Cluster-based Kriging approximation algorithms for complexity reduction

Stein, B. van; Wang, H.; Kowalczyk, W.J.; Emmerich, M.T.M.; Bäck, T.H.W.

Applied Intelligence
https://doi.org/10.1007/s10489-019-01549-7

Cluster-based Kriging approximation algorithms for complexity reduction

Bas van Stein[1] · **Hao Wang**[1] · **Wojtek Kowalczyk**[1] · **Michael Emmerich**[1] · **Thomas Bäck**[1]

Abstract

Kriging or *Gaussian Process Regression* is applied in many fields as a non-linear regression model as well as a surrogate model in the field of evolutionary computation. However, the computational and space complexity of Kriging, that is cubic and quadratic in the number of data points respectively, becomes a major bottleneck with more and more data available nowadays. In this paper, we propose a general methodology for the complexity reduction, called cluster Kriging, where the whole data set is partitioned into smaller clusters and multiple Kriging models are built on top of them. In addition, four Kriging approximation algorithms are proposed as candidate algorithms within the new framework. Each of these algorithms can be applied to much larger data sets while maintaining the advantages and power of Kriging. The proposed algorithms are explained in detail and compared empirically against a broad set of existing state-of-the-art Kriging approximation methods on a well-defined testing framework. According to the empirical study, the proposed algorithms consistently outperform the existing algorithms. Moreover, some practical suggestions are provided for using the proposed algorithms.

Keywords Kriging · Gaussian process regression · Fuzzy clustering · Clustering · Model trees · Time complexity

1 Introduction

Kriging, or *Gaussian Process Regression* [31] is a popular and elegant kernel based regression model capable of modeling very complex functions. Kriging is used in many fields e.g. engineering, mining and geology, as a tool for the analysis of datasets, for prediction purposes and for *Surrogate model based optimization* [36]. Many other regression models exist, such as parametric models, which are easy to interpret but may lack expressive power to model complex functions. On the other hand, *Regression Tree* based methods like *Random Forests* [3] or *Gradient Boosted Decision Trees* lack the advantage of interpretation [11] but have more expressive power. Another method is *Linear Model Trees* [41], which uses a tree structure with linear models at the leaves of the tree. There are also more complex algorithms like *Neural Networks*, or *Extreme Learning Machines* [21], that are able to model very complex functions but are usually not easy to work with in practice. There are also different kernel based methods such as *Support Vector Machines* [43] and *Radial Basis Functions* [5]. The main advantage of Kriging over other regression methods is that Kriging provides not only the estimate of the value of a function, but also the mean squared error of the estimation, the so-called *Kriging variance*. The Kriging variance can be seen as the uncertainty assessment of the model and has been exploited in surrogate model based optimization and many other applications. Despite the clear advantage of the Kriging variance, Kriging suffers from one major problem, the high training time and space complexity, which are $O\left(n^3\right)$ and $O\left(n^2\right)$, respectively. Where n denotes the number of points. To overcome this complexity problem, Kriging approximation algorithms such as [42] and [19] are introduced. Unfortunately, these approximation algorithms are usually less accurate than the original Kriging algorithm.

In this paper, a novel algorithmic framework, called *Cluster Kriging* (CK), is proposed to reduce the time / space complexity issue when fitting the Kriging model to a large dataset. The Cluster Kriging framework consists of three steps: 1) The whole dataset is partitioned into small clusters; 2) Multiple Kriging models are trained, one for each partition; 3) All Kriging models are combined to predict unknown locations. Each of the steps can be realized using

Bas van Stein and Hao Wang contributed equally to this work

✉ Bas van Stein
b.van.stein@liacs.leidenuniv.nl

[1] Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, Leiden, The Netherlands

Published online: 09 September 2019

several alternative methods, resulting in a potential wide set of instance algorithms under the Cluster Kriging framework. Here, four instance algorithms are proposed and empirically compared to the current state-of-the-art Kriging approximation algorithms. A well-defined testing framework for Kriging approximation algorithms [7] is adopted for the comparison.

This paper is organized as follows. In Section 2, a concise introduction is given on the Kriging method, along with the time / space complexity thereof. In Section 3, the current state-of-the-art approaches to reduce the complexity issue are briefly reviewed and categorized. The Cluster Kriging framework is then formulated in Section 4, which is followed by a descriptions of four Cluster Kriging "flavors" / algorithm instances in Section 5. The experimental setup as well as a detailed discussion on the results are provided for Cluster Kriging and other state-of-the-art approaches in Section 6. Finally, we conclude the paper and point out the future research directions in Section 7.

2 Kriging

Notation Throughout this paper, we shall use n, k, d to denote the number of data points, the number of clusters and the dimensionality of the input space, respectively. In addition, we consider a regression function $f : \mathbb{R}^d \to \mathbb{R}$. In complexity statements in this paper we ignore d since Kriging is generally used on low dimensional datasets. Without loss of generality, the column vector convention is adopted as the notation used throughout this paper.

Loosely speaking, Kriging is a stochastic interpolation method in which the output value of a stochastic process is predicted as a linear function of the observed output values [24, 40]. In particular, Kriging is the best linear unbiased predictor (BLUP) and the corresponding mean squared error of prediction is used for uncertainty qualification. Kriging originates from the field of spatial analysis/geostatistics and more recently is being widely used in Bayesian optimization and design and analysis of computer experiments (DACE) [22, 34]. The model features in providing the theoretical uncertainty measurement of estimations.

When this stochastic process is assumed to be Gaussian, Kriging is equivalent to Gaussian Process Regression (GPR), where the *posterior* distribution of the regression function (posterior Gaussian process) is inferred through Bayesian statistics. In this paper, we shall consider this special case and adopt the mathematical treatment of the Gaussian process. Assume that input data points are summarized in the set $\mathcal{X} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\right\} \subseteq \mathbb{R}^d$ and the corresponding output variables are represented as $\mathbf{y} = \left[y\left(\mathbf{x}^{(1)}\right), y\left(\mathbf{x}^{(2)}\right), \ldots, y\left(\mathbf{x}^{(n)}\right)\right]^\top$. Specifically, the mostly used variant of Kriging, *Ordinary Kriging*, models the regression function f as a random process, that is a combination of an *unknown* constant trend μ with a centered Gaussian Process ε. The response y is linked to f through the homoscedastic Gaussian noise γ (namely the noises are independent and identically distributed):

$$y(\mathbf{x}) = f(\mathbf{x}) + \gamma(\mathbf{x}) = \mu + \varepsilon(\mathbf{x}) + \gamma(\mathbf{x}),$$

$$\varepsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma_\varepsilon^2(\mathbf{x})), \quad \gamma(\mathbf{x}) \sim \mathcal{N}(0, \sigma_\gamma^2).$$

Note that the noise process γ is assumed independent from ε. The centered Gaussian process ε is a stochastic process which possesses zero mean everywhere and any finite collection of its random variables has a joint Gaussian distribution [31]. It can be completely specified by providing function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to calculate the pairwise covariance:

$$\mathrm{Cov}[\varepsilon(\mathbf{x}), \varepsilon(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}').$$

The covariance function $k(\cdot, \cdot)$ is a kernel function which implies a Reproducing Kernel Hilbert Space (RKHS) of the regression function f. Moreover, the variance $\sigma_\varepsilon^2(\mathbf{x})$ of a Gaussian process ε is independent from the input $\mathbf{x}$ and thus denoted as σ_ε^2 in the following. In practice, a common choice is the Gaussian covariance function (also known as squared exponential kernel):

$$k(\mathbf{x}, \mathbf{x}') = \sigma_\varepsilon^2 \prod_{i=1}^{d} \exp\left(-\theta_i (x_i - x_i')^2\right), \tag{1}$$

where θ_i's are called hyper-parameters, that are either predetermined or estimated through model fitting, and σ_ε^2 is inferred by the maximum likelihood method, which is omitted here for simplicity. Using the Gaussian kernel, the stochastic process ε is stationary.

