Pattern Recognition Letters 161 (2022) 130–136

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Attention Mechanism Based Mixture of Gaussian Processes

Tao Li, Jinwen Ma*

Department of Information Science, School of Mathematical Sciences and LMAM, Peking University, Beijing, 100871, China

ARTICLE INFO

Article history:
Received 17 December 2021
Revised 23 March 2022
Accepted 8 August 2022
Available online 10 August 2022

Edited by: Jiwen Lu

Keywords:
Gaussian processes
Attention
Mixture model

ABSTRACT

The mixture of Gaussian processes (MGP) is a powerful model, which is able to characterize data generated by a general stochastic process. However, conventional MGPs assume the input variable obeys certain probabilistic distribution, thus cannot effectively handle the case where the input variable lies on a general manifold or a graph. In this paper, we first clarify the relationship between the MGP prediction strategy and the attention mechanism. Based on the attention mechanism, we further design two novel mixture models of Gaussian processes, which do not rely on probabilistic assumptions on the input domain, thus overcoming the difficulty of extending MGP models to manifold or graph. Experimental results on real-world datasets demonstrate the effectiveness of the proposed methods.

1. Introduction

Gaussian process (GP) [1] is the dominant non-parametric Bayesian model to learn and infer over temporal data or uncertain functions, which has been widely used in many fields. In the machine learning community, a trained Gaussian process with zero mean function and commonly used covariance function is always stationary, thus a single Gaussian process cannot effectively model multimodal data generated by a non-stationary source. To tackle this problem, the mixture of Gaussian processes (MGP) [2,3] has been proposed to enhance the model flexibility. Specifically, the generative mixture of Gaussian processes has been thoroughly studied in recent years [4–7]. We refer to these models as the conventional mixture of Gaussian processes. Usually, a conventional MGP assumes the input variable obeys certain probabilistic distribution over the input domain. However, this assumption is only valid when the input variable lies in a Euclidean space because most probabilistic distributions are defined on a Euclidean space (or its subset). Although some probabilistic distributions on specific manifolds have been investigated (such as the von-Mises distribution on a circle), they are fairly complicated and highly depend on the concrete definition of the underlying manifold. When the input variable lies on a general manifold or a graph, it would be difficult to assume that the input variable obeys certain probabilistic distribution.

Attention mechanism [8] is one of the most popular research topics in machine learning and pattern recognition, which attempts to implement the action of selectively concentrating on a few relevant things while ignoring others. As a flexible module, it has been widely used in natural language processing [9], computer vision [10], graph neural networks [11] and so on. The basic idea of the attention mechanism is to concentrate on related information based on similarity, which can be understood as a "soft" hash table. Most recent research works embed the attention module into a large neural network structure, or develop novel attention modules based on traditional statistical learning methods.

In this paper, we first propose to rethink MGP from the perspective of attention mechanism, and we find that the prediction strategy of MGP can be regarded as an attention mechanism. This observation inspires us to extend MGP to a general (Riemannian) manifold and graph as long as there is a well-defined similarity or distance function. We propose two novel mixture of Gaussian processes models as show in Fig. 1. The first one is the local mixture of Gaussian processes (LMGP), which trains many Gaussian processes locally and weight their predictions via the attention mechanism. The second one is a clustering based mixture of Gaussian processes, which divides training samples into groups by clustering method, then training a Gaussian process model within each group and combining their predictions via attention mechanism.

2. Related Works

The connection between traditional statistical learning methods and the attention mechanism has been an active research area in recent years. Gaussian distributions and Gaussian mixture models have been utilized to design novel attention modules for specific tasks, such as neural machine translation [12,13], speech enhancement [14], speech recognition [15], image clustering [16], scene text recognition [17] and computer vision [18]. In these applica-

* Corresponding author
E-mail addresses: li_tao@pku.edu.cn (T. Li), jwma@math.pku.edu.cn (J. Ma).

https://doi.org/10.1016/j.patrec.2022.08.003
0167-8655/

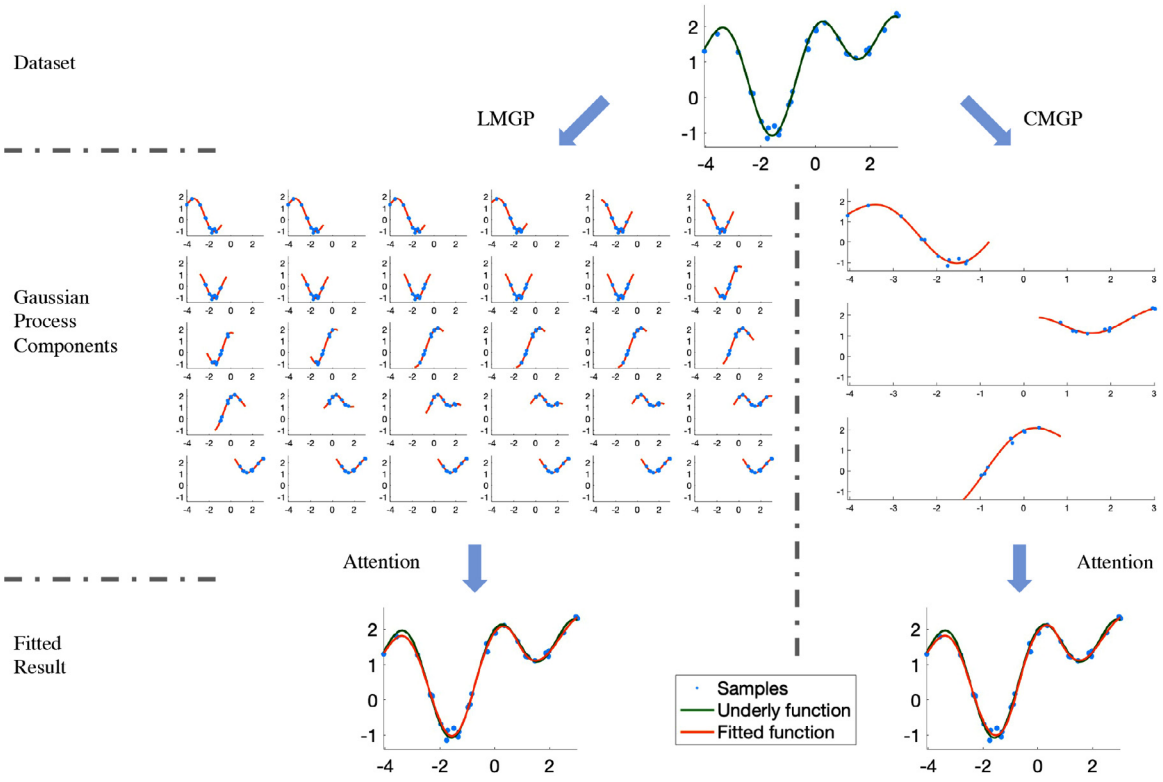


Fig. 1. The illustration of LMGP and CMGP.

