A quasi-Monte Carlo data compression algorithm for machine learning

Michael Feischl

Quasi-Monte Carlo for PDEs

- More than 100 years of QMC

  [Weyl 1916], [Koksma 1942], [Hlawka 1962], [Sobol 1967], [Niederreiter 1988], [Faure 1982],...

$$-\mathrm{div}(A(\boldsymbol{y})\nabla u(x,\boldsymbol{y})) = f(x) \quad \text{compute} \quad \int_{[0,1]^s} G(u(\cdot,\boldsymbol{y}))\,d\boldsymbol{y}$$

- QMC for parametric equations
  - [Graham, Kuo, Nuyens, Scheichl, Sloan 2011]
  - [Dick, Kuo, Le Gia, Nuyens, Schwab 2015]
  - [Graham, Kuo, Nichols, Scheichl, Schwab, Sloan 2015]

- Higher-order QMC for random PDEs
  - [Dick, Kuo, Le Gia, Nuyens, Schwab 2014]

Improvement: $\mathcal{O}(N^{-1/2}) \longrightarrow \mathcal{O}(N^{-1+\delta}) \longrightarrow \mathcal{O}(N^{-\alpha})$

Neural Networks

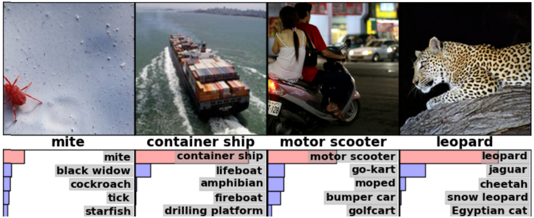
Parametrized set of functions $f(x, \mathcal{W})$

- $x \in \mathbb{R}^n \ldots$ input, $\mathcal{W} \in \mathbb{R}^N \ldots$ weights

Feedforward networks

$$f(x, \mathcal{W}) = W_d \phi(W_{d-1} \phi(\ldots \phi(W_0 x + b_0) \ldots)) + b_d$$

- Image classification (AlexNet, 2012)



- Speech recognition (transformers 2017, long-term-short-term networks 1997)

Machine learning in numerical analysis

Deep Ritz method [E-Yu, 2018]

- Solving the PDE

$$-\Delta u = f \quad \text{with b.c.} \quad u = 0$$

- equivalent to minimizing

$$\min_{u \in H_0^1(D)} \frac{1}{2} \|\nabla u\|_{L^2(D)}^2 - \langle f \, , \, u \rangle_{L^2(\Omega)}$$

Use NN to parametrize $u(x) = u(x, \mathcal{W})$, minimize [Fu-Wang 2021 $\rightarrow$ QMC]

$$\min_{\mathcal{W}} \frac{1}{N} \sum_{i=1}^{N} |\nabla u|^2(x_i) - 2f(x_i)u(x_i)$$

Machine learning in numerical analysis

Weak Adversarial Networks [Zang et al., 2019]

- Solving the PDE

$$\langle \nabla u\,,\,\nabla v\rangle_{L^2(D)} = \langle f\,,\,v\rangle_{L^2(D)} \quad \text{for all } v \in H^1_0(D)$$

- solve min-max problem

$$\min_{u\in H^1_0(D)} \max_{v\in H^1_0(D)} \left|\langle \nabla u\,,\,\nabla v\rangle_{L^2(D)} - \langle f\,,\,v\rangle_{L^2(D)}\right|^2$$

Use NN to parametrize $u(x) = u(x, \mathcal{W})$ and $v(x) = v(x, \mathcal{W}')$

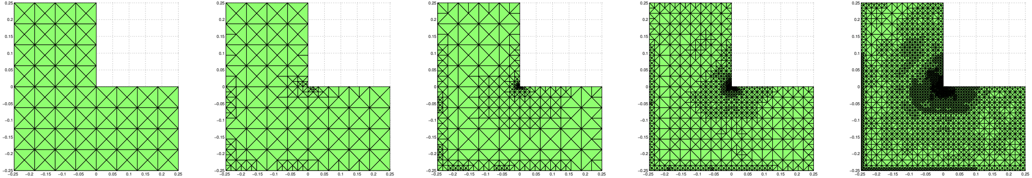
$$\min_{\mathcal{W}} \max_{\mathcal{W}'} \left|\frac{1}{N}\sum_{i=1}^{N} \nabla u(x_i)\nabla v(x_i) - f(x_i)v(x_i)\right|^2$$

Machine learning for adaptive mesh refinement

- Galerkin method: find $U_\ell \in \mathcal{X}_\ell$ s.t.

$$\langle \nabla U_\ell \,,\, \nabla V \rangle = \langle f \,,\, V \rangle \quad \text{for all } V \in \mathcal{X}_\ell$$

$$\boxed{\text{compute } U_\ell} \longrightarrow \boxed{\text{estimate } \rho_\ell} \longrightarrow \boxed{\text{mark}} \longrightarrow \boxed{\text{refine}}$$



Optimal adaptivity

Question: how to obtain optimal sequence $\mathcal{X}_\ell$

$$\|u - U_\ell\| \leq C(\dim \mathcal{X}_\ell)^{-s}$$

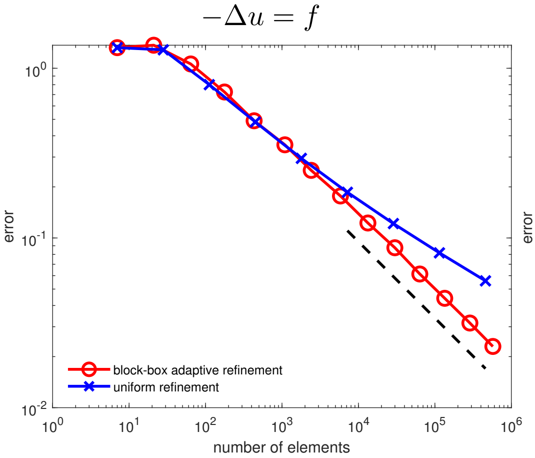
- Traditional adaptive algorithms are problem dependent
- Remedy: black-box algorithms with recurrent neural networks

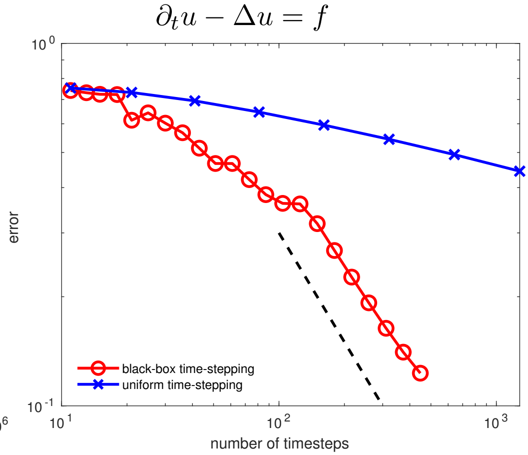
Theorem [Bohn-F. 2019]

The deep RNN $\mathrm{ADAPTIVE}$ is (almost) optimal with high probability, i.e.

