CSC2532: Statistical Learning Theory Winter 2024

Project Report: Transformers As Statisticians

Ben Agro

1 Introduction

Part of the success of large language models (implemented with transformers [1]) can be attributed to their apparent ability to learn new tasks from in-context examples, as opposed to traditional models, which require explicit training for a specific task. Towards a formal understanding of ICL, in this report, we distill two main ideas from the recent paper "Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection" (TaS) [2]. Section 2 provides preliminaries on transformers and in-context learning. **Main idea 1:** Section 3 discusses how transforms can perform *in-context gradient descent*, which is the underlying mechanism by which transformers can implement the many in-context algorithms, including ridge regression, generalized linear models, and lasso. **Main idea 2:** Section 4 describes one mechanism by which transformers can select the appropriate in-context learning algorithm based on its context. TaS owes its namesake to this mechanism because it is similar to how a statistician can look at data and choose the appropriate model.

2 Transformers and In-Context Learning

Transformers:[1]

Denote the transformer input sequence as a matrix $\boldsymbol{H} = [\boldsymbol{h}_1, \dots, \boldsymbol{h}_n] \in \boldsymbol{R}^{D \times N}$, where each $\boldsymbol{h}_i$ is a *token* in the sequence. We will focus on "encoder-mode" transformers, which have multiple layers, and each consisting of a *self-attention* (SA) layer and a residual *multi-layer perceptron* (MLP) layer. Differing from the popular transformer architecture, TaS replaces the softmax used in SA with a (normalized) ReLU activation function $\sigma(t) = \text{ReLU}(t) = \max(0, t)$;

$$\tilde{\boldsymbol{h}}_i = [\text{SA}_{\boldsymbol{\theta}}(\boldsymbol{H})]_i = \boldsymbol{h}_i + \sum_{m=1}^{M} \frac{1}{N} \sum_{j=1}^{N} \sigma(\langle \boldsymbol{Q}_m \boldsymbol{h}_i, \boldsymbol{K}_m \boldsymbol{h}_j \rangle) \boldsymbol{V}_m \boldsymbol{h}_j, \tag{2.1}$$

where M is the number of attention heads, and they do not include layer normalization.[2] The residual MLP layer takes the form $\text{MLP}_{\boldsymbol{W}_1, \boldsymbol{W}_2}(\boldsymbol{H}) = \boldsymbol{H} + \boldsymbol{W}_2 \sigma(\boldsymbol{W}_1 \boldsymbol{H})$, where $\boldsymbol{W}_1, \boldsymbol{W}_2$ are learned weight matrices.

In-Context Learning: An ICL instance consists of tokens of the form

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N & \boldsymbol{x}_{N+1} \\ y_1 & y_2 & \dots & y_N & 0 \\ \boldsymbol{p}_1 & \boldsymbol{p}_2 & \dots & \boldsymbol{p}_N & \boldsymbol{p}_{N+1} \end{bmatrix} \in \mathbb{R}^{D \times (N+1)}, \quad \boldsymbol{p}_i = \begin{bmatrix} \boldsymbol{0}_{D-(d+3)} \\ 1 \\ \mathbb{I}\{i < N+1\} \end{bmatrix}, \tag{2.2}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d \overset{\text{iid}}{\sim} \mathsf{P}_{\boldsymbol{x}}$ are the input features and $y_i \in \mathbb{R} \sim \mathsf{P}_{y|\boldsymbol{x}_i}$ are the labels. Here $t_i = \mathbb{I}\{i < N+1\}$ indicates the "training"[3] examples; the test example is at token index $N+1$. Each ICL instance may have a different data distribution, and the goal is to learn/construct a fixed transformer that can reliably predict the hidden test label y_{N+1} on large set of data distributions. Note that we include the $\boldsymbol{0}_{D-(d+3)}$ in the input token (D to be specified) to serve as a *placeholder* for the transformer to save intermediate results. Moreover, we can use the 1 to implement constant offsets.

[1] We assume the reader is already familiar with transformers [1].

[2] These changes allow for convenient constructions of transformers in the proofs of TaS, but they are a minor detail as prior work has shown an equivalence between normalized ReLU and softmax activations [3, 4].

[3] In ICL, the model weights are not updated with this data; the prediction is made in the inference pass of the transformer.