To infer output value $y^{(t)} = y\left(\mathbf{x}^{(t)}\right)$ at an unobserved data point $\mathbf{x}^{(t)}$, the joint distribution of $y^{(t)}$ and observed outputs $\mathbf{y}$ are derived, conditioning on the input dataset $\mathcal{X}$, $\mathbf{x}^{(t)}$ and the unknown prior mean μ. Such a joint distribution is a multivariate Gaussian and is expressed as follows;

$$\begin{bmatrix} y^{(t)} \\ \mathbf{y} \end{bmatrix} \Bigg| \, \mathcal{X} \sim \mathcal{N}\left(\mu \mathbf{1}_{n+1}, \begin{bmatrix} \sigma_\varepsilon^2 + \sigma_\gamma^2 & \mathbf{c}^\top \\ \mathbf{c} & \boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I} \end{bmatrix}\right), \tag{2}$$

$$\mathbf{c}_i = k(\mathbf{x}^{(t)}, \mathbf{x}^{(i)}), \quad \boldsymbol{\Sigma}_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}),$$

where $\mathbf{1}_{n+1}$ denotes a column vector of length $n+1$ that contains only 1's. The homogeneous variance σ_γ^2 of the noise can be either determined by the user or estimated through the maximum likelihood method. The *posterior* distribution of $y^{(t)}$ can be calculated by marginalizing μ out and conditioning on the observed output variables $\mathbf{y}$ [31]. Without any derivations, the posterior distribution for Ordinary Kriging is again Gaussian [17]:

$$y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)} \sim \mathcal{N}\left(m\left(\mathbf{x}^{(t)}\right), s^2\left(\mathbf{x}^{(t)}\right)\right) \tag{3}$$

where the posterior mean and variance are expressed as:

$$m\left(\mathbf{x}^{(t)}\right) = \hat{\mu} + \mathbf{c}^\top \left(\boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} (\mathbf{y} - \hat{\mu}\mathbf{1}_n) \tag{4}$$

$$s^2\left(\mathbf{x}^{(t)}\right) = \sigma_\gamma^2 + \sigma_\varepsilon^2 - \mathbf{c}^\top \left(\boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{c} + \frac{(1 - \mathbf{c}^\top \left(\boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{1}_n)^2}{\mathbf{1}_n^\top \left(\boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{1}_n}$$

$$\hat{\mu} = \frac{\mathbf{1}_n^\top \left(\boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{y}}{\mathbf{1}_n^\top \left(\boldsymbol{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{1}_n} \tag{5}$$

Note that the estimation of the trend, $\hat{\mu}$ is obtained by the maximum a posteriori principle (MAP). The posterior mean function (4) is used as the predictor while the posterior variance (5) is the so-called *Kriging variance* that measures the uncertainty of the prediction.

3 Relevant research

Despite the theoretically sound development of the Kriging model, it suffers from several issues when applied to large datasets. The major bottleneck is the high time and memory complexity of the model fitting process: The inverse of the covariance matrix $\boldsymbol{\Sigma}^{-1}$ needs to be computed for both the posterior mean and variance (4 and 5), which has roughly $O(n^3)$ time complexity.[1] Moreover, when optimizing the hyper-parameters of the kernel function, the log likelihood function of those parameters is again calculated through $\boldsymbol{\Sigma}^{-1}$, resulting in a $O(n^3)$ computational cost per each optimization iteration. Thus, for a large dataset, such a high overhead in model fitting renders Kriging inapplicable in practice. Various attempts have been made to overcome the computational complexity issue of Kriging [31]. The contributions towards solving this issue can be split into three categories:

3.1 Subset methods

The first category of approximation algorithms uses only a subset of the complete dataset to approximate a full Kriging model. The idea behind these methods is to get a realistic representation of the complete dataset by taking only a small portion of the data points. The main issue with these subset approximation algorithms is how to identify a subset that represents the complete dataset.

Subset of Data (SoD) [26] is a naive approach in reducing complexity by taking a subset of $m < n$ data points. The points are ussualy taken at random. The obvious disadvantage of such an approach is that possible valuable information is lost in the process. Taking a representative subset of data points is a non-trivial task.

Subset of Regressors (SoR) [35] approximates Kriging by a linear combination of kernel functions on a set of basis points. The basis points are linearly weighted to construct the predictor. The choice of the basis points *does* influence the final outcome significantly. As noted also in [30], there are only m (number of basis points) degrees of freedom in the model because the model degenerates (finite linear-in-the-parameters), which might be too restrictive.

3.2 Approximation using sparsity

The second category of approximation algorithms approximate the covariance matrix using sparsity based methods. Most of these algorithms also use a (relevant) subset of the data like in the category mentioned above.

Sparse On-Line Gaussian Processes (OGP) [9] uses a Bayesian on-line algorithm, together with a sequential construction of a subsample of the data that specifies the prediction of the GP model. The idea behind constructing a subsample of basis vectors is very similar to The Fully Independent Training Conditional mentioned next. The advantage of OGP is that additional data points can be added to the OGP model without always completely retraining the model.

Fast Kriging with Gaussian Markov Random Fields [19] is an algorithm that uses an approximation of the covariance matrix with a sparse precision matrix. It uses *Gaussian Markov Random Fields* (GMRF) on a reasonable dense grid to exploit the computational benefits of a Markov field while keeping the formula of Kriging weights. This method reduces the complexity for simple and ordinary Kriging, but might not always be efficient with universal Kriging.

The Fully Independent Training Conditional (FITC) [28, 37]. Snelson and Ghahramani proposed what they called Sparse Gaussian Processes using Pseudo-inputs. It uses a more sophisticated likelihood approximation with a richer covariance. It is a non-degenerate version of the *SoR* algorithm. By providing a set of basis points (Pseudo inputs), the model is fitted and validated on the training data. As with *SoR* the choice of basis points is a problem, this is usually either a subset of the training data or a uniform distribution over the input space.

[1] There are asymptotically faster algorithms for matrix inversion, e.g. Strasssen's $O(n^{2.807})$ and Stothers $O(n^{2.373})$, but their practical performance is worse than some methods with $o(n^3)$ time complexity.

3.3 Divide and conquer methods

The last category contains methods that divide a (big) dataset into several smaller datasets and build a model for each of them. How to split the dataset into smaller datasets and how to combine the different models is what makes these algorithms unique. The proposed Cluster Kriging algorithms also belong to this category.

Bayesian Committee Machines (BCM) [42] is an algorithm similar to the ones we propose, but developed from a completely different perspective. The basic motivation is to divide a huge training set into several relatively small subsets and then construct Kriging models on each subset. The benefit of this approach is that the training time on each subset is satisfactory and the training task can be easily parallelized. After training, the prediction is made by a weighted combination of estimations from all the Kriging models. BCM uses batch prediction to speed up the computation even further. However, BCM does not seem to correct for different hyper parameters per module, neither for badly fitted modules, which becomes a major problem when the number of modules increases.

Product of Experts (PoE) [20] and the Generalized variant (GPoE) [6] are aggregation methods that differs from Bayesian Committee Machines in the following sense: the predictive distribution is assumed as a product of the posterior distribution of each local Kriging model. Moreover, Generalized Product of Experts (GPoE) imposes individual exponents on each probability distribution in the product, where the exponent stands for the reliability of each local Kriging model at a particular point.

Generalized Robust Bayesian Committee Machine (GRBCM) [27] generalizes BCM by introducing a so-called *global communication expert* (Kriging model) that is trained on a random (hence global) subset of data and interacts with each local Kriging model. The rationale behind it is: the local predictive distribution (on each local Kriging model) can be improved by incorporating information on the global communication expert into each local Kriging model.

Nested Pointwise Aggregation of Experts (NPAE) [33] considers the fact that local predictors as $m(\cdot)$ are again Gaussian random variables. Under this consideration, it is straightforward to express the covariance among all local predictors and the unknown output $y^{(t)}$. Then, the overall predictive distribution of $y^{(t)}$ can be obtained by conditioning on all local predictors.

Several other attempts have been made to divide the Kriging model in sub-models [8, 29], each solution for different domains. In [8], a *Bagging* [2] method is proposed to increase the robustness of the Kriging algorithm, rather than speeding up the algorithm's training time. In [29], a partitioning method is introduced to separate the data points into local Kriging models and combine the different models using a distance metric.

All of these approximation algorithms have their advantages and disadvantages and they are compared to our Cluster Kriging algorithms.

For the empirical study, eight state of the art algorithms: *SoD*, *FITC*, *BCM*, *RBCM*, *GRBCM*, *PoE*, *GPoE* and *NPAE* are selected to be compared with the proposed approaches in this paper, as they seem to be the most popular and prominent in the field.

