In-Context Learning for Transformers in Kernelized Logistic Regression

Circle Chen
circlecly@berkeley.edu

Lingqi Zeng
lzengaf@berkeley.edu

Zekai Wang
zekai.wang@berkeley.edu

Yanjie Chen
yanjiechen2003@berkeley.edu

Abstract

This study investigates the in-context learning capabilities of Generative Pre-trained Transformers (GPT) models in performing logistic regression, including its kernelized variant using the Radial Basis Function (RBF) kernel. We explore whether GPT models can effectively learn logistic regression in-context, comparing their performance with traditional machine learning algorithms like k-NN, SVM, and Gaussian Process Classifier. Our methodology involves generating synthetic data for logistic regression tasks and training GPT models on these datasets, both with and without noise. We also examine the impact of various factors on model accuracy, such as scaling, label noise, out-of-distribution data, and model capacity. Our findings indicate that GPT models show promising results in learning logistic regression in-context, outperforming or matching most baselines in various scenarios. This research contributes to understanding the potential of GPT models in statistical tasks and opens avenues for further exploration into their in-context learning mechanisms. [1].

Keywords: GPT models, in-context learning, logistic regression, machine learning, RBF kernel.

1 Introduction

In recent years, the advent of Generative Pre-trained Transformers (GPT) has revolutionized the field of machine learning, particularly in the domain of natural language processing. Among the various capabilities of these models, in-context learning has emerged as a particularly intriguing phenomenon. In-context learning refers to the model's ability to adapt to new tasks based on a given set of examples, without the need for explicit retraining or parameter adjustment. This capability has significant implications for the flexibility and applicability of GPT models in various domains.

Previous studies, such as [1] [5], attempt to let GPT learn to fit various function classes, including linear regression, logistic regression, neural networks with ReLU non-linearity, and decision trees. Among these function classes, logistic regression, a fundamental statistical method for binary classification, piques our interest due to its simplicity and widespread use. In this study, we attempt to first reproduce results from [1] that pertain to unkernelized logistic regression, and then extend our scope to kernelized logistic regression.

Our investigation focuses on several key research questions: (1) Can GPT models effectively learn to perform logistic regression in-context, compared with traditional ML baselines? (2) Would there be a difference if the problem is extended to RBF-kernelized logistic regression? (3) Is there a discernible

[1]The code for this project can be found at `https://github.com/zenglingqi647/ICL.git`

similarity between how GPT models compute logistic regression and traditional logistic regression algorithms?

Extending the codebase of [5], we empirically address this question and have the following findings:

1. **Transformers can learn logistic regression functions, both linear and RBF-kernelized, in-context.** By generating synthetic data, pretraining the Transformer model on input-output pairs, and evaluating on random input-output examples, we notice that Transformers perform on par, or sometimes better than, many of the traditional ML algorithms used to perform classification. Furthermore, decision boundary plots show that Transformers are able to induce a similar decision boundary to the ground truth data generator.
2. **Transformers can learn these functions with noise.** Even when the class labels are flipped randomly during training, Transformers can still achieve decent performance on the data.
3. **Transformers learn problems of varying dimensions, and bigger Transformers learn better.** By varying the number of dimensions and the model capacity, we discover that Transformers can learn logistic regression functions up to 40 dimensions, and increasing the number of parameters for the Transformer model improves accuracy.

2 Related Work

Transformers Transformers, whose underlying mechanism is known as "attention", were introduced in [16], and have shown to be prominent in language tasks. GPT models, which make use of stacked Transformer blocks, have especially been the hot topic of research, due to their human-like language generation abilities. They have been used in various areas, such as educational contexts [14], question-answering chatbots [19], propaganda and disinformation detection [20], to name a few.

In-Context Learning Since the discovery of Transformers' ability to learn new tasks without gradients updates and with few demonstrations (few-shot learning) in [3], in-context learning has been an area of interest. Previous work [5] has examined Transformer's (more precisely, decoder-only architectures with stacked causal self-attention layers) ability to conduct in-context learning on the class of linear functions. Experiments show that the transformer achieves comparable performance (in terms of L_2 loss) to the "ground truth" linear regression baseline and outperforms other baselines such as K Nearest Neighbor and simple averaging. Moreover, when tested on out-of-distribution samples (i.e., at test time sample $x_i \sim \mathcal{D}_{\text{test}}$ that is different from $\mathcal{D}_X$), transformer outperforms the baselines and exhibits certain characteristic of linear regression (i.e. Double Descent). This heuristically shows transformer might indeed be learning how to do linear regression. Other works [1] have also examined other classes of functions, including sparse linear regression (LASSO), decision tree, 2-layer neural network, and linear classification.

In addition to fitting functions, [4] explores deeper into the types of distributions under which in-context learning emerges, such as those with high "burstiness". These studies show that In-Context Learning is an intriguing topic of research.

More works focus on analyzing the impact of demonstration samples on ICL performances. [18] concludes that input-label pairing format as well as input distribution can greatly affect the in-context learning performances. [9] shows that the sample order also contributes to the accuracy of in-context learning.

Curriculum Learning Curriculum learning is a philosophy of speeding up training using organized examples which "illustrates gradually more concepts, and gradually more complex ones" [2]. Since its introduction, it has been adopted for wide usage in studies of neural networks, such as in [6] and [7] for speeding up training of models, as well as helping with reinforcement learning training [10]. Previous work [5] reports that training the transformer with a curriculum can improve training speed. Specifically, by initially sampling functions from a lower-dimensional subspace and gradually increasing the complexity of the function class and prompt length, the authors observe significant improvement in the speed of training.

Kernelized Logistic Regression We focus our experiment on the logistic regression with the RBF (Radial Basis Function) kernel. The RBF kernel is a kernel function defined as follows: for all

$\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\sigma^2}\right)$. The RBF kernel corresponds to the dot product of $\mathbf{x}, \mathbf{x}'$ projected to an infinite dimensional feature space. The RBF kernel is widely used in SVM and other machine-learning algorithms.

