TRANSACTIONS OF THE
PATTERN RECOGNITION LABORATORY
Volume 00, Number 0, Pages 000–000
S 0002-9947(XX)0000-0

META-LEARNING WITH KERNEL RIDGE

Remark 1. This may be an oversimplification but here it goes... A conventional learning problem where the aim is to estimate a ground truth from several observations. In a meta-learning problem, we have a distribution of learning tasks (Let's not specify anything regarding the distribution yet). In this given case we can observe a set of conventional learning problems with their specific observations. We are still interested in a single learning task, but we assume that we can leverage some information coming from other learning tasks originating from the same learning problem distribution.

Well, the story of meta-learning starts with LSTMs, etc. But these days they turned into smart initialization finding for the distribution of tasks at hand. For deep learning applications. The "Learning Around a Common Mean" paper creates a convex meta-learning model where all the parameters of the model are adjusted for the task distribution. In the meta-learning settings achieving this convex problem is not abundant (Especially in a nonlinear problem setting where you cannot use linear models effectively.).

Remark 2. In our investigation of the MAML we have observed that this Ridge-based meta-learning method can achieve more stable results compared to SGD-based meta-learning methods. However, we are only able to observe this in the linear setting as there is no non-linear version present. Then, I asked the question with a similar penalization technique used in the "Learning Around the Common Mean" paper;

Research Question 1. Can a convex kernelized meta-learning method be constructed that can learn some of its parameters from task distribution and has a comparable or better performance and is more interpretable compared to conventional deep learning based meta-learning models in a few-shot learning setting (where there is a limited number of data available from a task of interest coming from the same task distribution)?

Research Question 2. If we can create a model like that can we give statistical guarantees under specific assumptions that the expected risk of this model will be better than its conventional counterpart on a given learning task?

Well, I acknowledge that it might be too ambitious for a research question, and in the end what I am doing ended up not being a convex problem but still I believe there are still some interesting research questions moving forward. (Apparently it is not that ambitious for some of you. Then, I wonder why I do not see any method like this? And although it is not ambitious then I still find it hard to come up with a model that is capable of solving this type of problem.) So, please bear with me maybe I can convince you at least to a certain extent that meta-learning can be useful in certain extrapolation problems.

1. Basic Setup

A task T_a is a d-dimensional regression problem in which the input-output relation is given by

$$y = f_a(\mathbf{x}) + \varepsilon, \tag{1.1}$$

where $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^d$, and ε is taken to be standard normal. The task space is defined by a distribution p_f over functions.

Remark 3. I am not considering any specific distribution over functions for now. However, we can consider a Gaussian Process $f(\mathbf{x}) \sim GP(m(\mathbf{x}), cov(\mathbf{x}, \mathbf{x}'))$ as a distribution over functions for instance.

Let us assume that the input distribution is denoted as $p_{\mathbf{x}}$ and for every task, we have the same input distribution. Moreover, we have observed N_a samples from the corresponding regression problem which are collected in

$$Z_a := ((\mathbf{x}_i, y_i))_{i=1}^{N_a}. \tag{1.2}$$

For the time being, the loss we consider is the standard least squares loss, as for now my interest lies with the regression problems ($\mathcal{L}_a = \sum_i^{N_a} (\hat{f}(\mathbf{x}_i) - y_i)^2$).

Remark 4. Now, with Marco's reading suggestions I ended up finding the two approaches to incorporate the information gained from tasks p_f (noting that I am still not specifying p_f on purpose) to improve the prediction of T_a. The first method uses the Semi-parametric Representer Theorem and the second method is Biased Regularization. The reason we are only considering these two approaches is to see if we can extend the "Learning Around a Common Mean" paper to a nonlinear setting while still preserving the convex problem setting. In the end, we might need to of course select other methods to compare our method, but for now, there are two options that I suspect have the stability and the performance of their linear counterpart which can perform better than most used meta-learning problems in the nonlinear setting. I have not encountered any kernel-based meta-learning model, especially with the construction that I have in mind. I am wondering why other people have not pursued this path.

Let us first start with the method that uses Semi-parametric Representer Theorem.

2. Method

Remark 5. For the sake of brevity let's forget about the so-called task space of the meta-learning problem and focus on the single learning problem T_a. Where we are trying to learn $f_a(\mathbf{x})$. Then, from now on I will abuse the notation and will use $\hat{f}_a(\mathbf{x}) \triangleq f$ for our estimator.

Thanks to Nonparametric Representer Theorem for g to be a strictly increasing function and $\mathcal{L}$ being a monotonic loss function, then

$$\hat{f} = \min_{f \in \mathcal{H}} \quad \sum_i^{N_a} (f(\mathbf{x}_i) - y_i)^2 + g(||f||_{\mathcal{H}}) \tag{2.1}$$

has the solution of the form, $f(\cdot) = \sum_i^{N_a} \alpha_i k(\cdot, \mathbf{x}_i)$ [1]. This is the most fundamental step to kernalize the Linear Ridge Regression method. In the Semi-parametric Representer Theorem $\tilde{f} = f + h$ where $h \in span\{\psi_p\}$ and $\{\psi_p\}_{p=1}^M$ are real-valued functions and $\mathcal{L} = \sum_i^N (\tilde{f}(\mathbf{x}_i) - y_i)^2$. Then the solution to

$$\hat{\tilde{f}} = \min_{\tilde{f} \in \mathcal{H}} \quad \sum_i^{N_a} (\tilde{f}(\mathbf{x}_i) - y_i)^2 + g(||f||_{\mathcal{H}}) \tag{2.2}$$

has the form $\tilde{f}(\cdot) = \sum_i^N \alpha_i k(\cdot, \mathbf{x}_i) + \sum_j^M \beta_j \psi_j(\cdot)$.