4 Cluster Kriging

The main idea behind the proposed approach, *Cluster Kriging*, is to combine multiple Kriging models trained on each partition of data, where the partitions are obtained from clustering algorithms. Loosely speaking, if the whole dataset is partitioned into clusters of similar sizes, Cluster Kriging will reduce the time complexity by a factor of k^2 resulting in $k\left(\frac{n}{k}\right)^3$ (where k is the number of clusters) if Kriging models are fitted sequentially. When exploiting k CPU processes in parallel, the time complexity will be further reduced to $\left(\frac{n}{k}\right)^3$. In practice this means that if we take k depending on n our algorithm becomes quadratic in time, and using k CPUs it even reaches linear time complexity. For the output value $y^{(t)}$ at an unobserved data point $\mathbf{x}^{(t)}$, each Kriging model provides a (local) prediction for $y^{(t)}$. To obtain a global prediction, it is proposed to either combine the predictions from all the Kriging models or select the most proper Kriging model for the prediction.

There are many options for the data partitioning, e.g. K-means and Gaussian mixture models (GMM), and the local Kriging models trained on the clusters can also be combined in different manners. By varying the options in each step of the Cluster Kriging procedure, many algorithms can be generated. Four of them will be explained in the next section. In this section, the options in each step of the algorithms are introduced gradually.

4.1 Clustering

The first step in the Cluster Kriging methodology is the clustering of the input data $\mathcal{X}$ (and the output variables) into several smaller datasets. In general, the goal is to obtain a set $\mathcal{S}$ containing k clusters on the input dataset $\mathcal{X}$.

$$\mathcal{S} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k\}, \quad \text{where} \bigcup_{i=1}^{k} \mathcal{X}_i = \mathcal{X}. \tag{6}$$

In addition, the output values $\mathbf{y}$ are also grouped accordingly: $\mathbf{y} = \left[\mathbf{y}_1^\top, \mathbf{y}_2^\top, \ldots, \mathbf{y}_k^\top\right]^\top$. The clustering can be done

in many ways, with the most simple and feasible approach being random clustering. For our framework the following three simple but effective clustering methods are used. Note that, in principle, more sophisticated clustering method, e.g., K-Medoids based methods [18], spectral-based method [12] and density peaks clustering [13, 46] can also be adopted here.

4.1.1 Hard clustering

Hard clustering splits the data into k smaller *disjoint* datasets:

$$\bigcap_{i=1}^{k} \mathcal{X}_i = \emptyset$$

This can be achieved by various methods, for instance the K-means algorithm (7). K-means clustering minimizes the within-cluster sum of squares, that is expressed as:

$$\arg\min_{\mathcal{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{X}_i} ||\mathbf{x} - \boldsymbol{\mu}^{(i)}||^2, \tag{7}$$

where $\boldsymbol{\mu}^{(i)}$ is the centroid of cluster i and is calculated as the mean of the points in $\mathcal{X}_i$. The evaluation of the within-cluster sum of squares takes $O(nkd)$ execution time.

4.1.2 Soft clustering

Instead of using a hard clustering approach, a fuzzy clustering algorithm can be used to introduce slight overlap between the various smaller datasets, which might increase the final model accuracy. To incorporate fuzzy clustering, instead of directly applying cluster labels, the probabilities that a point belongs to a cluster are calculated (8) and for each cluster $(n \cdot o)/k$ points with the highest membership values are assigned, where o is a user defined setting that defines the overlap. o is set between 1.0 (no overlap) and 2.0 (completely overlapping clusters).

In principle, any fuzzy clustering algorithm can be used for the partitioning. In this paper the *Fuzzy C-Means* (FCM) [14] clustering algorithm and the *Gaussian Mixture Models* (GMM) [32] are used. FCM is a clustering algorithm very similar to the well known *K-means*. The algorithm differs from K-means in that it has additional membership coefficients and a fuzzifier. The membership coefficients of a given point give the degrees that this point belongs to each cluster. These coefficients are normalized so they sum up to one. The algorithm can be fitted on a given dataset and returns the coefficients for each data point to each cluster. The number of clusters is a user defined parameter. Fuzzy C-means optimizes the objective function given in (8) iteratively. In each iteration, the membership coefficients of each point being in the clusters are computed using (9). Subsequently, the centroid of each cluster $\boldsymbol{\mu}^{(j)}$ is computed as the center of mass of all data points, taking the membership coefficients as weights. The objective of fuzzy C-means is to find a set of centroids that minimizes the following function:

$$\sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{m} ||\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)}||^2, \tag{8}$$

where w_{ij} are the membership values (see (9)) and m is the so-called fuzzifier (set to 2 in this paper). The fuzzifier determines the level of cluster fuzziness as follows:

$$w_{ij}^{m} = \frac{1}{\sum_{c=1}^{k} \left(\frac{||\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)}||}{||\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(c)}||} \right)^{\frac{2}{m-1}}} \tag{9}$$

The other fuzzy clustering procedure used is the Gaussian Mixture Models. GMM are used together with the *expectation-maximization* (EM) algorithm for fitting the Gaussian models. The mixture models are fitted on the training data and later used in the weighted combination of the Kriging models by estimating cluster membership probabilities of the unseen data points. The advantage of this clustering technique is that it is fairly robust and that the number of clusters can be specified by the user. For the GMM method one could use the full covariance matrix whenever the dimensionality of the input data is small. However, when working with high dimensional data a diagonal covariance matrix can be used instead. The time complexity of GMM depends on the underlying EM algorithm. In each iteration EM, it takes $O(nk)$ operations to re-estimate the model parameters.

4.1.3 Regression tree partitioning

The third method used is the partitioning by use of a Regression Tree [4] on the complete training set. The regression tree splits the dataset recursively at the best splitting point using the variance reduction criterion. Each leaf node of the Regression Tree represents a cluster of data points. The number of leaves (or the number of samples per leave) can be set by the user. By reducing the variance in each leaf node and therefore the variance in each dataset, the Kriging models can be fitted to the local datasets much better as will be presented later on. The time complexity of using a Regression Tree for the partitioning is $O(n)$, given that the depth of the tree or the number of leaf nodes is set by the user.

The partitioning done by the regression tree depends on the splitting criterion. For a faster execution of the Cluster Kriging algorithm we could choose to use a splitting criterion that splits the dataset in each node evenly,

balancing the load for each of the local Kriging models attached to the leafs. From emperical experience we know that splitting using the standard variance reduction function generally results in better performing models than using such an evenly splitting criterion. This is likely due to the fact that datasets with a lower variance can be more easily fitted by a Kriging model.

4.2 Modeling

After partitioning the dataset into several clusters, Kriging models are fitted on each of the smaller datasets. The Kriging algorithm is applied on each cluster individually, this way each model will be optimized on its own training set and will have different hyper-parameters. For simplicity we assume, in this paper, the kernel functions used on each cluster to be the same. As for the regression tree approach, the dataset, or more precisely the input space, is partitioned by the tree algorithm and, for each leaf node, a Kriging model is computed using the data belonging to this node (Fig. 1). A similar technique is introduced in the context of combining linear regression models [25, 41, 45]. For each cluster $l = 1, 2, \ldots, k$, the predictive (posterior) distribution of the response $y^{(t)}$ is:

$$y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)} \sim \mathcal{N}\left(m_l\left(\mathbf{x}^{(t)}\right), \sigma_l^2\left(\mathbf{x}^{(t)}\right)\right), \tag{10}$$

where m_l and σ_l^2 are specified again by (4) and (5) except that $\mathcal{X}, \mathbf{y}$ are replaced by $\mathcal{X}_l, \mathbf{y}_l$ here. Note that building the Kriging models can be easily parallelized, which gives an additional speedup to Cluster Kriging. Another benefit of building each model separately, is that each model has usually a much better local fit than a single global Kriging model would obtain.

4.3 Prediction

After training the various Kriging models, unseen data points need to be predicted. For this prediction, there are several options. Depending on the partitioning method used before, the simplest approach to predict the unseen data point is by using a single local model. When the partitions are overlapping a combination of the different local models into one global model is required.