tions, domain knowledge can be effectively described by Gaussian distributions and Gaussian mixture models. Gaussian kernels also play an important role in remodeling self-attention [19]. Gaussian process channel attention [20] models the correlations among the channels via GPs and further interprets the channel attention schemes in a probabilistic way. Most of these related works emphasize on *Gaussian distributions* or *Gaussian mixture models*, while we emphasize on *mixture of Gaussian processes*. Besides, the main aim of these related works is to develop more powerful attention modules or interpret the attention mechanism based on traditional statistical learning methods, while we aim to understand MGPs from the perspective of the attention mechanism and design novel MGP models.

3. Preliminaries

3.1. Gaussian Processes and Mixture of Gaussian Processes

In machine learning, a Gaussian process is defined by a mean function $\mu(\cdot)$ and a covariance function $c(\cdot,\cdot)$. In practice, $\mu(\cdot)$ is usually assumed to be 0, and $c(\cdot,\cdot;\boldsymbol{\theta})$ is parameterized by $\boldsymbol{\theta}$. Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, Gaussian process regression assumes that $y = f(\mathbf{x})$, where f is a GP. The parameter $\boldsymbol{\theta}$ is learned by maximizing the marginal likelihood $\log p(\{y_i\}_{i=1}^N|\{\mathbf{x}_i\}_{i=1}^N;\boldsymbol{\theta})$. When a new input $\mathbf{x}_*$ is given, the corresponding response y_* is predicted according to the conditional property of Gaussian distributions. Theoretically, the learning procedure requires $\mathcal{O}(N^3)$ complexity and the prediction procedure requires $\mathcal{O}(N^2)$ complexity, which is prohibitively large for big data.

In MGP, the data are assumed to be generated from K independent GP components. First, latent indicators $\{z_i\}_{i=1}^N$ are generated according to $p(z_i = k) = \pi_k$. For those samples with $z_i = k$, the driven variables $\{\mathbf{x}_i|z_i = k, i = 1,\cdots,N\}$ are generated by a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, then the responses $\{y_i|z_i = k, i = 1,\cdots,N\}$ are generated by a Gaussian process with parameters $\boldsymbol{\theta}_k$. The learning procedure of MGP is challenging, and state-of-the-art methods are based on the EM algorithm. Suppose the EM algorithm iterates for T steps, the time complexity of learning an MGP model using the hard-cut EM algorithm [4] is approximate $\mathcal{O}(N^3T/K^2)$.

The prediction strategy of MGP is more complicated than a single Gaussian process. Given an input $\mathbf{x}_*$, we have

$$p(y_*|\mathbf{x}_*, \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \{\boldsymbol{\theta}_l\}_{l=1}^K) = \sum_{k=1}^K p(z_* = k|\mathbf{x}_*, \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \{\boldsymbol{\theta}_l\}_{l=1}^K) p(y_*|\mathbf{x}_*, z_* = k, \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \boldsymbol{\theta}_k). \quad (1)$$

Therefore, we need to predict the response in each GP component separately, and weight the predictions with the posterior probability that $\mathbf{x}_*$ belongs to the corresponding component. The time complexity of prediction is approximate $\mathcal{O}(N^2/K)$.

3.2. Attention Mechanism

Consider a dictionary or a hash table $\{k_1 : v_1, \cdots, k_K : v_K\}$ where $\{k_i\}_{i=1}^K$ are keys while $\{v_i\}_{i=1}^K$ are corresponding values. Given a query q, we first determine whether q appears in the keys. If $q \in \{k_1, \cdots, k_K\}$, say $q = k_i$, then return the corresponding value v_i. Otherwise, the program raises an error. The attention mechanism can be considered as a "soft" hash table. When a query q is given, we first calculate the similarity $\mathrm{sim}(q, k_i)$ between q and each k_i, then return the weighted average of values:

$$\sum_{i=1}^K \omega_i v_i \quad \text{where} \quad \omega_i = \frac{\exp(\mathrm{sim}(q, k_i))}{\sum_{j=1}^K \exp(\mathrm{sim}(q, k_j))}. \quad (2)$$

Here, $\{\omega_i\}_{i=1}^K$ are attention weights. The intuition behind this equation is that if q and k_i are similar, then ω_i is relatively large, and the importance of the corresponding value v_i is high.

The key of attention mechanism is the similarity function, which directly determines attention weights. In deep learning, the

similarity between q and k_i is usually defined as the cosine similarity after projecting them to a low-dimensional space by learnable linear transformations. If a distance function $d(\cdot,\cdot)$ is defined, we can also naturally define the similarity function by composing a decreasing function h with d, *i.e.*, $\mathrm{sim}(q,k_i)=h(d(q,k_i))$. Specifically, if $\mathrm{sim}(q,k_i)=\beta\mathbb{I}(q=k_i)$, then the attention mechanism degenerates to a dictionary as $\beta\to\infty$.

The attention mechanism is similar to gating networks. However, gating networks are usually learnable functions of inputs, and gating network based mixture of experts are learned by the EM algorithm. The attention mechanism emphasizes more on the similarity between samples, and the pre-defined or learnable similarity function encodes more prior knowledge.

