Machine Learning (2022) 111:1239–1257
https://doi.org/10.1007/s10994-021-06118-z

Few-shot learning for spatial regression via neural embedding-based Gaussian processes

Tomoharu Iwata[1] · Yusuke Tanaka[1]

Received: 1 December 2020 / Revised: 4 August 2021 / Accepted: 25 October 2021 /
Published online: 12 November 2021

Abstract
We propose a few-shot learning method for spatial regression. Although Gaussian processes (GPs), or kriging, have been successfully used for spatial regression, they require many observations in the target task to achieve a high predictive performance. Our model is trained using spatial datasets on various attributes in various regions, and predicts values on unseen attributes in unseen regions given a few observed data. With our model, a task representation is inferred from given small data using a neural network. Then, spatial values are predicted by neural networks with a GP framework, in which task-specific properties are controlled by the task representations. The GP framework allows us to analytically obtain predictions that are adapted to small data. By using the adapted predictions in the objective function, we can train our model efficiently and effectively so that the test predictive performance improves when adapted to newly given small data. In our experiments, we demonstrate that the proposed method achieves better predictive performance than existing meta-learning methods using spatial datasets.

Keywords Few-shot learning · Spatial regression · Gaussian processes · Meta-learning · Neural networks

1 Introduction

Thanks to the development of sensors, GPS devices, and satellite systems, a wide variety of spatial data are being accumulated, including climate (Stralberg et al. 2015; Wang et al. 2016b), traffic (Zheng et al. 2014; Yuan et al. 2011), economic, and social data (Haining 1993; Shadbolt et al. 2012). Analyzing such spatial data is critical in various fields, such as environmental sciences (Jerrett et al. 2005; Hession and Moore 2011), urban planning (Yuan et al. 2012), socio-economics (Smith-Clarke et al. 2014; Rupasingha and Goetz 2007), and public security (Bogomolov et al. 2014; Wang et al. 2016a).

Editor: Andrea Passerini.

Tomoharu Iwata
tomoharu.iwata.gy@hco.ntt.co.jp

[1] NTT Communication Science Laboratories, Kyoto, Japan

Collecting data on some attributes is difficult if the attribute-specific sensing devices are very expensive, or experts that have extensive domain knowledge are required to observe data. Also, collecting data in some regions is difficult if they are not readily accessible. To counter these problems, many spatial regression methods have been proposed (Gao et al. 2006a, b; Ward and Gleditsch 2018); they predict missing attribute values given data observed at some locations in the region. Although Gaussian processes (GPs) (Rasmussen and Williams 2006; Banerjee et al. 2008) have been successfully used for spatial regression, they fail when the data observed in the target region are insufficient.

In this paper, we propose a few-shot learning method for spatial regression. Our model learns from spatial datasets on various attributes in various regions, and predicts values when observed data in the target task is scant, where both the attribute and region of the target task are different from those in the training datasets. Figure 1 illustrates the framework of the proposed method. Some attributes in some regions are expected to exhibit similar spatial patterns to the target task. Our model uses the knowledge learned from such attributes and regions in the training datasets to realize prediction in the target task.

Our model uses a neural network to embed a few labeled data into a task representation. Then, target spatial data are predicted based on a GP with neural network-based mean and kernel functions that depend on the inferred task representation. We call our model the neural embedding-based Gaussian processes. Using the task representation yields a task-specific prediction function. By basing the modeling on GPs, the prediction function can be rapidly adapted to small labeled data in a closed form without iterative optimization, which enables efficient back-propagation through the adaptation. As the mean and kernel functions employ neural networks, we can flexibly model spatial patterns in various attributes and regions. By sharing the neural networks across different tasks in our model, we can learn from multiple attributes and regions, and use the learned knowledge to handle new attributes and regions. The neural network parameters are estimated by maximizing the expected prediction performance when a few observed data are given, which is calculated using training datasets by an episodic training framework (Ravi and Larochelle 2017; Santoro et al. 2016; Snell et al. 2017; Finn et al. 2017; Li et al. 2019).

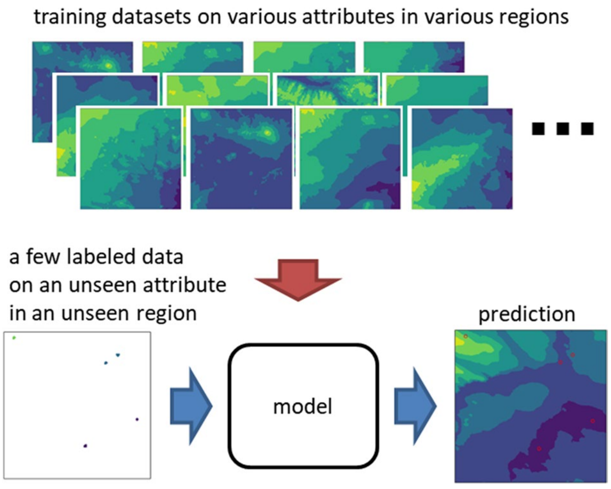

Fig. 1 Our framework. In a training phase, our model learns from training datasets containing various attributes from various regions. In a test phase, our model predicts spatial values of a target attribute in a target region given a few observations; the target attribute and region are not present in the training datasets

The main contributions of this paper are as follows:

1. We present a framework of few-shot learning for spatial regression.
2. We propose a GP-based model that uses neural networks to learn spatial patterns from various attributes and regions.
3. We empirically demonstrate that the proposed method performs well in few-shot spatial regression tasks.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. In Sect. 3, we define our task, propose a few-shot learning model for spatial regression based on neural embedding-based Gaussian processes, and develop its training procedure. Section 4 experimentally demonstrates the effectiveness of the proposed method using climate data. Finally, we present concluding remarks and discuss future work in Sect. 5.

2 Related work

GPs, or kriging (Cressie 1990), have been widely used for spatial regression (Banerjee et al. 2008; Luttinen and Ilin 2009; Park et al. 2011; Stein 2012; Gu and Hu 2012). They achieve high prediction performance at locations that are close to the observed locations. However, if the target region is large and only a few observed data are given, performance falls at locations far from the observed locations. For improving generalization performance, neural networks have been used for mean and/or kernel functions of GPs (Wilson et al. 2011; Huang et al. 2015; Calandra et al. 2016; Wilson et al. 2016a, b; Iwata and Ghahramani 2017; Iwata and Otsuka 2019; Jean et al. 2018). However, these methods require a lot of training data.

Many few-shot learning, or meta-learning, methods have been proposed (Schmidhuber 1987; Bengio et al. 1991; Ravi and Larochelle 2017; Andrychowicz et al. 2016; Vinyals et al. 2016; Snell et al. 2017; Bartunov and Vetrov 2018; Finn et al. 2017; Li et al. 2017; Kim et al. 2018; Finn et al. 2018; Rusu et al. 2019; Yao et al. 2019; Edwards and Storkey 2016; Garnelo et al. 2018a; Kim et al. 2019; Hewitt et al. 2018; Bornschein et al. 2017; Reed et al. 2017; Rezende et al. 2016). Since our task is regression, few-shot classification methods, such as matching networks (Vinyals et al. 2016) and prototypical networks (Snell et al. 2017), are not applicable. Existing few-shot learning methods that can handle regression tasks are applicable for our task, such as model-agnostic meta-learning (Finn et al. 2017) and conditional neural processes (Garnelo et al. 2018a). However, they are not intended for spatial regression. On the other hand, our model is based on GPs, which have been successfully used for spatial regression. Some few-shot learning methods based on GPs have been proposed (Harrison et al. 2018; Tossou et al. 2019; Fortuin et al. 2019). Adaptive learning for probabilistic connectionist architectures (ALPaCA) (Harrison et al. 2018) and adaptive deep kernel learning (Tossou et al. 2019) incorporate the information in small labeled data in kernel functions using neural networks, but they assume zero mean functions. Although meta-learning mean functions (Fortuin et al. 2019) use a neural network for the mean function, the mean function does not change outputs depending on the given small labeled data. On the other hand, the proposed method uses a neural network-based mean function that outputs task-specific values by extracting a task representation from the small labeled data. The effectiveness of our mean function is shown in the ablation study in our experiments. Task-similarity aware nonparametric meta-learning

(TANML) (Venkitaraman and Wahlberg 2020) is related to the proposed method since both are meta-learning methods that use kernels. TANML uses kernels for calculating the similarity between tasks. In contrast, the proposed method uses kernel for calculating covariance between locations in each task.

