m^{\mathrm{NLL}}$ [Zhang et al., 2019a] differ only in the choice of norm used to measure the projected gradient. They all reduce to Active Subspace for φ a linear isometry. DRiLLS [Teng et al., 2023] relaxes the invertibility constraint on φ by adding a penalty to $\mathcal{J}_m^{\mathrm{NLL}}(\varphi)$.

Test cases

$$u_1(x) = \sin(\|\mathbf{x}\|^2), \qquad u_2(x) = \exp\left(\frac{1}{d}\sum_{i=1}^{d}\sin(x_i)e^{\cos(x_i)}\right)$$

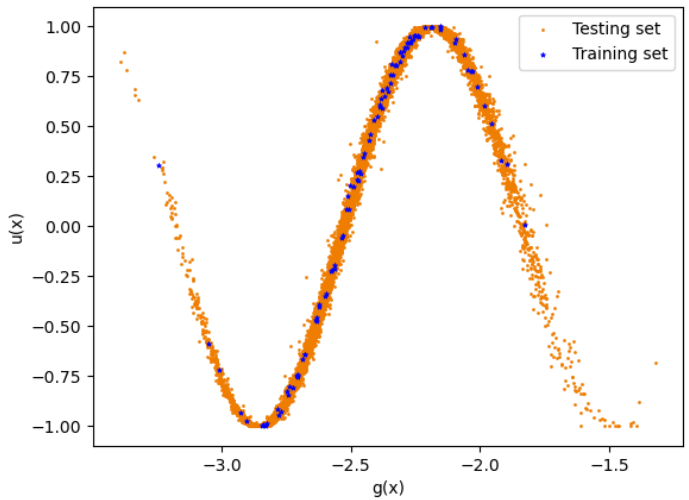
- u_1 can be decomposed with a polynomial feature map;
- u_2 cannot be decomposed with a polynomial feature map.

u_1 on $\mathcal{X} = [0,1]^{20}$						
Points	Structure	#Param	MSE%	NRMSE%	RL_2%	RL_1%
$n_{\text{train}} = 100$	BACF_IS(5)	2100	$(3.83 \pm 3.85) \times 10^{-2}$	9.11 ± 3.59	26.14 ± 10.30	18.98 ± 6.00
	$\text{BACF\_IS}_{d+2}(4)$	2024	$(\mathbf{8.01} \pm 4.88) \times \mathbf{10^{-3}}$	$\mathbf{4.29} \pm 1.27$	$\mathbf{12.32} \pm 3.65$	$\mathbf{8.57} \pm 2.35$
	BACF_FS(5)	2100	$(7.80 \pm 1.67) \times 10^{-2}$	13.89 ± 1.43	39.99 ± 4.12	31.51 ± 2.98
	$\text{BACF\_FS}_{d+2}(4)$	2024	$(5.84 \pm 2.59) \times 10^{-2}$	11.81 ± 2.56	33.73 ± 7.32	26.08 ± 6.13
	PFM	230	$(9.73 \pm 4.36) \times 10^{-2}$	15.23 ± 3.47	43.77 ± 9.98	13.38 ± 7.86
$n_{\text{train}} = 500$	BACF_IS(5)	2100	$(1.42 \pm 0.95) \times 10^{-3}$	1.81 ± 0.53	5.22 ± 1.53	3.34 ± 1.36
	$\text{BACF\_IS}_{d+2}(4)$	2024	$(1.91 \pm 2.00) \times 10^{-3}$	1.96 ± 0.96	5.64 ± 2.77	4.21 ± 3.09
	BACF_FS(5)	2100	$(1.70 \pm 0.82) \times 10^{-3}$	2.01 ± 0.46	5.75 ± 1.32	3.99 ± 1.12
	$\text{BACF\_FS}_{d+2}(4)$	2100	$(1.92 \pm 1.19) \times 10^{-3}$	2.12 ± 0.57	6.07 ± 1.62	4.32 ± 1.46
	PFM	230	$(\mathbf{1.49} \pm 4.13) \times \mathbf{10^{-4}}$	$\mathbf{0.39} \pm 0.48$	$\mathbf{1.13} \pm 1.39$	$\mathbf{0.44} \pm 0.15$
	NLL	-	-	5.09	-	7.46
	DRiLLS	-	-	28.73	-	79.29

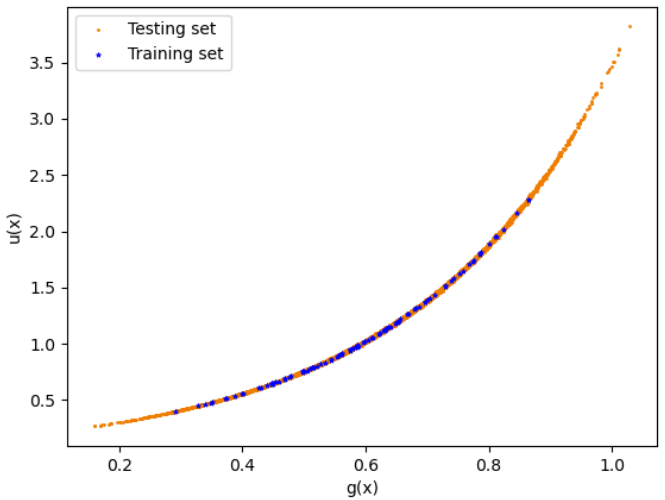
Table: Approximation errors for u_1 with $m = 1$, the NLL and DRiLLS results are copied from [Teng et al., 2021] Table 3, $\lambda = 10^{-4}$

