Transformers Learn Prior and Regularization for In-context Learning of Inverse Linear Regression

Fei Lu

Johns Hopkins University

Joint work with Yue Yu, Lehigh University

MSML2025, University of Naples
August 8, 2025

DMS 2238486

Outline

1. In-context learning: forward and inverse problems
2. A Transformer for ICL-ILR: nonlocal attention operator (NAO)
3. NAO learns the prior and regularization
4. Conclusions and outlook

1. In-context learning (ICL):

$$\begin{pmatrix} x_1 & x_2 & \dots & x_n & x_{n+1} \\ y_1 & y_2 & \dots & y_n & ? \end{pmatrix}$$

Context

$(x_{1:n}, y_{1:n})$

$$\hat{y}_{n+1} = f_\theta(x_{n+1} \mid x_{1:n}, y_{1:n})$$

Forward prediction

Training data: $\{(x^{(m)}_{1:n+1}, y^{(m)}_{1:n+1})\}_{m=1}^{M}$

$f^{(m)}$ $\quad y_i = f(x_i) + \epsilon_i$

Limited data per context

Big data of cross tasks

Brown et al: Language models are few-shot learners. NeuRIPS, 2020.

1. In-context learning (ICL):

$$\left(\begin{array}{cccc|c} x_1 & x_2 & \dots & x_n & x_{n+1} \\ y_1 & y_2 & \dots & y_n & ? \end{array}\right)$$

Context

$(x_{1:n}, y_{1:n})$

$$\hat{y}_{n+1} = f_\theta(x_{n+1} \mid x_{1:n}, y_{1:n})$$ Forward prediction

Training data: $\left\{(x^{(m)}_{1:n+1}, y^{(m)}_{1:n+1})\right\}_{m=1}^{M}$ $f^{(m)}$ $y_i = f(x_i) + \epsilon_i$

Many open questions: $x_i \in \mathbb{R}^d, \quad y_i \in \mathbb{R}: \quad f_\theta : \mathbb{R}^{n(d+1)+d} \to \mathbb{R}$

- Why and how do transformers work?
- Task complexity: How large can the class of functions $\{f^{(m)}\}$ be? (low-dimensional structure in f(Z) or Z?)
- ....

Brown et al: Language models are few-shot learners. NeuRIPS, 2020.

1. In-context learning (ICL): linear regression

$$\begin{pmatrix} x_1 & x_2 & \dots & x_n & x_{n+1} \\ y_1 & y_2 & \dots & y_n & ? \end{pmatrix}$$

Context

$(x_{1:n}, y_{1:n})$

$$\hat{y}_{n+1} = f_\theta(x_{n+1} \mid x_{1:n}, y_{1:n})$$

Forward prediction

Training data: $\left\{ (x^{(m)}_{1:n+1}, y^{(m)}_{1:n+1}) \right\}_{m=1}^{M}$

$$y_i = \langle x_i, w \rangle + \epsilon_i$$

$$x_i, w \in \mathbb{R}^d$$

- Transformers as gradient descent [1,2,...]
- Scaling limits in token/context length, task diversity [3]

$$n \geq d$$

[1] Ahn et al. Transformers learn to implement preconditioned gradient descent for in-context learning. NeurIPS 2023.

[2] Fu et al. Transformers Learn to Achieve Second-Order Convergence Rates for In-Context Linear Regression. NeurIPS 2024

[3] Lu, Y. M., Letey, M., Zavatone-Veth, J. A., Maiti, A., Pehlevan, C.. Asymptotic theory of in-context learning by linear attention. *PNAS 2025.*

1. In-context learning (ICL): linear regression

$$\begin{pmatrix} x_1 & x_2 & \dots & x_n & x_{n+1} \\ y_1 & y_2 & \dots & y_n & ? \end{pmatrix}$$

Context

$(x_{1:n}, y_{1:n})$

$$\hat{y}_{n+1} = f_\theta(x_{n+1} \mid x_{1:n}, y_{1:n})$$

Forward prediction

Training data: $\left\{(x^{(m)}_{1:n+1}, y^{(m)}_{1:n+1})\right\}_{m=1}^{M}$

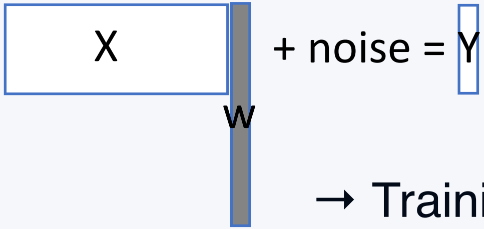
$$y_i = \langle x_i, w \rangle + \epsilon_i$$

