Enhancement of artificial neural networks based on H^2-matrices

Daria Sushnikova

Coauthors: Pavel Kharyuk, Ivan Oseledets

Skoltech, Moscow

May 16, 2019

Weighted sum

Sources $x = \{x_i\}_{i=1\ldots N}$, $x_i \in \mathbb{R}^d$, receivers $y = \{y_i\}_{j=1\ldots N}$, $y_j \in \mathbb{R}^d$, $d = 1, 2, 3$.

$$y_i = \sum_{j=1}^{N} a_{ij} x_j, \tag{1}$$

- N-body problem;
- Discretization of Integral equation (IE).

$$y = Ax.$$

Computation - $\mathcal{O}(N^2)$ operations. Storage - $\mathcal{O}(N^2)$ items of memory.

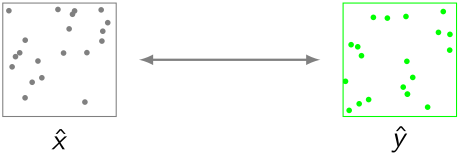
Our goal: reduce the computation and storage complexity to $\mathcal{O}(N)$

Separation property

If sets $\hat{x} \in \mathbb{R}^P$, $\hat{y} \in \mathbb{R}^K$ – **spatially separated**:

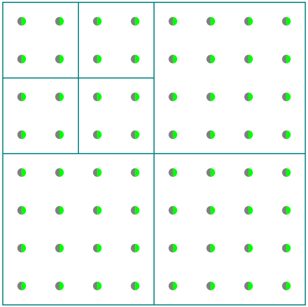
$$\hat{y} = \widehat{A}\hat{x}, \quad \widehat{A} \in \mathbb{R}^{K \times P}$$

has approximate **low rank**.

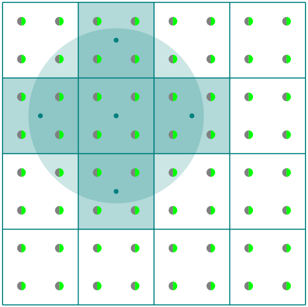


- Works both for N-body problem;
- Works for IEs with smooth kernels

Hierarchical grid



(a) Hierarchical division of the grid.



(b) Close and far blocks.

Close and far blocks

$$y = Ax = Cx + F_L x.$$

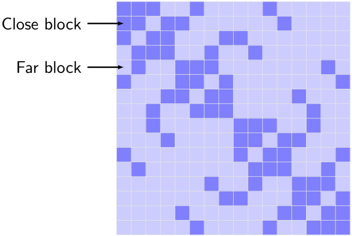


Figure 2: Matrix A

Bottom-level compression

$$F_{ij} \approx \widetilde{U}_i \widetilde{F}_{ij} \widetilde{V}_j, \quad \forall i, j \in 1, \ldots, M_L$$

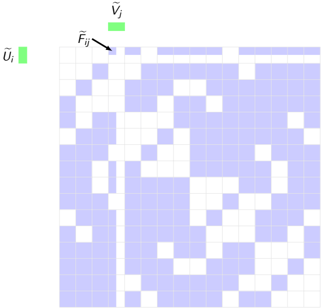
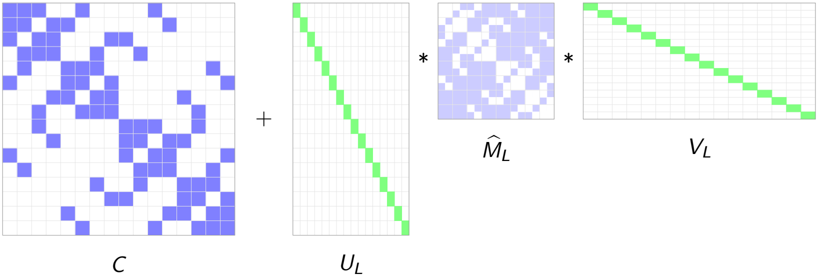


Figure 3: Matrix F_L

Bottom-level compression

$$y = Cx + U_L \widehat{M}_L V_L x,$$



Interaction list

Far block is in **interaction list** on level l if the block that contains it on level $l-1$ is a close block.

