EXPECTED LOSS OF MAML:A COMPERATIVE STUDY

Author1, Author2
Affiliation
Univ
City
{Author1, Author2}email@email

Author3
Affiliation
Univ
City
email@email

ABSTRACT

This works aims to investigate the generalization performance via the expected loss investigation of, arguably the most impactful meta-learning algorithm Model-Agnostic Meta-Learning (MAML) [1], which claims to have "good" generalization capabilities for a given task in few-shot image classification, few-shot regression, and reinforcement learning scenarios. However, the claimed generalization properties of this method remain rather elusive due to the non-convex problem settings that it is being utilized. This work aim illustrates first in a linear and convex, then in a non-linear and non-convex supervised learning setting. Throughout the work compared with the Bayes error is used as a baseline and models with convex losses are used as benchmarks with or without meta-information for the problems being considered.

***K*eywords** meta-learning · expected error · biased ridge · model-agnostic meta-learning

1 Introduction

Learning-to-learn(meta-learning) is a hot research field that treats the learning of the traditional machine learning task as the learning problem. The utility that comes with this specific learning paradigm enhances the capability of the machine-related prediction tasks employing increased efficiency of data utilization resulting from the context detection capability of the algorithms presented under this paradigm.

Model-Agnostic Machine Learning (MAML)[1] is arguably one of the most impactful meta-learning algorithms that is proposed recently. The reason for this method to gain this much traction can be associated with it being non-parametric in the meta-level. In a broad sense, this algorithm seeks to find an intermediate model in the environment of tasks. The way to obtain this model is to look at the future loss of possible gradient steps from the tasks observed. This makes MAML highly attractive for the gradient-based methods as the implementation effort that goes into any model that relies on gradients is minimal. Given the wide-spread use of gradient descent in machine learning applications, the research avenue that [1] opens up is quite wide.

Although, there are multiples of works that built upon the ideology presented in [1] (*e.g.*[2, 3, 4, 5, 6, 7]) to improve upon its deficiencies. The claimed optimization for generalization [3, 1] aspect for the adapted task is investigated to a limited extent. Especially compared to models single task learning models without meta-learning on any given task. Similarly, this paper aims to investigate the generalization of MAML for a given task to find out to what extent the claims regarding generalization are valid, to what extent a gradient-based meta-learner that tries to find a warm starting point for given task distribution. Hence, the main of this paper is to provide an empirical study of a gradient-based meta-learning algorithm. The main research questions investigated in this paper are:

- *What is the extent of MAML's generalization capabilities?* We will compare the single task learning models (*e.g.*Least-Squares and Ridge Regression) with the Bayes Error as the baseline to see how much benefit does MAML provides in terms of generalization compared to other methods in various problems.
- *Is MAML algorithm, model agnostic?* Through considering linear models performance of MAML is investigated in a convex problem setting. We try to investigate if there is performance gain for linear problem setting, where less parameters are involved.

2 Related Work

Since its publications, MAML [1] opened a wide range of research path that tries to understand gradient-based meta-learning, which aims to find an intermediate model to be used for adaptation at a later stage. There is both empirical and theoretical work dedicated to the understanding of MAML.

The empirical works try to understand and improve the MAML algorithm. For example, the sensitivity of MAML to architectural details and the difficulty in the training of the MAML algorithm is addressed and tackled in [7] under the name of MAML++. This method tries to provide a much more stable algorithm by overcoming the gradient instability, the absence of batch statistics accumulation, the shared bias per layer, constant learning rates utilized in the algorithm. Although, its training difficulties the generalization capabilities are empirically investigated in [6] and concluded that the generalization to new tasks can be related to the similarity between the trajectories encountered during meta-training and the adaptation procedures. Moreover, in the same work, L_2 regularization like regularization for the inner loop is proposed for improved generalization performance.

Aside from empirical studies, some theoretical work tries to understand the convergence and the generalization of MAML over the past years. However, since the theoretical investigation of the over-parametrized neural network is quite cumbersome, most of the effort goes into understanding this algorithm in much milder settings with strong assumptions. For instance, in [8] various gradient-based meta-learning algorithms including MAML and its derivatives are investigated for convex optimization problems and their usefulness is shown compared to single-task learning. In addition, negative learning rates in the inner loop are shown to be optimal during the meta-training stage, theoretically, in [9] for a mixed linear regression problem. Moreover, in [10] the effect of task distribution is investigated for linear regression problems and show that when tasks seen in training are similar to to the ones seen at adaptation step. Finally, in [11] the generalization of MAML is investigated from the algorithmic stability and generalization bounds perspective and concluded that the MAML generalizes well even to an unseen task if the training and test task distributions are sufficiently close.

Another line of work tries to come up with other meta-learning scenarios involving convex settings by construction, which is a quite rare setting in meta-learning paradigm. In [12, 13] meta-learning models inspired by the biased regularization works [14, 15], where bias is tried to be learned from the task environment by minimizing the *Transfer Risk* (Expected Loss), is proposed. Due to the convex nature of the problem, the theoretical foundation is also provided for the proposed models. For learning-to-learn and continual learning settings. On top of this work, [16] investigate the need for *train-test* split for the same models and concludes that the *train-train* model achieves strictly better generalization performance for structured tasks in the setting of learning around a common mean problem presented in [12]. However, the comparision of this method to other meta-learning methods (*e.g.*MAML) are left out.

Our Contribution: The *Expected Loss* investigation for most of the meta-learning methods are left out for various reasons. We try to investigate this for two different meta learning approaches. On one hand, the loss MAML [1] is trying to optimize is the loss that the model would make given a batch of tasks obtained from the environment of certain task distribution. The model parameters update is done by looking at the possible loss for each task if the model parameters are changed for a certain task. On the other hand, the methods proposed by [12] are trying to optimize for the so-called *Transfer Risk* over for the task distribution. The main contribution of this paper is to investigate in linear and nonlinear settings the average performance of the MAML algorithm by looking at its expected loss, which will then be compared to individual learning task performance where there is no information coming from the task environment. The reason that the expected loss of MAML is not investigated can be speculated to be the computational burden that it bestows upon the problem. Here, for the linear problem case, this problem is tried to be elevated utilizing NUMBA [17], which can create compiled code for Python. However, due to the inflexibility of the NUMBA, pure PyTorch [18] implementation, which is slower, is used for the nonlinear problem with a lesser degree of fidelity.

3 Problem Setting

Throughout this work uppercase bold letters (*e.g.*$\mathbf{X}$), lowercase bold letters (*e.g.*$\mathbf{x}$), and lowercase letters (*e.g.*x) are used for matrices, vectors and scalars respectively. Moreover, the vectors are assumed to be stored in columns. Finally, the $\mathbf{I}_D$ represents a $D \times D$ identity matrix. Moreover, the $\mathbf{1}_D$ and $\mathbf{0}_D$ represents $D \times 1$ vector of ones and zeros respectively.

3.1 Learning Problems

3.1.1 Linear Problem

Consider the conventional linear regression problem in $\mathbb{R}^D$ is given by

$$y = \mathbf{x}^{\mathrm{T}}\mathbf{a} + \varepsilon \tag{1}$$

where, $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{a} \in \mathbb{R}^D$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Assuming that the each realization of a scale term $\mathbf{a}$ corresponds to a task $\mathcal{T}$ observed in the environment and each set of observed N input ($\mathbf{x}$) and its corresponding label (y) is represented by a dataset $\mathcal{Z}_j := (\mathbf{x}, y)_{i=1}^N$.

3.1.2 Nonlinear Problem

Consider a nonlinear regression problem in $\mathbb{R}^D$ is given by

$$y = \sin(\mathbf{x} + \boldsymbol{\phi})^{\mathrm{T}}\mathbf{a} + \varepsilon \tag{2}$$

where, $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{a} \in \mathbb{R}^D$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Assuming that the each realization of scale term $\mathbf{a}$ and $\boldsymbol{\phi}$ corresponds to a task observed in the environment $\mathcal{T}$ and each set of observed N input ($\mathbf{x}$) and its corresponding label (y) is represented by a dataset $\mathcal{Z}_j := (\mathbf{x}_i, y_i)_{i=1}^N$.