In other words, on top of our observed dataset, $Z_a \triangleq Z$ we are given real-valued functions $\{\psi_p\}_{p=1}^M$ and our final estimator will use data points and extra information coming from multiple functions for the estimator $\tilde{f}$.

Remark 6. You might wonder; "Why are we doing this?", "What does it mean in the sense of meta-learning?". Well, as I have stated earlier we have access to the observations from the task space. Because of this, we might be able to put meta-information via those functions $\{\psi_p\}_{p=1}^M$ for now I am not specifying anything again. However, one might imagine that we might use trained models, simple interpolation (If there is enough data available of course), or exact functions if we have access to those. Then with this procedure, we will be technically weighing them with the observed data to come up with our estimator for a specific task at hand.

Well after some tiny bit of taking derivatives and equating to zero for $g(||f||_{\mathcal{H}}) := \lambda||f||^2$, the optimal solutions are given by:

Nonparametric Representer Theorem:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} \tag{2.3}$$

and

Semiparameteric Representer Theorem:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{KK} + \lambda\mathbf{K} - \mathbf{K}\boldsymbol{\psi}(\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{\psi})^{-1}\boldsymbol{\psi}^{\mathrm{T}}\mathbf{K})^{-1}(\mathbf{K} - \mathbf{K}\boldsymbol{\psi}(\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{\psi})^{-1}\boldsymbol{\psi})\mathbf{y} \tag{2.4}$$

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{\psi}^{\mathrm{T}}\boldsymbol{\psi})^{-1}(\boldsymbol{\psi}^{\mathrm{T}}\mathbf{y} - \boldsymbol{\psi}^{\mathrm{T}}\mathbf{K}\hat{\boldsymbol{\alpha}}) \tag{2.5}$$

Remark 7. Again, abusing the notations to the fullest extent but the bold letters represent the collections of the values of the non-bold versions in row vectors as follows; $\boldsymbol{\alpha} \in \mathbb{R}^{N_a \times 1}$, $\boldsymbol{\beta} \in \mathbb{R}^{M \times 1}$, $\boldsymbol{\psi} \in \mathbb{R}^{N_a \times M}$, $\mathbf{K} \in \mathbb{R}^{N_a \times N_a}$ and $\mathbf{y} \in \mathbb{R}^{N_a \times 1}$.

Remark 8. Well, I could convince Marco neither that this problem is convex nor my derivation is correct so let me elaborate a tiny bit more. Our objective function with the observed samples takes the form

$$(\mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\psi}\boldsymbol{\beta} - \mathbf{y})^{\mathrm{T}}(\mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\psi}\boldsymbol{\beta} - \mathbf{y}) + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{K}\boldsymbol{\alpha}. \tag{2.6}$$

Remember that our adjustable parameters are $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}$. They are in the multiplicative form in our objective function. This indicates that although concerning either one of them objective function is convex the overall problem is not necessarily convex. I hope this is convincing. Let us now consider the derivatives with respect to those collections of parameters. I am warning you that I will exploit some tensor

[1]Here $\mathcal{H}$ represents the Reproducing Kernel Hilbert Space

algebra tricks to differentiate with respect to vectors and exploit the fact that $\mathbf{K}$ is symmetric. First, consider the derivative with respect to $\boldsymbol{\alpha}$;

$$2(\mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\psi}\boldsymbol{\beta} - \mathbf{y})^{\mathrm{T}}\mathbf{K} + 2\lambda\alpha^{\mathrm{T}}\mathbf{K} = 0. \tag{2.7}$$

With little manipulations we end up with the above optimum for $\boldsymbol{\alpha}$ in Equation 2.5. If we take the derivative with respect to $\boldsymbol{\beta}$;

$$2(\mathbf{K}\boldsymbol{\alpha} + \boldsymbol{\psi}\boldsymbol{\beta} - \mathbf{y})^{\mathrm{T}}\boldsymbol{\psi} = 0. \tag{2.8}$$

Now, again with a little manipulation, we end up with the optimum expression for $\boldsymbol{\beta}$. Well, here I would like to emphasize again that the ψ is not an exact inverse operation (I am utilizing pseudo-inverse, in my implementation.). Thus, it is visible now that you cannot get a simple expression without $\boldsymbol{\beta}$.

Remark 9. Now, with this derivation we can see that the solution to $\tilde{f}(\cdot)$ is not necessarily convex, (BOOOO!!) But with David's suggestion, we can get a reasonable estimate by starting with a random vector and iteratively following the optimum (HURRAYYY!!!).

Let me show you with an example, why I am a bit hyped about this approach. Note that I will utilize the above remark on iteratively going from optimums like David suggested. In Figure **??** I am sampling 10 ($M = 10$) different sine functions with random phases for $\{\psi_p\}_{p=1}^{M}$. Then observing a random sine function again with a different random phase for my test case and from that sine function, I observe 3 random points without noise $N_a = 3$. Note that I am using 2 points for training and looking at the third point performance for adjusting my kernel length-scale and regularization parameter, see how we can get the sine function now. Isn't it cool to predict a random sine function with 3 points? Moreover, you see the standard Kernel Ridge performance as well which is only good in the interpolation regime.