3 In-Context Learning via Gradient Descent

TaS proves that transformers can implement a broad class of standard machine learning algorithms in context, including least-squares, ridge regression, generalized linear models, and lasso. The linchpin of these results is the construction of a transformer that can perform *in-context gradient descent* (GD).

Desiderata: Let $\hat{L}_N(\boldsymbol{w}) := \frac{1}{N}\sum_{j=1}^{N} \ell(\boldsymbol{w}^T \boldsymbol{x}_j, y_j)$ be the empirical risk to minimize via GD:

$$\boldsymbol{w}_{GD}^{t+1} := \boldsymbol{w}_{GD}^{t} - \eta \nabla \hat{L}_N(\boldsymbol{w}_{GD}^{t}), \tag{3.1}$$

for some loss function ℓ. We want to construct a transformer with $L+1$ layers that can approximately implement GD, meaning that $\boldsymbol{h}_i^l, \forall l \in \{1, \ldots, L\}$ will contain weights $\hat{\boldsymbol{w}}^l$ that approximate the GD weights $\boldsymbol{w}_{GD}^l$ on the training examples, and the final tokens will contain the predicted labels $\hat{y}_i = \langle \hat{w}^L, \boldsymbol{x}_i \rangle$. Further, the $||\hat{w}^l - \hat{w}_{GD}^l||$ increases linearly in t, and so does the test error $|\hat{y}_{N+1} - y_{N+1}|$ in L.

Transformer Construction: The idea is that the first L layers will each approximate one step of GD, and that the $(L+1)$-st layer will output the predicted label. Let us start by constructing a single GD step with an attention layer. We assume that the previous layers have an approximation of the weights $\hat{\boldsymbol{w}}^t$ (or zero for the initial tokens). We need to assume that $\partial_s \ell(s,t)$ is *approximable by sum of M ReLUs*, which means that there are some $c_m, a_m, b_m, d_m, \forall m \in \{1, \ldots, M\}$ such that

$$\sup_{(s,t)} |\sum_{m=1}^{M} c_m \sigma(a_m s + b_m t + d_m) - \partial_s \ell(s,t)| \leq \epsilon, \tag{3.2}$$

which is true for a broad class of functions [2]. Note that $\nabla \hat{L}_N(\boldsymbol{w}) = \frac{1}{N}\sum_{j=1}^{N} \partial_s \ell(\boldsymbol{w}^T \boldsymbol{x}_j, y_j)^T \boldsymbol{x}_j$, so that the gradient of the empirical risk and gradient update step based on the training data can be written as:

$$\frac{1}{N}\sum_{m=1}^{M}\sum_{j=1}^{N} c_m \sigma(a_m \langle \boldsymbol{w}, \boldsymbol{x}_j \rangle + b_m y_j + d_m) \boldsymbol{x}_j t_j = \nabla \hat{L}_N(\boldsymbol{w}) + O(\epsilon) \tag{3.3}$$

$$\hat{\boldsymbol{w}}^l - \frac{1}{N}\sum_{m=1}^{M}\sum_{j=1}^{N} \eta c_m \sigma(a_m \langle \hat{\boldsymbol{w}}^l, \boldsymbol{x}_j \rangle + b_m y_j + d_m) \boldsymbol{x}_j t_j = \hat{\boldsymbol{w}}^l - \eta \nabla \hat{L}_N(\hat{\boldsymbol{w}}^l) + O(\epsilon). \tag{3.4}$$

We notice the similarity between the LHS of the above expression and the self-attention operation eq. (2.1). Indeed, simple algebra (Proposition C.2 in TaS) shows that for input tokens of the form in eq. (2.2) with the $\hat{\boldsymbol{w}}^l$ stored as an intermediate output in the placeholder slot, one can choose $\boldsymbol{Q}_m, \boldsymbol{K}_m, \boldsymbol{V}_m$ to obtain an SA update that equals the LHS of eq. (3.4). A natural question is whether chaining these approximate GD updates together results in a good approximation of the final weights. One can show (Lemma 14 in TAS) that if we additionally assume ℓ is convex in the first argument, then the weight-space error is linear in l: $||\hat{\boldsymbol{w}}^l - \hat{\boldsymbol{w}}_{GD}^l|| = O(l\epsilon)$. Thus, we can stack L of these attention layers (setting the residual MLPs to output identity) to build a transformer that can approximate L steps of GD with weight-space error in $O(L\epsilon)$. Additionally, using the fact that $\sigma(t) - \sigma(-t) = t$ for $t > 0$, it is easy to construct a 2-head attention layer that implements $\langle \hat{\boldsymbol{w}}^L, \boldsymbol{x}_i \rangle$ (each head implements one of the σ's), and with the linear error bound on $\hat{\boldsymbol{w}}^L$, the prediction error is also linear in L.