3.1.3 General Problem Setting

For both problems presented in Sections 3.1.1 and 3.1.2 sample distribution is given by $p_{\mathcal{Z}}$ for a given $\mathcal{T}_k$ and the task distribution is represented by $p_{\mathcal{T}}$. A model parameterized by $\bar{\mathbf{w}}$[1] is represented by $\mathcal{M}(\mathbf{x}, \bar{\mathbf{w}}) : \mathbf{x} \to y$. An estimator that is trained with $\mathcal{Z}_j$ that is obtained from the $\mathcal{T}_k$ is represented by $\hat{\mathcal{M}}(\mathbf{x})$. The discrepancy between the prediction of the estimator $\hat{\mathcal{M}}$ and y is measured in terms of squared loss $\mathcal{L} := (\hat{\mathcal{M}}(x) - y)^2$. The main loss that this paper tries to investigate is the *Expected Error* of an estimator $\hat{\mathcal{M}}$ over the $p_{\mathcal{T}}$. Then the expected error is represented as

$$\mathcal{E} := \iiint (\hat{\mathcal{M}}(x) - y)^2 p(\mathbf{x}, y) p_{\mathcal{Z}} p_{\mathcal{T}} d\mathbf{x} dy d\mathcal{Z} d\mathcal{T}. \tag{3}$$

For the defined expected error and the problem definitions, the *Bayes Error* is given by σ^2 that is coming from the noise term, which represents a model that is the perfect estimator.

3.1.4 Experimental Assumptions

For all the problems the input distribution is given by $p_{\mathbf{x}} \sim \mathcal{N}(0, k\mathbf{I},)$ where k is a parameter for the variance of the inputs. For the linear problem the $p_{\mathcal{T}} := p(\mathbf{a}) \sim \mathcal{N}(m\mathbf{1}_D, c\mathbf{I}_D,)$ and for nonlinear problem the task distribution takes the form of a joint distribution $p_{\mathcal{T}} := p(\mathbf{a}, \boldsymbol{\phi})$ where $p_{\mathbf{a}} \sim \mathcal{N}(\mathbf{1}_D, c_1\mathbf{I}_D,)$ and $p_{\boldsymbol{\phi}} \sim \mathcal{N}(\mathbf{0}_D, c_2\mathbf{I}_D,)$

3.2 Models

3.2.1 Single Task Learning Models

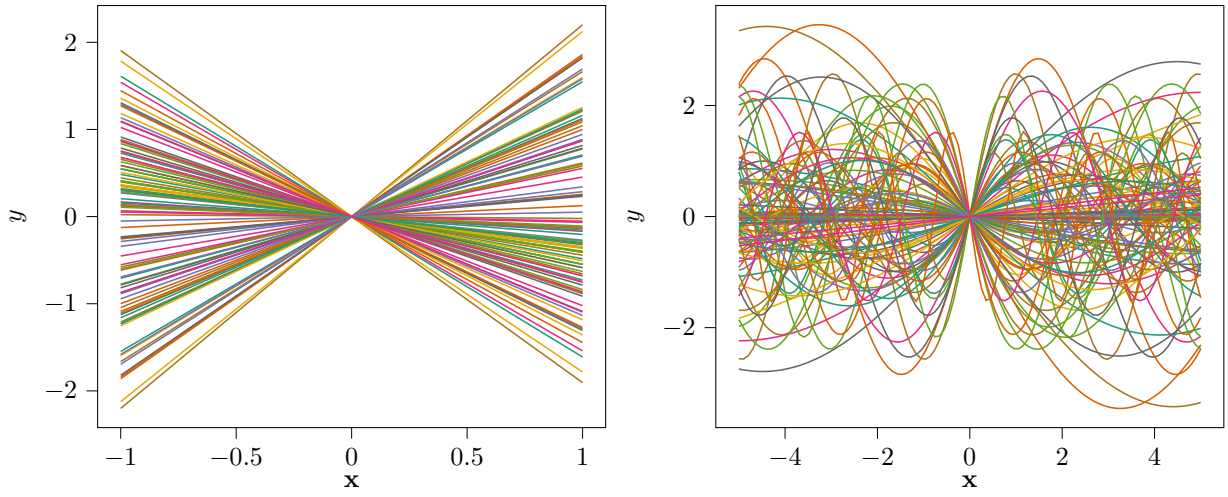
Assuming a linear model in the form of $\mathcal{M}(\mathbf{x}, \mathbf{w}, b) := \mathbf{x}\mathbf{w} + b$ or $\mathcal{M}(\mathbf{x}, \bar{\mathbf{w}}) := \bar{\mathbf{x}}\bar{\mathbf{w}}$ with $\mathbf{x} \in \mathbb{R}^{1\times D}$, $\mathbf{w} \in \mathbb{R}^{D\times 1}$, $\bar{\mathbf{x}} \in \mathbb{R}^{1\times D+1}$ and $\bar{\mathbf{w}} \in \mathbb{R}^{D+1\times}$ where $\bar{\mathbf{w}} := [\mathbf{w}, b]^{\mathrm{T}}$ and $\bar{\mathbf{x}} := [\mathbf{x}, 1]$. The optimum collection of parameters ($\hat{\bar{\mathbf{w}}}$) for different models are obtained as follow:

Linear Estimator is given by the least-squares solution, $\hat{\bar{\mathbf{w}}} := (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$, where $\mathbf{X} \in \mathbb{R}^{N\times D}$ is the design matrix where the observed input data is stored in rows.

Ridge Estimator is given by $\hat{\bar{\mathbf{w}}} := (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$ which is obtained by minimizing the squared loss with the additional term of $\lambda||\bar{\mathbf{w}}||_2^2$. Thus, overall loss takes the form $\mathcal{L} + \lambda||\bar{\mathbf{w}}||_2^2$.

Kernel Ridge Estimator is given by $\bar{\mathbf{w}} = \mathbf{X}^{\mathrm{T}}\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} := (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}y$ where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the *Gram Matrix* obtained by replacing $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ inner product by a kernel $\kappa(\mathbf{X}, \mathbf{X})$. Then, the prediction of the estimator takes the form $\hat{\mathcal{M}}(\mathbf{x}^*, \bar{\mathbf{w}}) = \boldsymbol{\alpha}^{\mathrm{T}}\kappa(\mathbf{x}^*, \mathbf{X})$ where $\mathbf{x}^* \in \mathbb{R}^{D\times 1}$.

[1] Bar on top of the parameters are used to indicate the bias terms inclusion.



(a) $y = \mathbf{x}^{\mathrm{T}}\mathbf{a}$ (b) $y = \sin(\mathbf{x} + \boldsymbol{\phi})^{\mathrm{T}}\mathbf{a}$

Figure 1: 100 sample tasks drawn from $p_{\mathcal{T}}$ for both linear ($m = 0$ and $c = 1$) and nonlinear ($c_1 = 1$ and $c_2 = 1$) problems.

Gradient Descent Estimator for the Linear Model for a given number of iterations n_{iter} the gradient descent estimator is given by $\mathbf{w}_{j+1} = \mathbf{w}_j - \eta \sum_i^N \mathbf{x}_i(\hat{\mathcal{M}}(\mathbf{x}, \bar{\mathbf{w}}_j) - y_i)_{j=0}^{n_{iter}-1}$ and $b_{j+1} = b_j - \eta \sum_i^N (\hat{\mathcal{M}}(\mathbf{x}, \bar{\mathbf{w}}_j) - y_i)_{j=0}^{n_{iter}-1}$.

3.2.2 Meta Learning Models

Model-Agnostic Meta-Learning (MAML) aims to obtain an intermediate model that can generalize well after adaptation to a dataset $\mathcal{Z}$ observed from a new and unseen task $\mathcal{T}_i$ drawn from $p_{\mathcal{T}}$ where the number of training points K and the number of iterations n_{iter} is limited. The general procedure for supervised learning problems is given in Algorithm 1.