ICL of inverse linear regression

Inverse problem

$$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}.$$

$$\widehat{w} = w_\theta(X, Y)$$



Rank deficient: **n << d**

→ Training data encodes a prior (cross-task information)

Priors + Regularization

1. Motivation: Learning Kernels in Operators

Learn convolution kernels from few input–output pairs

$$R_\phi[u](x) = \int \phi(z) g[u](x,z)dy = f(x) + \epsilon(x)$$

$$\{(u_{1:n_0}, f_{1:n_0})\}$$

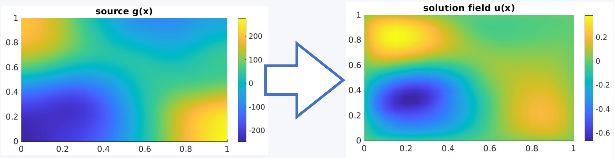
Forward operator

↑

Inverse problem

Integral operators

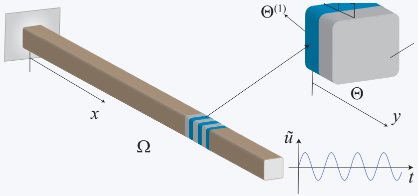
Darcy's equation:
kernel = Green's functions
g[u](x,z) = u(x-z)



Nonlocal Operators

nonlocal diffusion, fractional operators

$g[u](x,z) = u(x+z) - u(x)$



Aggregation Operators

Mean-field limits

$g[u](x,z) = \partial_x[u(x-z)u(x)]$

Popkin. Nature(2016)

1. Motivation: Learning Kernels in Operators

Learn convolution kernels from few input–output pairs

$$R_\phi[u](x) = \int \phi(z) g[u](x,z) dy = f(x) + \epsilon(x)$$

$$\{(u_{1:n_0}, f_{1:n_0})\}$$

Forward operator

↑

Inverse problem

Limited data per task: $Xw = Y + \epsilon; \quad X \in \mathbb{R}^{n\times d}, \quad \mathbf{n} << \mathbf{d}$ **Rank deficient**

Big data of cross tasks: $\left\{(u_{1:n_0}^{(m)}, f_{1:n_0}^{(m)})\right\}_{m=1}^{M} \leftrightarrow \phi^{(m)}$

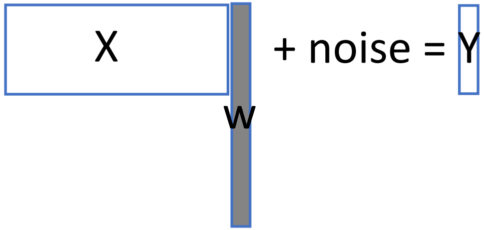
$$\left\{(X^{(m)}, Y^{(m)})\right\}_{m=1}^{M} \leftrightarrow w^{(m)}$$

→ Training data encodes a prior (cross-task information)

Priors + Regularization

1. ICL of inverse linear regression

$$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}. \qquad \widehat{w} = w_\theta(X, Y)$$



High-dimensional, nonlinear: $w_\theta : \mathbb{R}^{n(d+1)} \to \mathbb{R}^{d \times 1}$

- Do transformers learn the prior and regularization?
- Do we need low **task complexity (low-dimensional structure)?**
- How to determine if a transformer is working well?

2. A variant transformer: nonlocal attention operator (NAO)

$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n\times d}.$ $\quad\quad \widehat{w} = w_\theta(X, Y)$

Input $\quad E^{\ell=0} = E = (X, Y) \in \mathbb{R}^{n\times(d+1)}$

Linear Attention $\quad \mathrm{Attn}(E) = W_V\, E(W_K\, E + B_K \mathbf{1}_{d+1})^\top (W_Q\, E + B_Q \mathbf{1}_{d+1})$

