What Can Transformers Learn In-Context?

A Case Study of Simple Function Classes

CSCI-699: Computational Perspectives on the Frontiers of Machine Learning

Paper by Ekin et al. (NeurIPS 2022 oral)

Presenter: Jingmin Wei. Apr 3, 2023

Outline

- In-context Learning
- Experiment Settings
- In-context Learning of Linear Functions
- Extrapolating Beyond the Training Distribution (Shift)
- In-context Learning of More Complex Function Classes
- Investigating Key Factors for In-context Learning
- Conclusions

In-context Learning

In-context Learning

ICL for sentiment analysis.

J Delicious food -> 1, The food is awful -> 0, Terrible dishes -> 0, Good meal -> ?

"Good meal" can be considered as 1 in a sentiment analysis context, as it is generally a positive statement about the food.

Inference with prompts, without parameter updates.

English translations of French words after being prompted on a few such translations.

J maison -> house, chat -> cat, chien -> ?

The French word "chien" means "dog" in English.

Many LLMs exhibit ability to perform in-context learning.

*What's the difference between ICL and ZSL?

Problem and Experiments

Problem Def: Given data derived from some functions class, can we train a Transformer model to in-context learn “most” functions from this class?

Experiment 1:

- Standard Transformers can be trained from scratch to in-context learn linear functions.
- Even under some distribution shift, in-context learning is possible.

Experiment 2:

- Transformers can be trained to in-context learn more complex function classes.

Experiment 3:

- What are the key factors for in-context learning?

Experiment Settings

Prompt, In-context Exps and Query

$D_{\mathcal{F}}$: Function distribution. $D_{\mathcal{X}}$: Data distribution.

P : prompt $P = (x_1, f(x_1), \cdots, x_k, f(x_k))$

Sample a random function f from the class according to $D_{\mathcal{F}}$, create a set of random inputs $x_1, \cdots, x_{k+1}$ drawn independently from $D_{\mathcal{X}}$.

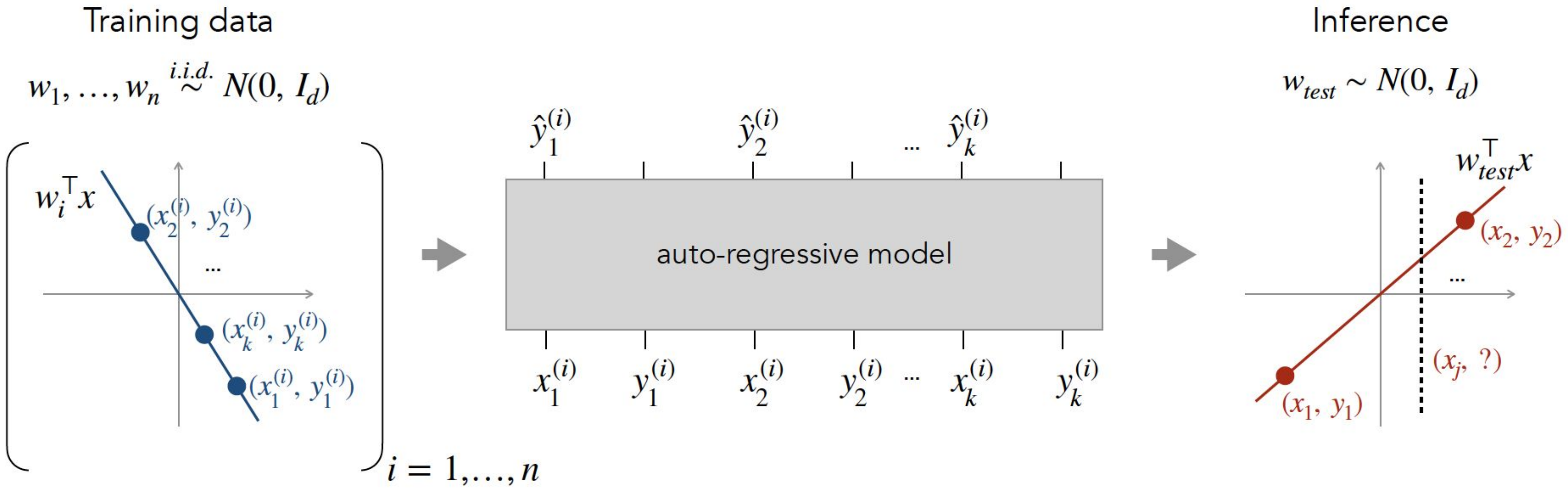
E.g. Sample n Inputs, weights. Each input $x_i = (x_1, x_2, \cdots, x_k)^{(i)}$, weight w_i are i.i.d. from isotropic Gaussian distribution $N(\mathbf{0}, I_d)$. Then set $f(x_i) = w_i^T x_i$, get prompt sequence $(x_1, f(x_1), \cdots, x_k, f(x_k))$.

USC

Transformers Structure

Decoder-only Transformer architecture from GPT-2.

12 layers, 8 attention heads, and a 256 - dimensional embedding space (22.4 M parameters).



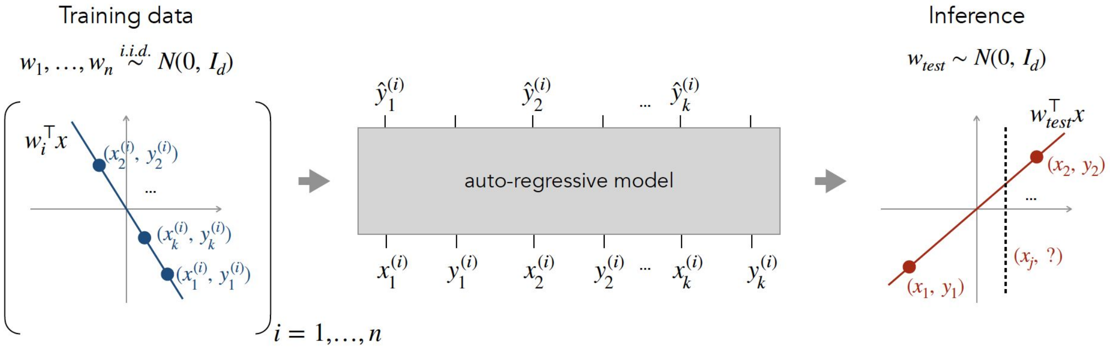
ICL Pipeline

Sample n training inputs and weights. Each input $x_i = (x_1^{(i)}, x_2^{(i)}, \cdots, x_k^{(i)})$, weight w_i are i.i.d. from isotropic Gaussian distribution $N(0, I_d)$. Then set $f(x_i) = w_i^T x_i$, get prompt sequence $(x_1, f(x_1), \cdots, x_n, f(x_n))$.

For each i , given $(x_1^{(i)}, f(x_1^{(i)}), \cdots, x_{k-1}, f(x_{k-1}^{(i)}), x_k^{(i)})$, train the Transformer model to auto-regressively predict $\hat{f}(x_k^{(i)})$.