$$\sup_{\substack{\ell\in\mathbb{N}\\ \|u-U_\ell\|\geq\varepsilon_{\mathrm{tol}}}} \|u - U_\ell\| N_\ell^s \leq C$$

Beats non-adaptive methods





Success at high cost

- Electronic Schrödinger equation [Gerard, Scherbela, Marquetand, Grohs 2021]
  - 50-100k epochs
- High dimensional Black-Scholes equation [Grohs, Hornung, Jentzen, Wurstemberger 2018]
- Weak GAN
  - 20k epochs for error $\approx 10^{-2}$
- AlexNet
  - 90 epchos with 1.2 million images

$\Longrightarrow$ Need for efficient training methods

Data fitting

- data $X = \{x_1, \dots, x_N\} \subset [0,1]^s$
- responses $Y = \{y_1, \dots, y_N\} \subset \mathbb{R}$

Aim: Find a predictor

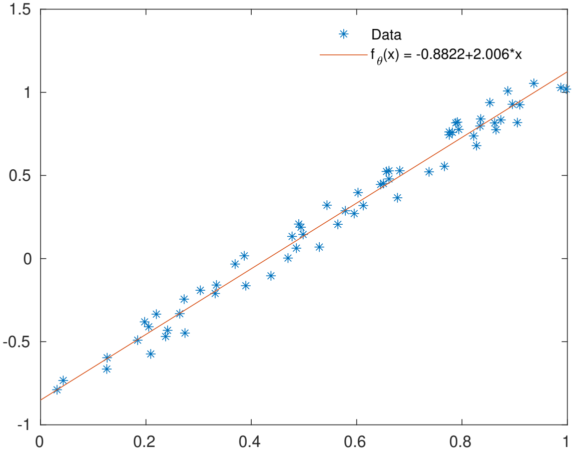
$$f_{\mathcal{W}} : [0,1]^s \to \mathbb{R}, \quad f_{\mathcal{W}}(x_i) \approx y_i, \quad i = 1, 2, \dots, N$$

Example:

$$f_{\mathcal{W}}(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_s x_s, \quad \mathcal{W} = (\beta_0, \beta_1, \dots, \beta_s);$$

Example: $f_{\mathcal{W}}$ is Neural Network

Data fitting



Training

Quality of predictor:

$$\begin{aligned}\mathrm{err}_{X,y}(f_{\mathcal{W}}) &= \frac{1}{N}\sum_{n=1}^{N}(f_{\mathcal{W}}(x_n) - y_n)^2 + R_{\mathcal{W}} \\ &= \frac{1}{N}\sum_{n=1}^{N} f_{\mathcal{W}}^2(x_n) - \frac{2}{N}\sum_{n=1}^{N} y_n f_{\mathcal{W}}(x_n) + \frac{1}{N}\sum_{n=1}^{N} y_n^2 + R_{\mathcal{W}}\end{aligned}$$

- choose the parameters $\mathcal{W}$ such that $\mathrm{err}_{X,y}(f_{\mathcal{W}})$ is 'small'.
- many evaluations of $\mathrm{err}_{X,y}(f_{\mathcal{W}})$ might be required
- Lasso (least squares with ℓ_1-constraint), gradient descent

Cost reduction techniques

Gradient descent requires to compute

$$\frac{1}{N}\sum_{n=1}^{N}\nabla_{\mathcal{W}}(f_{\mathcal{W}}(x_n)-y_n)^2$$

- Batch gradient descent: compute random sample of $M \ll N$ terms of sum

$$\frac{1}{N}\sum_{n=1}^{N}\nabla_{\mathcal{W}}(f_{\mathcal{W}}(x_n)-y_n)^2 \approx \frac{1}{M}\sum_{k=1}^{M}\nabla_{\mathcal{W}}(f_{\mathcal{W}}(x_{n_k})-y_{n_k})^2$$

- Support points [Mak-Joseph 2018]: target distribution Y

$$\arg\min_{z_1,\ldots,z_M}\Big\{\frac{2}{M}\sum_{i=1}^{M}\mathbb{E}\|z_i-Y\| - \frac{1}{M^2}\sum_{i,j=1}^{M}\|z_i-z_j\|\Big\}$$

- Sketching algorithms, see e.g. [Ahfock, Astle 2017]

Idea

Find an approximation of $\mathrm{err}_{X,y}(f_{\mathcal{W}})$ which can be computed in less than N operations.

We need to approximate two terms:

$$\frac{1}{N}\sum_{n=1}^{N} f_{\mathcal{W}}^2(x_n)$$

and

$$\frac{1}{N}\sum_{n=1}^{N} y_n f_{\mathcal{W}}(x_n).$$

Idea

Let $P = \{z_1, \dots, z_M\} \subset [0,1]^s$ be a set of points with $M \ll N$

- calculate weights $(W_{X,P}(z_m))_{1 \leq m \leq M}$ such that

$$\frac{1}{N} \sum_{n=1}^{N} f_{\mathcal{W}}^2(x_n) \approx \sum_{m=1}^{M} f_{\mathcal{W}}^2(z_m) W_{X,P}(z_m)$$

- weights $(W_{X,y,P}(z_m))_{1 \leq m \leq M}$ such that

$$\frac{1}{N} \sum_{n=1}^{N} y_n f_{\mathcal{W}}(x_n) \approx \sum_{m=1}^{M} f_{\mathcal{W}}(z_m) W_{X,y,P}(z_m)$$

Decoupling training from data

We then find 'optimal' parameters $\mathcal{W}$ by minimizing

$$\sum_{m=1}^{M} f_{\mathcal{W}}^2(z_m) W_{X,P}(z_m) - 2 \sum_{m=1}^{M} f_{\mathcal{W}}(z_m) W_{X,y,P}(z_m) + \frac{1}{N} \sum_{n=1}^{N} y_n^2 + R_{\mathcal{W}}$$

- Computing the weights $W_{X,P}(z_m)$ and $W_{X,y,P}(z_m)$ depends on N and M, but not on $\mathcal{W}$.
- Finding 'optimal' $\mathcal{W}$ depends on M and the number of optimization steps, but not on N.

$$\#\text{opt.steps} + \#\text{data} \ll \#\text{opt.steps} \times \#\text{data}$$

Digital Nets

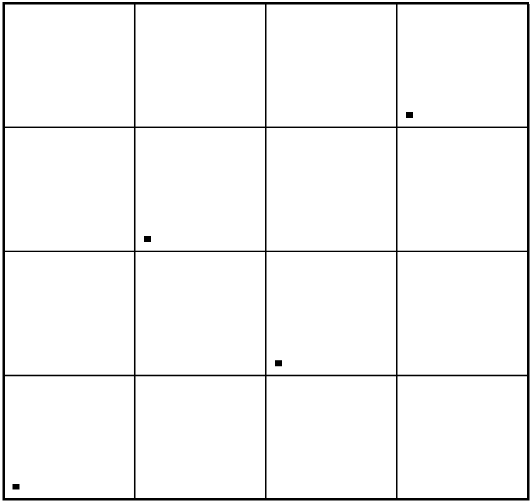
$P = \{z_1, \ldots, z_M\}$ is a (t, m, s) net in base b $(M = b^m)$

