DEEL
DEpendable & Explainable Learning

Towards instance-dependent approximation guarantees for scientific machine learning using Lipschitz neural networks

P. Novello, IRT Saint Exupery
C. Gauchy, CEA
M. Dalery, Laboratoire de Mathématiques de Besançon
M. Peyron, CERFACS & EVIDEN
S. Saha, Indian Statistical Institute

IVADO
SAINT EXUPÉRY
CRIAQ
ANITI Université Fédérale Toulouse Midi-Pyrénées
Artificial & Natural Intelligence Toulouse Institute
Institut intelligence et données

Challenges of SciML

Scientific Machine Learning is thriving [2] …

- Extends traditional surrogate modeling and function approximation to larger scale problems (mesh data) [5,7].
- Encompasses new techniques like Physics informed learning ([5,6]., this workshop) to refine the quality of the approximation and foster practitioner's trust in those models

…but surrogate models and numerical schemes are not considered equals

- Such models are data driven and lack strict guarantees as seen classical numerical schemes
- Some workaround to leverage ML without affecting the guarantees:
  - ML-driven preconditioning [9], Mesh initialization [13],…

Still, the performances of next gen surrogate models are so good as is…

…Couldn't we provide strict **approximation guarantees for SciML models**?

Finding an uniform error bound for g

We approximated a function $f: \mathcal{X} \in \mathbb{R}^d \rightarrow \mathbb{R}$ using a neural network g and a set of learning points $\big(X_1, Y_1 = f(X_1)\big), \dots, \big(X_n, Y_n = f(X_n)\big)$

Now, **can we provide approximation guarantees after the training using g and $\big(X_1, Y_1 = f(X_1)\big), \dots, \big(X_n, Y_n = f(X_n)\big)$ only?**

By finding bounds on

$$J_g = \|f - g\|_\infty = \max_{x \in \mathcal{X}} |f(x) - g_\theta(x)|$$

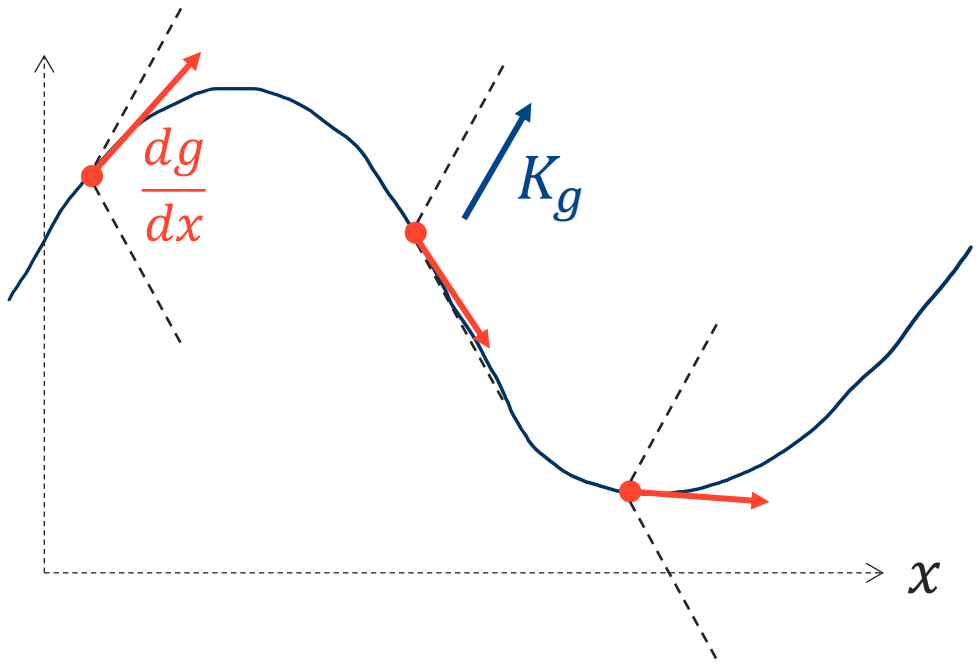
In the following, we try to bound $\|f - g\|_\infty$, the max. of the absolute error with a bound $\bar{J}_g$.
To that end, we will leverage the properties of Lipschitz neural networks

Lipschitz Neural Networks

A function f is said Lipschitz continuous, of constant K_f if :

$$\forall x, y \in \mathbb{R}^d, |f(x) - f(y)| \leq K_f \times \|x - y\|$$

A neural network g is said K_g-Lipschitz when it satisfies the above property.



Its rate of change is bounded by K_g

Lipschitz Neural Networks

Usual applications of Lipschitz neural networks:

- **Improved (and certified) robustness to adversarial attacks [11]**
- Better generalization for classification tasks [3]
- Better explainability [10]
- Perform well in Wasserstein distance estimation [3,11]

Original
(class: w/o)

Minimum adversarial perturbation
Classical neural net.
(class: w)

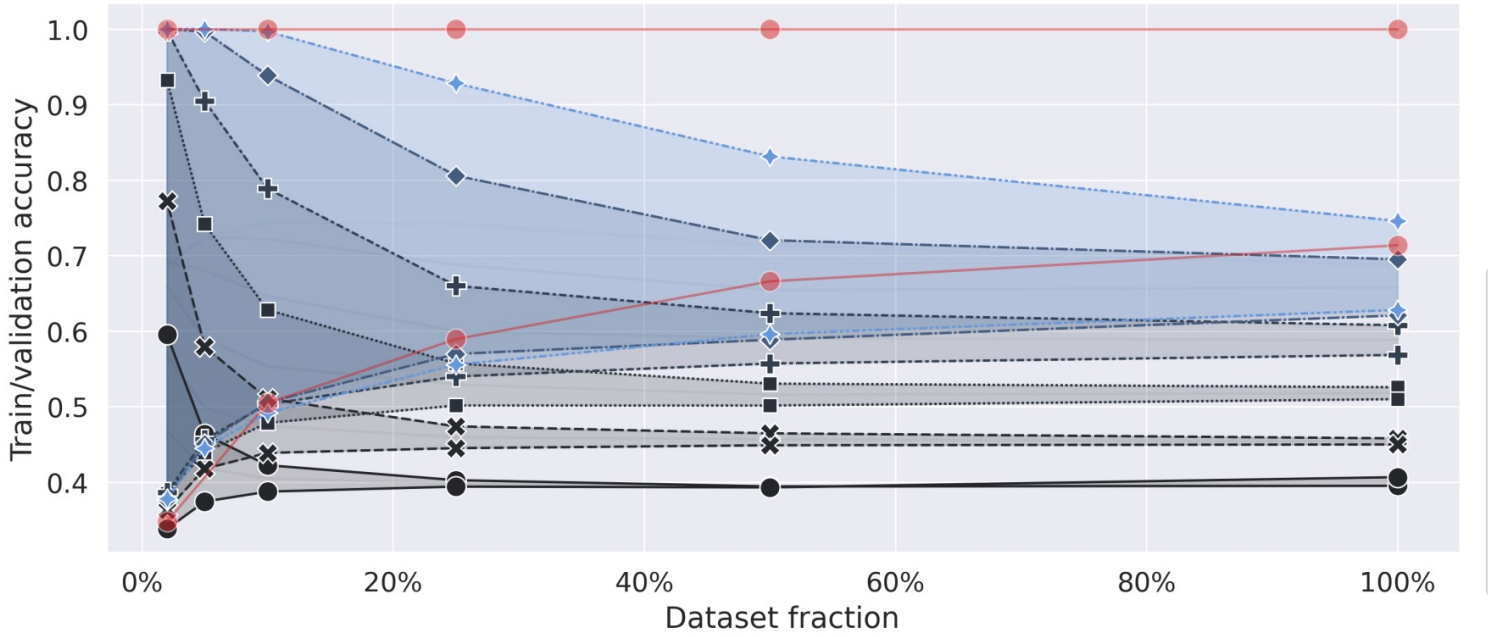
Minimum adversarial perturbation
Lipschitz neural net.
(class: w)

Adversarial perturbation on CelebA dataset
(binary classification of w vs w/o glasses

Lipschitz Neural Networks

Usual applications of Lipschitz neural networks:

- Improved (and certified) robustness to adversarial attacks [11]
- **Better generalization for classification tasks [3]**
- Better explainability [10]
- Perform well in Wasserstein distance estimation [3,11]

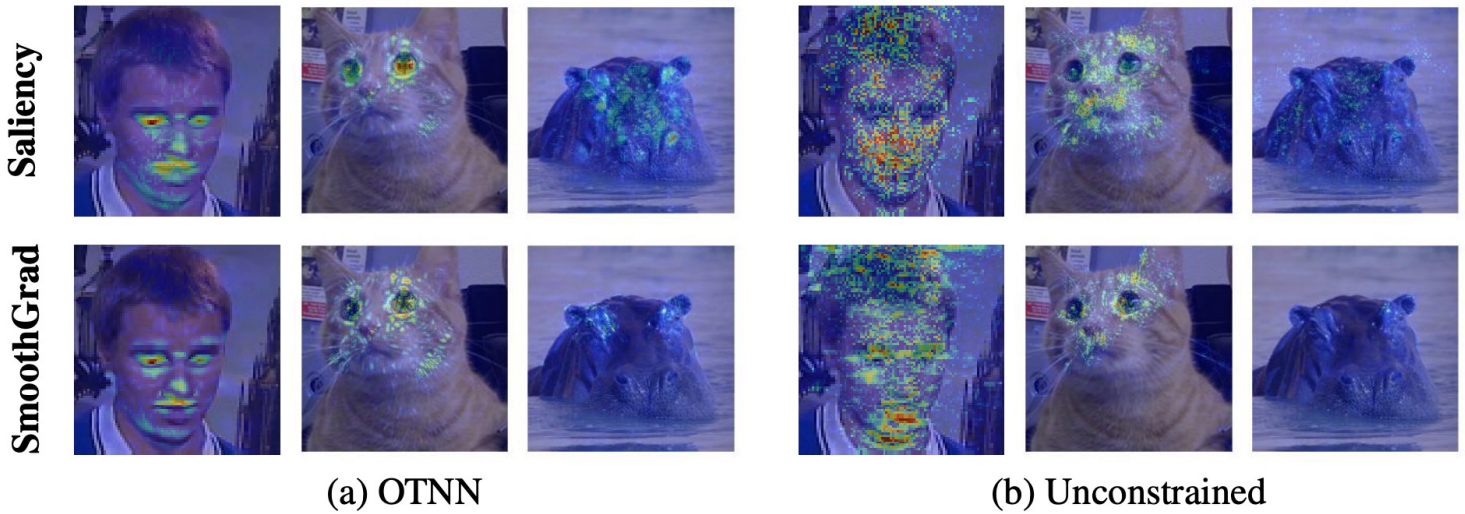


Generalization gap for Lipschitz NN with different K_g vs a classical neural network (in red)

Lipschitz Neural Networks

Usual applications of Lipschitz neural networks:

- Improved (and certified) robustness to adversarial attacks [11]
- Better generalization for classification tasks [3]
- **Better explainability [10]**
- Perform well in Wasserstein distance estimation [3,11]



Explanation maps for a Lipschitz network (OTNN) vs a classical network (Unconstrained)

Usual applications of Lipschitz neural networks:

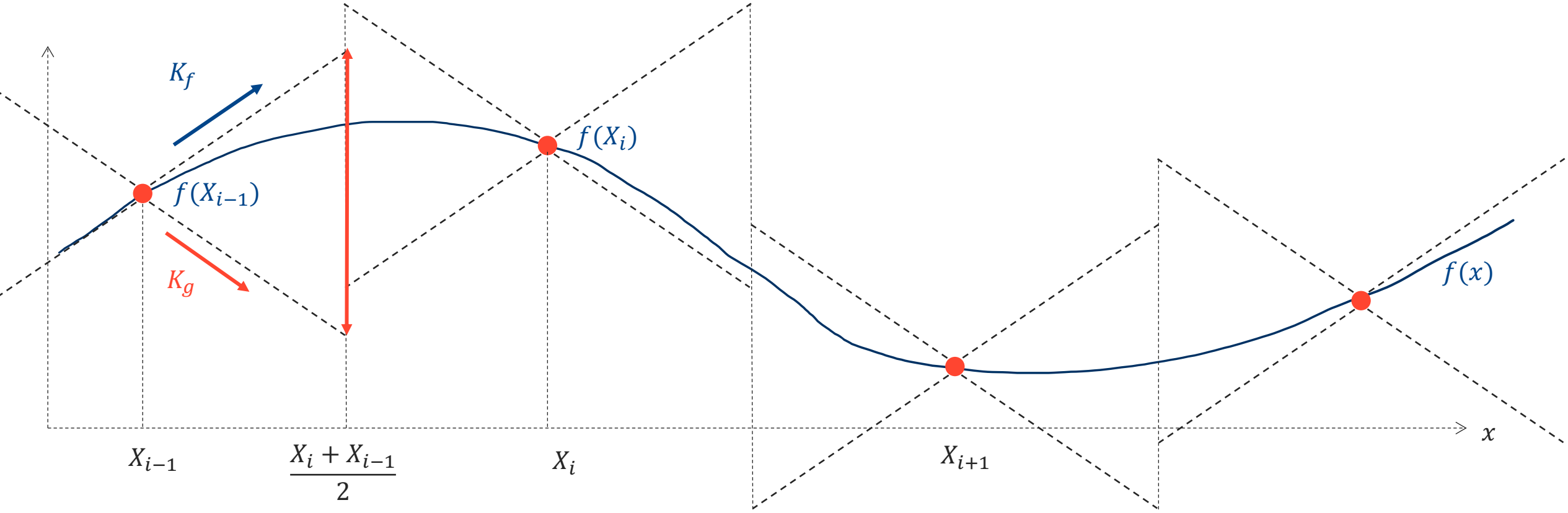
- Improved (and certified) robustness to adversarial attacks [11]
- Better generalization for classification tasks [3]
- Better explainability [10]
- **Perform well in Wasserstein distance estimation [3,11]**