Then sample input $x = (x_1, x_2, \cdots, x_j)$, weights w_{test} from $N(0, I_d)$. Set $f(x) = w_{test}^T x$, get in-context pair $(x_1, f(x_1), \cdots, x_{j-1}, f(x_{j-1}), x_j)$ and label $f(x_j)$,where x_j represents x_{query} .

Predict $\hat{f}(x_j)$ using model, evaluate the squared error with $f(x_j)$.



Target

P^i (Prompt prefix): containing i in-context examples and $i+1$ query input:
$P^i = (x_1, f(x_1), \cdots, x_i, f(x_i), x_{i+1})$.

M_θ : model with param θ to minimize loss (over all prompt prefixes).

$l(\cdot, \cdot)$: an appropriately chosen loss function.

$$\min_\theta \mathbb{E}_p \left[\frac{1}{k+1} \sum_{i=1}^{k} l\left(M_\theta(P^i), f(x_{i+1})\right) \right]$$

In-context Learning of Linear Functions

Notations

Functions consider function class $\mathcal{F} = \{f | f(x) = w^T x, w \in \mathbb{R}^d\}$ with $d = 20$.

Inputs & weights: $x_1, \cdots, x_k, x_{query}$; w from isotropic Gaussian distribution $N(0, I_d)$.

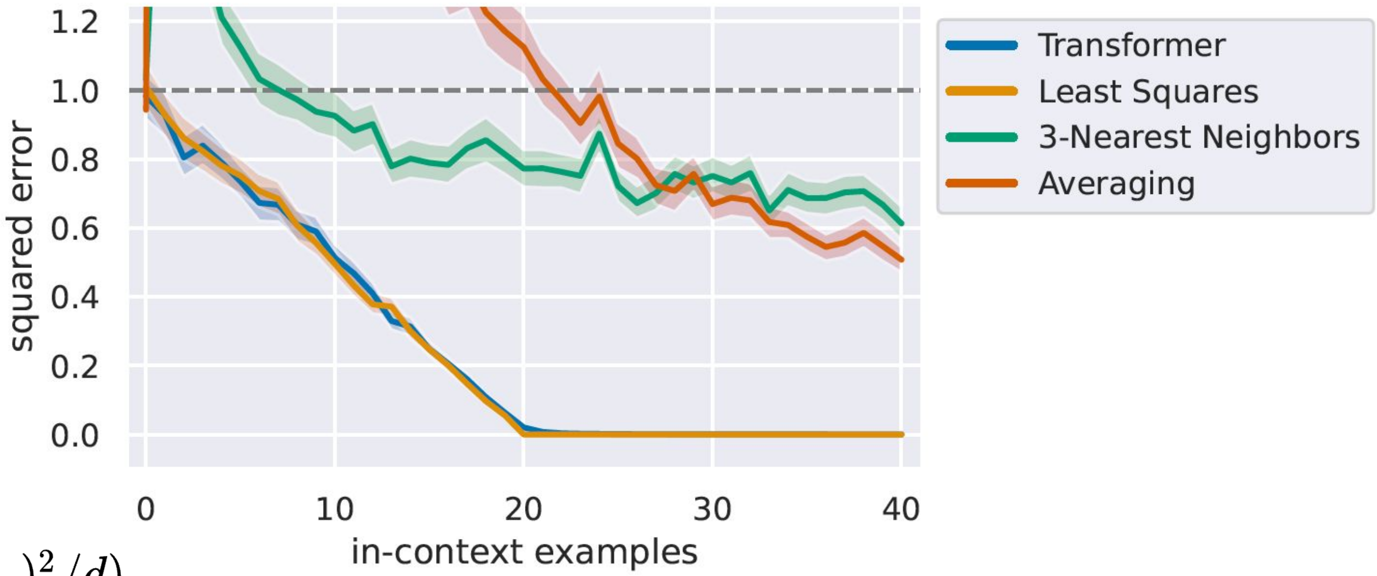
Labels: compute $y_i = w^T x_i$

Prompt: $P = (x_1, y_1, \cdots, x_k, y_k, x_{query})$

Baselines: compare the in-context Transformer with other learning baseline algorithms:

1. Least squares estimator (min-norm linear fit to (x_i, y_i))
2. N-nearest neighbors (averaging the y_i values for the n nearest neighbors of x_{query})
3. Directly calculate $w = avg(y_i x_i)$, use this to compute $w^T x_{query}$

In-context Learning Linear Functions



$$(M(P) - w^T x_{query})^2 / d)$$

Evaluate the trained Transformer on in-context learning linear functions

What functions the model learn?

The model learn from prompt input $P = (x_1, w^T x_1, \cdots, x_k, w^T x_k, x_{query})$, ideally output $w^T x_{query}$.

Prefix-conditioned Function: If we fix the prefix given by k in-context examples, we can view the output of the model as a function $\hat{f}_{w, x_{1:k}}(x_{query})$, that approximates $w^T x_{query}$.

When $k < d$, the ideal model should approximate $(proj_{x_{1:k}}(w))^T x_{query}$, where $proj_{x_{1:k}}(w)$ is the projection of w onto the subspace spanned by $x_1, \cdots, x_k$.