4. Proposed Methods

4.1. Rethinking Mixture of Gaussian Processes From the Attention Perspective

We first point out that the prediction strategy of MGP can be regarded as an attention mechanism. Suppose the response predicted by the k-th Gaussian process component is $y_*^{(k)}$, and the posterior probability that $z_*=k$ is ω_k, then Eq. (1) can be rewritten as $\sum_{k=1}^K \omega_k y_*^{(k)}$, which has exactly the same form as Eq. (2). In fact, in the prediction phase of MGP, we can consider a dictionary $\{\text{Component } 1: y_*^{(1)},\cdots,\text{Component } K: y_*^{(K)}\}$, the final prediction equals to the result of querying $\mathbf{x}_*$ in this dictionary with the attention mechanism. Here, the weight ω_k is given by

$$\omega_k=\frac{\pi_k|\mathbf{\Sigma}_k|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_k)^\intercal\mathbf{\Sigma}_k^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_k)\right)}{\sum_{l=1}^K\pi_l|\mathbf{\Sigma}_l|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_l)^\intercal\mathbf{\Sigma}_l^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_l)\right)}. \tag{3}$$

From the attention perspective, Eq. (3) is equivalent to define the similarity function as

$$\begin{aligned}&\mathrm{sim}(\mathbf{x}_*,\text{Component } k)\\&=\log\pi_k-\frac{1}{2}(\mathbf{x}_*-\boldsymbol{\mu}_k)^\intercal\mathbf{\Sigma}_k^{-1}(\mathbf{x}_*-\boldsymbol{\mu}_k)-\frac{1}{2}\log|\mathbf{\Sigma}_k|.\end{aligned} \tag{4}$$

The key-point here is that Eq. (4) is totally determined by the probabilistic assumption on $z\to\mathbf{x}$. Since we assume that in the k-th component inputs are subject to $\mathcal{N}(\boldsymbol{\mu}_k,\mathbf{\Sigma}_k)$, the similarity between $\mathbf{x}_*$ and the k-th component is defined as the log probability that $\mathbf{x}_*$ comes from $\mathcal{N}(\boldsymbol{\mu}_k,\mathbf{\Sigma}_k)$. There are some research works [21–24] that modify the probabilistic assumption on $z\to\mathbf{x}$, which boils down to change the definition of similarity function between $\mathbf{x}_*$ and components in the attention viewpoint.

Furthermore, the attention perspective enables us to abandon probabilistic assumptions on $z\to\mathbf{x}$, as long as the similarity between an input and a Gaussian process component is properly defined. Although imposing probabilistic assumptions on $z\to\mathbf{x}$ makes the data generating process clear, it also has several drawbacks. First, learning parameters is challenging since the samples are correlated and there are exponentially many summations in the Q-function as indicated in [4,7]. Second, assuming $z\to\mathbf{x}$ follows certain distribution [4,22–24] is only valid for Euclidean data. If $\mathbf{x}$ lies on a manifold, we have to use distributions on manifolds, which is fairly complicated. On the other hand, defining the similarity function between an input and a GP component is straightforward and flexible. In the next subsections, we introduce two novel mixtures of Gaussian processes based on the attention mechanism.

4.2. Local Mixture of Gaussian Processes (LMGP)

Conventional mixtures of Gaussian processes build the model in a top-down way: samples are divided into groups and in each group, the samples are generated by a Gaussian process. The local mixture of Gaussian processes builds the model in a bottom-up way, which is similar to locally weighted regression [25]. For each sample $(\mathbf{x}_i,y_i)$, we find its $(K-1)$-nearest neighbors and train the i-th GP on these K data points (including $(\mathbf{x}_i,y_i)$). There are N such local GPs in total. In the prediction phase, when a new input $\mathbf{x}_*$ is given, we naturally define the similarity between $\mathbf{x}_*$ and the i-th GP as $\mathrm{sim}(\mathbf{x}_*,\text{Component } i)=-d^2(\mathbf{x}_*,\mathbf{x}_i)$ where $d(\cdot,\cdot)$ is a predefined distance function. Suppose that the prediction of i-th GP is $y_*^{(i)}$, then the final prediction is

$$\sum_{i=1}^N\omega_i y_*^{(i)}\quad\text{where}\quad\omega_i=\frac{\exp(\mathrm{sim}(\mathbf{x}_*,\text{Component } i))}{\sum_{j=1}^N\exp(\mathrm{sim}(\mathbf{x}_*,\text{Component } i))}. \tag{5}$$

Note that this model is applicable to non-Euclidean data (*i.e.*, $\mathbf{x}$ lies on a manifold or a graph) as long as the distance function $d(\cdot,\cdot)$ is well-defined, and we do not need to specify a probabilistic distribution on the input domain.

The choice of K is important in practice. It is well-known that Gaussian processes quickly converge to the mean function in extrapolation. Suppose the inputs of the dataset used for training i-th GP is $\mathcal{X}_i$ and $\mathrm{conv}(\mathcal{X}_i)$ denotes the convex hull [26] of $\mathcal{X}_i$, it is possible that $\mathbf{x}_*\notin\mathrm{conv}(\mathcal{X}_i)$ for any i if K is too small. See Fig. 2 for example. As a criterion, we suggest to set K such that $\cup_{i=1}^N\mathrm{conv}(\mathcal{X}_i)=\mathrm{conv}\left(\cup_{i=1}^N\mathcal{X}_i\right)$.

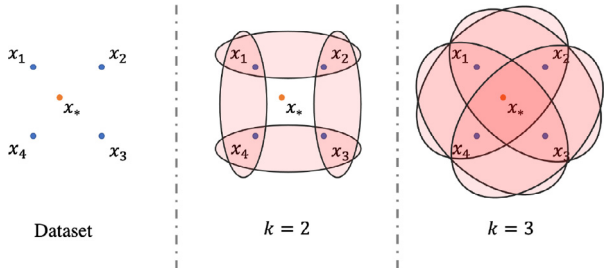


Fig. 2. The choice of K in LMGP influences the covered region.

Suppose the time of calculating $d(\cdot,\cdot)$ for two inputs takes $\mathcal{O}(P)$. For each sample, finding its $(K-1)$-nearest neighbor requires $\mathcal{O}(N(P+K))$[1], and training the GP takes $\mathcal{O}(K^3)$. Therefore, the time complexity of learning an LMGP model is $\mathcal{O}(N^2(P+K)+NK^3)$, and the prediction complexity is $\mathcal{O}(NK^2+NP)$. Theoretically, both learning and prediction of LMGP are faster than GP and MGP. In practice, however, LMGP may be rather slow since iterating over N samples training N Gaussian processes sequentially is time-consuming due to implementation reasons, and the cost can be further reduced by parallelization.