3 Problem Statement

We define the problem of in-context learning in the same way as [5]. We train the model with $\mathbf{x}_1, f(\mathbf{x}_1), \ldots, \mathbf{x}_i, f(\mathbf{x}_i)$ in an in-context way so that given the input $\mathbf{x}_{i+1}$, the model can predict $f(\mathbf{x}_{i+1})$, where f is sampling randomly from a function class. The data $\mathbf{x}_1, \ldots, \mathbf{x}_i$ are generated synthetically from the Gaussian distribution $\mathcal{N}(\mathbf{0}, I_d)$. The corresponding label y_k, where $k = 1, 2, \cdots, i$ are generated using the function f. The prompts are in the form

$$P = (\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, \ldots, \mathbf{x}_i, y_i, \mathbf{x}_{i+1}). \tag{1}$$

If we define our model to be M_θ, the loss of our task function is $\ell(\cdot, \cdot)$, and the prefix prompt containing the first i examples be P_i, then minimizing the following loss would allow training of the Transformer to learn function class f:

The objective we are going to minimize can be expressed as in [15]:

$$\mathcal{L}(\theta) = \mathbb{E}_P \left[\frac{1}{k+1} \sum_{i=0}^{k} \ell\left(M_\theta(P_i), f(x_{i+1})\right) \right], \tag{2}$$

In other words, whenever the transformer sees a token that corresponds to an input, the generated output of the transformer is considered its prediction of the function evaluated at that input, and we are averaging the function loss for the Transformer over all the input-output pairs. The gradient of this loss function could then be back-propagated into the model weights, which allows training the model for a specific function class.

4 Methodology

4.1 Prompt Distribution

Vanilla Logistic Regression For the vanilla logistic regression model, we follow the work of [1] and consider the function class where

$$\mathcal{F} = \left\{ f | f(\mathbf{x}) = \mathbf{1}_{\{\sigma(\mathbf{w}^T\mathbf{x}+\mathbf{b}) \geq 0.5\}}, w \in \mathbb{R}^d, b \in \mathbb{R}^d \right\}$$

where d is the dimension of the input, and σ is the sigmoid function. The weights $\mathbf{w}$, bias $\mathbf{b}$, and data points $\mathbf{x}$ are drawn from Gaussian distribution

$$\mathbf{w} \sim \mathcal{N}(0, I_d), \mathbf{b} \sim \mathcal{N}(0, 1), \mathbf{x} \sim \mathcal{N}(0, I_d).$$

The label y_i for each feature vector $\mathbf{x}_i$ is assigned as

$$y_i = \mathbf{1}_{\{\sigma(w^T x + b) \geq 0.5\}}.$$

Kernelized Logistic Regression For the kernelized version of logistic regression, we consider a function class that utilizes the Radial Basis Function (RBF) Kernel. The function class is defined as:

$$\mathcal{F}_{\text{kernel}} = \left\{ f | f(\mathbf{x}) = \mathbf{1}_{\{g(\mathbf{x}) > \theta\}}, g(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}'), \theta \in \mathbb{R}, \mathbf{x}' \in \mathbb{R}^d, \mathbf{x} \in \mathbb{R}^d \right\},$$

where $K(\mathbf{x}, \mathbf{c})$is the RBF kernel. α_i are coefficients, and θ is the threshold.

The kernel width σ is drawn from a Gaussian distribution and is constrained to be non-zero. The threshold θ is a learned parameter. The feature vectors $\mathbf{x}'$ and $\mathbf{x}$ are drawn from Gaussian distributions

$$\mathbf{x}' \sim \mathcal{N}(0, I_d), \mathbf{x} \sim \mathcal{N}(0, I_d).$$

For assigning labels y_i for each feature vector $\mathbf{x}_i$, our approach is to utilize a radial function

$$f(\mathbf{x_i}) = \|\mathbf{x_i} - \mathbf{c}\|_2, \mathbf{c} \ \sim \mathcal{N}(0, I_d)$$

Figure 1: A randomly generated dataset for the RBF classification task with 2 dimensions. The green cross is the center, and the red circle is the decision boundary. The blue dots are the data points with labels 1, and the orange dots are the data points with labels 0. Left: data without label noise. Right: data with label noise.

where $\mathbf{c}$ is a random "center point" whose coordinates are also drawn from Gaussian distributions. The label y_i for each feature vector $\mathbf{x}_i$ is assigned as

$$y_i = \mathbf{1}_{\{f(\mathbf{x}_i) > \theta\}}.$$

where the value of θ is set as the median among all $f(\mathbf{x}_i)$s to avoid imbalanced labels, in which case it is possible that the generated labels are all zeros or all ones. As we are working with Euclidean distances, a good decision boundary for this type of data would be in the shape of a circle. Figure 1 illustrates a set of inputs that we generated for this task.

Model Architecture We use the GPT-2 model architecture [12]. GPT-2 employs a transformer architecture with parameters. It adopts self-attention layers and is autoregressive in nature. Even though it is primarily used for text generation, its versatile generative capabilities make it also possible for in-context learning. Therefore, instead of taking natural languages and embedding them into vectors, we can provide data points as "prompt" and ask it to fit certain mathematical functions.

Training To train the model, we first generated prompts according to our previous formulations of vanilla and RBF-kernelized regression. Then, we compute a forward pass of the model using the prompt formulated as in Equation 1. We then compute the loss as defined in Equation 2, and perform backward pass into the Transformer model parameters.

To introduce noise, each label y_i can be flipped with a probability p during training. This allows us to assess the robustness of the model to perturbations.

4.2 Evaluation

After training the model for enough iterations, we examine the performance of the model. We generate random functions using the same configurations as in training, and sample random input features to compute ground truth labels. The predictions given by the Transformer model would be compared against the ground truth labels, allowing us to compute the accuracy of the Transformer model. Because Transformer models have access to all previous tokens in the prompt, it is expected to learn from the previous function examples, and we would examine how its prediction accuracy vary as the number of in-context examples increase.

To better assess the performance of GPT, we also established a set of baselines encompassing various classical machine learning algorithms. These ML algorithms are trained using the same in-context examples that the Transformer models would have access to.

4.2.1 Vanilla Logistic Regression

For the standard logistic regression task, our baseline models include **Linear Discriminant Analysis (LDA)**, **Support Vector Machines (SVM)**, **k-Nearest Neighbors (k-NN)**, and a vanilla **logistic regression model**.

Linear Discriminant Analysis (LDA) Linear Discriminant Analysis (LDA) [17] is a classical method in statistics and machine learning for classification and dimensionality reduction. The fundamental idea of LDA is to project the data onto a lower-dimensional space and seek to find a linear combination of features that best separates them. Given a dataset with d-dimensional feature vectors $\mathbf{x}_i$ and corresponding class labels y_i, where $i = 1, \ldots, n$, the goal of LDA is to find a projection vector $\mathbf{w}$ that maximizes the ratio of the cross-class variance to the in-class variance.

The between-class scatter matrix S_B and the within-class scatter matrix S_W are defined as follows [8]:

$$S_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T, \tag{3}$$

$$S_W = \sum_{i:y_i=1} (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^T + \sum_{i:y_i=2} (\mathbf{x}_i - \boldsymbol{\mu}_2)(\mathbf{x}_i - \boldsymbol{\mu}_2)^T, \tag{4}$$

where $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are the means of the feature vectors for each class.