- Elementary interval in base b:

$$I_{\boldsymbol{a},\boldsymbol{d},m} = \prod_{j=1}^{s} \left[\frac{a_j}{b^{d_j}}, \frac{a_j + 1}{b^{d_j}}\right), d_1 + \cdots + d_s = m - t, 0 \leq a_j < b^{d_j}$$

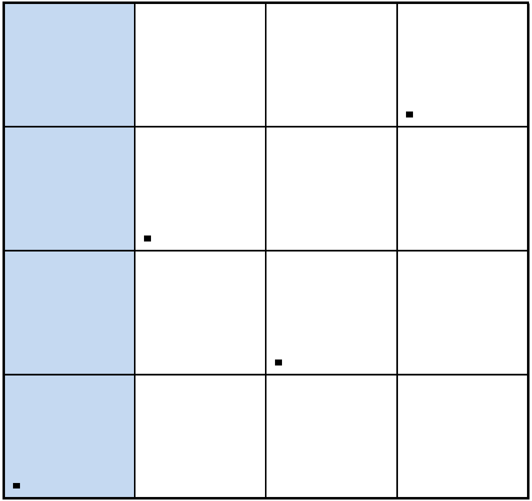
- For fixed $\boldsymbol{d} = (d_1, \ldots, d_s)^\top$ we get a partition of $[0,1)^s$ as $\boldsymbol{a} = (a_1, \ldots, a_s)^\top$ runs through all admissible choices.
- $\mathrm{Vol}(I_{\boldsymbol{a},\boldsymbol{d}}) = b^{t-m}$
- P is a (t, m, s)-net if each elementary interval contains exactly b^t points.

Digital net



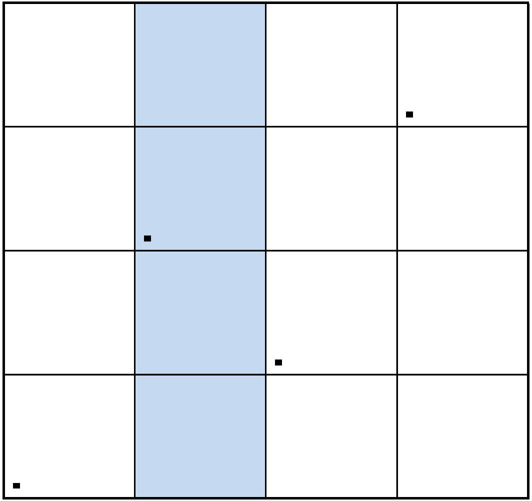
The shaded region contains exactly one point.

Digital net



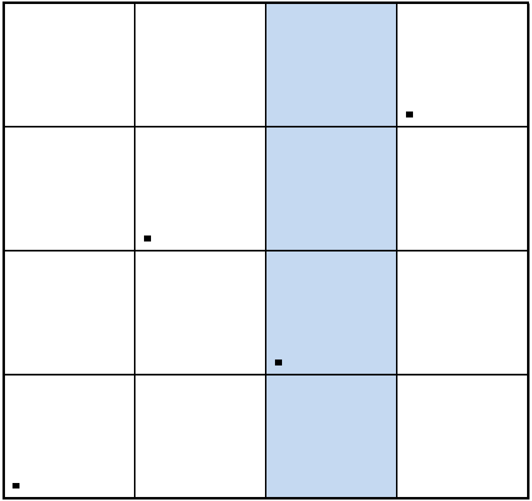
The shaded region contains exactly one point.

Digital net



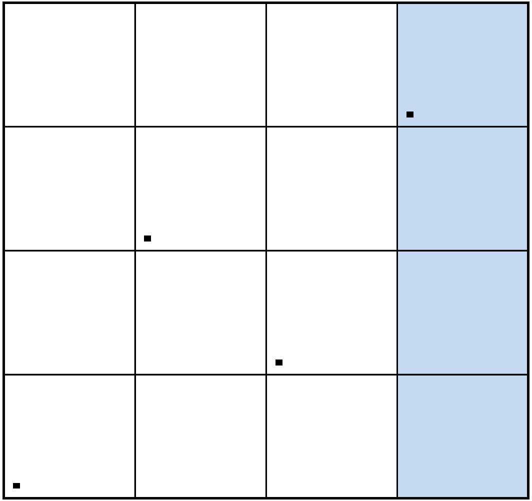
The shaded region contains exactly one point.

Digital net



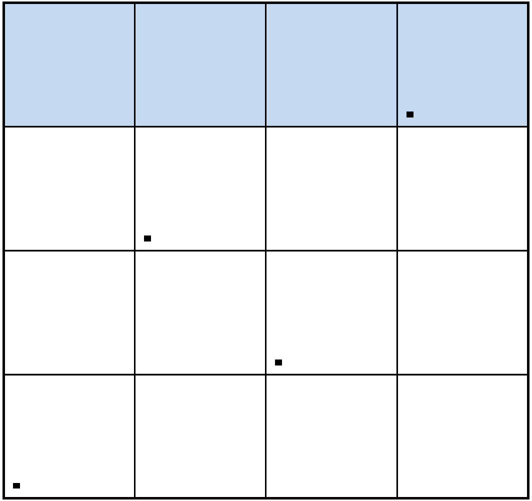
The shaded region contains exactly one point.

Digital net



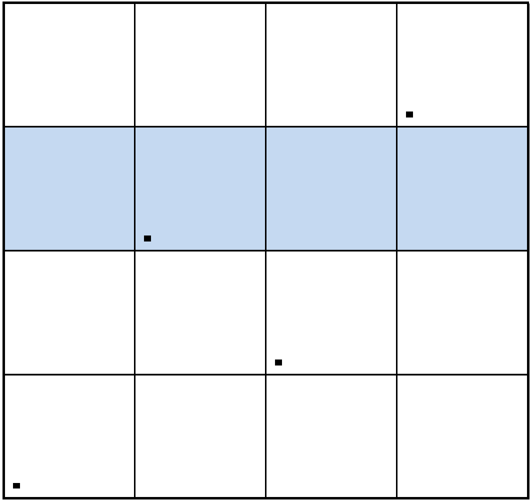
The shaded region contains exactly one point.

Digital net



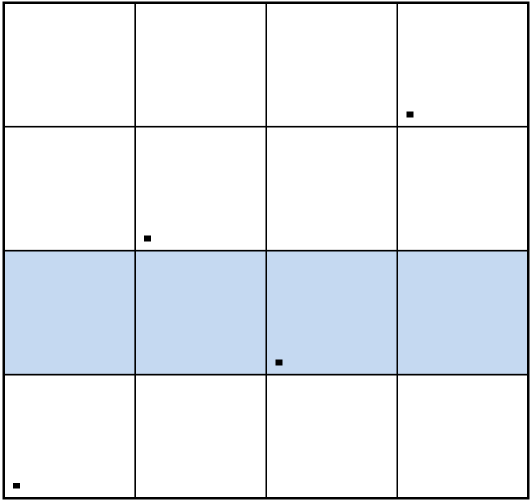
The shaded region contains exactly one point.