If $\mathbf{x}_*$ is very far from $\mathbf{x}_i$, then the i-th GP is not important for predicting y_*. Approximately, we can first find L-nearest neighbors of $\mathbf{x}_*$ and replace the summation in Eq. (5) by these L-nearest neighbors. In this way, the prediction complexity is reduced to $\mathcal{O}(LK^2+N(P+L))$.

4.3. Clustering Mixture of Gaussian Processes (CMGP)

One drawback of LMGP is that we need to train far more GPs than conventional MGP. Although the training complexity of LMGP is theoretically less than conventional MGP, in practice it may take a longer time due to underlying implementation issues such as vectorization acceleration and caching. Besides, these GP components are heavily overlapped and thus redundant. Consider an extreme example, if $K=N$ in LMGP, then we need to train GPs on the same data points for N times.

[1] See https://stats.stackexchange.com/questions/219655/k-nn-computational-complexity for a detailed discussion.

The clustering mixture of Gaussian processes is more similar to conventional MGP. The number of GP components K is pre-defined. We first cluster the samples into K groups using the k-medoids algorithm, and the cluster label of i-th sample is denoted by z_i. Then in each cluster, we train a Gaussian process. Given a new input $\mathbf{x}_*$, we define the similarity between $\mathbf{x}_*$ and the k-th GP as $\text{sim}(\mathbf{x}_*, \text{Component } k) = -\min_{z_i=k} d^2(\mathbf{x}_*, \mathbf{x}_i)$ where $d(\cdot,\cdot)$ is a pre-defined distance function. The final prediction is given by the same formula Eq. (5) as LMGP (with N replaced by K).

Compared with conventional MGP models, one distinguishing feature of CMGP is we do not need to perform EM iterations until convergence. The learning procedure is straightforward and easy: first cluster the samples, then train GPs in each cluster separately. Besides, CMGP is able to process non-Euclidean data provided that we can perform the clustering algorithm on the inputs. For the k-medoids clustering algorithm, this requirement is equivalent to we have a pre-defined distance function on the input domain. A typical implementation of k-medoids clustering takes $\mathcal{O}(KN^2pT)$ [27,28], where T is the number of iterations and calculating $d(\cdot,\cdot)$ is assumed to take $\mathcal{O}(p)$. Training K Gaussian processes takes approximately $\mathcal{O}(N^3/K^2)$, thus the total time complexity of training a CMGP model is $\mathcal{O}(KN^2pT + N^3/K^2)$. As for prediction, calculating attention weights takes $\mathcal{O}(Np)$, and obtaining predictions of K Gaussian process components needs $\mathcal{O}(N^2/K)$, so the total time complexity of prediction is $\mathcal{O}(Np + N^2/K)$. Theoretically, the time complexity of CMGP has no significant difference compared with conventional MGP models. However, CMGP is usually faster in practice since it avoids tedious EM iterations.

4.4. Discussions and Remarks

In the training process of LMGP and CMGP, GPs are independently trained and they are not mixed together. However, in the prediction phase, separate predictions of GP components are mixed, thus we refer to them as mixture models.

The proposed frameworks can be generalized in many aspects. For example, it is valid to consider other regression functions instead of GPs in each component. We choose to develop the theory with GPs because our initial motivation is to extend conventional MGP to the case that input variables lie on a general manifold/graph. Compared with other regression techniques, covariance functions of GPs can effectively utilize the structural information of the input domain (see Eq. (6)), which is very suitable for the case we considered. In addition, since GPs can be used for classification, LMGP and CMGP can also be applied for classification too.

5. Experimental Results

5.1. Datasets and Experimental Settings

We use the following datasets to evaluate the prediction performances of the proposed methods:

- *Berkeley Earth*[2] *[29]* records monthly averaged temperature anomalies (unit: centigrade) from 1850 to recent at $1^\circ \times 1^\circ$ latitude-longitude grid. In this experiment, we use the records of the first half-year of 2020, and we refer to these sub-datasets as 2020/1 to 2020/6, respectively.
- *Accessibility*[3] *[30] describes travel times (unit: hour) to major cities (cities of 50,000 or more people in year 2000) with a resolution of 30 arc seconds.*
- *Vegetation*[4] *records percent tree coverage rate at 30 arc seconds.*

[2] http://berkeleyearth.org/data/.
[3] https://forobs.jrc.ec.europa.eu/products/gam/index.php.
[4] https://globalmaps.github.io/ptc.html. Source: Geospatial Information Authority of Japan, Chiba University and collaborating organizations.

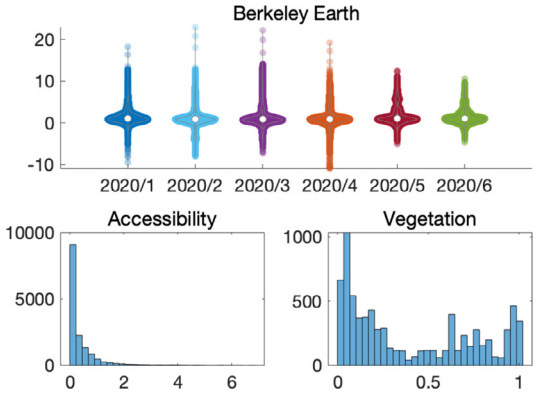


Fig. 3. Detailed distributions of these datasets.

For Berkeley Earth datasets, we use the records at $10^\circ \times 10^\circ$ grid as the training set, and for the Accessibility dataset and the Vegetation dataset we use the records at $5^\circ \times 5^\circ$ grid as the training set. The aim is to predict outputs at other $1^\circ \times 1^\circ$ grids. This setting is similar to statistical downscaling simulation in the geographic information system community. Invalid values are excluded. For example, travel times and percent tree coverage rates are only meaningful on dry land and islands. In summary, there are 648 training samples and 64152 testing samples in Berkeley Earth datasets, 599 training samples and 14480 testing samples in the Accessibility dataset, 301 training samples and 7388 testing samples in the Vegetation dataset. Detailed distributions of these datasets are shown in Fig. 3.

Besides LMGP and CMGP, we consider the following competing methods in this experiment:

(a) AVE: averaging the responses of all training samples.
(b) 1-NN: 1-nearest neighbor regression.
(c) SVR: support vector regression with Gaussian kernel.
(d) NN: feedforward neural network with three hidden layers containing 10, 10, 5 units, respectively.
(e) GP: a single Gaussian process.
(f) MGP: a mixture of Gaussian processes.

