Data driven modeling & optimization of industrial processes

Stein, B. van

5

Cluster Kriging

Data driven models and model-driven optimization are two of the most important parts of the data driven framework (Section 2.3). *Kriging* or *Gaussian Process Regression* is a popular kernel based data driven regression model. Kriging is capable of achieving high accuracy predictions and providing, in addition, the predicted variance, also known as the Kriging variance. The Kriging variance is used in many applications as the uncertainty or quality measure of a prediction and is heavily exploited in surrogate model-based optimization.

However, the computational and space complexity of Kriging, that is cubic and quadratic in the number of data points respectively, is a major bottleneck, especially with the quantities of data available for the PROMIMOOC project. To address these issues, Cluster Kriging is proposed as a solution for the time and space complexity bottleneck. In addition, Cluster Kriging can be used as the surrogate model for *Efficient Global Optimization* to allow model-driven optimization on big data.

In this chapter, a general methodology for the complexity reduction of Kriging, called *Cluster Kriging*, is proposed. The main principle of Cluster Kriging is that 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. It is also shown that Cluster Kriging can be used with the popular model-based optimization framework; Efficient Global Optimization. According to the empirical study, the proposed algorithms consistently outperform the existing algorithms. Moreover, some practical suggestions are provided for using the proposed algorithms.
The content of this chapter is primarily based on the publications [53, 54, 55, 56].

5.1 Introduction

Kriging, or *Gaussian Process Regression* [57] 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 data sets, for prediction purposes and for *Surrogate model-based optimization* [58]. 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* [59] or *Gradient Boosted Decision Trees* lack the advantage of interpretation [60] but have more expressive power. Another method is *Linear Model Trees* [61], 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* [62], 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* [63] and *Radial Basis Functions* [64]. 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 $O(n^3)$ and $O(n^2)$, respectively. Where n denotes the number of points. To overcome this complexity problem, Kriging approximation algorithms such as [65] and [66] are introduced. Unfortunately, these approximation algorithms are usually less accurate than the original Kriging algorithm.

An overview of Kriging approximation methods is presented and a novel divide and conquer based approach, *Cluster Kriging* (CK), is introduced. The novel Cluster Kriging framework contains three steps, Partitioning, Modeling and Predicting. Each of the steps can be implemented using a wide range of approaches, which are explained in detail in this chapter. Using these approaches four algorithms are implemented and compared against each other and the state of the art. One particular interesting and novel algorithm that uses the Cluster Kriging

methodology is the proposed **Model Tree Cluster Kriging** (MTCK). MTCK uses a regression tree with a specified number of leaf nodes to partition the data in the objective space. A Kriging model is then built on each partition defined by the tree's leaves. MTCK uses only one of the trained Kriging models per unseen record to predict, depending on which leaf node the unseen record is assigned to. The proposed algorithms are evaluated and compared to several state-of-the-art alternative Kriging approximation algorithms. A well-defined testing framework for Kriging approximation algorithms [67] is adopted for the comparisons.

In addition to the Cluster Kriging varients proposed, it is also shown that Cluster Kriging can be integrated in the Efficient Global Optimization framework. Enabling efficient global optimization for large data sets.

5.2 Kriging

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 [68, 69]. 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) [70, 71]. The model features in providing the theoretical uncertainty measurement of estimations.

When the 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 chapter, 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 $\mathcal{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\} \subseteq \mathbb{R}^d$ and the corresponding output variables are represented as $\mathbf{y} = [y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(2)}), \ldots, y(\mathbf{x}^{(n)})]^\top$. 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 output variables are considered as the "noisy" observation of f, that is perturbed by a Gaussian random noise γ:

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

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

Note that the noises (error terms) γ are assumed to be homoscedastic (identically distributed) and independent, both from each other and the Gaussian Process 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 [57]. It can be completely specified by providing a covariance function $k(\cdot, \cdot)$ to calculate the pairwise covariance:

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

The covariance function $k(\cdot, \cdot)$ is a kernel function performing the so-called "kernel trick", which computes the inner product on the feature space as a function in the input space. Consequently, 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):

$$k(\mathbf{x}, \mathbf{x}') = \sigma_\varepsilon^2 \prod_{i=1}^{d} \exp\left(-\theta_i (x_i - x_i')^2\right), \tag{5.1}$$

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. By using the Gaussian kernel, the resulting Gaussian process is stationary in the sense that the process variance is constant: $\forall \mathbf{x} \in \mathbb{R}^d, k(\mathbf{x}, \mathbf{x}) = \sigma_\varepsilon^2$.

To infer output value $y^{(t)} = y(\mathbf{x}^{(t)})$ 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 data set $\mathcal{X}$, $\mathbf{x}^{(t)}$ and the unknown prior mean μ. Such a joint distribution is a

multivariate Gaussian and is expressed as follows;

$$\begin{bmatrix} y^{(t)} \\ \mathbf{y} \end{bmatrix} \mid \mathcal{X} \sim \mathcal{N}\left(\mu \mathbf{1}_{n+1}, \begin{bmatrix} \sigma_\varepsilon^2 + \sigma_\gamma^2 & \mathbf{c}^\top \\ \mathbf{c} & \mathbf{\Sigma} + \sigma_\gamma^2 \mathbf{I} \end{bmatrix}\right), \tag{5.2}$$

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

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 maximum likelihood method. The *posterior* distribution of $y^{(t)}$ can be calculated by marginalizing μ out and conditioning on the observed output variables $\mathbf{y}$ [57]. Without any derivations, the posterior distribution for Ordinary Kriging is again Gaussian [72]:

$$y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)} \sim \mathcal{N}\left(m(\mathbf{x}^{(t)}), s^2(\mathbf{x}^{(t)})\right) \tag{5.3}$$

where the posterior mean and variance are expressed as:

$$m(\mathbf{x}^{(t)}) = \hat{\mu} + \mathbf{c}^\top \left(\mathbf{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} (\mathbf{y} - \hat{\mu}\mathbf{1}_n) \tag{5.4}$$

$$\begin{aligned} s^2(\mathbf{x}^{(t)}) = \sigma_\gamma^2 + \sigma_\varepsilon^2 - \mathbf{c}^\top \left(\mathbf{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{c} \\ + \frac{(1 - \mathbf{c}^\top \left(\mathbf{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{1}_n)^2}{\mathbf{1}_n^\top \left(\mathbf{\Sigma} + \sigma_\gamma^2 \mathbf{I}\right)^{-1} \mathbf{1}_n} \end{aligned} \tag{5.5}$$

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

Note that the estimation of the trend, $\hat{\mu}$ is obtained by maximum a posteriori principle (MAP). The posterior mean function (Eq. 5.4) is used as the predictor while the posterior variance (Eq. 5.5) is the so-called *Kriging variance* that measures the uncertainty of the prediction.