LayerNorm with Multi-heads $\quad E^{\ell+1} = E^\ell + \mathrm{LayerNorm}(\Delta E^\ell)$

$$\Delta E^\ell = \sum_{h=1}^{H} \mathrm{Attn}(E^{\ell,h})$$

Output $\quad w_\theta(X, Y) = \mathrm{Attn}^{inv}(E^{\ell=L})$

$$\mathrm{Attn}^{inv}(E) = (W_K\, E + B_K \mathbf{1}_{d+1})^\top (W_Q\, E + B_Q \mathbf{1}_{d+1}) W_P$$

$$\theta = \big((W_K^\ell, W_Q^\ell, W_V^\ell, B_K^\ell, B_Q^\ell)_{\ell=1}^{L}, W_P^L\big)$$

Loss function $\quad L(\theta) = \mathbb{E}_{Data} \| X w_\theta(X, Y) - Y \|_2^2$

Yu, Y., Liu, N., Lu, F., Gao, T., Jafarzadeh, S., and Silling, S. A.. Nonlocal attention operator: … NeurIPS, 2024.

2. A variant transformer: nonlocal attention operator (NAO)

$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n\times d}.$ $\qquad \widehat{w} = w_\theta(X, Y)$

Input $\qquad E^{\ell=0} = E = (X, Y) \in \mathbb{R}^{n\times(d+1)}$

Linear Attention $\qquad \mathrm{Attn}(E) = W_V\, E(W_K\, E + B_K \mathbf{1}_{d+1})^\top (W_Q\, E + B_Q \mathbf{1}_{d+1})$

$W_V \in \mathbb{R}^{n\times n}, W_K, W_Q \in \mathbb{R}^{d_k\times n}$

- Outperforms softmax attention in tests
- Computationally efficient: O(n^2) ⟶ O(n) [1]
- $E^\top$ v.s. E: converges to RKHS as $d \to \infty$ (n < d)

[1] Katharopoulos, A., Vyas, A., Pappas, N. and Fleuret, F.. Transformers are RNNs: Fast autoregressive transformers with linear attention. ICML, 2020.

3. The transformer learns the prior: Gaussian settings

$y_i = \langle x_i, w \rangle + \epsilon_i, \quad 1 \leq i \leq n < d$ $\quad x_i \sim \mathcal{N}(0, \Sigma_x) \quad w \sim \mathcal{N}(w_0, \Sigma_w) \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$

$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}.$ $\quad \Sigma_x, \Sigma_w \in \mathbb{R}^{d \times d} \quad \mathrm{rank}(\Sigma_w) = r_w$

- The transformer outputs approximate the posterior

- **Gaussian**: Extract the prior from the transformer outputs
  + draw new samples of test contexts (X^j, Y^j);
  + get posterior samples $\hat{w}^j = w_\theta(X^j, Y^j)$;
  + Extract the prior mean and covariance: $\widehat{w}_0, \widehat{\Sigma}_w$