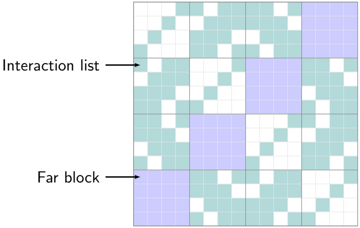


Figure 4: Matrix $\widehat{M}_L$

$$y = Cx + U_L M_L V_L x + U_L F_{L-1} V_L x.$$

Level compression

Repeat the compression procedure for matrix F_{L-1}.

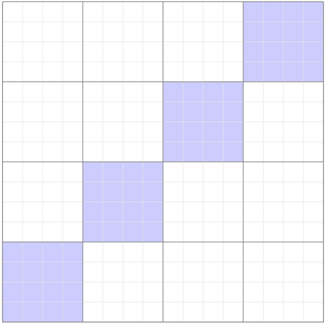


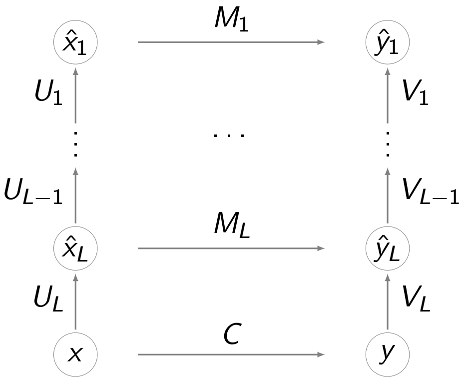
Figure 5: Matrix F_{L-1}

Repeat the compression till the level $l = 1$. Obtain:

$$y = Cx + \sum_{l=L}^{1} U_l M_l V_l x.$$

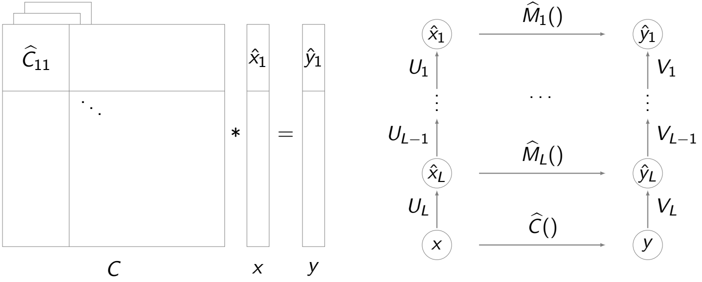
$\mathcal{H}^2$ matrix by vector product

$$y = Cx + \sum_{l=L}^{1} U_l M_l V_l x.$$



NN architecture based on $\mathcal{H}^2$ matrix

- x, y - samples;
- C, U_l, M_l, V_l, $l \in 1 \dots L$ – unknowns;
- Duplicate each block in C, M_l, $l \in 1 \dots L$
- Add nonlinearity



(a) $\hat{y}_1 = g(\widehat{C}_{11} * \widehat{C}_{11} * \widehat{C}_{11} * \hat{x}_1)$

(b) $\mathcal{H}^2$ -NN

Alternative idea of $\mathcal{H}^2$ to NN transformation: [Fan Y. et al. A multiscale neural network based on hierarchical nested bases //Research in the Mathematical Sciences. 2019. . 6. . 2. . 21.]

Building $\mathcal{H}^2$ -nn architecture for given x, y

1. If grid for x and y is known:
   - Define separation criterion
   - Define rank $r < B$
   - Repeat $\mathcal{H}^2$ building procedure, writing block sizes instead of blocks.
2. If grid for x and y is unknown: assume tensor grid of the same dimension and go to 1;

Motivation

Useful if:

- Direct solution of problem $F(x) = y$ is resource intensive;
- Operator F is unknown;
- Operator F is nonlinear;

Experiments

Radiative transfer equation (RTE)

PDE:

