Meta-learning Implicit Neural Representation for Sparse Time Series Functional Data Analysis

Bofan Chen
Department of Pure Mathematics and Mathematical Statistics
University of Cambridge
cbfcbf.byron@gmail.com

Abstract

Sparse time series data is everywhere in our daily life, which is formed by sparsely observed samples from a given function distribution. However, traditional statistical methods struggle to analyze such data effectively. Here, we introduce MetaINR, a method for learning the implicit neural representation (INR) of time series from sparse and irregular observations. MetaINR incorporates meta-learning into the functional data analysis (FDA) framework, efficiently learning sample distributions. It utilizes a specific SIREN-based network architecture to recover complete samples based on a few observation points. With MetaINR, estimations of the mean function, covariance operator, and principal components become straightforward. Our experiments, conducted on both synthetic and real data, demonstrate that MetaINR significantly outperforms traditional baseline methods in terms of estimation accuracy.

1 Introduction

Sparse time series data is common in daily life, such as medical data measured at different times for patients or height data collected at various stages of adolescence. However, the sparsity and irregularity of such data make traditional statistical methods hard to apply directly. Additionally, the consideration of continuity and differentiability properties in most time series data pose challenges for multivariable statistical analysis.

Recent research suggests that meta-learning [1–3] exhibits great advantages in handling sparse data. It possesses the capability to identify shared patterns among related tasks [4], facilitating few-shot learning or even one-shot learning [5] with extremely limited training data. Moreover, there has been substantial progress in statistical analysis utilizing dense functional data to address continuous time series issues [6]. These findings inspire us to leverage meta-learning techniques to transform sparse time series samples into dense formats (recover the samples). Subsequently, we can extend the established functional data analysis methodologies tailored for dense data to analyze sparse time series datasets effectively.

In this paper, we introduce a novel approach, MetaINR, which combines meta-learning and functional data analysis to address the estimation challenges posed by sparse time series data. Through this method, we use meta-learning to realize the implicit neural representation (INR) to reconstruct the time series function for each sample based on sparse time series observations. Furthermore, we can estimate the mean, covariance, and principal components of the distribution behind these samples. The advantange of MetaINR can be summarized as follows:

- Efficient and accurate reconstruction of time series samples with sparse observations by leveraging information from the entire dataset

Preprint. Under review.

- Effective estimation of mean function, covariance operator and principal components for sparse time series dataset
- Independence from any initial assumptions about the distribution of time series samples
- Preservation of fine details in the samples and easy calculation for their derivatives through the SIREN-based INR

Therefore, MetaINR method provides a new solution for analyzing sparse time series data.

2 Related works

2.1 Functional Data Analysis (FDA)

Multivariate data versus functional data Traditional multivariate statistics take finite-dimensional vectors as the objects of study to analyze their statistical properties, such as mean vectors and covariance matrices. This approach is appropriate and is easy to compute when the dimension of the statistical objects is relatively small. However, if the dimension of the space is increased to infinite, for example, in L^2 space, traditional methods face certain challenges. Besides the expensive computational cost, in such functional spaces, people's focus is not only on the values of vectors in a particular dimension but also on properties such as continuity, smoothness and derivatives. Additionally, as the space grows to infinite dimensions, many useful tools in multivariate statistics, such as probability density functions, will lose their meaning. It is the differences in statistical properties and people's focus between infinite-dimensional and finite-dimensional spaces, that have led to widespread attention to functional data analysis (FDA) [6, 7] among researchers. In FDA, scientists treat functions as basic elements for analysis and have developed various methods to estimate the statistical properties for a functional distribution, such as mean function, covariance operator and functional principal components.

Sparse time series data Time series is a typical object that needs to be addressed in an infinite-dimensional function space. This is because time series data collected in reality often exhibit irregularity and continuity, and, in certain situations, people may focus on topics such as derivatives and time alignment (registration). Many previous time series models were limited to multivariate statistical methods, such as Vector AutoRegression (VAR)[8], AutoRegressive Integrated Moving Average model (ARIMA) [9], Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) [10], to name a few. From the perspective of function approximation, these multivariate-statistics-based studies can indeed reflect certain properties of time series. However, in many cases, this approach is challenging to generalize, as is the case with sparse time series data. When the observation frequency of time series is low and irregular, sparse time series data is formed. We can represent the sparse dataset of n samples as

$$D = \{(t_{ij}, y_{ij})_{j=1}^{J_i}\}_{i=1}^{n}$$

where $y_{ij} = x_i(t_{ij}), i = 1, \cdots, n, j = 1, \cdots, J_i$ and J_i is the total observations of the i-th sample. This type of data is particularly common in the medical field [11, 12], as the frequency of medical visits varies from person to person, with irregular patterns. Due to irregularity and sparsity, this form of data cannot be analyzed using traditional multivariate statistical methods or dense-data-based functional data analysis (FDA). Instead, it requires the introduction of new methods.

Our benchmarks and the drawbacks of traditional FDA method for sparse data There have been previous explorations in sparse functional data analysis. We introduce the following two traditional solutions for sparse time series data as benchmarks in this paper:

- **Local polynomial regression + dense-data-based FDA (Pre-smoothing method)**: In this method, we first utilize local polynomial regression directly to recover samples with sparse observations, and then we apply dense-data-based methods to the recovered dense samples. However, this method is inherently unreliable due to the extremely sparse observations for individual samples in the dataset. This sparsity leads to unreliable local polynomial reconstructions, undermining the overall reliability of the method.
- **Principal Analysis by Conditional Expectation (PACE method)** [13]: In this method, we start by employing local polynomial regression to estimate the mean function and variance kernel. Next, we calculate principal components based on these estimates. Finally, we

recover samples by computing the scores associated with each principal component (see details in the appendix). Comparing to Pre-smoothing method, this approach leverages information from observations of other samples in the dataset during the sample recovery process, thereby enhancing the reliability of the reconstruction. However, it's important to note that PACE also comes with limitations, primarily because it heavily depends on the assumption that scores follow a normal distribution while recovering time series samples.