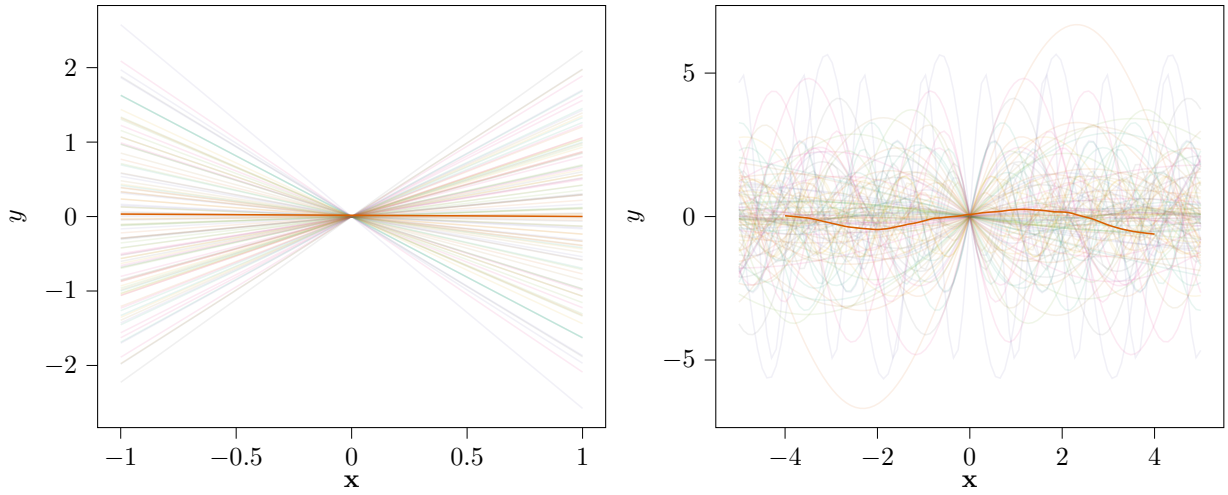
Data: $p_{\mathcal{T}}$, α, β
Result: Intermediate Model $\mathcal{M}(\bar{\mathbf{w}}_{meta})$
initialize $\bar{\mathbf{w}}$ randomly;
while *not done* **do**
  sample a batch of tasks $\mathcal{T}_i$ from $p_{\mathcal{T}}$
  forall $\mathcal{T}_i$ **do**
    Obtain future gradients: $\nabla_{\bar{\mathbf{w}}} \mathcal{L}_{\mathcal{T}_i}(\mathcal{M}(\bar{\mathbf{w}}))$ wrt. $\mathcal{Z}_K$
    Possible future parameters: $\bar{\mathbf{w}}_i' = \bar{\mathbf{w}}_i - \alpha \nabla_{\bar{\mathbf{w}}} \mathcal{L}_{\mathcal{T}_i}(\mathcal{M}(\bar{\mathbf{w}}))$
  end
  Update: $\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} - \beta \nabla_{\bar{\mathbf{w}}} \sum \mathcal{L}_{\mathcal{T}_i}(\mathcal{M}(\bar{\mathbf{w}}_i'))$
end

Algorithm 1: MAML[1] Algorithm

General Ridge Estimator is a version of *Ridge Estimator* with the capability to penalize towards a given vector instead of a zero vector. It is given by $\hat{\bar{\mathbf{w}}} := (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda \mathbf{I}_D)^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{y} + \lambda \mathbf{h})$ by minimizing the squared loss with the additional term $\lambda ||\bar{\mathbf{w}} - \mathbf{h}||_2^2$, where $\mathbf{h} \in \mathbb{R}^{D \times 1}$. Then, the overall loss takes the form $\mathcal{L} + \lambda ||\bar{\mathbf{w}} - \mathbf{h}||_2^2$.

4 Results and Discussion

This section is dedicated to the expected error results of certain meta-learning models and some individual task learning models with the aim to see their performance differences certain scenarios. The utilized models that have meta information are;



(a) Linear Problem

(b) Nonlinear Problem

Figure 2: Visualizing the MAML intermediate model. 100 sample tasks drawn from $p_{\mathcal{T}}$ for both linear ($m = 0$ and $c = 1$) and nonlinear ($c_1 = 2$ and $c_2 = 2$) problems shown transparent and intermediate model trained (obtained from MAML algorithm) for 1D cases of the experimentation given with solid dark orange line.

- **GD**: corresponds to gradient descent with n_{iter} with the adjustable parameters obtained from the mean of the tasks $\mathbb{E}[p_{\mathcal{T}}]$. (*e.g.*considering the linear problem the with $p_{\mathbf{a}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the initial starting point of the linear model is $\bar{\mathbf{w}} = \mathbf{0}$.)
- **MAML**(for linear problem): corresponds to gradient descent with n_{iter} with the adjustable parameters obtained from the mean of the tasks $\mathbb{E}[p_{\mathcal{T}}]$ with small perturbation $\delta \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I})$. (In [19] it is shown that the learning rate can have a regularizing effect in a similar fashion one might conjecture that the learning rate and the stopping point of the meta-training phase can have a similar effect. Moreover, in a more realistic scenario the stopping point is rather arbitrary. Thus, this model is especially interesting to investigate.)
- **MAML**(for nonlinear problem): MAML algorithm trained with the information given in [1] for the sinusoidal regression problem. It should be noted that network architecture and all the other hyper-parameters are taken from the paper exactly.
- **General Ridge**: unlike the conventional regularization additional $\mathbf{h}$ is needed for this method which is taken to be the $\mathbb{E}[p_{\mathcal{T}}]$. In [12], a framework is presented on how to find the $\mathbf{h}$, however, in our experimentation we assume it to be known a priori. This is similar to GD and MAML models presented above.

Models without the meta information are;

- **Linear**: standard least squares solution.
- **Ridge**: standard least squares solution with $L_2 - regularization$.
- **random GD**: gradient descent with n_{iter} with the adjustable parameters starting from a random initialization.
- **Kernel Ridge**: kernalized (with Radial Basis Function Kernel)

It should be noted that the hyper-parameters of the utilized models, if there are any, are not tuned, properly as it would increase the computational burden of the problem to another level. However, a simple grid search is employed with 20 different values and only the one with the lowest mean expected error over the parameter under investigation is presented for the results.

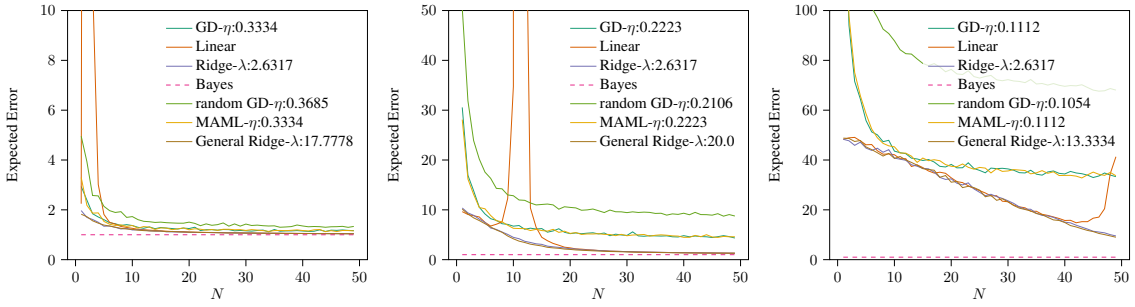
4.1 Linear Problem

The linear problem introduced in Section 3.1.1 has the parameters controlling the dimensionality D, number of training samples N, number of gradient steps n_{iter}, the task variance c and task mean m, and the variance of the input samples k. For the sake of brevity only some of the parameters are discussed in this section. Unless the parameter in configuration is under investigation, the default values are utilized. And, the default values are given as,

- $\sigma = 1$,
- $m = 0$,
- $k = 1$,
- $c = 1$,
- $n_{iter} = 1$.