5.3 Relevant Research

Despite the theoretically sound development of the Kriging model, it suffers several issues when applied to large data sets. The major bottleneck is the high time

and memory complexity of the model fitting process: The inverse of the covariance matrix $\mathbf{\Sigma}^{-1}$ needs to be computed for both the posterior mean and variance (Eq. 5.4 and 5.5), 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 $\mathbf{\Sigma}^{-1}$, resulting in a $O(n^3)$ computational cost per each optimization iteration. Thus, for a large data set, 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 [57]. The contributions towards solving this issue can be roughly split into three categories:

Subset Methods

The first category of approximation algorithms uses only a subset of the complete data set to approximate a full Kriging model. The idea behind these methods is to get a realistic representation of the complete data set 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 data set.

Subset of Data (SoD) [73] 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) [74] 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. As noted also in [75], there are only m (number of basis points) degrees of freedom in the model because the model is degenerate (finite linear-in-the-parameters), which might be too restrictive.

[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.

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) [76] 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 [66] 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) [77, 78]. 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.

Divide and Conquer Methods

The last category contains methods that divide a (big) data set into several smaller data sets and build a model for each of them. How to split the data set into

smaller data sets 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) [65] is an algorithm similar to the one 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.

Several other attempts have been made to divide the Kriging model in sub-models [79, 80], each solution for different domains. In [79], a *Bagging* [81] method is proposed to increase the robustness of the Kriging algorithm, rather than speeding up the algorithm's training time. In [80], 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, three state of the art algorithms: *SoD*, *FITC* and *BCM* are selected to compare with the proposed approaches in this paper, as they seem to be the mostly used in their category.

5.4 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 data set is partitioned into clusters of similar sizes, Cluster Kriging will reduce the time complexity by a factor of k^2 resulting in $k \cdot \left(\frac{n}{k}\right)^3$ (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 clusters 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 Kriging model on clusters can also be combined in different manners. By varying the options in each step of the cluster Kriging, 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.

5.4.1 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 data sets. In general, the goal is to obtain a set $\mathcal{S}$ containing k clusters on the input data set $\mathcal{X}$.

$$\mathcal{S} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k\}, \quad \text{where } \bigcup_{i=1}^{k} \mathcal{X}_i = \mathcal{X}. \tag{5.6}$$

In addition, the output values $\mathbf{y}$ are also grouped according to the clustering of $\mathcal{X}$: $\mathbf{y} = [\mathbf{y}_1^\top, \mathbf{y}_2^\top, \ldots, \mathbf{y}_k^\top]^\top$. The clustering can be done in many ways, with the

most simple and feasible approach being random clustering. For our framework however we introduce three more sophisticated partitioning methods that are used in the experiments later on.

Hard Clustering

The hard clustering splits the data into k smaller *disjoint* data sets:

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

This can be achieved by various methods, for instance the K-means algorithm (Eq. 5.7). K-means clustering minimizes the within-cluster sum of squares, that is expressed as:

$$\arg\min_{\mathrm{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{X}_i} ||\mathbf{x} - \boldsymbol{\mu}^{(i)}||^2, \tag{5.7}$$

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

Fuzzy Clustering

Instead of using a hard clustering approach, a fuzzy clustering algorithm can be used to introduce slight overlap between the various smaller data sets, 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 (Eq. 5.8) 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 section the *Fuzzy C-Means* (FCM) [82] clustering algorithm and the *Gaussian Mixture Models* (GMM) [83] 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 Eq. 5.8 iteratively. In each iteration, the membership coefficients of each point being in the clusters are computed using Eq. 5.9. 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:

$$\sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}^{m}||\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)}||^2, \tag{5.8}$$

where w_{ij} are the membership values (see Eq. 5.9) and m is the so-called fuzzifier (set to 2 in this chapter). The fuzzifier determines the level of cluster fuzziness as follows:

$$w_{ij}^{m} = \frac{1}{\sum_{c=1}^{k}\left(\frac{||\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)}||}{||\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(c)}||}\right)^{\frac{2}{m-1}}} \tag{5.9}$$

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.

Regression Tree Partitioning

The third method used is the partitioning by use of a Regression Tree [84] 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 records 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.

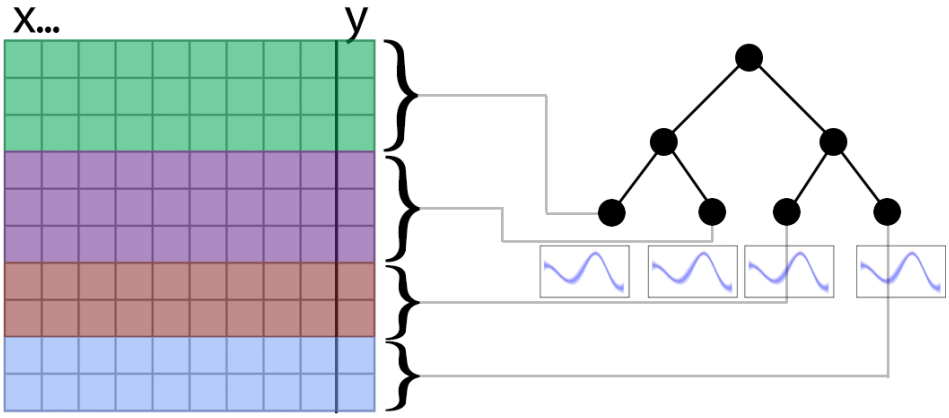


Figure 5.1: Visualisation of a Model Tree. The top node is the root and the bottom nodes are the leaves with attached models. Each record in the data (on the left) is assigned to a leaf node of the regression tree.

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.

5.4.2 Modeling

After partitioning the data set into several clusters, Kriging models are fitted on each of the smaller data sets. 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 chapter, the kernel functions used on each cluster to be the same. As for the regression tree approach, the data set, 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. 5.1). A similar technique is introduced in the context of combining linear regression models [85, 86, 61]. In general, the predictive (posterior) distribution of the target variable $y^{(t)}$ on each cluster is:

$$y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)} \sim \mathbf{N}\left(m_l(\mathbf{x}^{(t)}), \sigma_l^2(\mathbf{x}^{(t)})\right), \quad l = 1, \ldots, k, \tag{5.10}$$

where m_l and σ_l^2 are specified again by Eq. 5.4 and 5.5 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.