2.2 Machine learning for functional data

Unlike traditional methods, this paper introduces machine learning techniques to analyze sparse time series data. Specifically, this involves two aspects:

- We will use Implicit Neural Representation (INR) [14] to recover and represent each sample.
- We will leverage meta-learning methods to effectively estimate the INR for each sample with sparse observations.

2.2.1 Implicit Neural Representation

Traditional grid-based representation versus implicit neural representation Traditionally, the representation of functional data $y(t)$ involves using a standard grid-based method, wherein a finite set of coordinates and corresponding function values are given as pairs (t_i, y_i). However, this approach has certain drawbacks: the original functional data in an infinite-dimensional space is discretized into a finite-dimensional space. This not only hampers operations like differentiation but also significantly impacts the resolution of the function within a smaller time interval. With the emergence of deep learning, it has been discovered that utilizing deep neural networks can be an effective way to represent functions implicitly. Implicit neural representation (INR) parameterize the functional data by deep neural networks:

$$y(t) = F_\phi(t),$$

where $F_\phi(\cdot)$ is a deep neural network parameterized by ϕ. This type of continuous representation not only proves to be more memory-efficient but also enables higher resolution and the handling of more complex operations on functional data.

SIREN architecture for time series and its advantages Many prior works have utilized ReLU-based multilayer perceptrons (MLP) as the neural network architecture for Implicit Neural Representations (INR) [15, 16]. However, ReLU-based INR often lack the ability to capture fine details in signal and struggle to effectively express higher-order derivatives. In addressing these limitation, [14] introduced the SIREN architecture as an alternative to traditional ReLU-MLPs:

$$\Phi(\mathbf{x}) = \mathbf{W}_n \left(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0\right)(\mathbf{x}) + \mathbf{b}_n, \quad \mathbf{x}_i \mapsto \phi_i\left(\mathbf{x}_i\right) = \sin\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right)$$

where $\phi_i : \mathbb{R}^{M_i} \mapsto \mathbb{R}^{N_i}$ is the i^{th} layer of the network with weight matrix $\mathbf{W}_i \in \mathbb{R}^{N_i \times M_i}$ and biases $\mathbf{b}_i \in \mathbb{R}^{N_i}$. SIREN employs the sine function as the activation function, effectively modeling signals with fine details. A remarkable observation is that even after taking derivatives of a function represented using SIREN, it can still be expressed using the SIREN architecture. These properties of SIREN make it the preferred choice for INR. While previous research predominantly applied SIREN to represent images and videos [17], its application in time series representation has been limited. Inspired by these and other seminal works, this paper explores the use of SIREN-based INR of time series samples and demonstates its superiority in further FDA analysis.

2.2.2 Meta-learning for sparse data INR

Meta-learning framework Meta-learning was initially introduced to enhance the efficiency of machine learning, often referred to as "learning to learn". By employing the deep learning model $F_\phi(\cdot)$ to address similar tasks, the "shared characteristics" among these tasks can be leveraged to expedite the machine learning process [4]. To fomulate, given the observation dataset D^* of the target task and the observation dataset $D = \bigcup_{i=1}^{n} D_i$ of n similar tasks, we need to find a meta-function to effectively learn the parameters of the given task, namely $\phi^* = f_\theta(D^*)$, where θ is the meta-parameter of the meta-function $f_\theta(\cdot)$ that can be learned from D. Accordingly, the original target task can be solved as $F_{\phi^*}(\cdot)$. In fact, the representation of meta-function $f_\theta(\cdot)$ can be diverse.

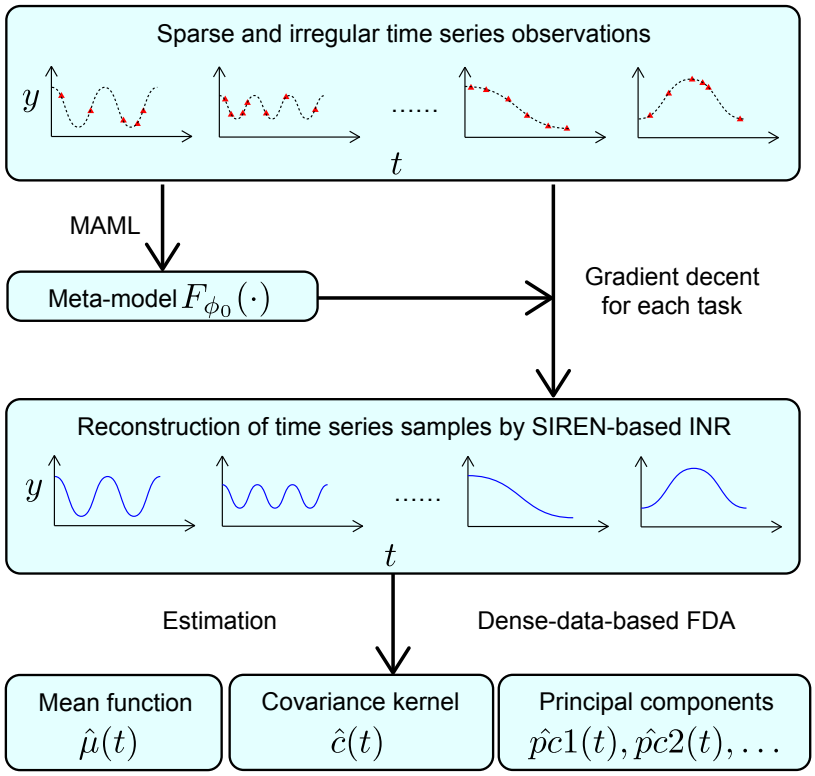

Figure 1: **MetaINR workflow.** Given the sparse and irregular observations from several samples of an functional distribution, MAML algorithm is implemented to learning the optimal initial value ϕ_0 for SIREN architecture F. The meta-model F_{ϕ_0} contain the "shared characteristics" from the distribution, hence the recovery of samples can be implemented efficiently and accurately. The dense-data-based FDA methods then can be applied effectively on the INRs of samples, such as the estimation of mean function, covariance kernel and principal components.