The optimal projection vector $\mathbf{w}$ is obtained by solving the following optimization problem:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}. \tag{5}$$

Support Vector Machines (SVM) Support Vector Machines (SVM) is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The hyperplane, also referenced as the decision boundary, aims to maximize the margin of the training data. The decision function is defined as

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

where $\mathbf{w}$ is the normal vector to the hyperplane, $\mathbf{x}$ is the feature vector, b is the bias term, and sign is the sign function. The training involves adjusting $\mathbf{w}$ and b to maximize the margin of the classifier. It can also be extended to non-linear classification with the kernel trick.

k-Nearest Neighbors (k-NN) k-Nearest Neighbors (k-NN) operates on the principle that similar data points are likely to be in close proximity to each other in the feature space. The k-NN algorithm classifies a data point based on how its neighbors are classified.

Given a dataset with d-dimensional feature vectors $\mathbf{x}_i$ and corresponding class labels y_i, where $i = 1, \ldots, n$, the k-NN algorithm classifies a new data point $\mathbf{x}_{\text{new}}$ by identifying the k closest data points (neighbors) to $\mathbf{x}_{\text{new}}$ in the training dataset. The classification of $\mathbf{x}_{\text{new}}$ is then determined by a majority vote among its k neighbors.

4.2.2 Kernelized Logistic Regression

In terms of kernelized logistic regression baselines, we used the **Gaussian Process Classifier**, both with and without RBF Kernels, **SVM**s equipped with RBF Kernels, the vanilla **logistic regression model**, and an **RBF-kernelized logistic regression model**. Additionally, we incorporated **k-NN algorithms**, considering both standard and RBF-kernelized versions.

Gaussian Process Classifier (GPC) Gaussian Process Classifier (GPC) [13] is a probabilistic, non-parametric model. The dataset consists of d-dimensional feature vectors $\mathbf{x}_i$ and corresponding binary class labels $y_i \in \{0, 1\}$, where $i = 1, \ldots, n$. GPC places the Gaussian process prior over the latent function $f(\mathbf{x})$ that maps the feature vectors to a continuous value, which is then transformed into a probability for the binary classification through a link function, typically the logistic function.

Given a new data point $\mathbf{x}_{\text{new}}$, the GPC predicts the class label by computing the posterior distribution of the latent function $f(\mathbf{x}_{\text{new}})$, given the training data. The prediction is the probability that the latent function value exceeds a certain threshold corresponding to the decision boundary. For our experiments, we used $k = 3$.

5 Experiments

5.1 Logistic Regression In-Context Learning

We first trained the Transformer in-context with generated logistic regression data, without noise. We experimented with unkernelized and RBF-kernelized regression. We used the same curriculum learning strategy as in [1] and [5], which gradually increases the dimension and number of data points. For this subsection, we gradually increase the number of points from 11 to 41, and gradually increase the number of dimensions from 5 to 20. We trained the model for around a total of 3×10^4 iterations with a batch size of 64, due to constraints of time and compute.

After training the clean version of vanilla and RBF-kernelized logistic regression, we then introduced different levels of noise in training labels for the Transformer model. The training labels are flipped independently with a certain probability $p = [0.05, 0.1, 0.2]$, but in evaluation, the clean version of the label is used. For this experiment, we also both experimented with unkernelized and RBF-kernelized regression.

Our training curves are shown in Figure 2, and the accuracies are shown in Table 1 2. The evaluation curves demonstrate that Transformers are able to perform better or equal to most baseline models. One noteworthy discovery is that Transformers did not perform well in the unkernelized case when labels are flipped with a probability of 0.2. For the other noisy cases, the Transformers still perform well (more details in the Appendix Figure 11 12). In the discussion section, we will attempt to explore the reasons.

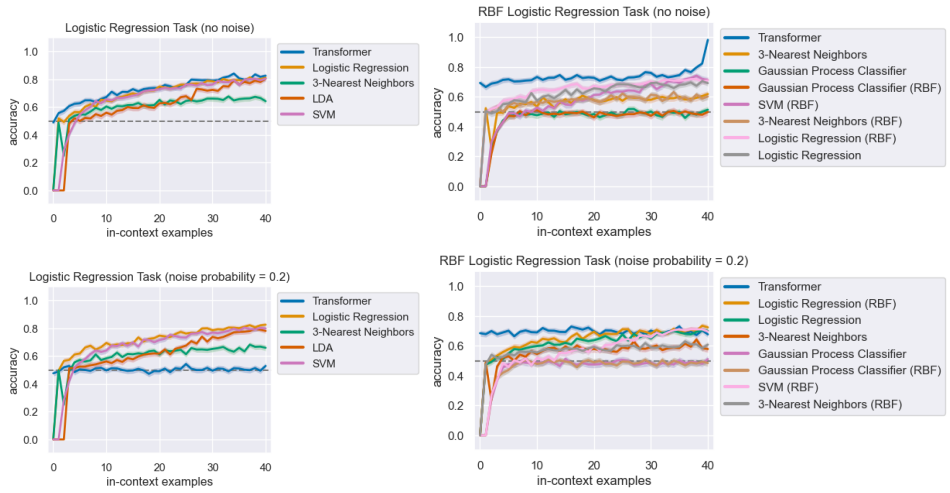


Figure 2: Evaluation accuracy curves. Top left: logistic regression with no noise and no kernels. Top right: no noise, RBF kernel. Bottom left: label noise, no kernel. Bottom right: label noise, RBF kernel. For other noise probabilities, see Figures 9 and 10.

Table 1: Evaluation Accuracy of Logistic Regression without noise for Transformers and baselines.

Number Of Examples	10	20	40
Transformer	**0.6727**	0.7383	**0.8266**
Logistic Regression	0.6477	**0.7484**	0.8109
3-Nearest Neighbors	0.6047	0.6523	0.6430
LDA	0.5602	0.6273	0.8055
Linear SVM	0.6555	0.7289	0.8047

5.2 Effect of Scaling on Accuracy

We also experimented with the robustness of Transformer models with respect to scaling of inputs. [1] concluded that Transformer models are robust to scaling of inputs in the linear regression task,

Figure 3: Scaling robustness experiment results. Left: robustness to scaling on unkernelized logistic regression. Right: robustness to scaling on RBF-kernelized regression.

and we are intrigued by whether the same thing holds for logistic regression. For this experiment, we multiplied a scalar constant to all input values before sending it to the model, while keeping the output labels the same. We then plotted the accuracies and the scale factor. Our results are shown in Figure 3.

Different from linear regression, the model performance for logistic regression is quite sensitive to scaling, with a peak at when there is no scaling, and quickly dropping at both sides of the peak. It is noteworthy that for the RBF kernel, the accuracy quickly falls off to 0.5 compared with the unkernelized logistic regression. We hypothesize that this is because with large scaling, all the points would be cluttered together, so there is no way to tell points apart. Another noteworthy feature is that the number of examples does not impact learning significantly for RBF-kernelized regression, but affects the accuracy for unkernelized logistic regression.