Digital net



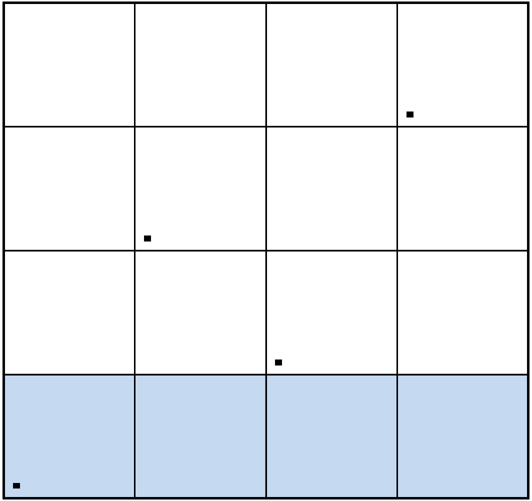
The shaded region contains exactly one point.

Digital net



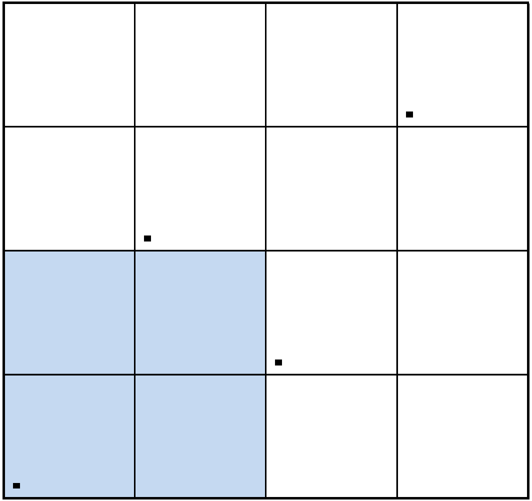
The shaded region contains exactly one point.

Digital net



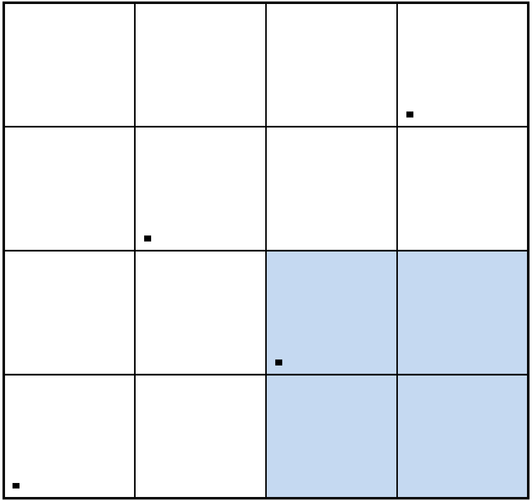
The shaded region contains exactly one point.

Digital net



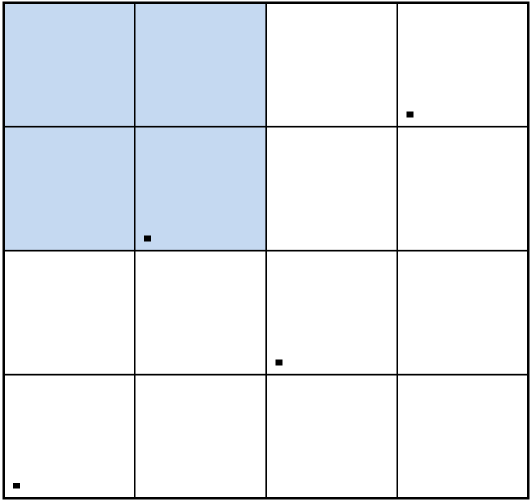
The shaded region contains exactly one point.

Digital net



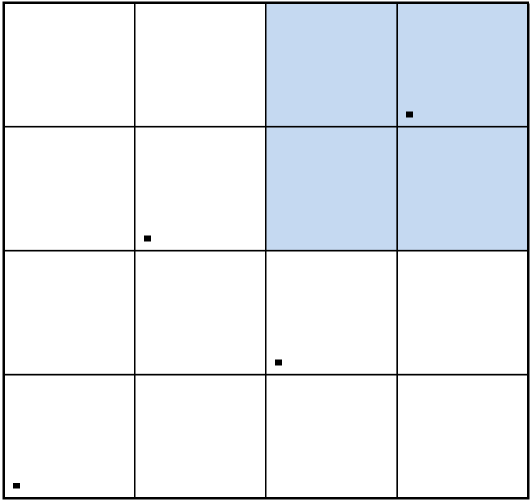
The shaded region contains exactly one point.

Digital net



The shaded region contains exactly one point.

Digital net



The shaded region contains exactly one point.

Digital net constructions

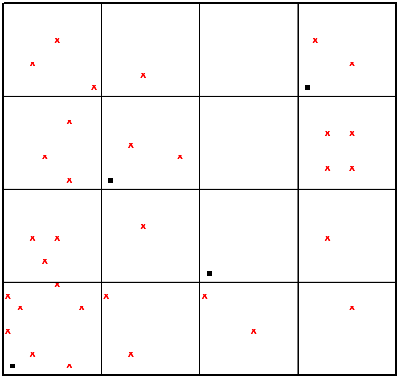
- Choose generator matrices $C_1, \dots, C_s$
- define i-th coordinate of digital net point z_m by

$$z_{m,i} = c_1 b^{-1} + c_2 b^{-2} + \dots \quad \text{where} \quad \boldsymbol{c} = C^i \boldsymbol{m} \text{ in } \mathbb{F}_b$$

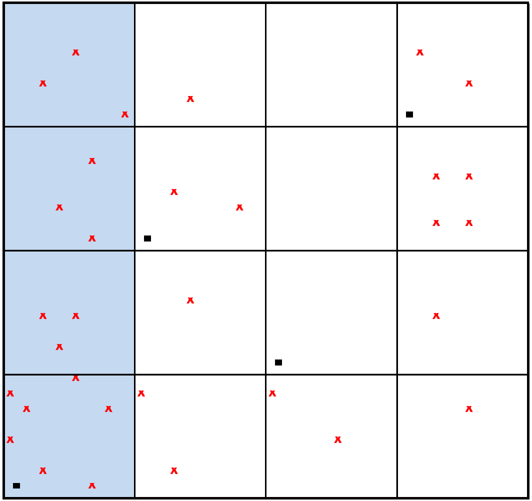
  with $m = m_0 + m_1 b + m_2 b^2 + \dots$
- Sobol' sequence (1967)
- Niederreiter sequences (1988) based on polynomials in $\mathbb{F}_b$
- t-value optimized Sobol' sequences [Joe-Kuo 2008]

How should we choose the weights $W_{X,P}(z_m)$?

Given: Digital net P (black) and data points X (red).

