Homework # 6: Implementation of Transformer-Based Few Shot Learning

We are interested in running experiments with Transformers applied to functional data. There are three papers we are focusing on, with associated code:

- For the linear attention examples, we are focused on this paper: https://arxiv.org/abs/2212.07677 and the GitHub for the code is here: https://github.com/google-research/self-organising-systems/tree/master/transformers_learn_icl_by_gd

- For the examples with softmax attention (traditional Transformer design), we are interested in this paper: https://arxiv.org/abs/2208.01066 and the GitHub for the code is here: https://github.com/dtsip/in-context-learning

- The MAML GitHub may also be useful: https://github.com/cbfinn/maml .

For this homework, it is ok to use/modify the above code to implement experiments.

Consider the following process for generating contextual data for a linear model: weights $w_m \in \mathbb{R}^{10}$ are drawn for context m as $w_m \sim \mathcal{N}(\mu, I_{10})$, where $\mu \in \mathbb{R}^{10}$ is a fixed mean vector. Covariates $x_i \in \mathbb{R}^{10}$ are drawn as $x_i \sim \mathcal{N}(0_{10}, I_{10})$. For contextual data $\mathcal{C}_m$ draw one weight vector w_m, as above. For a context of length N draw $x_{m,i}$, $i = 1, \ldots, N$ as above, and for each $x_{m,i}$ constitute a corresponding $y_{m,i} = w_m^T x_{m,i}$. The contextual data so drawn are represented as $\mathcal{C}_m = (x_{m,1}, y_{m,1}, \ldots, x_{m,N}, y_{m,N})$. Finally, draw a query associated with $\mathcal{C}_m$, $x_{m,N+1}$, and the model is to predict $y_{m,N+1}$ given $x_{m,N+1}$, based on the context $\mathcal{C}_m$. In the following experiments set μ as a vector of all values equal to one: $\mu = (1, \ldots, 1)^T$. In all experiments, consider the number of pairs in the context as $N = 5, \ldots, 20$ (examine performance of the methods over this range of N).

(a) Perform in-context learning based on MAML, and assume a linear model $f_w(x) = w^T x$, with model parameters w. Use MAML to learn good initialization parameters w_0, based on $\mathcal{C}_m = (x_{m,1}, y_{m,1}, \ldots, x_{m,N}, y_{m,N})$ for $m = 1, \ldots, M$, and then apply the model to predict $y_{M+1,N+1}$ for $\mathcal{C}_{M+1} = (x_{M+1,1}, y_{M+1,1}, \ldots, x_{M+1,N}, y_{M+1,N}, y_{M+1,N+1})$. Show results for various sizes of M and N.

(b) Add noise to the observed $y_{m,i}$, where now it is $y_{m,i} = w_m^T x_{m,i} + \epsilon_{m,i}$, where $\epsilon_{m,i} \sim \mathcal{N}(0, \sigma^2)$. Examine different sizes of σ^2 and examine the robustness of MAML to such noise-induced model mismatch.

(c) For (a) above, rather than doing MAML, use a Transformer with linear attention. As in https://arxiv.org/abs/2212.07677, show results as a function of M and N, and compare your results to MAML, and also to ordinary least squares (OLS). Do the linear-attention Transformer two ways: (i) based on the *designed* Transformer parameters, and also when the Transformer parameters are learned, based on $\{\mathcal{C}_m\}_{m=1,M}$. Compare the results of the Transformer with designed and learned parameters. There are similar experiments in the paper, that you should design your results on. Consider performance with and without noise $\epsilon_{m,i}$ added to the observed outcomes, like in (b).

Now consider contextual data $\mathcal{C}_m = (x_{m,1}, y_{m,1}, \ldots, x_{m,N}, y_{m,N})$, where in each case $y_{m,i} = f_{w_m}(x_{m,i}) = w_m^T x_{m,i}$, where each $w_m \sim \mathcal{N}(0_d, I_d)$, where $d = 10$. This is as above, but now the manner with which $x_{m,i}$ are drawn is different: Consider two 10-dimensional real-valued vectors: $v = (v_1, \ldots, v_{10})^T$ and $u = (u_1, \ldots, u_{10})^T$, where $v_j = \cos(j\pi/5)$ and $u_j = \sin(j\pi/5)$, for $j = 1, \ldots, 10$. Each $x_{m,i} = \alpha v + \beta u + \epsilon$, where $\alpha \sim \mathcal{N}(0,1)$, $\beta \sim \mathcal{N}(0,1)$, and $\epsilon = (\epsilon_1, \ldots, \epsilon_{10})^T$, with $\epsilon_j \sim \mathcal{N}(0, 1/100)$. Note, what this says is that each $x_{m,i} \in \mathbb{R}^{10}$ is expressed as a randomly scaled sum of two factors, with the two factors represented by u and v (with respective random weights α and β), and the vector $\epsilon\mathbb{R}^{10}$ represents a small amount of additive noise.

(d) What is the covariance of the data $X \in \mathbb{R}^{10}$ drawn in the manner specified above?

(e) For data generated as above, use the designed Transformer with linear attention to perform few-shot learning on new contextual data $\mathcal{C}_{M+1} = (x_{M+1,1}, y_{M+1,1}, \ldots, x_{M+1,N}, y_{M+1,N}, x_{M+1,N+1})$, and compare that to the performance of a linear Transformer *trained* on $\{\mathcal{C}_m\}_{m=1,M}$ (in the latter, you *learn* the Transformer parameters via $\{\mathcal{C}_m\}_{m=1,M}$). Consider performance for various settings of M, for the trained model. For this problem, pay special attention to the GD++ framework in https://arxiv.org/abs/2212.07677; we will also discuss this in class.

(f) Repeat the experiments in (c)-(d) using softmax attention and the full Transformer, as in https://arxiv.org/abs/2208.01066; in this case we *do not* make the linear-attention assumption and follow the framework in the referenced paper (and that we discussed in class). Compare the performance of the learned Transformer with softmax attention to what you got in (e) with linear attention. Which method works better, in the sense of model accuracy as a function of Transformer depth?