Wasserstein-1 distance:

$$W_1(\mu,\eta) = \max_{f \in L_1} \int f(x) d(\mu - \eta)(x)$$

Can be found by approximating the set of 1-Lipschitz functions with 1-Lipschitz neural nets and perform the optimization

Motivation: Error bound in 1D

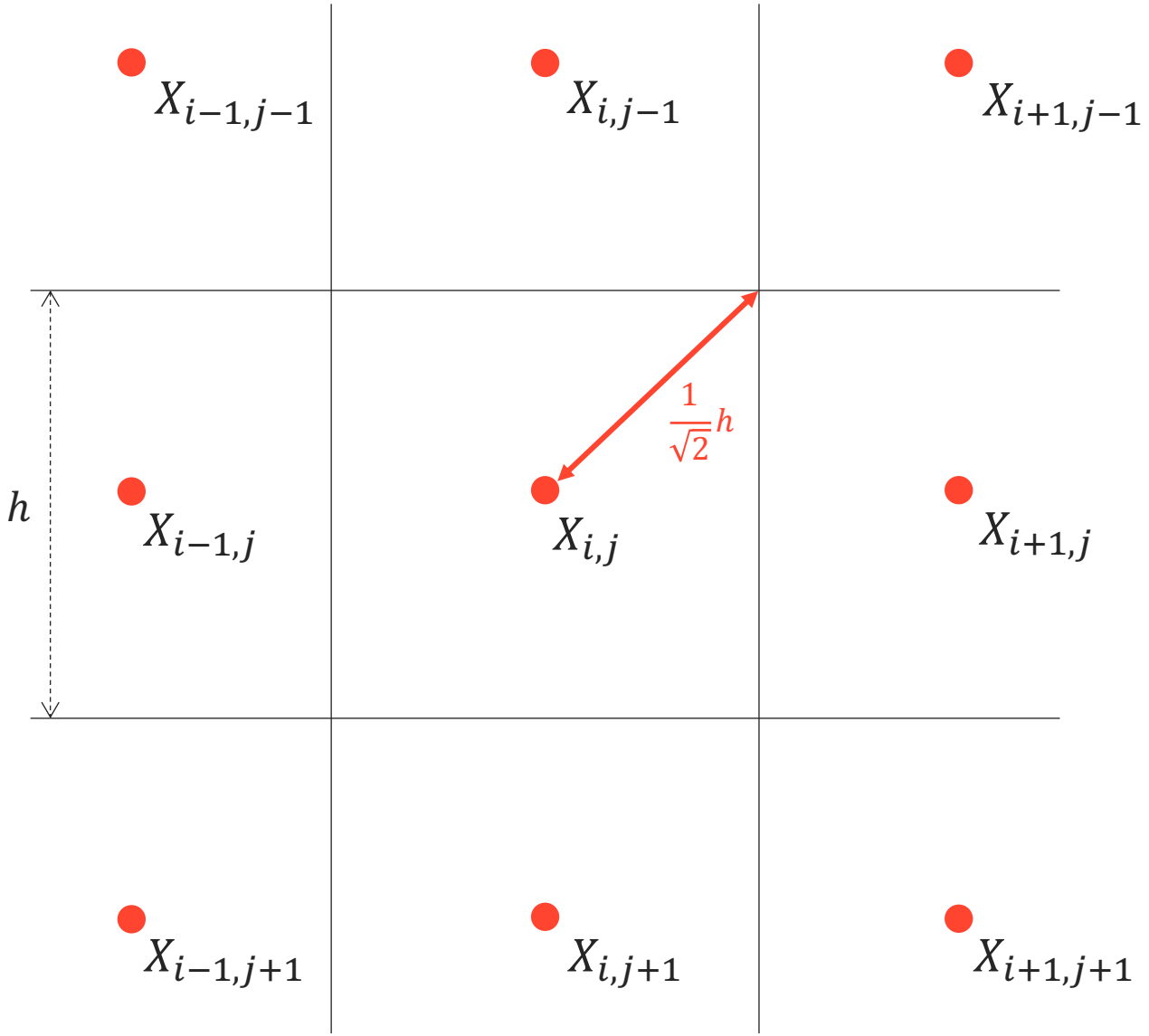


Take the difference between maximum variation of f and g on each subdivision:

$$J_g \leq \max_{i \in \{1,\dots,n\}} \frac{1}{2}\left(K_g + K_f\right)\|X_i - X_{i-1}\| + \boxed{|f(X_i) - g(X_i)|}$$

$= 0$ in this example

Motivation: Error bound in 2D (and beyond)



Bound in 2D ($d = 2$):

- Consider n^2 learning points $\{X_{i,j}\}_{i,j\in\{1,\dots n\}\text{^}2}$ at the center of a grid with cells of edge size h.

In the k-th cell of center $X_{i,j}$:

$$J_g^k \leq \left|f(X_{i,j}) - g(X_{i,j})\right| + \frac{1}{\sqrt{2}}(K_f + K_g)h = \bar{J}_g^k$$

Bound in ND ($d = N$):

In the k-th cell of center X_p:

$$J_g^k \leq \left|f(X_p) - g(X_p)\right| + \frac{\sqrt{N}}{2}(K_f + K_g)h = \bar{J}_g^k$$

Then,

$$J_g \leq \max_k \bar{J}_g^k$$

Breaking free from grids

Main problem: Learning points are rarely structured as a grid

What about learning in the context of Scientific ML?

We control the design of experiment so we could build it as a grid

Very constraining:

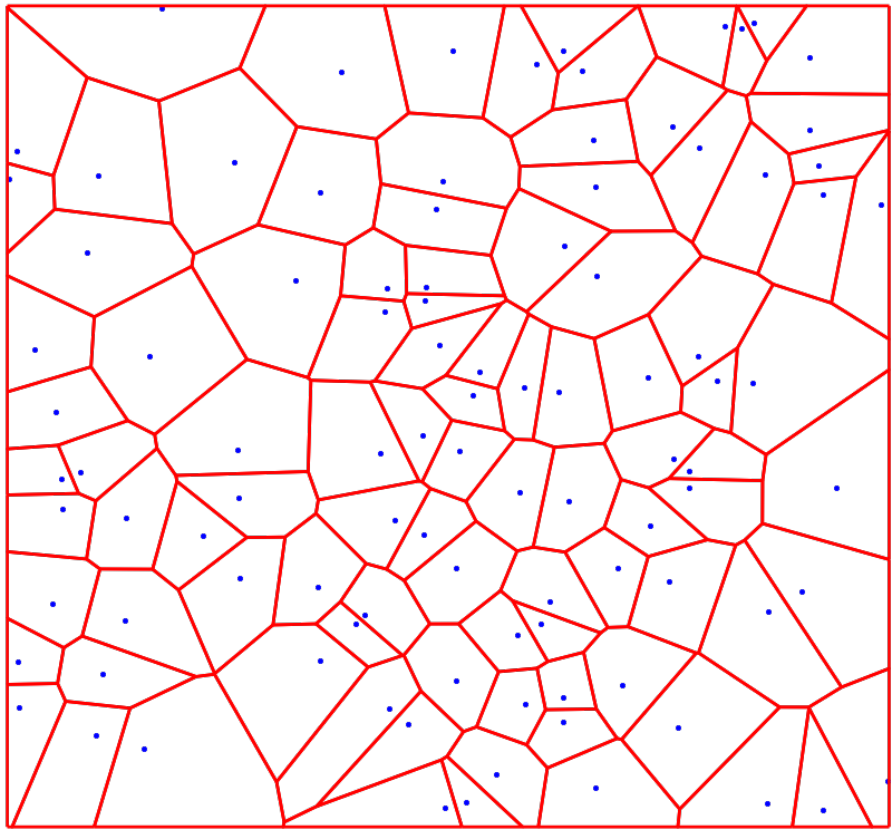
- The DOE should be defined in advance and we could not add points sequentially
- Grids suffer from the curse of dimensionality, the number of f evaluations would grow exponentially with d
- Monte Carlo is convenient

Aim of this work: find ways to build upper bounds for J_g when $\left(X_1, Y_1 = f(X_1)\right), \ldots, \left(X_n, Y_n = f(X_n)\right)$ is not structured as a grid

Outline

- Introduction
- **Error bound with Voronoï diagrams**
- Overcoming Voronoï diagrams complexity
- Conclusion & Takeaway

Definition of a Voronoï diagram (and some notations)



A Voronoï diagram $\mathcal{V}^d$ is built on a set of points $\mathbf{X} = \{X_1, \dots, X_n\}, \mathrm{X_i} \in \mathcal{X} \subset \mathbb{R}^d$.

Each point is called a site, and the diagram is defined by its cells $\{\mathcal{V}^d(X_1), \dots, \mathcal{V}^d(X_n)\}$ themselves defined by

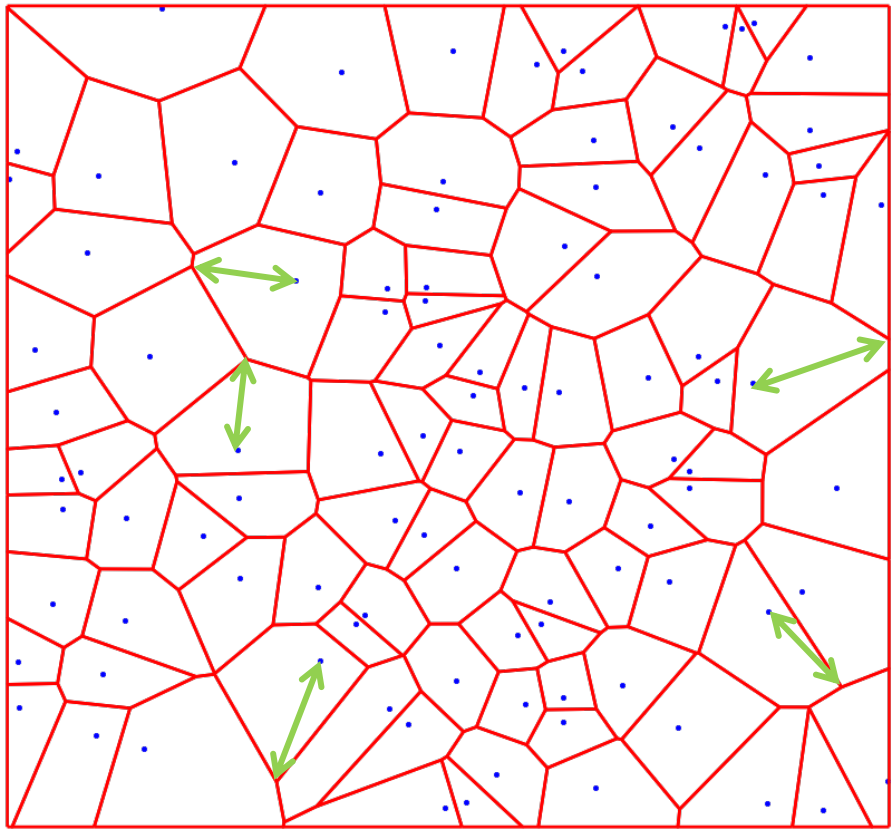
$$\mathcal{V}^d(X_i) = \{x \in \mathcal{X} | \forall j \in \{1, \dots, n\}, \|x - X_i\| \leq \|x - X_j\|\}$$

If $x \in \mathcal{V}^d(X_i)$, then X_i is the nearest neighbor of x

We have that $\mathcal{X} = \mathrm{U}_{i \in \{1,\dots,n\}} \mathcal{V}^d(X_i)$, so to obtain $\bar{J}_g$, it is enough finding $\bar{J}_g^i$, an upper bound for

$$J_g^i = \max_{x \in \mathcal{V}^d(X_i)} |f(x) - g(x)|$$

Error bound using Voronoï diagram



Let $N: x \to \underset{X_i \in \mathbf{X}}{argmin} \|x - X_i\|$ (nearest neighbor map)

Then by the Lipschitz property of g and f, we have that $\forall x \in \mathcal{X}$,

$$|f(x) - g(x)| \leq (K_f + K_g)\|x - N(x)\| + |f(N(x)) - g(N(x))|$$

Lemma 1

Goes well with Voronoï diag!