Moreover, the number of tasks drawn ($N_{\mathcal{T}}$), and dataset draws ($N_{\mathcal{Z}}$) for approximating the expected error given in Equation 3 are taken to be 100 each. Finally, the test set size is taken as 1000.

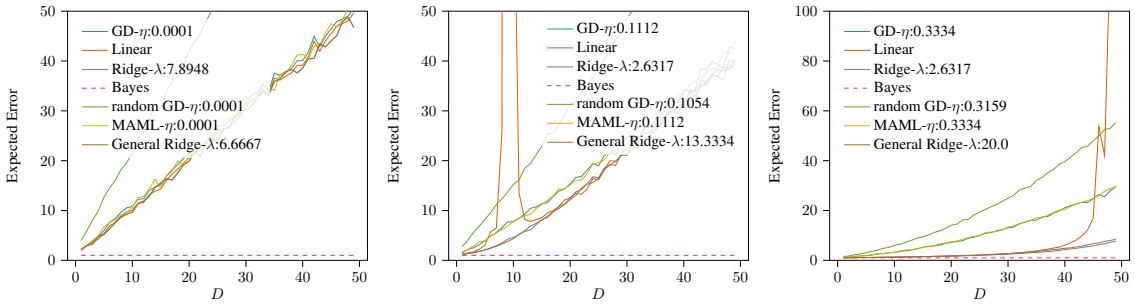
Effect of Training Samples N: The results of this experiment can be found in Figure 3 for increasing problem dimensionality. It can be seen that the "Linear" model suffers from singularities, however, it is able to have comparable performance over all the selected problem dimensionalities. As one might expect as the dimensionality increases the difference between the models with analytical optimum and gradient descent utilizing methods. Moreover, for the increasing training samples case the Ridge regression variants perform much better as for all the cases they Converge towards the "Bayes Error". However, for the gradient descent variant models whether there exists task information (*e.g.*MAML, GD) are unable converge towards the Bayes error. Which can be attributed to the regularizing effect of the gradient steps (n_{iter}) allowed for the models. Overall, the improvement that the addition task related information brings to the gradient based models, as the "random GD" model is orders of magnitude higher expected error. Although, task information inclusion decreases the Expected Error as the number of gradient step limitation hinders the gradient based models capability to decrease the expected error further.



(a) $D = 1$ (b) $D = 10$ (c) $D = 50$

Figure 3: The Expected Error for changing number of training samples for various dimensional problems.

Effect of Dimensionality D: These results are quite similar to the ones obtained with the experiments investigating the effect of training samples. It can be observed from Figure 4, aside from singularities the Linear model variants yield lower expected error compared to the gradient descent variants and this gap increases as the dimensionality of the problem or the number of training samples increases. Looking at Figure 3b it can be observed that for number of training samples around the dimensionality of the problem Ridge variant models perform much better, but as the dimensionality of the problem increases the gap between the performances of the Ridge variants and the gradient descent variants.



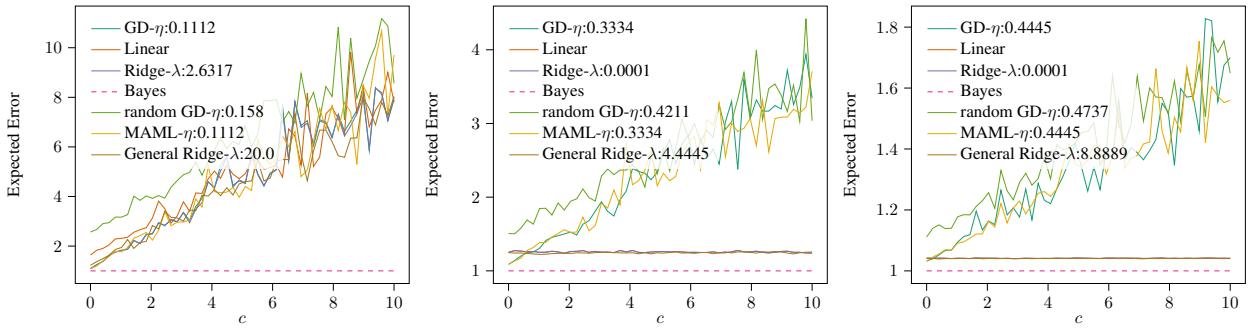
(a) $N = 1$ (b) $N = 10$ (c) $N = 50$

Figure 4: The Expected Error for changing number of dimensions for various number of training points.

Table 1: Mean Expected Error for the range $c : [0, 1]$ range with various gradient steps for the MAML and GD models with $\eta = 0.334$. Note that only $D = 1$, $N = 10$ case (see Figure 5b) is presented.

	n_{iter}									
	1	2	3	4	5	6	7	8	9	10
GD	1.2152	**1.1936**	1.2048	1.2140	1.2268	1.2390	1.2604	1.2970	1.3825	1.5748
MAML	1.2132	**1.1938**	1.2067	1.2171	1.2318	1.2476	1.2773	1.3330	1.4622	1.7556

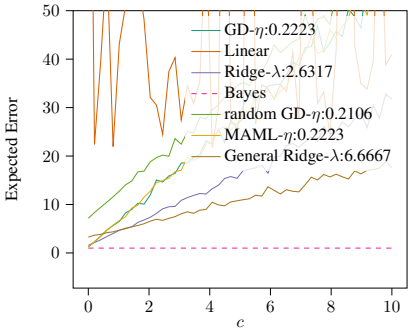
Effect of Task Variance c: The results of increasing task variance for various problem dimensions and various number of training samples can be found in Figure 5. The most obvious observation is that, for all the models that utilizes gradient descent, expected error increases, whereas the Linear model and Ridge variants are only effected by this phenomenon only for problem dimensionality $D \geq N$. In the light of this, observation another important result is the fact that for $N \geq D$ for small task variance the gradient descent variants, other than randomly initialized model, the expected error is lower than the Ridge variants. Although this performance diminishes with increasing problem dimensionality and the increasing number of training points. It is interesting to see a better performance from just one gradient step. That is why an extra mini-experimentation is done for the GD and MAML models to investigate if there is a performance improvement with the additional gradient steps. The results of this experimentation is given in Table 1. It is observed from the table that there exists point at which the gradient steps are hurting the expected error one would get in this range after the second gradient step. Then, it can be conjectured that the number of gradient steps have a regularizing effect for the task distributions with small variance. Although, this surprising performance of MAML like algorithms, the performance of Ridge variants is much stable and performs reasonably better than gradient based methods for $N \geq D$.