Remark 10. Well, up until now I have not specified anything regarding the functions $\{\psi_p\}_{p=1}^{M}$. If we are to assume these functions are observed tasks $\psi_p = f_a$ the relation to the meta-learning setting becomes obvious (at least for me). We can go one step further and assume that we do not have access to the underlying functions $\{f_a\}_a = 1^M$ but have access to estimators obtained for tasks $\{\hat{f}_a\}_a = 1^M$ too. But, let's keep it simple for now and say we have access to the exact underlying functions of each task!

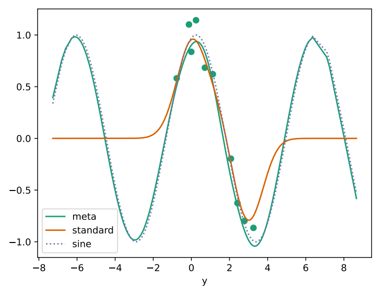
3. Initial Investigation

Here, I would like to present some initial hypotheses and some assumptions that I will make during the experimentation stage. I will call the conventional Kernel Ridge Regression model a "standard" learner and a Semi-parametric Kernel Ridge Regression "meta" learner (with quoted words representing the tags of both) moving forward.

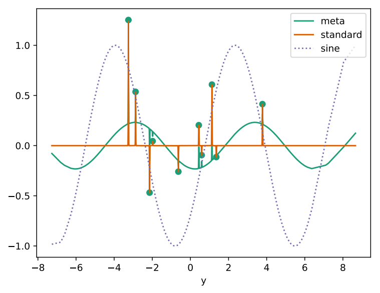
Let us assume we are trying to solve a problem for a 1-dimensional regression problem (D=1),

$$y = \sin(\mathbf{x} + \phi_a) + \varepsilon, \tag{3.1}$$

where $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^D$, $\phi_a \sim \mathcal{N}(\mathbf{0}, c\mathbf{I})$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Let us take $\sigma = 0$ for the time being as well. And further assume that $p_{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, b\mathbf{I})$ Note that each



(A) Optimized hyper-parameters



(B) No hyper-parameter tuning with small length scale and no regularization.

FIGURE 1. Demonstration

phase realization corresponds to a task T_a observed in the environment T and let us assume that we have access to M tasks.

My initial hypotheses for a given task T_a with observed N_a

- $\hat{f}_{meta}$ will perform better for increasing M and increasing σ^2 and b compared to $\hat{f}_{standard}$.
- Increasing the N_a will improve $\hat{f}_{meta}$ more than $\hat{f}_{standard}$.
- The performance of the $\hat{f}_{meta}$ will decrease if we assume we that we do not have access to the real functions $\{\psi_p\}_{p=1}^{M}$ but estimations of these functions $\{\hat{\psi}_p\}_{p=1}^{M}$.
- The quality of the performance of $\hat{f}_{meta}$ will depend highly on the $\boldsymbol{\psi}^{-1}$ (when $M \neq N$, $\boldsymbol{\psi}$ is not even a square matrix).

Remark 11. In order to investigate this for a single task T_a I will look at learning curves for a noiseless case and then move to more complicated cases. At a later stage, the expected performance over the whole task space might be necessary. This procedure has to be repeated for an interval of M and σ and b.

Remark 12. I am keeping the biased regularization case for now for the sake of brevity, but according to David and my initial investigation, we end up with a similar type of expression. But for now, let's focus on the given problem at hand.

Remark 13. You might think this is a trivial extension for our BNAIC paper. I do not know what is not trivial if you are basically following a path of thoughts ending up here was not trivial but next step from a certain point can definitely be trivial. For me the setting I am sketching now might seem trivial, but putting this into a real meta-learning framework, where the functions $\{\psi_p\}_{p=1}^{M}$ are inferred from the task distribution itself does not seem trivial for me. Especially, if we can give statistical guarantees for our proposed method. Moreover, if we can make this type of method memory and time efficient we might have ourselves a better performing and much more interpretable meta-learning model in our hands. And we have our initial study to compare our performance to.

4. Marco's Suggested Formulation

- Concatenated formulation gives the same result with my derivations, just machine precision differences.

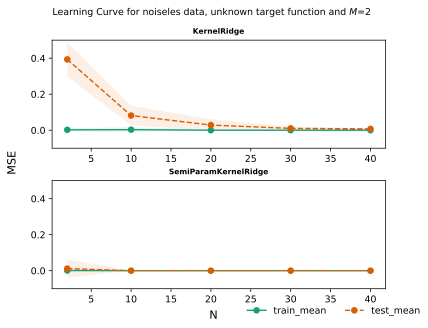
The derivation is as follows,

$$(\mathbf{A}\mathbf{w}-\mathbf{y})^{\mathrm{T}}(\mathbf{A}\mathbf{w}-\mathbf{y})+\lambda\mathbf{w}^{\mathrm{T}}\mathbf{B}\mathbf{w}, \tag{4.1}$$