u_2 on $\mathcal{X} = [-\frac{\pi}{2}, \frac{\pi}{2}]^8$						
Points	Structure	#Param	MSE%	NRMSE%	RL_2%	RL_1%
$n_{\text{train}} = 100$	BACF_IS(6)	432	$(3.06 \pm 1.74) \times 10^{-2}$	4.75 ± 1.25	14.32 ± 3.77	10.63 ± 2.72
	$\text{BACF\_IS}_{d+2}(4)$	440	$(\mathbf{2.61} \pm 2.45) \times \mathbf{10^{-3}}$	$\mathbf{1.40} \pm 0.64$	$\mathbf{3.98} \pm 1.82$	$\mathbf{2.54} \pm 1.01$
	BACF_FS(6)	432	$(2.06 \pm 0.58) \times 10^{-1}$	12.40 ± 1.83	37.94 ± 5.60	29.41 ± 4.41
	$\text{BACF\_FS}_{d+2}(4)$	440	$(1.57 \pm 0.77) \times 10^{-1}$	10.80 ± 2.39	33.12 ± 7.33	23.98 ± 3.25
	PFM	44	$(1.93 \pm 0.24) \times 10^{-1}$	12.03 ± 0.79	37.13 ± 2.43	30.78 ± 2.22
$n_{\text{train}} = 500$	BACF_IS(6)	432	$(5.15 \pm 2.14) \times 10^{-3}$	2.01 ± 0.34	6.00 ± 1.02	4.46 ± 0.99
	$\text{BACF\_IS}_{d+2}(4)$	440	$(\mathbf{4.50} \pm 3.08 \times \mathbf{10^{-4}}$	$\mathbf{0.56} \pm 0.22$	$\mathbf{1.67} \pm 0.65$	$\mathbf{1.04} \pm 0.44$
	BACF_FS(6)	432	$(9.22 \pm 2.79) \times 10^{-3}$	2.77 ± 0.42	8.12 ± 1.22	5.61 ± 0.85
	$\text{BACF\_FS}_{d+2}(4)$	440	$(7.47 \pm 4.78) \times 10^{-3}$	2.16 ± 0.71	6.96 ± 2.30	4.81 ± 1.45
	PFM	44	$(1.94 \pm 0.17) \times 10^{-1}$	12.08 ± 0.56	37.69 ± 1.71	31.26 ± 1.44

Table: Approximation errors for u_2 on $\mathcal{X} = [-\frac{\pi}{2}, \frac{\pi}{2}]^d$ and with $m = 1$, $\lambda = 10^{-4}$



(a) u_1 on $\Omega = [0,1]^{20}$ with BACF_IS$_{d+2}(4)$, $n_{\text{train}} = 100$



(b) u_2 on $\Omega = [-1,1]^8$with BACF_IS$_{d+2}(4)$, $n_{\text{train}} = 100$

Figure: Scatter plot $\{(g(x^i), u(x^i))\}_{i\geq 1}$ for a random testing set of $n_{\text{test}} = 10000$ points $m = 1$

Application to autoencoders

Synthetic dataset in $\mathbb{R}^d$

We generate a submanifold of intrinsic dimension $m_{\text{int}} < d$ with:

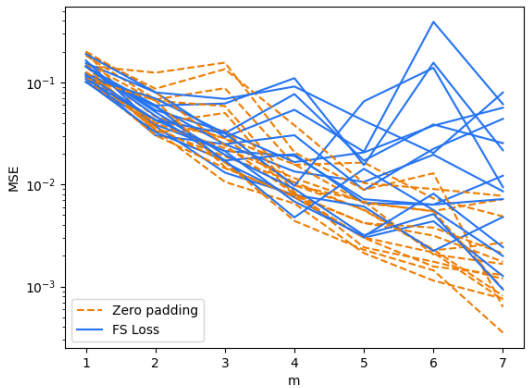
$$\Psi_Q : \begin{cases} \mathbb{R}^{m_{\text{int}}} & \to & \mathbb{R}^d \\ Z & \mapsto & (Z^\top Q_1 Z, \dots, Z^\top Q_d Z) \end{cases}$$

with $Q = (Q_1, \dots, Q_d) \in (\mathbb{Z}^{m_{\text{int}} \times m_{\text{int}}})^d$. For a given Q, $\mathbf{Z}$ is sampled according to the probability distribution $\mathcal{U}([-2,2]^{m_{\text{int}}})$ and we compute $\mathbf{X} = \Psi_Q(\mathbf{Z})$ to generate the dataset. For $d = 8$ and $m_{\text{int}} = 3$ (respectively $d = 16$ and $m_{\text{int}} = 5$), we set $Q = Q_8$ (respectively $Q = Q_{16}$) and we generate the datasets as described above. Q_8 and Q_{16}

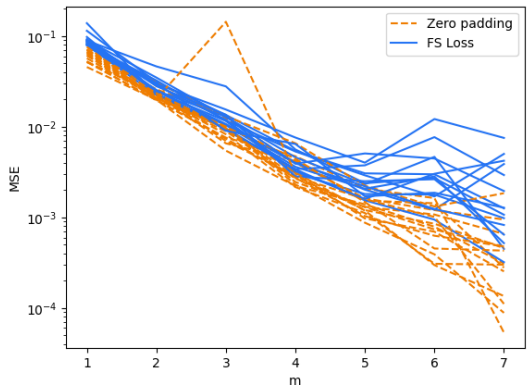
Implementation

The error $\mathbb{E}(\|\boldsymbol{X} - f \circ g(\boldsymbol{X})\|^2)$ is computed on a test set of size $n_{\text{test}} = 10^4$. We plot the test error as a function of m for the 15 training sets.