Let $r(X_i)$ be the radius of $\mathcal{V}^d(X_i)$ defined by

$$r(X_i) = \max_{x \in \mathcal{V}^d(X_i)} \|x - X_i\|$$

Then, it holds that

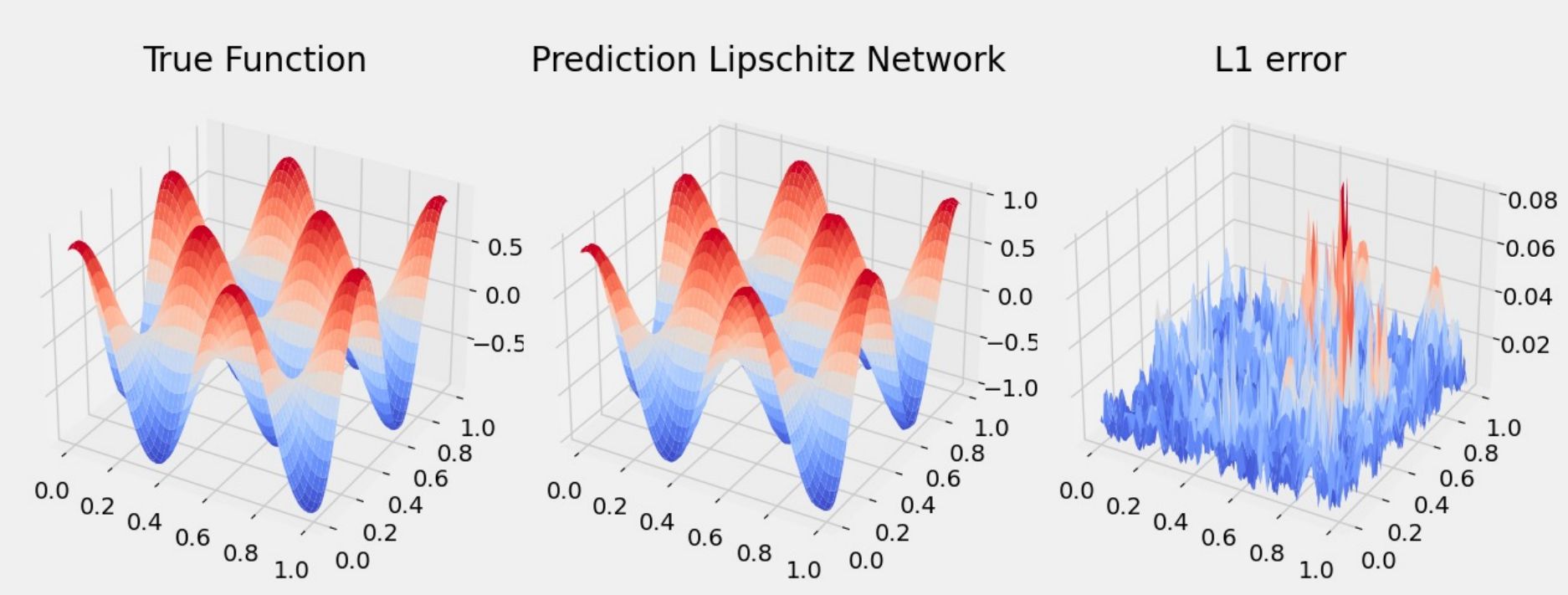
$$J_g^i \leq |f(X_i) - g(X_i)| + (K_f + K_g) r(X_i)$$

Hence,

$$J_g \leq \max_{i \in \{1,\dots,n\}} |f(X_i) - g(X_i)| + (K_f + K_g) r(X_i)$$

- All we need is to compute $r(X_i)$

Experiments on toy functions

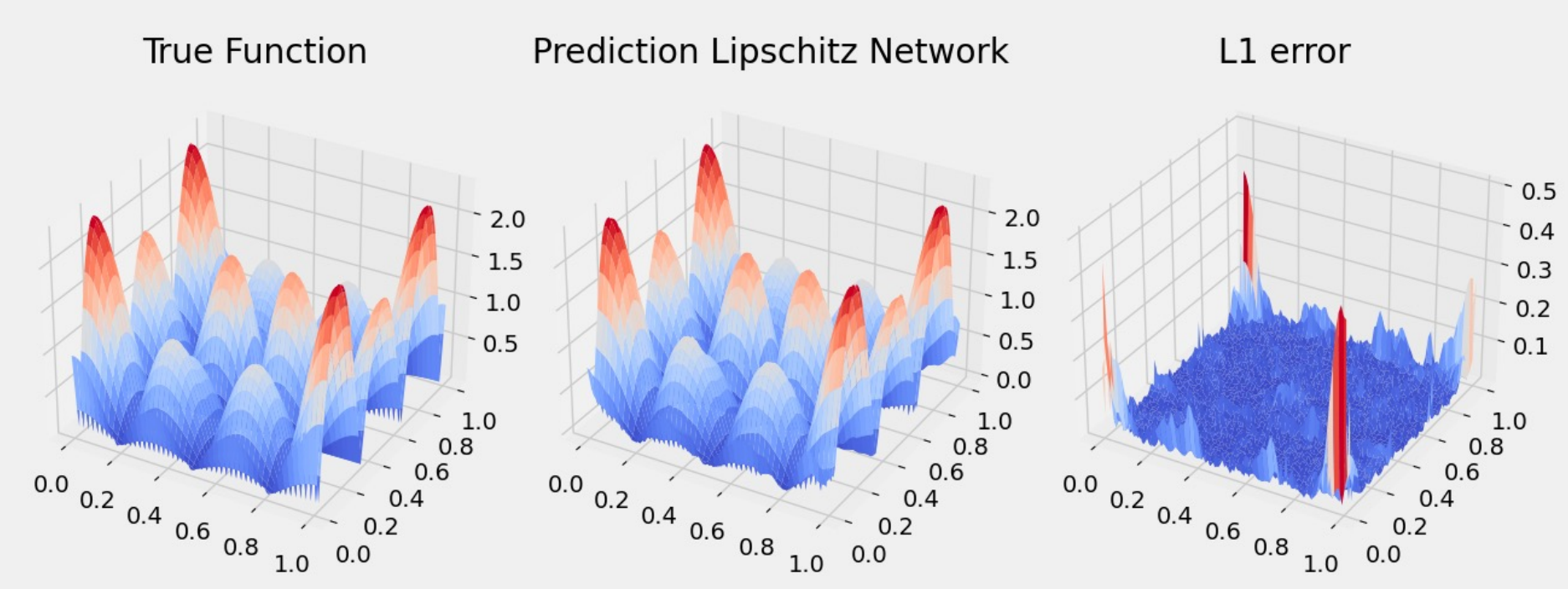


Sinus function

$$f : x, y \rightarrow \sin(x) \times \sin(y)$$

10000 training points

Experiments on toy functions



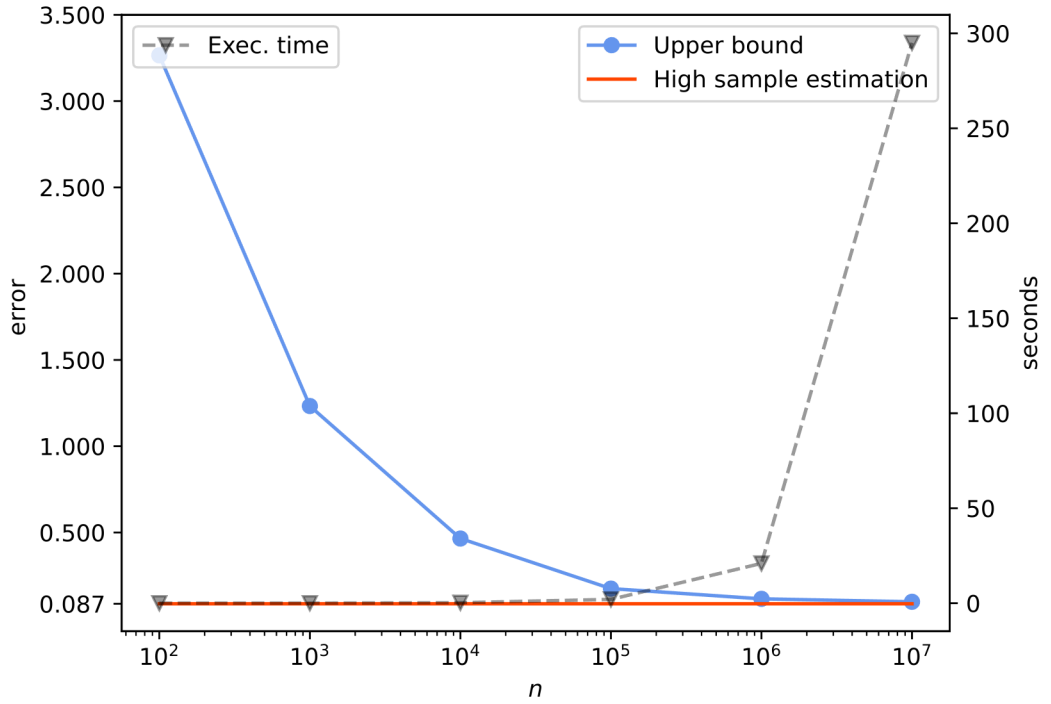
Holder table function

$$f : x, y \rightarrow \left| \sin(x) \cos(y) \exp\left(\left| 1 - \frac{\sqrt{x^2 + y^2}}{\pi} \right| \right) \right|$$

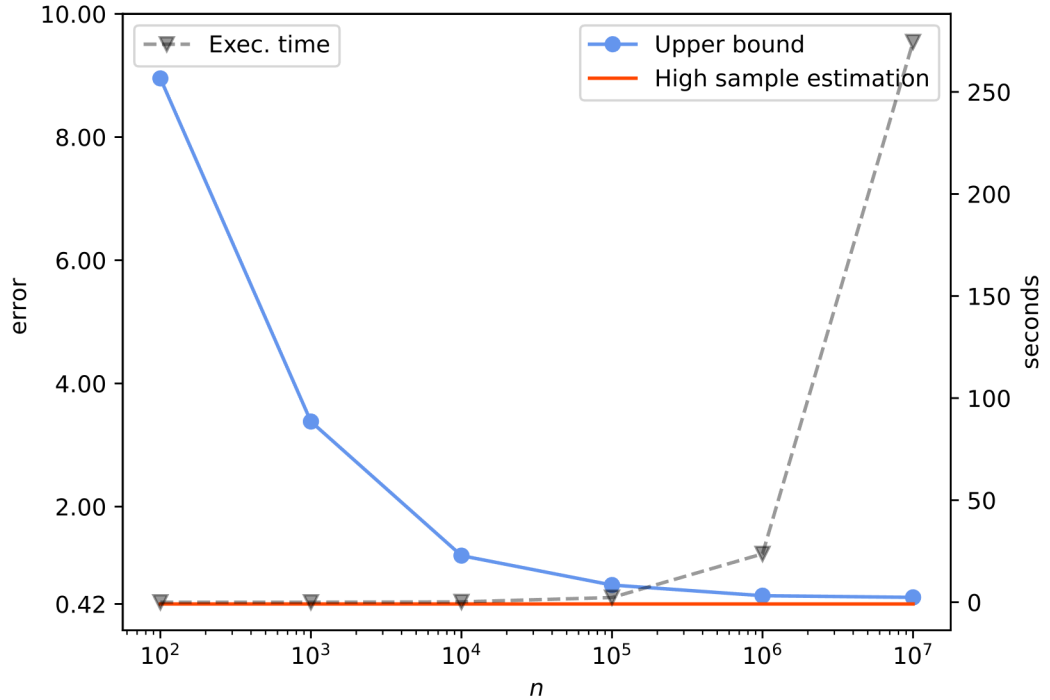
10000 training points

Complexity of Voronoï diagrams

Upper bound of L_∞ error with computation time for Sinus function (left) and Holder table function (right)



Best upper bound: 0.098
High sample estimation: 0.087



Best upper bound: 0.53
High sample estimation: 0.42

Problem: Voronoï diagram's complexity is exponential…

… what about higher d and n?

Learning heat diffusion

Diffusion in 2D:

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

- We simulate heat diffusion on a homogeneous surface , with 4 Dirichlet boundary conditions and observe the field at convergence.
- The final heat field depends on the boundary conditions, but not on the initial state nor the diffusivity.

Design of experiment:

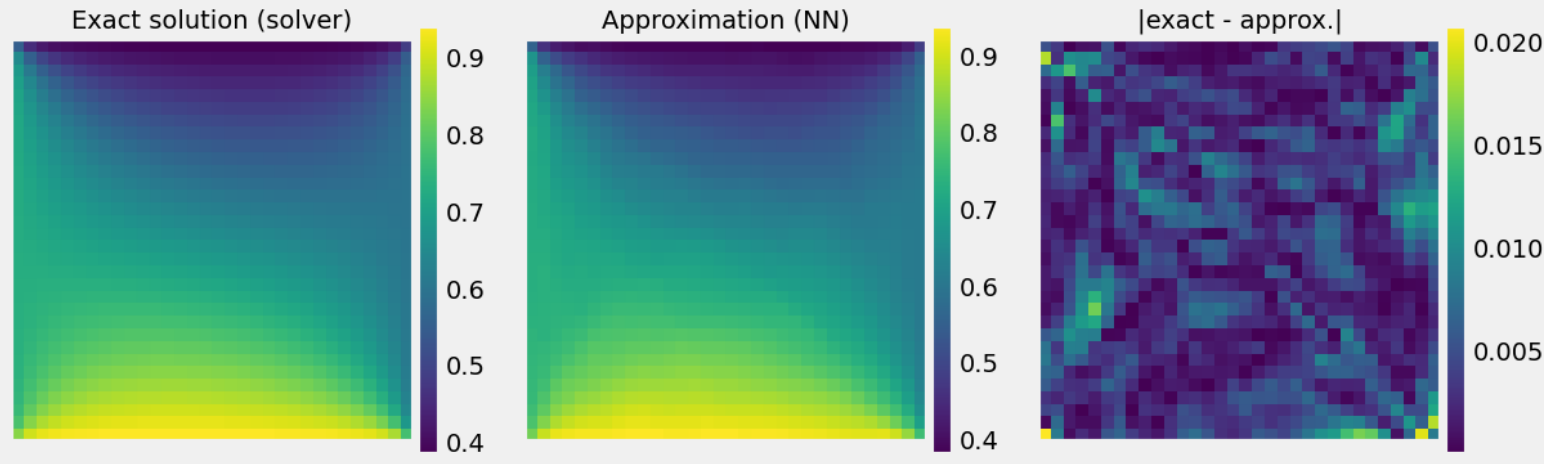
- Sample $n = 5000$ boundary conditions $\{(a_i, b_i, c_i, d_i)\}_{i\in\{1,\dots,n\}}$ uniformly on $[0,1]^4$.
- Conduct n simulations on a $p\times p$ grid ($p = 32$), yielding a temperature field $\{T_{jk}\}_{j,k\in\{1,\dots,p\}^2}$.