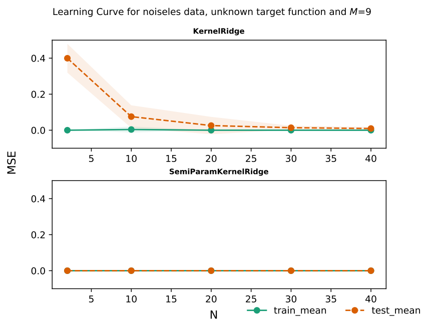
$\mathbf{A} := [\mathbf{K}\boldsymbol{\psi}] \in \mathbb{R}^{N\times(N+M)}$ and $B := [\mathbf{K},0;0,0] \in \mathbb{R}^{(N+M)\times(N+M)}$
Then the optimum parameters is found to be

$$\hat{\mathbf{w}} = (\mathbf{A}^{\mathrm{T}}\mathbf{A}+\lambda\mathbf{B}^{\mathrm{T}})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y}. \tag{4.2}$$

5. Learning Curves and Additional Function Curve Effect

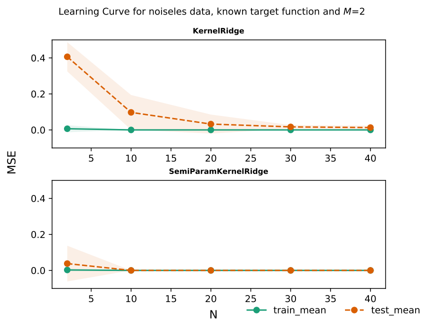


(A) $M=2$

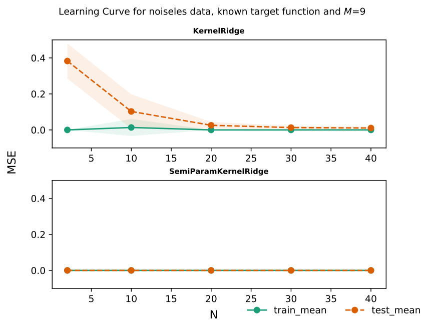


(B) $M=9$

Figure 2. Noiseless Learning Curves for $M = 2$ and $M = 9$ Hyperparameters tuned and the drawn objective task is not present in $\boldsymbol{\psi}$.



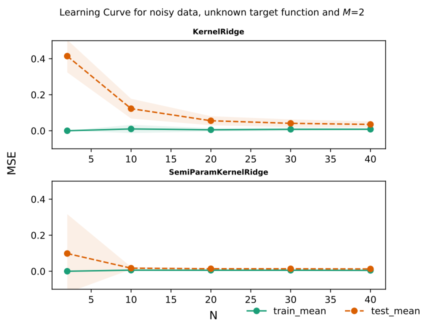
(A) $M=2$



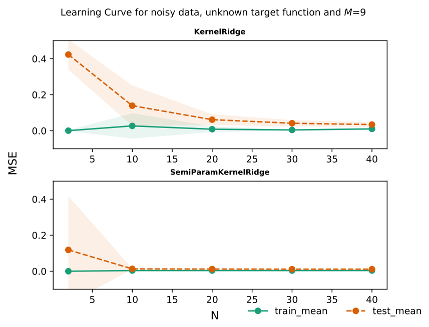
(B) $M=9$

Figure 3. Noiseless data Learning Curves for $M = 2$ and $M = 82$ Hyperparameters tuned and the drawn objective task is present in $\boldsymbol{\psi}$.

- Benefit of additional functions is evident.

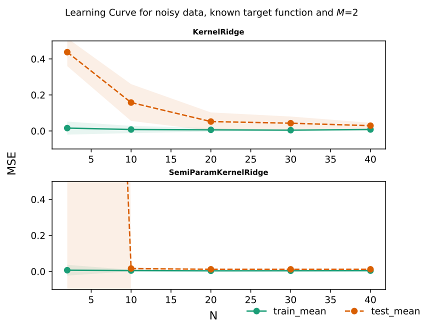


(A) $M = 2$

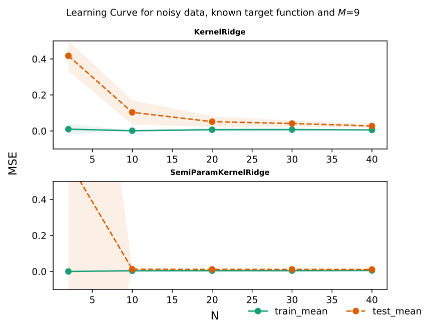


(B) $M = 9$

FIGURE 4. Noisy data Learning Curves for $M = 2$ and $M = 9$ Hyperparameters tuned and the drawn objective task is not present in $\boldsymbol{\psi}$.



(A) $M = 2$



(B) $M = 9$

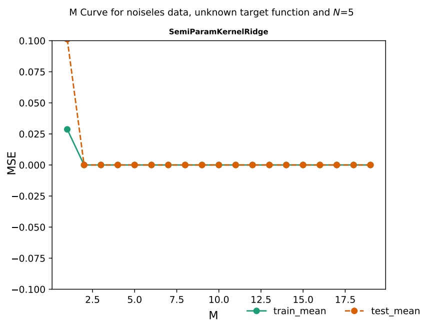
FIGURE 5. Noisy data Learning Curves for $M = 2$ and $M = 9$ Hyperparameters tuned and the drawn objective task is present in $\boldsymbol{\psi}$.

- Looking at additional functions for unknown function case after having two functions we can almost go to zero test error.