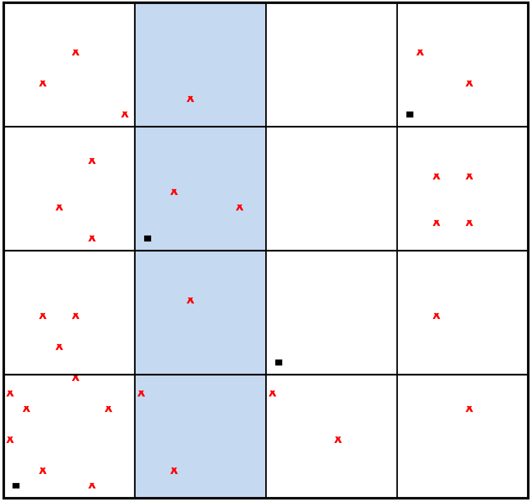


How should we choose the weights $W_{X,P}(z_m)$?



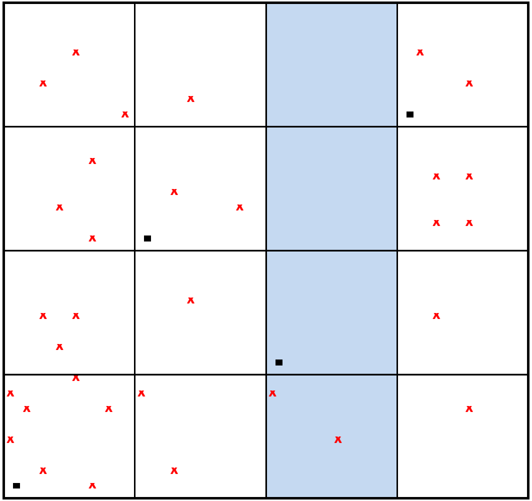
The shaded region contains 16 out of 32 data points

How should we choose the weights $W_{X,P}(z_m)$?



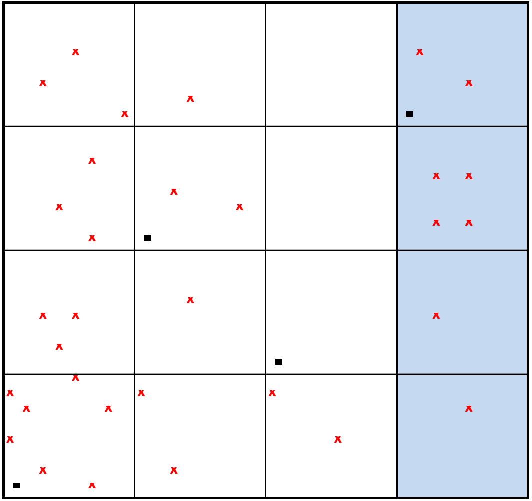
The shaded region contains 6 out of 32 data points

How should we choose the weights $W_{X,P}(z_m)$?



The shaded region contains 2 out of 32 data points

How should we choose the weights $W_{X,P}(z_m)$?



The shaded region contains 8 out of 32 data points

How should we choose the weights $W_{X,P}(z_m)$?

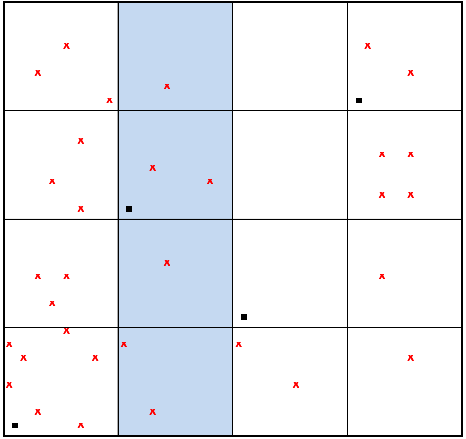
So the weights based on this partition should be

$$
\begin{aligned}
W_{X,P}(z_1) &= \frac{16}{32} = \frac{1}{2}, \\
W_{X,P}(z_2) &= \frac{6}{32} = \frac{3}{16}, \\
W_{X,P}(z_3) &= \frac{2}{32} = \frac{1}{16}, \\
W_{X,P}(z_4) &= \frac{8}{32} = \frac{1}{4};
\end{aligned}
$$

$\Rightarrow$ We can also use all the other possible partition into elementary intervals simulatenously.

Average data density

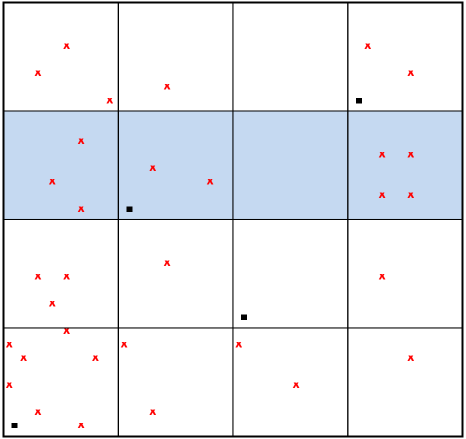
"How many data points can be reached from z_m with elementary interval of volume $2^{-\ell}$"



The shaded region contains 6 out of 32 data points

Average data density

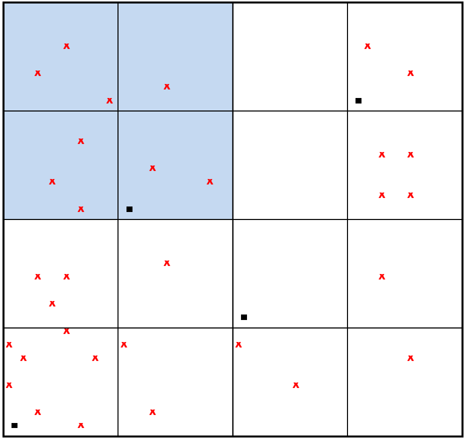
"How many data points can be reached from z_m with elementary interval of volume $2^{-\ell}$"



The shaded region contains 9 out of 32 data points

Average data density

“How many data points can be reached from z_m with elementary interval of volume $2^{-\ell}$”



The shaded region contains 9 out of 32 data points

Final result

The formula for $W_{X,P}(z_m)$ using all possible partitions is

$$\sum_{q=0}^{s-1}(-1)^q\binom{s-1}{q}\frac{b^{-(m'-m)-q}}{N}\sum_{\substack{\boldsymbol{d}\in\mathbb{N}_0^s\\|\boldsymbol{d}|=m-q}}\sum_{\substack{\boldsymbol{a}\in K_{\boldsymbol{d}}\\z_m\in I_{\boldsymbol{a},\boldsymbol{d}}}}\#(\mathcal{X}\cap I_{\boldsymbol{a},\boldsymbol{d},m}).$$

We use the approximation

$$\frac{1}{N}\sum_{n=1}^{N}f(x_n)^2\approx M_{m,m'}(f,\mathcal{X})=\sum_{\ell=1}^{b^{m'}}f(z_\ell)^2W_{\mathcal{X},P}(z_\ell).$$