Training dataset:

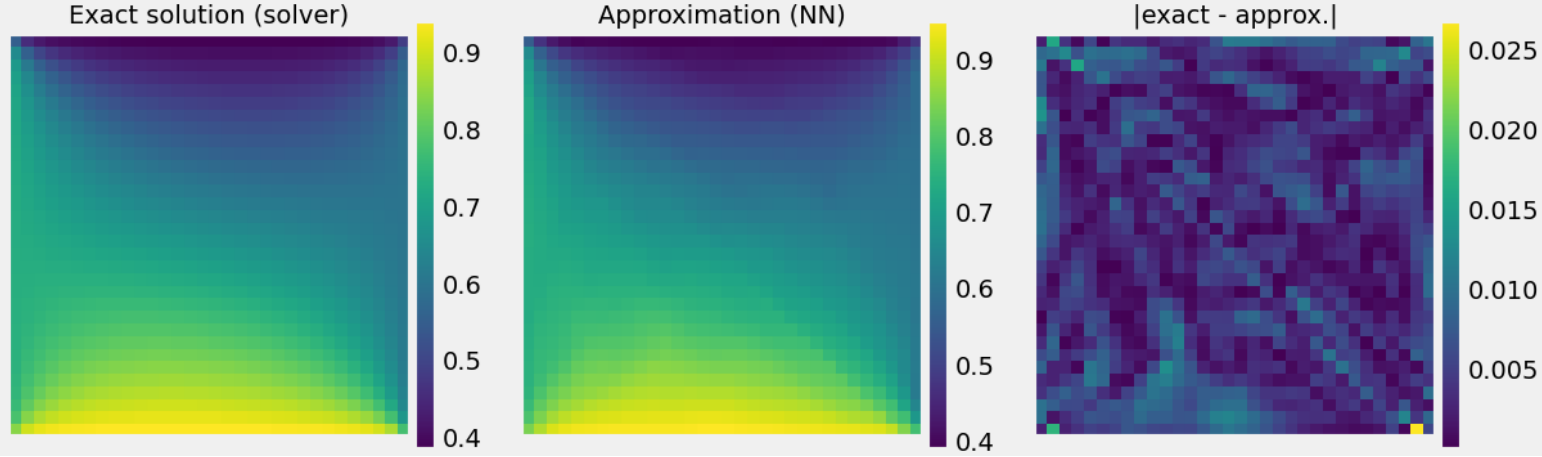
- A subset of $n\times p\times p/10 = 512{,}000$ points $\{(a_i, b_i, c_i, d_i, x_j, x_k), T_{j,k}\}_{i\in\{1,\dots,n\},j,k\in\{1,\dots,p\}^2}$

Neural implicit representation approach!

Approximation results



Lipschitz network, MSE=6.3×10^{-5}



Standard fully connected, MSE=4.1×10^{-5}

How to handle unknown K_f?

Two ways:

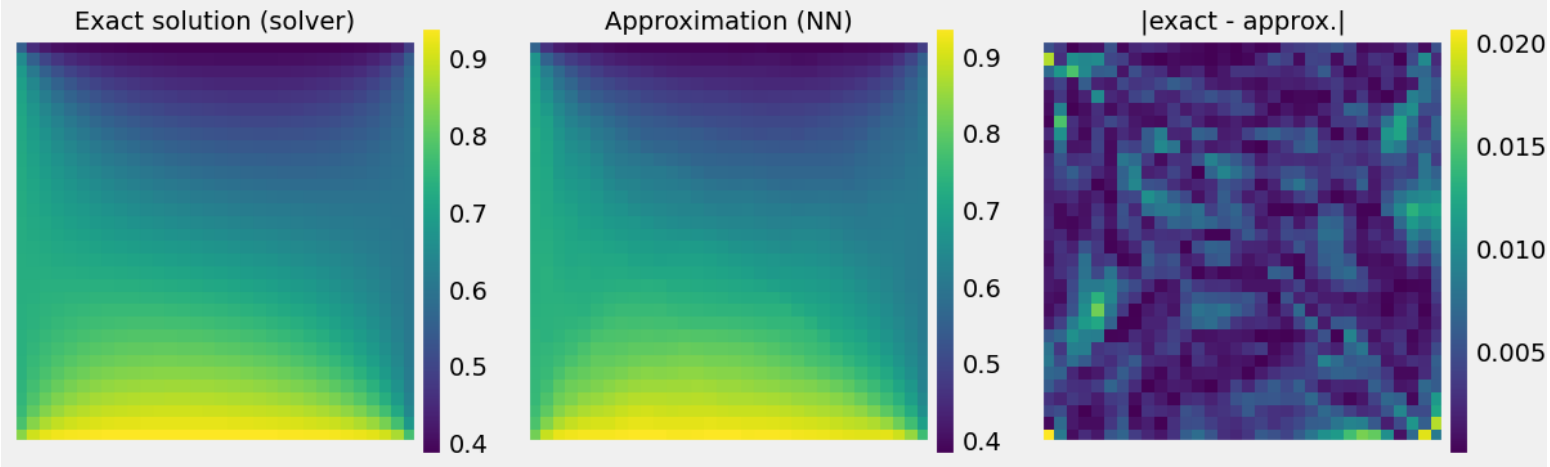
1. Empirical estimation of Lipschitz constant using:

$$\widehat{K_f} = \max_{i \in \{1,\dots,n\}} \left(\max_{X \in \mathcal{N}_k(X_i)} \frac{|f(X) - f(X_i)|}{\|X - X_i\|} \right)$$

   Where $\mathcal{N}_k(X_i)$ is the set of the k-th nearest neighbors of X_i.

2. Hypotheses of f:
   - In [8], the authors compute the Lipschitz constant of f when it is a Gaussian Process interpolating the data.
   - Could apply to polynomial regression
   - We might find the Lipschitz constant by studying the physics [4]

Error bound



Lipschitz network, MSE=6.3×10^{-5}

- Maximum empirical L_1 error: $\mathbf{0.17}$

Voronoï diagram with a subset of 20000 points. Takes ≈ 3000 seconds ($exponential$ complexity…)

- Error bound: $\mathbf{84}$!! Not very appealing…
- We have to find workarounds to use all the $n\times p\times p = 5{,}120{,}000$ points

Outline

- Introduction
- Error bound with Voronoï diagrams
- **Overcoming Voronoï diagrams complexity**
  - Mixed random and mesh datasets
  - Mapping to grid (for a tighter bound?)
- Conclusion & Takeaway

Mixed random and mesh datasets

Suppose that you have a set of points $\mathbf{X}^d = \{X_1^d, \dots, X_n^d\}$ uniformly sampled on a domain $[0,1]^d$.

Now consider a set of points $\{x_1, \dots, x_p\}$ evenly spaced on $[0,1]$.

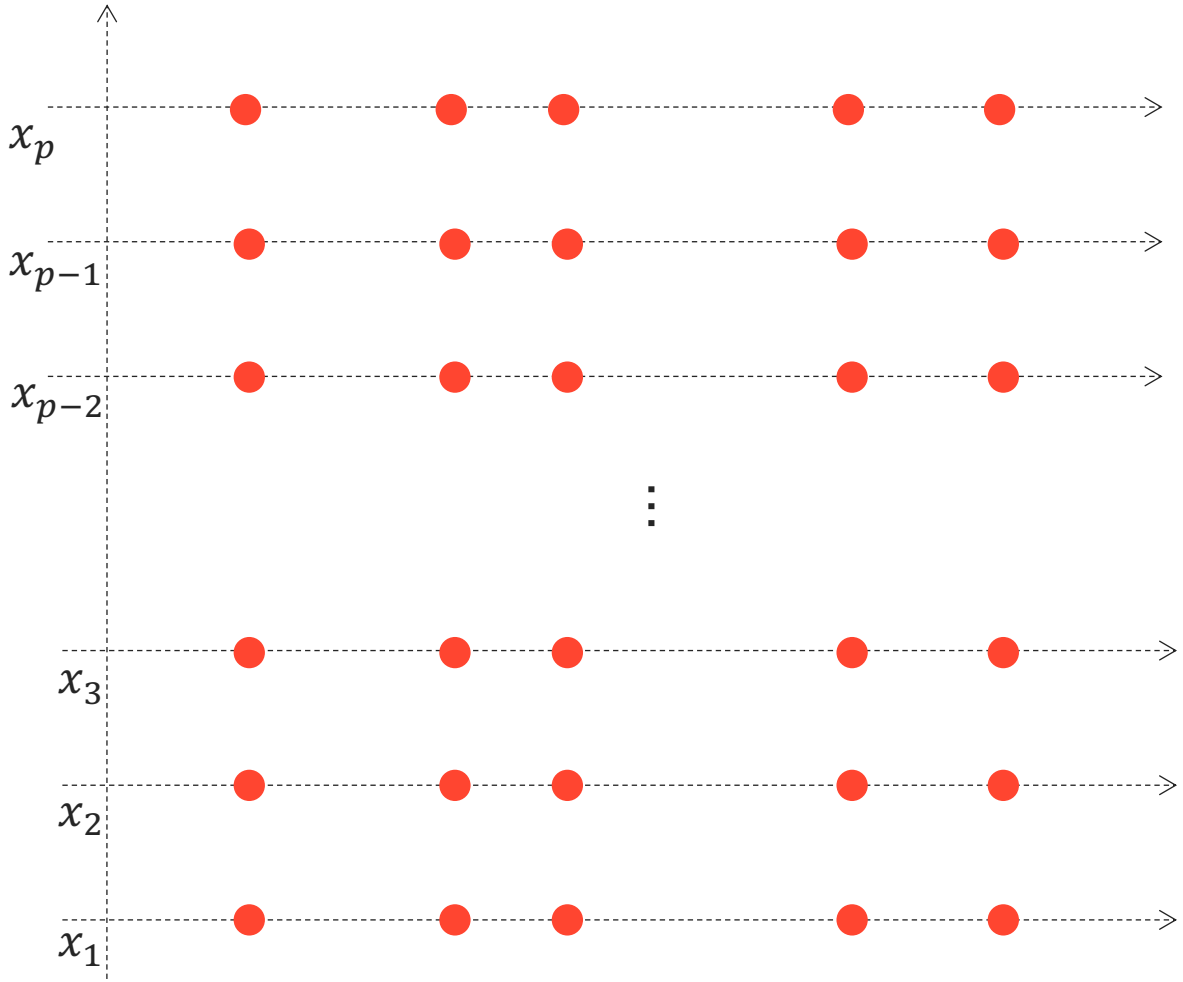
Then define the set of points

$$\mathbf{X}^{d+1} = \{X_{i,j}^{d+1}\}_{i \in \{1,\dots,n\}, j \in \{1,\dots,p\}}$$

such that

$$X_{i,j}^{d+1} = \left((X_i^d)_1, \dots, (X_i^d)_d, x_j \right),$$

where $X_i^d = \left((X_i^d)_1, \dots, (X_i^d)_d \right)$



Here, $d = 1$

Mixed random and mesh datasets

Suppose that you have a set of points $\mathbf{X}^d = \{X_1^d, \dots, X_n^d\}$ uniformly sampled on a domain $[0,1]^d$.

Now consider a set of points $\{x_1, \dots, x_p\}$ evenly spaced on $[0,1]$.