5.3 Varying Problem Dimensions and Model Capacity

We also experimented with different problem dimensions ($d = [10, 30, 40]$) and three different model sizes. For the problem dimension, we use the standard GPT-2 model and keep the noise probability at 0.1, while ablating problem dimensions for the RBF-kernelized regression task to 10, 30, and 40. We also ablated the capacity of the model, keeping problem dimensions at 20 and noise probability as 0.1, while varying the number of layers. The evaluation results are shown in Figure 4. We noticed that the Transformer models are still able to learn to perform RBF-kernelized regression in-context, even when the model size or the problem dimensions change.

5.4 Extrapolating beyond the training distribution

Expanding on the framework by Garg et al. [5], our experiments were designed to evaluate the model's extrapolation capabilities beyond its training distribution. We kept the function distribution D_F constant and varied the prompt input distribution D_X. The scenarios assessed included:

Table 2: Evaluation Accuracy of RBF Logistic Regression without noise for Transformers and baselines.

Number Of Examples	10	20	40
Transformer	**0.7336**	**0.7281**	**0.9828**
3-Nearest Neighbors	0.5344	0.5781	0.8109
Gaussian Process Classifier	0.4773	0.4852	0.5141
Gaussian Process Classifier (RBF Kernel)	0.5109	0.4625	0.4992
SVM (RBF Kernel)	0.4898	0.6086	0.7195
3-Nearest Neighbors (RBF Kernel)	0.5672	0.5758	0.6031

- **Opposite Quadrants:** D^X_{train} and D^X_{test} distributed exclusively in two opposite quadrants.
- **Random Quadrants:** D^X_{train} limited to two random quadrants, with D^X_{test} following a Gaussian distribution across the space.
- **Overlapping:** A dimension reduction in D^X_{test}.
- **Standard:** D^X_{train} and D^X_{test} are generated from Gaussians.

This study strictly used clean data, avoiding noise introduction. Further details on data distribution are available in the appendix. The input sequence length varied from 10 to 8, and the data dimension was set to 20.

As shown in our results (Figure 5), the model performs best in-distribution. In vanilla logistic regression, scenarios with opposite and overlapping distributions show better performance. Conversely, for RBF-kernelized logistic regression, the in-distribution case generally surpasses other out-of-distribution (OOD) scenarios, especially as sequence length increases. Notably, the random quadrant configuration for D^X_{train} and D^X_{test} results in the lowest accuracy for both model types. This is likely due to D^X_{test} encompassing quadrants not covered during training.

5.5 Does the Label Space Matter?

Inspired by the findings of Min et al. [11], which suggest that substituting gold labels with random labels leads to only a minimal drop in performance, we designed a series of experiments to further investigate this claim. Our approach involves introducing randomness into the label space in three distinct ways: (1) generating labels randomly from a Gaussian distribution, (2) generating labels randomly from a uniform distribution, and (3) randomly permuting the labels.

We conducted these experiments using logistic regression and RBF-kernelized regression in an environment with a feature dimension set to $d = 20$. Our findings 6 revealed that the accuracy of logistic regression dropped to approximately 0.5 after the introduction of noise, indicating a lack of meaningful learning by the model. A similar trend was observed in RBF-kernelized regression, where accuracy also hovered around 0.5. These results underscore the significance of the label space in our task, demonstrating that it plays a crucial role in model learning and performance.

6 Discussion

6.1 What does the decision boundary of the model look like?

We've seen that Transformers can do RBF-kernelized logistic regression, but how do they work? In this section, we try to open the Transformer "black box" by visualizing the decision boundary learned by our model. We trained the Transformer on 2-dimensional data for the RBF-kernelized regression task and plotted the decision boundary of the model. This is done by iterating through a 2D grid of (x, y) coordinate pairs, and asking the Transformer to predict the function value at that point, given all in-context examples. Figure 7 shows the decision boundary of our model, as well as the original

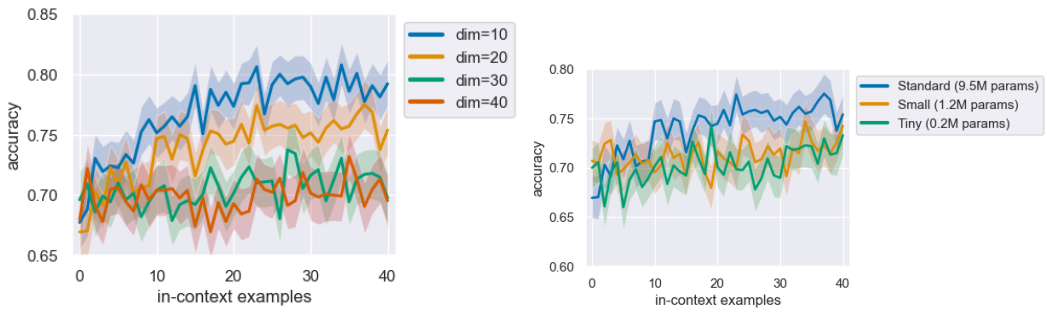


Figure 4: Experiment results for varying problem dimensions and model capacities. Left: Evaluation accuracy curves for different problem dimensions. Right: Different model capacities.

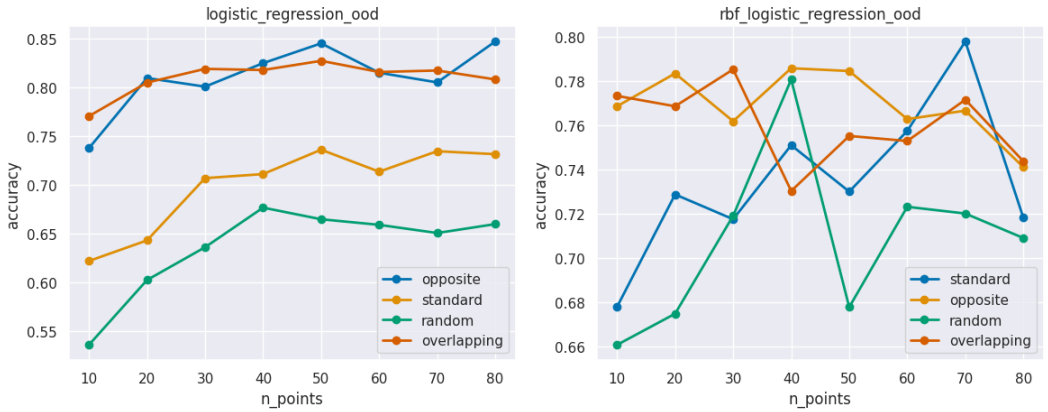


Figure 5: Logistic regression (left) and RBF-kernelized logistic regression (right) with out-of-distribution data.

data point and intended decision boundary. It can be seen that our Transformer model seems to have learned a circular decision boundary similar to the one used in the original dataset.