Efficient computation

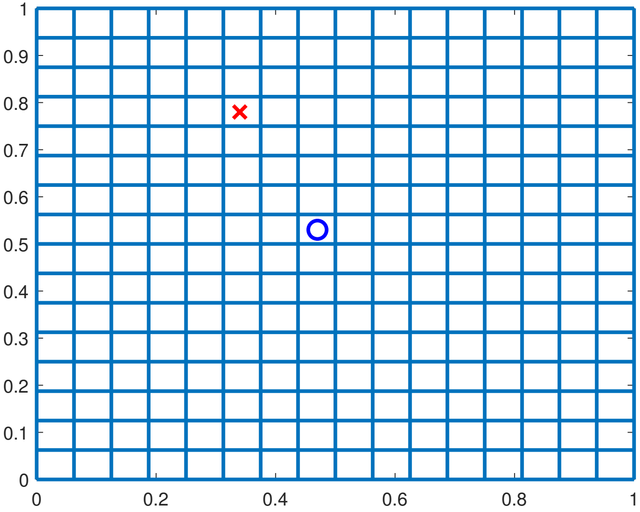
Main challenge is to compute

$$S_\ell(z_m) := \sum_{\substack{\boldsymbol{d}\in\mathbb{N}_0^s \\ |\boldsymbol{d}|=\ell}} \sum_{\substack{\boldsymbol{a}\in K_{\boldsymbol{d}} \\ z_m\in I_{\boldsymbol{a},\boldsymbol{d}}}} \#(\mathcal{X}\cap I_{\boldsymbol{a},\boldsymbol{d}})$$

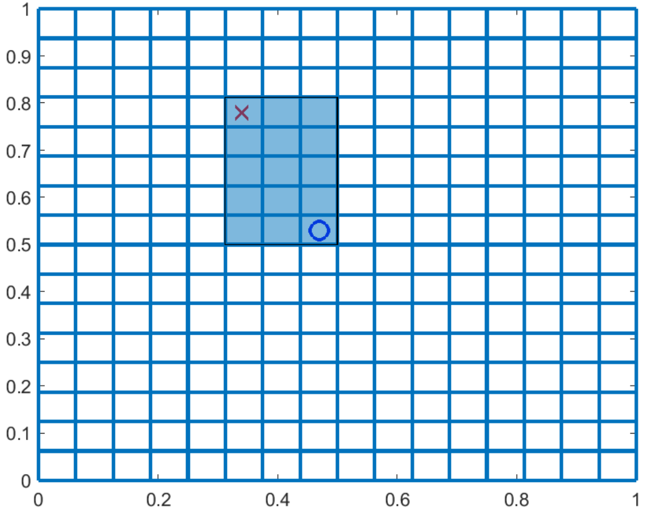
- $b^\ell\binom{\ell+s-1}{s-1}$ elementary intervals
- direct computation intractable

$\Longrightarrow$ Use properties of dyadic intervals

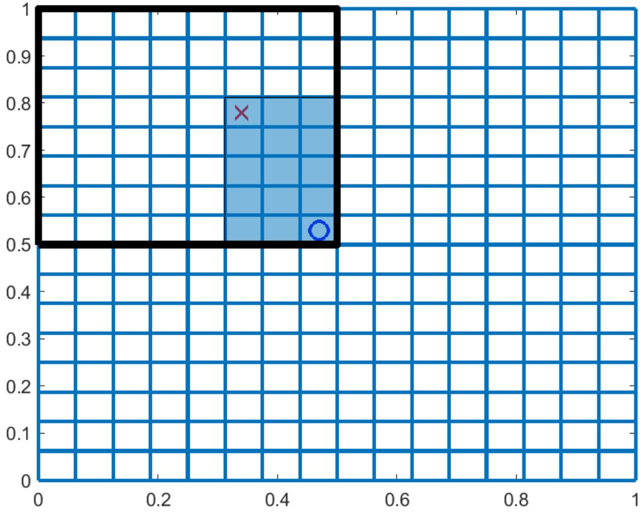
Efficient computation



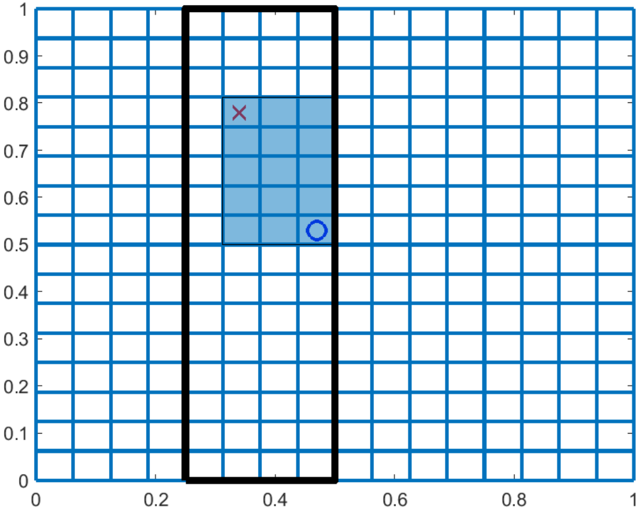
Efficient computation



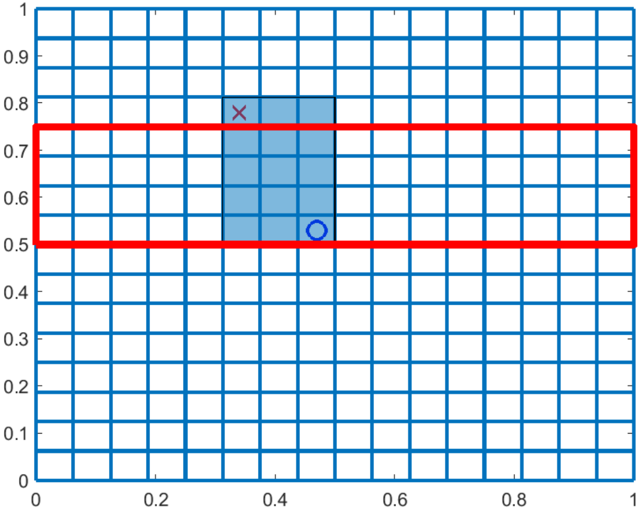
Efficient computation



Efficient computation



Efficient computation



Efficient computation

Given data point x and net point z_m

- for each dimension $j = 1, \ldots, s$
  - find largest $i_j \in \mathbb{N}$ with

$$ab^{-i_j} \leq x_j, z_{m,j} < (a+1)b^{-i_j} \quad \text{for some } a \in \mathbb{N}$$

- use equivalence

$$\#\big\{I_{\boldsymbol{a},\boldsymbol{d}} \,:\, z_m, x \in I_{\boldsymbol{a},\boldsymbol{d}}, |I_{\boldsymbol{a},\boldsymbol{d}}| = b^{-\ell}\big\} = \#\big\{\boldsymbol{d} \in \mathbb{N}^s \,:\, |\boldsymbol{d}| = \ell, \boldsymbol{d} \leq \boldsymbol{i}\big\}$$

- sum over all data points $x \in \mathcal{X}$

Efficient computation

Remains to compute

$$N(\ell,s,\boldsymbol{i}) = \#\{\boldsymbol{d}\in\mathbb{N}^s \,:\, |\boldsymbol{d}|=\ell, \boldsymbol{d}\le\boldsymbol{i}\}$$