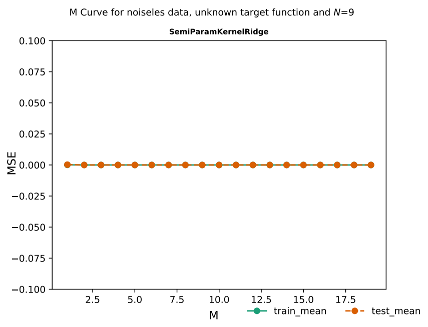
6. FUNCTIONAL PCA

Given the amount of functionals that can be present identifying some form of representative form with less space but more information we can use dimensionality reduction techniques for functionals.

- I am reading the Book Functional Data Analysis by Ramsay and Silverman.
- Smoothing and interpolation before using the data is essential.
- All the functionals have to be evaluated at the same input points $\{x_i\}_{i=1}^N$. In other words, we need to have the same grid. After putting them in to $\{f_i(\cdot)\}_{i=1}^P$
- For the functional PCA we want to find a set of K orthonormal functions so that the expension of each curve in terms of these basis functions approximates the curve as closely as possible.

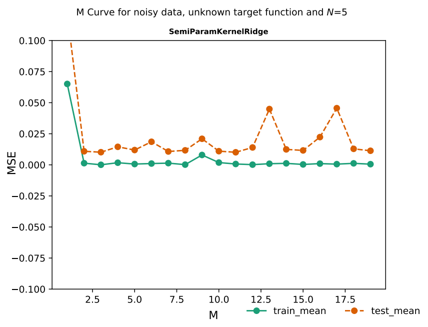


(A) $N = 5$

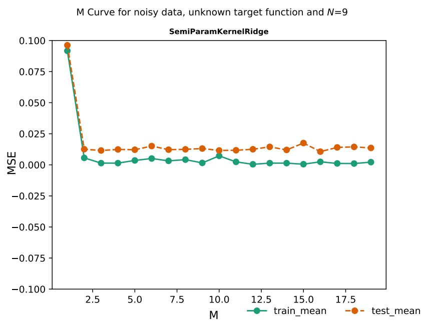


(B) $N = 9$

FIGURE 6. Noiseless data Complexity Curves for $N = 5$ and $N = 9$ Hyperparameters tuned and the drawn objective task is not present in $\boldsymbol{\psi}$.



(A) $N = 5$



(B) $N = 9$

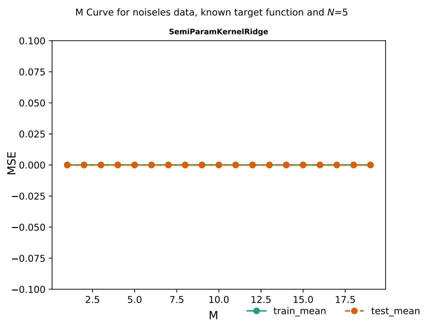
FIGURE 7. Noisy data Complexity Curves for $N = 5$ and $N = 9$ Hyperparameters tuned and the drawn objective task is not present in $\boldsymbol{\psi}$.

- I do not see the point of writing the formulations I see in the book, but here is in a broad mathematical sense waht we are doing. Lets say our functions can be represented with linear combination of the values.

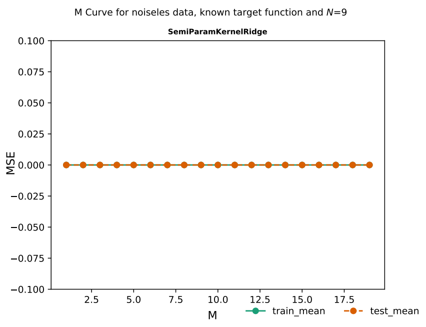
$$f_i = \{\sum_{j=1}^{p} \beta_j x_{ji}\}_{i=1}^{N} \tag{6.1}$$

- What functional PCA tries to do is to find the β_j that maximizes the MSE with the constraint $||\beta_j|| = 1$ is satsified for every functional value.
- This way the largest variational modes are identified.

- It is obvious that we can represent the whole range of sine to cosine with just two eigenfunctions.
- What is more interesting is that we observed that for unknown task after $M = 2$ we are able to obtain almost zero test error.

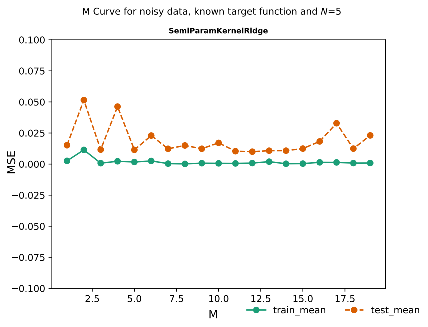


(A) $N = 5$

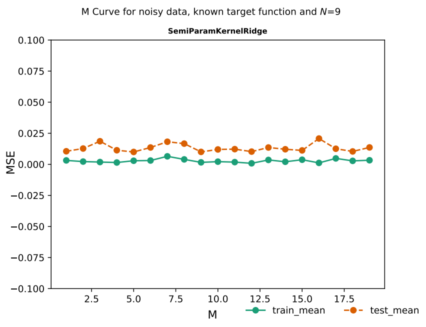


(B) $N = 9$

FIGURE 8. Noiseless data Complexity Curves for $N = 5$ and $N = 9$ Hyperparameters tuned and the drawn objective task is present in $\boldsymbol{\psi}$.



(A) $N = 5$



(B) $N = 9$

FIGURE 9. Noisy data Complexity Curves for $N = 5$ and $N = 9$ Hyperparameters tuned and the drawn objective task is present in $\boldsymbol{\psi}$.