Our model is related to conditional neural processes (NPs) (Garnelo et al. 2018a, 2018b) as both use neural networks for task representation inference and for prediction with inferred task representations. However, since NP prediction is based on fully parametric models, they are less flexible in adapting to the given target observations than GPs, which are nonparametric models. In contrast, our GP-based model enjoys the benefits of the non-parametric approach, swift adaptation to the target observations, even though the mean and kernel functions are modeled parametrically. Our model is also related to similarity-based meta-learning methods, such as matching networks (Vinyals et al. 2016) and prototypical networks (Snell et al. 2017), since the kernel function represents similarities between data points. Although existing similarity-based meta-learning methods were designed for classification tasks, our model is designed for regression tasks.

The proposed method is also related to model-agnostic meta-learning (MAML) (Finn et al. 2017) in the sense that both methods trains models so that the expected error on unseen data is minimized when adapted to a few observed data. For the adaptation, MAML requires costly back-propagation through iterative gradient descent steps. On the other hand, the proposed method achieves an efficient adaptation in a closed form using a GP framework. Ridge regression differentiable discriminator (R2D2) (Bertinetto et al. 2018) is a neural network-based meta-learning method, where the last layer is adapted by solving a ridge regression problem in a closed form. Although R2D2 and (Lee et al. 2019) adapt with a linear model, the proposed method adapts with a nonlinear GP model, which enables us to adapt to complicated patterns more flexibly.

Adaptively initialized task optimizer (AVIATOR) (Ye et al. 2020) and multimodal MAML (MMAML) (Vuorio et al. 2019) extended MAML by modifying models using task representations. In particular, AVIATOR uses task representations for generating initial model parameters, and MMAML uses task representations for generating parameters that modulate models. The proposed method is related to them in that the task representation is used to define a model, then the model is adapted by minimizing a loss on the support set.

Transfer learning methods, such as multi-task GPs (Yu et al. 2005; Bonilla et al. 2008; Wei et al. 2017) and co-kriging (Myers 1982; Stein and Corsten 1991), have been proposed; they transfer knowledge derived from source tasks to target tasks. However, they do not assume a few observations in target tasks. In addition, since these methods use target data to learn the relationship between source and target tasks, they require computationally costly re-training given new tasks that are not present in the training phase. On the other hand, the proposed method can be applied to unseen tasks by inferring task representations from a few observations without re-training.

3 Proposed method

3.1 Task

In a training phase, we are given spatial datasets for $|\mathcal{R}|$ regions, $\mathcal{D} = \{\mathcal{D}_r\}_{r\in\mathcal{R}}$, where $\mathcal{R}$ is the set of regions, and $\mathcal{D}_r$ is the dataset for region r. For each region, there are $|\mathcal{C}_r|$ attributes, $\mathcal{D}_r = \{\mathcal{D}_{rc}\}_{c\in\mathcal{C}_r}$, where $\mathcal{C}_r$ is the set of attributes in region r, and $\mathcal{D}_{rc}$ is the dataset of attribute c in region r. The attribute sets can be different across the regions. Each dataset consists of a set of location vectors and attribute values, $\mathcal{D}_{rc} = \{(\mathbf{x}_{rcn}, y_{rcn})\}_{n=1}^{N_{rc}}$, where $\mathbf{x}_{rcn} \in \mathbb{R}^2$ is a two-dimensional vector specifying the location of the nth point, e.g., longitude and latitude, and $y_{rcn} \in \mathbb{R}$ is the scalar value on attribute c at that location.

In a test phase, we are given a few labeled observations in a target region, $\mathcal{D}_{r^*c^*} = \{(\mathbf{x}_{r^*c^*n}, y_{r^*c^*n})\}_{n=1}^{N_{r^*c^*}}$, where target region r^* is not one of the regions in the training datasets, $r^* \notin \mathcal{R}$, and target attribute c^* is not contained in the training datasets, $c^* \notin \mathcal{C}_r$ for all $r \in \mathcal{R}$. Our task is to predict target attribute value $\hat{y}_{r^*c^*}$ at location $\mathbf{x}_{r^*c^*}$ in the target region.

Location vector $\mathbf{x}_{rcn}$ represents the relative position of the point in region r. We used longitudes and latitudes normalized with zero mean for the location vectors in our experiments. Spatial data sometimes include auxiliary information such as elevation. In that case, we can include the auxiliary information in $\mathbf{x}_{rcn} \in \mathbb{R}^{M+2}$, where M is the number of additional types of auxiliary information.

3.2 Preliminaries: Gaussian processes

Before introducing the proposed model, we review GP regression, which forms the basis of the proposed model. In GP regression, GPs are used for the prior of a nonlinear function,

$$g(x) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where $m(\mathbf{x})$ is a mean function, and $k(\mathbf{x}, \mathbf{x}')$ is a kernel function. Let $\mathbf{y} = (y_n)_{n=1}^N$ be the N-dimensional vector of the attribute values, and $\mathbf{X} = (\mathbf{x}_n)_{n=1}^N$ be the $N \times N$ matrix of the location vectors. The joint distribution of $\mathbf{y}$ given $\mathbf{X}$ with GP regression follows a Gaussian distribution,

$$p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{m}, \mathbf{K}), \tag{2}$$

where $\mathbf{m} = (m(\mathbf{x}_n))_{n=1}^N$ is the N-dimensional vector with the values of mean function $m(\cdot)$ at location vectors $\mathbf{X}$, $\mathbf{K}$ is the $N \times N$ matrix of the kernel function evaluated between location vectors $\mathbf{X}$, and $\mathbf{K}_{nn'} = k(\mathbf{x}_n, \mathbf{x}_{n'})$. The predictive distribution at test point $\mathbf{x}$ given $\mathbf{X}$ and $\mathbf{y}$ as the training data is,

$$p(y|\mathbf{x}, \mathbf{y}, \mathbf{X}) = \mathcal{N}(m(\mathbf{x}) + \mathbf{k}^\top \mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}), k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^\top \mathbf{K}^{-1}\mathbf{k}), \tag{3}$$

where $\mathbf{k}$ is the N-dimensional vector of the kernel function between the test point and training points, $\mathbf{k} = (k(\mathbf{x}, \mathbf{x}_n))_{n=1}^N$.

3.3 Model

Let $\mathcal{S} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ be a few labeled observations, which are called the *support set*. We here present our neural embedding-based Gaussian processes for predicting attribute scalar value $\hat{y}$ at location vector $\mathbf{x}$, which is called the *query*, given support set $\mathcal{S}$. Our model is used for training as described in Sect. 3.4 as well as target spatial regression in a test phase. Figure 2 illustrates our model. Our model infers task representation $\mathbf{z}$ from support set $\mathcal{S}$ as described in Sect. 3.3.1. Then, using the inferred task representation $\mathbf{z}$, we predict attribute scalar value $\hat{y}$ of location vector $\mathbf{x}$ by a neural network-based GP as described in Sect. 3.3.2. We omit indices for regions and attributes for simplicity in this subsection.