4 In-Context Algorithm Selection

Just as a statistician can look at data and choose the appropriate model, TaS shows that transformers can implement *in-context algorithm selection*. We will investigate one of the two proposed mechanisms proposed in for in-context algorithm selection; *post-ICL validation*

Desiderata: Assume we have a validation loss $\hat{L}_{val}(f) = \frac{1}{N_{val}}\sum_{i=1}^{|\mathcal{D}|_{val}} \ell(f(\boldsymbol{x}_i), y_i)$, ℓ is approximable by a sum of ReLUs. We split the in-context examples into a training set $\mathcal{D}_{train}$ (labelled $t_i = 1$) and a validation set $\mathcal{D}_{val}$ (labelled $t_i = -1$). The goal is to build a transformer that (1) *Inference:* implements K in-context algorithms on $\mathcal{D}_{train}$, $\{f_1, \ldots, f_K\}$ and outputs their predictions on all in-context examples, (2) *Evaluation:* approximates the loss $\tilde{L}_{val}(f_k) \approx \hat{L}_{val}(f_k)$ of each f_k on $\mathcal{D}_{val}$, (3) *Selection:* for each x_i

including the test input ($t_i = 0$), outputs a convex combination $\hat{f}$ of the predictions $\{f_1(x_i), \ldots, f_k(x_i)\}$ that achieves near the minimum validation loss.

Transformer construction : *To implement (1)*, we note that transformers can be *combined in parallel*, meaning that if we have K transformers (possibly with a different number of layers or heads), each implementing a sequence-to-sequence function $P_k : \mathbb{R}^{D\times(N+1)} \to \mathbb{R}^{D\times(N+1)}$,

$$\mathrm{TF}_{\boldsymbol{\theta}_k} : \boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{H}^{(0)} \\ \boldsymbol{H}_k^{(1)} \end{bmatrix} \to \begin{bmatrix} \boldsymbol{H}^{(0)} \\ P_k(\boldsymbol{H}_k) \end{bmatrix}, \forall k \in \{1, \ldots, K\}, \tag{4.1}$$

then, there is a transformer that implements all K sequence-to-sequence functions in parallel:

$$\mathrm{TF}_{\boldsymbol{\theta}} : \begin{bmatrix} \boldsymbol{H}^{(0)} \\ \boldsymbol{H}_1^{(1)} \\ \cdots \\ \boldsymbol{H}_K^{(1)} \end{bmatrix} \to \begin{bmatrix} \boldsymbol{H}^{(0)} \\ P_1(\boldsymbol{H}_1) \\ \cdots \\ P_K(\boldsymbol{H}_K) \end{bmatrix} \tag{4.2}$$

In section section 3, we discussed that transformers can implement many in-context algorithms, so we can stack K in parallel to implement a single transformer that outputs $[*, f_1(\boldsymbol{x}_i), \ldots, f_K(\boldsymbol{x}_i), \mathbf{0}_{K+1}]$ in the placeholder slots of $\boldsymbol{h}_i$, where we use $*$ to hide the inputs and labels.

To implement (2), we construct a single attention layer that ingests $[*, f_1(\boldsymbol{x}_i), \ldots, f_K(\boldsymbol{x}_i), \mathbf{0}_{K+1}]$ and outputs $[*, f_1(\boldsymbol{x}_i), \ldots, f_K(\boldsymbol{x}_i), \tilde{L}_{val}(f_1), \ldots \tilde{L}_{val}(f_K), 0]$ in the placeholder slots of all $\boldsymbol{h_i}$. Mirroring our discussion of gradient descent in section 3, because ℓ is approximable by a sum of ReLUs, we can construct a single attention layer that implements $\tilde{L}_{val} \approx \hat{L}_{val}$ using the validation indicator $t_i = -1$.