For MGP and CMGP, we set $K = 2$. For LMGP, we set $K = 128$ as discussed in Section 4.2. We postpone the sensitivity analysis of parameters until Section 5.3.

We use the squared exponential covariance function [1] in Gaussian processes. Specifically, given inputs $\mathbf{x}_1$ and $\mathbf{x}_2$, the squared exponential covariance function calculates the covariance between these two points by

$$c(\mathbf{x}_1, \mathbf{x}_2; \boldsymbol{\theta}) = \theta_1^2 \exp(-\theta_2^2 d_{\mathcal{X}}(\mathbf{x}_1, \mathbf{x}_2)) + \theta_3^2 \mathbb{I}(\mathbf{x}_1 = \mathbf{x}_2)\,. \tag{6}$$

The equation above requires a pre-defined distance function $d_{\mathcal{X}}$. As stated in Section 4.3 and Section 4.2, attention mechanism based MGPs also require us to define the similarity function, or equivalently, the distance function. For geographic data, the spherical distance (great-circle distance) is a natural choice for $d_{\mathcal{X}}$. For comparison, we also consider $d_{\mathcal{X}}$ to be the Euclidean distance for comparison.

5.2. Performance Evaluation

We report the results of these methods in Table 1. The performances are evaluated by Rooted Mean Square Errors (RMSE) and Mean Absolute Errors (MAE). We also record the running times (including training and testing). For NN, MGP, CMGP, the results are affected by the randomness of initialization, so we run each method for 10 times and report the averaged results.

Table 1
Rooted Mean Squared Errors (RMSEs), Mean Absolute Errors (MAEs) and running times (in seconds) of proposed methods and competing methods on the Berkeley Earth 2020/1-2020/6 datasets, the Accessibility dataset and the Vegetation dataset.

Method	$d_{\mathcal{X}}$	Berkeley Earth 2020/1			Berkeley Earth 2020/2			Berkeley Earth 2020/3			Berkeley Earth 2020/4		
		RMSE	MAE	time	RMSE	MAE	time	RMSE	MAE	time	RMSE	MAE	time
AVE	-	1.3475	0.9285	0.07	2.0667	1.1952	0.00	2.3122	1.3928	0.00	1.8123	1.1310	0.00
1-NN	-	0.7771	0.4440	15.36	0.9753	0.5169	13.58	0.9601	0.5174	13.75	0.9340	0.4953	13.80
SVR	-	1.2421	0.8101	0.14	1.7934	1.0305	0.03	2.0354	1.1414	0.03	1.4696	0.9204	0.03
NN	-	0.9990	0.6764	0.96	1.3734	0.8583	0.51	1.3822	0.8897	0.63	1.1341	0.7314	0.69
GP	Euclidean	0.6785	0.4103	21.65	0.8327	0.4903	18.68	0.8168	0.4772	19.08	0.8077	0.4823	19.35
	Sphere	0.5906	0.3219	33.97	0.7350	0.3828	30.23	0.7680	0.4190	28.67	0.7483	0.4099	26.77
MGP	Euclidean	0.6865	0.4184	11.98	0.8440	0.4934	8.77	0.8846	0.4888	28.94	0.7963	0.4361	26.05
	Sphere	0.6036	0.3262	72.68	0.7669	0.4051	56.57	0.7801	0.4064	74.00	0.7506	0.4026	84.95
CMGP	Euclidean	0.6422	0.3774	11.18	0.8377	0.4931	4.80	0.7951	0.4500	6.81	0.7635	0.4352	7.24
	Sphere	**0.5803**	**0.3082**	40.31	0.7322	0.3795	37.46	**0.7408**	**0.3767**	38.21	**0.7183**	**0.3698**	39.33
LMGP	Euclidean	0.6333	0.3590	499.63	0.7772	0.4378	488.67	0.7808	0.4371	521.08	0.7723	0.4320	487.80
	Sphere	0.5883	0.3169	3424.55	**0.7321**	**0.3790**	3298.64	0.7409	**0.3767**	3242.84	0.7232	0.3756	3348.82

Method	$d_{\mathcal{X}}$	Berkeley Earth 2020/5			Berkeley Earth 2020/6			Accessibility			Vegetation		
		RMSE	MAE	time	RMSE	MAE	time	RMSE	MAE	time	RMSE	MAE	time
AVE	-	2.0251	1.5515	0.00	1.4950	1.0863	0.00	0.6595	0.4000	0.00	0.3463	0.3148	0.00
1-NN	-	0.9003	0.4921	13.84	0.6673	0.3935	13.64	0.2060	0.1339	3.38	0.3810	0.2592	1.21
SVR	-	1.4319	0.9319	10.56	1.1247	0.7468	0.02	0.4168	0.1847	0.09	0.3182	0.2510	0.04
NN	-	1.0127	0.6627	0.58	0.7148	0.4910	0.70	0.2579	0.1636	1.42	0.3144	0.2622	0.70
GP	Euclidean	0.8163	0.5584	13.70	0.5591	0.3825	18.22	0.2176	0.1341	13.83	0.3021	0.2536	1.30
	Sphere	0.6325	0.3427	22.57	0.5077	0.3003	22.34	0.2016	0.1200	16.62	0.2948	0.2487	3.20
MGP	Euclidean	0.7154	0.4486	8.84	0.5157	0.3149	6.09	0.2373	0.1379	14.47	0.3183	0.2751	3.18
	Sphere	0.6360	0.3474	73.02	0.4686	0.2765	68.33	0.2260	0.1288	22.17	0.2974	0.2505	23.74
CMGP	Euclidean	0.6988	0.4356	6.95	0.5386	0.3616	6.92	0.2039	0.1292	7.05	0.2952	0.2515	2.06
	Sphere	**0.6243**	0.3387	38.32	**0.4568**	0.2651	39.60	0.1928	0.1203	18.97	**0.2937**	**0.2446**	11.71
LMGP	Euclidean	0.6694	0.4054	525.27	0.5037	0.3065	500.11	0.1969	0.1262	708.07	0.3075	0.2649	124.59
	Sphere	0.6255	**0.3357**	3549.57	0.4569	**0.2631**	3312.25	**0.1917**	**0.1194**	1434.83	0.2950	0.2487	807.48