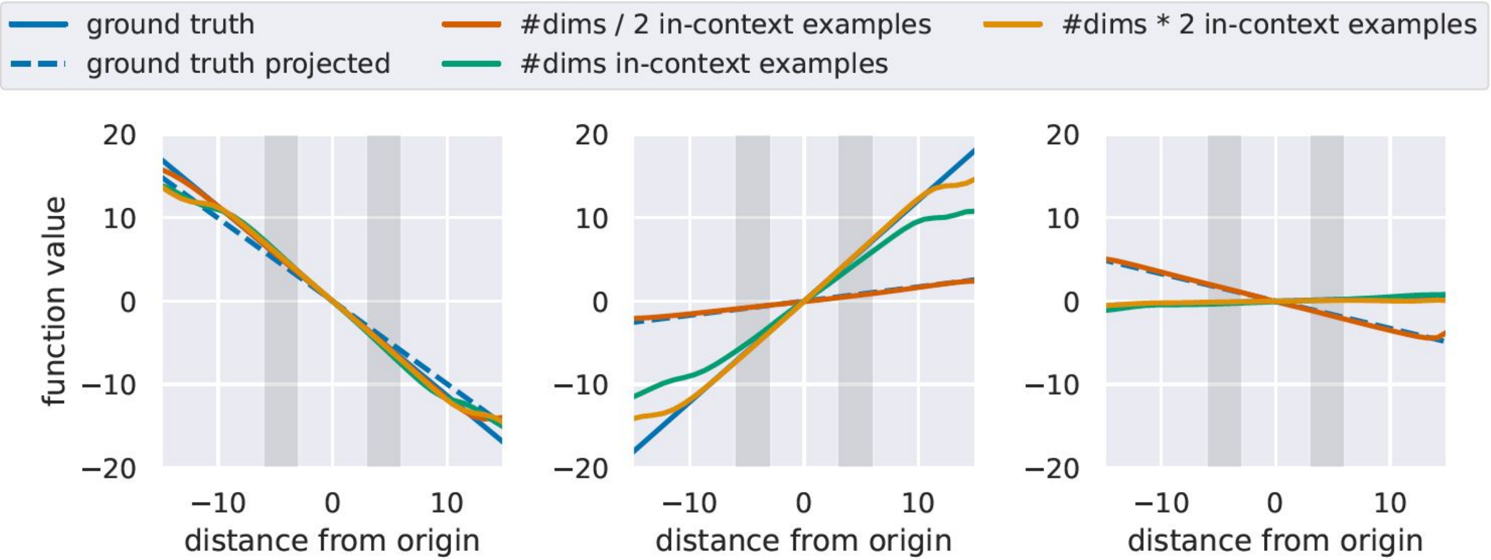
Prefix-conditioned Function

y : visualize the function $\hat{f}_{w,x_{1:k}}(x_{query})$.

x : vary query input along a random direction x .

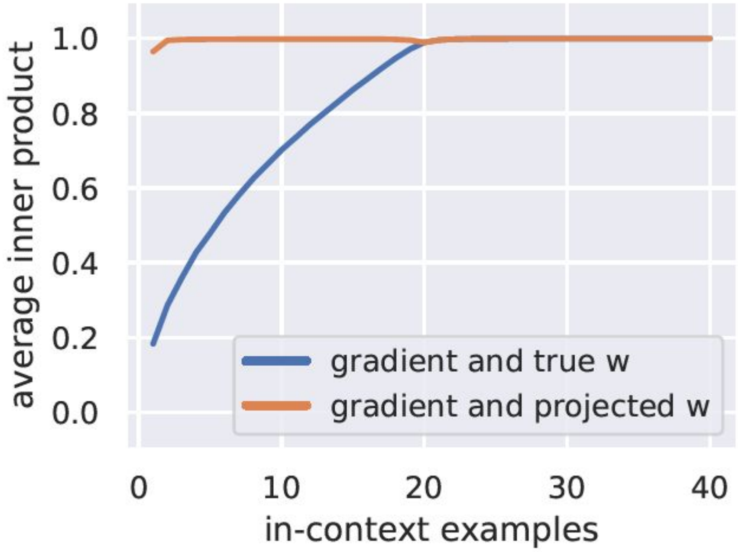
λ : the distance of the query input from origin.

Pick random unit vector x , evaluate $\hat{f}_{w,x_{1:k}}(\lambda x)$ as vary λ .



Visualization Along a Random Direction

Local Correctness



$$proj_{x_{1:k}}(w) = w, when\ k \geq d$$

The inner product between the gradient and $proj_{x_{1:k}}(w)$

Extrapolating Beyond the Training Distribution

Notations

$D_{\mathcal{F}}^{train}$: distribution of functions used during training

$D_{\mathcal{X}}^{train}$: corresponding distribution of prompt training inputs

$D_{\mathcal{F}}^{test}$: distribution of functions sampled during inference

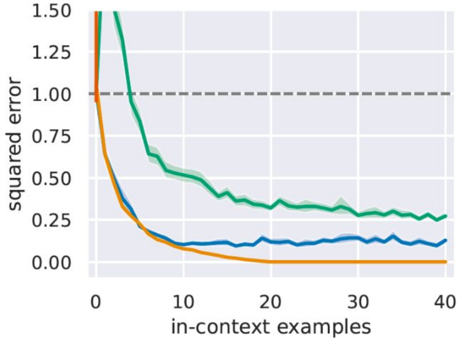
$D_{\mathcal{X}}^{test}$: corresponding distribution of prompt test inputs

D_{query}^{test} : query is sampled from $D_{\mathcal{X}}^{test}$, but potentially dependent on the rest of in-context inputs $x_1, \cdots, x_k$. Remember before, we create a set of random inputs $x_1, \cdots, x_{k+1}$ drawn i.i.d from $D_{\mathcal{X}}$.

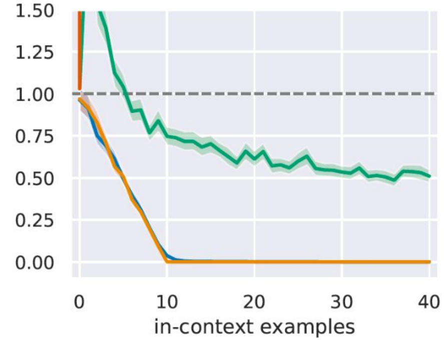
Two different distribution shift:

- Prompt train inputs and prompt test inputs are from different distribution: $D_{\mathcal{X}/\mathcal{F}}^{train} \neq D_{\mathcal{X}/\mathcal{F}}^{test}$
- Mismatch between in-context examples and the query input: $D_{query}^{test} \neq D_{\mathcal{X}}^{test}$

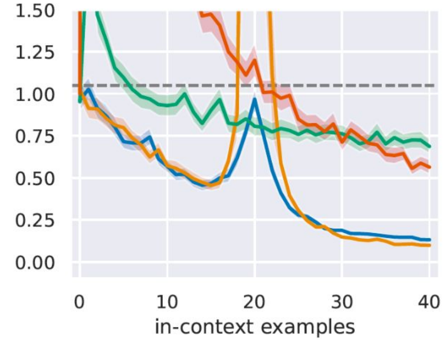
In-context Learning on Out-of-distribution Prompts