Use recursion

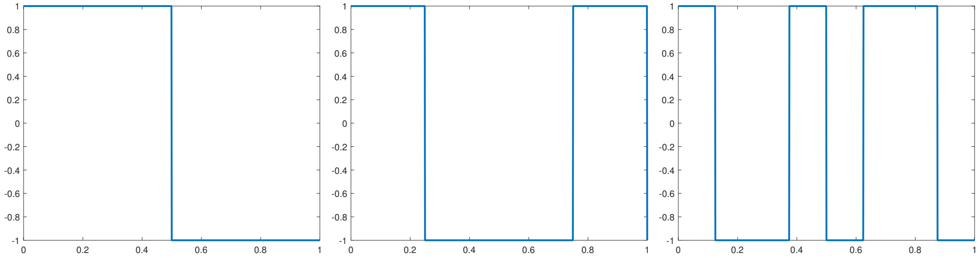
$$N(\ell,s,\boldsymbol{i}) = \sum_{k=\ell-i_s}^{\ell} N(k,s-1,\boldsymbol{i})$$

Theorem [Dick-F. 2021]

- Startup cost of computing $M_{m,m'}(f,\mathcal{X})$ is $\mathcal{O}(sm'b^{m'}N)$
- Recomputing $M_{m,m'}(f,\mathcal{X})$ for the same data points $\mathcal{X}$ but different f costs $\mathcal{O}(sb^{m'})$

Error estimation

Walsh functions $w_k(x) := \exp(2\pi/b(k_0x_1 + \ldots + k_{j-1}x_j)$



Multidimensional analog

$$w_{\boldsymbol{k}}(x) := \prod_{j=1}^{s} w_{k_s}(x_s)$$

- $L^2([0,1]^s)$-orthogonal basis
- good to approximate smooth functions [Dick 2013]

Error estimation

Goal: approximate

$$\frac{1}{N}\sum_{n=1}^{N} c_n f(x_n) \approx \frac{1}{N}\sum_{n=1}^{N} c_n f_K(x_n) = \int_{[0,1]^s} f_K \phi_K \, dx = \int_{[0,1]^s} f \phi_K \, dx$$

- Walsh series approximation

$$f(x) \approx f_K(x) := \sum_{\boldsymbol{k}\in K} f_{\boldsymbol{k}} w_{\boldsymbol{k}}(x)$$

- approximate data density

$$\phi_K(x) := \sum_{\boldsymbol{k}\in K} \frac{1}{N}\sum_{n=1}^{N} c_n w_{\boldsymbol{k}}(x_n)\overline{w_{\boldsymbol{k}}}(x)$$

Error estimation

Goal: approximate

$$\frac{1}{N}\sum_{n=1}^{N} c_n f(x_n) \approx \frac{1}{N}\sum_{n=1}^{N} c_n f_K(x_n) = \int_{[0,1]^s} f_K \phi_K \, dx = \int_{[0,1]^s} f \phi_K \, dx$$

- use digital net to approximate integral

$$\int_{[0,1]^s} f \phi_K \, dx \approx b^{-m} \sum_{k=1}^{b^m} f(z_k)\phi_K(z_k)$$

- choose $K := \bigcup_{|\boldsymbol{d}|=m'} \{\boldsymbol{a} \,:\, \boldsymbol{a} \leq b^{\boldsymbol{d}}\}$ for

$$b^{-m} \sum_{k=1}^{b^m} f(z_k)\phi_K(z_k) = M_{m,m'}(f, \mathcal{X})$$

Error estimation

$$\frac{1}{N}\sum_{n=1}^{N} c_n f(x_n) \approx \int_{[0,1]^s} f\phi_K\,dx \approx b^{-m}\sum_{k=1}^{b^m} f(z_k)\phi_K(z_k)$$

Two approximation errors

- Walsh series approximation error

$$\|f - f_K\|_{L^\infty([0,1]^s)} \lesssim m'^q b^{-m'}\|f\|$$

- quadrature error for order α digital nets

$$\Big|\int_{[0,1]^s} f\phi_K\,dx - b^{-m}\sum_{k=1}^{b^m} f(z_k)\phi_K(z_k)\Big| \lesssim m^q b^{-\alpha(m-m')}\|f\|$$

Main result [Dick-F. 2021]

New algorithm to approximate large sums with QMC

$$\frac{1}{N}\sum_{n=1}^{N} c_n f(x_n) \approx M_{m,m'}(f,\mathcal{X})$$

- optimal choice $m' \simeq m/(1+1/\alpha)$ leads to

$$\left|\frac{1}{N}\sum_{n=1}^{N} c_n f(x_n) - M_{m,m'}(f,\mathcal{X})\right| \lesssim b^{-\alpha m/(1+\alpha)}$$

- with startup cost

$$\mathcal{O}(N b^{m/(1+\alpha)})$$

- and online cost

$$\mathcal{O}(b^m)$$

Remarks on main result

If data is QMC pointset $\implies \phi_K = 1 \implies \text{error} \lesssim b^{-\alpha m}$ [Longo, Mishra, Rusch, Schwab, 2021]

All error estimates depend on $\|f\|$

$$\partial_{\mathcal{W}_1,\dots,\mathcal{W}_k} f$$

- can be calculated explicitely for feed-forward networks
- use holomorphic activation function

$$\phi(x) = \tanh(x), \frac{x}{1+e^{-x}}, \frac{e^x}{1+e^x}$$

- can be controlled in terms of $\|\mathcal{W}\|$ [Longo, Mishra, Rusch, Schwab, 2021]

Finding good digital nets

most challenging part

$$\int_{[0,1]^s} f \phi_K \, dx$$

- f is smooth
- ϕ_K is piecewise constant Walsh expansion of order m'

$\implies$ (t, m, s)-net with $m - t \geq m'$ will integrate ϕ_K exactly

- if $m - t < m'$, approximation is bad
- strict limits on t-value of nets [Niederreiter-Xing]

$$t \gtrsim s - \log(s)$$

Weighted spaces?

Weighted spaces solve the problem for smooth functions

- definition of weighted spaces [Sloan-Woźniakowski 1998]

$$-\mathrm{div}(A(\boldsymbol{y})\nabla u) = f$$

Anisotropic expansion

$$A(x, \boldsymbol{y}) := \phi_0(x) + \sum_{i=1}^{\infty} y_i \phi_i(x)$$

leads to anisotropic derivatives

$$\|\partial_{y_{i_1} \ldots y_{i_k}} u\| \lesssim \Gamma_k \prod_{j=1}^{k} \gamma_{i_j}$$

$\implies$ t-value independent computation of $\mathbb{E}(G(u))$ in high-dimensions

Weighted (t, m, s)-nets?

- Digital nets with product and/or order dependent t-values

$$\Pi_{i_1,\ldots,i_k} P \quad \text{is } (t(i_1,\ldots,i_k), m, s)\text{-net}$$

- If ϕ_K of order m' represents data (images, . . .)

$$\int_{[0,1]^s} f\phi_K \, dx \approx \frac{1}{N} \sum_{i=1}^{b^m} f(z_i)\phi_K(z_i)$$

with $m \simeq m'$ "independent" of t-values

Weighted (t, m, s)-nets?