Then define the set of points

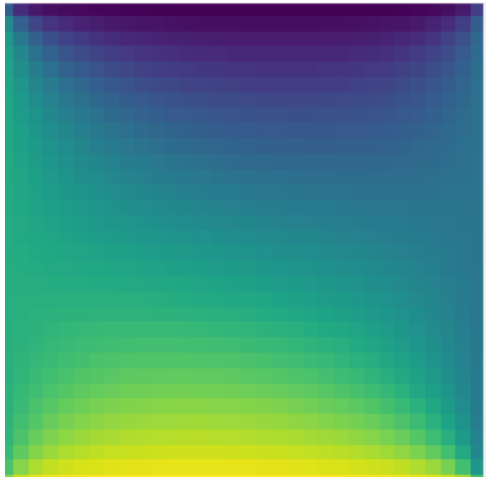
$$\mathbf{X}^{d+1} = \{X_{i,j}^{d+1}\}_{i\in\{1,\dots,n\},j\in\{1,\dots,p\}}$$

such that

$$X_{i,j}^{d+1} = \left(\left(X_i^d\right)_1, \dots, \left(X_i^d\right)_d, x_j\right),$$

where $X_i^d = \left(\left(X_i^d\right)_1, \dots, \left(X_i^d\right)_d\right)$

Example: numerical simulation



- Sample n different boundary conditions uniformly $\{\partial b_1, \dots, \partial b_n\}$
- compute the n simulations on a mesh of size $p\times p$

➢ $n\times p\times p$ learning points $\{(x_i, y_j, \partial b_k)\}_i$

mesh random unif

Mixed random and mesh datasets

Define the set of points

$$\mathbf{X}^{d+1} = \{X_{i,j}^{d+1}\}_{i\in\{1,\dots,n\},j\in\{1,\dots,p\}}$$

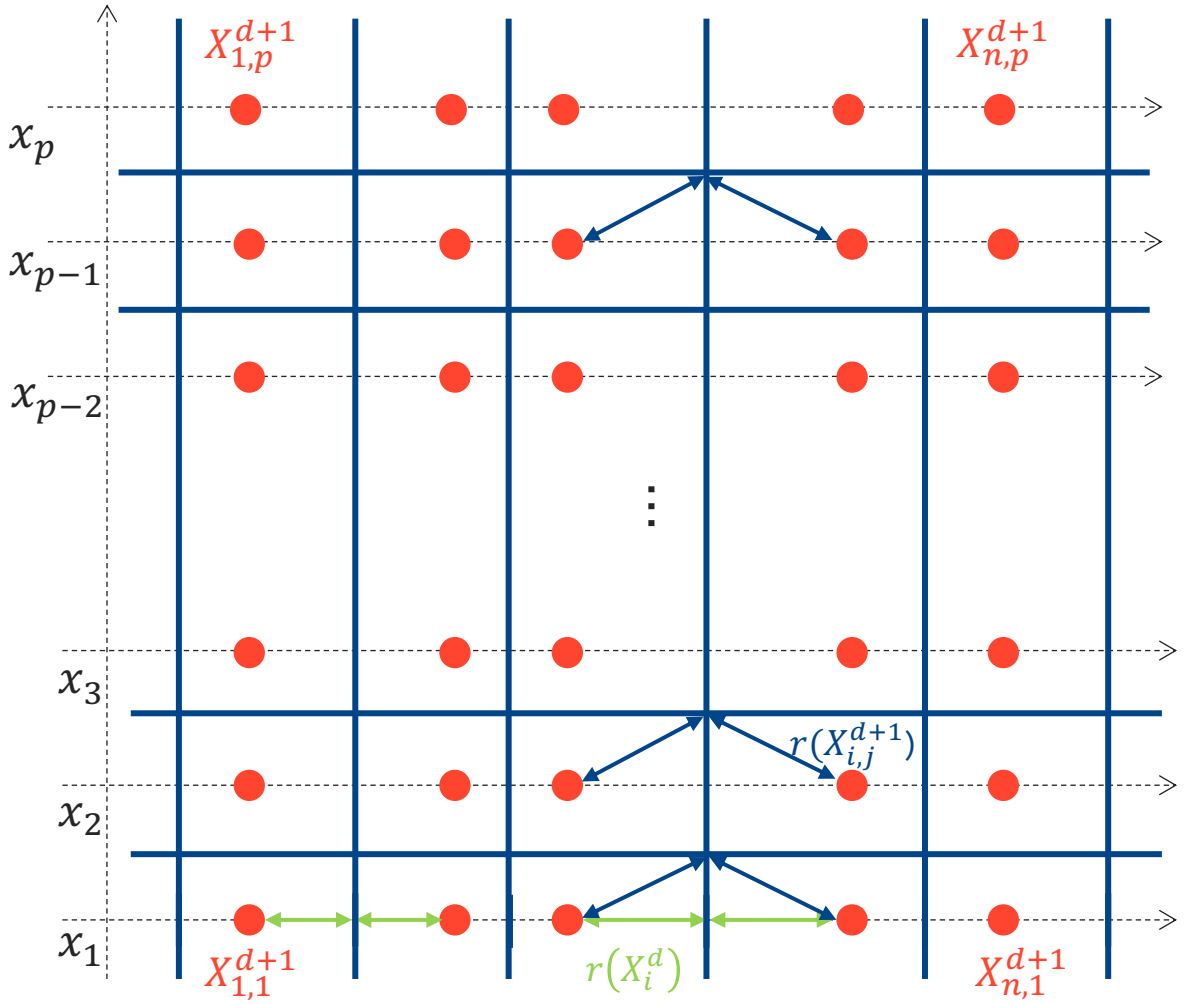
such that

$$X_{i,j}^{d+1} = \left(\left(X_i^d\right)_1, \dots, \left(X_i^d\right)_d, x_j\right),$$

Now, consider $\mathcal{V}^d$ the Voronoï diagram of $\{X_1^d, \dots, X_n^d\}$ and $r(X_i^d)$ the radius of $\mathcal{V}^d(\mathrm{X_i^d})$.

Then: $\forall i \in \{1, \dots, n\}, \forall j, k \in \{1, \dots, p\}^2$,

$$r(X_{i,j}^{d+1}) = r(X_{i,k}^{d+1}) = \sqrt{\frac{1}{4p^2} + r\left(X_i^d\right)^2}$$



Mixed random and mesh datasets

Define the set of points

$$\mathbf{X}^{d+1} = \{X_{i,j}^{d+1}\}_{i\in\{1,\dots,n\},j\in\{1,\dots,p\}}$$

such that

$$X_{i,j}^{d+1} = \left(\left(X_i^d\right)_1, \dots, \left(X_i^d\right)_d, x_j\right),$$

Now, consider $\mathcal{V}^d$ the Voronoï diagram of $\{X_1^d, \dots, X_n^d\}$ and $r(X_i^d)$ the radius of $\mathcal{V}^d(\mathrm{X_i^d})$.

Then: $\forall i \in \{1, \dots, n\}, \forall j, k \in \{1, \dots, p\}^2$,

$$r(X_{i,j}^{d+1}) = r(X_{i,k}^{d+1}) = \sqrt{\frac{1}{4p^2} + r\left(X_i^d\right)^2}$$

In that case, we only need to compute the Voronoï diagram $\mathcal{V}^d$ for $\{X_i^d\}_{i\in\{1,\dots,n\}}$ to obtain $r(X_{i,j}^{d+1})$ and compute the bound!

Practical consequences:

- Compute a Voronoï diagram in dimension $d+1$ with $\mathrm{n}\times p$ points

Becomes

- Compute a Voronoï diagram in dimension d with n points

Mixed random and mesh datasets

Define the set of points

$$\mathbf{X}^{d+1} = \{X_{i,j}^{d+1}\}_{i\in\{1,\dots,n\},j\in\{1,\dots,p\}}$$

such that

$$X_{i,j}^{d+1} = \left(\left(X_i^d\right)_1, \dots, \left(X_i^d\right)_d, x_j\right),$$

Now, consider $\mathcal{V}^d$ the Voronoï diagram of $\{X_1^d, \dots, X_n^d\}$ and $r(X_i^d)$ the radius of $\mathcal{V}^d(\mathrm{X_i^d})$.

Then: $\forall i \in \{1, \dots, n\}, \forall j, k \in \{1, \dots, p\}^2$,

$$r(X_{i,j}^{d+1}) = r(X_{i,k}^{d+1}) = \sqrt{\frac{1}{4p^2} + r\left(X_i^d\right)^2}$$

In that case, we only need to compute the Voronoï diagram $\mathcal{V}^d$ for $\{X_i^d\}_{i\in\{1,\dots,n\}}$ to obtain $r(X_{i,j}^{d+1})$ and compute the bound!

Practical consequences:

Recursivity: For a 2D Grid

- Compute a Voronoï diagram in dimension $d+2$ with $\mathrm{n}\times p\times p$

Becomes

- Compute a Voronoï diagram in dimension d with n points

Mixed random and mesh datasets

Define the set of points

$$\mathbf{X}^{d+1} = \{X_{i,j}^{d+1}\}_{i\in\{1,\dots,n\},j\in\{1,\dots,p\}}$$

such that

$$X_{i,j}^{d+1} = \left(\left(X_i^d\right)_1, \dots, \left(X_i^d\right)_d, x_j\right),$$

Now, consider $\mathcal{V}^d$ the Voronoï diagram of $\{X_1^d, \dots, X_n^d\}$ and $r(X_i^d)$ the radius of $\mathcal{V}^d(\mathrm{X_i^d})$.

Then: $\forall i \in \{1, \dots, n\}, \forall j, k \in \{1, \dots, p\}^2$,

$$r(X_{i,j}^{d+1}) = r(X_{i,k}^{d+1}) = \sqrt{\frac{1}{4p^2} + r\left(X_i^d\right)^2}$$

In that case, we only need to compute the Voronoï diagram $\mathcal{V}^d$ for $\{X_i^d\}_{i\in\{1,\dots,n\}}$ to obtain $r(X_{i,j}^{d+1})$ and compute the bound!

Practical consequences:

Recursivity: For a 3D Grid

- Compute a Voronoï diagram in dimension $d+3$ with $\mathrm{n}\times p\times p\times p$

Becomes

- Compute a Voronoï diagram in dimension d with n points

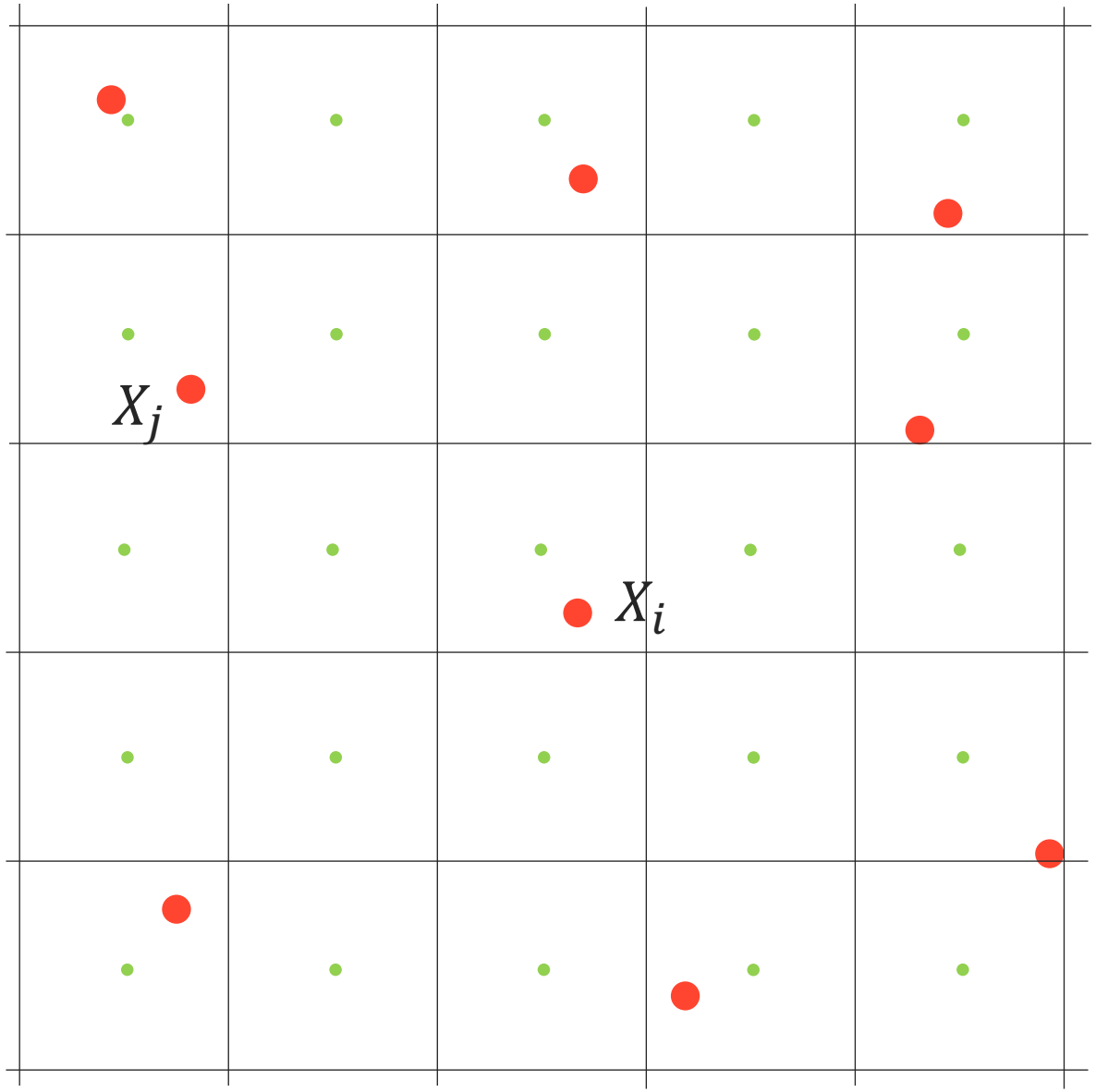
Results on Heat Diffusion

	Classical Voronoï	Mixed random/mesh
Nb points used	20×10^3	$\mathbf{512\times10^4}$
Total eval time (sec.)	> 3000	$\mathbf{1.72}$
Max L_1 error (est.)	0.1716	0.1716
Upper bound	84	$\mathbf{1.6320}$

Results of the different methods for computing $\bar{J}_g$

What if we cannot leverage a mixed grid-random dataset structure?

Mapping to grid



Let's consider $\mathbf{X} = \{X_1, \dots, X_n\}$ uniformly distributed on $[0,1]^d$.

By **lemma 1**, $\forall x \in [0,1]^d$,

$$|f(x) - g(x)| \leq (K_f + K_g)\|x - N(x)\| + |f(N(x)) - g(N(x))|$$

We can do better because **we can evaluate $g(x)$**!