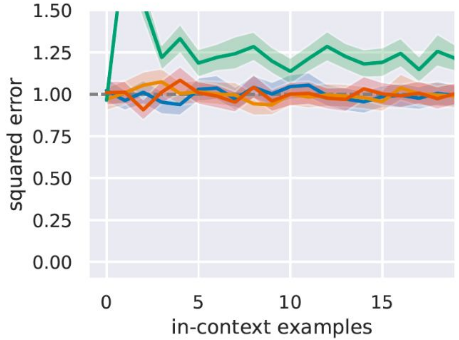
(a) skewed covariance



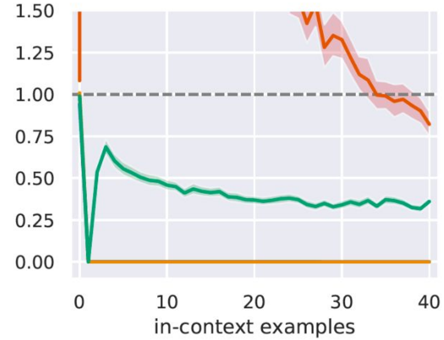
(b) $d/2$-dimensional subspace



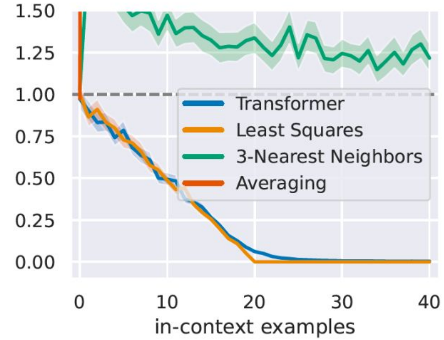
(c) noisy output



(d) orthogonal query



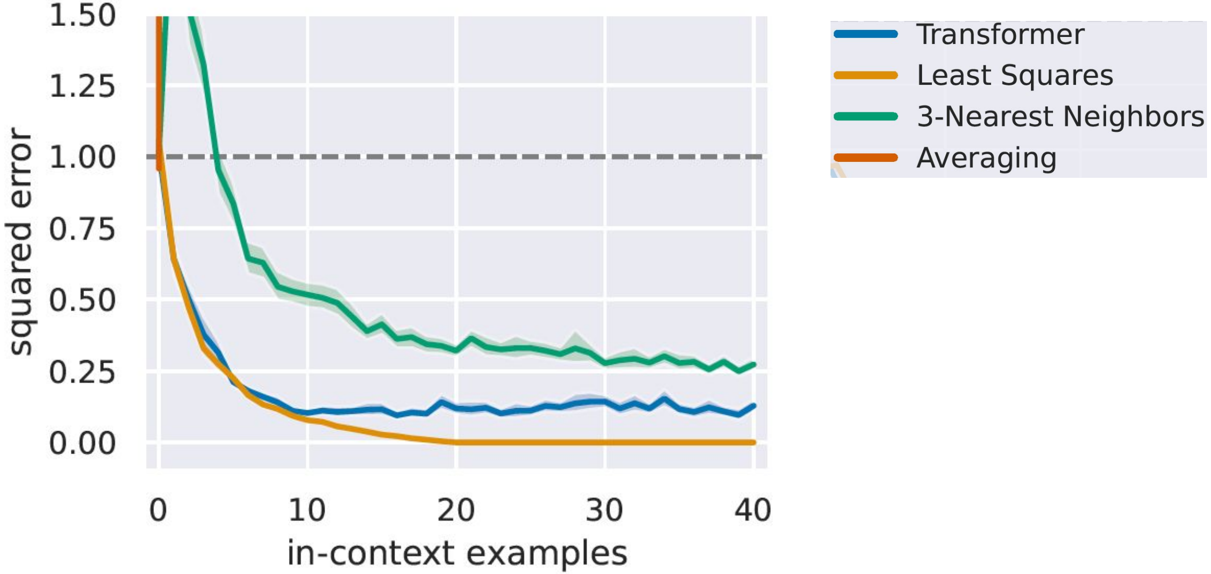
(e) query matches in-context example



(f) different orthants

Skewed Covariance

Not perfectly robust but still relatively well



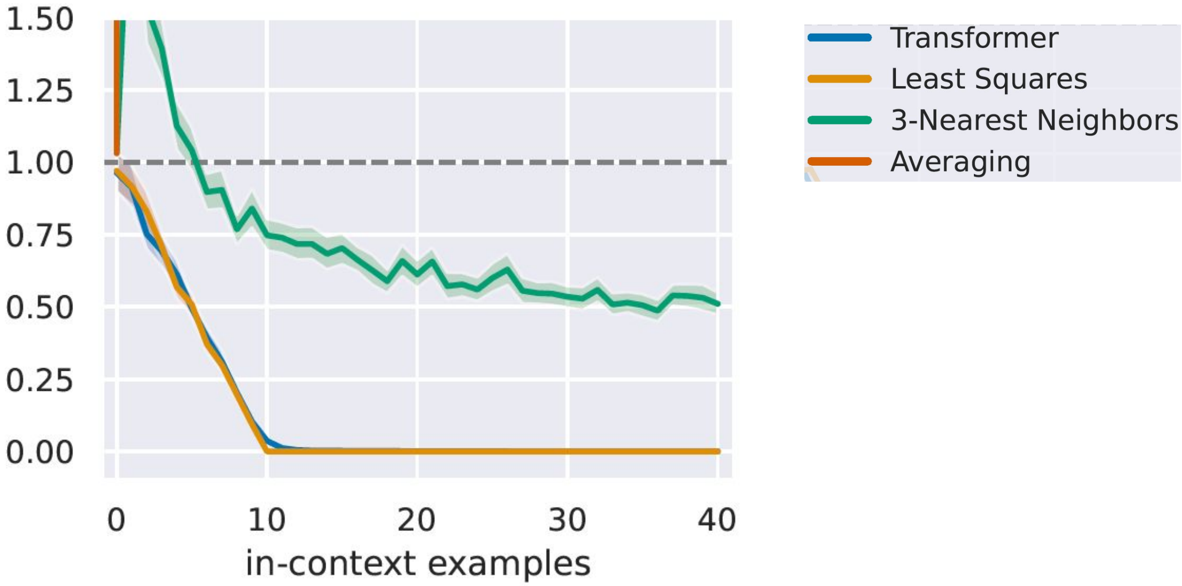
Sample prompt from $N(0, \Sigma)$

USC

Local-dimensional Subspace

d/2-dimensional Subspace

Encodes a valid orthogonalization procedure for these inputs.