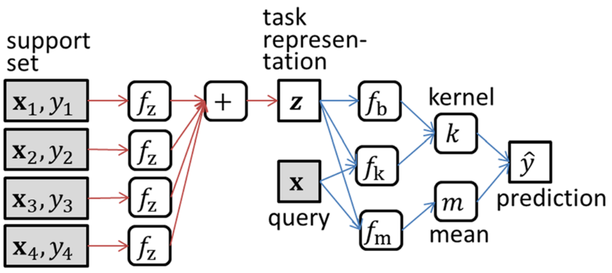

Fig. 2 Our model. Each pair of location vector $\mathbf{x}_n$ and attribute value y_n in a few labeled data set (support set) is fed to neural network f_z. By averaging the outputs of the neural network, we obtain task representation $\mathbf{z}$. The task representation and query location vector $\mathbf{x}$ are fed to neural networks f_b, f_k, and f_m to calculate kernel k and mean m. Attribute value $\hat{y}$ of the query location vector is predicted by using the kernel and mean based on a GP. Shaded nodes represent observed data

3.3.1 Inferring task representation

First, each pair of the location vector and attribute value, $(\mathbf{x}_n, y_n)$, in the support set is converted into K-dimensional latent vector $\mathbf{z}_n \in \mathbb{R}^K$ by a neural network: $\mathbf{z}_n = f_z([\mathbf{x}_n, y_n])$, where f_z is a feed-forward neural network with the $(M + 3)$-dimensional input layer and K-dimensional output layer, and $[\cdot, \cdot]$ represents the concatenation. Second, the set of latent vectors $\{\mathbf{z}_n\}_{n=1}^N$ in the support set are aggregated to K-dimensional latent vector $\mathbf{z} \in \mathbb{R}^K$ by averaging: $\mathbf{z} = \frac{1}{N}\sum_{n=1}^N \mathbf{z}_n$, which is a representation of the task extracted from support set $\mathcal{S}$. We can use other aggregation functions, such as summation (Zaheer et al. 2017), attention-based (Kim et al. 2019), and recurrent neural networks (Vinyals et al. 2016).

3.3.2 Predicting attribute values

Our prediction function assumes a GP with neural network-based mean and kernel functions that depend on the inferred task representation $\mathbf{z}$. In particular, the mean function is modeled by

$$m(\mathbf{x};\mathbf{z}) = f_m([\mathbf{x}, \mathbf{z}]), \tag{4}$$

where f_m is a feed-forward neural network that outputs a scalar value. The kernel function is modeled by

$$k(\mathbf{x}, \mathbf{x}';\mathbf{z}) = \exp\left(- \| f_k([\mathbf{x}, \mathbf{z}]) - f_k([\mathbf{x}', \mathbf{z}]) \|^2\right) + f_b(\mathbf{z})\delta(\mathbf{x}, \mathbf{x}'), \tag{5}$$

where f_k is a feed-forward neural network, f_b is a feed-forward neural network that outputs a positive scalar value, $\delta(\cdot, \cdot)$ is the Kronecker delta, $\delta(\mathbf{x}, \mathbf{x}') = 1$ if $\mathbf{x}$ and $\mathbf{x}'$ are identical, and zero otherwise. The kernel function is positive definite since it is a Gaussian kernel and $f_b(\mathbf{z})$ is positive. By incorporating task representation $\mathbf{z}$ in the mean and kernel functions using neural networks, we can model nonlinear functions that depend on the support set.

In GPs, zero mean functions are often used since the GPs with zero mean functions can approximate an arbitrary continuous function, if given enough data (Micchelli et al. 2006). However, GPs with zero mean functions predict zero at areas far from observed data points (Iwata and Ghahramani 2017), which is problematic in few-shot learning. Modeling the mean function by a neural network (4) allows us to predict values effectively even in areas far from observed data points in a target region due to the high generalization performance of neural networks.

Location vector $\mathbf{x}$ is transformed by neural network f_{k} before computing the kernel function by the Gaussian kernel in (5). The use of the neural network yields flexible modeling of the correlation across locations depending on the task representation. The noise parameter is also modeled by neural network f_{b}, which enables us to infer the noise level from the support set without re-training.

The predicted value for query $\mathbf{x}$ is given by

$$\hat{y}(\mathbf{x}, \mathcal{S}; \boldsymbol{\Phi}) = f_{\mathrm{m}}([\mathbf{x}, \mathbf{z}]) + \mathbf{k}^{\top}\mathbf{K}^{-1}(\mathbf{y} - \mathbf{m}), \tag{6}$$

where $\mathbf{K}$ is the $N \times N$ matrix of the kernel function evaluated between location vectors in the support set, $\mathbf{K}_{nn'} = k(\mathbf{x}_n, \mathbf{x}_{n'})$, $\mathbf{k}$ is the N-dimensional vector of the kernel function between the query and support set, $\mathbf{k} = (k(\mathbf{x}, \mathbf{x}_n))_{n=1}^{N}$, $\mathbf{y}$ is the N-dimensional vector of attribute values in the support set, $\mathbf{y} = (y_n)_{n=1}^{N}$, $\mathbf{m}$ is the N-dimensional vector of the mean function evaluated on locations in the support set, $\mathbf{m} = (f_{\mathrm{m}}([\mathbf{x}_n, \mathbf{z}]))_{n=1}^{N}$, and $\boldsymbol{\Phi}$ are the parameters of neural networks f_{z}, f_{m}, f_{k}, and f_{b}. An advantage of our model is that the predicted value given the support set is analytically calculated without iterative optimization, by which we can minimize the expected prediction error efficiently based on gradient-descent methods.

When noise $f_{\mathrm{b}}(\mathbf{z})$ is small, the predicted value approaches the observed values at locations close to the observed locations. This property of GPs is beneficial for few-shot regression without re-training. If a neural network without GPs is used for the prediction function, the predicted values might differ from the observations even at the observed locations when re-training based on the observations is not conducted. The first term in (6) is similar to conditional neural processes, where a neural network is used for the prediction function. The second term in (6) is related to similarity-based meta-learning methods since the second term uses the similarities between the query and support set that are calculated by the kernel function. Therefore, our model can be seen as an extension of the conditional neural process and similarity-based meta-learning approach, where both of them are naturally integrated within a GP framework. When $\mathbf{x}$ is far from (close to) the observed locations, the first (second) term becomes dominant due to kernel $\mathbf{k}$ (Iwata and Ghahramani 2017). This is reasonable since similarity-based approaches are more reliable when there are observations nearby. The variance of the predicted attribute value of the query is given by

$$\mathbb{V}[y|\mathbf{x}, \mathcal{S}; \boldsymbol{\Phi}] = k(\mathbf{x}, \mathbf{x}; \mathbf{z}) - \mathbf{k}^{\top}\mathbf{K}^{-1}\mathbf{k}. \tag{7}$$