We have the following observations. First, the distance function significantly influences the performance. The squared exponential covariance function with Euclidean distance wrongly describes the relationships between points. On the other hand, using spherical distance in the square exponential covariance function consistently helps to improve the performance, which demonstrates the importance of choosing a proper distance function in Gaussian processes. Second, introducing a mixture structure does not necessarily improve the performance. On most datasets, we observe that MGP obtains comparable results as GP. However, we cannot conclude that mixture structure is useless since CMGP and LMGP outperform GP in most cases. Therefore, choosing a proper distance function is important for mixture models, and the main reason that conventional MGP fails to gain performance improvement is that it ignores the non-Euclidean structure of the input domain. Third, we find that the proposed methods consistently achieve the best performance both in terms of RMSE and MAE, which demonstrates the effectiveness of the proposed methods. The running times of LMGP are very long. The theoretical complexity of LMGP grows quadratically with respect to the number of training points, it may be very slow in practice due to implementation reasons. CMGP is usually faster than MGP because there is no iteration process in CMGP. Although the structure of CMGP looks similar to MGP, CMGP often outperforms MGP even using the Euclidean distance function. The reason is that CMGP clusters the points merely based on the input, while MGP determines the class labels in consideration of both input and output. In practice, we find the boundary of clusters found by MGP has an irregular shape, while CMGP tends to divide all input points according to the northern and southern hemispheres.

We illustrate the prediction results of GP, MGP, LMGP and CMGP on 2020/6 in Fig. 4. From Table 1, we can see that LMGP and CMGP achieve better performances than GP and MGP. Fig. 4 visualizes the differences among their prediction results and highlights some regions where the difference is significant. We observe that LMGP and CMGP can obtain finer prediction results compared with GP and MGP.

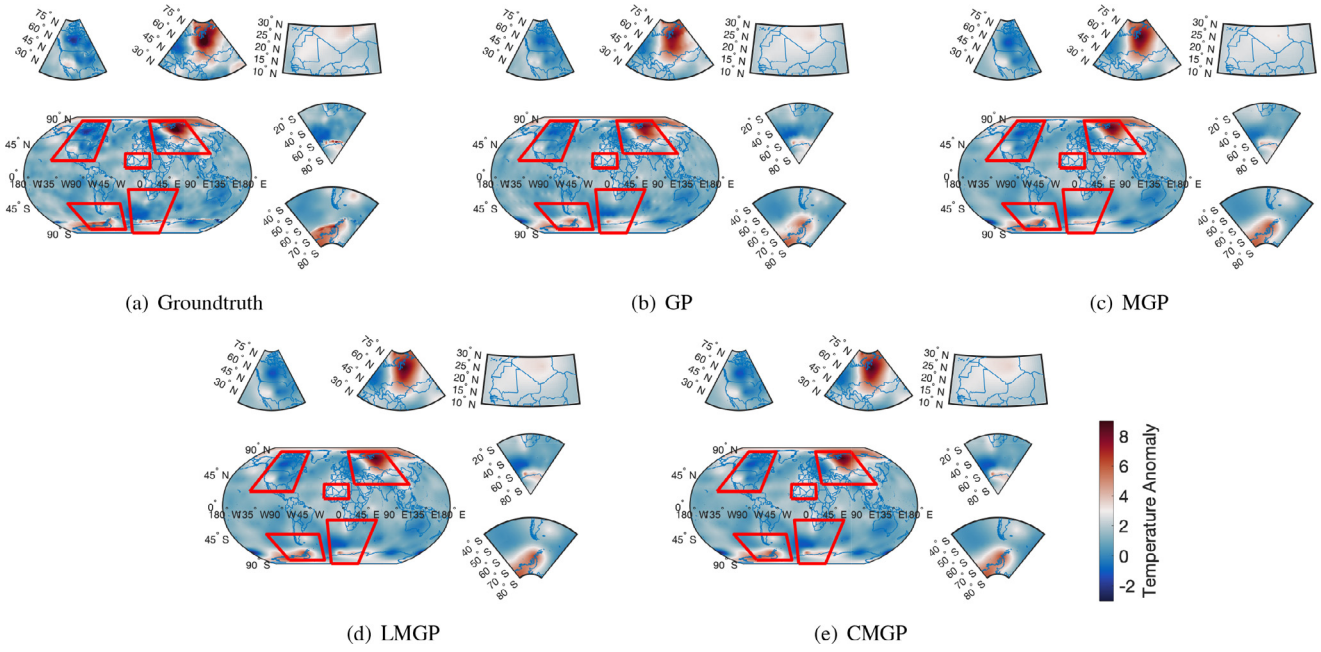
5.3. Sensitivity Analysis of Parameters

For LMGP, we vary the parameter K in $\{16, 32, 64, 128\}$ and plot the RMSE and MAE in Fig. 5. The best performances are usually achieved by $K = 128$. When $K = 64$, the results are comparable with the best results. The performances degenerate significantly if we further decrease K. This observation is consistent with the discussion in Section 4.2 and Fig. 2. However, if we further increase K, the overlap between two GP components may be too large, and there may be too many redundant components. Besides, from Table 1 we can see the time cost of LMGP with $K = 128$ is high. Therefore, we do not increase K and set $K = 128$ in our experiments.

For CMGP, we set the number of clusters K in $\{2, 3, 5, 7\}$ and plot the RMSE and MAE in Fig. 6. From Fig. 6, we can see that the performances of CMGP are not sensitive to the choice of K when K varies in a reasonable range. On Berkeley Earth datasets, the performances are comparable when $K = 2$ or 3, but tend to get slightly worse as we further increase K. However, on the Accessibility dataset and the Vegetation dataset, the results are marginally better with larger K. The reason is the input regions do not cover the globe in these two datasets and exhibits a stronger clustering property as indicated in Fig. 7.