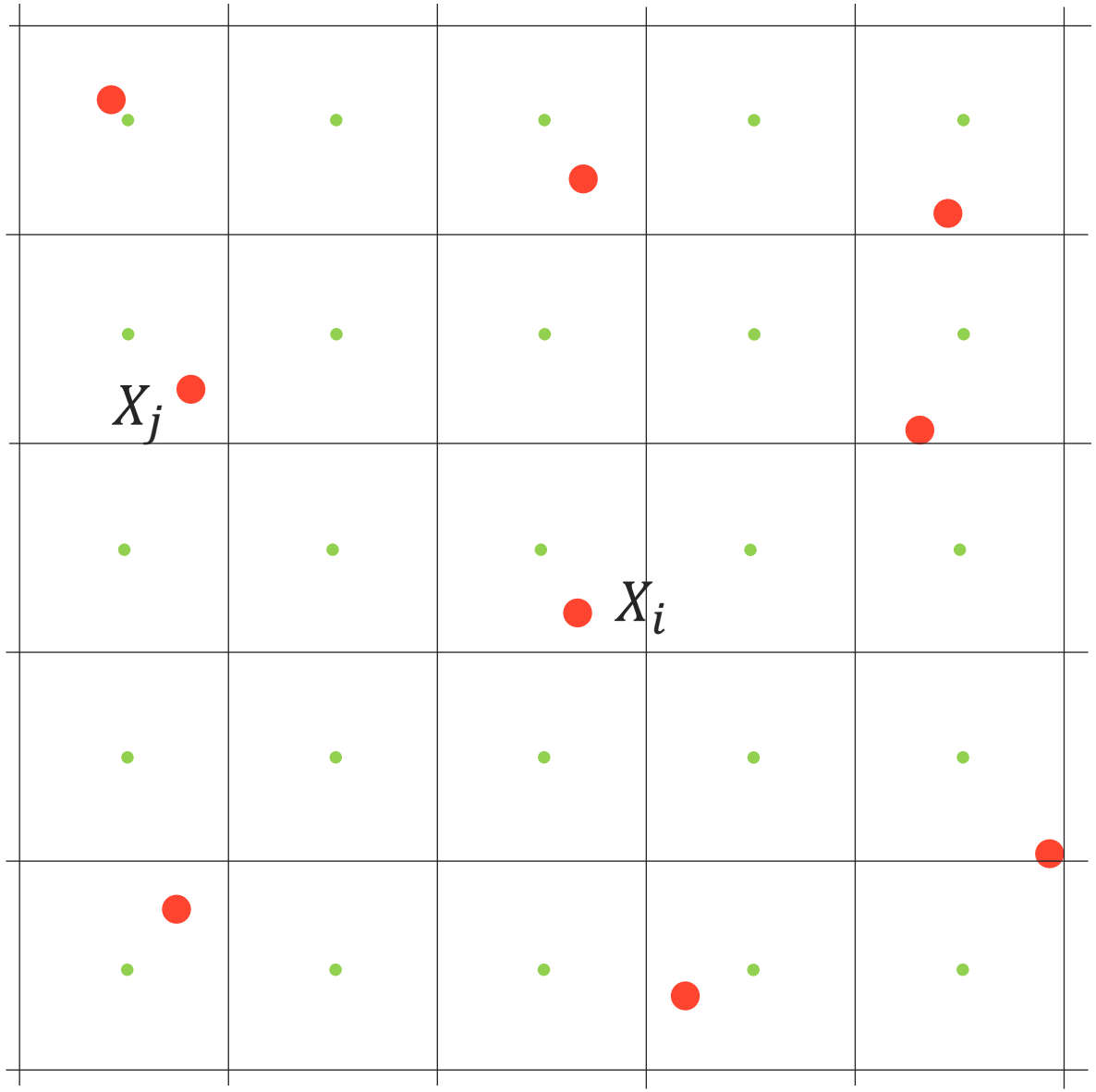
$\forall x \in [0,1]^d$,

$$|f(x) - g(x)| \leq K_f\|x - N(x)\| + |g(x) - g(N(x))| + |f(N(x)) - g(N(x))|$$

Lemma 2

Now, consider a grid of p^d cells with centers $\{c_1, \dots, c_{p^d}\}$

Mapping to grid

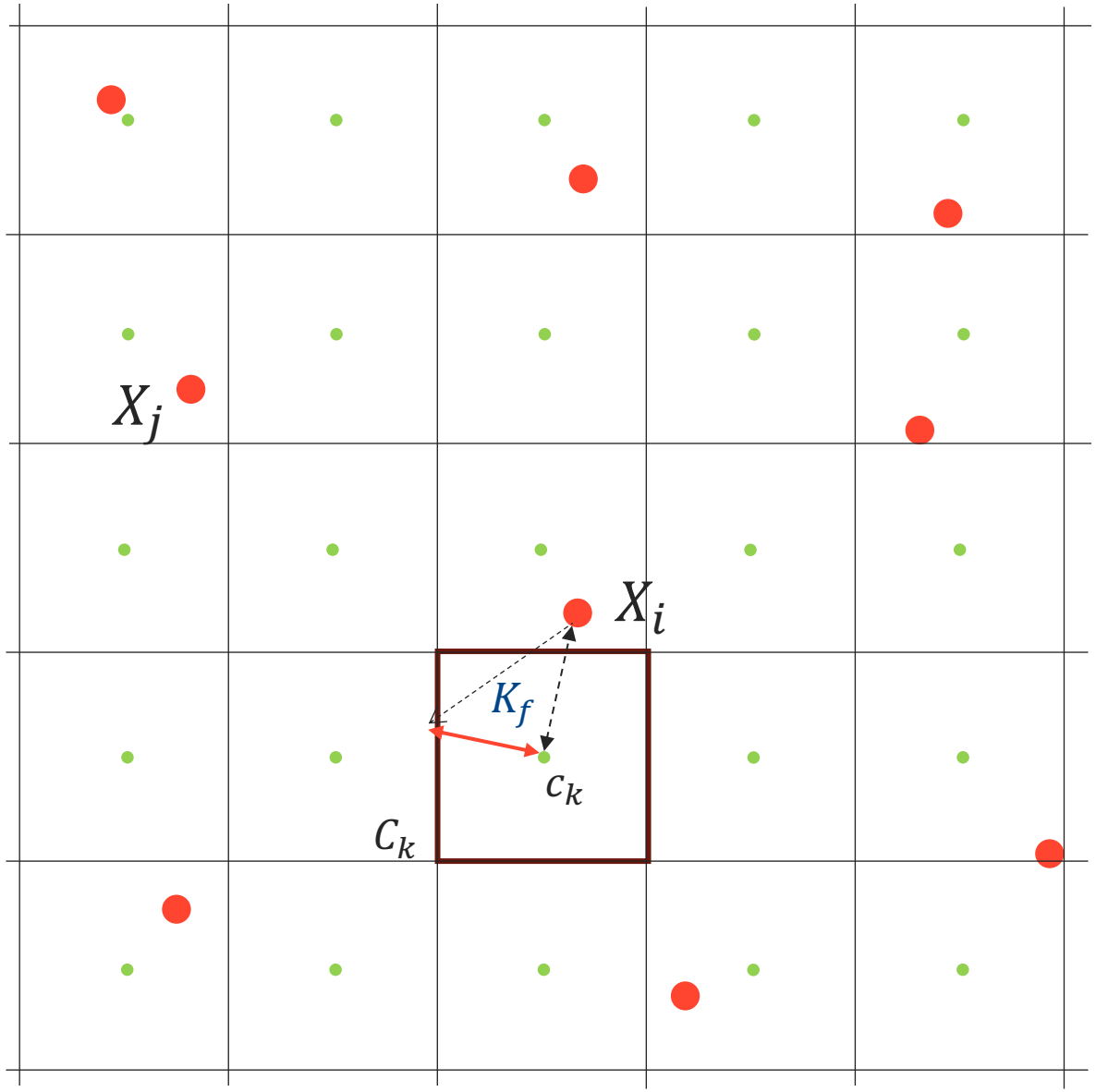


Now, consider a grid of p^d cells $\{C_1, \dots, C_{p^d}\}$ with centers $\{c_1, \dots, c_{p^d}\}$

$\forall k \in \{1, \dots, p^2\}$,

$$|f(c_k) - g(c_k)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + |f(N(c_k)) - g(N(c_k))|$$

Mapping to grid

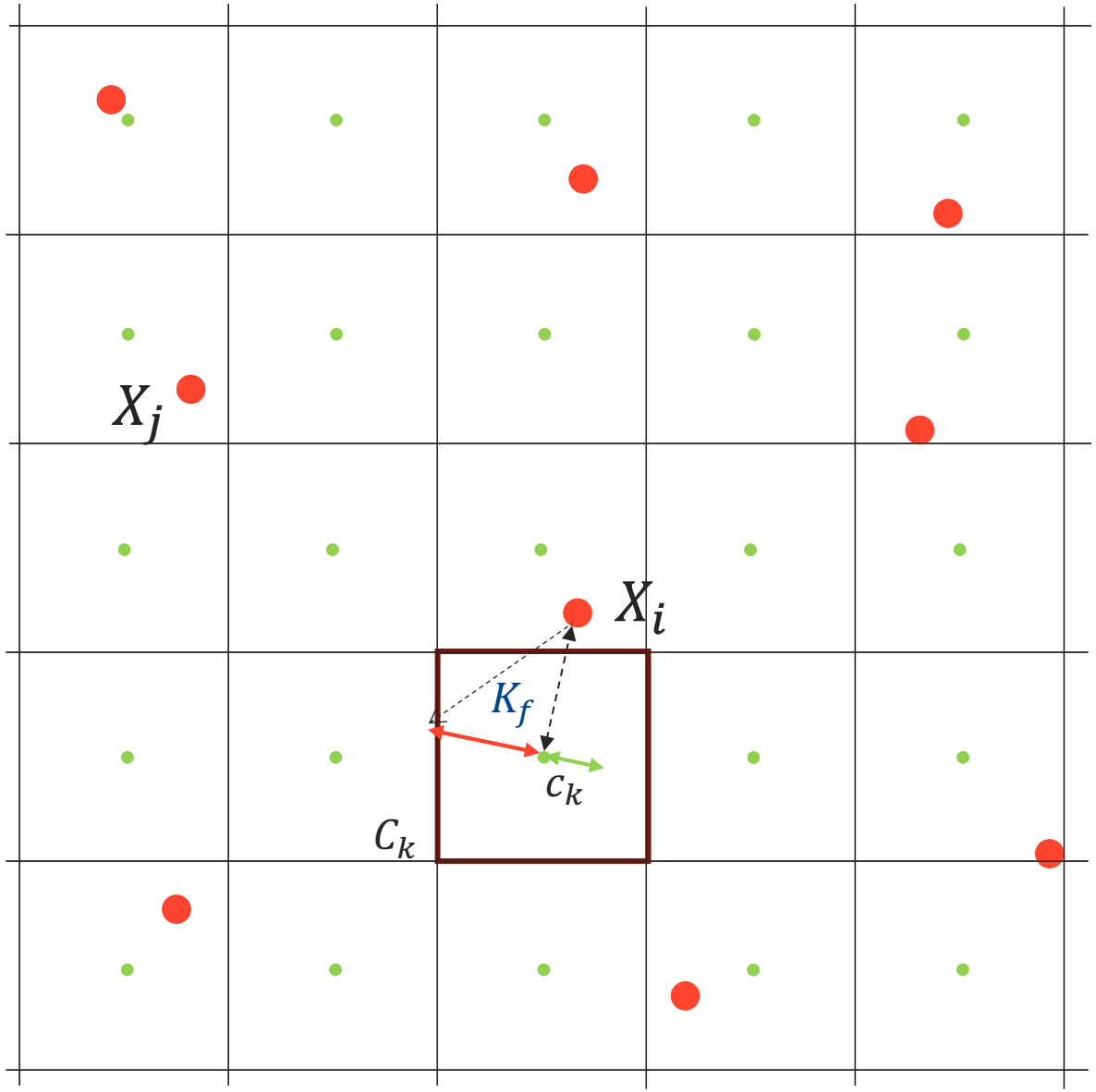


Now, consider a grid of p^d cells $\{C_1, \dots, C_{p^d}\}$ with centers $\{c_1, \dots, c_{p^d}\}$

$\forall k \in \{1, \dots, p^2\}$,

$$|f(c_k) - g(c_k)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + |f(N(c_k)) - g(N(c_k))|$$