3.4 Learning

We estimate neural network parameters $\boldsymbol{\Phi}$ by minimizing the expected prediction error on a query set given a support set using an episodic training framework (Ravi and Larochelle 2017; Santoro et al. 2016; Snell et al. 2017; Finn et al. 2017; Li et al. 2019). Although training datasets $\mathcal{D}$ contain many observations, they should be used in a way that closely simulates the test phase. Therefore, with the episodic training framework, support and query sets are generated by a random subset of training datasets $\mathcal{D}$ for each training iteration. In particular, we use the following objective function:

$$\hat{\boldsymbol{\Phi}} = \arg\min_{\boldsymbol{\Phi}} \mathbb{E}_{r\sim\mathcal{R}}[\mathbb{E}_{c\sim\mathcal{C}_r}[\mathbb{E}_{(\mathcal{S},\mathcal{Q})\sim\mathcal{D}_{rc}}[L(\mathcal{S},\mathcal{Q};\boldsymbol{\Phi})]], \tag{8}$$

where $\mathbb{E}$ represents an expectation,

$$L(\mathcal{S},\mathcal{Q};\boldsymbol{\Phi}) = \frac{1}{N_{\rm Q}} \sum_{(\mathbf{x},y)\in\mathcal{Q}} \parallel \hat{y}(\mathbf{x},\mathcal{S};\boldsymbol{\Phi}) - y \parallel^2, \tag{9}$$

is the mean squared error on query set $\mathcal{Q}$ given support set $\mathcal{S}$, and $N_{\rm Q}$ is the number of instances in the query set. Usually, GPs are trained by maximizing the marginal likelihood of training data (support set), where test data (query set) are not used. On the other hand, the proposed method minimizes the prediction error on a query set when a support set is observed, by which we can simulate a test phase and learn a model that improves the prediction performance on target tasks. When we want to improve the predictive density for each test location, we can use the following negative predictive log likelihood:

$$L(\mathcal{S},\mathcal{Q};\boldsymbol{\Phi}) = -\frac{1}{N_{\rm Q}} \sum_{(\mathbf{x},y)\in\mathcal{Q}} \log \mathcal{N}(y|\hat{y}(\mathbf{x},\mathcal{S};\boldsymbol{\Phi}), \mathbb{V}[y|\mathbf{x},\mathcal{S};\boldsymbol{\Phi}]), \tag{10}$$

instead of the mean squared error (9). This is related to training GPs with the log pseudo-likelihood (Rasmussen and Williams 2006), where the leave-one-out predictive log likelihood is used as the objective function. When we want to improve the predictive joint density for a set of test locations, we can use the following negative predictive log joint likelihood:

$$L(\mathcal{S},\mathcal{Q};\boldsymbol{\Phi}) = -\log \mathcal{N}(\mathbf{y}_{\rm Q}|\hat{\mathbf{y}}(\mathbf{X},\mathcal{S};\boldsymbol{\Phi}), \mathbb{V}[\mathbf{y}_{\rm Q}|\mathbf{X},\mathcal{S};\boldsymbol{\Phi}]), \tag{11}$$

where $\mathbf{y}_{\rm Q}$ is the $N_{\rm Q}$-dimensional vector of attribute values in the query set, $\hat{\mathbf{y}}(\mathbf{X},\mathcal{S};\boldsymbol{\Phi}) = (\hat{\mathbf{y}}(\mathbf{x},\mathcal{S};\boldsymbol{\Phi}))_{\mathbf{x}\in\mathcal{Q}}$ is the $N_{\rm Q}$-dimensional vector of predicted attribute values of the query set by Eq. (6), $\mathbb{V}[\mathbf{y}_{\rm Q}|\mathbf{X},\mathcal{S};\boldsymbol{\Phi}] = \mathbf{K}_{\rm Q} - \mathbf{K}_{\rm Q}^\top\mathbf{K}^{-1}\mathbf{K}_{\rm Q} \in \mathbb{R}^{N_{\rm Q}\times N_{\rm Q}}$ is the covariance of the query set, and $\mathbf{K}_{\rm Q} \in \mathbb{R}^{N_{\rm Q}\times N_{\rm Q}}$ is the kernel matrix evaluated on the query set by Eq. (5). The predictive likelihood has been used for a meta-learning method (Chen et al. 2020) instead of the marginal likelihood.

The training procedure of our model is shown in Algorithm 1. In each iteration, we randomly generate support and query sets (Lines 2 – 5) from dataset $\mathcal{D}_{rc}$ by randomly selecting region r and attribute c for simulating a test phase. Given the support and query sets so generated, we calculate the loss (Line 6). We update model parameters by using any of the stochastic gradient-descent methods, such as Adam (Kingma and Ba 2015) (Line 7). By training the model using randomly generated support and query sets, the trained model can predict values with a wide variety of observed location distributions, attributes and regions in a test phase.

The computational complexity for evaluating loss (9) and (10) is $O(N_{\rm Q} + N_{\rm S}^3)$, where $N_{\rm S}$ is the number of instances in the support set since we need the inverse of the kernel matrix with size $N_{\rm S} \times N_{\rm S}$. In few-shot learning, the number of target observed data is very small, and so a very small support size $N_{\rm S}$ is used in training. Therefore, our model can be optimized efficiently with the episodic training framework. This is in contrast to the high computational complexity of training for standard GP regression, which is cubic in the number of training instances. The computational complexity for evaluating loss (11) is $O(N_{\rm Q}^3 + N_{\rm S}^3)$ since we need the inverse of the covariance with size $N_{\rm Q} \times N_{\rm Q}$. When we use a large query set size for training, losses (9) and (10) are preferable to (11) in terms of computational efficiency.

Algorithm 1 Training procedure of our model.

```
Input: Spatial datasets $\mathcal{D}$, support set size $N_{\mathrm{S}}$, query set size $N_{\mathrm{Q}}$
Output: Trained neural network parameters $\boldsymbol{\Phi}$
1: while Stopping criteria is not met do
2:     Randomly sample region $r$ from $\mathcal{R}$
3:     Randomly sample attribute $c$ from $\mathcal{C}_r$
4:     Randomly sample support set $\mathcal{S}$ from $\mathcal{D}_{rc}$
5:     Randomly sample query set $\mathcal{Q}$ from $\mathcal{D}_{rc} \setminus \mathcal{S}$
6:     Calculate loss by Eq. (9), (10), or (11, and its gradients
7:     Update model parameters $\boldsymbol{\Phi}$ using the loss and its gradients
8: end while
```

4 Experiments

4.1 Data

We evaluated the proposed method using the following three spatial datasets: NAE, NA, and JA. NAE and NA were the climate data in North American, which were obtained from https://sites.ualberta.ca/~ahamann/data/climatena.html. As the location vector, NA used longitude and latitude. With NAE, elevation in meters above sea level was additionally used in the location vector. With NAE and NA data, we used the following 26 bio-climate values as attributes shown in Table 1. We generated 1829 non-overlapping regions covering North America, where the size of each region was 100×100km, and attribute values

Table 1 Attributes in NAE and NA data

Annual heat-moisture index	Climatic moisture deficit
Degree-days above 18 °C	Degree-days above 5 °C
Degree-days below 0 °C	Degree-days below 18 °C
Monthly reference evaporation	Length of the frost-free period
Mean annual precipitation	Mean annual temperature
Mean temperature of the warmest month	Number of frost-free days
Precipitation as snow	Summer (Jun–Aug) precipitation
Winter (Dec–Feb) precipitation	Monthly average relative humidity
Summer (Jun–Aug) mean temperature	Winter (Dec–Feb) mean temperature
Beginning of the frost-free period	End of the frost-free period
Summer heat-moisture index	
Mean temperature of the coldest month	
Mean summer (May–Sep) precipitation	
Extreme minimum temperature over 30 years	
Extreme maximum temperature over 30 years	
Difference between MCMT and MWMT as a measure of continentality	

Table 2 Attributes in JA data

Precipitation	Maximum temperature	Minimum temperature
Average temperature	Maximum snow depth	Sunshine duration
Total solar radiation		

Table 3 Statistics of NA, NAE and JA data

	NA/NAE	JA
# Training attributes	20	4
# Validation attributes	5	2
# Test attributes	1	1
# Training regions	50	200
# Validation regions	18	28
# Test regions	549	45

were observed at 1×1km grid squares in each region. JA was the climate data in Japan, which was obtained from http://nlftp.mlit.go.jp/ksj/gml/datalist/KsjTmplt-G02.html. We used the following seven climate values as attributes shown in Table 2. The data contained 273 regions, where attribute values were observed at 1×1km grid square, and there were at most 6,400 locations in a region. For all data, we randomly selected training, validation, and target regions without replacement. Also, we splitted the attributes into training, validation, and target attributes. The statistics of each data set are shown in Table 3. In each target region, values on a target attribute at five locations were observed, and values at the other locations were used for evaluation. The location vectors and attributes were normalized with zero mean and one standard deviation for each region and for each attribute.