5.4. Empirical Analysis on the Time Complexity of LMGP

In this subsection, we empirically study the time complexity of LMGP. From the discussion in Section 4.2, the training complexity of LMGP mainly depends on dataset size N and parameter K, and the approximate testing complexity of LMGP mainly depends on parameters K, L. In Fig. 8, we illustrate the relationship between time complexity of LMGP and these parameters. We use the non-linear function $\sin(x/20) + \cos(x/10) + \sin(\cos(x/30)) + \cos(\sin(x/40)) + \exp(0.05 * x/100) - x/200$ to generate N random samples in $[-100, 100]$ for training, where N varies in $\{2000, 2500, 3000, 5000, 6000, 7500, 10000\}$. Then we vary



(a) Groundtruth (b) GP (c) MGP

(d) LMGP (e) CMGP

Fig. 4. Prediction results of GP, MGP, LMGP and CMGP on 2020/6

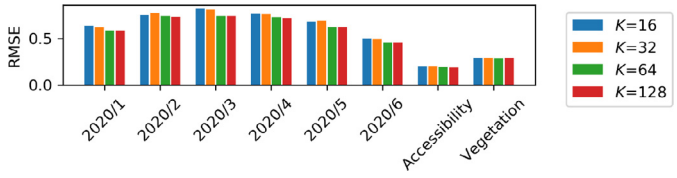


Fig. 5. Sensitivity analysis of LMGP with respect to K.

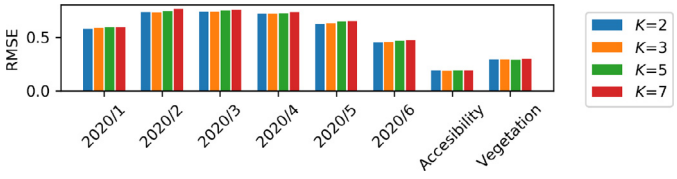


Fig. 6. Sensitivity analysis of CMGP with respect to K.

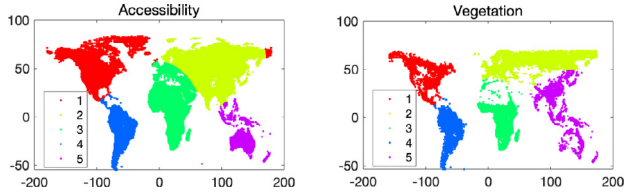
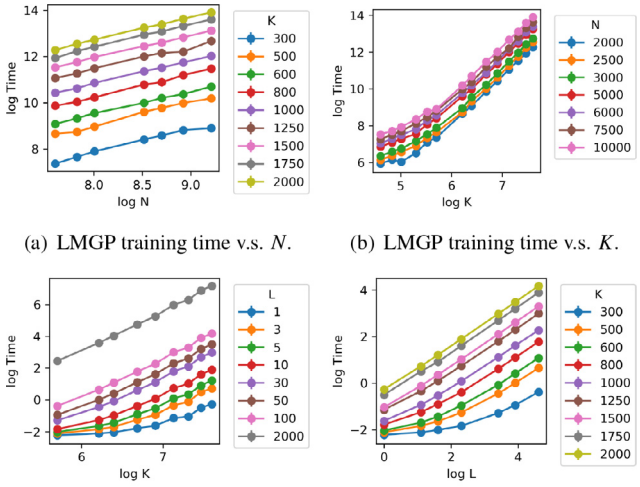


Fig. 7. Clustering results of the accessibility dataset and the Vegetation dataset.



(a) LMGP training time v.s. N. (b) LMGP training time v.s. K.

(c) LMGP testing time v.s. K. (d) LMGP testing time v.s. L.

Fig. 8. Time complexity of LMGP (in log scale).

K in $\{300, 500, 600, 800, 1000, 1200, 1500, 1750, 2000\}$ and L in $\{1, 3, 5, 10, 30, 50, 2000\}$ to investigate the running times of LMGP under various parameter settings. In the testing phase, we fix $N = 2000$, so $L = 2000$ means we use all Gaussian process components to make prediction without approximation.

For training, the theoretical complexity is $\mathcal{O}(N^2(p+K)+NK^3)$, whose dominating term grows quadratically with respect to N. However, in practice, the coefficient of the linear term K^3 is so large that $N^2(p+K) \ll NK^3$, thus the dominating term is NK^3, which grows linearly with respect to N and cubically with respect to K. In Fig. 8(a), the slopes of these curves are approximately 1 in the log scale, which indicates that the practical training time of LMGP grows linearly with respect to N. In Fig. 8(b), the slopes of these curves are approximately 3 in the log scale, which indicates that the practical training time of LMGP grows cubically with respect to K. The theoretical complexity for testing is $\mathcal{O}(LK^2 + N(p+L))$. Similarly, since $K^2 > N$ in our setting, the dominating term is LK^2. From Fig. 8(c) and 8(d) we can also validate the practical testing time of LMGP grows linearly with respect to L and quadratically with respect to K, especially when K and L are relatively large.

5.5. Shape Classification

In this section, we conduct experiments on shape classification to show that CMGP and LMGP can also be applied for classification tasks. In shape classification, the input is the shape recorded by landmarks (pre-shapes), which can be transformed to complex vectors in the Kendall shape space [31]. Therefore, we use the met-

Table 2
Classification accuracy and running time (in seconds) of the proposed methods and competing methods on the Apes dataset and the Mice dataset.

MethodDataset	Apes		Mice	
	Accuracy	Time	Accuracy	Time
k-NN ($k=1$)	74.47%	0.00	73.08%	0.00
k-NN ($k=3$)	78.72%	0.00	73.08%	0.00
k-NN ($k=5$)	78.72%	0.00	73.08%	0.00
k-NN ($k=10$)	74.47%	0.00	69.23%	0.00
SVM	70.21%	0.60	100.00%	0.04
GP	78.72%	17.16	69.23%	6.68
MGP ($K=2$)	23.40%	17.14	46.15%	8.50
MGP ($K=3$)	19.15%	16.23	50.00%	11.43
MGP ($K=4$)	21.28%	20.42	46.15%	12.59
CMGP ($K=2$)	80.85%	19.57	76.92%	8.10
CMGP ($K=3$)	78.72%	23.30	73.08%	11.24
CMGP ($K=4$)	78.72%	26.36	73.08%	12.54
LMGP ($K=16$)	74.47%	664.86	73.08%	168.75
LMGP ($K=32$)	80.85%	840.93	76.92%	243.28
LMGP ($K=64$)	82.98%	1269.32	-	-