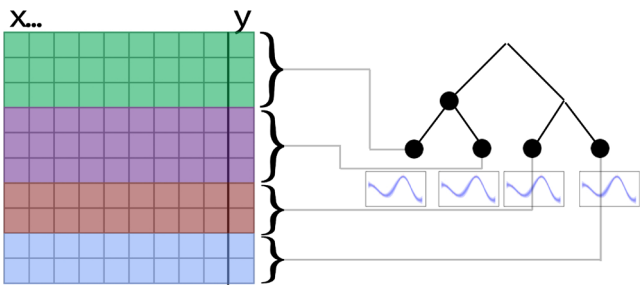


Fig. 1 Visualisation of a Model Tree. The top node is the root and the bottom nodes are the leaves with attached models. Each record in the data (on the left) is assigned to a leaf node of the regression tree

4.3.1 Single model prediction

The simplest method is to pick just one local Kriging model for each data point and use this local model for the prediction. This does require the partitioning used to create partitions based on locality like k-means clustering or a regression tree. First the partitioning method is used to predict which cluster the new data point belongs to, then the Kriging model trained using this particular cluster is used to predict the mean and variance at the new data point.

In case of the Regression Tree procedure, the targets are predicted from new unseen data points by first deciding which model needs to be used, using the Regression Tree. The target is then predicted using the specific Kriging model assigned to the leaf node (Fig. 1). The main advantage of this method is that there is no combination of different predictions and only one of the local Kriging models needs to provide a prediction. This results in a significant speed-up for the prediction task. Disadvantages of this method are 1) a potential lack of the global trend of the target function and 2) discontinuities are created artificially. In Fig. 2, we visualize a Cluster Kriging model using regression trees, in which the intersections between the different local models are marked by black dashed lines. It can be observed that the edges of the local models are not completely matching, meaning that the predictions near the border are not as smooth as they would be in a global Kriging model. It can also be observed that the area covered by each cluster is not the same, this is due to the splitting criterion of the regression tree. While the splitting criterion could be chosen in such a way that it balances the cluster sizes, using variance reduction as the splitting criterion generally gives better fitted local models.

4.3.2 Optimal weighting procedure

Instead of using single model predictions, the multiple local models can be combined into one global model using various combination procedures. When the input dataset is separated by hard clustering methods, the posterior Gaussian processes on each cluster are independent from each other. In this sense, it is possible to construct a global Gaussian process model as the superposition of Gaussian processes from all the clusters. In addition, a weighting scheme is used to model how much "belief" should be put

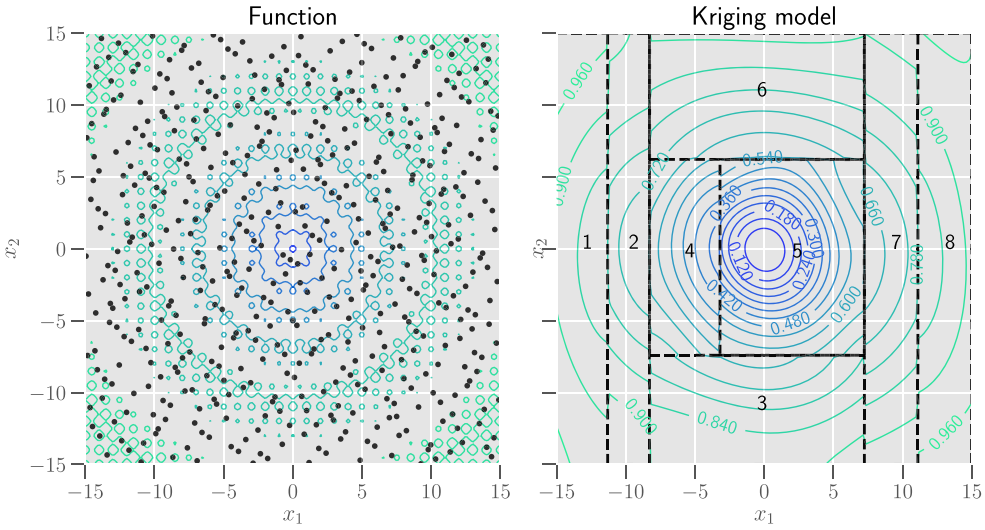


Fig. 2 The landscape of the two dimensional Ackley function on the left, and on the right are the contours of the Model Tree Cluster Kriging mean function, with the tree partitionings visualized by dashed lines. The index of the clusters are shown in the middle of each rectangle

on the prediction from each local model. The weighted posterior processes is [38]:

$$y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)} \sim \mathcal{N}\left(\sum_{l=1}^{k} w_l m_l\left(\mathbf{x}^{(t)}\right), \sum_{l=1}^{k} w_l^2 \sigma_l^2(\mathbf{x}^{(t)})\right).$$

Note that the prediction and its variance depend on the weights. Therefore, the optimal predictor can be achieved by minimizing the variance with respect to the weight, which is formulated as follows:

$$\begin{aligned} &\underset{\{w_1,\ldots,w_k\}}{\text{minimize}} \quad \sum_{l=1}^{k} w_l^2 \sigma_l^2(\mathbf{x}^{(t)}) \\ &\text{subject to} \quad \sum_{l=1}^{k} w_l = 1, \quad w_l \geq 0, \quad l = 1, \ldots, k. \end{aligned}$$

The optimal weights are obtained by solving the problem above (see the previous work of the authors [38, 39] for details):

$$w_l^* = \frac{1/\sigma_l^2\left(\mathbf{x}^{(t)}\right)}{\sum_{i=1}^{k} 1/\sigma_i^2\left(\mathbf{x}^{(t)}\right)}. \tag{11}$$

The optimal weights are then used to construct the optimal predictor, which is the inner product of the model predictions with the optimal weights.

4.3.3 Membership probabilities

For the GMM and other soft clustering approaches, the membership probabilities can be used for unseen records to define the weights for the combination of predictions. For each unseen record, the membership probabilities that this record belongs to the k clusters are calculated and directly used as the weights in the weighted sum of predictions and variances given by the Kriging models:

$$w_l = \Pr\left(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}\right), \quad \text{for } l = 1, \ldots, k, \tag{12}$$

where C is the cluster indicator variable ranging from 1 to k. The rationale behind such a weighting scheme can be shown from the following derivation. In general, the goal here is to express the predictive distribution of variable $y^{(t)}$ that is the conditional density function on the whole dataset $\mathcal{X}$, using the posterior densities from all clusters. By applying the total probability with respect to the cluster indicator variable C, such a density function p can be written as [39]:

$$\begin{aligned} &p\left(y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right) \\ &= \sum_{l=1}^{k} p\left(y^{(t)}, C = l \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right) \\ &= \sum_{l=1}^{k} p\left(y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}\right) \Pr\left(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}\right). \end{aligned} \tag{13}$$

In (13), the first term within the summation is the predictive density function obtained from cluster l. The second term represents the probability that data point $\mathbf{x}^{(t)}$ belonging to a cluster, which is the weight in (12). Consequently, the overall predictive density function is a *mixture* of predictive distributions of all the Gaussian process models on clusters. To predict $y^{(t)}$, the expectation

of the conditional density function $p\left(y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right)$ is calculated:

$$\begin{aligned}
&\mathrm{E}\left[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right] \\
&= \sum_{l=1}^{k} \int_{-\infty}^{\infty} y^{(t)} \sum_{l=1}^{k} g\left(y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}\right) \\
&\quad \times \Pr\left(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}\right) \mathrm{d}y^{(t)} \\
&= \sum_{l=1}^{k} \Pr\left(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}\right) \mathrm{E}\left[y^{t} \mid \mathbf{X}_l, \mathbf{y}_l, \mathbf{x}^{t}\right] \\
&= \sum_{l=1}^{k} w_l m_l\left(\mathbf{x}^{(t)}\right). \qquad (14)
\end{aligned}$$

Recall that $m_l(\mathbf{x}^t)$ is the mean function as shown in (10). Equation 14 suggests that the overall prediction made on the whole dataset can be expressed as a convex combination of the local predictions on each cluster of data, in which the combination weights are membership probabilities of GMM or similar clustering approaches. Furthermore, the variance of the prediction (expectation) above is derived as follows:

$$\begin{aligned}
&\mathrm{Var}\left[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right] \\
&= \mathrm{E}\left[{y^{(t)}}^2 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right] - \mathrm{E}\left[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right]^2 \\
&= \sum_{l=1}^{k} w_l \left(\mathrm{Var}\left[y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}\right] + \mathrm{E}\left[y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}\right]^2\right) \\
&\quad - \mathrm{E}\left[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}\right]^2 \\
&= \sum_{l=1}^{k} w_l \left(\sigma_l^2\left(\mathbf{x}^{(t)}\right) + m_l^2\left(\mathbf{x}^{(t)}\right)\right) - \left(\sum_{l=1}^{k} w_l m_l(\mathbf{x}^{(t)})\right)^2.
\end{aligned}$$

Note that $\sigma_l^2\left(\mathbf{x}^{(t)}\right)$ is again the Kriging variance at point $\mathbf{x}^{(t)}$ from cluster l.

