Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection

NeurIPS 2023

Yu Bai, Fan Chen, Huan Wang,Caiming Xiong, Song Mei

Stanford CS/MS&E 331

Motivation

- In-context learning (ICL): transformer trained to produce map
  - **Input:** sequences $\left[\left(x_1, f(x_1)\right), \left(x_2, f(x_2)\right), \dots, x_n\right]$
  - **Output:** prediction of $f(x_n)$
- **This paper:** algorithmic reasoning as a lens to understand ICL
- Algorithmic task: regression
  - $\boldsymbol{x} \in \mathbb{R}^d, f(\boldsymbol{x}) \in \mathbb{R}$
- ICL isn't learning a **regressor;** rather a regression **algorithm**
  - ICL doesn't explicitly specify inner learning procedure
  - Procedure exists only implicitly through transformer's parameters

Motivation

Goal: Algorithmic reasoning as a lens to understand ICL

Prior work: transformers (TFs) can mimic regression algorithms
[e.g., Akyürek et al., ICLR'23]

Humans choose algorithms **adaptively** based on data

Can transformers also *select* which algorithm to use?

This paper: TFs perform adaptive in-context alg. selection

Contributions

Core idea: Transformers act as adaptive statistical learners

- Represent and execute many standard ML algorithms
- TFs choose which algorithm fits the observed data
- Adapt automatically to task characteristics (e.g., noise, sparsity)

Mechanisms enabling selection:

- **Post-ICL validation:**
  Compare candidate predictors on held-out examples
- **Pre-ICL testing:**
  Identify task type before learning (e.g., regression vs classification)

In-context learning (ICL)

ICL instance $(\mathcal{D}, \boldsymbol{x}_{N+1})$

- Dataset $\mathcal{D} = [(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_n)]$ of labeled examples
- $\boldsymbol{x}_i \in \mathbb{R}^d$ sampled from distribution (e.g., $\mathcal{N}(\mathbf{0}, I_d)$)
- $y_i \in \mathbb{R}$ are labels (e.g., real-valued regression, binary classification, …)
- Test input $\boldsymbol{x}_{N+1}$

Each instance $(\mathcal{D}, \boldsymbol{x}_{N+1})$ drawn from a different distribution P_j

- E.g., defined by different linear models with $y_i = \boldsymbol{w}_j^\top \boldsymbol{x}_i$

Goal: construct fixed TF to perform ICL on large set of P_js

Outline

1. **Theory**
2. Empirics

In-context gradient descent (ICGD)

Akyürek et al. [ICLR'23] proved guarantees for single-step ICGD

- Focus on expressivity: layers, width, …
- What about TFs that **approximate** GD up to some error?

This paper: ϵ-approximation analysis for multi-step ICL

1. For any desired single-step ICGD error tolerance $\epsilon > 0$:
   A transformer can be constructed to meet that target
2. For L-layer TF, error accumulates linearly $\left(O(L\epsilon)\right)$, not exponentially

Provides explicit dependence on ϵ, L, and model parameters

In-context gradient descent (ICGD)

