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

Abstract

In-context learning involves a model's capacity to utilize a prompt sequence composed of contextual examples (input-output pairs for a specific task) along with a new query input, subsequently generating the corresponding output. Notably, this type of learning occurs solely during inference without any adjustments to the model's parameters. While Large Language Models (LLMs), like GPT-3, demonstrate some proficiency in in-context learning, the relationship between successful tasks and the training data remains ambiguous...

1 Evaluation Method

In this section, we explain the data, metrics, and the baseline learning algorithms used to examine the performance of in-context learning of Transformers.

To create the training and test prompts $P = (x_1, f(x_1), \ldots, x_k, f(x_k + 1))$ these two steps are followed: First, sample a random function f from the class over D_F. Second, generate a set of random inputs $x_1, \ldots, x_{k+1}$ drawn independently from D_x and evaluate f on these inputs.

To have a better idea of the performance of the model, we compare the Transformer performance with three well-known learning algorithms:

- **Least Square Estimator:** In the Least Square method, which is the optimal estimator for this problem, we minimize the sum of squared residuals:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is the predicted value. The closed-form solution is:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

- **N-Nearest Neighbors:** For a given x, we average the y_i values of the N nearest neighbors of x_{query} using Euclidean distance:

$$\text{Distance}(p, p') = \sqrt{\sum_{i=1}^{n} (p_i - p'_i)^2}$$

- **Averaging:** Estimate v by averaging the values $y_i x_i$.

The evaluation metric is Squared Error (SE), defined as:

$$\text{SE} = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

2 Experimental Results

We now dive into the implementation and evaluation of the model.

2.1 Training a Transformer

To train a Transformer, we follow the general structure: given functions f from distribution D_F and inputs from D_x, we sample prompts

$$P = (x_1, f(x_1), \ldots, x_k, f(x_k), x_{k+1})$$

and train a model to anticipate $f(x_{k+1})$.