- After this observation I would like to use these PCA components as my $\psi_j(\cdot)$. But, know the same input point assumption is hindering flexibility.

7. Learning Curves

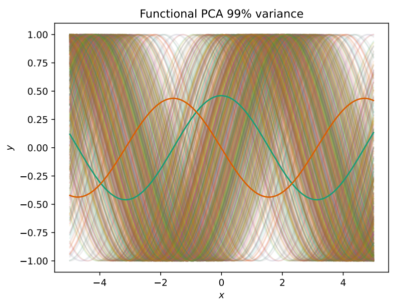
Some ideas on the similar method applied to learning curves.

- Get learning curves for multiple hyper parameters for the same dataset then try to extrapolate one of them with the help of others.
- Try to extrapolate from untuned models to a tuned model.

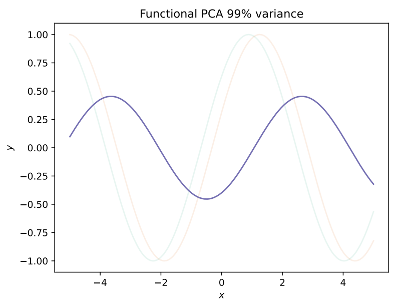
7.1. **Ridge Regression Learning Curves.**

From the Equation 1.1 observe data with $p(x) \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, 1)$. Obtain the learning curves for different ridge parameters λ for $N : 5, ..., 100$

NOTE: After the PCA mean of the functions are added to all the components. The first principle component can deal relatively well with this thus the weight for it becomes just close to one!



(A) $P = 2$



(B) $P = 9$

FIGURE 10. Observing the functional PCA results with 99% of variance for different number of available functionals. Note that P introduced here is analogous to M in our above setting for Semi-Parametric method.