4.2 Proposed method setting

As the neural networks in our model, f_{z}, f_{b}, f_{k}, and f_{m}, we used three-layered feed-forward neural networks with 256 hidden units. The dimensionality of the output layer with f_{z} and f_{k} was 256, and that with f_{b} and f_{m} was one. We used rectified linear unit, $\mathrm{ReLU}(x) = \max(0, x)$, for activation. Optimization was performed using Adam (Kingma and Ba 2015) with learning rate 10^{-3} and dropout rate 0.1. The maximum number of training epochs was 5000, and the validation datasets were used for early stopping. The support set size was $N_{\mathrm{S}} = 5$, and query set size was $N_{\mathrm{Q}} = 64$.

4.3 Comparison methods

We compared the proposed method with conditional neural process (NP), Gaussian process regression (GPR), Gaussian process autoencoder (GPVAE), neural network (NN), fine-tuning with NN (FT), model-agnostic meta-learning with NN (MAML), adaptively initialized task optimizer (AVIATOR), multimodal MAML (MMAML), prototypical networks (PN), and ridge regression differentiable discriminator (R2D2).

With NP, a task representation was inferred from the support set using a neural network as in the proposed method, and then the attribute values of queries were predicted using another neural network. We used the same neural network architecture with the proposed method for inferring task representations f_z. The architecture of the neural network for prediction was the same as that with f_m in the proposed method, which was used as the mean function.

GPR predicted the attribute values by a GP regression with a Gaussian kernel given the support set. The kernel parameters, which were the signal variance, length scale, and noise variance, were estimated from the training datasets by minimizing the expected prediction error using the episodic training framework.

GPVAE was a variational autoencoder (Kingma and Welling 2014) with GP priors on latent variables (Casale et al. 2018; Ashman et al. 2020). With GPVAE, latent variables were encoded using a neural network from location vectors. Then, attribute values were predicted by a decoder neural network from the latent variables. The parameters of encoder and decoder neural networks were estimated using the training datasets.

NN used a three-layered feed-forward neural network with 256 hidden units, and the ReLU activation was used. The input of the NN was a location vector, and its output was the predicted value of the attribute. NN parameters, which were shared across all tasks, were estimated using the training datasets. The NN did not use labeled data in target tasks.

FT fine-tuned the parameters of the trained NN with labeled data for each target task. For fine-tuning, we used Adam with learning rate 10^{-3}. The number of epochs for fine-tuning was 100, which was selected from $\{10, 100\}$ based on the target performance.

MAML used the same neural network as NN. The parameters were trained so that the prediction performance was improved when fine-tuned with a support set. The number of fine-tuning epochs was five. MAML was implemented with Higher, which is a library for higher-order optimization (Grefenstette et al. 2019).

AVIATOR and MMAML obtained a task representation using a neural network as in the proposed method. They were trained the neural network as in MAML, where the neural network was defined by the task representation. AVIATOR generated the initial parameters of a neural network using the task representation. MMAML generated parameters that modulate a neural network using the task representation.

With PN, a three-layered feed-forward neural network with 256 hidden and output units was used for embedding location vectors. Attribute values were predicted by a weighted average of the support instances, where the weights were calculated by softmaxed negative squared Euclidean distance between the query and support embedded instances.

R2D2 used a neural network of the same architecture with PN for embedding. A ridge regression model was used for predicting attribute values given embedded instances, where the ridge regression parameters were adapted using the support set for each region.

NP, GPR, GPVAE, NN, MAML, AVIATOR, MMAML, PN, and R2D2 used the episodic training framework in the same way as the proposed method. All the methods were optimized with Adam with learning rate 10^{-3}, and implemented with PyTorch (Paszke et al. 2017).

4.4 Results

The test mean squared errors (a) and test log likelihoods (b) in the target tasks averaged over ten experiments are shown in Table 4. The test log likelihoods were calculated for each test location. For the test mean squared error evaluations, all methods were trained with the mean squared error objective function in Eq. (9), and for the test log likelihood

Table 4 (a) Test mean squared errors and (b) test log likelihoods averaged over ten experiments

(a) Test mean squared errors

Data	Ours	NP	GPR	GPVAE	NN	FT
NAE	**0.316**	0.348	0.476	0.972	0.963	0.497
NA	**0.552**	0.593	0.701	0.991	0.986	0.746
JA	**0.653**	0.756	0.703	0.987	1.016	0.871

Data	MAML	AVIATOR	MMAML	PN	R2D2
NAE	0.710	0.346	0.342	0.647	0.400
NA	0.824	0.611	0.604	0.835	0.875
JA	0.873	0.956	0.985	1.073	1.914

(b) Test log likelihoods

Data	Ours	NP	GPR	GPVAE	NN	FT
NAE	**– 0.987**	– 1.025	– 1.067	– 1.316	– 1.281	– 1.271
NA	**– 1.166**	– 1.190	– 1.230	– 1.348	– 1.322	– 1.311
JA	**– 1.227**	– 1.307	**– 1.220**	– 1.369	– 1.333	– 1.598

Data	MAML	AVIA-TOR	MMAML	PN	R2D2
NAE	– 1.256	– 1.088	– 1.090	– 1.234	– 1.119
NA	– 1.306	– 1.216	– 1.208	– 1.300	– 1.337
JA	– 1.314	– 1.747	– 1.828	– 1.413	-1.879

Values in bold typeface are not statistically significantly different at the 5% level from the best performing method in each data according to a paired t-test

evaluations, all methods were trained with the negative log likelihood objective function in Eq. (10). The proposed method achieved the best performance in all cases except for the test likelihood with JA data. NP was worse than the proposed method because its predictions were poor when task representations were not properly inferred. On the other hand, the proposed method performed well with any tasks in at least areas close to the observations, as its GP framework offers a smooth nonlinear function that passes over the observations. GPR was worse than the proposed method since GPR only shares kernel parameters across different tasks. In contrast, the proposed method shares neural networks across different tasks, which enables us to learn flexible spatial patterns in various attributes and regions and use them for target tasks. NN and GPVAE suffered the low performance since they cannot use the target data. Fine-tuning (FT) decreased the error, but it remained worse than that of the proposed method. This is because FT consisted of two separate steps: pretrain and fine-tuning, and did not learn how to transfer knowledge. In contrast, the proposed method trained the neural network in a single step so that test performance is maximized when the support set is given in the episodic training framework. MAML performance was low since it had difficulty in learning the parameters that fine-tuned well with just a small number of epochs with various regions and attributes, where target function shapes vary drastically. Note that due to the high computational complexity of MAML, where it demands that the gradients of many gradient-descent steps be calculated, MAML makes it infeasible to use a large number of fine-tuning epochs. On the other