However, the decision boundary for RBF-kernelized logistic regression is sensitive to noise. When we introduced a label noise with probability 0.1, the decision boundary is significantly offset from the one used in the problem. We noticed that these decision boundaries have a tendency to be zero-centered, and we hypothesize that the reason of this is label noise effectively acts as a form of regularization, similar to dropout in neural networks. In trying to fit a noisy dataset, the model may opt for a simpler, more generalizable decision boundary. This impacts the accuracy of the model, because the ground truth decision boundary is often not zero centered.

6.2 Why does unkernelized logistic regression not work with noise?

In the experiments section, we noticed that vanilla logistic regression with a label noise probability of 0.2 causes the Transformer to only achieve the same accuracy as a random classifier. To explore this phenomena, we first plotted the distribution of distances to the decision boundary for both noisy and clean data, in Figure 8. As can be seen, when the noise probability reaches 0.2, there is a significant overlap between the distribution of distances, which may impede the model's performance. When we reduce noise probability to 0.1 and 0.05, however, the distribution overlap becomes less severe.

When the Transformer models are trained under noise probabilities of 0.1 and 0.05, they do perform on par with machine learning baselines, which aligns with our previous explorations. This implies

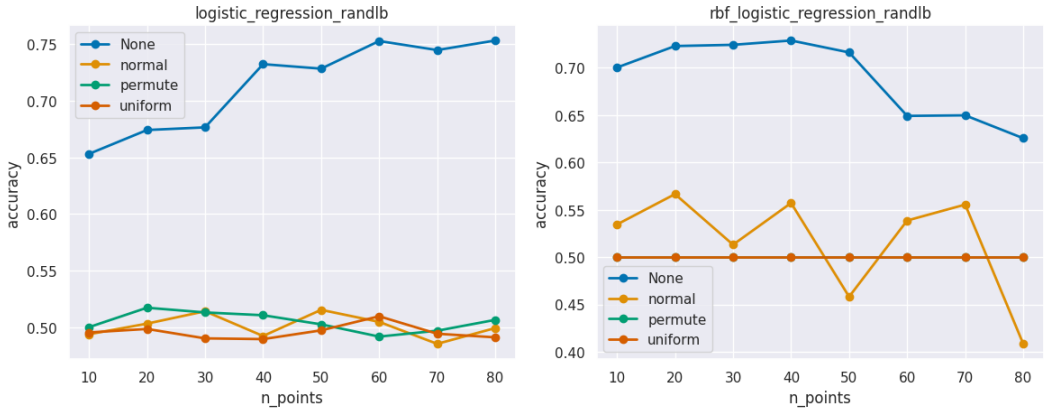


Figure 6: Logistic regression and RBF-kernelized logistic regression with random labels. For RBF-kernelized logistic regression, the outcome of permutation and uniform distribution overlapped.

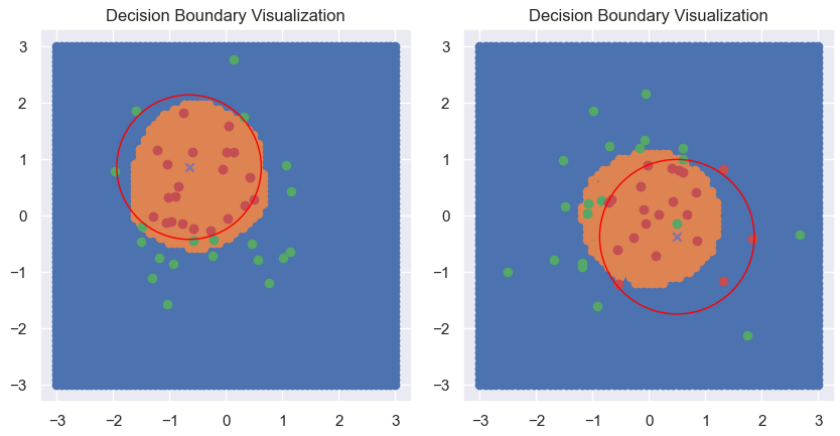


Figure 7: Decision boundary of the Transformer model. Ground truth data: red points (class 0) and green points (class 1) are the in-context examples sent to the model. The circle and cross are the intended center and decision boundary of the data. Model predictions: The orange region is what the model decides as class 0, and the blue region is what the Transformer model decides as class 1. Left: Decision Boundary when there is no noise. Right: Decision Boundary when there is noise with 0.1 probability.

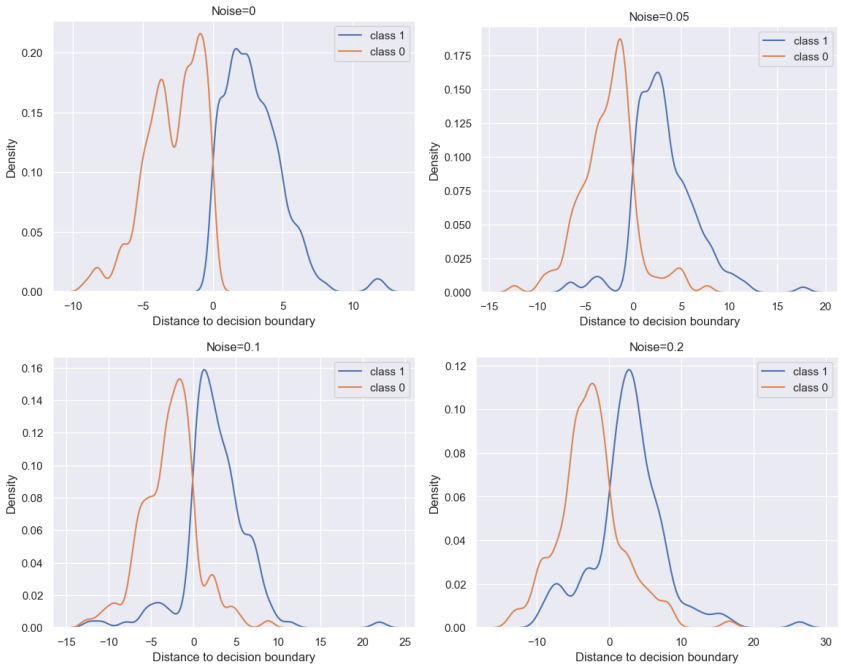


Figure 8: Kernel Density Estimation (KDE) histograms of the distribution of distances to the decision boundary, with varying noise probabilities.