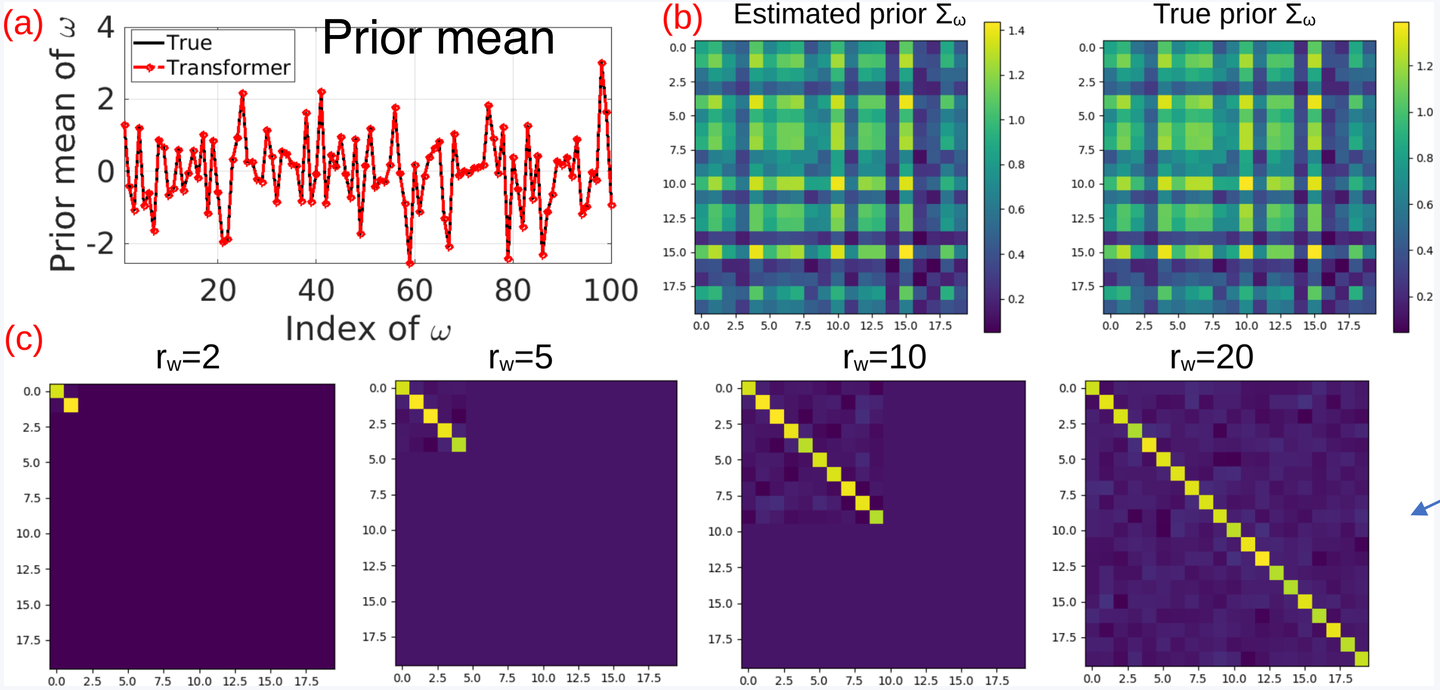
3. The transformer learns the prior: Gaussian settings

$$y_i = \langle x_i, w \rangle + \epsilon_i, \quad 1 \le i \le n < d \qquad x_i \sim \mathcal{N}(0, \Sigma_x) \quad w \sim \mathcal{N}(w_0, \Sigma_w) \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$$

$$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}. \qquad \Sigma_x, \Sigma_w \in \mathbb{R}^{d \times d} \quad \mathrm{rank}(\Sigma_w) = r_w$$

d=100

n=50



Prior Covariance

3. The transformer learns the regularization: baselines

$$y_i = \langle x_i, w \rangle + \epsilon_i, \quad 1 \le i \le n < d \qquad x_i \sim \mathcal{N}(0, \Sigma_x) \quad w \sim \mathcal{N}(w_0, \Sigma_w) \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$$

$$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}. \qquad \Sigma_x, \Sigma_w \in \mathbb{R}^{d \times d} \quad \mathrm{rank}(\Sigma_w) = r_w$$

Ridge estimator (RE):

$$\hat{w}^{RE} = (X^\top X + \lambda I_d)^{-1} X^\top Y$$

Two-stage Ridge estimator (TRE): 1. Estimate the prior; 2. Ridge regression

$$\hat{w}^{TRE} = (X^\top X + \lambda \hat{\Sigma}_w^\dagger)^\dagger X^\top (Y - X\hat{w}_0) + \hat{w}_0$$

Oracle Ridge estimator (ORE):

$$\hat{w}^{ORE} = (X^\top X + \sigma_\varepsilon^2 \Sigma_w^\dagger)^\dagger X^\top (Y - X w_0) + w_0$$

= Posterior mean

Unknown

F. Lu and Y. Yu. Transformers learn cross-task prior and regularization for ICL-ILR. 2025.

3. The transformer learns the regularization: optimal bounds

$y_i = \langle x_i, w \rangle + \epsilon_i, \quad 1 \le i \le n < d$ $\quad x_i \sim \mathcal{N}(0, \Sigma_x) \quad w \sim \mathcal{N}(w_0, \Sigma_w) \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$

$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}.$ $\quad \Sigma_x, \Sigma_w \in \mathbb{R}^{d \times d} \quad \mathrm{rank}(\Sigma_w) = r_w$

As n increases (with $r_w/n < 1/2$),

$$\mathbb{E}\left\| \widehat{w}^{\mathrm{ORE}} - w \right\|^2 = O\left(\frac{r_w\, \sigma_\varepsilon^2}{n\, \lambda_{\min}(\Sigma_x)}\right).$$

- Error ~ r_w/n (task dimension/context length)
- Importance of well-conditioned contexts

