CS 839: Foundation Models
In-Context Learning

Fred Sala

University of Wisconsin-Madison

Oct. 5, 2023
Announcements

• Logistics:
  • Homework 1 due in 5 days---hopefully you’ve gotten started 😊

• Class roadmap:

<table>
<thead>
<tr>
<th>Thursday Oct. 5</th>
<th>In-Context Learning: Practice and Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday Oct. 10</td>
<td>Fine-Tuning, Specialization, Adaptation</td>
</tr>
<tr>
<td>Thursday Oct. 12</td>
<td>Training</td>
</tr>
<tr>
<td>Tuesday Oct. 17</td>
<td>RLHF</td>
</tr>
<tr>
<td>Thursday Oct. 19</td>
<td>Data</td>
</tr>
</tbody>
</table>
Outline

• **Back to In-Context Learning**
  • Basic idea, two ways of thinking about ICL, metalearning

• **Analysis and Theory**
  • Setup, learning simple function classes, implicit training, existence, learning results

• **Prompting Review**
  • Everything we’ve talked about so far
Outline

• **Back to In-Context Learning**
  • Basic idea, two ways of thinking about ICL, metalearning

• **Analysis and Theory**
  • Setup, learning simple function classes, implicit training, existence, learning results

• **Prompting Review**
  • Everything we’ve talked about so far
Reminder: **In-context learning**

**Also called few-shot:** but *sometimes* means fine-tune on this dataset, then prompt

**In-context learning:** do not fine-tune. Model weights unchanged.
• Everything happens in forward pass

Weng / SST

Dong et al, ‘23
ICL: Two ways to think about it

**One way** to think about few-shot is recovering some *fixed* model
- I.e., sentiment analysis model
- Here: goal of few-shot examples is to *activate* this model

**Other way**: be able to learn a function in a function class
- Learn $w^Tx$ for any $w$
ICL: Various ways to think about it

Other way: be able to learn a function in a function class

• Learn $w^Tx$ for any $w$

• Note: this is **metalearning**
  • Learn a transformers-based model that can then learn other functions
  • Traditionally a hard bilevel optimization problem

Hospedales et al ’20
ICL: Various ways to think about it

**Other way**: be able to learn a function in a function class

• Learn $w^T x$ for any $w$
• Metalearning approach

• **Q**: How is this possible?
  • Note that we’re not changing the trained model, so has to itself do a full training (inner-level) procedure in a forward pass
  • Perhaps doable with a big model?
Break & Questions
Outline

• **Back to In-Context Learning**
  • Basic idea, two ways of thinking about ICL, metalearning

• **Analysis and Theory**
  • Setup, learning simple function classes, implicit training, existence, learning results

• **Prompting Review**
  • Everything we’ve talked about so far
What Can Transformers Learn?

Can study from theoretical or empirical points of view

Theoretical setup:

\[
\arg\min_0 \mathbb{E}_{x_1, \ldots, x_n \sim p(x)} \left[ \sum_{i=1}^{n} \mathcal{L}(f(x_i), T_\theta([x_1, f(x_1) \ldots, x_i])) \right]
\]

Akyurek et al ‘23

Note: trained model T is the in-context learner: it’s given a “dataset” in a prompt, plus a test point
What Can Transformers Learn?

Theoretical setup:

$$\arg\min_{\theta} \mathbb{E}_{x_1, \ldots, x_n \sim p(x)} \left[ \sum_{i=1}^{n} \mathcal{L}(f(x_i), T_\theta([x_1, f(x_1), \ldots, x_i])) \right]$$

Here, during training we’re not learning one function \( f \)

- We’re training an ICL to “learn” new functions!
  - At test time!
  - This is the metalearning idea
What Can Transformers Learn? **Linear Case**

Let’s try learning linear functions

\[ f(x) = w^T x \]

Loss function: squared loss

\[ L(y, f(x)) = (w^T x - y)^2 \]

Regularized empirical risk

\[ \sum_i (w^T x_i - y_i)^2 + \lambda \|w\|^2_2 \]
Let’s try learning linear functions $f(x) = w^T x$

How do we learn a function like $f$?

Two ways:

1. **Closed-form:**
   $$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

2. **Gradient descent:**
   $$w_t = w_{t-1} - \alpha \frac{\partial}{\partial w}(\text{Loss})$$
   $$w_{t-1} - 2\alpha (x w_{t-1}^T x - yx + \lambda w)$$
What Can Transformers Learn? Implicit GD

Note: the ICL model can’t learn some specific w

- Different for each input/prompt...
- But, what if it can learn the **procedure** that generates a solution?