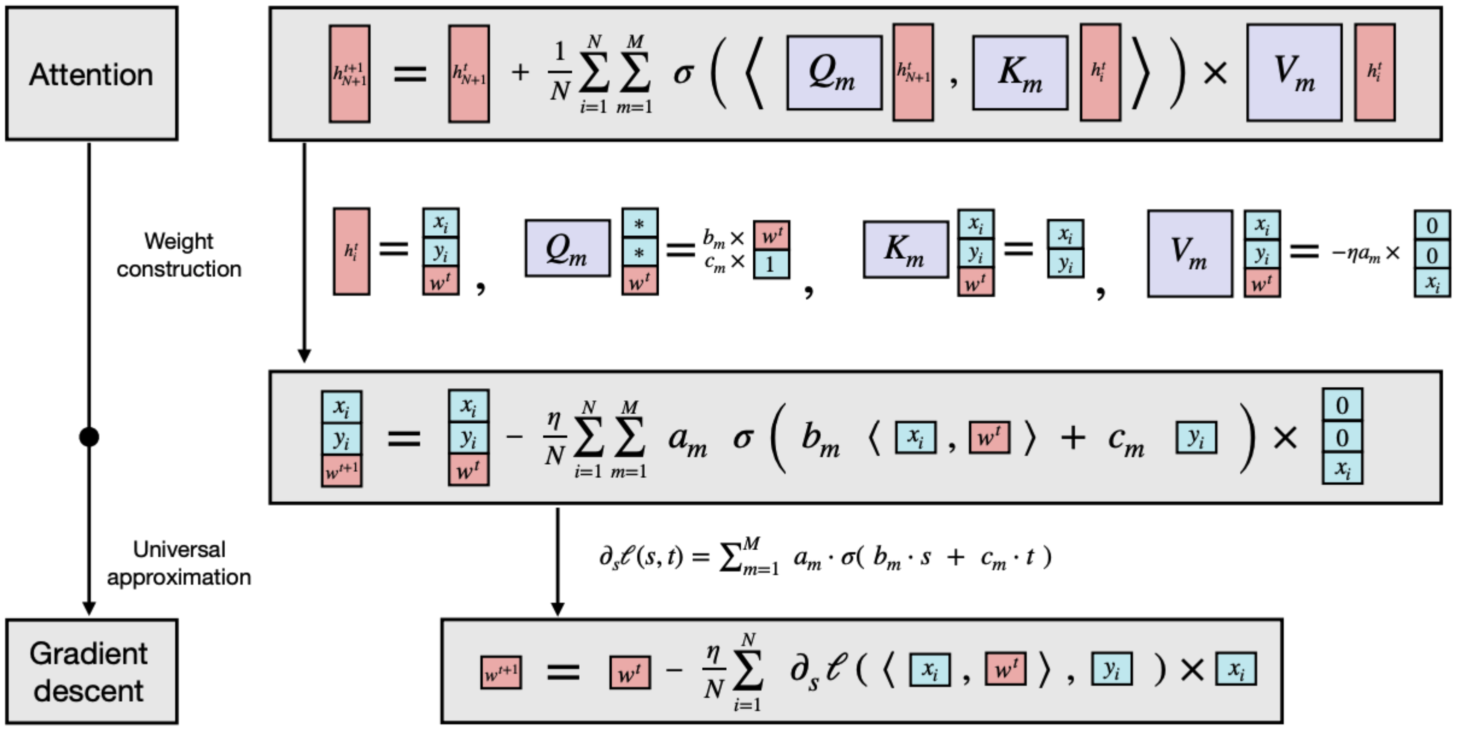


Figure 3 in [extended version]() of paper explicates TF weights

Ridge Regression / Least Squares

ICGD results serve as a **reusable foundation** for ICL regression

Akyürek et al. [ICLR'23]: single-step GD for linear models only

This work extends to a **broader class** of objectives, e.g.,:

- Lasso: via proximal gradient descent
- Logistic regression for linear classification

Each inherits ϵ-approximation guarantees

Mechanism: Post-ICL algorithm selection

- **Objective:** Allow single TF to adapt across different tasks
- **Setup:** Input dataset $\mathcal{D} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$
  - K learning algorithms realizable by ICGD (e.g., Ridge w/ different λs)
  - Convex loss function
- **Training phase:** compute predictors $f_1, \ldots, f_K$ with $\mathcal{D}_{\text{train}}$
- **Validation phase:** evaluate each f_i on $\mathcal{D}_{\text{val}}$; loss $\hat{L}_{\text{val}}(f_i)$
- **Selection:** choose nearly-optimal candidate

$$\hat{f} \in \text{conv}\left\{f_i : \hat{L}_{\text{val}}(f_i) \le \min_{i^* \in [K]} \hat{L}_{\text{val}}(f_{i^*}) + \gamma\right\}$$

Convex loss and sufficiently large $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}} \Rightarrow \hat{f}$ nearly optimal

Mechanism: Post-ICL algorithm selection

- **Example:** Noisy linear models with mixed noise levels
- **Setup:** Data generating distribution π:
  - $\boldsymbol{w} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right), \boldsymbol{x}_i \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right)$
  - K noise levels $\sigma_1, \dots, \sigma_K$ and $\Lambda =$ distribution over $\{\sigma_1, \dots, \sigma_K\}$
  - Sample $\sigma_k \sim \Lambda$, define $y_1 = \boldsymbol{w}^\top \boldsymbol{x}_1 + \mathcal{N}(0, \sigma_k), \dots, y_N = \boldsymbol{w}^\top \boldsymbol{x}_N + \mathcal{N}(0, \sigma_k)$
  - $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$

$$\mathrm{BayesRisk}_\pi = \inf_{\mathcal{A}} \mathbb{E}_{\pi} \left[\frac{1}{2} \left(\underline{\mathcal{A}(\mathcal{D})(\boldsymbol{x}_{N+1})} - y_{N+1} \right)^2 \right]$$