There are two main classes of methods [3]: Parameterized Policy Gradient Methods (PPG) and Black Box Methods.

Parameterized Policy Gradient Methods and MAML The PPG method use an iterative algorithm to represent f_θ, which has an innerloop of the form:

$$\phi^*_{j+1} = \phi^*_j - \alpha_\theta \nabla_{\phi^*_j} \hat{L}_\theta \left(D^*, F_{\phi^*_j} \right),$$

with the initial value $\phi^*_0 = \phi^*_0(\theta)$. This is very intuitive because it is similar to the traditional gradient decent method to optimize. The only difference is that the initial value, loss function and gradient step should be carefully trained by the observations of other tasks D in this scenario. One important example of PPG method is Model-Agnostic Meta-Learning (MAML) [1], where $\phi^*_0(\theta) = \theta$ and $\hat{L}_\theta$ can be any loss function independent of θ. MAML can be regarded as learning an initial value for a task distribution, which retains the gradient decent framework of traditional optimization algorithms.

Black box methods Since we only care about the target function $F_\phi(\cdot)$ rather than the value of parameters ϕ, an alternative way to treat $f_\theta(\cdot)$ is to see it as a black box. Based on this idea, we can rewrite the target function:

$$F_\phi(t) = F_{f_\theta(D^*)}(t) = g_\theta(t, D^*)$$

Here we do not need to explicitly specify the value of ϕ, and treat the meta-learning task as another "big" task, which adds all the sparse sample observations into input. From this perspective, some existing deep learning models for learning differential equations, such as Neural ODE [18] and Neural Laplace [19], can actually be classified as Black box methods. Instead of treating INR for a single time series as a task, these models consider learning the commonalities among a group of time series as a task, where t and D^* are all their input.

Advantages of MAML for sparse INR tasks For black box methods, because of the sequential properties of input observations, sequential model architectures such as RNN are always used in g_θ to encode the observations D^*, which unavoidably change the initial architecture F_ϕ. In contrast, PPG methods use the meta-parameters θ to find the task parameters ϕ explicitly, which maintain the initial network architecture. In this sense, black box methods are less explainable and exhibit an inductive bias derived from data, whereas PPG methods are more straightforward but carry an inductive bias within their network architecture. In the context of time series Implicit Neural Representation (INR), selecting an SIREN-based appropriate network architecture is crucial for modeling complicated signals and their derivatives. Only PPG methods like MAML can learn the "shared characteristics" among samples while preserving the network architecture. Therefore, in this paper, we choose MAML as our meta-learning method for the INR task.

3 Method

Overview of MetaINR Given sparse observations of n similar time series, the goal of FDA for sparse data is to reconstruct each sample as accurately as possible and identify the principal components (PC) of the distribution behind these time series. As discussed in the previous section, traditional FDA approaches either do not work well with sparse data or rely on many assumptions. Here, we choose an alternative way by using the method of meta-learning, which involves three main steps:

- Learning the meta-model: Use SIREN architecture and MAML algorithm to learn the meta-model for the INR of given time series.
- Recovery of samples: Calculate the SIREN-based INR efficiently for each time series sample based on the mata-model.
- Functional data estimation: Apply dense-data-based traditional FDA methods to the MetaINR of each sample to estimate the mean function, covariance kernel and principal components effectively.

We now discuss each step.

Learning the meta-model The tasks here are realizing the implicit neural representation of samples by the SIREN architecture $F_\phi(t)$. Given the sparse observations of n similiar time series $D = \{(t_{ij}, y_{ij})_{j=1}^{J_i}\}_{i=1}^n$, we use MAML algorithm 1 to learning the optimal initial value of parameters ϕ_0 for the given sample distribution. We denote our meta-model $F_{\phi_0}(t)$, which is supposed to contain the features of the sample distribution learned by MAML.

Algorithm 1 Model-Agnostic Meta-Learning for Time Series Implicit Neural Representation

Input: $D = \{(t_{ij}, y_{ij})_{j=1}^{J_i}\}_{i=1}^n$: the sparse observations for tasks
Input: α, β : step size hyperparameters
  randomly initialize θ
  while not done **do**
    for all i **do**
      $\theta^0 \leftarrow \theta$
      for $q = 0$ to $Q - 1$ **do**
        Evaluate $\nabla_{\theta^q} \mathcal{L}_i \left(F_{\theta^q}\right)$ using support set $D_i^{\text{support}} = (t_{ij}, y_{ij})_{j=1}^{\lfloor 0.5 * J_i \rfloor}$
        Compute adapted parameters with gradient descent: $\theta_i^{q+1} = \theta_i^q - \alpha \nabla_{\theta_i^q} \mathcal{L}_i \left(F_{\theta_i^q}\right)$
      end for
      Compute the loss $\mathcal{L}_i \left(F_{\theta_i^Q}\right)$ in sample i using query set $D_i^{\text{query}} = (t_{ij}, y_{ij})_{j=\lfloor 0.5 * J_i \rfloor}^{J_i}$
    end for
    Compute the total loss $\mathcal{L} = \sum_{i=1}^n \mathcal{L}_i \left(F_{\theta_i^Q}\right)$
    Update meta-parameters $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}$
  end while
  $\phi_0 \leftarrow \theta$
Output: the meta-parameters ϕ_0

Recovery of samples Based on this meta-model, we can use gradient decent algorithm 2 to calculate the optimal parameters ϕ_i^* for the sample i and hence obtain the INR $F_{\phi_i^*}(t)$.