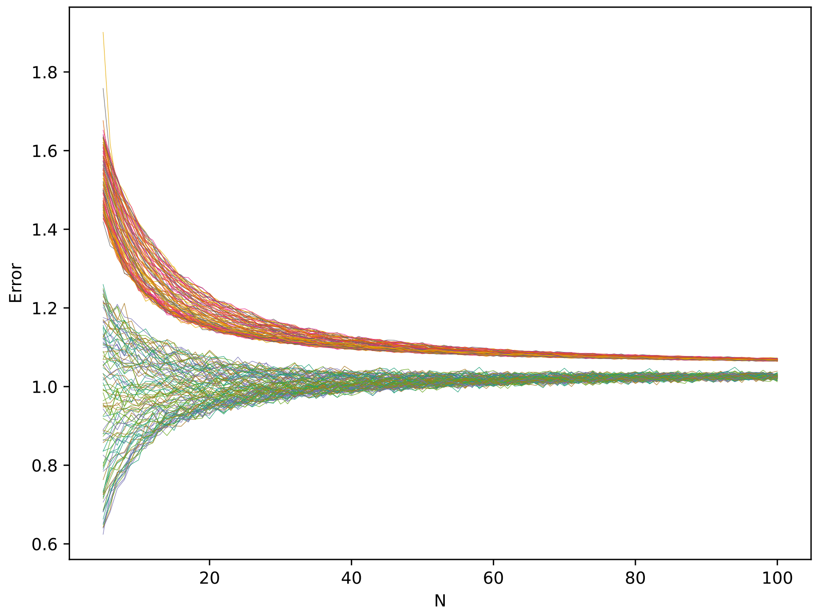


FIGURE 11. Learning Curves for Ridge Regression varying λ

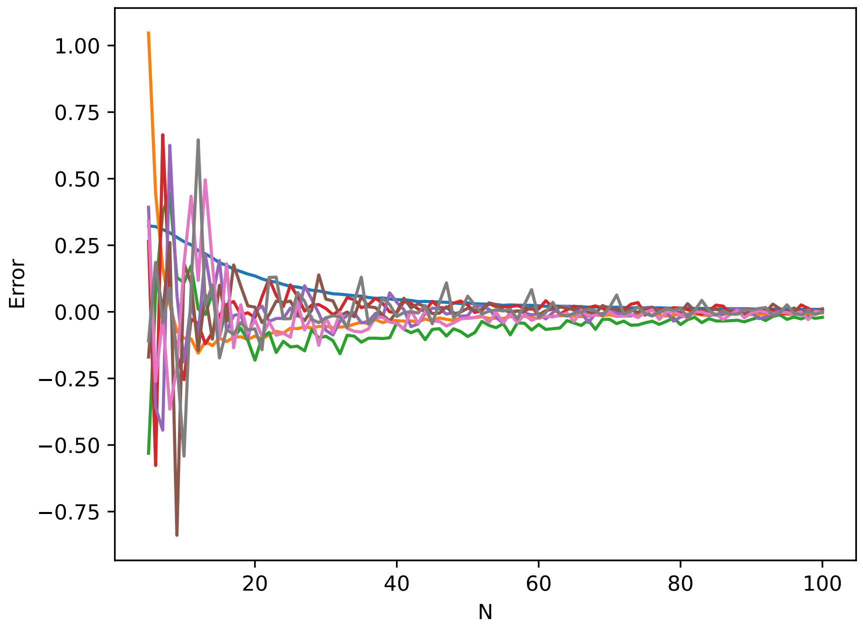


FIGURE 12. PCA 99% covering principle components (not smoothed!)λ

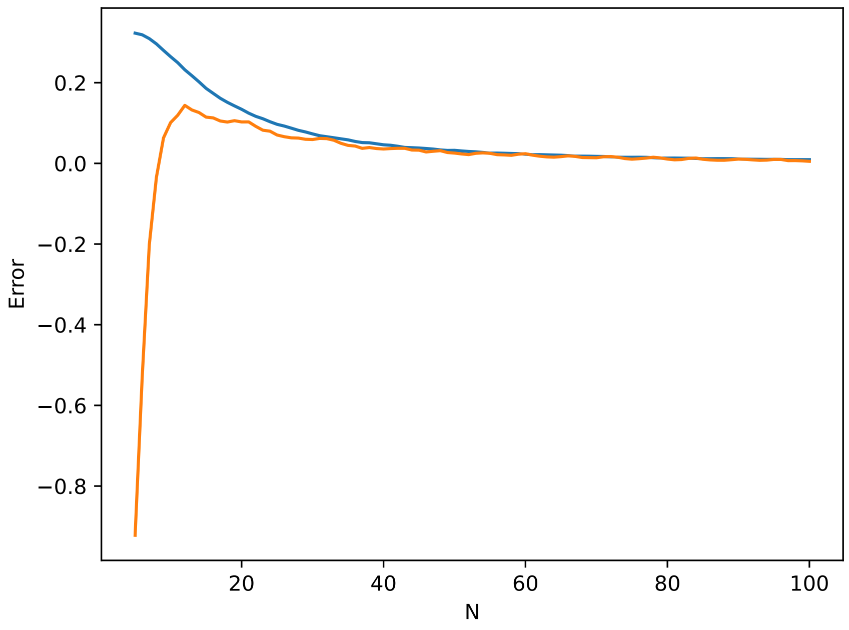


Figure 13. PCA 99% covering principle components (smoothed!)λ

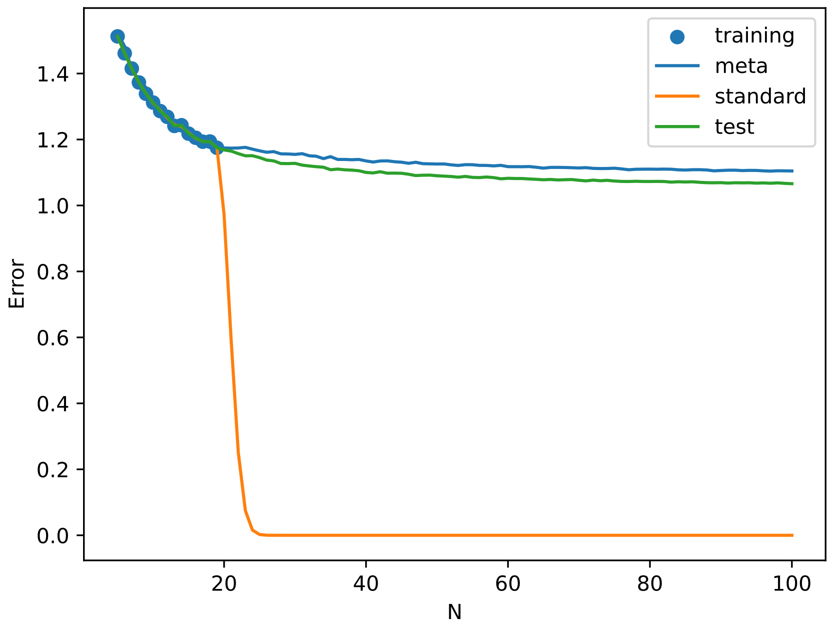


FIGURE 14. Semi Parametric Kernel Ridge by using 1 other function.

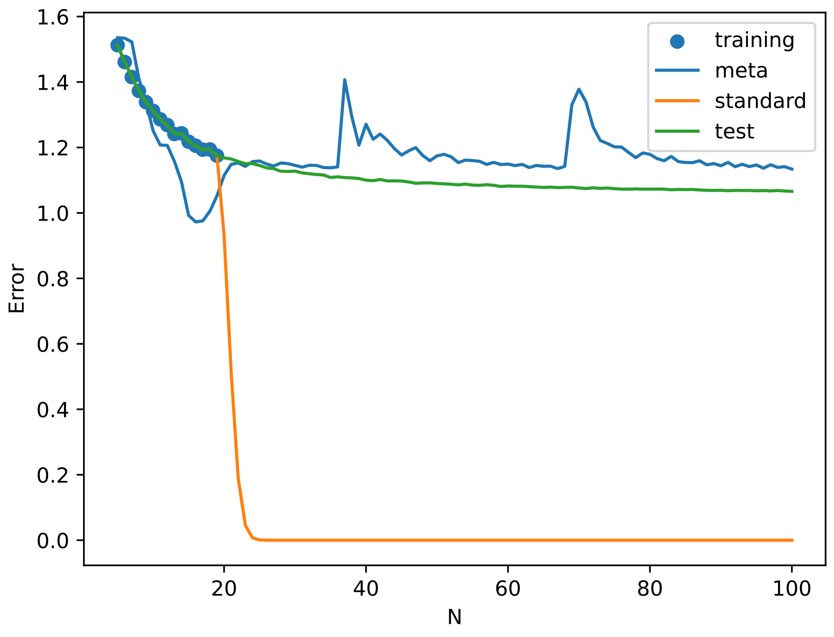


FIGURE 15. Semi Parametric Kernel Ridge by using 3 other functions.