Prediction of learning algorithm $\mathcal{A}$ on test instance $\boldsymbol{x}_{N+1}$ when trained on $\mathcal{D}$

Mechanism: Post-ICL algorithm selection

- **Example:** Noisy linear models with mixed noise levels
- **Setup:** Data generating distribution π:
  - $\boldsymbol{w} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right), \boldsymbol{x}_i \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right)$
  - K noise levels $\sigma_1, \ldots, \sigma_K$ and Λ = distribution over $\{\sigma_1, \ldots, \sigma_K\}$
  - Sample $\sigma_k \sim \Lambda$, define $y_1 = \boldsymbol{w}^\top \boldsymbol{x}_1 + \mathcal{N}(0, \sigma_k), \ldots, y_N = \boldsymbol{w}^\top \boldsymbol{x}_N + \mathcal{N}(0, \sigma_k)$
  - $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$

$$\mathrm{BayesRisk}_\pi = \inf_{\mathcal{A}} \mathbb{E}_{\pi}\left[\frac{1}{2}\left(\mathcal{A}(\mathcal{D})(\boldsymbol{x}_{N+1}) - y_{N+1}\right)^2\right]$$

- Optimal $\mathcal{A}$ = mixture of $\mathrm{RidgeRegression}\left(\lambda_k = \frac{d\sigma_k^2}{N}\right)$

Mechanism: Post-ICL algorithm selection

- **Example:** Noisy linear models with mixed noise levels
- **Thm:** TF with $O(\log N)$ layers, $O(K)$ heads outputs $\hat{y}_{N+1}$ s.t.

$$\mathbb{E}_{\pi}\left[\frac{1}{2}(\hat{y}_{N+1} - y_{N+1})^2\right] \leq \mathrm{BayesRisk}_{\pi} + O\left(\sqrt[3]{\frac{\log K}{N}}\right)$$

- Akyürek et al. [ICLR'23]:
  Empirical: TFs achieve nearly-optimal risk under any fixed σ
- This theorem: Single TF can achieve nearly-optimal Bayes risk under a mixture of K noise levels

Mechanism: Pre-ICL testing

Objective: select algorithm before learning in context

- Distinguish regression/scalar labels from binary labels

Theorem: exists TF with $O\left(\log\frac{1}{\epsilon}\right)$ layers such that:

- If y_i's are in $\{0,1\}$:
  - Outputs $\hat{y}_{N+1}$ that ϵ-approximates logistic regression
- Otherwise, outputs $\hat{y}_{N+1}$ that ϵ-approximates least squares