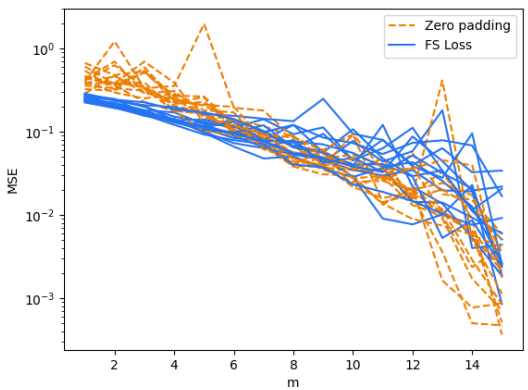
Reconstruction error using either zero padding or FS Loss. We use BACF(4) with $\lambda = 10^{-2}$.



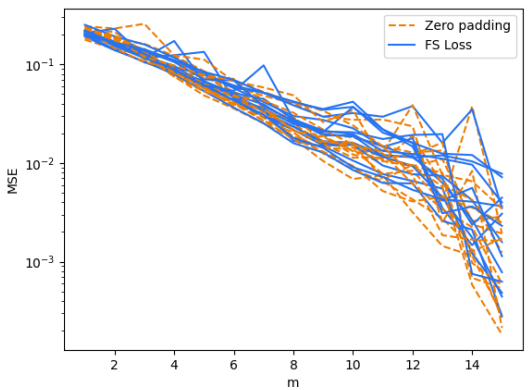
(a) Q_8 datasets with $n_{train} = 100$



(b) Q_8 datasets with $n_{train} = 500$



(c) Q_{16} datasets with $n_{train} = 100$



(d) Q_{16} datasets with $n_{train} = 500$

How to tackle higher dimension?

There are two main issues in the aforementioned methodology.

- Minimizing the upper bound $\mathcal{J}_m^{\mathrm{FS}}(\varphi)$ over $\varphi \in \mathcal{D}$ still requires evaluating the Jacobian $\nabla\varphi(\mathbf{X})$ for each sample and at every iteration of the optimization procedure.
- Moreover, since no theoretical bound is available for the Poincaré constant $\mathbb{C}_m(\mathbf{X} \mid \mathcal{D})$, minimizing $\mathcal{J}_m^{\mathrm{FS}}(\varphi)$ does not necessarily ensure a corresponding reduction in the $\mathbb{L}^2_\pi$ approximation error.

To address the first limitation, we make use of the Hutchinson estimator [Hutchinson, 1989] to estimate the Frobenius norm of a matrix $A \in \mathbb{R}^{d\times d}$, that is we use a Monte Carlo estimator based on the identity $\|A\|_F^2 = \mathbb{E}_{\mathbf{Z}}[\|A\mathbf{Z}\|_2^2]$, for $\mathbf{Z} \sim \mathcal{N}(0, I_d)$.

This estimator provides an efficient approximation of $\mathcal{J}_m^{\mathrm{FS}}(\varphi)$ without explicitly computing the full Jacobian $\nabla\varphi(\mathbf{X})$, thereby enabling scalability to high-dimensional settings.

We address the second limitation by considering as loss function the $\mathbb{L}^2_\pi$ approximation error, regularized by the bound in the feature space:

$$\mathcal{L}_m(\varphi) = \mathbb{E}_{\mathbf{X}}\left[\|u(\mathbf{X}) - f(U_m^\top \varphi(\mathbf{X}))\|^2\right] + \mu\,\mathbb{E}_{\mathbf{X},\mathbf{Z}}\left[\left\|U_\perp^\top \nabla\varphi(\mathbf{X})^{-\top}\nabla u(\mathbf{X})^\top \mathbf{Z}\right\|_2^2\right] + \lambda\,\mathbb{E}_{\mathbf{X}}\left[\frac{\|\varphi(\mathbf{X})\|_\Sigma^2}{2} - \log|\det \nabla\varphi(\mathbf{X})|\right],$$

where $\mathbf{Z} \sim \mathcal{N}(0, I_q)$ and $\mu, \lambda \in \mathbb{R}$ are tunable hyperparameters, $U_m \in \mathbb{R}^{d\times m}$ and , $U_\perp \in \mathbb{R}^{d\times(d-m)}$ denoting the matrices respectively formed by the first m and the last $d-m$ columns of the identity matrix in $\mathbb{R}^d$.

In the following, we focus on two important special cases:

1. The case $q = 1$, $\nabla u(\mathbf{X})^\top$ reduces to the gradient of the model output, and the Poincaré cost function $\mathcal{L}_m^{\mathrm{FS}}(\varphi)$ can be computed using vector–Jacobian products.
2. The case $u(\mathbf{X}) = \mathbf{X}$, we then have $\nabla u(\mathbf{X})^\top = I_d$, and the framework naturally reduces to unsupervised feature learning with invertible autoencoders.

Numerical illustration

The first case does not require the evaluation of the Jacobian, thus the use of the Hutchinson estimator.

