Learning from Linear Algebra: A Graph Neural Network Approach to Preconditioner Design for Conjugate Gradient Solvers

Vladislav Trifonov [1], Alexander Rudikov [2,1], Oleg Iliev [3], Yuri M. Laevsky [4], Ivan Oseledets [2,1], Ekaterina Muravleva [1]

[1]Skoltech [2]AIRI [3]Fraunhofer ITWM [4]ICMMG SB RAS

Skoltech

Problem statement

We consider large sparse systems with symmetric and positive definite (SPD) matrices arising from the discretisation of PDEs:

$$Ax = b,\ A \in \mathbb{R}^{n\times n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^n,\ \mathrm{nnz}(A) \ll n^2\,. \tag{1}$$

The standard method to solve such systems is conjugate gradient (CG) method. Convergence rate of CG is determined by $\sqrt{\kappa(A)}$, where $\kappa(A) = |\lambda_{\max}|/|\lambda_{\min}|$. Practitioners need to rely on effective preconditioners P for faster convergence of CG.

In this work we make us of the duality between sparse matrices and graphs to obtain vertices and edges:

$$Ax = b \to \mathcal{G} = (\mathcal{V}, \mathcal{E}),\ \text{where } a_{i,j} = e_{i,j} \in \mathcal{E}, b_i = v_i \in \mathcal{V}\,. \tag{2}$$

Main idea is to learn such a preconditioner $P(\theta)$ that can outperform classical preconditioner P in terms of effect on spectrum:

$$\kappa(P(\theta)^{-1}A) \ll \kappa(P^{-1}A) \ll \kappa(A)\,. \tag{3}$$

Previous work

Previous work [3] used a message-passing GNN to construct preconditioner in the form of Cholesky decomposition $P(\theta) = L(\theta)L(\theta)^T$, where $L(\theta) = \mathrm{GNN}(\theta, A, b)$.

Pros:

- Preserved sparsity pattern;
- Controlled properties;
- Shallow size and single inference.

Cons:

- Worse than classical P;
- Unstable learning;
- Loss overflow for large systems.

Proposed approach

We propose a PreCorrector. Our architecture fed with L from the IC decomposition to GNN and trained to predict a correction for this decomposition:

$$L(\theta) = L + \alpha \cdot \mathrm{GNN}(\theta, L, b), \tag{4}$$

where α is a learned parameter coefficient.

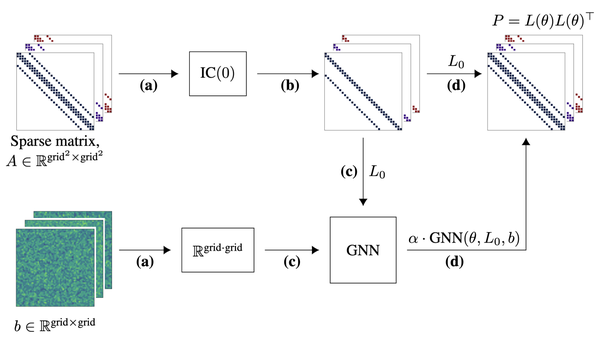

Figure 1. PreCorrector scheme that takes IC(0) as input.

Choice of loss function

A natural choice is $\|P - A\|_F^2$ [1]. It is not optimal since its tends to minimize high frequency components first. One can tackle low frequency components with loss $\|(P-A)A^{-1}\|_F^2$. We can rewrite latter loss with Hutchinson's estimator [2]:

$$\begin{aligned}\|(P-A)A^{-1}\|_F^2 &= \|PA^{-1} - I\|_F^2 = \mathrm{Tr}\left((PA^{-1} - I)^\top (PA^{-1} - I)\right) \\ &= \mathbb{E}_\varepsilon\left[\varepsilon^\top (PA^{-1} - I)^\top (PA^{-1} - I)\varepsilon\right] = \mathbb{E}_\varepsilon \|(PA^{-1} - I)\varepsilon\|_2^2,\quad \varepsilon \sim \mathcal{N}(0,1).\end{aligned} \tag{5}$$

Assuming we have a dataset of linear systems $A_i x_i = b_i\ \varepsilon = b_i,\ P = L(\theta)L(\theta)^\top$ and $A_i^{-1}b_i = x_i$, we obtain:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \|L(\theta)L(\theta)^\top x_i - b_i\|_2^2 \tag{6}$$

Loss (6) appeared earlier in [3]. Now we provide an understanding that this loss is a good choice because it mitigates low frequency components.

Loss	10^{-3}	10^{-6}	10^{-9}	$\kappa(P^{-1}A)^1$	$\lambda^1_{\min}$	$\lambda^1_{\max}$
$\|P-A\|_F^2$	123 ± 4.7	155 ± 5.3	182 ± 5.4	571	0.0033	**1.88**
$\|(P-A)A^{-1}\|_F^2$	**42** ± 1.7	**55** ± 2.1	**67** ± 2.5	**60**	**0.1406**	8.40

Table 1. Comparison of loss functions. Number of CG iterations on the diffusion equation with 0.7 on grid 64×64. During training Hutchinson trick is applied for both losses. [1]Condition number and eigenvalues are calculated on a single sampled linear system.