5 Flavors of cluster Kriging

Using the three stages and various components for each stage of the Cluster Kriging methodology, various algorithms can be implemented. In this paper we asses four different flavors of Cluster Kriging:

Optimally Weighted Cluster Kriging (OWCK), which uses a hard (K-means) clustering technique to partition the data into k clusters. Subsequently, a Kriging model is trained on each cluster and to predict unseen data points, the predictions and variances of each model are combined using the Optimal Weights Procedure (Section 4.3.2).

Optimally Weighted Fuzzy Cluster Kriging (OWFCK), which uses a soft clustering technique (Fuzzy C-Means) to partition the data into k overlapping clusters and also uses the Optimal Weights Procedure combining the different predictions (Section 4.3.2).

Gaussian Mixture Model Cluster Kriging (GMMCK), which uses Gaussian Mixture Models to partition the data into k overlapping clusters and the trained Kriging models are weighted using the membership probabilities assigned on the unseen data by the Gaussian Mixture Model (Section 4.3.3).

Model Tree Cluster Kriging (MTCK), the proposed novel algorithm, uses a regression tree with a fixed amount of leaf nodes to partition the data in the objective space. A Kriging model is then trained on each partition defined by the tree's leaves. MTCK uses only one of the trained Kriging models per unseen record to predict (Section 4.3.1), depending on which leaf node the unseen record is assigned to.

First a decision tree regressor is constructed using the complete dataset. The tree is generated from the root node by recursively splitting the training data using the target variable and the variance reduction criterion. Once a node contains less than the minimum samples needed to split or the node contains only one record, the splitting stops and the node is called a leaf. To control the number of clusters, the user can set the maximum number of leaves or the minimum leaf size. Next, each leaf node is assigned a unique index and each record belonging to the leaf is assigned to this index. For each leaf, a Kriging model is computed using only those records assigned to this leaf. Each Kriging model is now able to predict a particular region defined by the Regression Tree.

For the prediction of the target for unseen records, the regression tree decides which Kriging model should be used. The final predicted mean and variance is provided by this Kriging model.

6 Experimental setup and results

A broad variety of experiments is executed to compare all flavours of Cluster Kriging, e.g., Optimally Weighted Cluster Kriging or Model Tree Cluster Kriging, to a wide reference set of other Kriging approximation algorithms: *Bayesian Committee Machines*, both standard BCM as well as RBCM and GRBCM, *Subset of Data* (SoD), *Fully Independent Training Conditional* (FITC), *Product of Experts* (PoE and GPoE), *Nested Pointwise aggregation of Experts* (NPAE), *Optimally Weighted Cluster Kriging* (OWCK) using K-means clustering, *Fuzzy Cluster Kriging* using Fuzzy C-means (OWFCK), Fuzzy Cluster Kriging with Gaussian Mixture Models (GMMCK) and, finally, *Model Tree Cluster Kriging* (MTCK).

The above algorithms are evaluated on three different datasets from the *UCI machine learning repository* [1]:

- *Concrete Strength* [47], a dataset with 1030 records, 8 attributes and one target attribute. The task is to predict the strength of concrete.
- *Combined Cycle Power Plant* (CCPP) [23], a dataset of 9568 records, 3 attributes and one target attribute. The target is the hourly electrical energy output and the task is to predict this target.
- *SARCOS* [44], a dataset from *gaussianprocess.org* with a training set of 44484 records, 21 attributes and 7 target attributes. The task is to predict the joint torques of an anthropomorphic robot arm. All 21 attributes are used as training data but only the 1st target attribute is used as target. The dataset comes with a predefined test set of 4449 records.

In addition, 8 synthetic datasets with each 10.000 records, 20 attributes and one target attribute are used. The synthetic datasets are generated using benchmark functions from the `Deap` Python Package [16] and are often used in optimization. The functions are *Ackley*, *Schaffer*, *Schwefel*, *Rastrigin*, *H1*, *Rosenbrock*, *Himmelblau* and *Diffpow*. As Cluster Kriging as well as most of comparison algorithms are implemented in `Python` (except for BCM, RBCM, GRBCM and NPAE, which are available in `MATLAB` using the GPML toolbox), the experimentation is performed in `Python-2.7.5` on a `CentOS 7` platform. The parallelization mechanism of Cluster Kriging is implemented based on the `Python mpi4py` package [10]. For the BCM algorithms and the NPAE algorithm, the execution is done in `MATLAB` version `2014a`.

6.1 Hyper-parameters optimization

The hyperparameters in each local Kriging model are optimized using the Maximum Likelihood Estimation (MLE) method. As the constant trend μ is estimated using the GLS (Generalized Least Square) formula, we use the so-called profile log-likelihood as the objective function in the optimization. As for the choice of numerical optimization algorithm, we adopt a quasi-Newton method (BFGS) [15] with restarting heuristic. Each of the Kriging approximation algorithms has a hyper-parameter that can be tuned by the user to define the number of data points, clusters or inducing points, basically defining the trade-off between complexity and accuracy. For each of the algorithms a wide range of these hyper-parameters are used to see the effect and make a fair comparison between the different algorithms. The overlap for each of the Fuzzy algorithms is set to 10%, since from empirical experience we know that 10% works well. Although higher percentages (above 10%) usually increase accuracy, the increase of accuracy is not significant and costs additional training time as well. For the Model Tree variant, the number of leaves is enforced by setting a minimum number of data points per leaf and an optional maximum number of leaves. For the *Concrete Strength* dataset and all synthetic datasets: FITC is set to a range of inducing points starting from 32 and increasing in powers of 2 to 512. SoD is set to the same range as FITC but for SoD this means the number of data points. (GR)BCM, (G)PoE, NPAE and all *Cluster Kriging* variants are set to a range from 2 to 32 clusters, increasing with powers of 2. For the *Combined Cycle Power Plant* dataset: FITC is set to a range of inducing points starting from 64 and increasing in powers of 2 to 1024. *SoD* is set to the a range from 256 to 4092 data points. (GR)BCM, (G)PoE, NPAE and all *Cluster Kriging* variants are set to a range from 4 to 64 clusters.

Finally, for the *SARCOS* dataset, the range of FITC's inducing points stays the same as for the CCPP dataset, for SoD the range is from 512 to 8184 data points, and for all cluster based algorithms and the model tree variant, the range is set from 8 to 128 clusters.

6.2 Quality measurements

The quality of the experiments is estimated with the help of 5-fold cross validation, except of the *SARCOS* dataset, which uses its predefined test set. The experiments are performed in a test framework similar to the framework proposed by Chalupka et al. [7], i.e. several quality measurements are used to evaluate the performance of each algorithm. The *Coefficient of determination* R^2 score, *Mean Standardized Log Loss* (MSLL) (see [31] Chapter 8.1) and the *Standardized Mean Squared Error* (SMSE) are measured for each test run. The *Mean Standardized Log Loss* is a measurement that takes both the predicted mean and the predicted variance into account. Penalizing wrong predictions that have a small predicted variance more than wrong predictions with a large variance. Given a validation set of size n, MSLL is calculated as follows:

$$\text{MSLL} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\log\left(\pi\sigma_i + (y_i - \hat{y}_i)^2/\sigma_i\right) - triv_i,$$

where σ^t is the predicted variance for data point $\mathbf{x}_i$ and $\hat{y}_i$ is the prediction. With $triv_i$ the trivial score simulating a predictor that predicts the overall mean and standard deviation:

$$triv_i = \frac{1}{2}\log\left(\pi\sigma_y + (y_i - \bar{y})^2/\sigma_y\right)$$

For MSLL and SMSE lower scores are better, for R^2, 1.0 is the best possible score meaning a perfect fit and everything lower is worse.