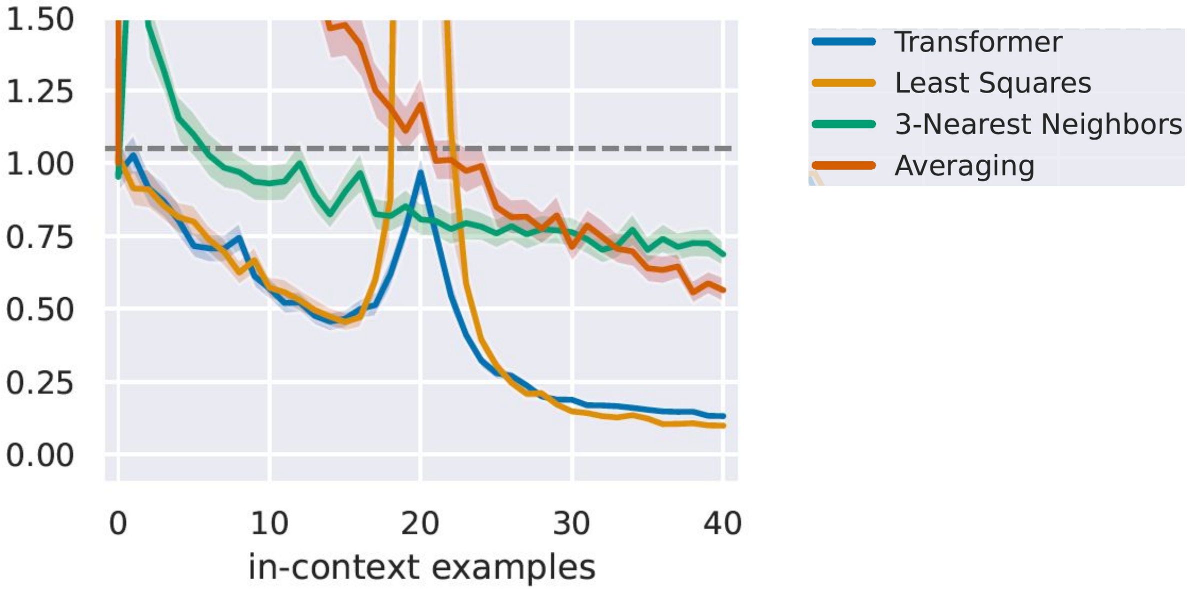


Sample prompt from a random 10 dimensional subspace

Noisy Output

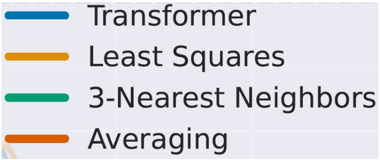
$$w^T x_i + \epsilon_i, \epsilon_i \sim N(0,1)$$

Train on noiseless data, evaluate with noisy linear functions.

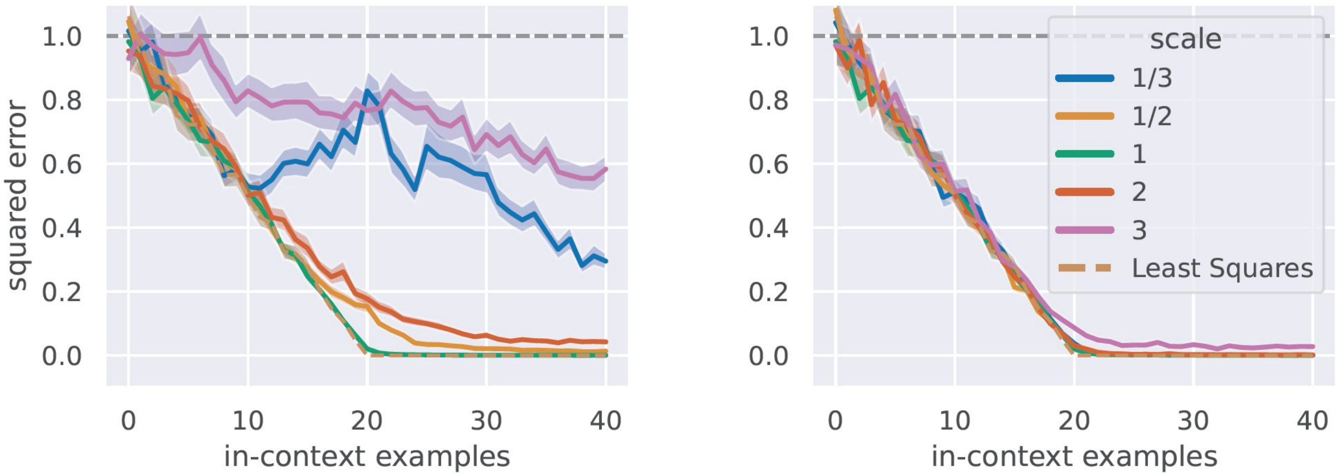


Sample prompt with noisy

Prompt Scale



Relatively robust for scaling weight,
not as robust for scaling prompt.



(a) scaled x, Transformer

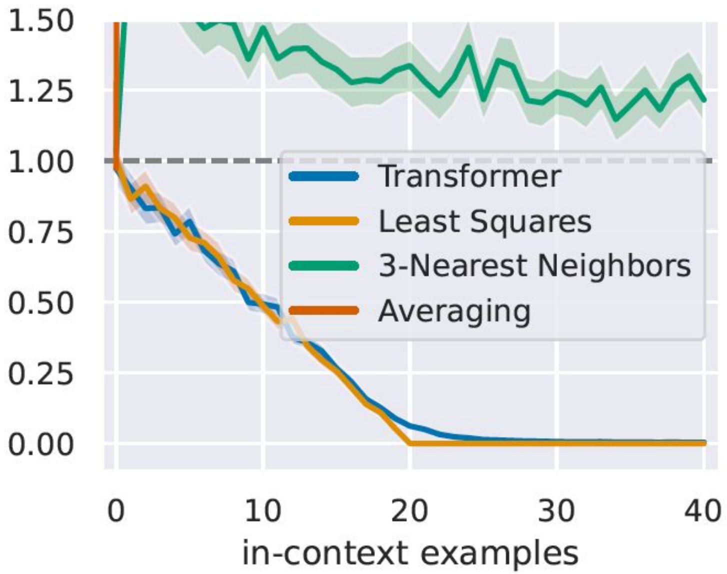
(b) scaled w, Transfomer

Sample prompt with different scale

USC

Different Orthants

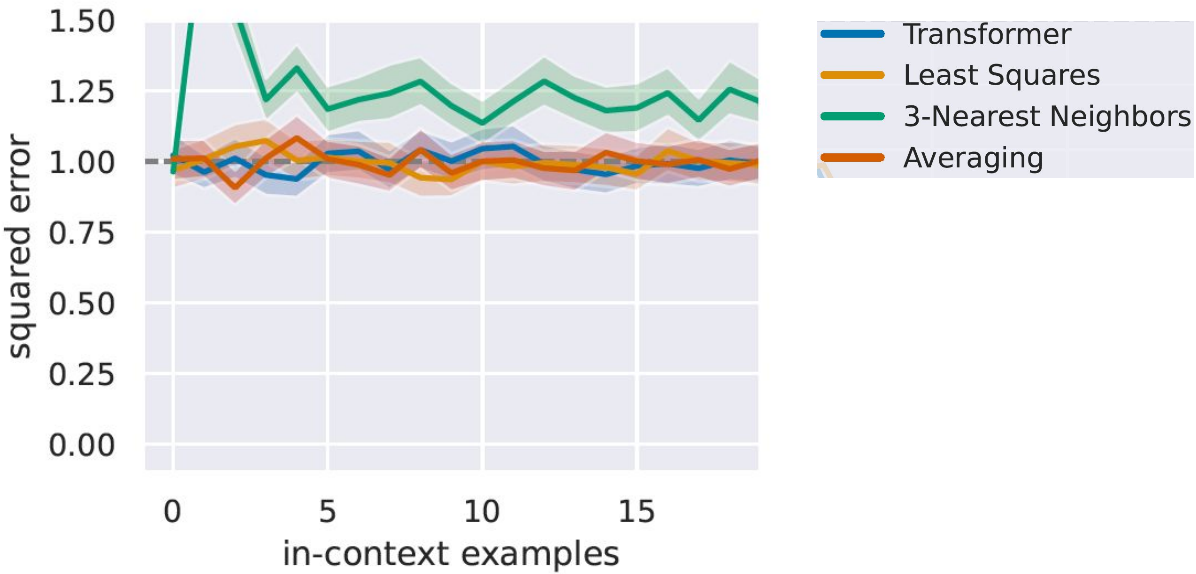
Not affected by the mismatch between in-context and query inputs, closely match performance of least squares.