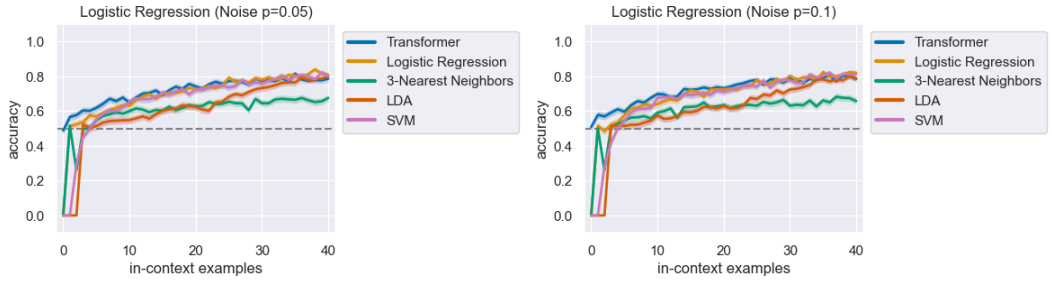


Figure 9: Evaluation accuracy curves for vanilla logistic regression with noise probability 0.05 and 0.1. Notice that Transformers are amongst the best models.

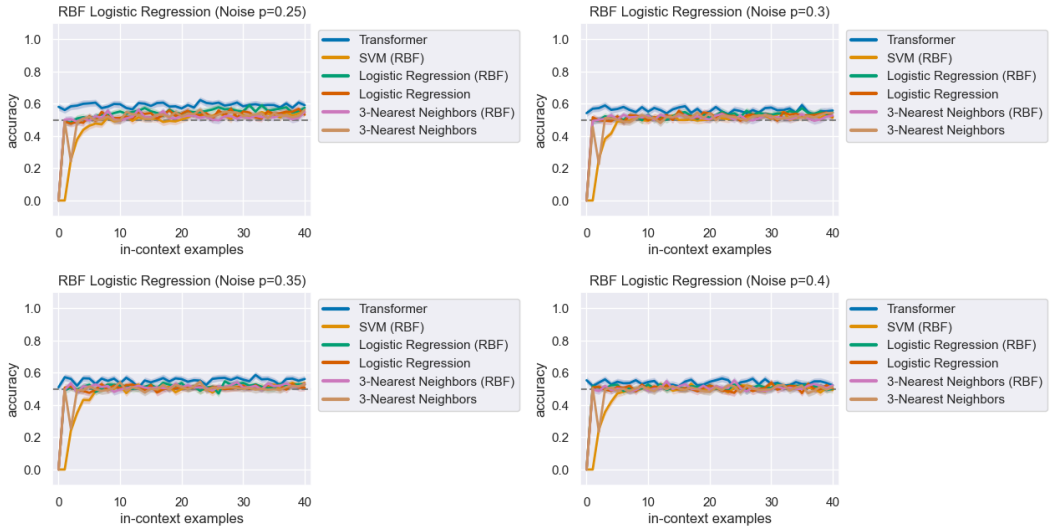


Figure 10: Evaluation accuracies of Transformer models under other noise probabilities in the RBF logistic regression task. Note that Transformers still performs amongst the top models.

that a noise probability of 0.2 is too high for Transformers to learn vanilla linear regression, and therefore **they are more sensitive to label noise than classical ML algorithms in this problem**.

Theoretically, flipping the class labels independently with probability p can be modeled as a Binary Symmetric Channel (BSC) with crossover probability p in communication theory. The capacity of such a channel is $1 - h(p)$ bits per channel use, where $h(p) = -p \log_2 p - (1-p) \log_2 (1-p)$, the binary entropy function. As can be seen here, the higher label noise is, the less useful information there will be to the model. When $p = 0.5$, the channel capacity is 0, so the channel is unusable. Here, any classifier could not achieve better performance than random. When $p = 0.2$, the capacity of the channel is roughly 0.278 bits per channel use, which is only around one-fourth of the capacity when there is no noise. When we use a noise probability of 0.1 and 0.05, however, the channel capacities are 0.531 and 0.714 bits per channel use, respectively. From this, we hypothesize that Transformers require more "bits" to be sent across the "channel", that is, the labels, than classical ML algorithms.

6.3 When would RBF-kernelized logistic regression stop working?

As mentioned in the previous subsection, unkernelized logistic regression does not work for Transformers with noisy labels $p = 0.2$, but works for $p = 0.05$ and $p = 0.1$. However, RBF-kernelized logistic regression works for Transformers for $p = [0.05, 0.1, 0.2]$. Therefore, another question arose: what is the point at which Transformers for RBF-kernelized regression would also perform poorly? To do this, we further increased the noise probability, using $p = [0.25, 0.3, 0.35, 0.4]$ to experiment. Our results are shown in Figure 10. Interestingly, with this amount of added noise, our Transformer still outperforms all other models, even though all of the accuracies are greatly degraded by the noise.

6.4 Summary and Future Directions

In this paper, we experimented with the ability for Transformers to learn to perform logistic regression in-context, in both the unkernelized and the RBF-kernel setting. We also noted differences in the robustness of Transformers to perturbations and transformations of the data. Despite having made significant process in verifying the feasibility of Transformers in learning how to perform kernelized logistic regression in-context, further experiments are needed into this problem. First of all, we still need to experiment with different Transformer architectures, including more recent architectures like GPT-3. We attempted to implement alternative Transformer structures, but unfortunately could not get them to run correctly. Secondly, more work is required to analyze the underlying mechanics of the GPT-2 models. Such exploration could include visualizations of the intermediate model weights and outputs. Thirdly, a more complex classification task other than the circular decision boundary problem could be used. For instance, we can first generate class labels according to a linear decision boundary randomly, and project it onto the RBF feature space using the RBF kernel.

References

[1] Yu Bai et al. "Transformers as Statisticians: Provable In-Context Learning with In-Context Algorithm Selection". In: *arXiv preprint arXiv:2306.04637* (2023).

[2] Yoshua Bengio et al. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380. URL: https://doi.org/10.1145/1553374.1553380.

[3] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[4] Stephanie Chan et al. "Data distributional properties drive emergent in-context learning in transformers". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 18878–18891.

[5] Shivam Garg et al. "What Can Transformers Learn In-Context? A Case Study of Simple Function Classes". In: *arXiv preprint*. 2022.

[6] Alex Graves et al. "Automated Curriculum Learning for Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1311–1320. URL: https://proceedings.mlr.press/v70/graves17a.html.

[7] Guy Hacohen and Daphna Weinshall. "On The Power of Curriculum Learning in Training Deep Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2535–2544. URL: https://proceedings.mlr.press/v97/hacohen19a.html.

[8] *Linear discriminant analysis (LDA) - San José State University*. URL: https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec11lda.pdf (visited on 11/28/2023).

[9] Yao Lu et al. "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity". In: *arXiv preprint arXiv:2104.08786* (2021).

[10] Sha Luo, Hamidreza Kasaei, and Lambert Schomaker. "Accelerating Reinforcement Learning for Reaching Using Continuous Curriculum Learning". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207427.