5.4.3 Prediction

After training 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 way of predicting the unseen data points 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.

Single Model Prediction

The most straightforward method which can be used to predict unseen data points is by using only one of the local Kriging models. 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 (Figure 5.1). 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. A possible disadvantage of this method is that you might loose the global information of the target function and that predictions close to the cluster borders may be less reliable. In Figure 5.2, a visualisation of predictions from a fitted Cluster Kriging model using regression trees is shown, marking the intersections between the different local models with 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 choosen in such a way that it balances the cluster sizes, using variance reduction as the splitting criterion generally gives better fitted local models.

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 data set is separated by hard clustering methods, the Gaussian processes built on different clusters 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 "trust" should be put on the prediction from each cluster. The

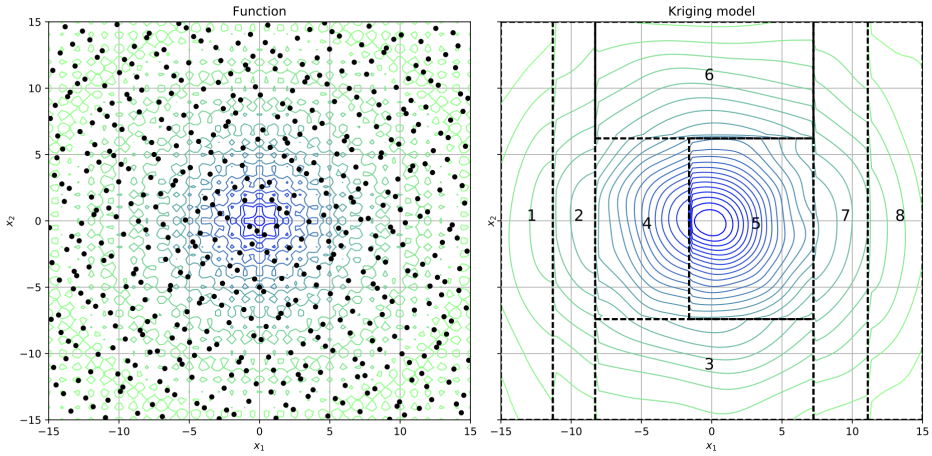


Figure 5.2: The landscape of the two dimensional Ackley function on the left, and on the right is the contours of the Model Tree Cluster Kriging mean function, with the tree partitioning visualized by dashed lines. The index of the clusters are shown in the middle of each rectangle.

weighted superposition of all Gaussian processes is [55]:

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

The overall prediction and its variance depend on the weights used in the equation above. Intuitively, the optimal prediction is achieved when the variance of the estimation is minimal. To obtain such an optimal predictor, the overall Kriging variance should be optimized with respect to the weights, resulting in the following optimization task:

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

The optimal weights are obtained by solving the problem above (see [55, 54] for details):

$$w_l^* = \frac{1/\sigma_l^2(\mathbf{x}^{(t)})}{\sum_{i=1}^{k} 1/\sigma_i^2(\mathbf{x}^{(t)})}. \tag{5.11}$$

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

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:

$$w_l = \Pr(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}), \quad \text{for } l = 1, \ldots, k \tag{5.12}$$

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 data set $\mathcal{X}$, 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 [54]:

$$\begin{aligned} & p(y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}) \\ &= \sum_{l=1}^{k} p(y^{(t)}, C = l \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}) \\ &\approx \sum_{l=1}^{k} p(y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}) \Pr(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}) \end{aligned} \tag{5.13}$$

The independence assumption between Gaussian process models still holds approximately when the amount of the overlap between clusters is small. Thus, the density function $g(y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)})$ *approximately* equals to Eq. 5.13. The first

term inside the sum in Eq. 5.13 is the predictive density function obtained from each cluster. The second term represents the probability of data point $\mathbf{x}^{(t)}$ belonging to a cluster, which is the weight in Eq. 5.12. 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 $p(y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)})$ is calculated:

$$\begin{aligned}
&\mathbf{E}[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}] \\
&= \sum_{l=1}^{k} \int_{-\infty}^{\infty} y^{(t)} \sum_{l=1}^{k} g(y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}) \Pr(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}) \, \mathrm{d}y^{(t)} \\
&= \sum_{l=1}^{k} \Pr(C = l \mid \mathcal{X}, \mathbf{x}^{(t)}) \mathbf{E}[y^t \mid \mathbf{X}_l, \mathbf{y}_l, \mathbf{x}^t] \\
&= \sum_{l=1}^{k} w_l m_l(\mathbf{x}^{(t)})
\end{aligned} \tag{5.14}$$

Note that $m_l(\mathbf{x}^t)$ is the mean function as shown in Eq. 5.10 and 5.14 suggests that the overall prediction made on the whole data set 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:

$$\begin{aligned}
&\mathrm{Var}[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}] \\
&= \mathbf{E}[{y^{(t)}}^2 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}] - \mathbf{E}[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}]^2 \\
&= \sum_{l=1}^{k} w_l \left(\mathrm{Var}[y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}] + \mathbf{E}[y^{(t)} \mid \mathcal{X}_l, \mathbf{y}_l, \mathbf{x}^{(t)}]^2 \right) \\
&\quad - \mathbf{E}[y^{(t)} \mid \mathcal{X}, \mathbf{y}, \mathbf{x}^{(t)}]^2 \\
&= \sum_{l=1}^{k} w_l \left(\sigma_l^2(\mathbf{x}^{(t)}) + m_l^2(\mathbf{x}^{(t)}) \right) - \left(\sum_{l=1}^{k} w_l m_l(\mathbf{x}^{(t)}) \right)^2
\end{aligned} \tag{5.15}$$

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

5.5 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 chapter 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 (Subsection 5.4.3).

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 (Subsection 5.4.3).

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 (Subsection 5.4.3).

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 (Subsection 5.4.3), 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.

5.6 Experimental Setup and Results

A broad variety of experiments is executed to compare Optimally Weighted Cluster Kriging and its Fuzzy and Model Tree variants, to a wide set of other Kriging approximation algorithms. The algorithms included in the test are; *Bayesian Committee Machines*, both with shared parameters (BCM sh.) and with individual parameters (BCM), *Subset of Data* (SoD), *Fully Independent Training Conditional* (FITC), *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 data sets from the *UCI machine learning repository* [25]:

- *Concrete Strength* [87], a data set with 1030 records, 8 attributes and one target attribute. The task is to predict the strength of concrete.
- *Combined Cycle Power Plant* (CCPP) [88], a data set of 9.568 records, 3 attributes and one target attribute. The target is the hourly electrical energy output and the task is to predict this target.
- *SARCOS* [89], a data set from *gaussianprocess.org* with a training set of 44.484 records, 21 attributes and 7 target attributes. The task is to predict the joint torques of a anthropomorphic robot arm. All 21 attributes are

used as training data but only the 1^{st} target attribute is used as target. The dataset comes with a predefined test set of 4.449 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 [90] and are often used in optimization. The functions are *Ackley*, *Schaffer*, *Schwefel*, *Rastrigin*, *H1*, *Rosenbrock*, *Himmelblau* and *Diffpow*.