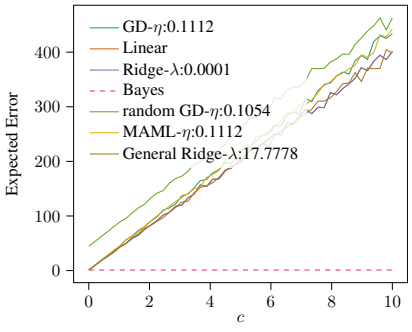
(a) $D = 1$, $N = 1$

(b) $D = 1$, $N = 10$

(c) $D = 1$, $N = 50$



(d) $D = 10$, $N = 10$

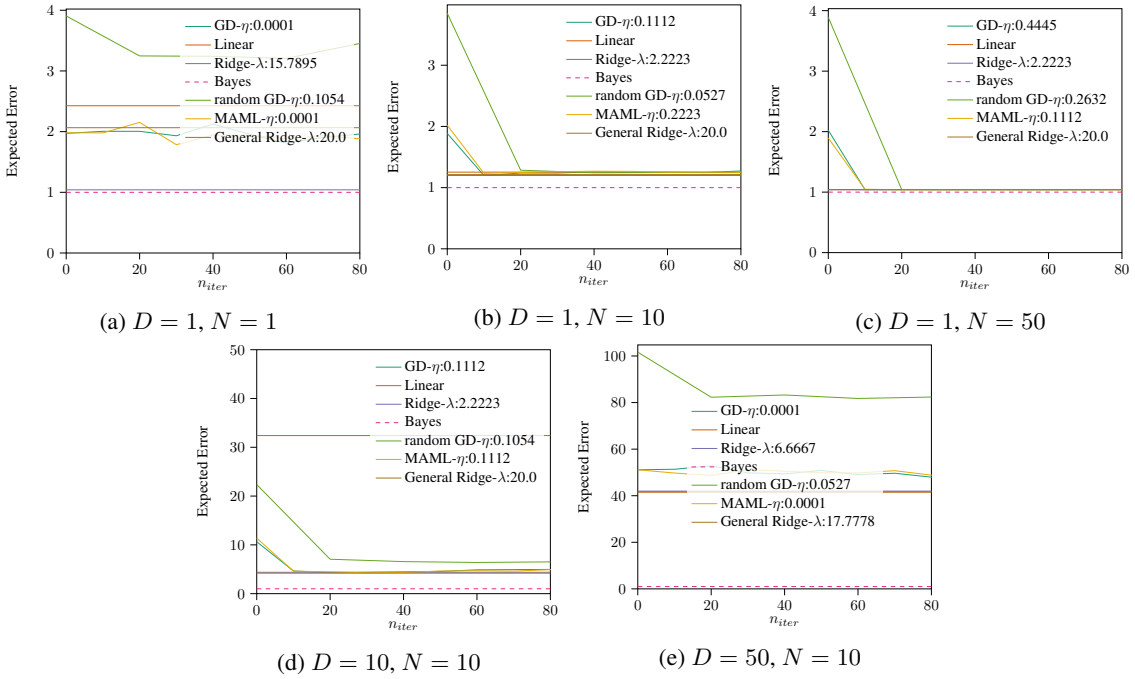


(e) $D = 50$, $N = 10$

Figure 5: The Expected Error for changing number of training samples for various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 5a, 5b, 5c and the effect of increasing dimensionality can be seen by looking at Figures 5b, 5d, 5e.

Effect of Number of Gradient Steps n_{iter}: Looking at the interesting results observed from the task variance c the effect of number of gradient steps taken become more relevant. These results can be seen in Figure 6. It can be observed from Figure 6a that for low number of training samples the gradient steps taken has little to no influence. But as the number of training samples increase for a given problem dimensionality the effect n_{iter} on the expected error gets much prominent. It is evident that compared to single task learning with gradient descent from a random initialization starting

from a more informative point (*e.g.*near the task mean) decreases the number of gradient steps for the convergence. Moreover, for the $D = N$ case it even improves generalization after convergence too (see Figures 6d and 6a). Overall, it can be observed that the increasing n_{iter} converges towards the Ridge model variants with the exception of $D = 1$ and $N = 1$ case.



(a) $D = 1$, $N = 1$ (b) $D = 1$, $N = 10$ (c) $D = 1$, $N = 50$

(d) $D = 10$, $N = 10$ (e) $D = 50$, $N = 10$

Figure 6: The Expected Error for changing number of gradient steps n_{iter} with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 6a, 6b, 6c and the effect of increasing dimensionality can be seen by looking at Figures 6b, 6d, 6e.

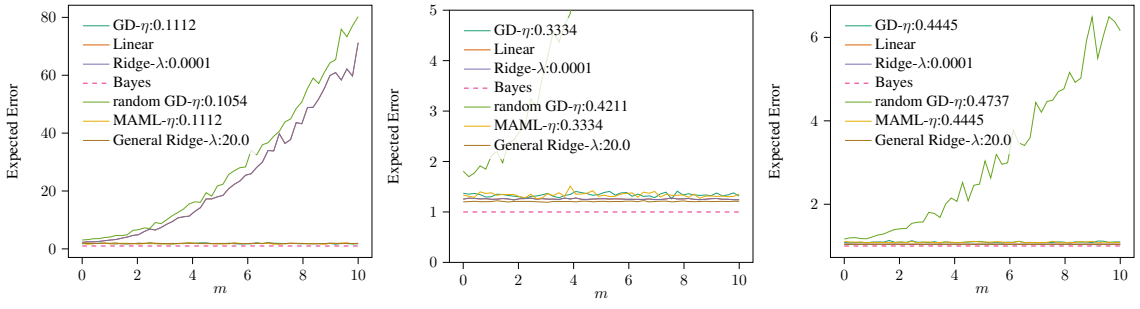
Effect of Task Mean m: The results can be seen in Figure 7. The most important observation from this experimentation is that the Ridge model has increasing expected error for the cases of $N \leq D$ cases (see Figures 7a, 7d and 7e) and mostly the best λ from the trials is found to be the lowest value, which makes the Ridge model behave same as the Linear model. However, General Ridge model that has prior information regarding the task centroid does not suffer from this problem. Furthermore, other models which have prior task information does not seem to be effected from the task mean shifting in the task space, as expected. Again, superiority of including information from the task space is evident as the conventional regularization cannot deal with the changing task distribution mean for $N \leq D$.

4.2 Nonlinear Problem

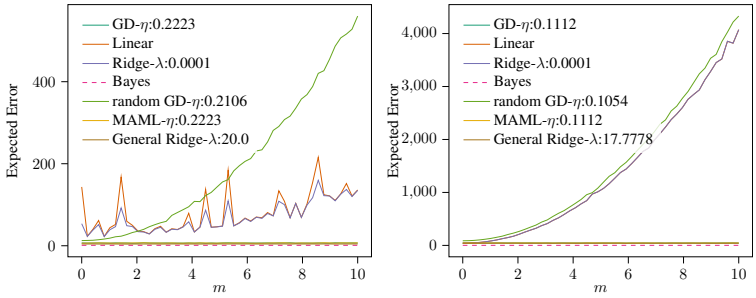
The nonlinear problem introduced in Section 3.1.2 has the parameters controlling the dimensionality D, number of training samples N, number of gradient steps n_{iter}, the task variances and means m_1 and m_2, c_1 and c_2, and the variance of the input samples k. For the sake of brevity only some of the parameters are discussed in this section. Unless the parameter in configuration is under investigation, the default values are utilized. And, the default values are given as,

- $\sigma = 1$,
- $m_1 = 1$,
- $m_2 = 0$,
- $c_1 = 2$,
- $c_2 = 2$,
- $k = 1$,
- $n_{iter} = 5$.

Moreover, the number of tasks drawn ($N_{\mathcal{T}}$), and dataset draws ($N_{\mathcal{Z}}$) for approximating the expected error given in Equation 3 are taken to be 50 each. Finally, the test set size is taken as 1000.



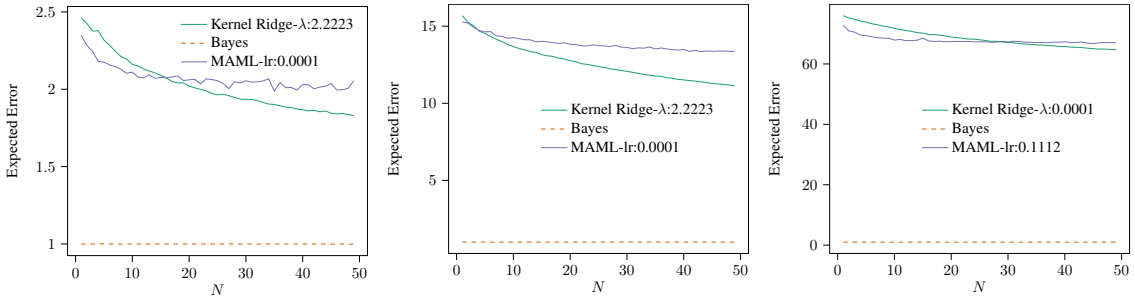
(a) $D = 1, N = 1$ (b) $D = 1, N = 10$ (c) $D = 1, N = 50$



(d) $D = 10, N = 10$ (e) $D = 50, N = 10$

Figure 7: The Expected Error for changing task mean m with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 7a, 7b, 7c and the effect of increasing dimensionality can be seen by looking at Figures 7d, 7e.