[11] Sewon Min et al. "Rethinking the role of demonstrations: What makes in-context learning work?" In: *arXiv preprint arXiv:2202.12837* (2022).

[12] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[13] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*. Vol. 1. Springer, 2006.

[14] Jaromir Savelka et al. "Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses?" In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ITiCSE 2023. ACM, June 2023. DOI: 10.1145/3587102.3588792. URL: http://dx.doi.org/10.1145/3587102.3588792.

[15] Dimitrios Tsipras. *In-Context Learning*. https://github.com/dtsip/in-context-learning. Accessed: 15 Nov 2023. 2022.

[16] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[17] William N Venables and Brian D Ripley. *Modern applied statistics with S-PLUS*. Springer Science & Business Media, 2013.

[18] What Makes In-Context Learning Work. "Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?" In: ().

[19] Yunze Xiao. "A Transformer-based Attention Flow Model for Intelligent Question and Answering Chatbot". In: *2022 14th International Conference on Computer Research and Development (ICCRD)*. 2022, pp. 167–170. DOI: 10.1109/ICCRD54409.2022.9730454.

[20] Yunze Xiao and Firoj Alam. *Nexus at ArAIEval Shared Task: Fine-Tuning Arabic Language Models for Propaganda and Disinformation Detection*. 2023. arXiv: 2311.03184 [cs.CL].

Appendix I

A OOD Data Visualization

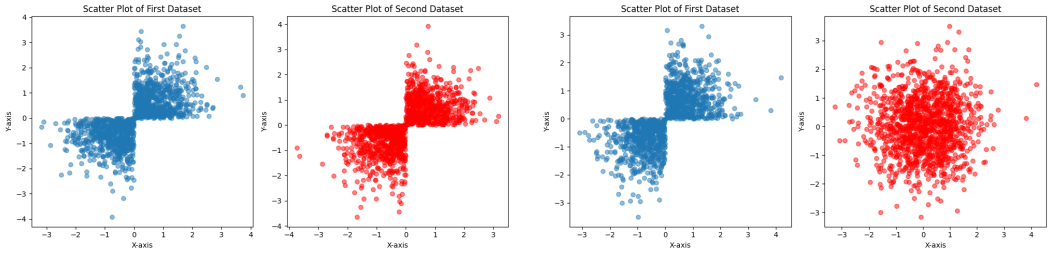


Figure 11: Opposite quadrants, and random quadrants distributions.

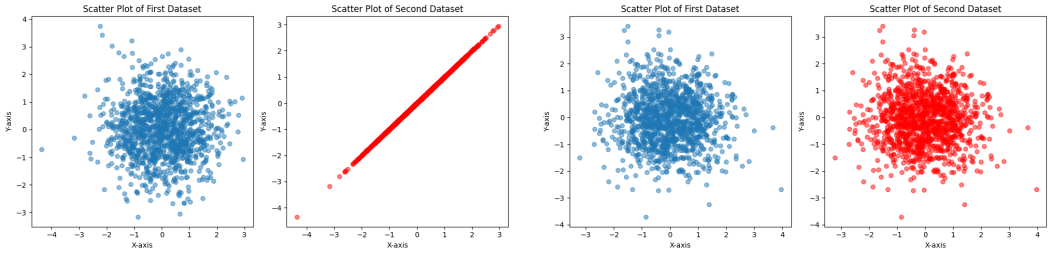
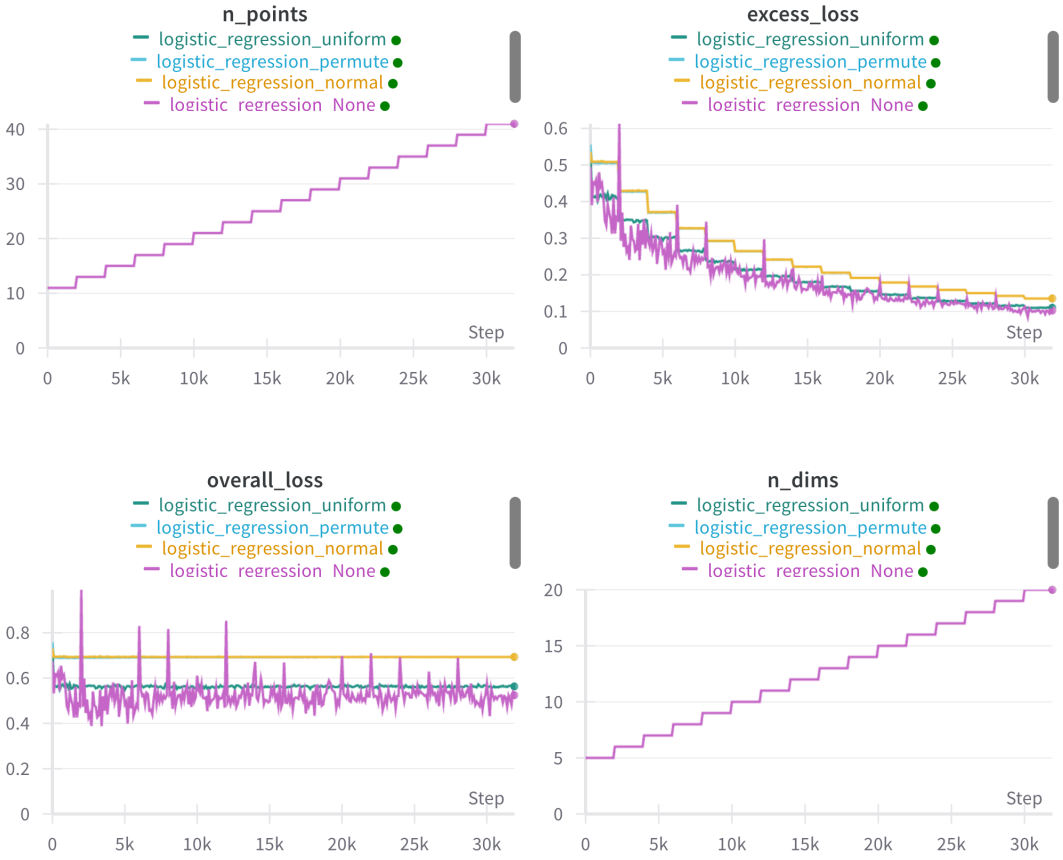
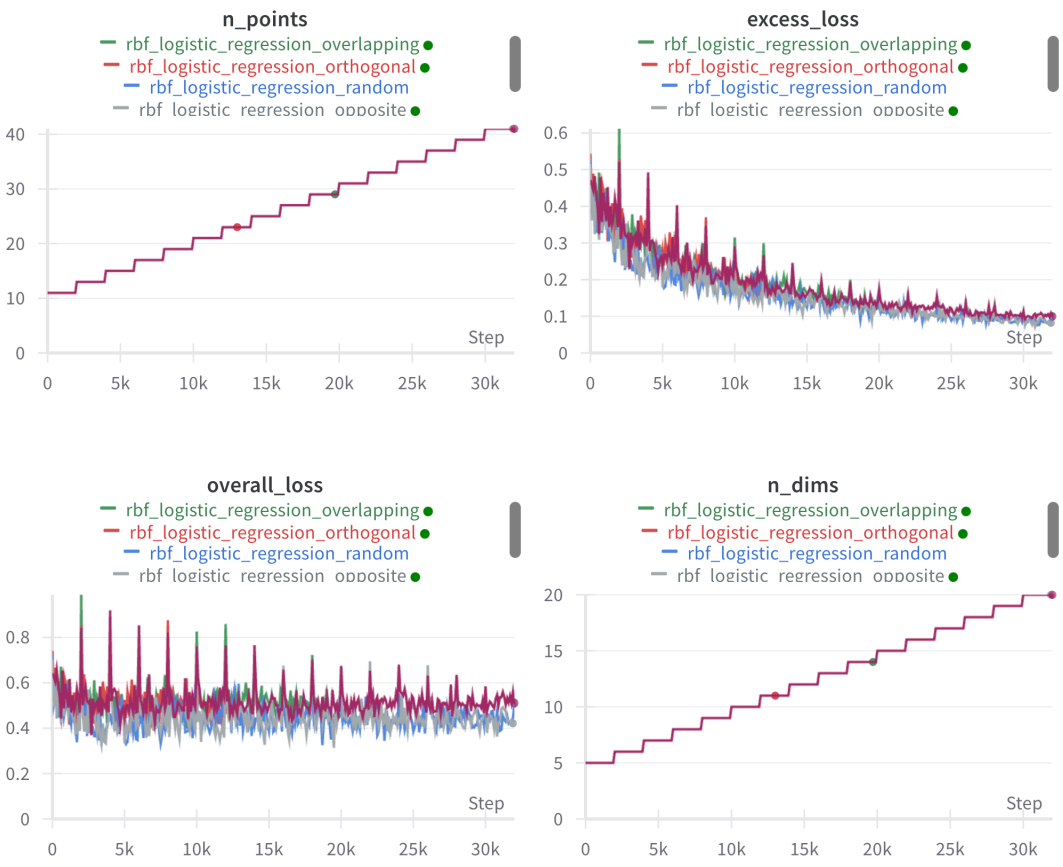