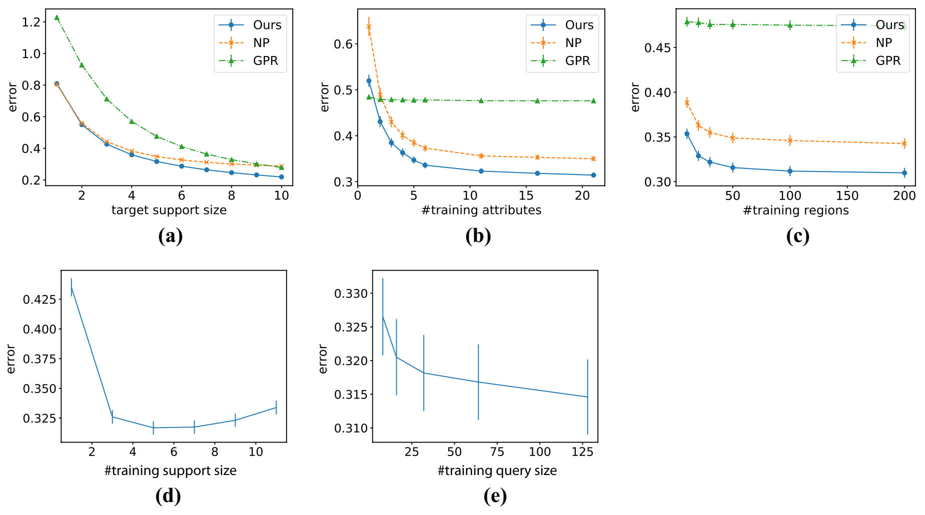

Fig. 3 Average test mean squared errors with (**a**) different target support sizes, (**b**) different numbers of training attributes, (**c**) different numbers of training regions, (**d**) different training support sizes, and (**e**) different training query sizes. The bar shows the standard error

hand, with the proposed method, since predicted values given the support set are calculated analytically based on a GP, the neural networks are optimized efficiently in terms of fitting the support set, and therefore the trained model attained high prediction performance for various attributes and regions. AVIATOR and MMAML used the task representation to obtain task-specific neural networks, and their performance was better than MAML. However, it was worse than the proposed method since the proposed method used GPs that were suitable for spatial regression. Since the number of training attributes was small with JA data, and training data were insufficient to train neural networks, the test likelihood of the proposed method was not significantly different from that of GPR. The expressive power of PN and R2D2 is low since they are adapted based on the weighted average and linear regression, respectively. Therefore, their performance was worse than the proposed method, which is adapated based on GPs.

Figure 3a shows the average test mean squared errors with different target support sizes with the proposed method, NP, and GPR. We omitted the results with NN, FT, and MAML since their performance was low as shown in Table 4. All methods yielded decreased error as the target support size increased. The proposed method achieved low errors with different target support sizes since it uses neural networks to learn the relationship between support and query sets using the training datasets. NP achieved low error rates when the target support size was small. However, NP had higher error rates than GPR when the size was ten. Since NP used a fixed trained neural network to incorporate the support set information, it was difficult to adapt prediction functions to a large support set. In contrast, since the proposed method and GPR can adapt them easily to support sets by calculating the posterior in a closed form, their errors were effectively decreased as the target support size increased.

Figure 3b shows the average test mean squared errors with different numbers of training attributes. The errors with the proposed method and NP decreased as the training attribute numbers increased. This is reasonable since the possibility that tasks similar to target tasks are included in the training datasets increases as the number of training attributes increases. Since GPR shared only kernel parameters across different tasks, its performance was not improved even when many attributes were used. Fig. 3c shows the average test mean squared errors with different numbers of training regions. The errors with the proposed method and NP decreased as the training regions increased. Figure 3d shows the average test mean squared errors with different training support sizes with the proposed method. When the support size in the training phase was the same with that in the test phase, i.e., $N_{\mathrm{S}} = 5$, the performance was best. When the test support size can differ in target tasks, we need to train with a wide range of training support sizes. Figure 3d shows the average test mean squared errors with different training query sizes with the proposed method. As the training query size increased, the performance improved. It would be because the evaluation of the test error in the training phase improved using larger training query size.

Table 5 shows the average computation time in seconds for learning from the training datasets and the time for predicting test attribute values for each region on computers with 2.60GHz CPUs. Although the proposed method had slightly longer training time than NP, GPR, or NN since it uses both the neural networks and GP, it was faster than MAML-based methods (MAML, AVIATOR and MMAML). All methods had short test times since the number of observed locations was small. The proposed method had shorter test time than FT because the proposed method calculated predictions analytically given the target data, while the FT required multiple updates for optimization given the target data.

Figure 4 visualizes the predictions for five attributes and regions of target tasks with the proposed method, NP, and GPR. The proposed method attained appropriate predictions in various attributes and regions. NP did not necessarily output predicted values that were similar to the observations. For example, in Fig. 4(a,NP), the predicted values of NP at two left observed locations differed from the true value. On the other hand, the proposed method and GPR predicted values similar to the observation at the locations. Since GPR could not extract the rich knowledge present in the training datasets, it sometimes failed to predict values. For example, in Fig. 4(a,GPR), the predicted values differed from the true values in the lower area. In contrast, the proposed method and NP predicted values at the area well using neural networks. The proposed method improved prediction performance by adopting both advantages of GPs and neural networks.

Table 6 shows the results of the ablation study of the proposed method. In terms of the test mean squared error, the proposed method with the mean squared error objective function (ErrObj) was better than that with the likelihood objective function (LikeObj). In terms of the test log likelihood, LikeObj was better than ErrObj. These results imply that the objective function should be selected properly depending on the applications. The proposed method with the marginal likelihood objective function (MarObjS and MarObjSQ) was worse than ErrObj and LikeObj. Although standard GPs are usually trained with the marginal likelihood of training data, it is different from the test mean squared error and test log likelihood. On the other hand, ErrObj and LikeObj directly minimize the evaluation measurements by simulating the test phase using the episodic training framework. This result demonstrates the effectiveness to use the test performance for the objective function for few-shot learning. The proposed method with the mean function without the support information (NoSptM) and that with the zero mean function (ZeroMean) performed worse than the proposed method. This result indicates the importance to use non-zero mean functions that incorporate the support information, and the advantage of the proposed method over existing GP-based meta-learning methods those that use zero mean functions (Harrison et al. 2018; Tossou et al. 2019) and those that do not use the support

Table 5 Average computational time in seconds for learning from the training datasets and the time for predicting test attribute values for each region

	Ours	NP	GPR	GPVAE	NN	FT
Train	2253.4	1031.9	637.6	1371.3	756.5	756.5
Test	0.142	0.050	0.005	0.080	0.026	0.161

	MAML	AVIATOR	MMAML	PN	R2D2
Train	7618.8	93387.1	100675.0	2161.7	1469.8
Test	0.067	0.139	0.160	0.233	0.109