Algorithm 2 Implicit Neural Representation Learning for Target Sample

Input: $D^* = (t_{ij}, y_{ij})_{j=1}^{J^*}$: the sparse observations for the target task
Input: α : step size hyperparameters
initialize $\phi \leftarrow \phi_0$
while not done **do**
Evaluate $\nabla_\phi \mathcal{L}(F_\phi)$ using support set $D^* = (t_j^*, y_j^*)_{j=1}^{J^*}$
Update parameters with gradient descent: $\phi = \phi - \alpha \nabla_\phi \mathcal{L}(F_\phi)$
end while
$\phi^* \leftarrow \phi$
Output: ϕ^*: the parameters of INR for the target model

Functional data estimation Based on the estimated INR $\hat{y}_i(t) = F_{\phi_i^*}(t)$, we subsequently estimate the mean function $\hat{\mu}(t)$ and covariance kernel $\hat{c}(t, s)$ as follows:

$$\hat{\mu}(t) = \frac{1}{n} \sum_{i=1}^{n} \hat{y}_i(t)$$

$$\hat{c}(t, s) = \frac{1}{n} \sum_{i=1}^{n} [\hat{y}_i(t)) - \hat{\mu}(t)][\hat{y}_i(s) - \hat{\mu}(s)]$$

According to the covariance operator, we can further estimate its principal components $\hat{\phi}_k(t)$ and the variance of scores $\hat{\lambda}_k$ by solving the eigenfunctions and eigenvalues of $\hat{c}(t, s)$.

$$\int \hat{c}(t, s)\hat{\phi}_k(s)ds = \hat{\lambda}_k \hat{\phi}_k(t)$$

Advantages of MetaINR Inspired by traditional methods, MetaINR utilizes meta-learning to complete implicit neural representation for samples with only sparse observations, enabling them to be applicable with various FDA methods based on dense data. It is worth noting that, we treat meta-learning here not merely as an efficient learning method for sparse training sets but as a means to learn commonalities among a large number of tasks. Compared to the Pre-smoothing method in Section 2, a more robust estimation is achieved with MetaINR's sample recovery, as it leverages all the information from the entire dataset. Furthermore, unlike the PACE method in Section 2, MetaINR imposes no constraints on the distribution of samples, making it more versatile. Compared to other learning-based black box methods, MetaINR stands out by its ability to handle a broader range of sample distributions, not restricted to those generated solely from differential equations. Additionally, MetaINR maintains the SIREN-based INR architecture, which facilitates a more intricate representation of individual samples. The overall comparison of these methods are shown in table 1.

Method	Irregularity of observations	Sparsity of observations	General distribution of samples	Preservation of architecture
Multivariable Statistics	✗	✗	✓	–
Pre-smoothing	✓	✗	✓	–
PACE [13]	✓	✓	✗	–
Neural ODE [18]	✓	✓	✗	✗
Neural Laplace [19]	✓	✓	✓	✗
MetaINR	✓	✓	✓	✓

Table 1: Comparison of analysis methods for sparse time series data

4 Experiments

We evaluate our method on the synthetic data, with benchmarks estimations provided by Pre-smoothing method and the PACE method. Our results clearly demonstrate that MetaINR is able to

effectively recover the samples from sparse observations. The estimations provided by MetaINR significantly outperforms the conventional baseline. Moreover, we extended our analysis to a real-world medical dataset to validate the robustness and applicability of our approach in practical scenarios.

4.1 Synthetic data

Data generating process The ground truth of our synthetic data distribution is given by:

$$y(t) = Asin(t) + Bsin(2t), \quad t \in [0, 10],$$

where A and B are independent and follows the uniform distribution

$$A \sim Uniform[0, 1], \quad B \sim Uniform[0, 0.5].$$

We randomly draw $n = 100$ sample tasks $\{y_i(t)\}_{i=1}^n$ from this distribution. For each sample task, $J_i = 20$ observations randomly chosen in irregular time points. The overall observation dataset used for analysis can be represented as

$$D = \{(t_{ij}, y_{ij})_{j=1}^{J_i}\}_{i=1}^n.$$

Based on our setup, it can be seen that our synthetic data contains two periodic principal components, namely

$$pc_1(t) = sin(t)$$
$$pc_2(t) = sin(2t)$$

It is worth noting that the scores corresponding to these two main components follow a uniform distribution, which does not meet the assumption of normal distribution required by traditional PACE methods. Therefore, in such a data setting, MetaINR demonstrates significant superiority.

Parameters setting Following our previous discussions, we integrate the SIREN architecture with MAML in the experiments. We set the initial parameters of SIREN $\omega_0 = 3$, to match frequency spectrum of the signal fluctuations in the samples (see the appendix for details). During the meta-learning training process, we divide the data into two halves, with each sample having 10 observation points: one half serves as the support set and the other half as the query set. In the outer loop of MAML, we set the learning rate of the meta-model to 0.0001. In the inner loop of MAML, we set the model's learning rate to 0.001 and the number of iterations for parameter optimization $Q = 1$. During subsequent gradient decent algorithm for sample recovery, the model's learning rate is set to 0.001 as well.

Meta-learning the commonalities of time series We observed the capability of meta-learning to capture commonalities among a series of similar time-series data. As shown in Figure 2a, when providing observations solely from the latter segment ($t \in [5, 10]$) of a particular sample, meta-learning could extract information from data of other samples, learning the periodicity inherent in this distribution. In contrast, local polynomial regression relies solely on the data of an individual sample for estimation, resulting in less accurate predictions for the first half ($t \in [0, 5]$) of the time series. Additionally, meta-learning inherently learns information about the mean function of this distribution. By directly plugging the initial values ϕ_0 learned by MAML into the SIREN architecture to form the meta-model $F_{\phi_0}(t)$, we observed a notable proximity between the meta-model and the means of these samples. As demonstrated in Figure 2b, the meta-learning process is accompanied by the meta-model gradually converging towards the mean. Aside from these two examples, [4] illustrates that MAML accelerates learning through feature reuse. This also clarifies why the meta-model encompasses various features of the dataset and MetaINR is a more reliable way for sample recovery than local polynomial regression.

