Learning Linear Regression with Low-Rank Tasks In-Context

Kaito Takanami[1], Takashi Takahashi[2,3], Yoshiyuki Kabashima[1,2]

[1] Department of Physics, The University of Tokyo, [2] Institute for Physics of Intelligence, The University of Tokyo, [3] RIKEN AIP

Introduction

In-Context Learning (ICL)

The ability to infer a task from a context of examples and solve it at test time, without updating model parameters.

Why task structure?

- In many applications, tasks are not isolated: they come from a family with shared statistical structure[1].
- Multi-task learning and meta-learning have been the two main approaches to exploit such task structure.

? Research Question

- How can ICL exploit task structures in pretraining?
- What happens when structure-mismatched tasks are presented during inference?

Theoretical Model

We study the asymptotic behavior of in-context learning for linear regression tasks[2] drawn from a shared low-rank task family.

- Pretraining regression weights $\mathcal{M}_0 = \{\boldsymbol{w}^\nu \in \mathbb{R}^D\}_{\nu=1,\dots,M_0}$ share a common structure $A \in \mathbb{R}^{D\times r}$ $(r < D)$:

$$\boldsymbol{w}^\nu = A\boldsymbol{v}^\nu, \quad \boldsymbol{v}^\nu \sim \mathcal{N}(\boldsymbol{0}, I_r)$$

- We construct the training task set by sampling M tasks with replacement from M_0 task types.
- For each task μ, we draw L examples $(\boldsymbol{x}_l^\mu, y_l^\mu)$:

$$y_l^\mu = \boldsymbol{w}^\mu \cdot \boldsymbol{x}_l^\mu + \varepsilon_l^\mu, \quad \varepsilon_l^\mu \sim \mathcal{N}(0, \sigma^2)$$

- Embedding of the input context:

$$C^\mu = \begin{pmatrix} \boldsymbol{x}_1^\mu & \dots & \boldsymbol{x}_L^\mu & \boldsymbol{x}_{L+1}^\mu \\ y_1^\mu & \dots & y_L^\mu & 0 \end{pmatrix} \in \mathbb{R}^{d\times(L+1)}$$

- We use a one-layer linear attention model with supervised pretraining. The prediction for $\boldsymbol{x}_{L+1}^\mu$ is given by $\hat{y}_{L+1}^\mu = \mathrm{Readout}(\mathrm{Attention}(C^\mu))$.

Decomposition of estimator

After pretraining, we present a new inference prompt:

$$P = \begin{pmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_{\tilde{L}} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{x} \\ y_1 & \dots & y_{\tilde{L}} & 0 & \dots & 0 & 0 \end{pmatrix} \in \mathbb{R}^{d\times(\tilde{L}+1)},$$

where $\boldsymbol{y}_l = \boldsymbol{w} \cdot \boldsymbol{x}_l$ is a new in-context example.

Result1: signal to noise decomposition

The prediction for $\boldsymbol{x}$ from a pretrained model can be decomposed into a signal term and noise terms:

$$\hat{y} = \underbrace{\hat{y}_{\mathrm{algo}}}_{\text{signal}} + \underbrace{\hat{y}_{\mathrm{mem}} + \hat{y}_{\mathrm{struct}}}_{\text{noise}}$$

Result2: asymptotics of each term and its interpretation

- $\alpha = L/D$ (in-context sample ratio in pretraining)
- $\tilde{\alpha} = \tilde{L}/D$ (in-context sample ratio in inference)

$$\hat{y}_{\mathrm{algo}} = \underbrace{\left(P_A \boldsymbol{w}^*_{\mathrm{prom}}\right)^\top \boldsymbol{x}_{L+1}}_{\text{linear regression}} + \mathcal{O}(1/\alpha), \qquad \overbrace{P_A}^{\text{projection onto learned structure}} = AA^\top$$

The inner algorithm:

1. Make a matched-filter estimate of the weight: $\boldsymbol{w}^*_{\mathrm{prom}} = \sum_l y_l \boldsymbol{x}_l/\tilde{\alpha}$
2. Project it onto the learned structure: $P_A \boldsymbol{w}^*_{\mathrm{prom}}$
3. Do linear regression: $P_A \boldsymbol{w}^*_{\mathrm{prom}} \cdot \boldsymbol{x}$

$$\hat{y}_{\mathrm{mem}} \sim \mathcal{N}\left(0, \text{const.} \times \underbrace{\left(A^\top \boldsymbol{w}^*_{\mathrm{prom}}\right)^\top \left(\sum_\mu \boldsymbol{v}^\mu (\boldsymbol{v}^\mu)^\top\right) \left(A^\top \boldsymbol{w}^*_{\mathrm{prom}}\right)}_{\text{Mahalanobis distance (memorized vs. estimated tasks)}}\right)$$

Memorization noise:
If the estimated task $\boldsymbol{w}^*_{\mathrm{prom}}$ is close to the memorized task $\boldsymbol{w}^\mu$, this noise is small. (→ recalling the memorized task)

$$\hat{y}_{\mathrm{struct}} \sim \mathcal{N}\left(0, \text{const.} \times \alpha \underbrace{\left|P_A^\perp \boldsymbol{w}^*_{\mathrm{prom}}\right|^2}_{\text{orthogonal component of estimated task}} / M_0\right)$$

Structure noise:
If the orthogonal component of the estimated task $\boldsymbol{w}^*_{\mathrm{prom}}$ relative to the learned structure A is small, this noise is small. (→ recalling structure)

☑ Takeaways

- Pretraining yields a structure-aware efficient linear regression algorithm.
- Prediction noise reduces generalization, but shrinks for memorized tasks and structures.

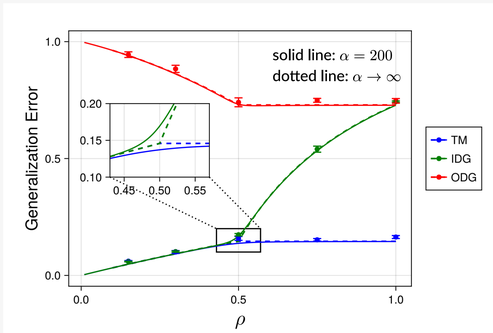
Trade-off from task difficulty

Three types of inference settings:

- **Task Memorization (TM)**: $w^* \sim \mathrm{Unif}(\mathcal{M}_0)$ (task is in the pretraining task set.)
- **In-distribution Generation (IDG)**: $w^* \sim A\boldsymbol{v}^*, \quad \boldsymbol{v}^* \sim \mathcal{N}(\boldsymbol{0}, I_r)$ (task follows the pretraining structure but is not one of the memorized pretraining tasks.)
- **Out-of-distribution Generation (ODG)**: $w^* \sim \mathcal{N}(\boldsymbol{0}, I_D)$ (task is not in the pretraining task set.)

Result3: phase transition of generalization error

Generalization performance shows a sharp phase transition as a function of ρ under $\alpha \to \infty$.



$\rho = r/D$ is the rank ratio (**task difficulty**).

$\kappa = M_0/D$ is the ratio of the number of task types (**task diversity**).

- $\rho < \kappa$ (**task-rich regime**): Better use of low-rank structure → better ID performance, worse OOD robustness.
- $\rho > \kappa$ (**task-deficient regime**): Diversity bottleneck → easier tasks ($\rho \downarrow$) do not improve TM/ODG performance.

☑ Takeaways

- Model capacity is determined by the balance between task-diversity and task-difficulty.
- Near $\rho \approx \kappa$, the model balances in-distribution performance and robustness in our setting.

[1] Han, L. and Zhang, Y. (2016). Multi-stage multi-task learning with reduced rank, AAAI 30(1).
[2] Lu et al. (2025), Asymptotic theory of in-context learning by linear attention, PNAS 122(28).

This work was supported by JST BOOST NAIS (Grant No. JPMJBS2418), JSPS KAKENHI (Grant No. 23K16960, 22H05117), and JST ACT-X (Grant No. JPMJAX24CG).