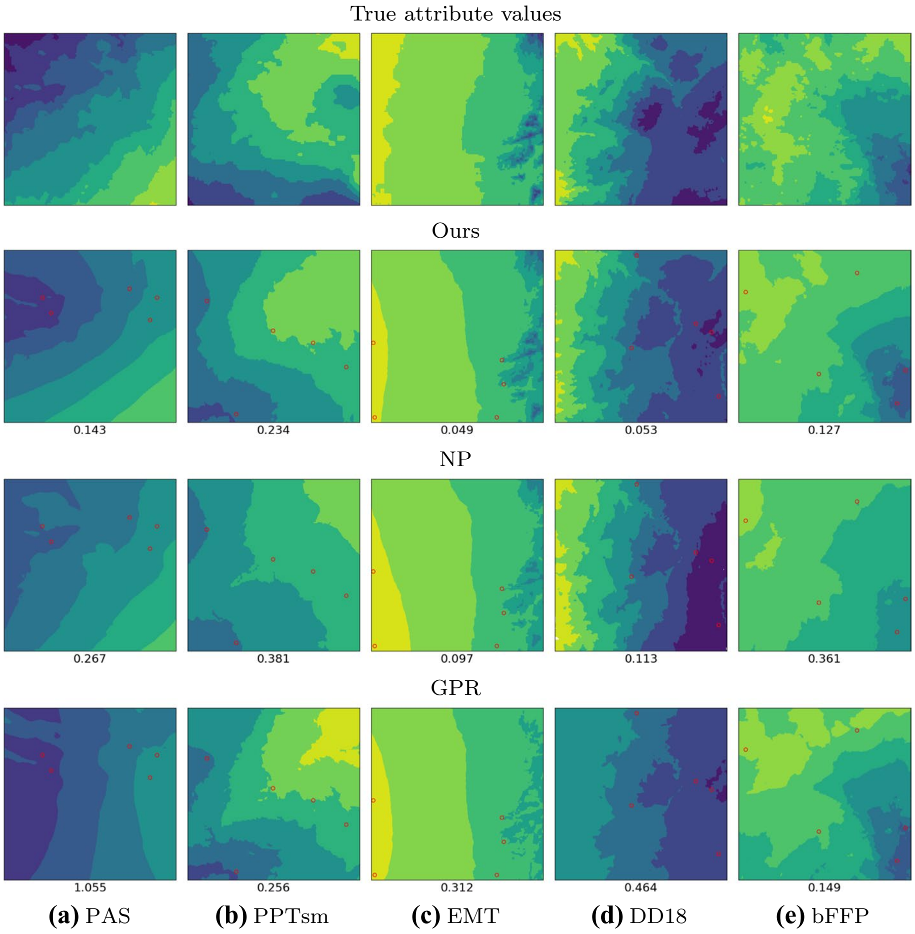

Fig. 4 Predictions for five attributes and regions of target tasks yielded by the proposed method, NP, and GPR. The top row shows the true attribute values. Red circles indicate observed locations. Values below each plot show the mean squared error

information (Harrison et al. 2018). Although the test mean squared error of the proposed method did not get worse with the kernel function without the support information

Table 6 Ablation study

	ErrObj	LikeObj	MarObjS	MarObjSQ	ZeroM	NoSptM	NoSptK	NoSptMK
MSE	**0.316**	0.329	0.476	0.497	0.394	0.386	**0.316**	0.470
LL	− 1.025	**− 0.987**	− 1.479	− 1.316	− 1.088	− 1.086	− 1.013	− 1.065

Test mean squared errors (MSE) and test log likelihoods (LL) on target tasks with NAE data. ErrObj is the proposed method with the mean squared error objective function in Eq. (9). LikeObj is that with the log likelihood objective function in Eq. (10), MarObjS is that with the marginal likelihood on the support set, MarObjSQ is that with the marginal likelihood on the support and query sets, and ZeroM is that with zero mean function. NoSptM is that with neural netwok-based mean function without the support information, $m(\mathbf{x}) = f_{\mathrm{m}}(\mathbf{x})$, and NoSptK is that with neural netwok-based kernel function without the support information, $k(\mathbf{x}, \mathbf{x}') = \exp\left(- \parallel f_{\mathrm{k}}(\mathbf{x}) - f_{\mathrm{k}}(\mathbf{x}') \parallel^2\right)$, and NoSptMK is that with neural netwok-based mean and kernel functions without the support information

(NoSptK), the test log likelihood got worse. This result implies that the kernel function with the support information is important for predicting the uncertainty. The performance of NoSptM was lower than that of NoSptK. This result indicates that incorporating the support information in the mean function is more beneficial for spatial regression.

5 Conclusion

We proposed a few-shot learning method for spatial regression. The proposed method can predict attribute values given a few observations, even if the target attribute and region are not included in the training datasets. The proposed method uses a neural network to infer a task representation from a few observed data. Then, it uses the inferred task representation to calculate the predicted values by a neural network-based Gaussian process framework. Experiments on climate spatial data showed that the proposed method achieved better prediction performance than existing methods. Although our results are encouraging, we must extend our approach in several directions. Although the proposed method uses a Bayesian framework given the mean and covariance functions based on GPs, the mean and covariance functions are trained by a point estimation. Therefore, when the number of tasks is small, there is the risk of meta-overfitting. We want to mitigate the risk of meta-overfitting using a Bayesian estimation of the mean and covariance functions (Rothfuss et al. 2020). In addition, we want to apply our framework to other types of tasks, such as spatio-temporal regression, regression for non-spatial data, and classification.

Declarations

Conflict of interest None.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., & De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 3981–3989.

Ashman, M., So, J., Tebbutt, W., Fortuin, V., Pearce, M., & Turner, R.E. (2020). Sparse Gaussian process variational autoencoders. In arXiv preprint arXiv:2010.10177.

Banerjee, S., Gelfand, A. E., Finley, A. O., & Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70*(4), 825–848.

Bartunov, S., & Vetrov, D. (2018). Few-shot generative modelling with generative matching networks. *International Conference on Artificial Intelligence and Statistics, 84,* 670–678.

Bengio, Y., Bengio, S., & Cloutier, J. (1991). Learning a synaptic learning rule. In *International Joint Conference on Neural Networks*.

Bertinetto, L., Henriques, J. F., Torr, P., & Vedaldi, A. (2018). Meta-learning with differentiable closed-form solvers. In *International conference on learning*.

Bogomolov, A., Lepri, B., Staiano, J., Oliver, N., Pianesi, F., & Pentland, A. (2014). Once upon a crime: towards crime prediction from demographics and mobile data. In *Proceedings of the 16th international conference on multimodal interaction*, pages 427–434. ACM.

Bonilla, E. V., Chai, K. M., & Williams, C. (2008). Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems,* 153–160.

Bornschein, J., Mnih, A., Zoran, D., & Rezende, D. J. (2017). Variational memory addressing in generative models. In *Advances in Neural Information Processing Systems*, 3920–3929.

Calandra, R., Peters, J., Rasmussen, C.E., & Deisenroth, M.P. (2016). Manifold Gaussian processes for regression. In *International Joint Conference on Neural Networks*, pp 3338–3345. IEEE.

Casale, F. P., Dalca, A. V., Saglietti, L., Listgarten, J., & Fusi, N. (2018). Gaussian process prior variational autoencoders. In *Neural Information Processing Systems*, 10390–10401.

Chen, Y., Friesen, A. L., Behbahani, F., Doucet, A., Budden, D., & Hoffman, M. W., de Freitas N. (2020). Modular meta-learning with shrinkage. In *Neural Information Processing Systems*.

Cressie, N. (1990). The origins of kriging. *Mathematical Geology, 22*(3), 239–252.

Edwards, H., & Storkey, A. (2016). Towards a Neural Statistician. arXiv preprint arXiv:1606.02185.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th international conference on machine learning*, pp 1126–1135.

Finn, C., Xu, K., & Levine, S. (2018). Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 9516–9527.

Fortuin, V., Strathmann, H., & Rätsch, G. (2019). Meta-learning mean functions for Gaussian processes. *arXiv preprint*arXiv:1901.08098.

Gao, X., Asami, Y., & Chung, C.-J.F. (2006). An empirical evaluation of spatial regression models. *Computers & Geosciences, 32*(8), 1040–1051.

Gao, J., Lu, Z., Tjøstheim, D., et al. (2006). Estimation in semiparametric spatial regression. *The Annals of Statistics, 34*(3), 1395–1435.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., & Eslami, S. A. (2018). Conditional neural processes. In *International conference on machine learning*, 1690–1699.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., & Teh, Y.W. (2018). Neural processes. *arXiv preprint*arXiv:1807.01622.