Mapping to grid

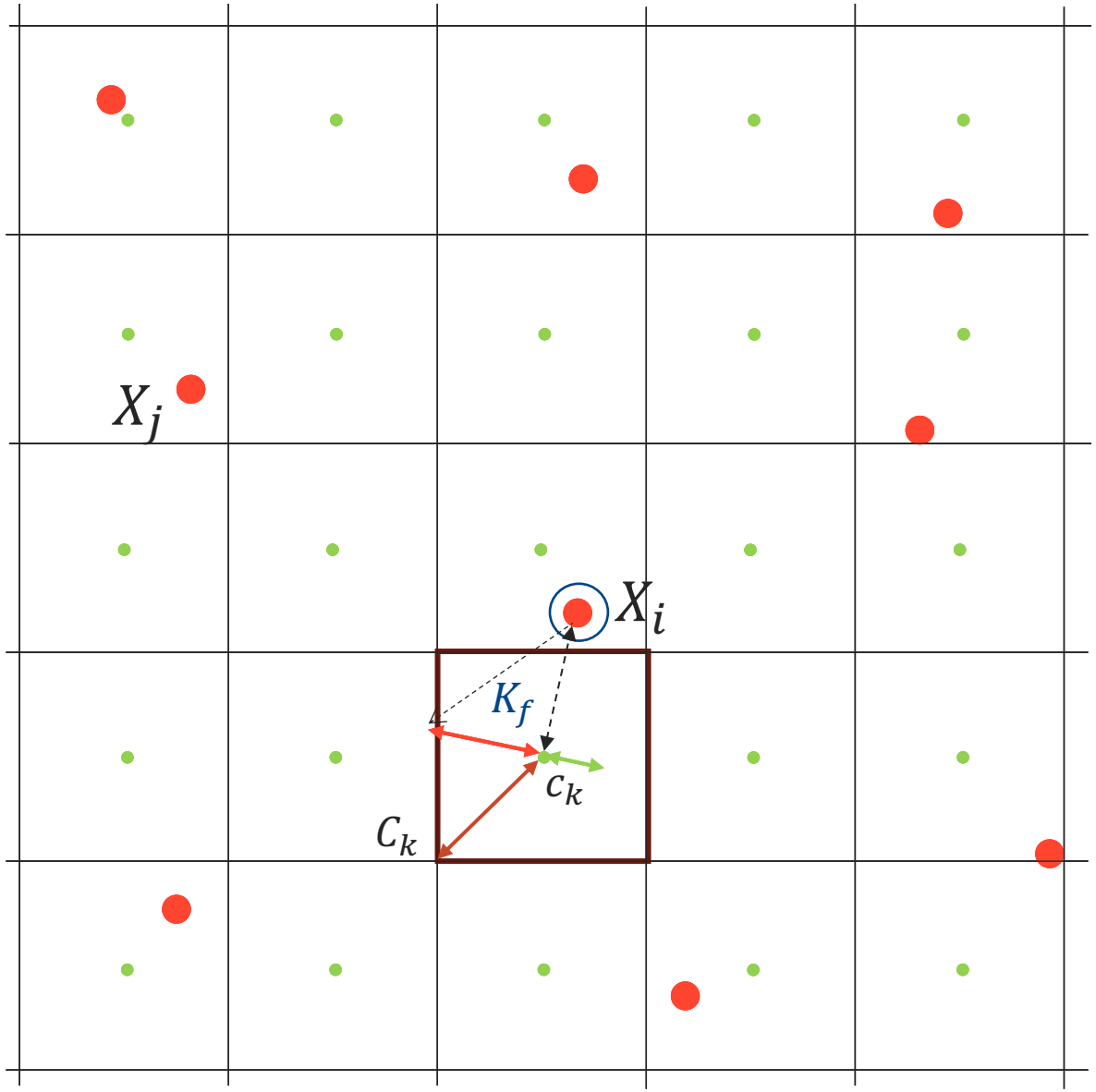


Now, consider a grid of p^d cells $\{C_1, \dots, C_{p^d}\}$ with centers $\{c_1, \dots, c_{p^d}\}$

$\forall k \in \{1, \dots, p^2\}$,

$$|f(c_k) - g(c_k)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + |f(N(c_k)) - g(N(c_k))|$$

Mapping to grid



Now, consider a grid of p^d cells $\{C_1, \dots, C_{p^d}\}$ with centers $\{c_1, \dots, c_{p^d}\}$

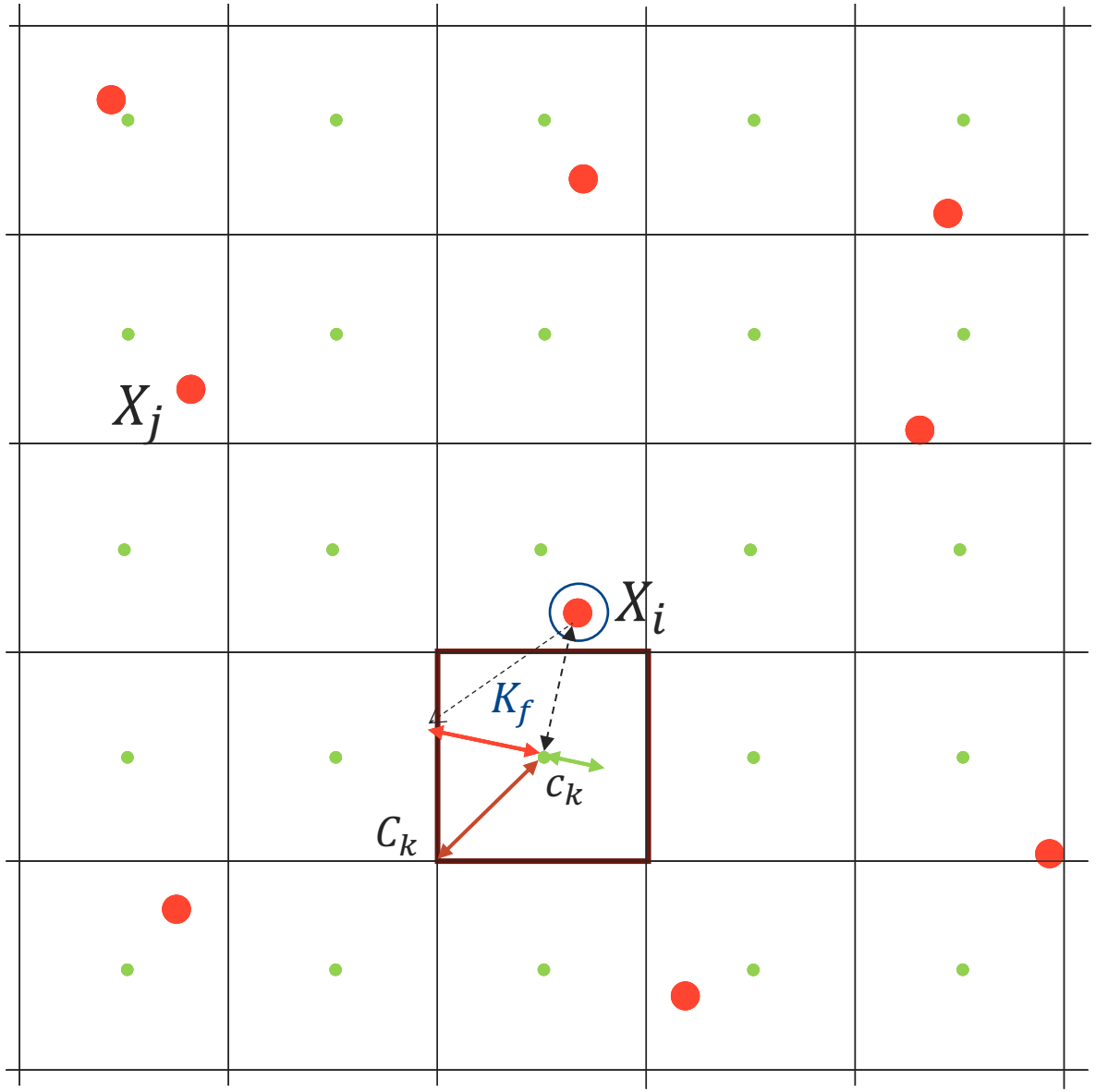
$\forall k \in \{1, \dots, p^2\}$,

$$|f(c_k) - g(c_k)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + |f(N(c_k)) - g(N(c_k))|$$

Since we know that $\forall x \in C_k$,

$$|f(x) - g(x)| \leq |f(c_k) - g(c_k)| + \frac{\sqrt{d}}{2p}(K_f + K_g)$$

Mapping to grid



Now, consider a grid of p^d cells $\{C_1, \dots, C_{p^d}\}$ with centers $\{c_1, \dots, c_{p^d}\}$

$\forall k \in \{1, \dots, p^2\}$,

$$|f(c_k) - g(c_k)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + |f(N(c_k)) - g(N(c_k))|$$

We have that $\forall x \in C_k$,

$$|f(x) - g(x)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + |f(N(c_k)) - g(N(c_k))| + \frac{\sqrt{d}}{2p}(K_f + K_g)$$

Mapping to grid

$\forall x \in C_k,$

$$|f(x) - g(x)| \leq \underbrace{K_f \|c_k - N(c_k)\|}_{\text{Requires calls to a nearest neighbor algorithm to find } N(c_k)} + \underbrace{|g(c_k) - g(N(c_k))|}_{\text{Requires evaluations of } g(c_k), \text{ which can be done in batch very efficiently}} + \underbrace{|f(N(c_k)) - g(N(c_k))|}_{\text{By definition, } N(c_k) \in \mathbf{X}. \text{ Error at each dataset point. Always evaluated for classical validation}} + \underbrace{\frac{\sqrt{d}}{2p}(K_f + K_g)}_{\text{free}}$$

Computational efforts needed:

- Nearest neighbor algorithm
  - Many very efficient libraries (immensely cheaper than Voronoï diagram – complexity not exponential)
  - The bound is still valid with approximate nearest neighbors
- Evaluation of g
  - Very efficient on GPU

Mapping to grid

$\forall x \in C_k$,

$$|f(x) - g(x)| \leq K_f \|c_k - N(c_k)\| + |g(c_k) - g(N(c_k))| + \left|f\left(N(c_k)\right) - g(N(c_k))\right| + \frac{\sqrt{d}}{p}(K_f + K_g)$$

Computational efforts needed:

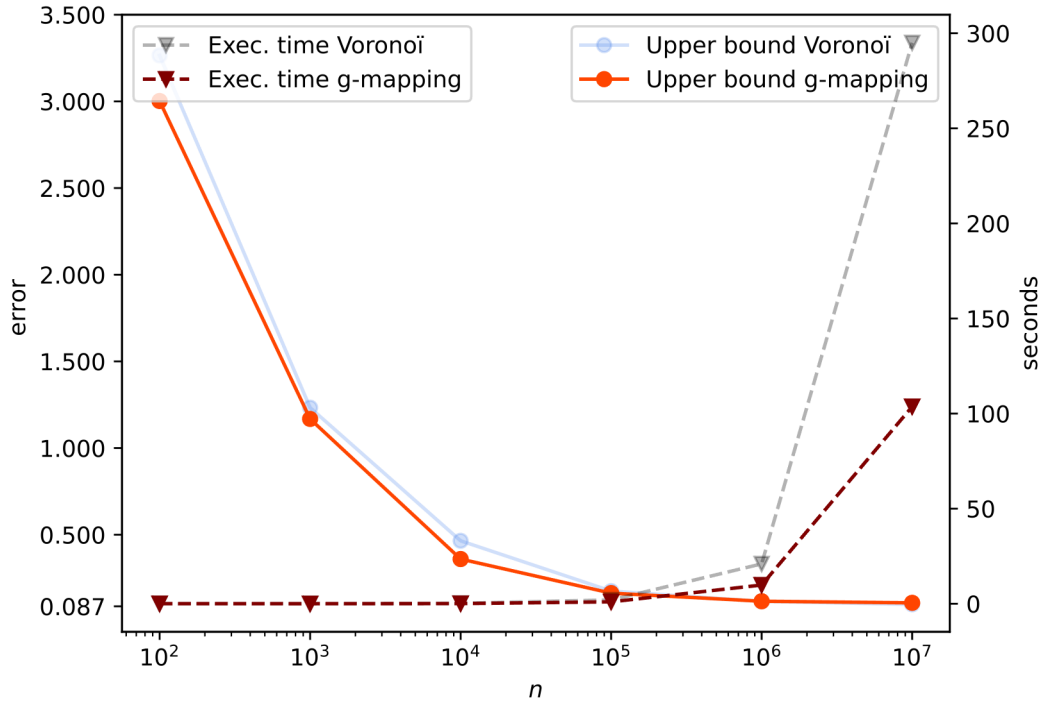
- Nearest neighbor algorithm
  - Many very efficient libraries (immensely cheaper than Voronoï diagram – complexity not exponential)
  - The bound is still valid with approximate nearest neighbors
- Evaluation of g
  - Very efficient on GPU

Beneficial side effect:

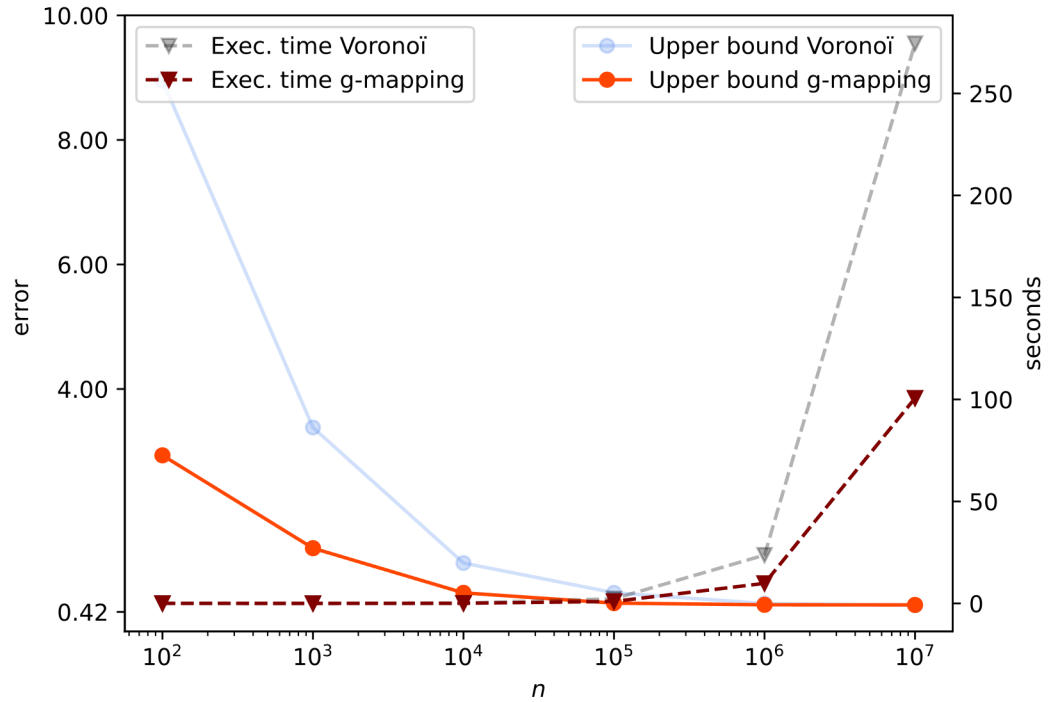
We were able to replace $K_g \|c_k - N(c_k)\|$ with $|g(c_k) - g(N(c_k))|$, which can make the bound tighter since by definition, $|g(c_k) - g(N(c_k))| \leq K_g \|c_k - N(c_k)\|$

Results on Toy functions

Upper bound of L_∞ error with computation time for Sinus function (left) and Holder table function (right).
The grid used is of size 1000×1000.