Estimation of mean, covariance and principal components Based on MetaINR workflow, we estimated the mean function, covariance kernel, and principal components of this synthetic dataset. As demonstrated in Figure 3, the estimators based on MetaINR for all metrics significantly outperformed the benchmarks.

In the context of sample recovery, the Pre-smoothing method did not take into account information from other samples in the estimation process, leading to significant deviations from the true values within the $[0, 1]$ interval. On the other hand, although the PACE method considered information from all samples and produced relatively reasonable sample recovery results, its performance still falls

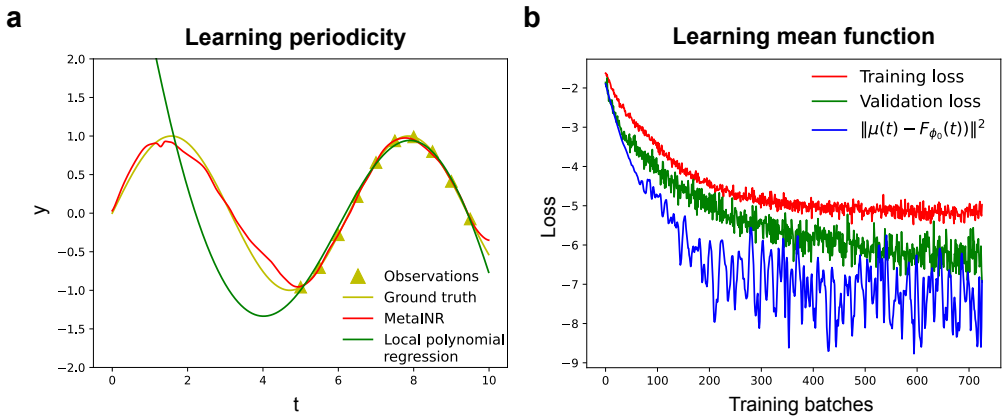

Figure 2: **Learning commonalities.** The figure illustrates two examples (periodicity & mean function) of the "shared characteristics" that MAML learned among these tasks. It is noticable that MetaINR is able to learn the periodicity between the interval $[0, 5]$ while local polynomial regression totally misses this information. It is also clear that the meta-model gradually approximates the true mean function during the training process.

short compared to MetaINR. This discrepancy may be attributed to the data generation process not meeting the assumptions of the PACE method.

Regarding mean function estimation, the Pre-smoothing method's estimation assigned equal weights to all samples at different time points. However, since the number and positions of observed points vary among different samples, the information provided by different samples for estimating the mean at a given time point varies. As a result, mean estimation based on the Pre-smoothing method still exhibits significant bias, whereas the estimations from PACE and MetaINR are comparatively better.

In covariance kernel estimation, both Pre-smoothing and PACE methods exhibit substantial differences from the ground truth. whereas the estimators based on MetaINR remain relatively reliable. During our experiments, we observed that the estimation results of these two methods are highly sensitive to the selection of bandwidth in the local polynomial regression process, indicating that their estimations are not sufficiently robust.

Regarding Principal Component (PC) estimation, we found that all three methods performed well in estimating PC1. However, both the Pre-smoothing and PACE methods struggled to accurately capture the variation of PC2 over time, while MetaINR nearly perfectly estimated PC2.

4.2 Real-world data

Data description In real-world scenarios, we apply MetaINR to the longitudinal data on primary biliary cirrhosis (PBC). The data resulted from a Mayo Clinic trial that was conducted in 1974 to 1984, which is also described in [20], [21] and [22]. They are available at `http://lib.stat.cmu.edu/datasets/pbc` and at a R package called "survival". The data contain various sparsely and irregularly sampled covariates for each of 312 patients. Among these medical measurements, serum bilirubin concentration is known to be elevated in the presence of chronic liver cirrhosis, such as PBC. Therefore, it is important to analyze the time series data of bilirubin measurements for PBC prediction. We extract data from the dataset for the initial 910 days, analyzing bilirubin measurements (in mg/dl) for different patients during the pre-onset phase. Our goal is to characterize the pattern of this measurement over time through estimates of mean function, covariance kernel and principal components. In practice, we applied a logarithmic transformation to the bilirubin measurements and removed data for patients with a survival time less than 910 days. A total of 260 patients in the study satisfied these criteria and only 2.9 observations per sample on average.

Parameters setting In the PBC dataset, we set the initial parameters of SIREN $\omega_0 = 0.00005$, since the frequency spectrum data distribuiton is relatively low. During the meta-learning training process, we equally divide the observations of each samples into support set and query set. In the outer loop of

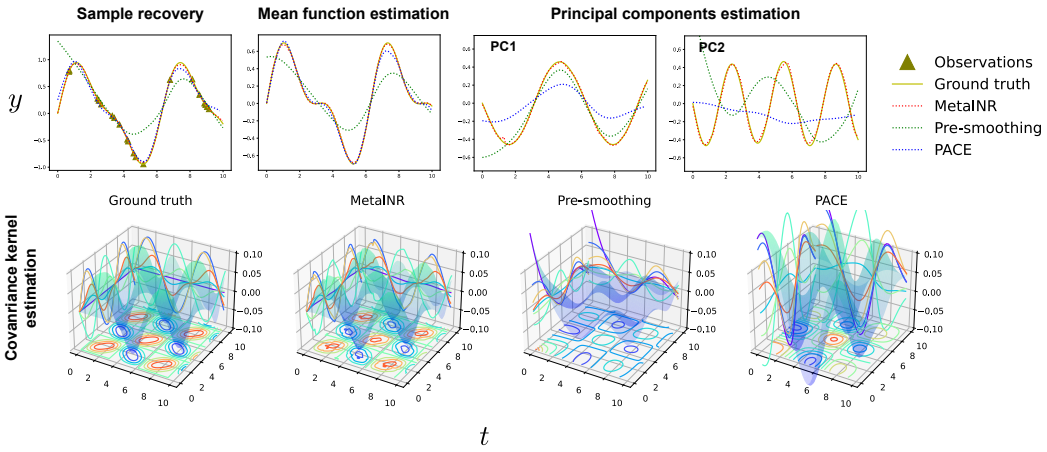

Figure 3: **Estimation for synthetic data.** The figure shows the results of the sample recovery and the estimation of mean function, covariance kernel and principal components in synthetic data. In each plot, the horizontal axis represents time, and the vertical axis represents the corresponding variable's value. It is clear that for all estimatiors, the performance of MetaINR is significantly better than that of Pre-smoothing and PACE methods.