Outline

1. Theory
2. **Empirics**

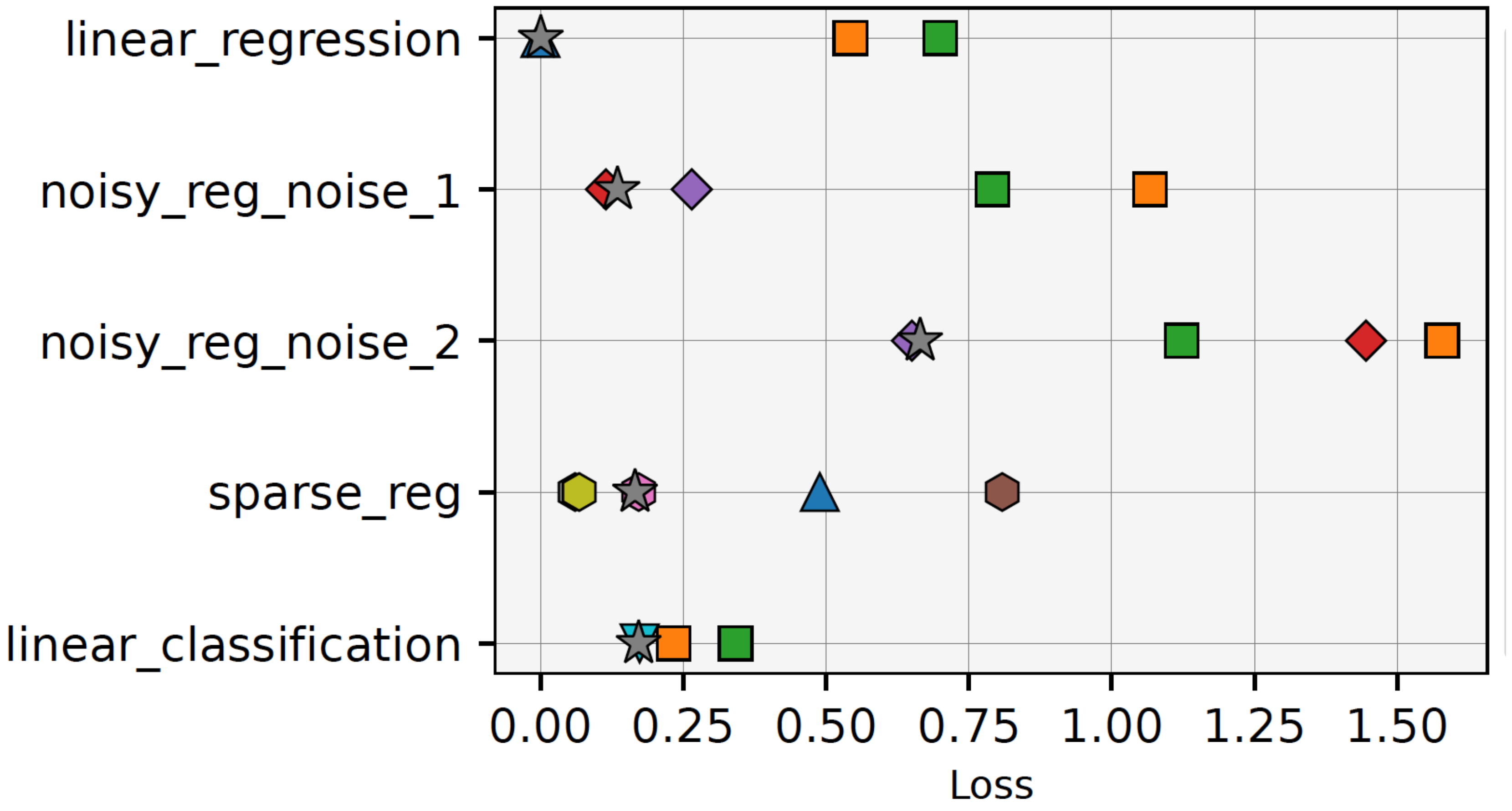
Experiments

12-layer transformer

"Base mode" setup: $d = 20, \boldsymbol{x}_i \sim \mathcal{N}(\mathbf{0}, I_d)$

- Linear model: $\boldsymbol{w} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right), y_i = \boldsymbol{w}^\top \boldsymbol{x}_i$
- Noisy linear model: $\boldsymbol{w} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right), y_i = \boldsymbol{w}^\top \boldsymbol{x}_i + \mathcal{N}(0, \sigma)$
  - Experiments: $\sigma \in \{\sigma_1, \sigma_2\} = \{0.1, 0.5\}$
- Sparse: $\boldsymbol{w}$ sampled from prior supported on $\|w\|_0 \leq s, y_i = \boldsymbol{w}^\top \boldsymbol{x}_i$
  - Experiments: $s = 3$
- Linear classification model: $\boldsymbol{w} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{d} I_d\right), y_i = \mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i)$