We compare different strategies: directly minimizing the $\mathbb{L}^2_\pi$ error $\mathcal{L}^0_m(\varphi)$, minimizing only the Poincaré upper-bound $\mathcal{J}^{\mathrm{IS}}_m(\varphi)$ or $\mathcal{J}^{\mathrm{FS}}_m(\varphi)$, minimizing the $\mathbb{L}^2$ error regularized with the Poincaré upper-bound $\mathcal{L}_m(\varphi)$.

The diffeomorphism φ is parameterized using a 8-layered Block Affine Coupling Flows (BACF), the profile function is a fully connected neural network of 2 layers of 20 neurons and a hyperbolic tangent activation function. The covariance matrix is set to $\Sigma = \mathrm{diag}(I_m, \sigma^2 I_{d-m}) \in \mathbb{R}^{d\times d}$, where I_m denotes the identity matrix in $\mathbb{R}^{m\times m}$ and $\sigma = 0.1$. This choice of σ promotes small variations in the uninformative features relative to the informative ones. Finally, when for $\mathcal{L}_m(\varphi)$, $\mu = 1$ and $\lambda = 0.01$ and for $\mathcal{J}^{\mathrm{FS}}_m(\varphi)$, $\lambda = 0.01$.

We consider the following benchmark functions:

$$u_1(x) = \sin\left(\frac{1}{d}\|\mathbf{x}\|^2\right), \qquad u_2(x) = \exp\left(\frac{1}{d}\sum_{i=1}^{d}\sin(x_i)\, e^{\cos(x_i)}\right),$$

where $x := (x_1, \ldots, x_d) \in \mathcal{X} \subset \mathbb{R}^d$. The function u_1 is defined on $\mathcal{X} = [0,1]^{150}$, while u_2 is defined on $\mathcal{X} = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]^{150}$, and in both cases $\mathbf{X}$ is uniformly distributed on $\mathcal{X}$.

Training samples $\{x^{(i)}\}_{i=1}^{n_{\text{train}}}$ are generated using Latin hypercube sampling with a maximin criterion, implemented via the `scikit-optimize` library [Head et al., 2022]. The test set consists of $n_{\text{test}} = 10^4$ independent samples of $\mathbf{X}$.

u_1 on $\mathcal{X} = [0,1]^{150}$							
Points	Latent dim	Loss	Training time(s)	MSE%	NRMSE%	RL_2%	RL_1%
$n_{\text{train}} = 750$	$m = 1$	$\mathcal{L}_m^0$	26.27 ± 5.19	$(1.00 \pm 0.30) \times 10^{-3}$	17.75 ± 3.32	7.26 ± 1.27	9.39 ± 1.76
		$\mathcal{L}_m$	78.82 ± 8.20	$(2.00 \pm 1.00) \times 10^{-4}$	7.61 ± 2.39	3.09 ± 1.05	4.03 ± 1.27
		$\mathcal{J}_m^{\mathrm{FS}}$	105.63 ± 18.51	$(5.00 \pm 0.21) \times 10^{-4}$	13.28 ± 0.25	5.61 ± 0.07	7.03 ± 0.13
		$\mathcal{J}_m^{\mathrm{IS}}$	$100.71 \pm 15,14$	$(9.07 \pm 2.22) \times 10^{-5}$	5.45 ± 0.63	2.30 ± 0.27	2.89 ± 0.33
	$m = 2$	$\mathcal{L}_m^0$	24.4 ± 3.88	$(1.10 \pm 0.40) \times 10^{-3}$	18.77 ± 3.54	7.21 ± 1.62	9.93 ± 1.87
		$\mathcal{L}_m$	71.70 ± 11.05	$(3.00 \pm 2.00) \times 10^{-4}$	8.88 ± 2.90	3.66 ± 1.24	4.70 ± 1.54
		$\mathcal{J}_m^{\mathrm{FS}}$	97.03 ± 9.13	$(5.00 \pm 0.34) \times 10^{-4}$	13.16 ± 0.45	5.56 ± 0.19	6.97 ± 0.24
		$\mathcal{J}_m^{\mathrm{IS}}$	117.04 ± 10.40	$(1.00 \pm 0.40) \times 10^{-4}$	5.96 ± 1.01	2.35 ± 0.41	3.15 ± 0.53
u_2 on $\mathcal{X} = [-\frac{\pi}{2}, \frac{\pi}{2}]^{150}$							
Points	Latent dim	Loss	Training time(s)	MSE%	NRMSE%	RL_2%	RL_1%
$n_{\text{train}} = 750$	$m = 1$	$\mathcal{L}_m^0$	22.58 ± 3.44	$(1.69 \pm 0.74) \times 10^{-2}$	16.85 ± 3.79	9.79 ± 2.24	12.58 ± 2.83
		$\mathcal{L}_m$	72.35 ± 11.30	$(6.70 \pm 4.80) \times 10^{-3}$	10.29 ± 3.57	5.96 ± 1.92	7.68 ± 2.67
		$\mathcal{J}_m^{\mathrm{FS}}$	135.51 ± 17.17	$(5.80 \pm 2.20) \times 10^{-3}$	9.93 ± 1.87	5.81 ± 1.14	7.41 ± 1.40
		$\mathcal{J}_m^{\mathrm{IS}}$	$145,65 \pm 31,11$	$(4.40 \pm 0.20) \times 10^{-3}$	8.75 ± 0.18	5.19 ± 0.11	6.53 ± 0.13
	$m = 2$	$\mathcal{L}_m^0$	26.35 ± 5.32	$(1.32 \pm 0.36) \times 10^{-2}$	15.08 ± 2.07	8.85 ± 0.98	11.26 ± 1.55
		$\mathcal{L}_m$	71.70 ± 7.26	$(7.40 \pm 5.60) \times 10^{-3}$	10.79 ± 3.79	6.14 ± 2.00	8.06 ± 2.83
		$\mathcal{J}_m^{\mathrm{FS}}$	118.33 ± 26.99	$(5.70 \pm 2.20) \times 10^{-3}$	9.81 ± 1.89	5.72 ± 1.11	7.32 ± 1.14
		$\mathcal{J}_m^{\mathrm{IS}}$	126.79 ± 27.81	$(4.30 \pm 0.10) \times 10^{-3}$	8.64 ± 0.12	5.10 ± 0.07	6.45 ± 0.09