MAML, we set the learning rate of the meta-model to 0.00005. In the inner loop of MAML, we set the model's learning rate to 0.00005 and the number of iterations for parameter optimization $Q = 3$. During subsequent gradient decent algorithm for sample recovery, the model's learning rate is set to 0.00005 and the total iterations is set to 500.

Estimation of mean, covariance and principal components Figure 4 illustrates the estimation results of MetaINR on the real-world medical sparse functional dataset mentioned above. We observed that despite the minimal number of observations per sample, MetaINR can effectively recover the sample function. Regarding the estimation of statistical quantities, MetaINR demonstrates similar performance to the benchmarks in this specific real dataset.

Conclusion From the estimation results, we draw the following conclusions: Over the initial 910 days of observation, patients' serum bilirubin concentration initially experiences a slight decrease, followed by a sharp increase at around 400 days (mean function). Apart from differences in "baseline values" which remain invariant with time (PC1), significant divergence in serum bilirubin concentration among patients emerges around 400 days (PC2).

5 Conclusion and future work

In this paper, we have shown a method named MetaINR utilizing meta-learning to analyze sparse time series data. MetaINR enables the estimation of functional form samples with very few observation points by leveraging information from the entire dataset through meta-learning. The application of MetaINR makes functional data analysis (FDA) methods, which are typically used in dense datasets, feasible for sparse datasets. Therefore, MetaINR provides a novel tool for further analyzing sparse time series data.

In future research, we aim to extend the application of MetaINR to other functional data analysis methods that were initially designed for dense datasets, such as Functional Partial Least Squares Regression (FPLS), functional regression, registration and more. We can also explore other algorithms of meta-learning, such as MANN [23] and SNAIL [24], and compare the performances between them.

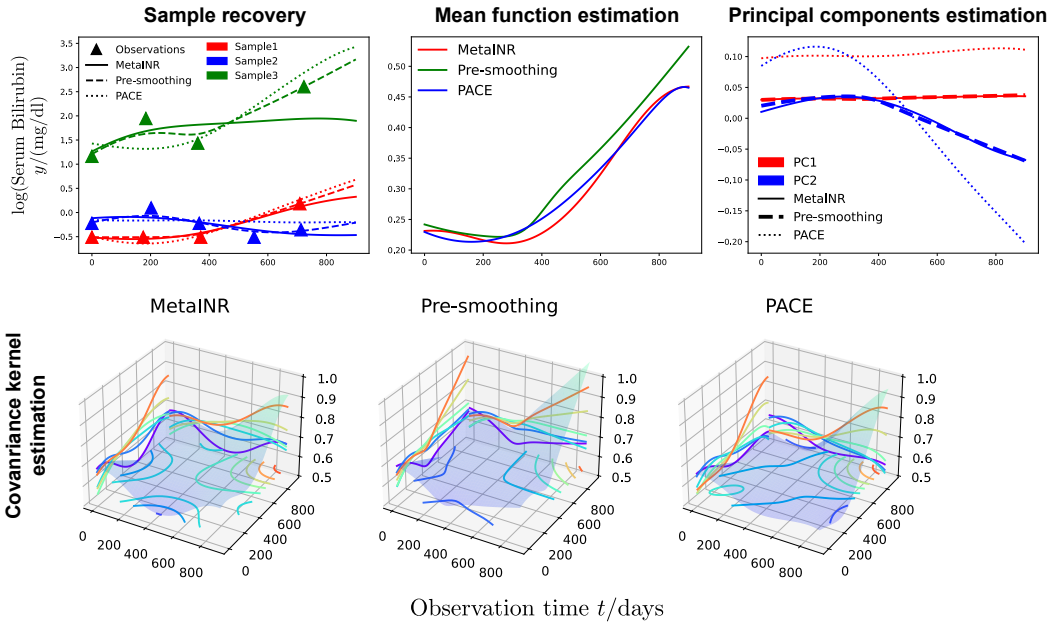

Figure 4: **Estimation for real-world data.**The figure demonstates the results of the sample recovery (three typical examples), and the estimation of statistical quantities (mean function, covariance kernel and principal components) in real-world PBC data. In each plot, the horizontal axis represents the observation time (days), and the vertical axis represents the value of logarithmic serum bilirubin concentration (mg/dl). The estimated results are similar in all three methods.

6 Appendix

6.1 Traditional FDA method for sparse functional data: PACE [13]

In functional data analysis, it is inappropriate to apply dense-data-based approaches directly on sparse dataset. However, PACE method gives a alternative solution.