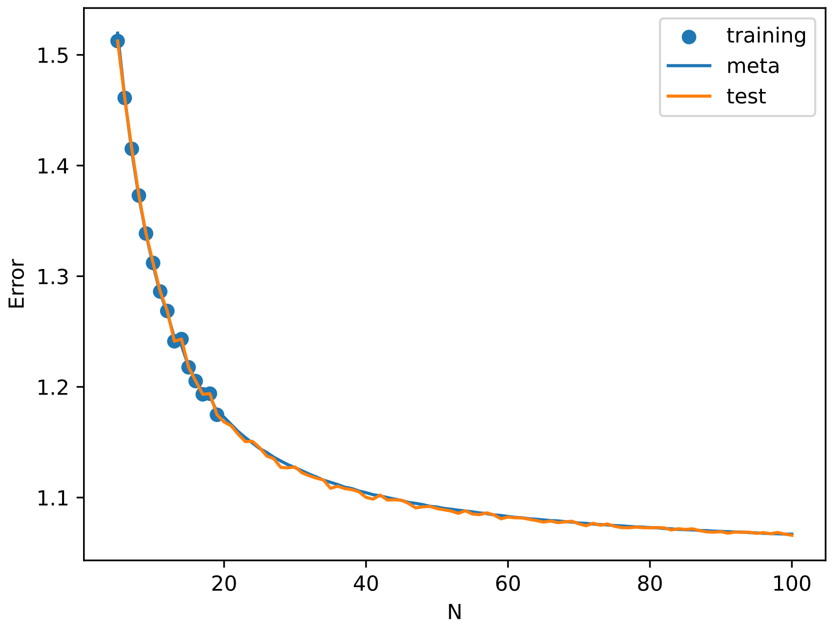


Figure 16. Semi Parametric Kernel Ridge by using 2 PCA component.

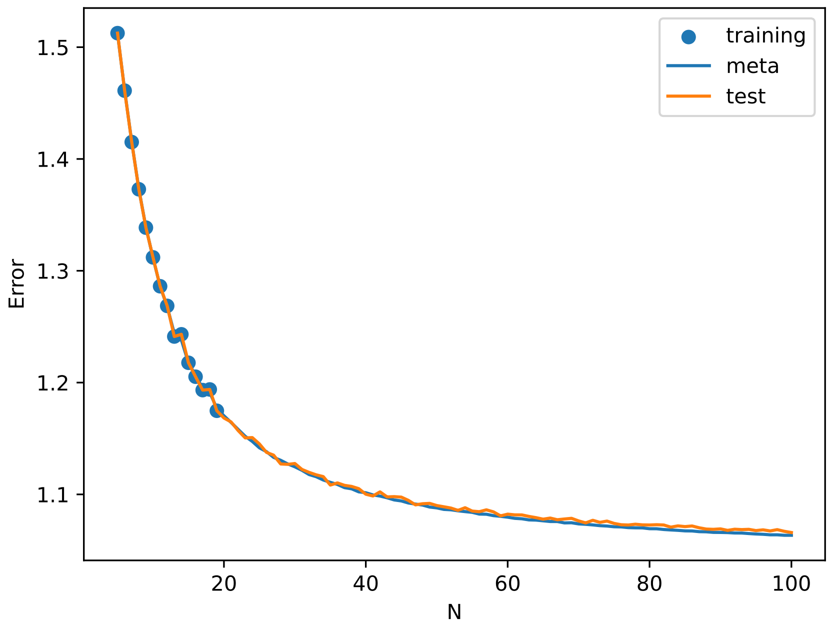


Figure 17. Semi Parametric Kernel Ridge by using 6 PCA components.

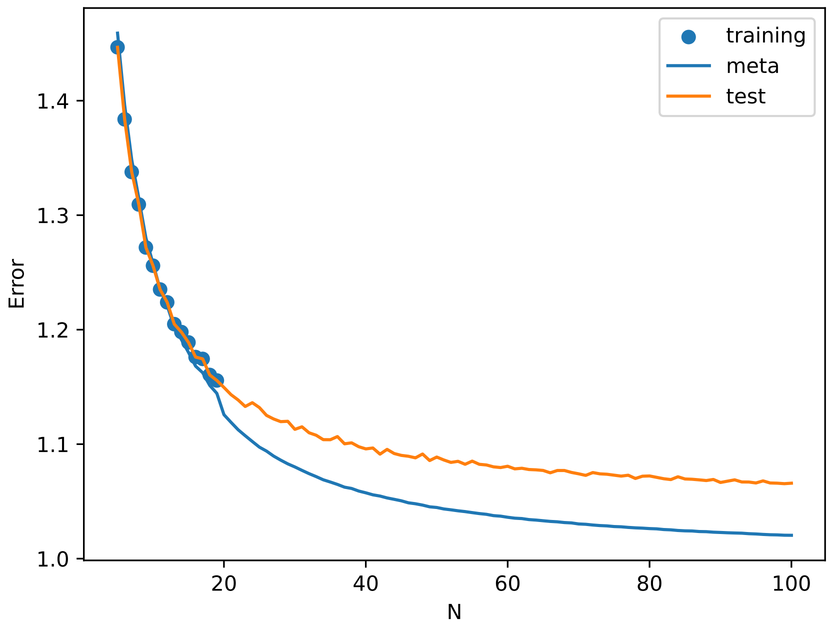


FIGURE 18. Semi Parametric Kernel Ridge by using mean of the all components.