Table: Approximation errors and training times for u_1 and u_2 using BACF with 8 layers (see [Verdière et al., 2023, Section 6]) for different training objectives. The results are averaged over 10 independent runs of the algorithm.

Numerical illustration for autoencoders

Context: $u : \mathcal{X} \to \mathbb{R}^d$, $u(\mathbf{X}) = \mathbf{X}$.

- The feature map $g(\cdot) = U_m^\top \varphi(\cdot)$ acts as an encoder that extracts a low-dimensional representation $\mathbf{Z}_m = g(\mathbf{X})$.
- The profile function f serves as a decoder such that $\mathbf{X} \approx f(\mathbf{Z}_m)$.

Choosing the profile function $z_m \mapsto \varphi^{-1}(z_m, 0_\perp)$ for $z_m \in \mathbb{R}^m$ leads to the following objective function:

$$\mathcal{L}_m(\varphi) = \mathbb{E}_{\mathbf{X}}\left[\|\mathbf{X} - \varphi^{-1}(U_m^\top \varphi(\mathbf{X}), 0_\perp)\|^2\right] + \mu\, \mathbb{E}_{\mathbf{X},\mathbf{Z}}\left[\left\|U_\perp^\top \nabla\varphi(\mathbf{X})^{-\top}\mathbf{Z}\right\|_2^2\right]$$
$$+ \lambda\, \mathbb{E}_{\mathbf{X}}\left[\frac{\|\varphi(\mathbf{X})\|_\Sigma^2}{2} - \log|\det \nabla\varphi(\mathbf{X})|\right],$$

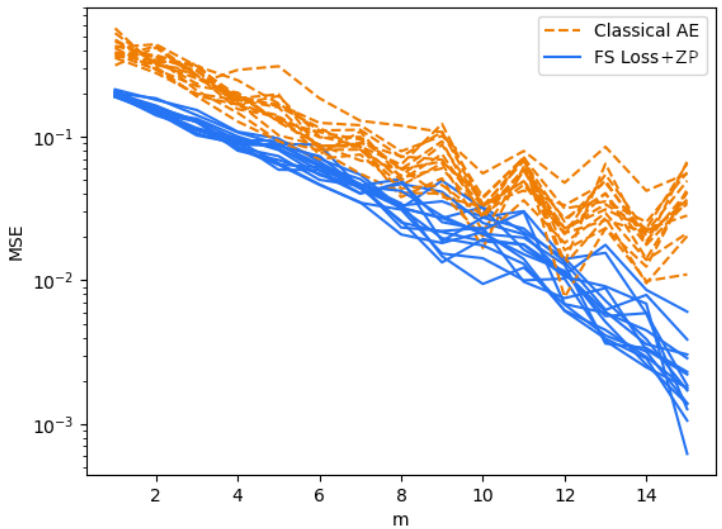
where $\boldsymbol{Z} \sim \mathcal{N}(0, I_d)$ and $\mu, \lambda \in \mathbb{R}$ are hyperparameters to be tuned.

We evaluate the different strategies on a synthetic dataset in $\mathbb{R}^d$, where the data lie on a submanifold of intrinsic dimension $m_{\mathrm{int}} < d$. The submanifold is generated via the mapping

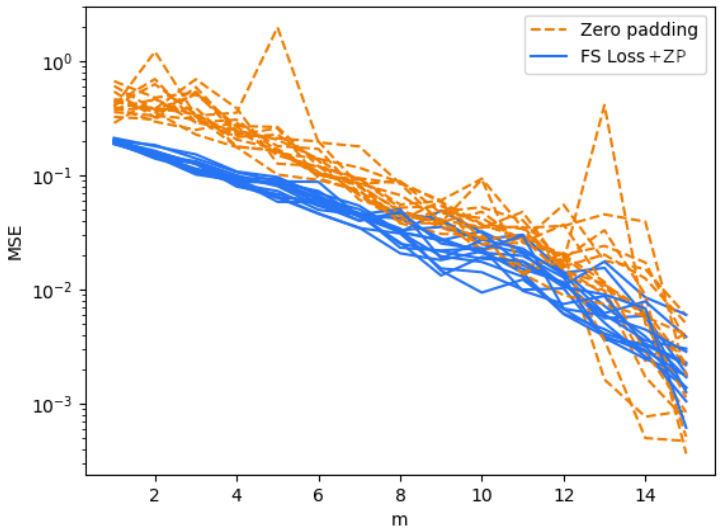
$$\begin{aligned} \Psi_Q : \quad \mathbb{R}^{m_{\mathrm{int}}} &\to \mathbb{R}^d, \\ Z &\mapsto (Z^\top Q_1 Z, \dots, Z^\top Q_d Z), \end{aligned}$$

where $Q = (Q_1, \dots, Q_d) \in (\mathbb{Z}^{m_{\mathrm{int}} \times m_{\mathrm{int}}})^d$ is a fixed collection of matrices. For a given Q, samples $\mathbf{Z}$ are drawn from a distribution $\pi_{\mathbf{Z}}$, and the dataset is generated by computing $\mathbf{X} = \Psi_Q(\mathbf{Z})$. In practice, we set $\pi_{\mathbf{Z}} = \mathcal{U}([-2, 2]^{m_{\mathrm{int}}})$ and the matrices Q_i are fixed to the values given in Q_{16} (see also [Verdière et al., 2023, Appendix F]).