TFs approximately match best baselines



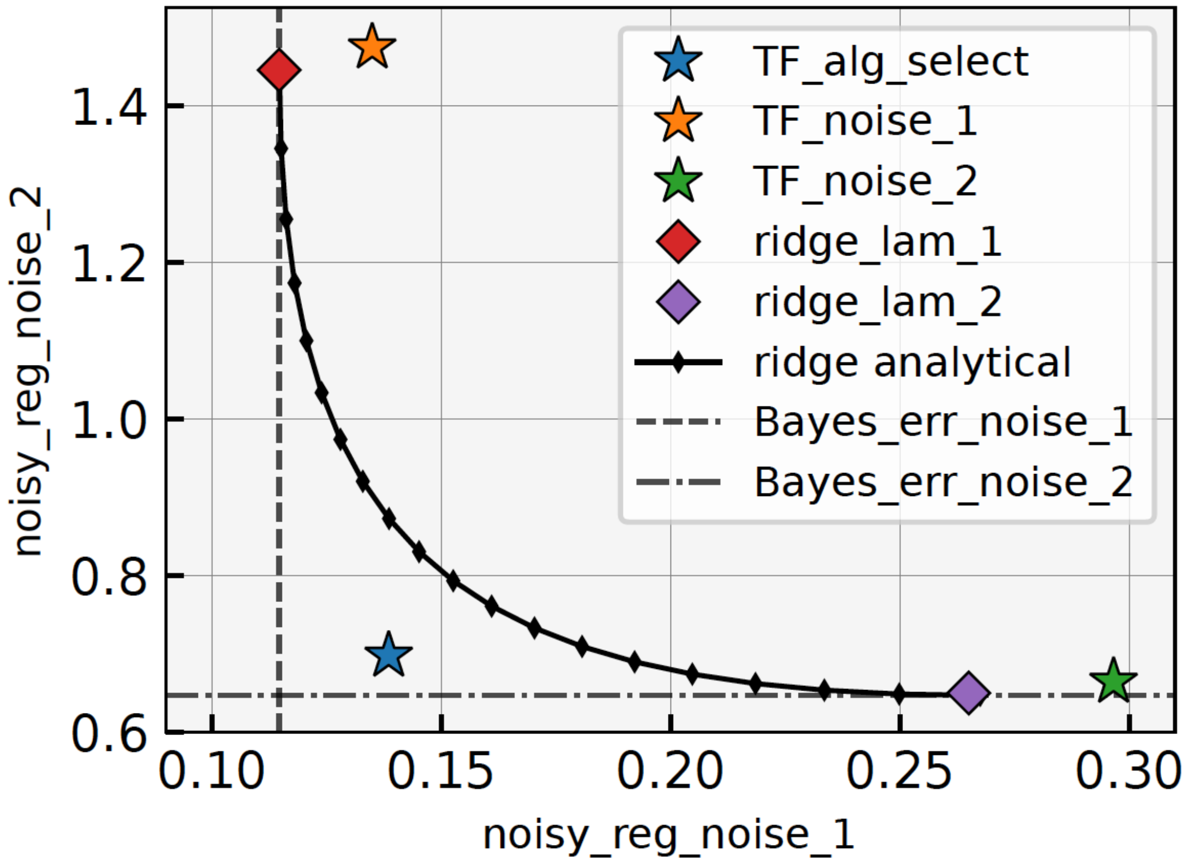
Experiments

"Mixture mode" setup: mixture of 2+ base modes

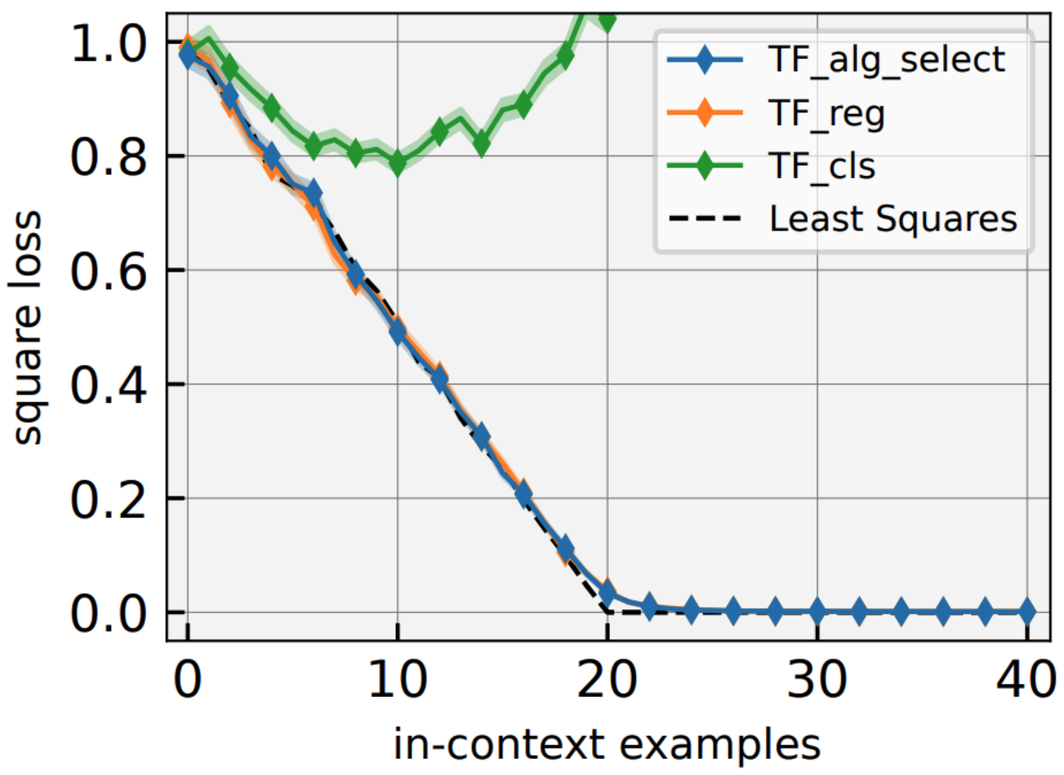
- Linear model + linear classification model
- Noisy linear model with noise levels $\sigma \in \{0.1, 0.25, 0.5, 1\}$