5.6.1 Hyper-Parameters

The hyper-parameters in each Kriging model are set via the Maximum Likelihood Estimation (MLE). As the constant trend μ also needs to be estimated, we use the so-called *concentrated* log likelihood for the estimation. For this numerical optimization task, we adopt a quasi-Newton method (BFGS) [91] 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. BCM, both shared and non-shared versions 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 4.092 data points. BCM, both shared and

non-shared versions 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 8.184 data points, and for all cluster based algorithms and the model tree variant, the range is set from 8 to 128 clusters.

5.6.2 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, K. et al.* [67], 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 [57] Section 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.

$$MSLL = \left\langle \frac{1}{2} \cdot log(\pi\sigma_i + (y_i - \hat{y}_i)^2/\sigma_i) - triv \right\rangle$$

Where σ^t is the predicted variance for record $\mathbf{x}_i$ and $\hat{y}_i$ the predicted mean. With *triv* the trivial score simulating a predictor that predicts the overall mean and standard deviation:

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

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.

Table 5.1: Average R^2 score per dataset for each algorithm

Dataset	SOD	OWCK	GMMCK	OWFCK	FITC	BCM	BCM sh.	MTCK
Concrete	0.784	0.826	0.839	0.696	0.675	-81.888	-242.459	**0.851**
CCPP	0.948	0.937	**0.968**	0.916	0.890	0.220	-24.602	**0.968**
Sarcos	0.964	0.894	0.996	0.570	0.941	-627.280	0.448	**0.999**
Ackley	0.952	0.957	0.951	0.954	0.260	0.921	-0.039	**0.981**
Schaffer	0.321	0.388	0.369	0.406	0.208	0.452	-0.050	**0.672**
Schwefel	0.990	0.973	0.977	0.947	0.006	0.969	-0.043	**0.999**
Rast	0.973	0.947	0.948	0.932	0.322	0.914	-0.043	**0.998**
H1	0.676	-0.082	0.527	-1.125	0.165	0.657	-0.046	**0.977**
Rosenbrock	0.999	0.997	0.997	0.981	0.000	0.994	-0.050	**1.000**
Himmelblau	0.997	0.995	0.995	0.981	0.291	0.994	-0.044	**1.000**
Diffpow	0.995	0.991	0.991	0.975	0.001	-0.001	-0.001	**1.000**

Table 5.2: Average MSLL score per dataset for each algorithm

Dataset	SOD	OWCK	GMMCK	OWFCK	FITC	BCM	BCM sh.	MTCK
Concrete	-0.837	-0.946	-1.100	-0.692	-0.629	18.590	68.013	**-1.140**
CCPP	-0.089	-1.438	**-1.525**	-1.109	-1.165	7.826	69.346	-1.193
Sarcos	-1.926	-1.371	-3.147	-0.302	-1.463	780.090	507.721	**-3.429**
Ackley	-1.622	-1.516	-1.517	-1.462	-0.104	7.352	13.010	**-2.012**
Schaffer	0.477	-0.073	0.081	-0.091	-0.107	16.872	11.707	**-0.514**
Schwefel	-2.554	-2.013	-2.162	-1.944	-0.002	-0.144	12.034	**-3.278**
Rast	-2.179	-1.686	-1.807	-1.642	-0.193	4.554	11.590	**-2.901**
H1	-0.766	-0.276	-0.540	-0.060	-0.059	9.018	17.393	**-1.967**
Rosenbrock	-3.479	-2.915	-3.074	-2.738	high*	0.612	18.575	**-4.054**
Himmelblau	-3.204	-2.646	-2.790	-2.553	-0.193	-1.422	12.826	**-3.739**
Diffpow	-3.020	-2.548	-2.666	-2.438	high*	high*	high*	**-3.744**

5.6.3 Results

The results of experiments on the real world data sets *Concrete Strength*, *CCPP* and *SARCOS* are shown in Figure 5.3 and the results on the Synthetic data sets

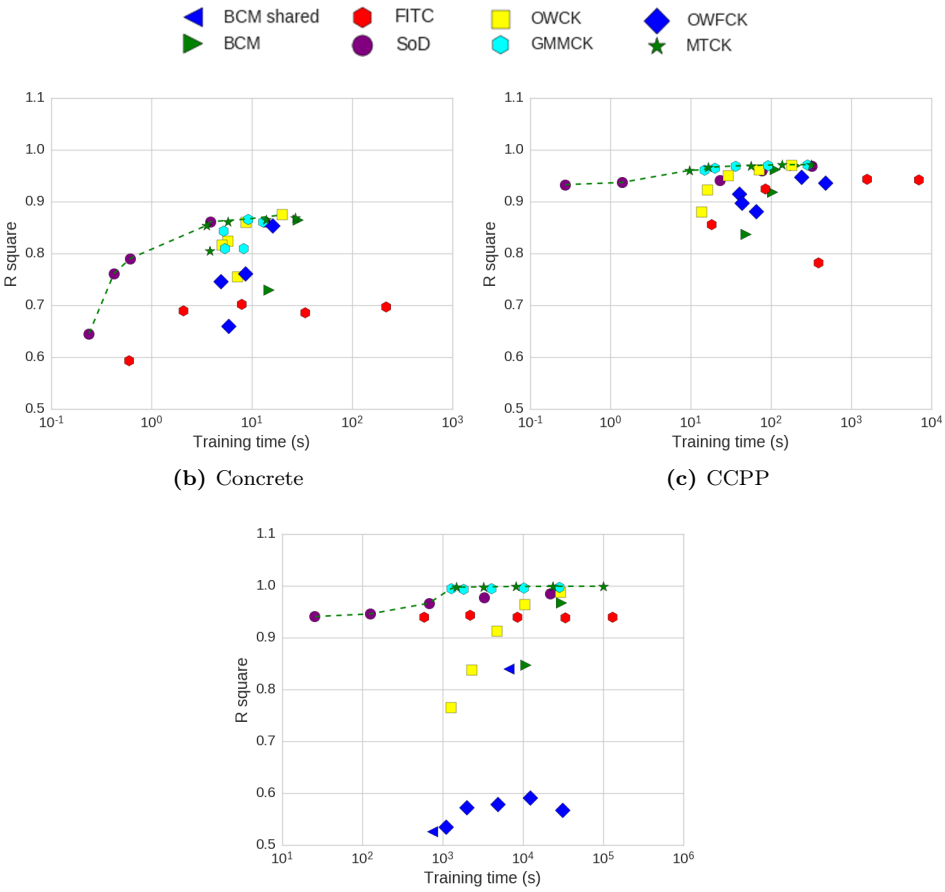
Table 5.3: Average SMSE score per dataset for each algorithm