Fix the sign of each coordinate to be positive or negative for all in-context inputs x_i

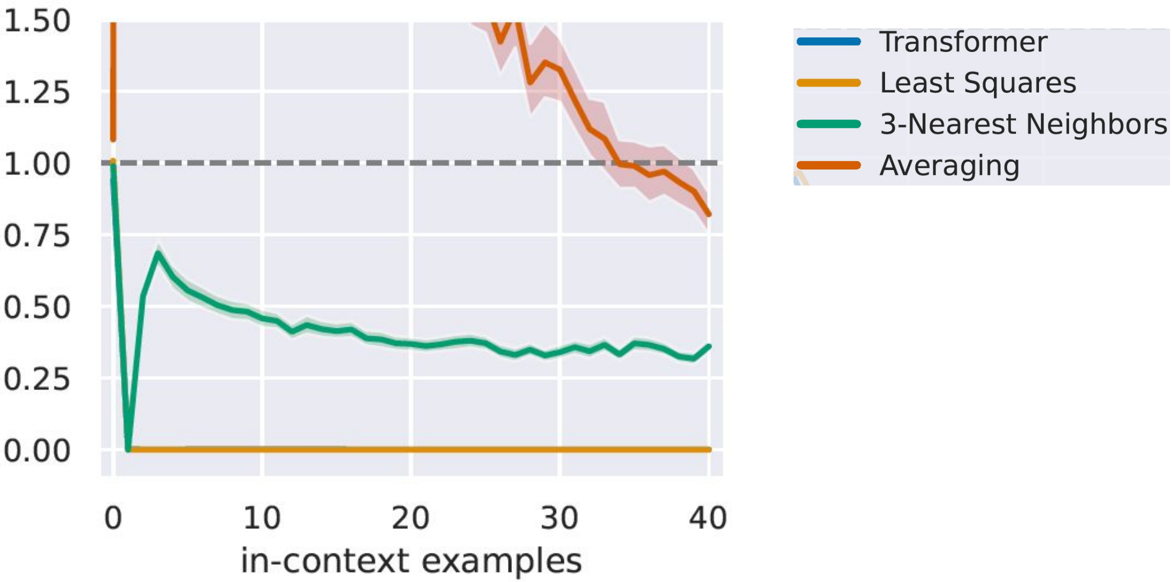
USC

Orthogonal Query



Sample the query from the subspace orthogonal to the subspace spanned by in-context inputs.

Query Matches In-context Example



Choose the query input from one of the in-context examples inputs uniformly at random

USC

More Complex Function Classes

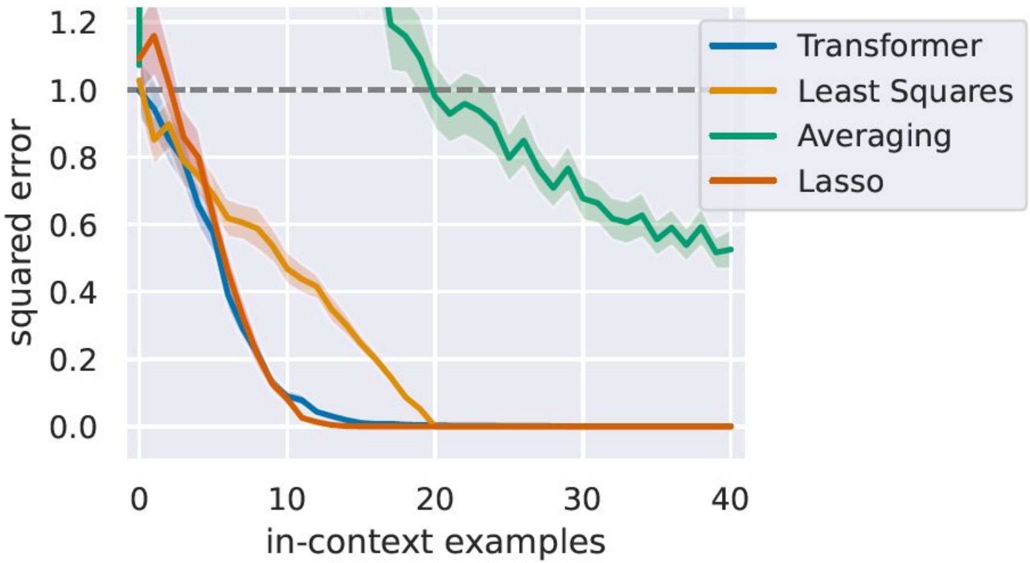
Sparse Linear Functions

$f(x) = w^T x, w \in \mathbb{R}$

zero out all but s coordinates of w uniformly at random

$x_i, x_{query} \sim N(0, I_d), w_i \sim N(0, I_d)$

*L1 regularization as a proxy for L0



A Transformer trained on prompts generated using sparse linear functions can in-context learn this class, with error decreasing at a rate similar to Lasso

USC

Decision trees

f is a binary tree with depth 4, the threshold is 0.

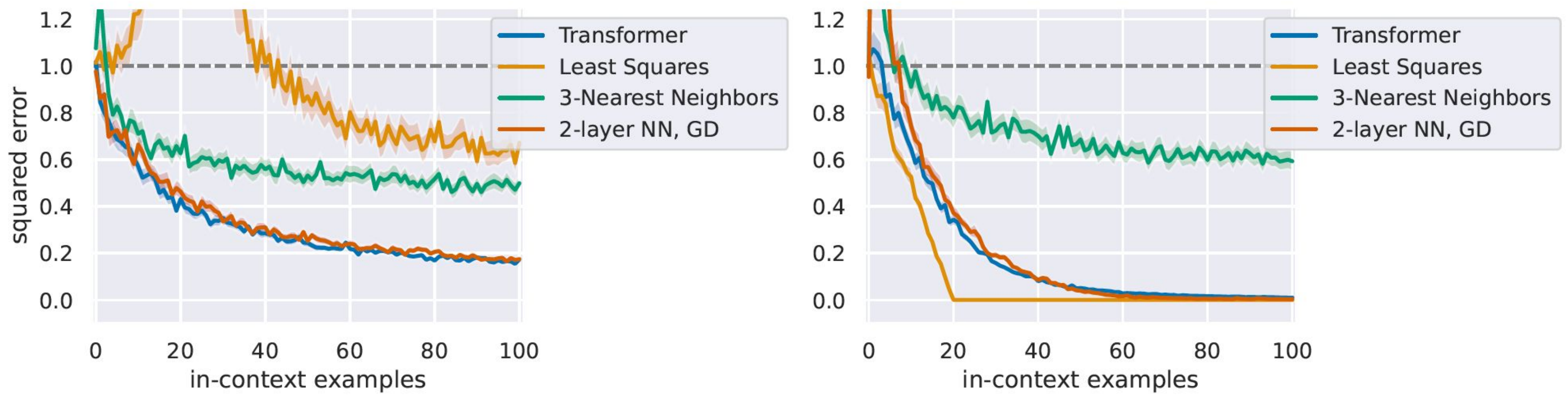
$$x_i, x_{query} \sim N(0, I_d)$$
$$non\ leaf\ nodes \sim \{1, \cdots, d\}$$
$$leaf\ nodes \sim N(0, 1)$$

A Transformer trained on prompts generated using random decision trees can in-context learn this class, which better performs than greedy tree learning or tree boosting.