Matrix	$\kappa(P^{-1}A)$	$\lambda_{\min}$	$\lambda_{\max}$	$\|LL^\top A^{-1} - I\|_F^2$	Bound $\lambda_{\min}$	Bound $\lambda_{\max}$
A	87565	17.5506	1536813.4	–	–	–
$(L_0L_0^\top)^{-1}A$	749	0.0016	1.2	$1.04 \cdot 10^6$	$9.7 \cdot 10^{-4}$	230
$(L(\theta)L(\theta)^\top)^{-1}A$	78	0.1981	15.5	$1.73 \cdot 10^3$	$8.4 \cdot 10^{-3}$	4157

Table 2. Condition number, spectrum and value of different losses for a sampled model from diffusion equation with variance 0.7 on grid 128×128. The loss value is calculated directly with A^{-1}.

Dataset

We test PreCorrector on SPD matrices obtained by discretization of elliptic equation:

$$\begin{aligned}-\nabla \cdot \left(k(x)\nabla u(x)\right) &= f(x),\ \text{in } \Omega \\ u(x)\big|_{x\in\partial\Omega} &= 0\end{aligned}\,. \tag{7}$$

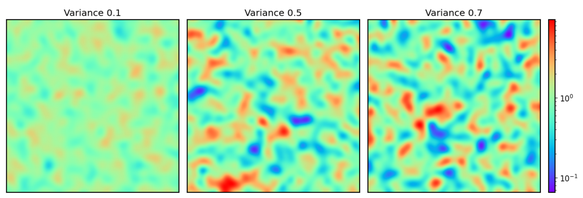

Figure 2. Coefficient function $k(x) = \exp(\phi(x))$ for grid 128×128 with different variances.

Resutls

Same algorithmic complexity

Grid	Method	Pre-time	Time (iters) to 10^{-3}	Time (iters) to 10^{-6}	Time (iters) to 10^{-9}
128×128	IC(0)	$5.1 \cdot 10^{-4}$	1.071 ± 0.189 (156 ± 5.6)	1.338 ± 0.229 (196 ± 5.5)	1.554 ± 0.261 (228 ± 5.7)
	PreCor[IC(0)]	$2.0 \cdot 10^{-3}$	**0.571** ± 0.078 (67 ± 3.2)	**0.720** ± 0.079 (85 ± 3.5)	**0.859** ± 0.079 (102 ± 3.9)
	ICt(1)	$3.8 \cdot 10^{-3}$	0.720 ± 0.098 (95 ± 3.5)	0.902 ± 0.122 (119 ± 3.4)	1.048 ± 0.141 (139 ± 3.5)
	PreCor[ICt(1)]	$5.4 \cdot 10^{-3}$	**0.470** ± 0.071 (50 ± 2.5)	**0.593** ± 0.078 (64 ± 2.9)	**0.708** ± 0.086 (77 ± 3.1)

More complex baseline

Grid	Method	Pre-time	Time (iters) to 10^{-3}	Time (iters) to 10^{-6}	Time (iters) to 10^{-9}
64×64	ICt(5)	$2.1 \cdot 10^{-3}$	**0.069** ± 0.003 (22 ± 0.8)	**0.087** ± 0.003 (28 ± 0.8)	**0.103** ± 0.003 (34 ± 0.8)
	PreCor[ICt(1)]	$2.3 \cdot 10^{-3}$	0.091 ± 0.070 (29 ± 1.4)	0.115 ± 0.071 (38 ± 1.7)	0.138 ± 0.072 (47 ± 1.9)
128×128	ICt(5)	$8.9 \cdot 10^{-3}$	0.570 ± 0.071 (48 ± 1.7)	0.705 ± 0.088 (60 ± 1.6)	0.833 ± 0.104 (70 ± 1.8)
	PreCor[ICt(1)]	$5.4 \cdot 10^{-3}$	**0.470** ± 0.071 (50 ± 2.5)	**0.593** ± 0.078 (64 ± 2.9)	**0.708** ± 0.086 (77 ± 3.1)

Previous work

Grid	Method	Pre-time	Time (iters) to 10^{-3}	Time (iters) to 10^{-6}	Time (iters) to 10^{-9}
32×32	IC(0)	$7.2 \cdot 10^{-5}$	$\underline{0.043} \pm 0.001$ (32 ± 0.6)	$\underline{0.055} \pm 0.002$ (42 ± 0.7)	$\underline{0.066} \pm 0.002$ (50 ± 0.6)
	PreCor[IC(0)]	$1.6 \cdot 10^{-3}$	**0.040** ± 0.092 (21 ± 0.4)	**0.050** ± 0.092 (28 ± 0.4)	**0.060** ± 0.092 (35 ± 0.4)
	GNN [3]	$2.5 \cdot 10^{-3}$	0.088 ± 0.070 (60 ± 5.9)	0.113 ± 0.071 (79 ± 7.6)	0.135 ± 0.071 (96 ± 9.1)
64×64	IC(0)	$1.6 \cdot 10^{-4}$	0.145 ± 0.010 (65 ± 1.4)	0.185 ± 0.012 (84 ± 1.0)	0.218 ± 0.014 (99 ± 0.9)
	PreCor[IC(0)]	$2.0 \cdot 10^{-3}$	**0.099** ± 0.087 (33 ± 0.6)	**0.126** ± 0.088 (43 ± 0.6)	**0.151** ± 0.089 (53 ± 0.6)
	GNN [3]	$2.5 \cdot 10^{-3}$	nan[1]	nan[1]	nan[1]

References

[1] Paul Häusner, Ozan Öktem, and Jens Sjölund. Neural incomplete factorization: learning preconditioners for the conjugate gradient method. *arXiv preprint arXiv:2305.16368*, 2023.

[2] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[3] Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient pde solvers. In *International Conference on Machine Learning*, pages 19425–19439. PMLR, 2023.

vladislav.trifonov@skoltech.ru