Although local polynomial regression is not suitable for a single sample with sparse observations, it is proved that one can employ this method to estimate the mean function and covariance operator consistently. Specifically, given our sparse observations of dataset $D = \{(t_{ij}, y_{ij})_{j=1}^{J_i}\}_{i=1}^{n}$, we can define the loss function to estimate mean function $\mu(t)$

$$L_t^{\mu}(\beta) = \sum_{i=1}^{n}\sum_{j=1}^{J_i}\left[y_{ij} - \sum_{r=0}^{R}\beta_r\left(t - t_{ij}\right)^r\right]^2 K\left(\frac{t - t_{ij}}{h}\right),$$

where $K(\cdot)$ is the kernel function and h is the bandwidth which shoud be carefully chosen based on data. By minimizing this function, we can get $\beta_t = (\beta_0(t), \cdots, \beta_R(t)) = arg\,min L_t^{\mu}(\beta)$. The estimator of mean function cam be defined as $\hat{\mu}(t) = \beta_0(t)$. Similarly, to estimate the covariance kernel $c(t, s)$, we define the loss function

$$L_{t,s}^{c}(\beta) = \sum_{i=1}^{n}\sum_{1\le j,l,\le J_i} K_{t,s}(\frac{t - t_{ij}}{h}, \frac{s - t_{il}}{h})\left[g(t_{ij}, t_{il}) - f\left(\beta, (t, s), (t_{ij}, t_{il})\right)\right]^2$$

where

$$g(t_{ij}, t_{il}) = (y_{ij} - \hat{\mu}(t_{ij}))(y_{il} - \hat{\mu}(t_{il}))$$
$$f\left(\beta, (t, s), (t_{ij}, t_{il})\right) = \beta_0 + \beta_{11}(t - t_{ij}) + \beta_{12}(s - t_{il})$$

By minimization of $L_{t,s}^{c}(\beta)$, we obtain $(\beta_0(t), \beta_{11}(t), \beta_{12}(t)) = arg\,min L_{t,s}^{c}(\beta)$. Then the estimator of the covariance kernel is defined as $\hat{c}(t, s) = \beta_0(t, s)$. Under suitable conditions, including the

fact that observations are random and well-spread, the estimators of mean function and covariance kernel are consistent [13] .

$$\sup_{t\in\mathcal{T}}|\hat{\mu}(t)-\mu(t)|=O_p\left(\frac{1}{\sqrt{n}h_\mu}\right)$$
$$\sup_{t,s\in\mathcal{T}}|\hat{c}(s,t)-c(s,t)|=O_p\left(\frac{1}{\sqrt{n}h_G^2}\right)$$

where h_μ and h_G are bandwidths for estimating $\hat{\mu}$ and $\hat{c}$ satisfying

$$h_\mu\to 0, nh_\mu^4\to\infty, \text{ and } nh_\mu^6<\infty;$$
$$h_G\to 0, nh_G^6\to\infty, \text{ and } nh_G^8<\infty.$$

Base on the estimated covariance kernel, we can derive the estimated eigenvalues $\hat{\lambda}$ and eigenfunctions $\hat{\phi}_k(t)$ which are also consistent, ***i.e.***

$$\left|\hat{\lambda}_k-\lambda_k\right|=O_p\left(\frac{1}{\sqrt{n}h_G^2}\right);$$
$$\sup_{t\in\mathcal{T}}\left|\hat{\phi}_k(t)-\phi_k(t)\right|=O_p\left(\frac{1}{\sqrt{n}h_G^2}\right),\quad k\in\mathcal{I}'.$$

Let

$$\tilde{x}_i=\left(x_i\left(t_{i,1}\right),\ldots,x_i\left(t_{i,J_i}\right)\right)^T,$$
$$\tilde{\mu}_i=\left(\widehat{\mu}\left(t_{i,1}\right),\ldots,\widehat{\mu}\left(t_{i,J_i}\right)\right)^T,$$
$$\tilde{\phi}_{i,k}=\left(\widehat{\phi}_k\left(t_{i,1}\right),\ldots,\widehat{\phi}_k\left(t_{i,J_i}\right)\right)^T.$$

Assume that $X_i(t)=\mu(t)+\sum_{k=1}^{\infty}a_{ik}\phi_k(t)$, with the $a_{ik}\sim N\left(0,\lambda_k\right)$, Principal Analysis by Conditional Expectation (PACE) reconstructs each sample and estimates their scores by $\widehat{a}_{ik}=\hat{\mathbb{E}}\left(a_{ik}\mid\tilde{x}_i\right)$. It can be shown that

$$\widehat{a}_{i,k}=\widehat{\lambda}_k\tilde{\phi}_{i,k}^T\Sigma_{x_i}^{-1}\left(\tilde{x}_i-\tilde{\mu}_i\right)$$

where $\left\{\Sigma_{x_i}\right\}_{j,l}=\widehat{c}\left(t_{ij},t_{il}\right)$. The recovered sample can be represented by the first K components

$$\widehat{X}_i^K(t)=\widehat{\mu}(t)+\sum_{k=1}^{K}\widehat{a}_{i,k}\widehat{\phi}_k(t)$$

It can also be proved [13] that under suitable conditions

$$\lim_{n\to\infty}\hat{a}_{ik}=a_{ik}\text{ in probability,}$$

and for all $t\in\mathcal{T}$,

$$\lim_{K\to\infty}\lim_{n\to\infty}\widehat{X}_i^K(t)=X_i(t)\text{ in probability.}$$

6.2 The initialization scheme of SIREN [14]

As discussed in the main text, SIREN is a restructured network architecture based on the traditional MLP, where the sine function replaces the original ReLU function as the activation function of the neural network. It can be represented as follows:

$$\Phi(\mathbf{x})=\mathbf{W}_n\left(\phi_{n-1}\circ\phi_{n-2}\circ\ldots\circ\phi_0\right)(\mathbf{x})+\mathbf{b}_n,\quad\mathbf{x}_i\mapsto\phi_i\left(\mathbf{x}_i\right)=\sin\left(\mathbf{W}_i\mathbf{x}_i+\mathbf{b}_i\right)$$

In practice, we need to carefully design its initialization scheme to ensure an effective training process. If we randomly initialize weights for SIREN, it can lead to inaccurate learning results and slow convergence speed.