We consider $d = 16$, $m_{\mathrm{int}} = 5$, and a fixed $Q = Q_{16}$. We train the models on 15 independent training sets and evaluate the approximation error $\mathbb{E}\left[\|\mathbf{X} - f \circ g(\mathbf{X})\|^2\right]$ on a test set of size $n_{\mathrm{test}} = 10^4$. When training with $\mathcal{L}_m$, we set $\mu = 1$, $\lambda = 10^{-2}$, and $\Sigma = \mathrm{diag}(I_m, \sigma^2 I_{d-m}) \in \mathbb{R}^{d \times d}$ with $\sigma = 0.1$.



(a) Classical autoencoder (Classical AE) vs Invertible autoencoder trained with $\mathcal{L}_m$ (FS Loss + ZP)



(b) Invertible autoencoder trained with $\mathcal{L}_m^0$ (Zero Padding) vs Invertible autoencoder trained with $\mathcal{L}_m$ (FS Loss + ZP)

Figure: Approximation error according to m for $d = 16$, $m_{\text{int}} = 5$ and $Q = Q_{16}$ with $n_{train} = 100$. Invertible autoencoder uses BACF neural networks with 4 layers and classical autoencoder uses fully connected neural network of equivalent number of parameters. When $\mathcal{L}_m$ is used for training, we set $\mu = 1$, $\lambda = 10^{-2}$ and $\Sigma = \text{diag}(I_m, \sigma^2 I_{d-m}) \in \mathbb{R}^{d \times d}$, where $\sigma = 0.1$.

Outline

- Conclusions
  - We proposed two dimension reduction strategies, both based on an upper-bound obtained from a Poincaré Inequality in either the input space or the feature space?
  - We implemented a trick consisting in augmenting the input dimension to add flexibility.
  - We applied these procedures on different test cases and on the unsupervised learning of an autoencoder.
  - We made use of the Hutchinson estimator to scale higher dimension.
  - We proposed a regularized $\mathbb{L}^2_\pi$ loss function to preserve accuracy while controling the numerical cost.

- Conclusions
  - We proposed two dimension reduction strategies, both based on an upper-bound obtained from a Poincaré Inequality in either the input space or the feature space?
  - We implemented a trick consisting in augmenting the input dimension to add flexibility.
  - We applied these procedures on different test cases and on the unsupervised learning of an autoencoder.
  - We made use of the Hutchinson estimator to scale higher dimension.
  - We proposed a regularized $\mathbb{L}^2_\pi$ loss function to preserve accuracy while controling the numerical cost.

- Perspectives
  - These approaches should be applied, in a near future, to a real case in biogeochemistry.
  - Our procedure should be tested on higher dimensional test cases.
  - Is it possible to leverage other functional inequalities than Poincaré to perform efficient dimension reduction?
  - ...

Thanks for your attention!

Some references I

Bigoni, D., Marzouk, Y., Prieur, C., and Zahm, O. (2022).

Nonlinear dimension reduction for surrogate modeling using gradient information.
Information and Inference: A Journal of the IMA, 11(4):1597–1639.

Bonnefont, M. and Joulin, A. (2022).

A note on eigenvalues estimates for one-dimensional diffusion operators.
Bernoulli, 28(1):64–86.

Cattiaux, P. and Guillin, A. (2022).

Functional inequalities for perturbed measures with applications to log-concave measures and to some bayesian problems.
Bernoulli, 28(4):2294–2321.

Dinh, L., Krueger, D., and Bengio, Y. (2015).

Nice: Non-linear independent components estimation.
ICLR workshop.

Dupont, E., Doucet, A., and Teh, Y. W. (2019).

Augmented neural odes.
Advances in neural information processing systems, 32.

Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., and Shcherbatyi, I. (2022).

scikit-optimize/scikit-optimize.

Some references II

Huang, C.-W., Dinh, L., and Courville, A. (2020).

Augmented normalizing flows: Bridging the gap between generative flows and latent variable models.
arXiv preprint arXiv:2002.07101.

Hutchinson, M. F. (1989).

A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines.
Communications in Statistics-Simulation and Computation, 18(3):1059–1076.

Ishikawa, I., Teshima, T., Tojo, K., Oono, K., Ikeda, M., and Sugiyama, M. (2022).

Universal approximation property of invertible neural networks.
arXiv preprint arXiv:2204.07415.

Lyu, J., Chen, Z., Feng, C., Cun, W., Zhu, S., Geng, Y., XU, Z., and Yongwei, C. (2022).

Para-cflows: C^k-universal diffeomorphism approximators as superior neural surrogates.
In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 28829–28841. Curran Associates, Inc.

Nguyen, T.-G. L., Ardizzone, L., and Köthe, U. (2019).