TF trained/evaluated on multiple base modes simultaneously

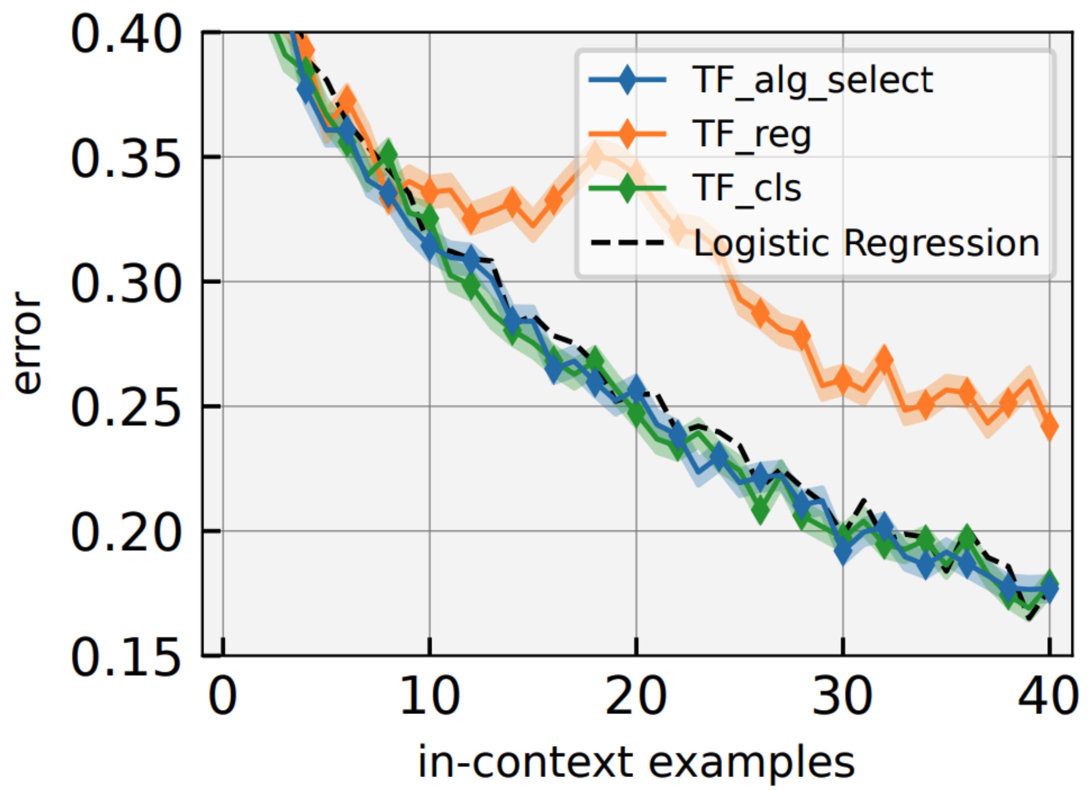
TF approaches Bayes risk on both tasks



TF nearly matches the best baseline



Regression



Classification

Summary

Core idea: Transformers act as adaptive statistical learners

- Represent and execute many standard ML algorithms
- TFs choose which algorithm fits the observed data
- Adapt automatically to task characteristics (e.g., noise, sparsity)

Mechanisms enabling selection:

- **Post-ICL validation:**
  Compare candidate predictors on held-out examples
- **Pre-ICL testing:**
  Identify task type before learning (e.g., regression vs classification)