The key principle of SIREN's initialization is to ensure that the output distribution of SIREN remains invariant regardless of the number of layers. Consider the input of layer $\mathbf{x}\in\mathbb{R}^n$, which follows arcsine distribution

$$\mathbf{x}\sim\text{Arcsin}(-1,1)$$

$$(X \sim \operatorname{Arcsin}(a,b), \text{ with CDF: } F_X(x) = \frac{2}{\pi}\arcsin\left(\sqrt{\frac{x-a}{b-a}}\right), \text{ with } b > a)$$

when we draw weights $\mathbf{W}$ whose components follow

$$w_i \sim Uniform(-\sqrt{6/n}, \sqrt{6/n}),$$

it is proved that the dot product of weights and input of each layer converges to the normal distribution

$$\mathbf{w}^T\mathbf{x} \sim \mathcal{N}(0,1),$$

and the output of each layer (input of next layer) $\mathbf{x}^+ = \phi_i(\mathbf{x}) = \sin(\mathbf{W}\mathbf{x} + \mathbf{b})$ converges to an another arcsine distribution

$$\mathbf{x}^+ \sim \operatorname{Arcsin}(-1,1).$$

This property makes it posible that the output distribution of SIREN become invariant to the network's depth if we choose $\mathbf{W} \sim Uniform(-\sqrt{6/n}, \sqrt{6/n})$.

Besides, we also need to carefully choose the weights in the frist layer to match the frequency spectrum of the signal. If the weights of first layer are too small, the activation sine function will degenerate to a linear function; if the weight are too large, the nonlinearity of activation function are too large: Both of these extremes can lead to a decrease in SIREN's learning ability. Therefore, we introduce a new factor ω_0 and draw the weights of first layer by

$$\mathbf{W_0} = \omega_0 \times \mathbf{W},$$

which should satisfy that the activation function of first layer $\sin(\omega_0 \cdot \mathbf{W}\mathbf{x_0} + \mathbf{b_0})$ spans multiple periods over $[-1,1]$.

In practice, it is also found that the training speed of SIREN can be accelerated by using ω_0 in the activation function of *all* layers. Replacing $\mathbf{W} \sim Uniform(-\sqrt{6/n}, \sqrt{6/n})$ by $\widehat{\mathbf{W}} \sim Uniform\left(-\sqrt{\frac{c}{\omega_0^2 n}}, \sqrt{\frac{c}{\omega_0^2 n}}\right)$ will not only keep the activations constant but also boosts the gradients.

Data and code availability

PBC data used in this paper is available at `http://lib.stat.cmu.edu/datasets/pbc` , and it also can be found in a R package called "survival". All the experimental code in this paper is available at `https://github.com/cbfcbf/code_metaINR`.

Acknowledgments

We express our gratitude to Prof. Mihaela van der Schaar for her invaluable supervision. Additionally, we extend our thanks to the research group members of the van der Scaar lab, Dr. Kasia Kobalczyk and Dr. Krzysztof Kacprzyk, for their insightful discussions and constructive suggestions. The input from these individuals significantly enhanced the quality of this paper.

References

[1] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[2] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.

[3] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A survey of meta-reinforcement learning," *arXiv preprint arXiv:2301.08028*, 2023.

[4] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," *arXiv preprint arXiv:1909.09157*, 2019.

[5] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 403–412.

[6] J.-L. Wang, J.-M. Chiou, and H.-G. Müller, "Functional data analysis," *Annual Review of Statistics and its application*, vol. 3, pp. 257–295, 2016.

[7] P. Kokoszka and M. Reimherr, *Introduction to functional data analysis*. Chapman and Hall/CRC, 2017.

[8] J. H. Stock and M. W. Watson, "Vector autoregressions," *Journal of Economic perspectives*, vol. 15, no. 4, pp. 101–115, 2001.

[9] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[10] C. Francq and J.-M. Zakoian, *GARCH models: structure, statistical inference and financial applications*. John Wiley & Sons, 2019.

[11] S. Holte, T. Randolph, J. Ding, J. Tien, R. McClelland, J. Baeten, and J. Overbaugh, "Efficient use of longitudinal cd4 counts and viral load measures in survival analysis," *Statistics in medicine*, vol. 31, no. 19, pp. 2086–2097, 2012.

[12] H.-g. Müller, "Functional modelling and classification of longitudinal data," *Scandinavian Journal of Statistics*, vol. 32, no. 2, pp. 223–240, 2005.

[13] F. Yao, H.-G. Müller, and J.-L. Wang, "Functional data analysis for sparse longitudinal data," *Journal of the American statistical association*, vol. 100, no. 470, pp. 577–590, 2005.

[14] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, vol. 33, pp. 7462–7473, 2020.

[15] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, "Learning shape templates with structured implicit functions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7154–7164.

[16] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.

[17] J. Lee, J. Tack, N. Lee, and J. Shin, "Meta-learning sparse implicit neural representations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 769–11 780, 2021.

[18] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[19] S. I. Holt, Z. Qian, and M. van der Schaar, "Neural laplace: Learning diverse classes of differential equations in the laplace domain," in *International Conference on Machine Learning*. PMLR, 2022, pp. 8811–8832.

[20] B. H. Markus, E. R. Dickson, P. M. Grambsch, T. R. Fleming, V. Mazzaferro, G. B. G. Klintmalm, R. H. Wiesner, D. H. Van Thiel, and T. E. Starzl, "Efficacy of liver transplantation in patients with primary biliary cirrhosis," *New England Journal of Medicine*, vol. 320, no. 26, pp. 1709–1713, 1989.

[21] T. R. Fleming and D. P. Harrington, *Counting processes and survival analysis*. John Wiley & Sons, 2013, vol. 625.

[22] P. A. Murtaugh, R. E. Dickson, G. M. Van Dam, M. Malinchoc, P. M. Grambsch, A. L. Langworthy, and C. H. Gips, "Primary biliary cirrhosis: prediction of short–term survival based on repeated patient visits," *Hepatology*, vol. 20, no. 1, pp. 126–134, 1994.

[23] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1842–1850.

[24] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.