USC

2-layer Neural Networks

$$f(x) = \sum_{i=1}^{r} \alpha_i \sigma(w_i^T x) \qquad x_i, x_{query} \sim N(0, I_d), \alpha_i \sim N(0, 2/r), w_i \sim N(0, I_d)$$

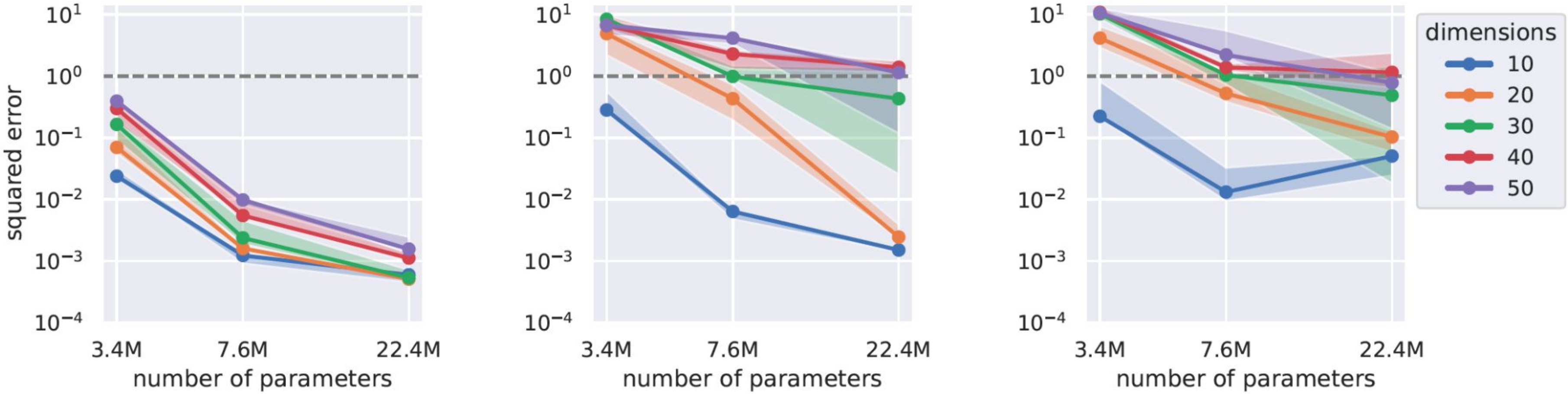


- A Transformer trained on prompts generated using random 2-layer ReLU neural networks can in-context learn this class.
- The model trained to in-context learn 2-layer neural networks is also able to in-context learn linear functions.

USC

Investigating Key Factors for In-context Learning

Problem Dimension and Capacity



(a) Standard (b) Different orthants (c) Skewed covariance

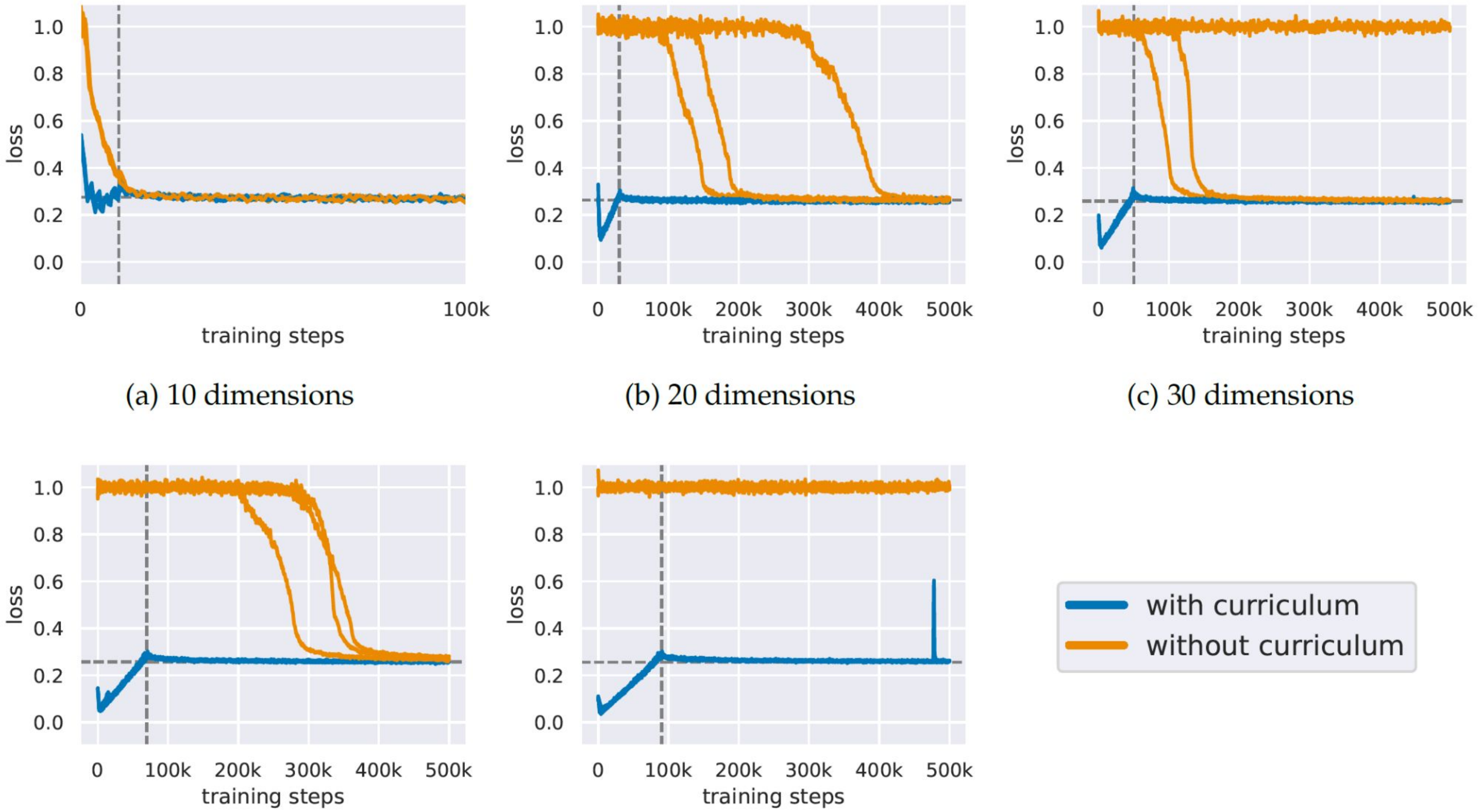
Consider models with fewer parameters,
train for different dimensional problems.