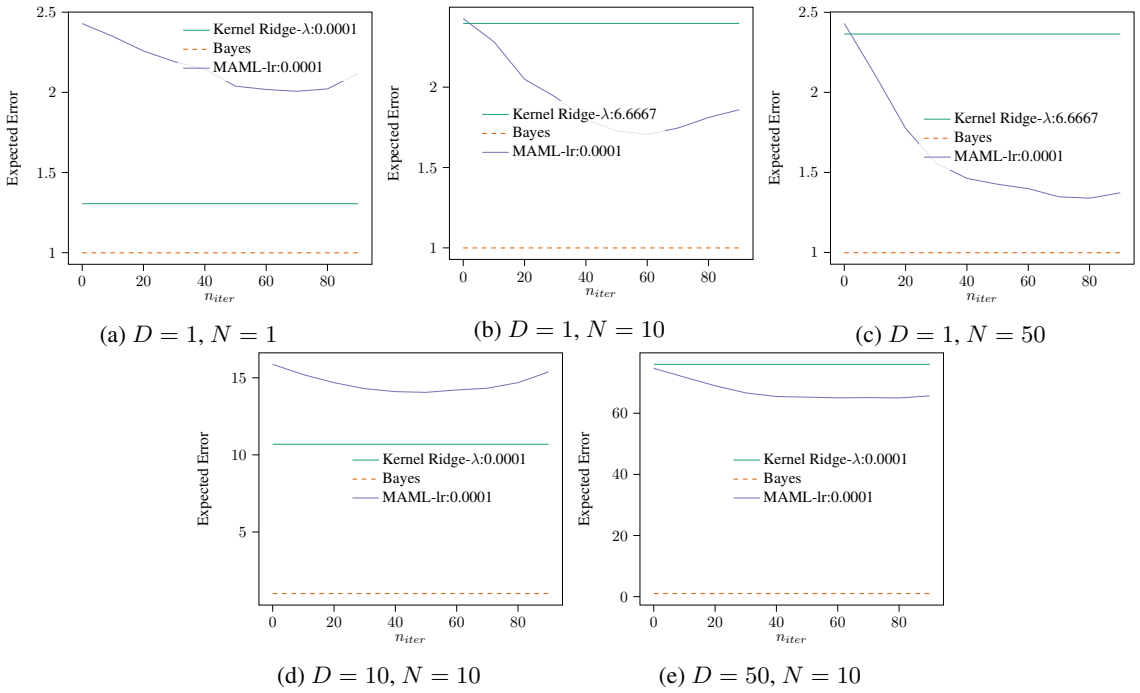
Effect of Training Samples N: By looking at Figure 8 it can be seen that for all the given dimensionalities there exists a training sample amount where the expected error of the Kernel Ridge model is higher than the MAML. The most notable behaviour for this experiment is that Kernel Ridge models tends towards the Bayes error while the MAML converges to a certain value and stays there. This might be again attributed to the fact that the number of gradient steps limitation. Although the expected error is quite high for increasing dimensionality, the results obtained for $D = 1$ (Figure 8a) is still and effective result that shows that for a small number of training samples with limited gradient steps MAML will outperform a convex model Kernel Ridge.



(a) $D = 1$ (b) $D = 10$ (c) $D = 50$

Figure 8: The Expected Error for changing number of training samples for various dimensional problems.

Effect of Number of Gradient Steps n_{iter}: Above presented learning curves, beg the investigation of the effect of n_{iter}. As it can be seen from Figure 9a, the single realization of Kernel Ridge can have lower expected error for an extreme value of 1 training sample. However, as the number of training samples increase for a given problem dimensionality MAML model starts showing a better expected error. However, the lowest expected error is not realized for a fairly large n_{iter}. Moreover, it can be observed that for $N \leq D$ Kernel Ridge can achieve lower expected error, and for all the other cases one might find a better MAML model given that sufficient gradient steps are allowed. Finally, it should be noted that the number of gradient steps required to perform better than Kernel Ridge is fairly low.



(a) $D = 1$, $N = 1$ (b) $D = 1$, $N = 10$ (c) $D = 1$, $N = 50$

(d) $D = 10$, $N = 10$ (e) $D = 50$, $N = 10$

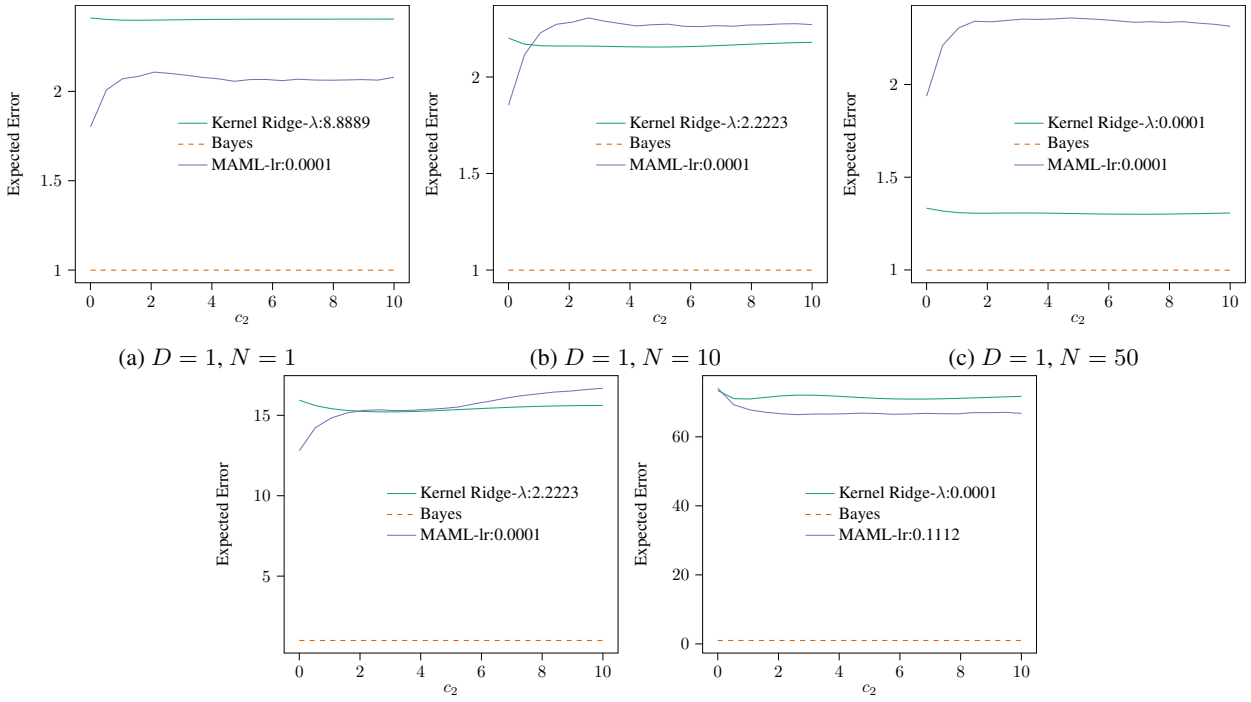
Figure 9: The Expected Error for changing number of gradient steps n_{iter} with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 9a, 9b, 9c and the effect of increasing dimensionality can be seen by looking at Figures 9b, 9d, 9e.

Effect of Phase Task Variance c_2: Remembering that the task variance effect for the linear problem had some interesting properties where even a single gradient step resulted in better expected error values. One might wonder if that is the case for the nonlinear problem as well. As can be seen in Figure 10 a similar effect is observed for the nonlinear problem too for small training sample size N values. For problem dimensionality of 1 and 10 there is a clear expected error rise between task variance $[0, 2]$ as shown in Figures 10a,10b,10c and 10d. Which indicates that the "MAML" even with a limited number of gradient step provide a clear benefit compared to a model that has analytical solution. However, mostly this superiority vanishes as the task variance increases as the increased values of variance lead to worse expected error compared to "Kernel Ridge".