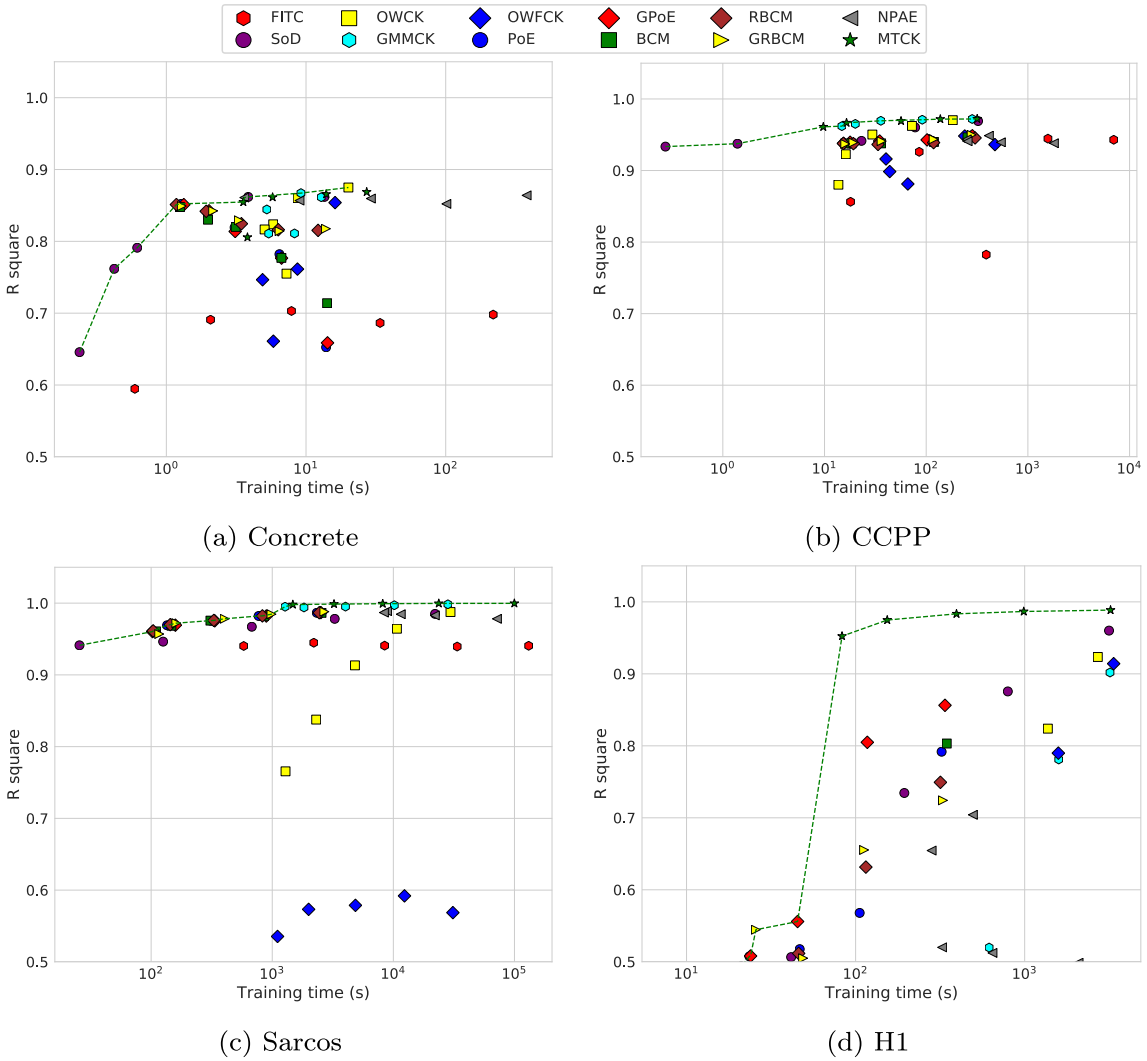


Fig. 3 Quality measurements of each algorithm with increasing sample sizes for SoD, increasing inducing points for FITC, and decreasing cluster sizes for the cluster based algorithms as explained in Section 6.1. The results are shown for the Concrete, CCPP and Sarcos datasets and the synthetic dataset of the H1 function. The training time is given on the x axis and the R^2 score on the y axis. The dashed green line indicates the non-dominated set

Table 1 Average R^2 score per dataset for each algorithm

Alg.	concrete	CCPP	sarcos	ackley	schaffer	schwefel	rast	h1	rosen.	himmel.	diffpow
OWCK	0.826	0.937	0.894	0.957	0.388	0.973	0.947	−0.082	0.997	0.995	0.991
GMMCK	0.839	**0.968**	0.996	0.951	0.369	0.977	0.948	0.527	0.997	0.995	0.991
OWFCK	0.696	0.916	0.570	0.954	0.406	0.947	0.932	−1.125	0.981	0.981	0.975
MTCK	0.851	**0.968**	**0.999**	**0.981**	**0.672**	**0.999**	**0.998**	**0.977**	**1.000**	**1.000**	**1.000**
SOD	0.784	0.948	0.964	0.952	0.321	0.990	0.973	0.676	0.999	0.997	0.995
FITC	0.675	0.890	0.941	0.260	0.208	0.006	0.322	0.165	0.000	0.291	0.001
PoE	0.789	0.942	0.975	0.943	0.498	0.924	0.532	0.576	**1.000**	**1.000**	0.997
GPoE	0.789	0.942	0.975	0.943	0.477	0.945	0.570	0.643	**1.000**	**1.000**	0.998
BCM	0.798	0.940	0.975	0.933	0.238	0.836	0.533	0.220	**1.000**	**1.000**	0.998
RBCM	0.830	0.939	0.975	0.934	0.363	0.909	0.557	0.465	**1.000**	**1.000**	0.998
GRBCM	0.831	0.942	0.975	0.942	0.491	0.941	0.528	0.583	0.160	0.160	0.998
NPAE	**0.859**	0.942	0.984	0.944	0.521	0.945	0.589	0.578	0.189	0.000	0.998

Table 2 Average SMSE score per dataset for each algorithm

Alg.	concrete	CCPP	sarcos	ackley	schaffer	schwefel	rast	h1	rosen.	himmel.	diffpow
OWCK	0.174	0.063	0.106	0.043	0.612	0.027	0.053	1.082	0.003	0.005	0.009
GMMCK	0.161	**0.032**	0.004	0.049	0.631	0.023	0.052	0.473	0.003	0.005	0.009
OWFCK	0.304	0.084	0.430	0.046	0.594	0.053	0.068	2.125	0.019	0.019	0.025
MTCK	**0.149**	**0.032**	**0.001**	**0.019**	**0.328**	**0.001**	**0.002**	**0.023**	**0.000**	**0.000**	**0.000**
SOD	0.216	0.052	0.036	0.048	0.679	0.010	0.027	0.324	0.001	0.003	0.005
FITC	0.325	0.110	0.059	0.740	0.792	0.994	0.678	0.835	1.000	0.709	0.999
PoE	0.452	0.240	0.156	0.238	0.684	0.134	0.684	0.626	**0.000**	**0.000**	0.046
GPoE	0.453	0.241	0.155	0.238	0.706	0.105	0.632	0.531	**0.000**	**0.000**	0.040
BCM	0.446	0.245	0.156	0.259	0.861	0.216	0.674	0.880	**0.000**	**0.000**	0.041
RBCM	0.411	0.246	0.154	0.258	0.764	0.137	0.653	0.683	**0.000**	**0.000**	0.044
GRBCM	0.410	0.240	0.150	0.240	0.688	0.102	0.687	0.608	8.400	8.400	0.042
NPAE	0.375	0.240	0.124	0.236	0.657	0.095	0.604	0.623	15.039	10.771	0.037

6.3 Results

The results of the experimentation are two-fold. For all datasets, the comparison is conducted with an increasing size of the training set for SoD, increasing inducing points for FITC, and a decreasing number of clusters for all cluster based algorithms, as explained in detail in Section 6.1. This in order to investigate how the performance of each algorithm is scaled with respect to the size of the training data (for each Kriging model). The result is illustrated in Fig. 3, where two performance measures, training time (measured in CPU time in seconds) and the goodness of fit (measured in the cross-validated coefficient of determination R^2) are shown for each pair of algorithm and dataset. Naturally, CPU time and R^2 form two conflicting objectives in the sense that it is difficult to improve one without making the other deteriorate. Thus, a good trade-off between those two performance measures is of particular interest here. In the figure, the so-called Pareto efficient frontier is drawn in green dashed lines, which contains points that are superior in both CPU time and R^2 to the points that are not on the front. It is clear that the MTCK algorithm is constantly on the Pareto front for all four datasets. On the other hand, SoD frequently dominates the other algorithms. Although it yields a relatively poor R^2 score, the training time of SoD is also very small, compared to the other algorithms.