Dataset	SOD	OWCK	GMM-CK	FCM-CK	FITC	BCM	BCM sh.	MTCK
Concrete	0.216	0.174	0.161	0.304	0.325	82.888	243.459	**0.149**
CCPP	0.052	0.063	**0.032**	0.084	0.110	0.780	25.602	**0.032**
Sarcos	0.036	0.106	0.004	0.430	0.059	628.280	0.552	**0.001**
Ackley	0.048	0.043	0.049	0.046	0.740	0.079	1.039	**0.019**
Schaffer	0.679	0.612	0.631	0.594	0.792	0.548	1.050	**0.328**
Schwefel	0.010	0.027	0.023	0.053	0.994	0.031	1.043	**0.001**
Rast	0.027	0.053	0.052	0.068	0.678	0.086	1.043	**0.002**
H1	0.324	1.082	0.473	2.125	0.835	0.343	1.046	**0.023**
Rosenbrock	0.001	0.003	0.003	0.019	1.000	0.006	1.050	**0.000**
Himmelblau	0.003	0.005	0.005	0.019	0.709	0.006	1.044	**0.000**
Diffpow	0.005	0.009	0.009	0.025	0.999	1.001	1.001	**0.000**

are shown in Figures 5.4 and 5.5. The results are shown with both objectives, time and accuracy (x and y axis respectively) in mind to show the trade-off and to show that some algorithms are performing better in both objectives. The R^2 scores of each dataset per algorithm, averaged over all folds, are shown in Table 5.1. The MSLL scores are provided in Table 5.2 and the SMSE scores in Table 5.3. The best results for each dataset are shown in bold face.

5.6.4 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

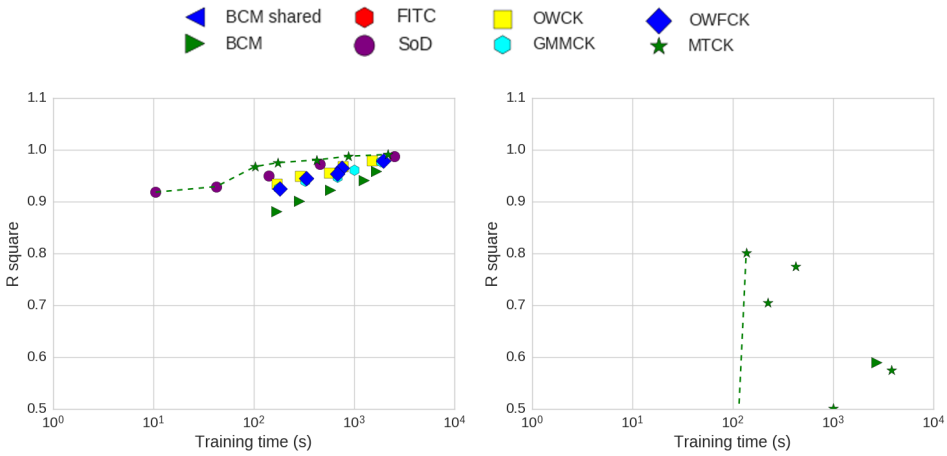


(b) Concrete **(c)** CCPP

(d) Sarcos

Figure 5.3: Quality measurements of each algorithm with the hyper-parameters increasing in sample sizes for FITC and SoD, and decreasing in cluster sizes for the cluster based algorithms as explained in Subsection 5.6.1. The results are shown for the Concrete, CCPP and Sarcos datasets. 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.

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.



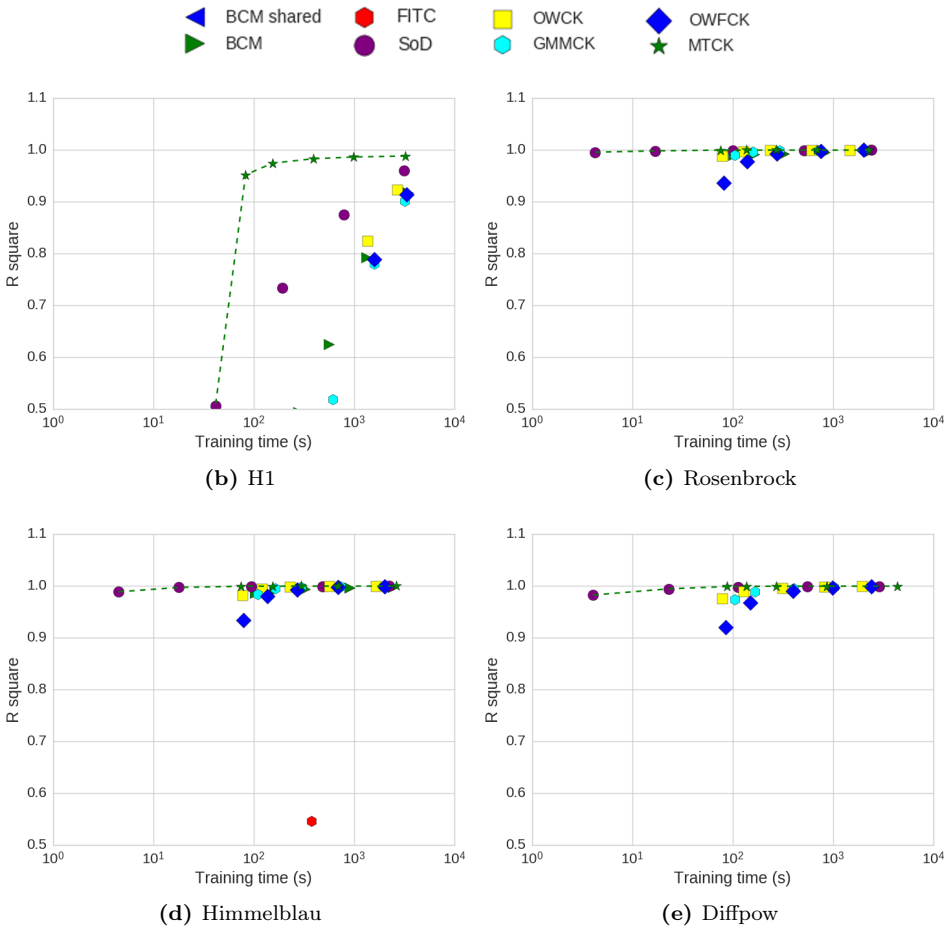
(b) Ackley **(c)** Schaffer

(d) Schwefel **(e)** Rastrigin

Figure 5.4: Quality measurements of each algorithm with the hyper parameters increasing in sample sizes for FITC and SoD, and decreasing in cluster sizes for the cluster based algorithms on the first half of synthetic datasets. The training time on the x axis and the R^2 score on the y axis. The green line indicates the non dominated set.