ric of the Kendall shape space to derive the distance function. We use the Apes dataset and the Mice dataset [31]. The Apes dataset records ape skull landmarks of 29 male and 30 female adult gorillas, 28 male and 26 female adult chimpanzees, and 30 male and 24 female adult orangutans. We randomly choose 120 samples for training and the rest for testing. The Mice dataset records T2 mouse vertebra landmarks, and the data have three categories: 30 control, 23 large, 23 small. We randomly choose 50 samples for training and the rest for testing. The results are reported in Table 2. We observe that the performances of MGP are very poor since it cannot correctly describe the distribution of input variables in the Kendall shape space. However, CMGP and LMGP still achieve satisfying results, and the performances are usually better than a single GP. We do not claim the superiority of CMGP and LMGP over other methods, since SVM obtains 100% accuracy on the Mice dataset. Nevertheless, CMGP and LMGP outperform a single GP by introducing the mixture structure, and they are more flexible than conventional MGPs.

6. Conclusion and Discussion

In this paper, the attention mechanism offers a new perspective to understand the mixture of Gaussian processes model. Furthermore, two novel mixture of Gaussian processes models (LMGP and CMGP) have been proposed based on the attention mechanism. Unlike conventional MGP models, the proposed methods do not require probabilistic assumptions on the inputs, thus it is applicable to non-Euclidean data. Experimental results on real-world datasets show that LMGP and CMGP are effective. In the future, it is interesting to explore the relationship between the attention mechanism and other statistical learning methods. Besides, based on the attention mechanism, it is promising to design new types of MGP models according to specific application scenes.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the National Key Research and Development Program of China under grant 2018AAA0100205.

References

[1] C.K. Williams, C.E. Rasmussen, Gaussian processes for machine learning, volume 2, MIT press Cambridge, MA, 2006.
[2] V. Tresp, Mixtures of gaussian processes, in: Advances in Neural Information Processing Systems, 2001, pp. 654–660.
[3] C.E. Rasmussen, Z. Ghahramani, Infinite mixtures of gaussian process experts, in: Advances in Neural Information Processing Systems, 2002, pp. 881–888.
[4] Z. Chen, J. Ma, Y. Zhou, A precise hard-cut em algorithm for mixtures of gaussian processes, in: International Conference on Intelligent Computing, Springer, 2014, pp. 68–75.
[5] T. Nguyen, E. Bonilla, Fast allocation of gaussian process experts, in: International Conference on Machine Learning, 2014, pp. 145–153.
[6] C. Luo, S. Sun, Variational mixtures of gaussian processes for classification, in: International Joint Conference on Artificial Intelligence, 2017, pp. 4603–4609.
[7] D. Wu, J. Ma, An effective em algorithm for mixtures of gaussian processes via the mcmc sampling and approximation, Neurocomputing 331 (2019) 366–374.
[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[9] J.D.M.-W.C. Kenton, L.K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL-HLT, 2019, pp. 4171–4186.
[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2020.
[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[12] W. You, S. Sun, M. Iyyer, Hard-Coded Gaussian Attention for Neural Machine Translation, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7689–7700.
[13] S. Zhang, Y. Feng, Modeling Concentrated Cross-Attention for Neural Machine Translation with Gaussian Mixture Model, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 1401–1411.
[14] J. Kim, M. El-Khamy, J. Lee, T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 6649–6653.
[15] Y. Kashiwagi, E. Tsunoo, S. Watanabe, Gaussian kernelized self-attention for long sequence data and its application to ctc-based speech recognition, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 6214–6218.
[16] C. Niu, J. Zhang, G. Wang, J. Liang, Gatcluster: Self-supervised gaussian-attention network for image clustering, in: European Conference on Computer Vision, Springer, 2020, pp. 735–751.
[17] Z. Qiao, X. Qin, Y. Zhou, F. Yang, W. Wang, Gaussian constrained attention network for scene text recognition, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 3328–3335.
[18] D. Ruan, D. Wang, Y. Zheng, N. Zheng, M. Zheng, Gaussian Context Transformer, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15129–15138.
[19] Y. Chen, Q. Zeng, H. Ji, Y. Yang, Skyformer: Remodel Self-Attention with Gaussian Kernel and Nystr\" om Method, Advances in Neural Information Processing Systems 34 (2021).
[20] J. Xie, Z. Ma, D. Chang, G. Zhang, J. Guo, GPCA: A probabilistic framework for gaussian process embedded channel attention, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).
[21] C. Yuan, C. Neubauer, Variational mixture of gaussian process experts, in: Advances in Neural Information Processing Systems, 2009, pp. 1897–1904.
[22] L. Zhao, J. Ma, A specialized probability density function for the input of mixture of gaussian processes, in: International Conference on Intelligence Science, Springer, 2018, pp. 70–80.
[23] X. Li, T. Li, J. Ma, The unν-hardcut em algorithm for non-central student-t mixtures of gaussian processes, in: 2020 15th IEEE International Conference on Signal Processing, volume 1, IEEE, 2020, pp. 289–294.
[24] X. Guo, X. Li, J. Ma, Variational em algorithm for student-t mixtures of gaussian processes, in: International Conference on Intelligent Computing, Springer, 2021, pp. 552–563.
[25] W.S. Cleveland, S.J. Devlin, Locally weighted regression: an approach to regression analysis by local fitting, Journal of the American statistical association 83 (403) (1988) 596–610.
[26] R.T. Rockafellar, Convex analysis, volume 28, Princeton university press, 1970.
[27] H.-S. Park, C.-H. Jun, A simple and fast algorithm for k-medoids clustering, Expert Systems with Applications 36 (2) (2009) 3336–3341.
[28] E. Schubert, P.J. Rousseeuw, Faster k-medoids clustering: improving the pam, clara, and clarans algorithms, in: International Conference on Similarity Search and Applications, Springer, 2019, pp. 171–187.
[29] R.A. Rohde, Z. Hausfather, The Berkeley Earth land/ocean temperature record, Earth System Science Data 12 (4) (2020) 3469–3479.
[30] H. Uchida, A. Nelson, Agglomeration index: Towards a new measure of urban, concentration (2009).
[31] I.L. Dryden, K.V. Mardia, Statistical shape analysis: with applications in R, volume 995, John Wiley & Sons, 2016.