How can we tell? Possibility result from Akyurek et al ’23

**Theorem 1.** A transformer can compute Eq. (11) (i.e. the prediction resulting from single step of gradient descent on an in-context example) with constant number of layers and $O(d)$ hidden space, where $d$ is the problem dimension of the input $x$. Specifically, there exist transformer parameters $\theta$ such that, given an input matrix of the form:

$$H^{(0)} = \begin{bmatrix} \ldots & 0 & y_i & 0 & 0 & \ldots \end{bmatrix},$$  \hspace{1cm} (12)

the transformer’s output matrix $H^{(L)}$ contains an entry equal to $w^T x_n$ (Eq. (11)) at the column index where $x_n$ is input.

See also Oswald et al
What Can Transformers Learn? Closed Form

What about the closed-form approach?

• Have to invert $X^T X + \lambda I$

• Note: these are sums of rank 1-terms + diagonal, can run it sequentially with Sherman-Morrison

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}.$$ 

• Here too,

**Theorem 2.** A transformer can predict according to a single Sherman–Morrison update:

$$w' = (\lambda I + x_i x_i^T)^{-1} x_i y_i = \left( \frac{I}{\lambda} - \frac{1}{\lambda} x_i x_i^T \frac{I}{\lambda} \right) x_i y_i$$

(14)

with constant layers and $O(d^2)$ hidden space. More precisely, there exists a set of transformer parameters $\theta$ such that, given an input matrix of the form in Eq. (12), the transformer’s output matrix $H^{(l)}$ contains an entry equal to $w^T x_n$ (Eq. (14)) at the column index where $x_n$ is input.
Does this Work?

These existence results show this is possible...

• I.e., there exist weights that produce this behavior
But does training actually get us these?

• A: Yes
What About Learning?

Is it possible to show that \textbf{training} a transformer produces this behavior?

\textbf{A: Yes: Ahn et al ’23}

\textbf{Model choice:} multi-layer linear self-attention

\textbf{Theorem 4.} Assume that $x^{(i)} \overset{iid}{\sim} N(0, \Sigma)$ and $w_k \sim N(0, \Sigma^{-1})$, for $i = 1...n$, and for some $\Sigma > 0$. Consider the optimization of in-context loss for a $k$-layer transformer with the parameter configuration in Eq. (9) given by:

$$\min_{\{A_i\}_{i=0}^{k}} f\left([A_i]_{i=0}^{k}\right).$$

Let $S \subset \mathbb{R}^{(k+1)\times d\times d}$ be defined as follows: $A \in S$ if and only if for all $i \in \{0, \ldots, k\}$, there exists scalars $a_i \in \mathbb{R}$ such that $A_i = a_i\Sigma^{-1}$. Then

$$\inf_{A \in S} \sum_{i=0}^{k} \|\nabla_{A_i} f(A)\|_F^2 = 0,$$

where $\nabla_{A_i} f$ denotes derivative wrt the Frobenius norm $\|A_i\|_F$. 
Break & Questions
Outline

• Back to In-Context Learning
  • Basic idea, two ways of thinking about ICL, metalearning

• Analysis and Theory
  • Setup, learning simple function classes, implicit training, existence, learning results

• Prompting Review
  • Everything we’ve talked about so far
Review: Few-Shot Choices

Examples/structure affect performance:
1. Prompt **format** (affects everything)
2. **Choice** of examples
3. **Order** of examples (permutation)

Zhao et al '21
Review: Choice of Examples

How to pick appropriate examples in few-shot?

- **Note:** only a “small” number of examples can be shown, unlike in supervised learning.

Many options. Sampling:
- Liu et al, ‘21: kNN in embedding space (semantic similarity)
- Su et al, ‘22: Encourage diversity in embeddings
- Diao et al, ‘23: “Active prompting”
2. Choice of Examples

How to pick appropriate examples in few-shot?

**Note:** only a “small” number of examples can be shown, unlike in supervised learning.

Many options. Sampling:

- Liu et al, ‘21: kNN in embedding space (semantic similarity)
- Su et al, ‘22: Encourage diversity in embeddings
- Diao et al, ’23: “Active prompting”
2. Choice of Examples

How to pick appropriate examples in few-shot?

• **Note**: only a “small” number of examples can be shown, unlike in supervised learning.

Many options. Sampling:

• Liu et al, ‘21: kNN in embedding space (semantic similarity)
• Su et al, ‘22: Encourage diversity in embeddings
• Diao et al, ’23: “Active prompting”
Performing complex reasoning is hard. Help the model:

**Standard Prompting**

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The answer is 27. ❌

**Chain-of-Thought Prompting**

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✔️

Wei et al ‘22
Chain-of-Thought: Generalizations

How do we really “reason”?  
• Not really by sampling a bunch of chains...

Yao et al ‘23
Tools: Program-aided LMs

Use external tools:
- Python interpreter

- How? *Interleave* the text explanations in CoT steps with lines of Python code

- LMs can already output code
  - Just need to *prompt* the right way

Gao et al ‘23
Bibliography

Thank You!