5.7 Efficient Global Optimization

In many real-world optimization problems, such as optimizing the manufacturing of car body parts or the production of steel, function evaluations are costly, either in time or money. *Efficient Global Optimization* (EGO) [70] is a procedure



(b) H1 **(c)** Rosenbrock

(d) Himmelblau **(e)** Diffpow

Figure 5.5: Quality measurements of each algorithm with the hyper parameters increasing in sample sizes for FITC and SoD, and decreasing in cluster sizes for the cluster based algorithms on the second half of the synthetic datasets. The training time on the x axis and the R^2 score on the y axis. The green line indicates the non dominated set.

designed to use a very low number of function evaluations while optimizing a specific function. The procedure uses a surrogate model to approximate the response surface of the real function. The surrogate model is fitted using an initial space filling *Design of Experiments* (DOE) [92]. Once the surrogate model is fitted on this data, optimization on the surrogate model's response surface can be performed

to find good candidate solutions for the black-box function to be optimized. This step does not require any additional expensive function evaluations since it uses the surrogate model. For the selection of these candidate points, EGO uses an infill-criterion, which is meant to provide a nice balance between exploration and exploitation. The newly found candidate solution is then evaluated against the black-box function and added to the data set and used to re-fit the surrogate model. This procedure is repeated untill the convergence criteria are met.

The Efficient Global Optimization [70] or Bayesian optimization [93, 94] is a sequential model-based global optimization algorithm that is built on stochastic models over the unknown objective function. The Kriging modeling technique [95] is originally proposed as the underlying model in EGO.

5.7.1 The Efficient Global Optimization Algorithm

EGO [70] is proposed to optimize expensive objective functions by sequentially choosing new candidate solutions from an underlying Kriging model. The candidate solutions are obtained by maximizing the so-called *acquisition function* or infill-criterion. Acquisition functions usually take the mean and variance of the posterior process (Eq. 5.10) into account, in order to balance the exploration and exploitation of the global search. Adding the newly obtained data points into the underlying Kriging model, its posterior process is modified and the acquisition function is updated accordingly. In this manner, a sequence of new solutions are generated iteratively. This algorithm is summarized in Algorithm 5.1. Many acquisition functions have been proposed and investigated [96]. The most popular ones are: Lower Bound (LB) [97], the Probability of Improvement (PI) [98, 99] and Expected Improvement (EI) [70]. In this chapter, we focus only on the expected improvement, that is defined as follows, in terms of minimization:

$$\begin{aligned} \mathrm{EI}(\mathbf{x}) &= \mathbf{E}[\max\{0, \min(\mathbf{y}) - y(\mathbf{x})\} \mid \mathbf{y}] \\ &= (\min(\mathbf{y}) - m(\mathbf{x}))\Phi\left(\frac{\min(\mathbf{y}) - m(\mathbf{x})}{s(\mathbf{x})}\right) \\ &+ s(\mathbf{x})\phi\left(\frac{\min(\mathbf{y}) - m(\mathbf{x})}{s(\mathbf{x})}\right) \end{aligned} \tag{5.16}$$

Algorithm 5.1 Efficient Global Optimization

1 Generate the initial data set $\mathcal{X}, \mathbf{y}$
2 Fit the Kriging model hyper-parameters on the initial data set $\mathcal{X}, \mathbf{y}$.
3 **while** the stop criteria are not fulfilled **do**
4 Find global optimum of the infill criterion:

$$\mathbf{x}^* = argmax_{\mathbf{x}} \, \mathrm{EI}(\mathbf{x})$$

5 Evaluate $\mathbf{x}^*$: $y^* = y(\mathbf{x}^*)$ and append $\mathbf{x}^*, y^*$ to $\mathcal{X}, \mathbf{y}$.
6 Re-estimate the Kriging model hyper parameters
7 **end while**

where $\Phi(\cdot), \phi(\cdot)$ denote the cumulative distribution function and the probability density function of the standard normal distribution, respectively. It takes into account the quantity of the expected improvement and also rewards a higher variance. In addition, the gradient of the expected improvement is given in Equation 5.17, as it is required by the quasi-Newton optimization procedure, that is used in the next subsections.

$$\begin{aligned} \nabla \, \mathrm{EI}(\mathbf{x}) &= \phi(u) \nabla s(\mathbf{x}) - \Phi(u) \nabla m(\mathbf{x}) \\ u &= \frac{\min(\mathbf{y}) - m(\mathbf{x})}{s(\mathbf{x})} \end{aligned} \tag{5.17}$$

When applying the EGO algorithm to a large initial data set (e.g., in the experiment design), the CPU time spent on the hyper-parameter re-estimation becomes computationally infeasible. To relax this issue, it is proposed to use time complexity reduction techniques that have been developed for the Kriging model.

5.7.2 Cluster Kriging-based EGO

It is proposed to exploit the Cluster Kriging variants in the EGO algorithm, for time complexity reduction. Although various complexity reduction (or approximation) methods exist for Kriging, we state that Cluster Kriging is more suitable for the EGO algorithm for the following reasons.

Firstly, the Kriging models on each cluster can be executed in parallel, which yields an additional linear speedup in practice. Secondly, after a new candidate solution is found through the acquisition function, the hyper-parameters of Kriging needs to be re-estimated. Taking the cluster information into account, it is proposed to only re-estimate the Kriging models on the clusters that this new solution belongs to. This operation results in another linear speedup in the hyper-parameter re-estimation procedure, as in the best scenario, only one Kriging model is subject to re-fitting. Thirdly, the acquisition function, e.g., the expected improvement is still well-defined on Cluster Kriging because either the posterior process (Eq. 5.10) or at least the mean and variance function (Eq. 5.14) can be derived. The algorithm is presented in Algorithm 5.2.