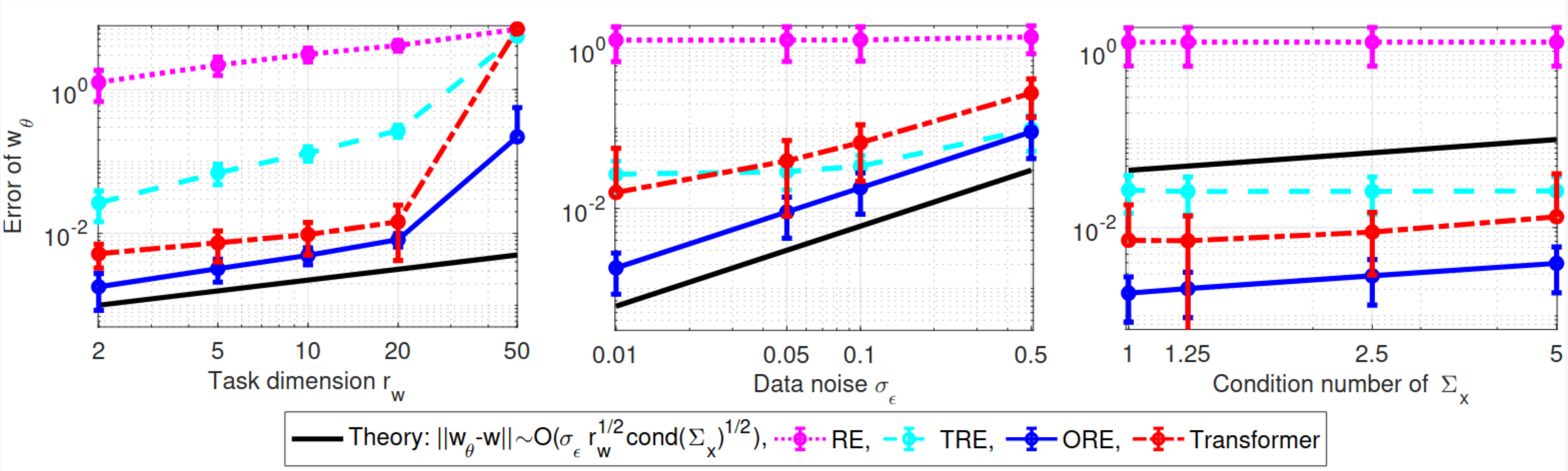
Oracle Ridge estimator (ORE): $\hat{w}^{ORE} = (X^\top X + \sigma_\varepsilon^2 \Sigma_w^\dagger)^\dagger X^\top (Y - X w_0) + w_0$

= Posterior mean

3. The transformer learns the regularization

$$y_i = \langle x_i, w \rangle + \epsilon_i, \quad 1 \le i \le n < d \qquad x_i \sim \mathcal{N}(0, \Sigma_x) \qquad w \sim \mathcal{N}(w_0, \Sigma_w) \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$$

$$Y = Xw + \varepsilon, \quad X \in \mathbb{R}^{n \times d}. \qquad \Sigma_x, \Sigma_w \in \mathbb{R}^{d \times d} \qquad \mathrm{rank}(\Sigma_w) = r_w$$



- **ORE > NAO > TRE > RE: NAO achieves scalings aligning with ORE.**

3. None-Gaussian prior / noise

NAO works for non-Gaussian distributions

+ It does not use any distribution information

+ Tests on a Uniform prior: the scaling patterns remain

$$O\left(\frac{\sigma_{\epsilon}^{2} r_{w}}{n \lambda_{min}(\Sigma_{x})}\right)$$

It leads to a new regularization strategy <u>tailored to each setting</u>.

4. Conclusion and outlook

ICL-ILR via transformers:

- The transformers learn priors and regularization strategies
- <u>Low task dimensionality</u> relative to context length is essential
- Errors scale with noise and condition number

4. Conclusion and outlook

ICL-ILR via transformers:

- The transformers learn priors and regularization strategies
- Low task dimensionality relative to context length is essential
- Errors scale with noise and condition number

Future work:

- ICL for ill-conditioned inverse problems

  (learning kernels in operators)
- Understand why and how transformers work for inverse problems
- Scaling limits, sample complexity, OOD

A new area, many important questions open.