4.2.1 Overall Discussion of the Results

It is observed that the MAML in linear and nonlinear problem settings might provided extra generalization performance although the gradient steps are limited. It should be noted that, this conjecture is valid only under certain setting. For instance, for the linear problem it is found out that the MAML variants perform better only when the task distribution variance is small enough and for almost all the other cases, limitation of the number of gradient step results in hindered performance in terms of generalization. Furthermore, there exists a clear optimum for number of gradient steps taken for adaptation (n_{iter}). Hence, in some cases the MAML variant methods can perform worse than single task learning methods on expectation. In the nonlinear problem setting, a single task learning setting is found to be competitive with a meta-learning approach especially for the increasing dimensionality the difference between the meta-learning algorithm and single task learning algorithm tends to small in expectation.

Additional Remarks: It is observed that the generalization improvement for the MAML variants that uses a few gradient steps after observing a few data from the task we are interested in. This validates some of the findings in [11], where it is found out that under strong convexity assumptions if the training and test task distributions are close enough there is generalization improvement. Moreover, this finding is also observed in the non-convex setting. Considering the nonlinear problem setting number of gradient steps as there is a clear optimum expected error for n_{iter} and the same case found to be the case for the linear problem as shown in Table 1. Another important comparison is the comparison between two different meta-learning approaches in linear problem setting (MAML [1] and biased regularization based meta learning [20].) It can be seen that given an extreme limitation on the gradient step in most of the cases biased regularization outperforms MAML variant models. Only in the aforementioned case of small tasks variance MAML



(a) $D = 1, N = 1$ (b) $D = 1, N = 10$ (c) $D = 1, N = 50$

(d) $D = 10, N = 10$ (e) $D = 50, N = 10$

Figure 10: The Expected Error for changing number of training samples for various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 10a, 10b, 10c and the effect of increasing dimensionality can be seen by looking at Figures 10b, 10d, 10e.

variant models provided and improvement over the expected error. Finally, in the linear problem setting the effect of the stopping point is investigated by means of starting from the exact mean of the tasks compared to the start in the region of the optimum. It can be seen from all the linear problems that there is no observable performance gain or loss regarding the starting point of the MAML variant methods.

Additional experimentation results for σ and k for linear case and σ and c_1 can be found in the Appendix.

5 Conclusion

Upon our investigation, it is found empirically that meta-information about the task-space can help the generalization performance in linear and nonlinear problem settings. It is found out that the MAML variant methods are superior under certain conditions. It is observed that the gradient-based meta-learning method MAML is only capable performing better only when the task distribution variance is small, compared to regularization-based meta-learning models. Moreover, it is shown that limiting gradient descent steps taken can be beneficial due to the regularizing effect in both linear and nonlinear problem setting.

Comparing the meta-regularization information utilization via biased regularization provided in [12] has clear advantages in linear problem setting, when compared to meta-information inclusion via training an intermediate model presented in [1]. Looking at the performance of the biased regularization (*e.g.*General Ridge model) observation including the single task learning performance of the Kernel Ridge regression for the selected investigations, begs the question; "If we can, somehow, incorporate information regarding the task distribution, can we come up with a simpler kernelized and convex model to achieve a more competing method?". Investigation of this type of method is especially interesting for the reasons of explainability and the ease of theoretical understanding compared to models which have non-convex loss spaces.

Acknowledgments

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *34th International Conference on Machine Learning, ICML 2017*, 3:1856–1868, 2017.

[2] Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-Learning with Warped Gradient Descent. pages 1–28, 2019.

[3] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv*, pages 1–15, 2018.

[4] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. ITAML: An Incremental Task-Agnostic Meta-learning Approach. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 13585–13594, 2020.

[5] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. Task-robust model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, 2020-Decem:1–30, 2020.

[6] Simon Guiroy, Vikas Verma, and Christopher Pal. Towards Understanding Generalization in Gradient-Based Meta-Learning. 2019.

[7] Antreas Antoniou, Amos Storkey, and Harrison Edwards. How to train your MAML. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–11, 2019.

[8] Mikhail Khodak, Maria Fiorina Balcan, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:651–675, 2019.

[9] Alberto Bernacchia. Meta-learning with Negative Learning Rates. (2020), 2021.

[10] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. How Does the Task Landscape Affect MAML Performance? pages 1–47, 2020.

[11] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Generalization of Model-Agnostic Meta-Learning Algorithms: Recurring and Unseen Tasks. pages 1–23, 2021.

[12] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):10169–10179, 2018.

[13] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:2814–2843, 2019.

[14] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. Scalable greedy algorithms for transfer learning. *Computer Vision and Image Understanding*, 156:174–185, 2017.

[15] Ilja Kuzborskij and Nicolò Cesa-Bianchi. Nonparametric online regression while learning the metric. *Advances in Neural Information Processing Systems*, 2017-Decem(2):668–677, 2017.

[16] Yu Bai, Minshuo Chen, Pan Zhou, Tuo Zhao, Jason D. Lee, Sham Kakade, Huan Wang, and Caiming Xiong. How Important is the Train-Validation Split in Meta-Learning? 2020.

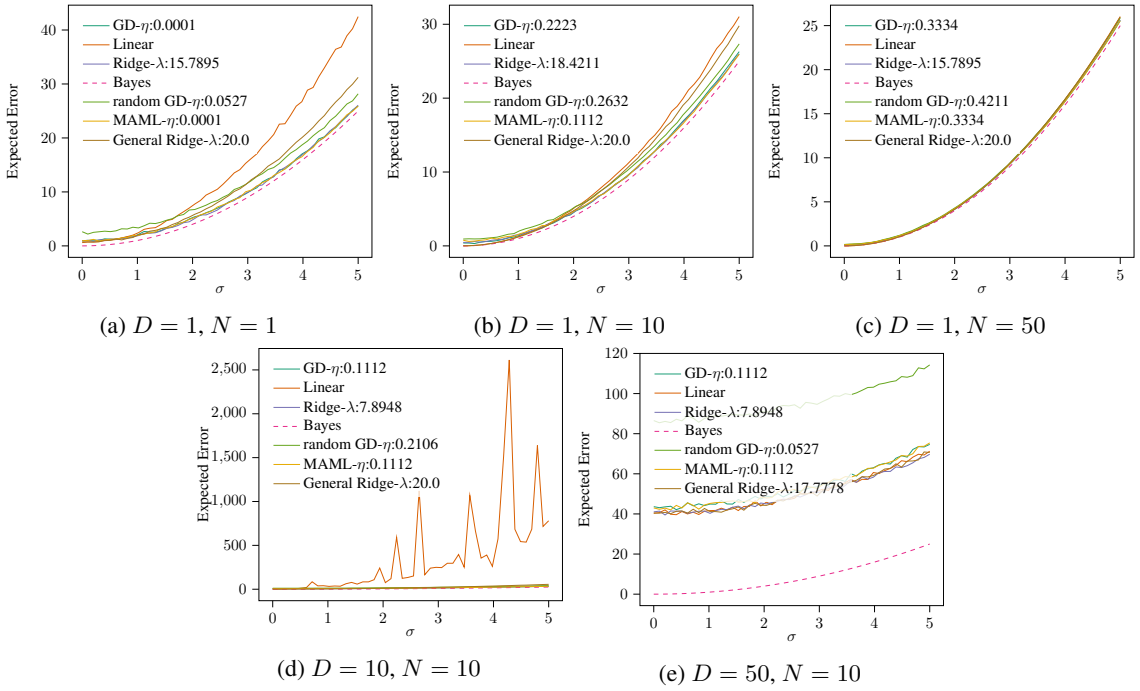
[17] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba. pages 1–6, 2015.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32(NeurIPS), 2019.

[19] Preetum Nakkiran. Learning Rate Annealing Can Provably Help Generalization, Even for Convex Problems. pages 1–9, 2020.

[20] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Incremental learning-to-learn with statistical guarantees. *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 1:457–466, 2018.

