Learning incomplete factorization preconditioner for GMRES

Paul Häusner*, Aleix Nieto Juscafresa*, Jens Sjölund

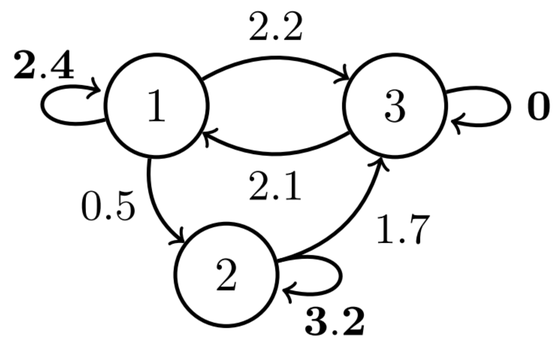
Department of Information Technology, Uppsala University, Sweden

NLDL 2025

Overview LearnedLU preconditioner

We accelerate the GMRES algorithm using graph neural networks for solving large-scale linear equation systems $Ax = b$

$$\begin{bmatrix} \mathbf{2.4} & 0 & 2.2 \\ 0.5 & \mathbf{3.2} & 0 \\ 2.1 & 1.7 & \mathbf{0} \end{bmatrix}$$



- Utilize the **connection of graphs and sparse matrices** to construct a GNN architecture
- Train the neural network to predict a **sparse factorization** of the matrix A which is used as a preconditioner for the GMRES method
- Analysis of different loss functions to train the preconditioner
- **Fast to compute and problem specific preconditioner**

Learned LU preconditioner

- **Replace hand-engineered preconditioners for the GMRES algorithm with outputs produced by a graph neural network**
- Two design requirements for preconditioning matrix P:
  - Non-singularity
  - Sparsity to limit resource requirements
- **Learn the sparse LU factorization of P instead**
- Mapping the matrix A to L and U is parameterzied by a graph neural network
- The training objective is to predict an incomplete factorization of the matrix A subject to **sparsity constraints**:

$$\begin{aligned} \min_\theta \quad & d(A, L_\theta U_\theta) \\ \text{s.t.} \quad & (L_\theta)_{ij} = 0 \quad \text{if} \quad A_{ij} = 0 \\ & (U_\theta)_{ji} = 0 \quad \text{if} \quad A_{ji} = 0 \end{aligned}$$

- The choice of distance function d influences the results
- Factorized P can be easily inverted using forward-backward substitution

Graph neural network architecture

- The problem matrix A is interpreted as the adjacency matrix of the graph (Coates graph representation)
- Sparsity constraints controlled via the edges used for message passing
  - Adding additional edges allows more non-zero elements in the preconditioner
  - This allows a better approximation of the inverse of A
  - But more computational resources are required to train the model
- Positional encoding via the edge features ot break permutation equivariance
- Ensuring invertability of preconditioner $P = LU$
  - The matrix L is constructed to have unit diagonal
  - Activation function for diagonal elements of matrix U
  - Continous approximation of the activation function during training

Background: GMRES

- GMRES is a iterative method for linear equation systems
- Method of choice for large-scale and sparse problems (e.g. PDE discretizations)
- Convergence depends on the spectral properties (singular values) of the matrix (and the right-hand side b)
- Clustered singular values are often better for convergence
- Faster convergence is obtained by solving a preconditioned system:

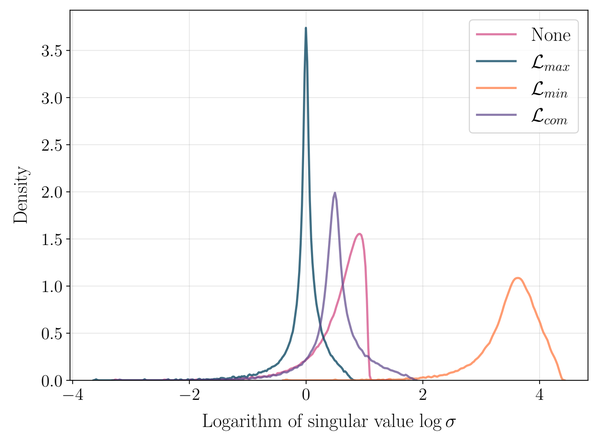
$$AP^{-1}y = b$$

  where $P^{-1} \approx A^{-1}$ is the preconditioner
- Trade-off between time required to compute the preconditioner P^{-1} and resulting speedup
- Extreme cases: $P^{-1} = A^{-1}$ (direct method) and $P^{-1} = I$ (no speedup)
- Typical preconditioners are often hand-engineered and domain specific: e.g. Jacobi, incomplete LU, multigrid methods

Loss functions

We analyze different loss functions and show their connections to singular values of the preconditioned system

- $\mathcal{L}_{\text{max}} = \|A - P\|_F$: Minimize large singular values
- $\mathcal{L}_{\text{min}} = \|PA^{-1} - I\|_F$: Maximize small singular values
- $\mathcal{L}_{\text{com}}$: Linear combination of both loss functions



Efficient loss approximation

Loss functions allow efficient approximation via Hutchinson's trace estimator

$$\|M\|_F^2 \approx \|Mz\|_2^2 \qquad z_i \quad \text{i.i.d. Gaussian distributed}$$

requires only matrix-vector products

Results

Testing on synthetic PDE problems from the finite element method:

	Method	$\sigma_{\text{min}} \uparrow$	$\sigma_{\text{max}} \downarrow$	$\kappa \downarrow$	$\|\boldsymbol{P} - \boldsymbol{A}\|_F \downarrow$	$\|\boldsymbol{PA}^{-1} - \boldsymbol{I}\|_F \downarrow$	Time ↓	Iterations ↓
Precond.	No preconditioner	0.0014	20.70	31 152.94	255.26	1 577.65	30.85	1 153
	Jacobi	0.0003	5.16	31 166.13	205.83	6 319.45	30.12	1 152
	ILU(0)	0.0006	30.32	120 740.90	138.31	3 688.45	**3.33**	**413**
	Learned IC	0.0006	7.58	27 405.57	143.23	3 719.47	12.84	692
Loss	$\mathcal{L}_{\text{max}}$: Equation (5)	0.0007	**5.00**	**16 139.46**	**88.76**	3 261.23	3.67	437
	$\mathcal{L}_{\text{min}}$: Equation (6)	**1.1375**	20 318.88	37 030.72	287.62	**50.05**	24.41	1 054
	$\hat{\mathcal{L}}_{\text{min}}$: Equation (7)	-	-	-	287.71	50.30	130.01	2 192
	$\mathcal{L}_{\text{com}}$: Equation (8)	0.0017	52.92	66 691.71	197.82	1 240.60	3.42	418

Summary & Conclusion

- Combination of machine learning and classical optimization algorithms
- Graph neural networks are natural computational backends for linear algebra and learned optimization
- Future research can integrate both learned and classical preconditioners

Link to paper

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

paul.hausner@it.uu.se, aleix.nieto-juscafresa@it.uu.se, jens.sjolund@it.uu.se