- digital nets are created via generator matrices

$$z_{m,j} = C_j \boldsymbol{m} \quad \text{in base } b$$

- t-value is direct consequence of rank properties of C_j

$\Longrightarrow$ direct search is infeasible (or is it? [L. Paulin's talk])

- many digital net constructions have explicit formulas for t-values
  - Sobol' sets: $t - \mathrm{val}(\Pi_{i_1,\ldots,i_k} P) = \sum_{j=1}^{k} \deg(p_{i_j})$
  - Similar formulas for Niederreiter-Xing sets

Not enough for what we want: given weighted t-values, construct pointset P

First order approximation property

Walsh series expansion with $K := \bigcup_{|\boldsymbol{d}|=m'} \{\boldsymbol{a} : \boldsymbol{a} \leq b^{\boldsymbol{d}}\}$ leads to

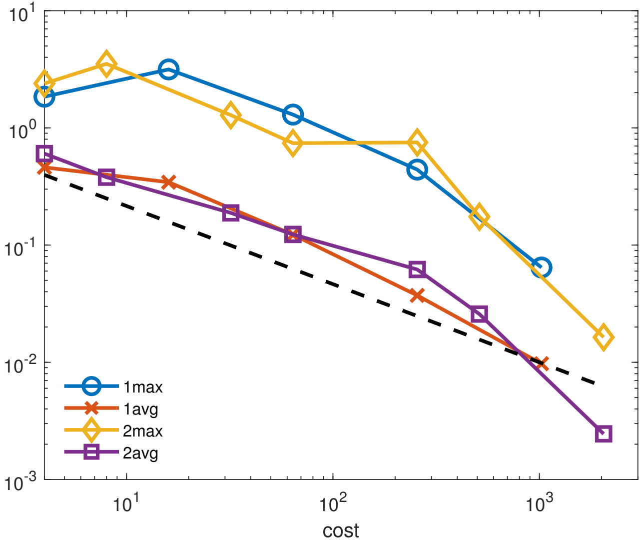
$$\|f - f_K\|_{L^\infty([0,1]^s)} \lesssim m'^q b^{-m'} \|f\|$$

- Can we improve that? $\mathcal{O}(b^{-\alpha m'})$
- Obviously: need larger $K \implies$ better digital nets which integrate ϕ_K exactly

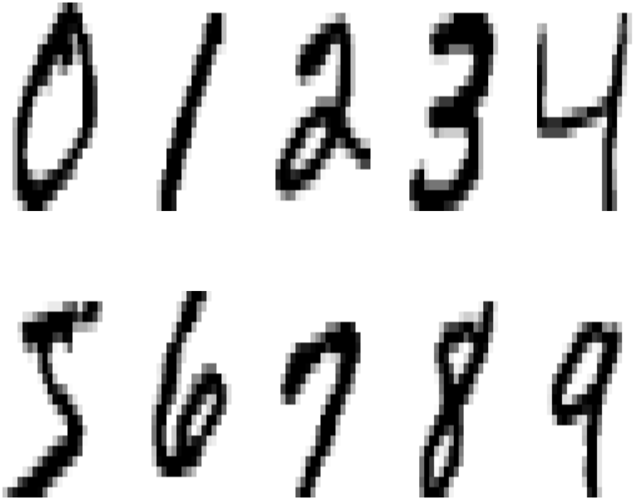
Similar results for "multiple rank-1 lattice rules"

- higher-order approximation with fourier series expansion
- [Kämmerer, Volkmer 2019], [Kämmerer 2016]

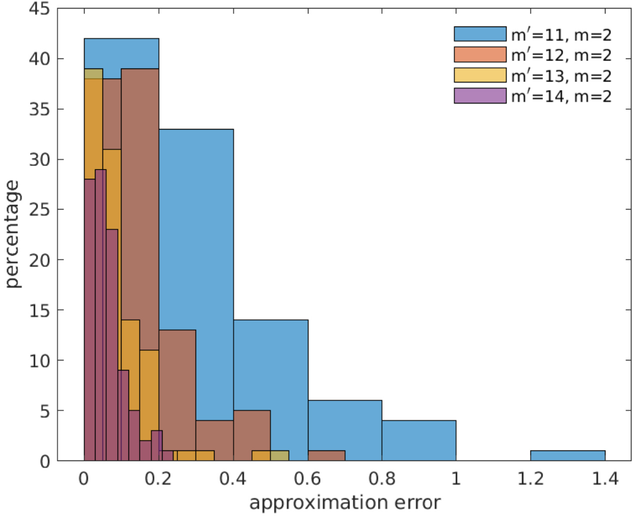
Numerical example: linear regression (6-dim)



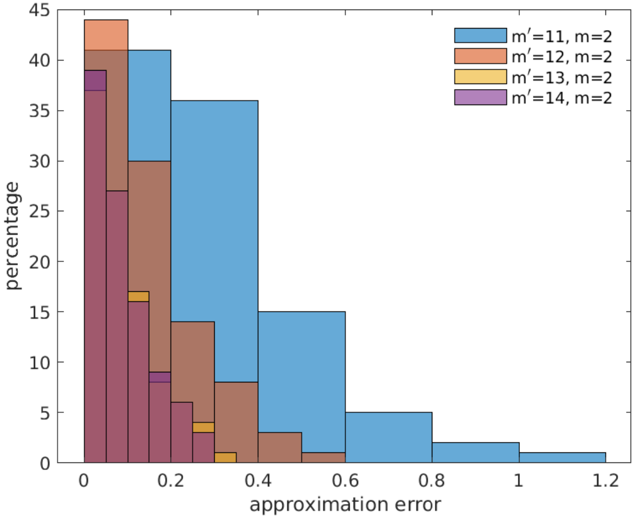
Numerical example: MNIST

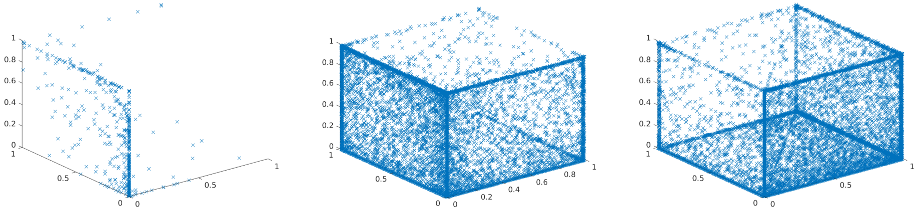


Numerical example: NN



Numerical example: DNN




1
0.8
0.6
0.4
0.2
0
1
0.5
0
0
0.5
1
1
0.8
0.6
0.4
0.2
0
1
0.5
0
0
0.2
0.4
0.6
0.8
1
1
0.8
0.6
0.4
0.2
0
1
0.5
0
0
0.5
1

Thanks for Listening!

Michael Feischl

TU Wien
Institute for Analysis and Scientific Computing
michaelfeischl.net

TECHNISCHE
UNIVERSITÄT
WIEN