Figure 12: Overlapping, and standard distributions.

B Training Curves of Random Label Experiments on Logistic Regression



C Training Curves of OOD Experiments on RBF Logistic Regression



D Updates

1. **Review-310: Could use more references for related works with respect to in-context learning.**
   - Included more works related to in-context learning in the related work section.
2. **Review-310: Add another performance analysis for logistic regression on another data distribution. Experiments currently limited to one case.**
   - Added new experiment s such as varying dimensions and model capacity, tuning noises, the impact of label space, and extrapolating beyond training distribution.
3. **Review-310: There are no instructions for reproducing results.**
   Review-214: I was not able to reproduce the results as I felt I did not have enough information to create the same experiment used in the paper. Specifically, it is unclear how the transformer model was setup and trained to solve the logistic regression problems.
   Review-194: The paper's presentation concerning language and format requires meticulous review and refinement. For instance, the annotation accompanying Figure 3 is erroneously replicated from Figure 2, leading to a misalignment with the actual content of Figure 3.
   Review-194: Thoroughly polish the language and format, especially in the description of experimental procedures and in the presentation of figures and annotations, which guarantees the precision and lucidity of the paper.
   Review-310: Figure 1 does not have labeled axes or a legend.
   Review-214: There are some typos and minor grammatical syntax errors, but otherwise the English in the paper is correct and clear.
   - Added instructions for reproducing results.
   - Explained the experiment settings in detail.
   - Fixed the typos.
4. **Review-180: The paper lacks in-depth analysis of the internal mechanics of the model and its relation to the geometric interpretations of the RBF kernel.**
   Review-194: There exists a paucity of in-depth theoretical analysis and insights into internal mechanics of in-context learning, which probably results from the tight time budget.
   - A more detailed version of the problem statement is included in the revised paper.
   - The data space is further investigated, as well as the decision boundary.
   - The math explanations are also revised.
5. **Review-194: A significant portion of the paper is devoted to explicating both standard and kernelized logistic regression models, as well as baseline models. This emphasis results in a disproportionate allocation of content, somewhat overshadowing the intended focus on in-context learning (ICL).**
   Review-194: Rebalance the content to focus more on in-context learning (ICL), ensuring it is the central theme rather than an overshadowed aspect. Consider allocating more time for a deeper theoretical analysis, providing richer insights into the internal mechanisms of in-context learning. For instance, the authors could not only adjust the transformer architectures to effectively generalize with scaling data, but fine-tune the original linear regression model to adapt to logistic regression tasks to speed up training.
   Review-214: The paper lacks sufficient detail about the experiment in order to replicate it. The paper also does not seek to explain the outlier case it experienced in the experiment with the unkernelized logistic regression problem with noise in which the transformer failed to learn a model of the data. It also does not attempt to explain the means by which the observed results materialize.
   Review-214: Please provide more details on the experiments you are performing, including a guide to replicating the experiments as well as a more in-depth explanation of the design choices behind the experiments.
   - We enriched the experiment section. More figures and detailed explanations are included in the revised paper to better illustrate our project.
   - Training speed is acceptable, and as a baseline, our focus is not on fine-tuning the linear regression model.
   - Added explanations in the analysis of experiments. - Added instructions for reproducing

results.

6. **Review-194: Considering the toy setting, the experiments are only conducted over logistic regression class, a relatively elementary statistical model.**
   - Logistic regression and kernelized logistic regression are the problems we are investigating. It is worth noting that the problem settings of this project are about toy problems.
7. **Review-194: The transformer could be further tested on real word data if time permits.**
   - Our experiment settings are about logistic regression and kernelized logistic regression. Real-world data does not align with our goal.
8. **Review-180: I think if the compute is allowed, trying out different model architecture, including the most advanced GPT4 via OpenAI API will be a plus. For people who's not coming from In context learning background, it's hard to navigate around the content. I would recommend having a diagram to show the experiment pipeline besides the charts for data.**
   - Our focus is not on the model architecture, though we initially hoped to investigate how the inner architecture affected the performance of the model. As for the diagram, as we mainly generate the data and then feed the data to the model, we think the focus should be the main experiments and extended experiments.
9. **Other modifications**
   - Modified the abstract to include our new experimental results.
   - Elaborated our contribution in detail in the Introduction.
   - Present our understanding regarding the findings of our project in the Discussion section.