In the algorithm, the initial fitting procedure can be parallelized (line 2). Usually, the cluster (and the Kriging model on it) that the new solution belongs to is updated (line 13-16). A counter c is incremented every time when a new candidate solution is generated (line 5). If the c value, that is the recently appended data points, are more than 10% of the initial data set, the clustering is performed again to keep the size of each cluster balanced and capture the information contained in the newly added points.

5.7.3 Experiments

Several experiments are conducted to show both the empirical time complexity and convergence rate of the proposed Cluster Kriging based EGO, including all the variants of Cluster Kriging discussed earlier in this chapter. The performance of the proposed algorithm is compared to the original EGO that uses *Ordinary Kriging* (OK). For our experiments, the benchmark functions chosen are *Ackley*, *Rastrigin* and *Schaffer*. These functions are chosen because they are used often in optimization experiments, are highly multi modal, and are of a relatively high complexity.

Experiment 5.1 The algorithms compared are: EGO with Ordinary Kriging (OK), Tree-based local Kriging models (MTCK), Superposition of Kriging models (OWCK) and the mixture of Kriging models (GMMCK). Each of the Cluster Kriging variants uses 5 clusters. Both execution time and convergence rate are

Algorithm 5.2 Cluster Kriging based Efficient Global Optimization (CK-EGO)

Input: Data set $\mathcal{X}, \mathbf{y}$ obtained on a black-box function f. The number of clusters q. The clustering method is chosen from K-means, GMM or regression trees by the user.

Initial Clustering: $\{\mathcal{X}_i, \mathbf{y}_i\}_{i=1}^{q} \leftarrow \mathcal{X}, \mathbf{y}$
Create the Kriging model for each cluster:

$$y \mid \mathcal{X}_i, \mathbf{y}_i \sim \mathbf{\mathcal{N}}\left(m_i(\mathbf{x}), s_i^2(\mathbf{x})\right), \quad i = 1, \ldots, q$$

$c \leftarrow 0$
while the stop criteria are not fulfilled **do**
  $\mathbf{x}^* = argmax_{\mathbf{x}}\, \mathrm{EI}(\mathbf{x})$
  Evaluation: $y^* = f(\mathbf{x}^*)$
  $c \leftarrow c + 1$
  if $c > 10\%$ the number of data points in $\mathcal{X}$ **then**
    Merge the data set: $\mathcal{X}, \mathbf{y} \leftarrow \{\mathcal{X}_i, \mathbf{y}_i\}_{i=1}^{q}$
    Clustering the data set $\mathcal{X}, \mathbf{y}$ and re-create the Kriging models for each cluster.
    $c \leftarrow 0$
  else
    for every cluster i that $\mathbf{x}^*$ belongs to **do**
      Append $\mathbf{x}^*, y^*$ to $\mathcal{X}_i, \mathbf{y}_i$.
      Re-estimate the hyper-parameter for the Kriging model on cluster i.
    end for
  end if
end while
return $\mathbf{x}^*$

being measured with a fixed set of EGO iterations and optimization budget. The convergence is measured by taking the absolute error between the real optimum of the benchmark functions and the found optimum for each iteration of EGO. Each EGO run performs 10 iterations for the three benchmark functions in two dimensions. Three different initial sample sizes are used to train the surrogate models, $500, 1000$ and 5000 points in order to illustrate the growth of CPU time required per algorithm, when the size of the data available increases. For each different experimental setup, the average time and distance to the optimum is recorded over 20 runs with different random seeds.

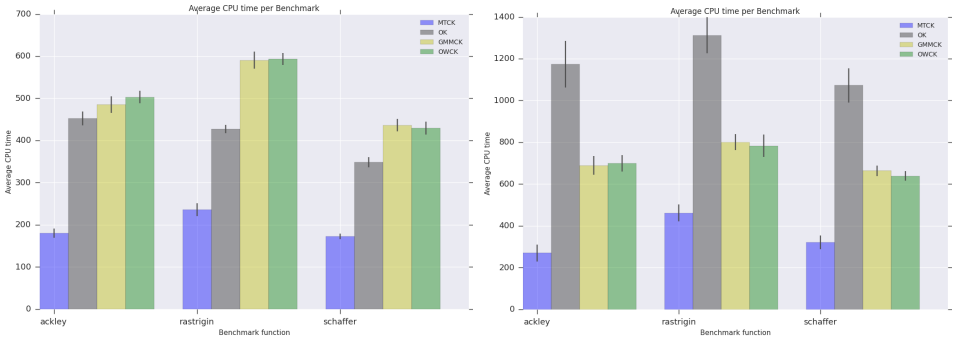
Experiment 5.2 The algorithms, OK, MTCK and OWCK are compared in five dimensions on the benchmark functions Ackley and Rastrigin also varying the algorithm that maximizes the expected improvement. CMA-ES and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm are compared.

Results From Figure 5.7 it can be observed that the Cluster Kriging based EGO variants perform very similar to OK, depending on the target function, a specific variant even outperforms Ordinary Kriging. Due to the relatively large variance in the results it is difficult to judge which algorithm performs better. However, from the CPU time in Figure 5.6 it can be observed that Cluster Kriging and in particular MTCK takes only a fragment of the time that Ordinary Kriging requires. Using a sample size of 500 points this difference is mainly due to the re-fitting of only one local model at a time. This can be seen by comparing MTCK with GMMCK and OWCK, since all three cluster Kriging variants use the same number of local models and only MTCK uses an adaptive local model strategy. When the number of points increases to 1.000 and even 5.000, the difference between the three cluster Kriging variants decreases but the difference with Ordinary Kriging becomes enormous. This shows that using EGO with Ordinary Kriging quickly becomes infeasible when the number of data points grow.

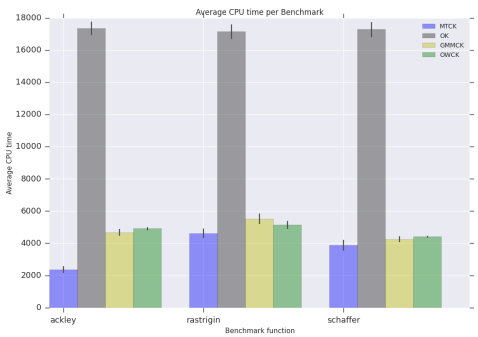
From Figure 5.8 it can be observed that also in higher dimensions Cluster Kriging does not under-perform Ordinary Kriging. In addition, it can be observed that using different optimization strategies for the expected improvement affects the convergence rate. However, the best optimization strategy clearly depends on the target function.