Training invertible neural networks as autoencoders.
pages 442–455.

Teng, Y., Wang, Z., Ju, L., Gruber, A., and Zhang, G. (2021).

Level set learning with pseudo-reversible neural networks for nonlinear dimension reduction in function approximation.
CoRR, abs/2112.01438.

Some references III

Teng, Y., Wang, Z., Ju, L., Gruber, A., and Zhang, G. (2023).

Level set learning with pseudoreversible neural networks for nonlinear dimension reduction in function approximation.

SIAM Journal on Scientific Computing, 45(3):A1148–A1171.

Teshima, T., Ishikawa, I., Tojo, K., Oono, K., Ikeda, M., and Sugiyama, M. (2020).

Coupling-based invertible neural networks are universal diffeomorphism approximators.

Advances in Neural Information Processing Systems, 33:3362–3373.

Verdière, R., Prieur, C., and Zahm, O. (2023).

Diffeomorphism-based feature learning using Poincaré inequalities on augmented input space.

working paper or preprint.

Zahm, O., Constantine, P. G., Prieur, C., and Marzouk, Y. M. (2020).

Gradient-based dimension reduction of multivariate vector-valued functions.

SIAM Journal on Scientific Computing, 42(1):A534–A558.

Zhang, G., Zhang, J., and Hinkle, J. (2019a).

Learning nonlinear level sets for dimensionality reduction in function approximation.

Advances in Neural Information Processing Systems, 32.

Zhang, H., Gao, X., Unterman, J., and Arodz, T. (2019b).

Approximation capabilities of neural ordinary differential equations.

arXiv preprint arXiv:1907.12998, 2(4):3–1.

More on dimension augmentation trick

go back

Let $z \mapsto f_g^*(z) := \mathbb{E}[u(\boldsymbol{X})|g(\boldsymbol{X}, \boldsymbol{\Xi}) = z]$. It realizes the minimum in $\mathcal{E}^k(g) = \min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}, \boldsymbol{\Xi}))^2]$.

Let $\widehat{u} : x \mapsto f_g^* \circ g(x, \boldsymbol{\Xi})$ and define $\widetilde{u} : x \mapsto \mathbb{E}[\widehat{u}(x)]$. Then

$$\begin{aligned}
\mathcal{E}^k(g) &= \min_{f:\mathbb{R}^m \to \mathbb{R}} \mathbb{E}[(u(\boldsymbol{X}) - f \circ g(\boldsymbol{X}, \boldsymbol{\Xi}))^2] \\
&= \mathbb{E}[(u(\boldsymbol{X}) - \widehat{u}(\boldsymbol{X}))^2] \\
&= \mathbb{E}[(u(\boldsymbol{X}) - \widetilde{u}(\boldsymbol{X}))^2] + \mathbb{E}[\mathrm{Var}(\widehat{u}(\boldsymbol{X}))|\boldsymbol{X})] \\
&= \mathbb{E}[(u(\boldsymbol{X}) - \widetilde{u}(\boldsymbol{X}))^2] + S_{\boldsymbol{\Xi}}^{tot,unnorm},
\end{aligned}$$

with $S_{\boldsymbol{\Xi}}^{tot,unnorm}$ the unnormalized total Sobol' index of output $\widehat{u}(\boldsymbol{X})$ with respect to input $\boldsymbol{\Xi}$. Thus if $S_{\boldsymbol{\Xi}}^{tot,unnorm}$ is negligible, $\widehat{u}(x)$ can be approximated by fixing $\boldsymbol{\Xi}$ to a nominal value.

Matrix Q_8

go back

$$Q_8 = (Q_{8,1}, Q_{8,2}, \ldots, Q_{8,8}) \in (\mathbb{Z}^{3\times 3})^8$$

$$Q_{8,1} = \begin{pmatrix} 4 & 0 & -5 \\ 0 & 2 & 1 \\ -3 & -5 & -3 \end{pmatrix} \quad Q_{8,2} = \begin{pmatrix} -1 & 3 & 2 \\ -1 & 0 & 1 \\ -3 & -2 & -3 \end{pmatrix}$$

$$Q_{8,3} = \begin{pmatrix} 4 & 4 & -3 \\ 1 & -4 & 0 \\ 1 & 3 & 0 \end{pmatrix} \quad Q_{8,4} = \begin{pmatrix} -4 & 3 & 4 \\ 1 & -5 & 0 \\ 3 & 1 & 0 \end{pmatrix}$$

$$Q_{8,5} = \begin{pmatrix} -2 & -2 & -3 \\ -1 & 1 & -5 \\ 1 & 2 & -1 \end{pmatrix} \quad Q_{8,6} = \begin{pmatrix} 4 & -2 & -5 \\ -5 & -5 & 2 \\ -5 & 1 & -2 \end{pmatrix}$$

$$Q_{8,7} = \begin{pmatrix} -4 & -3 & 0 \\ -5 & -1 & 1 \\ 4 & 4 & 2 \end{pmatrix} \quad Q_{8,8} = \begin{pmatrix} -2 & -3 & -1 \\ -5 & -5 & -1 \\ -1 & -3 & -2 \end{pmatrix}$$