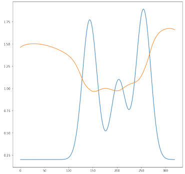
$$v\nabla_x\varphi(x,v) + \mu_t(x,v) = \mu_s(x)u(x) + f(x), \quad \text{in } \Omega \times \mathcal{S}^{d-1}, \Omega \in \mathcal{R}^d,$$
$$\varphi(x,v) = 0, \quad \text{on } \{(x,v) \in \partial\Omega \times \mathcal{S}^{d-1} : n(x)\dot{v} < 0\},$$
$$u(x) = \frac{1}{4\pi}\int_{\mathcal{S}^{d-1}} \varphi(x,v)dv$$

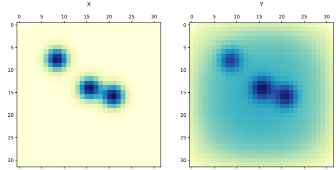
IE:

$$u(x) = \mathcal{I}(\mu_s(x))$$

Mapping:

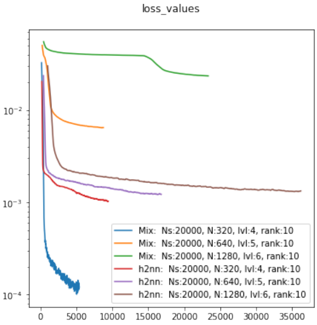
$$\mu_s(x) \rightarrow u(x)$$





Experiments

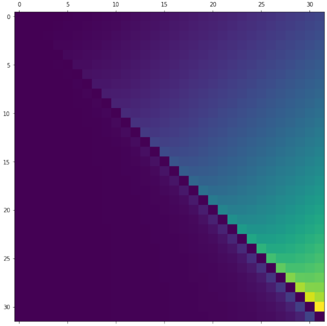
Radiative transfer equation (RTE)

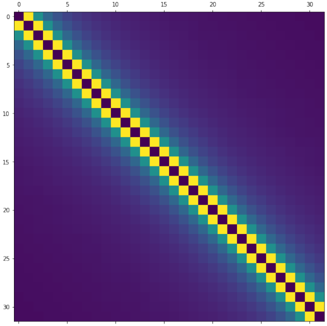


N	h2nn	MNN-$\mathcal{H}^2$ -Mix
320	Train: 0.02350, Test: 0.02346	Train: 0.01197, Test: 0.01196
640	Train: 0.02676, Test: 0.02687	Train: 0.06179, Test: 0.06381
1280	Train: 0.03240, Test: 0.03233	Train: 0.12397, Test: 0.12510778

Counter example

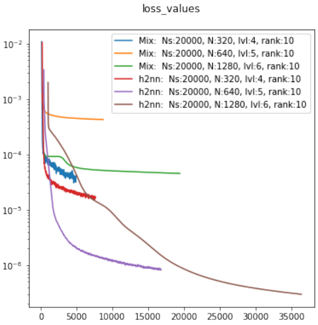
$$a_{ij} = \frac{1}{\|x_i - y_j\|} + f(i,j)$$





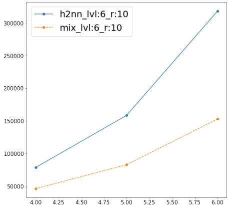
Experiments

Counter example

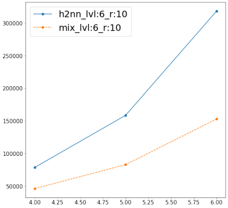


N	h2nn	MNN-$\mathcal{H}^2$ -Mix
320	Train:0.00790, Test: 0.00789	Train: 0.00857, Test: 0.00858
640	Train: 0.00424, Test: 0.00425	Train: 0.14413, Test: 0.14592
1280	Train: 0.01464, Test: 0.00425	Train: 0.19069, Test: 0.19070

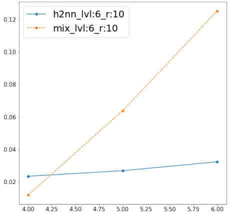
Experiments



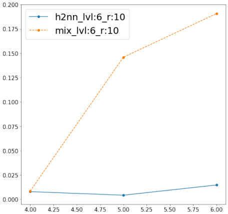
(a) RTE memory comparison



(b) CE memory comparison



(c) RTE error comparison



(d) CE error comparison