Grefenstette, E., Amos, B., Yarats, D., Htut, P. M., Molchanov, A., Meier, F., Kiela, D., Cho, K., & Chintala, S. (2019). Generalized inner loop meta-learning. *arXiv preprint*arXiv:1910.01727.

Gu, D., & Hu, H. (2012). Spatial Gaussian process regression with mobile sensor networks. *IEEE Transactions on Neural Networks and Learning Systems, 23*(8), 1279–1290.

Haining, R. (1993). *Spatial data analysis in the social and environmental sciences*. Cambridge University Press.

Harrison, J., Sharma, A., & Pavone, M. (2018). Meta-learning priors for efficient online Bayesian regression. *arXiv preprint*arXiv:1807.08912.

Hession, S. L., & Moore, N. (2011). A spatial regression analysis of the influence of topography on monthly rainfall in east africa. *International Journal of Climatology, 31*(10), 1440–1456.

Hewitt, L. B., Nye, M. I., Gane, A., Jaakkola, T., Tenenbaum, J. B. (2018). The variational homoencoder: learning to learn high capacity generative models from few examples. *arXiv preprint*arXiv:1807.08919.

Huang, W.-B., Zhao, D., Sun, F., Liu, H., & Chang, E. Y. (2015). Scalable Gaussian process regression using deep neural networks. In *IJCAI,* pp 3576–3582.

Iwata, T., Ghahramani, Z. (2017). Improving output uncertainty estimation and generalization in deep learning via neural network Gaussian processes. *arXiv preprint*arXiv:1707.05922.

Iwata, T., Otsuka, T. (2019). Efficient transfer Bayesian optimization with auxiliary information. *arXiv preprint*arXiv:1909.07670.

Jean, N., Xie, S. M., & Ermon, S. (2018). Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In *Advances in Neural Information Processing Systems*, pp 5322–5333.

Jerrett, M., Burnett, R. T., Ma, R., Pope, C. A., III., Krewski, D., Newbold, K. B., et al. (2005). Spatial analysis of air pollution and mortality in los angeles. *Epidemiology, 1,* 727–736.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., Teh, Y. W. (2019). Attentive neural processes. In *International conference on learning representations*.

Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., & Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International conference on learning representations*.

Lee, K., Maji, S., Ravichandran, A., & Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp10657–10665.

Li, D., Zhang, J., Yang, Y., Liu, C., Song, Y.-Z., & Hospedales, T. M. (2019). Episodic training for domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp 1446–1455.

Li, Z., Zhou, F., Chen, F., Li, H. (2017). Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint*arXiv:1707.09835.

Luttinen, J., & Ilin, A. (2009). Variational Gaussian-process factor analysis for modeling spatio-temporal data. *Advances in Neural Information Processing Systems, 22,* 1177–1185.

Micchelli, C. A., Xu, Y., & Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research, 7*(Dec), 2651–2667.

Myers, D. E. (1982). Matrix formulation of co-kriging. *Journal of the International Association for Mathematical Geology, 14*(3), 249–257.

Park, C., Huang, J. Z., & Ding, Y. (2011). Domain decomposition approach for fast Gaussian process regression of large spatial data sets. *Journal of Machine Learning Research, 12*(May), 1697–1728.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian processes for machine learning*. MIT press.

Ravi, S., & Larochelle, H. (2017). Optimization as a model for few-shot learning. In *International conference on learning representations*.

Reed, S., Chen, Y., Paine, T., Oord, A. v. d., Eslami, S., Rezende, D., Vinyals, O., de Freitas, N. (2017). Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint*arXiv:1710.10304.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., & Wierstra, D. (2016). One-shot generalization in deep generative models. In *Proceedings of the 33rd international conference on international conference on machine learning*, pp 1521–1529.

Rothfuss, J., Fortuin, V., Josifoski, M., Krause, A. (2020). PACOH: Bayes-optimal meta-learning with PAC-guarantees. *arXiv preprint*arXiv:2002.05551.

Rupasingha, A., & Goetz, S. J. (2007). Social and political forces as determinants of poverty: A spatial analysis. *The Journal of Socio-Economics, 36*(4), 650–671.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., & Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *International conference on learning representations*.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp 1842–1850.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Master's thesis, Technische Universitat Munchen.

Shadbolt, N., O'Hara, K., Berners-Lee, T., Gibbins, N., Glaser, H., Hall, W., et al. (2012). Linked open government data: Lessons from data. *IEEE Intelligent Systems, 27*(3), 16–24.

Smith-Clarke, C., Mashhadi, A., & Capra, L. (2014). Poverty on the cheap: Estimating poverty maps using aggregated mobile communication networks. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp 511–520.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems, 1,* 4077–4087.

Stein, M. L. (2012). *Interpolation of spatial data: Some theory for kriging*. Springer.

Stein, A., & Corsten, L. (1991). Universal kriging and cokriging as a regression procedure. *Biometrics, 1,* 575–587.

Stralberg, D., Matsuoka, S., Hamann, A., Bayne, E., Sólymos, P., Schmiegelow, F., et al. (2015). Projecting boreal bird responses to climate change: The signal exceeds the noise. *Ecological Applications, 25*(1), 52–69.

Tossou, P., Dura, B., Laviolette, F., Marchand, M., Lacoste, A. (2019). Adaptive deep kernel learning. *arXiv preprint*arXiv:1905.12131.

Venkitaraman, A., Wahlberg, B. (2020). Task-similarity aware meta-learning through nonparametric kernel regression. *arXiv preprint*arXiv:2006.07212.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems, 29,* 3630–3638.

Vuorio, R., Sun, S.-H., Hu, H., & Lim,J.J. (2019). Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*.

Wang, H., Kifer, D., Graif, C., & Li, Z. (2016). Crime rate inference with big data. *Proceedings of the ACM SIGKDD 22nd international conference on knowledge discovery and data mining*, pp 635–644.

Wang, T., Hamann, A., Spittlehouse, D., & Carroll, C. (2016). Locally downscaled and spatially customizable climate data for historical and future periods for North America. *PloS One, 11*(6), E0156720.

Ward, M. D., & Gleditsch, K. S. (2018). *Spatial regression models*. Sage Publications.

Wei, P., Sagarna, R., Ke, Y., Ong, Y.-S., & Goh, C.-K. (2017). Source-target similarity modelings for multi-source transfer Gaussian process regression. In *Proceedings of the 34 th international conference on machine learning*, pp 3722–3731.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., & Xing, E. P. (2016). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, 2586–2594.

Wilson, A.G., Knowles, D.A., & Ghahramani, Z. (2011). Gaussian process regression networks. *arXiv preprint*arXiv:1110.4411.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep kernel learning. *Artificial Intelligence and Statistics, 51,* 370–378.

Yao, H., Wei, Y., Huang, J., & Li, Z. (2019). Hierarchically structured meta-learning. In *In International conference on machine learning*, 7045–7054.

Ye, H.-J., Sheng, X.-R., & Zhan, D.-C. (2020). Few-shot learning with adaptively initialized task optimizer: A practical meta-learning approach. *Machine Learning, 109*(3), 643–664.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on machine learning*, pp 1012–1019.

Yuan, J., Zheng, Y., & Xie, X. (2012). Discovering regions of different functions in a city using human mobility and POIs. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, pp 186–194. ACM.

Yuan, J., Zheng, Y., Xie, X., & Sun, G. (2011). T-drive: Enhancing driving directions with taxi drivers intelligence. *IEEE Transactions on Knowledge and Data Engineering, 25*(1), 220–232.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep sets. In *Advances in Neural Information Processing Systems*, 3391–3401.

Zheng, Y., Capra, L., Wolfson, O., & Yang, H. (2014). Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST), 5*(3), 38.