Best upper bound (Voronoï): 0.098
Best upper bound (grid mapping): 0.107
High sample estimation: 0.087



Best upper bound (Voronoï): 0.53
Best upper bound (grid mapping): 0.53
High sample estimation: 0.42

Results on Heat Diffusion

	Classical Voronoï	Mixed random/mesh	Grid mapping	Grid mapping
Nb points used	20×10^3	$\mathbf{512\times10^4}$	512×10^3	$\mathbf{512\times10^4}$
Total eval time (sec.)	> 3000	$\mathbf{1.72}$	$37 + 80$	$385 + 80$
Max L_1 error (est.)	0.1716	0.1716	0.1716	0.1716
Upper bound	84	1.6320	1.3014	$\mathbf{1.1953}$

Results of the different methods for computing $\bar{J}_g$ ($+80$ is the time for net predictions on the grid)

For Approx. Voronoï, we used a grid of size $p = 14$

- Computed nearest neighbors for 7,529,536 points
- Used faiss[1] library on GPU

1 https://github.com/facebookresearch/faiss

Takeaways - conclusions

We built algorithms to compute strict uniform upper bounds for $\|f - g\|_\infty$, where g is a Lipschitz neural net approximating for f. Can be very tight for low dimension.

- Voronoï based, **very costly** because of Voronoï diagram's **exponential complexity**.
- Can be made **way cheaper** by leveraging the **mesh structure** of some data dimensions.
- Can be **relaxed** by building a **grid** and bounding each center's error.

Perspectives:

- The method is applicable to **any K-lip model** like Gaussian Processes [8] or Polynomial interpolation.
- The algorithms make it possible **to locate the error**, which could be useful for **active learning** (we could provably reduce the error bound) or **sequential optimization**.
- Estimation of K_f:
  - build **local estimators** to refine the bound, possibly using interpolators [8].
  - Could we find K_f by using underlying **PDEs knowledge** [4]?
- Goes well with the **Neural Implicit Representation** approach. Could be paired with **neural operator** learning by low dimension parametrization of boundary conditions/initialization.
- **Hybridization** between ML and classical solvers

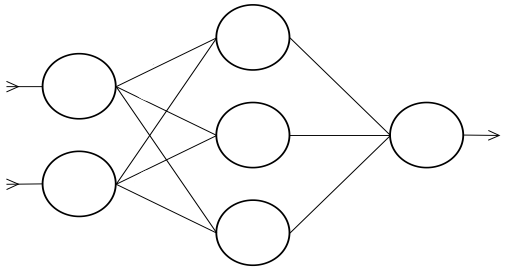
Check out "Accelerating hypersonic reentry simulations using deep learning-based hybridization (with guarantees)" Novello et al., freshly accepted in the Journal of Computational Physics!

References

1. Anil, Cem, James Lucas, and Roger Grosse. "Sorting out Lipschitz Function Approximation."ICML, June 11, 2019. https://doi.org/10.48550/arXiv.1811.05381.
2. Baker, Nathan, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, et al. "Workshop Report on Basic Research Needs for Scientific Machine Learning: Core Technologies for Artificial Intelligence," February 2019. https://doi.org/10.2172/1478744.
3. Béthune, Louis, Thibaut Boissin, Mathieu Serrurier, Franck Mamalet, Corentin Friedrich, and Alberto González-Sanz. "Pay Attention to Your Loss: Understanding Misconceptions about 1-Lipschitz Neural Networks."NeurIPS, October 17, 2022. https://doi.org/10.48550/arXiv.2104.05097.
4. Bunin, Gene A., and Grégory François. "Lipschitz Constants in Experimental Optimization." arXiv, January 14, 2017. https://doi.org/10.48550/arXiv.1603.07847.
5. Goswami, Somdatta, Aniruddha Bora, Yue Yu, and George Em Karniadakis. "Physics-Informed Deep Neural Operator Networks." arXiv, July 17, 2022. http://arxiv.org/abs/2207.05748.
6. Karniadakis, George, Yannis Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-Informed Machine Learning," May 24, 2021, 1–19. https://doi.org/10.1038/s42254-021-00314-5.
7. Kovachki, Nikola, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. "Neural Operator: Learning Maps Between Function Spaces." arXiv, April 7, 2023. http://arxiv.org/abs/2108.08481.
8. Lederer, Armin, Jonas Umlauft, and Sandra Hirche. "Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control."NeurIPS, December 19, 2019. http://arxiv.org/abs/1906.01376.
9. Li, Yichen, Peter Yichen Chen, Tao Du, and Wojciech Matusik. "Learning Preconditioners for Conjugate Gradient PDE Solvers." In *Proceedings of the 40th International Conference on Machine Learning*, 19425–39. PMLR, 2023. https://proceedings.mlr.press/v202/li23e.html.
10. Serrurier, Mathieu, Franck Mamalet, Thomas Fel, Louis Béthune, and Thibaut Boissin. "On the Explainable Properties of 1-Lipschitz Neural Networks: An Optimal Transport Perspective."NeurIPS, June 22, 2023. https://doi.org/10.48550/arXiv.2206.06854.
11. Serrurier, Mathieu, Franck Mamalet, Alberto Gonzalez-Sanz, Thibaut Boissin, Jean-Michel Loubes, and Eustasio del Barrio. "Achieving Robustness in Classification Using Optimal Transport with Hinge Regularization." In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 505–14. Nashville, TN, USA: IEEE, 2021. https://doi.org/10.1109/CVPR46437.2021.00057.
12. Wang, Ruigang, and Ian Manchester. "Direct Parameterization of Lipschitz-Bounded Deep Networks." In *Proceedings of the 40th International Conference on Machine Learning*, 36093–110. PMLR, 2023. https://proceedings.mlr.press/v202/wang23v.html.
13. Novello, Paul, Gaël Poëtte, David Lugato, Simon Peluchon, and Pietro Marco Congedo. "Accelerating Hypersonic Reentry Simulations Using Deep Learning-Based Hybridization (with Guarantees)." Journal of Computational Physics, September 30, 2022. https://doi.org/10.48550/arXiv.2209.13434.

Construction of 1-Lipschitz neural networks

Classical fully connected neural network:



Construction of 1-Lipschitz neural networks

Classical fully connected neural network:

$$\begin{cases} g(x) = g_l \circ g_{l-1} \circ \cdots \circ g_1(x) \\ g_k(x) = \sigma_k(W_k \cdot x + b_k) \end{cases}$$

with (for layer k):

- Activation function σ_k
- Weights matrix W_k
- Bias vector b_k

How to make it 1-Lipschitz ?

- Ensure that **each g_k is 1-Lipschitz**
  - ✓ Most activation functions are 1-Lipschitz
  - ✓ Bias is a simple shift
  - ✓ What about the weights ?

The naïve way:

During training, set $\boldsymbol{W_k \leftarrow \frac{W_k}{\|W_k\|}}$, where $\|W_k\|$ is the spectral norm of W_k.

Construction of 1-Lipschitz neural networks

DEEL

How to make it 1-Lipschitz ?

- Ensure that **each g_k is 1-Lipschitz**

The naïve way:

During training, set $\boldsymbol{W_k \leftarrow \frac{W_k}{\|W_k\|}}$, where $\|W_k\|$ is the spectral norm of W_k.

Problem: Eigenspaces of successive W_k may not be aligned:

- it might happen that $K_g \ll 1$

Construction of 1-Lipschitz neural networks

The orthogonal neural networks way:

During training, enforce orthonormality of each W_k [1].

- Implemented in **deel-torchlip**[1] using Bjork orthonormalization algorithm at each training iteration
- Use GroupSort [1] activation function, whose gradient is always 1
- In that case, $\boldsymbol{K_g = 1}$!!

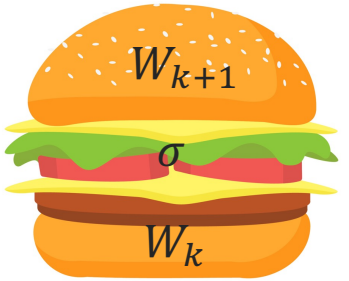
Problem: Enforcing orthonormality has an effect on the class of function g can approximate

- might hinder expressivity for regression tasks…
  - And orthonormalization is an iterative algorithm so prone to error if not converged
  - And it takes more time to train

1 https://github.com/deel-ai/deel-torchlip

Construction of 1-Lipschitz neural networks

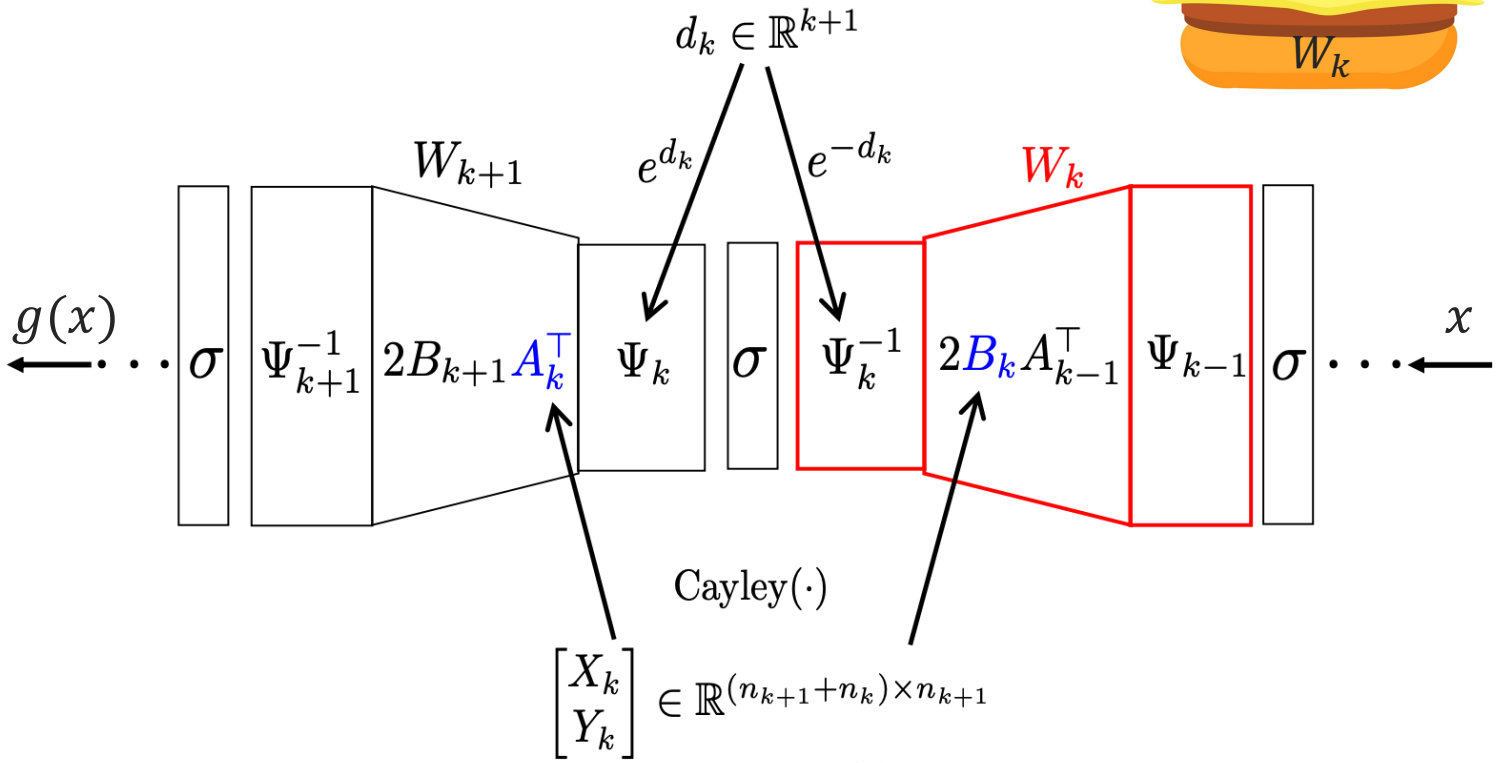
The « sandwich » layers way [12]:



Direct parametrization of W_k by trainable $\{X_k, Y_k, b_k, d_k\}$ such that the whole network g is K_g-Lipschitz.

- Each layer can be $> K_g$-Lipschitz, the whole network will still be K_g-Lipschitz.
- Very efficient, only involve matrix multiplication.
- The constraint is enforced by design (no approximation).

- … And each layer (kind of) looks like a sandwich.



A "sandwich" layer [12]

Construction of 1-Lipschitz neural networks

How to make it 1-Lipschitz ?

1. **The orthogonal neural networks way**
2. **The sandwich layers way**

How to make it K-Lipschitz ?

- Let each g_k be $\sqrt[l]{K}$-Lipschitz.
  Have to know in advance the desired value of K

- Let g_l be K-Lipschitz (by alleviating constraints on W_l)
  K can be learnt