USC

Loss Progression with Curriculum under Different Dimensions

Curriculum Learning: gradually increasing the complexity of the function class.

Speed up!



(a) 10 dimensions

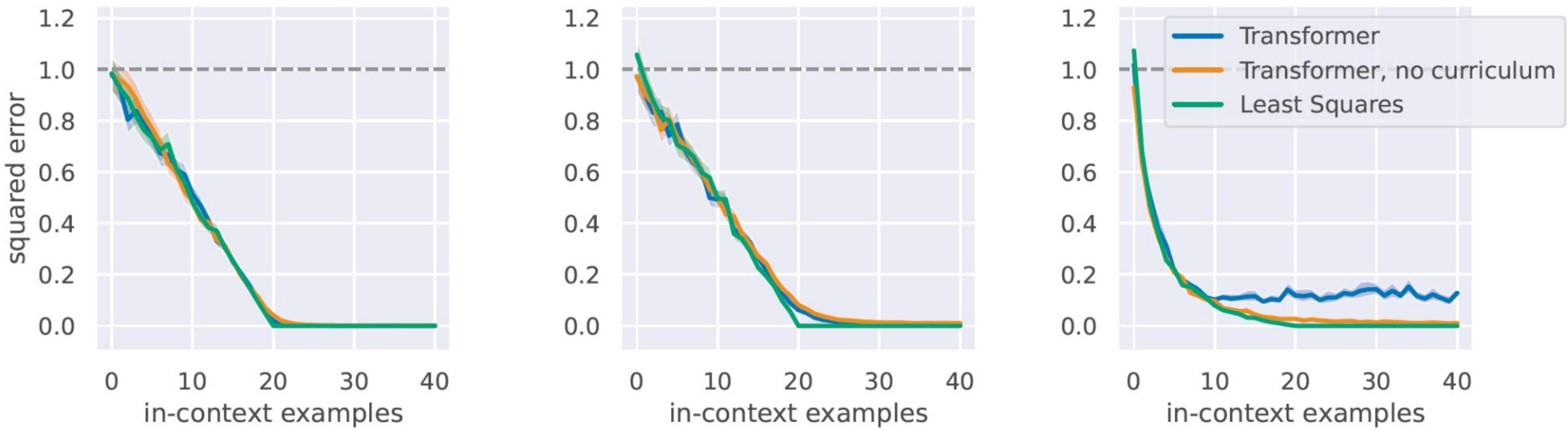
(b) 20 dimensions

(c) 30 dimensions

(d) 40 dimensions

(e) 50 dimensions

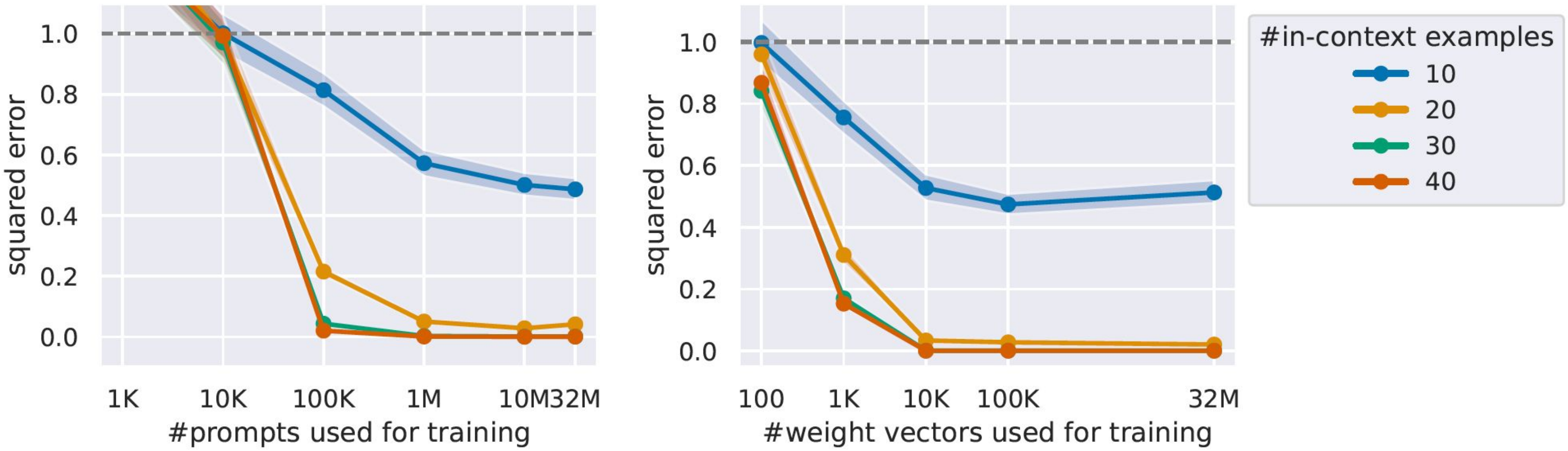
In-context Learning with Curriculum and Distribution Shift



(a) standard (b) different orthants (c) skewed

No major qualitative difference if we use curriculum or not

Number of Distinct Prompts or Functions Seen During Training



The amount of training data required is relatively small.

$$n_p = 100k, n_w = 1k$$

$$n_p = 1M, n_w = 10k$$

USC

Conclusions

Conclusions

Transformer models trained from scratch can in-context learn the class of linear functions, with performance comparable to the optimal least squares estimator, even under distribution shifts.

In-context learning can performs with some more complex functions: sparse linear functions, decision trees, and two-layer neural networks.

Capacity of model, number of in-context learning samples, and prompts / weight vectors used for training help perform better in-context learning. Curriculum can speed up the training process.

Transformers can encode complex learning algorithms that utilize in-context examples in a far-from-trivial manner. In fact, this is the case for standard Transformer architectures trained with standard optimization procedures.The extent to which such non-trivial in-context learning behavior exists in LLMs is still open.

Thanks for Listening

CSCI-699: Computational Perspectives on the Frontiers of Machine Learning

What Can Transformers Learn In-Context? A Case Study of Simple Function Classes

Presenter: Jingmin Wei