For both real-world and synthetic datasets, the R^2 scores averaged from cross-validation, are arranged in Table 1. The best results for each dataset are marked by bold face. It is clear that the MTCK algorithm outperforms in most scenarios while the BCM algorithms are also competitive on *Rosenbrock* and *Himmelblau* problems. In addition, the MSLL scores are provided in Table 3. Here, a similar

Table 3 Average MSLL score per dataset for each algorithm

Alg.	concrete	CCPP	sarcos	ackley	schaffer	schwefel	rast	h1	rosen.	himmel.	diffpow
OWCK	−0.946	−1.438	−1.371	−1.516	−0.073	−2.013	−1.686	−0.276	−2.915	−2.646	−2.548
GMMCK	−1.100	−1.525	−3.147	−1.517	0.081	−2.162	−1.807	−0.540	−3.074	−2.790	−2.666
OWFCK	−0.692	−1.109	−0.302	−1.462	−0.091	−1.944	−1.642	−0.060	−2.738	−2.553	−2.438
MTCK	−1.140	−1.193	**−3.429**	**−2.012**	−0.514	−3.278	**−2.901**	**−1.967**	−4.054	**−3.739**	**−3.744**
SOD	−0.837	−0.089	−1.926	−1.622	0.477	−2.554	−2.179	−0.766	−3.479	−3.204	−3.020
FITC	−0.629	−1.165	−1.463	−0.104	−0.107	−0.002	−0.193	−0.059	*	−0.193	*
PoE	5.822	3.083	15.953	2.303	3.883	−1.728	4.102	3.663	*	*	−0.130
GPoE	−0.797	−1.424	−1.895	−1.429	−0.403	−3.291	−0.586	−1.100	−13.260	*	−3.349
BCM	2.637	3.047	15.944	2.750	3.154	−0.821	3.019	3.788	**−13.793**	*	−0.295
RBCM	1.591	7.294	40.446	2.983	0.313	−0.388	4.708	−0.198	−11.576	*	13.613
GRBCM	−0.673	**−1.426**	−1.739	−1.417	−0.456	−1.736	−0.375	0.485	6.311	6.411	0.731
NPAE	−0.986	−1.425	−2.056	−1.441	**−0.555**	**−3.681**	−0.721	−0.634	0.987	2.734	−3.351

A * indicates that the value was way above 1000 (very bad fit)

comparison can be observed as in Table 1. Finally, the SMSE scores are presented in Table 2.

6.4 Parameter setting recommendations

To use the Cluster Kriging algorithms, the minimum cluster size or the number of clusters has to be set as a user defined parameter. It is recommended to set this parameter in such a way that each individual cluster contains between 100 and 1000 records. 1000 records is still computationally tractable by Kriging in terms of execution time and 100 records is in most cases still doable in terms of fitting the Kriging model. Selecting smaller cluster sizes is likely to result in poorly fitted models and selecting cluster sizes larger than 1000 will in most cases not increase accuracy but will only increase execution time. These recommendations are purely based on empirical observations and depend highly on the dataset one is working with. For MTCK smaller cluster sizes are usually still fine because of the low variance in the records per leaf due to the splitting criterion of the Regression Tree (Table 3).

7 Conclusions and further research

In this paper, a novel Kriging framework, Cluster Kriging, is proposed to reduce the time and space complexity of the Kriging method if it were trained directly on a large dataset. Essentially, Cluster Kriging combines smaller Kriging models that are trained on partitions of the whole dataset. Four different algorithms using this methodology are proposed and described in details. A broad comparison between the novel algorithms and other state of the art Kriging approximation algorithms is done. The results of the experiments (as given in Section 6) clearly show that for each dataset, the *Gaussian Mixture Models Cluster Kriging* (GMMCK) and the Model Tree Cluster Kriging (MTCK) outperform the other algorithms in all measurements. It can also be observed that the *Bayesian Committee Machine* algorithms, are unstable when the number of clusters is above 8. This is most likely due to the poor recombination of models with different hyper-parameters and the chance of poor fitting of one of the clusters. In terms of training time, *Subset of Data* is faster than any of the other algorithms when setting the number of data points small. However, it pays for this complexity reduction by a decrease in accuracy. Both for SoD and FITC, the training time increases faster with the number of data points than the training time of the cluster based algorithms. For the GMMCK algorithm, it is shown that the membership probabilities of the Gaussian Mixture Model can be used as weights in the combination of the various Kriging models' predictions. It is also shown that a Model Tree of Kriging models works very well in high dimensional problems and requires less prediction time due to the fact that only one Kriging model is used to predict an unseen data point.

For future research it would be interesting to automatically determine optimal cluster sizes for the different algorithms and optimize the nugget parameter of the Kriging models. The nugget plays an important role in the fitting of the Kriging model as it determines the amount of marginalization.

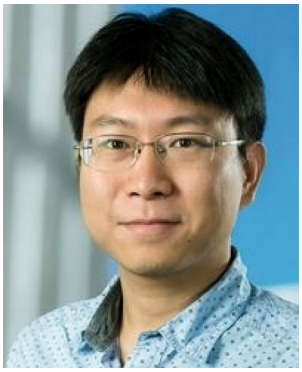
Acknowledgements The authors acknowledge support by NWO (Netherlands Organisation for Scientific Research) PROMIMOOC project (project number: 650.002.001).

References

1. Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
2. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140. https://doi.org/10.1007/BF00058655
3. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
4. Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC press
5. Buhmann MD (2004) Radial basis functions: theory and implementations. Cambridge Monogr Appl Comput Math 12:147–165
6. Cao Y, Fleet DJ (2014) Generalized product of experts for automatic and principled fusion of gaussian process predictions. arXiv:1410.7827
7. Chalupka K, Williams CKI, Murray I (2013) A framework for evaluating approximation methods for gaussian process regression. J Mach Learn Res 14(1):333–350. http://dl.acm.org/citation.cfm?id=2502581.2502592
8. Chen T, Ren J (2009) Bagging for gaussian process regression. Neurocomput 72(7-9):1605–1610
9. Csató L, Opper M (2002) Sparse on-line gaussian processes. Neural Comput 14(3):641–668
10. Dalcín L, Paz R, Storti M (2005) Mpi for python. J Parallel Distrib Comput 65(9):1108–1115. https://doi.org/10.1016/j.jpdc.2005.03.010. http://www.sciencedirect.com/science/article/pii/S0743731505000560
11. D'Ambrosio A, Aria M, Siciliano R (2012) Accurate tree-based missing data imputation and data fusion within the statistical learning paradigm. J Classif 29(2):227–258
12. Ding S, Cong L, Hu Q, Jia H, Shi Z (2019) A multiway p-spectral clustering algorithm. Knowl-Based Syst 164:371–377. https://doi.org/10.1016/j.knosys.2018.11.007. http://www.sciencedirect.com/science/article/pii/S0950705118305434
13. Du M, Ding S, Xue Y, Shi Z (2019) A novel density peaks clustering with sensitivity of local density and density-adaptive metric. Knowl Inf Syst 59(2):285–309. https://doi.org/10.1007/s10115-018-1189-7
14. Dunn JC (1973) A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J Cybern 3(3):32–57. https://doi.org/10.1080/01969727308546046