Matrix Q_{16}

go back

$$Q_{16} = (Q_{16,1}, Q_{16,2}, \ldots, Q_{16,16}) \in (\mathbb{Z}^{5\times 5})^{16}$$

$$Q_{16,1} = \begin{pmatrix} -2 & -5 & -4 & 4 & -4 \\ 2 & 3 & -1 & -1 & 2 \\ 3 & -5 & -3 & 0 & -5 \\ 0 & -1 & -5 & -1 & -3 \\ 4 & 1 & 3 & -4 & 3 \end{pmatrix} \quad Q_{16,2} = \begin{pmatrix} 1 & 1 & -5 & -5 & -5 \\ -5 & 1 & 4 & -3 & -5 \\ 2 & -5 & 1 & -5 & 1 \\ -2 & -2 & -2 & 2 & -3 \\ 2 & -5 & -5 & -2 & -2 \end{pmatrix}$$

$$Q_{16,3} = \begin{pmatrix} -3 & -4 & -1 & -3 & -3 \\ 3 & 3 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ -1 & -1 & 3 & 2 & -4 \\ 4 & 0 & 0 & -4 & 4 \end{pmatrix} \quad Q_{16,4} = \begin{pmatrix} -2 & 1 & 3 & 1 & 1 \\ -4 & -5 & 4 & -5 & -4 \\ 1 & 0 & -5 & 0 & 3 \\ 0 & 3 & -3 & 3 & 3 \\ 0 & -1 & -1 & -3 & 2 \end{pmatrix}$$

$$Q_{16,5} = \begin{pmatrix} -4 & 2 & -1 & 0 & 3 \\ -4 & 0 & -1 & -5 & 0 \\ -2 & 2 & -2 & 0 & 2 \\ 3 & -4 & 0 & 3 & 1 \\ 0 & -2 & 3 & 2 & -2 \end{pmatrix} \quad Q_{16,6} = \begin{pmatrix} 3 & 3 & 4 & 3 & 0 \\ -4 & -3 & -4 & 4 & -5 \\ -3 & 0 & -2 & 4 & 2 \\ -3 & -2 & -4 & -1 & -3 \\ -4 & -3 & 3 & 1 & -5 \end{pmatrix}$$

$$Q_{16,7} = \begin{pmatrix} 4 & 2 & 0 & -2 & 1 \\ 3 & 1 & 2 & 0 & -2 \\ 0 & 2 & 0 & -2 & 2 \\ -5 & 3 & 3 & -4 & 4 \\ -2 & -3 & -1 & 2 & 0 \end{pmatrix} \quad Q_{16,8} = \begin{pmatrix} 2 & 0 & -4 & 1 & 2 \\ 3 & -1 & 0 & 1 & -2 \\ -1 & 2 & 0 & 3 & 0 \\ -4 & -4 & 1 & 2 & 0 \\ 4 & -4 & 2 & 3 & 4 \end{pmatrix}$$

$$Q_{16,9} = \begin{pmatrix} 1 & -4 & -1 & 0 & -3 \\ 4 & -3 & -2 & 2 & 3 \\ 0 & 1 & 3 & 3 & -3 \\ 0 & -5 & -3 & -3 & 4 \\ 1 & 4 & 3 & -1 & -4 \end{pmatrix} \quad Q_{16,10} = \begin{pmatrix} -2 & -5 & 2 & -3 & 1 \\ -4 & 2 & -1 & 0 & -2 \\ -5 & -3 & -2 & -3 & -4 \\ -3 & -3 & -4 & 1 & -1 \\ 2 & -2 & 3 & -4 & -4 \end{pmatrix}$$

$$Q_{16,11} = \begin{pmatrix} 1 & 0 & -5 & 1 & 2 \\ 4 & 0 & 0 & 0 & -3 \\ 4 & -3 & -4 & 3 & 2 \\ 1 & 3 & 3 & -2 & 2 \\ -5 & -1 & 1 & 0 & 2 \end{pmatrix} \quad Q_{16,12} = \begin{pmatrix} 1 & -5 & 2 & -4 & 2 \\ -2 & -3 & -4 & 4 & 2 \\ 2 & -4 & -1 & 0 & -2 \\ 3 & 1 & -4 & 2 & 4 \\ 1 & -3 & 1 & 4 & 0 \end{pmatrix}$$

$$Q_{16,13} = \begin{pmatrix} 0 & 1 & -2 & 4 & -5 \\ -5 & 4 & 1 & 4 & -5 \\ 2 & 3 & 1 & -4 & 0 \\ 3 & 0 & 1 & -4 & 3 \\ 2 & -3 & -5 & 1 & -3 \end{pmatrix} \quad Q_{16,14} = \begin{pmatrix} 2 & -4 & 4 & 2 & 4 \\ -5 & 2 & -3 & 0 & 2 \\ 1 & -4 & -3 & 4 & 0 \\ 0 & -2 & -5 & -5 & 1 \\ -4 & -4 & -4 & -1 & -3 \end{pmatrix}$$

$$Q_{16,15} = \begin{pmatrix} -2 & -5 & 4 & -1 & 2 \\ 0 & 3 & -4 & -5 & 4 \\ 1 & 2 & -2 & 1 & 1 \\ -4 & 4 & -5 & 2 & 4 \\ -1 & 4 & -3 & 1 & 1 \end{pmatrix} \quad Q_{16,16} = \begin{pmatrix} -2 & -5 & -1 & 4 & 0 \\ 2 & -4 & -5 & 4 & 4 \\ -2 & 1 & 0 & 4 & 1 \\ 2 & -2 & -3 & 0 & 3 \\ 3 & 3 & 2 & -3 & 2 \end{pmatrix}$$