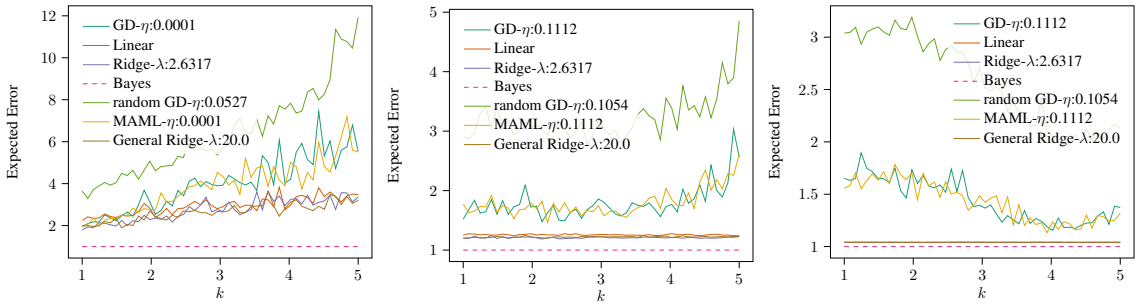
6 Appendix



(a) $D = 1$, $N = 1$ (b) $D = 1$, $N = 10$ (c) $D = 1$, $N = 50$

(d) $D = 10$, $N = 10$ (e) $D = 50$, $N = 10$

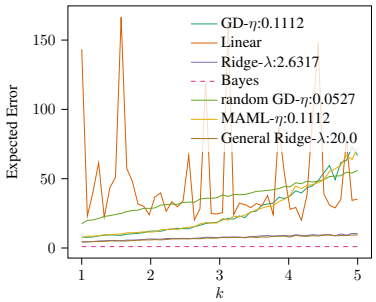
Figure 11: [**Linear Problem**] The Expected Error for changing noise standard deviation σ with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 11a, 11b, 11c and the effect of increasing dimensionality can be seen by looking at Figures 11b, 11d, 11e.



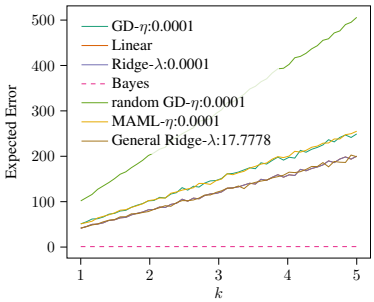
(a) $D = 1, N = 1$

(b) $D = 1, N = 10$

(c) $D = 1, N = 50$

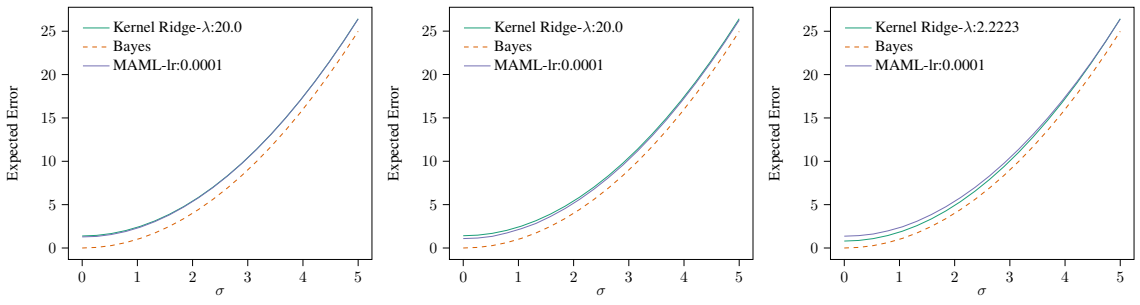


(d) $D = 10, N = 10$



(e) $D = 50, N = 10$

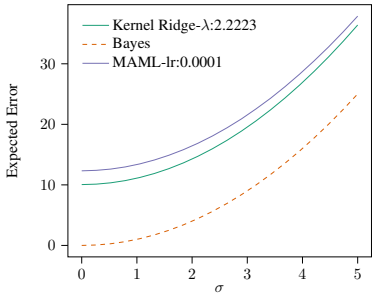
Figure 12: [**Linear Problem**] The Expected Error for changing noise standard deviation k with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 12a, 12b, 12c and the effect of increasing dimensionality can be seen by looking at Figures 12b, 12d, 12e.



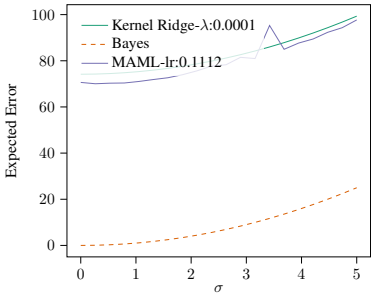
(a) $D = 1, N = 1$

(b) $D = 1, N = 10$

(c) $D = 1, N = 50$

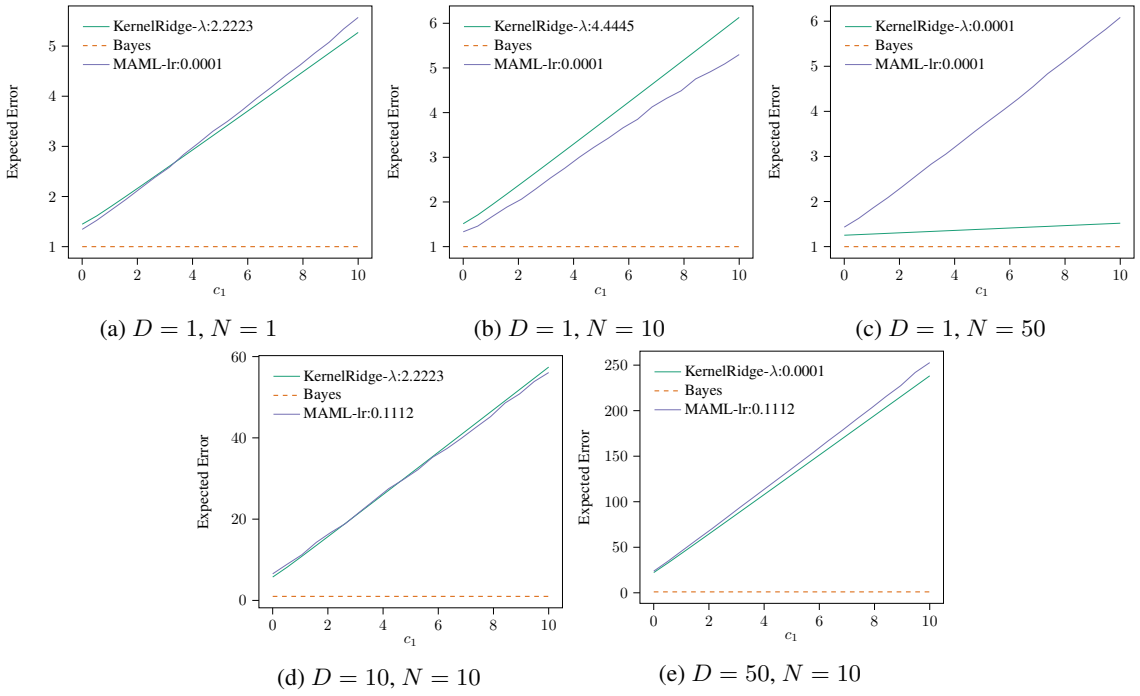


(d) $D = 10, N = 10$



(e) $D = 50, N = 10$

Figure 13: [**Nonlinear Problem**] The Expected Error for changing noise standard deviation σ with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 13a, 13b, 13c and the effect of increasing dimensionality can be seen by looking at Figures 13b, 13d, 13e.



(a) $D = 1$, $N = 1$ (b) $D = 1$, $N = 10$ (c) $D = 1$, $N = 50$

(d) $D = 10$, $N = 10$ (e) $D = 50$, $N = 10$

Figure 14: [**Nonlinear Problem**] The Expected Error for changing noise standard deviation c_1 with various training samples and various dimensional problems. For the given parameter the effect of increasing number of training samples can be seen by looking at Figures 14a, 14b, 14c and the effect of increasing dimensionality can be seen by looking at Figures 14b, 14d, 14e.