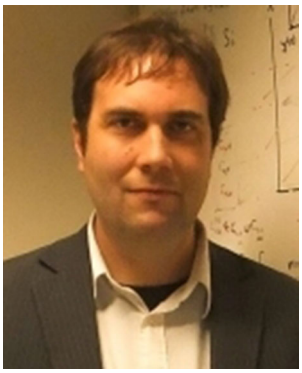
15. Fletcher R (2013) Practical methods of optimization, Wiley, New York
16. Fortin F, Michel F, Gardner MA, Parizeau M, Gagné C (2012) DEAP: Evolutionary algorithms made easy. J Mach Learn Res 13:2171–2175
17. Ginsbourger D, Le Riche R, Carraro L (2010) Kriging is well-suited to parallelize optimization. In: Computational intelligence in expensive optimization problems, pp 131–162. Springer
18. Hamzaoui Y, Amnai M, Choukri A, Fakhri Y (2018) Novel clustering method based on k-medoids and mobility metric. IJIMAI 5(1):29–33. https://doi.org/10.9781/ijimai.2017.11.001
19. Hartman L, Hössjer O (2008) Fast Kriging of large datasets with Gaussian Markov random fields. Comput Stat Data Anal 52(5):2331–2349. 10.1016/j.csda.2007.09.018
20. Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Comput 14(8):1771–1800
21. Huang G, Wang DH, Lan Y (2011) Extreme learning machines: A survey. Int J Mach Learn Cybern 2(2):107–122. https://doi.org/10.1007/s13042-011-0019-y
22. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13(4):455–492
23. Kaya H, Tùfekci P, Gùrgen SF (2012) Local and Global Learning Methods for Predicting Power of a Combined Gas & Steam Turbine. International Conference on Emerging Trends in Computer and Electronics Engineering (ICETCEE) 2012:13–18
24. Kleijnen JP (2009) Kriging metamodeling in simulation: a review. Eur J Oper Res 192(3):707–716
25. Landwehr N, Hall M, Frank E (2005) Logistic model trees. Mach Learn 59(1-2):161–205
26. Lawrence ND (2004) Gaussian process latent variable models for visualisation of high dimensional data. Adv Neural Inf Proces Syst 16(3):329–336
27. Liu H, Cai J, Wang Y, Ong YS (2018) Generalized robust bayesian committee machine for large-scale gaussian process regression. arXiv:1806.00720
28. Naish-Guzman A, Holden S (2007) The generalized FITC approximation. In: Advances in neural information processing systems, pp 1057–1064
29. Nguyen-Tuong D, Seeger M, Peters J (2009) Model learning with local Gaussian process regression. Adv Robot 23(15):2015–2034. https://doi.org/10.1163/016918609X12529286896877
30. Quiñonero-Candela J, Rasmussen CE (2005) A unifying view of sparse approximate gaussian process regression. J Mach Learn Res 6(1):1939–1959
31. Rasmussen C, Williams C (2006) Gaussian Processes for Machine Learning. Adaptive computation and machine learning series. University Press Group Limited
32. Reynolds D (2009) Gaussian mixture models. In: Encyclopedia of Biometrics, pp. 659–663. Springer
33. Rullière D, Durrande N, Bachoc F, Chevalier C (2018) Nested kriging predictions for datasets with a large number of observations. Stat Comput 28(4):849–867
34. Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. Statistical science pp 409–423
35. Silverman BW (1985) Some aspects of the spline smoothing approach to non-parametric regression curve fitting. J R Stat Soc Ser B Methodol 47(1):1–52
36. Simpson TW, Mauery TM, Korte JJ, Mistree F (2001) Kriging models for global approximation in simulation-based multidisciplinary design optimization. AIAA J 39(12):2233–2241
37. Snelson E, Ghahramani Z (2005) Sparse gaussian processes using pseudo-inputs. In: Advances in neural information processing systems, pp 1257–1264
38. van Stein B, Wang H, Kowalczyk W, Bäck T, Emmerich M (2015) Optimally weighted cluster kriging for big data regression. In: Advances in intelligent data analysis XIV, pp 310–321. Springer
39. van Stein B, Wang H, Kowalczyk W, Emmerich M, Bäck T (2016) Fuzzy clustering for optimally weighted cluster kriging. In: 2016 IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 939–945. IEEE
40. Stein ML (1999) Interpolation of spatial data: some theory for Kriging. Springer Science & Business Media
41. Torgo L (1997) Functional models for regression tree leaves. In: ICML, vol 97, pp 385–393. Citeseer
42. Tresp V (2000) A Bayesian committee machine. Neural Comput 12(11):2719–2741. https://doi.org/10.1162/089976600300014908
43. Vapnik V (2013) The nature of statistical learning theory. Springer Science & Business Media
44. Vijayakumar S, D'souza A, Schaal S (2005) Incremental online learning in high dimensions. Neural Comput 17(12):2602–2634
45. Wang Y, Witten IH (1996) Induction of model trees for predicting continuous classes. Department of Computer Science, University of Waikato
46. Xu X, Ding S, Xu H, Liao H, Xue Y (2019) A feasible density peaks clustering algorithm with a merging strategy. Soft Comput 23(13):5171–5183. https://doi.org/10.1007/s00500-018-3183-0
47. Yeh IC (1998) Modeling of strength of high-performance concrete using artificial neural networks. Cem Concr Res 28(12):1797–1808

Bas van Stein born 1989 in Sassenheim, The Netherlands, received his Masters degree in computer science in 2013 at the Leiden Institute of Advanced Computer Science (LIACS), Leiden University and his Ph.D. degree in 2018 in data mining and machine learning on industrial applications. His current research interests are algorithm design and improvements for data mining and machine learning as well as automated machine learning on deep neural networks.



Hao Wang born 1989 in Baoji, China, received Bachelor degree in computer science in the Beijing Language and Culture University (BLCU), Beijng, China, in 2011. He received his Ph.D. degree in 2018 from the Leiden Institute of Advanced Computer Science (LIACS), Leiden University. His current research interests are algorithm improvement for evolutionary strategy as well as statistical machine learning for real-world data classification, regression and prediction.

Wojtek Kowalczyk born in 1957 in Poland, received his MSc and Ph.D. degrees in Mathematics from University of Warsaw in 1980 and 1985, resp., and is currently an Assistant Professor at Leiden Institute of Advanced Computer Science (LIACS), Leiden University. His research concentrates on techniques for finding anomalies in massive data sets or streams and applying these techniques to real-life problems like fraud detection, cybersecurity, health care, industrial processes. He led numerous data mining projects for banks, insurance companies and governmental organizations.



Dr. Michael Emmerich is Associate Professor at LIACS, Leiden University and leader of the Multicriteria Optimization and Decision Analysis research group. He was born in 1973 in Coesfeld (Germany) and received his doctorate in 2005 from Dortmund University (H.-P. Schwefel promoter). He carried out projects as a researcher at ICD e.V. (Germany), IST Lisbon, University of the Algarve (Portugal), ACCESS Material Science e.V. (Germany), and the FOM/AMOLF institute on Fundamental Science of Matter (Netherlands). He is known for pioneering work on Gaussian-process models and indicator based multiobjective optimization, and has edited 4 books, and co-authored more than 120 papers in multicriteria optimization algorithms and their application in computational chemistry and engineering.

Thomas Bäck, born 1963 in Freiburg/Elbe, Germany, is Professor for Natural Computing at the Leiden Institute for Advanced Computer Science (LIACS) at Leiden University, Netherlands, since 2002. Prior to this, he was Associate Professor at Leiden University (1996 – 2002), department leader at the Center for Applied Systems Analysis in Dortmund, Germany (1994 – 2002), and CTO of NuTech Solutions Inc., Charlotte, NC (1999 – 2009). Thomas received his Diploma degree in Computer Science in 1990 and his Ph.D. degree in 1994, both from the University of Dortmund, Germany. In 2015, Thomas received the IEEE Evolutionary Computation Pioneer Award from the Computational Intelligence Society.

Thomas' current research interests include developing advanced evolutionary algorithms for optimization with a limited budget of objective function evaluations, the application of evolutionary algorithms to industrial problems, the combination of data mining and optimization, experimental optimization, drug design by evolutionary computation, and applications such as industrial process data mining and optimization. Thomas has more than 300 publications, including a monograph entitled Evolutionary Algorithms in Theory and Practice (OUP, 1996), a recent book on Contemporary Evolution Strategies (Springer, 2013), 2 handbooks, and 8 edited conference proceedings. He is associate editor of the Natural Computing book series and co-editor of the Handbook of Natural Computing (2013) as well as the Handbook of Evolutionary Computation (1997).