5.8 Conclusions

A novel Kriging approximation methodology, Cluster Kriging, is proposed, using a combination of smaller Kriging models trained on partitions of the data set. Four different algorithms using this methodology are proposed and explained in detail and 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 5.6) clearly show that for each data set, the *Gaussian Mixture Models Cluster Kriging* (GMM CK) and the Model Tree Cluster Kriging (MTCK)

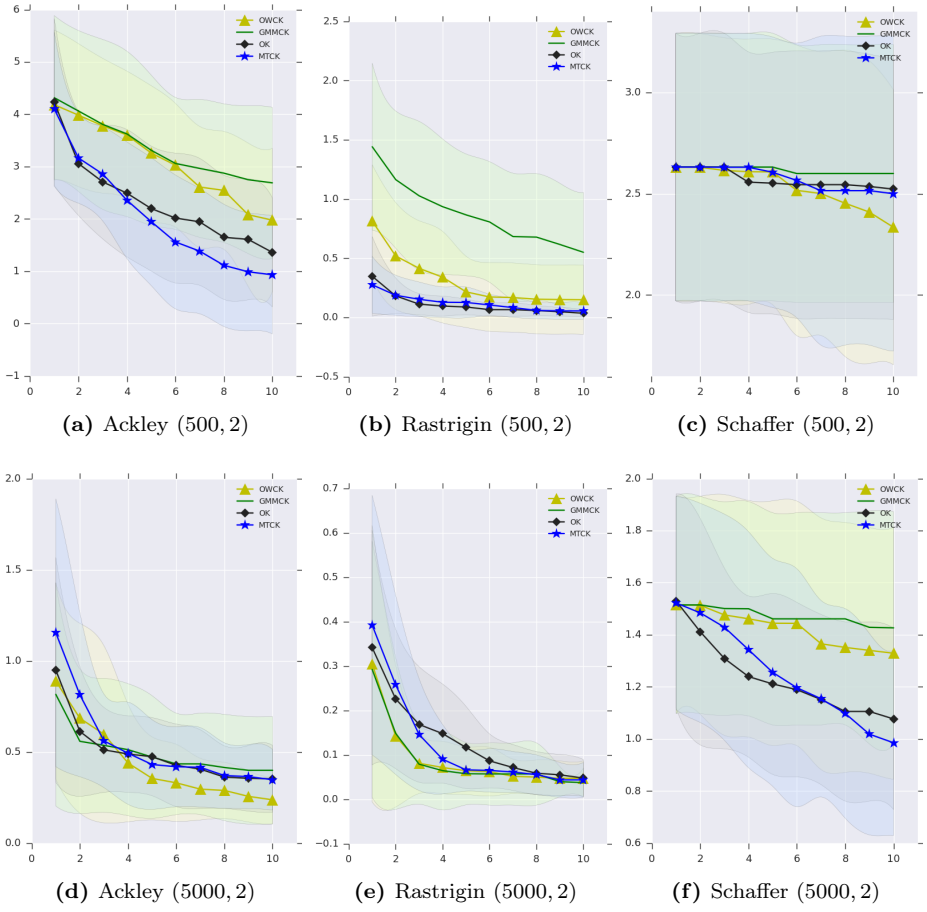


(a) CPU time $(500, 2)$ **(b)** CPU time $(1000, 2)$



(c) CPU time $(5000, 2)$

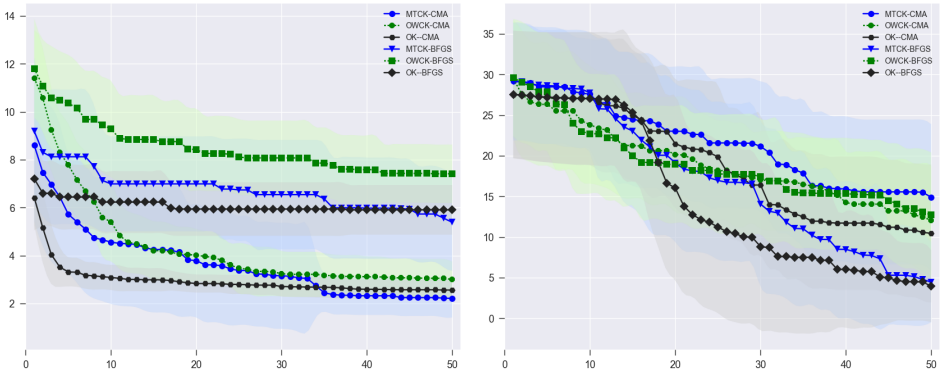
Figure 5.6: Average CPU time (in sec.) per benchmark function for varying sample sizes $(n_{samples}, d_{dimensions})$. In blue the MTCK algorithm, OK is denoted in grey, GMMCK in yellow and OWCK in green.



(a) Ackley $(500, 2)$ **(b)** Rastrigin $(500, 2)$ **(c)** Schaffer $(500, 2)$

(d) Ackley $(5000, 2)$ **(e)** Rastrigin $(5000, 2)$ **(f)** Schaffer $(5000, 2)$

Figure 5.7: Average convergence of the absolute error of three benchmark functions in two dimensions, with varying training sample sizes n and 10 iterations of EGO. Shown is the average over 20 runs (lines) and one standard deviation (shaded areas).



(a) Ackley $(500, 5)$ **(b)** Rastrigin $(500, 5)$

Figure 5.8: Average convergence of the absolute error of two benchmark functions in five dimensions using different optimization algorithms. 500 training samples and 50 iterations of EGO are used. Shown is the average over 20 runs (lines) and one standard deviation (shaded areas).

outperform the other algorithms in all measurements. It can also be observed that the *Bayesian Committee Machine* algorithms, both with shared parameters and with individual parameters, are very unstable when the number of clusters is above 8. This is most likely due to 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 much faster than any of the other algorithms, though it pays for this complexity reduction by a decrease in accuracy. Both for SoD and FITC, the training time increases faster than the training time of the cluster based algorithms. 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 per unseen data point is used for prediction.

It is shown that Cluster Kriging can be applied in the EGO algorithm for complexity reduction. Three variants of Cluster Kriging are validated in combination with EGO on some test functions. Based on the empirical results that are shown

in Section 5.7, it can be concluded that EGO using Cluster Kriging is much faster in terms of time complexity compared to the traditional EGO that employs an Ordinary Kriging model. Moreover, each of the Cluster Kriging variants perform very well compared to EGO using Ordinary Kriging in terms of convergence speed. From the results shown in Subsection 5.7.2, it can be inferred that the MTCK model fits the objective function well due to the reason that it captures local information much better than Ordinary Kriging.