Finally, *to implement (3)*, we construct a three-layer *selection transformer* mapping

$$[*, f_1(\boldsymbol{x}_i), \ldots, f_K(\boldsymbol{x}_i), \tilde{L}_{val}(f_1), \ldots \tilde{L}_{val}(f_K), 0] \to [*, f_1(\boldsymbol{x}_i), \ldots, f_K(\boldsymbol{x}_i), \tilde{L}_{val}(f_1), \ldots \tilde{L}_{val}(f_K), \hat{f}(\boldsymbol{x}_i)], \tag{4.3}$$

where $\hat{f}$ is a convex combination of the f_k that are within γ of the minimum validation loss:

$$\hat{f} = \sum_{k=1}^{K} \lambda_k f_k, \text{ s.t } \lambda_1, \ldots, \lambda_K \geq 0, \sum_{k=1}^{K} \lambda_k = 1, \text{ and } \lambda_k > 0 \text{ only if } \tilde{L}_{val}(f_k) \leq \min_{k^* \in [K]} \tilde{L}_{val}(f_{k^*}) + \gamma. \tag{4.4}$$

Define $c_k = \sum_{p \neq k} \sigma(\tilde{L}_{val}(f_k) - \tilde{L}_{val}(f_p))$, and notice that if $c_k \leq \gamma$ then $\tilde{L}_{val}(f_k) \leq \min_{k^* \in [K]} \tilde{L}_{val}(f_{k^*}) + \gamma$. We construct the first layer of the selection transformer with a residual MLP (the attention layer implements identity) that maps from the LHS of eq. (4.3) to $[f_1(\boldsymbol{x}_1), \ldots, f_K(\boldsymbol{x}_i), c_1, \ldots, c_K, 0]$ which is straightforward to construct because c_k is a sum of ReLUs.

Next, let $u_k = \sigma(1 - \gamma^{-1} c_k)$, which is in $[0, 1]$ and $u_k > 0$ iff $c_k \leq \gamma$. We construct the second layer with an MLP mapping from the above intermediate tokens to $[f_1(\boldsymbol{x}_1), \ldots, f_K(\boldsymbol{x}_i), u_1, \ldots, u_K, 0]$

Finally, selecting $\lambda_1 = 1 - \sigma(1 - u_1)$, $\lambda_k = \sigma(1 - u_1 - \cdots - u_{k-1}) - \sigma(1 - u_1 - \cdots - u_k), \forall k \geq 2$ satisfies the conditions of eq. (4.4). We can re-arrange to get the following expression for $\hat{f}$ (letting $f_0 = f_{K+1} = 0$):

$$\hat{f} = \sum_{k=1}^{K+1} \sigma(1 - u_1 - \cdots - u_{k-1})(f_k(\boldsymbol{x}_i) - f_{k-1}(\boldsymbol{x}_i)) = \sum_{k=1}^{K+1} \sum_{w \in \{0,1\}} \sigma(1 - u_1 - \cdots - u_{k-1})(-2w + 1) f_{k-w}(\boldsymbol{x}_i). \tag{4.5}$$

Notice that this has a very similar form to the SA operation in eq. (2.1), so we can implement this with a single attention layer with $2(K+1)$ heads. Thus, the transformer will predict $\hat{f}(\boldsymbol{x}_{N+1})$ on the test example.

Stacking the (1) *inference*, (2) *evaluation*, and (3) *selection* layers gives us a transformer that is a "statistician", as it can dynamically select the appropriate in-context algorithm based on the context.

5 Conclusion

One could imagine how these simple mechanisms of in-context gradient descent and algorithm selection, operating in the high-dimensional representation space of a large transformer, could result in the observed and impressive emergent in-context learning behaviors of LLMs. However, while the TaS's empirical experiments show that a transformer *can* do these in-context tasks, they do not investigate if it is by the proposed mechanisms, leaving room for future work to understand if large language models are implementing this algorithm selection mechanism in practice.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] Y. Bai, F. Chen, H. Wang, C. Xiong, and S. Mei, "Transformers as statisticians: Provable in-context learning with in-context algorithm selection," *Advances in neural information processing systems*, vol. 36, 2024.

[3] M. Wortsman, J. Lee, J. Gilmer, and S. Kornblith, "Replacing softmax with relu in vision transformers," *arXiv preprint arXiv:2309.08586*, 2023.

[4] K. Shen, J. Guo, X. Tan, S. Tang, R. Wang, and J. Bian, "A study on relu and softmax in transformer," *arXiv preprint arXiv:2302.06461*, 2023.