Optimally Weighted Cluster Kriging for Big Data Regression

Bas van Stein(✉), Hao Wang, Wojtek Kowalczyk, Thomas Bäck, and Michael Emmerich

Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, Leiden, The Netherlands
{b.van.stein,h.wang,w.j.kowalczyk,t.h.w.baeck, m.t.m.emmerich}@liacs.leidenuniv.nl

Abstract. In business and academia we are continuously trying to model and analyze complex processes in order to gain insight and optimize. One of the most popular modeling algorithms is *Kriging*, or *Gaussian Processes*. A major bottleneck with Kriging is the amount of processing time of at least $O(n^3)$ and memory required $O(n^2)$ when applying this algorithm on medium to big data sets. With big data sets, that are more and more available these days, Kriging is not computationally feasible. As a solution to this problem we introduce a hybrid approach in which a number of Kriging models built on disjoint subsets of the data are properly weighted for the predictions. The proposed model is both in processing time and memory much more efficient than standard Global Kriging and performs equally well in terms of accuracy. The proposed algorithm is better scalable, and well suited for parallelization.

Keywords: Kriging · Gaussian Processes · K-means · Clustering · Big-data · Regression

1 Introduction

Regression as supervised learning is an important tool for analysis of data sets and as sub goal for further optimization or gaining knowledge about the data sets and their underlying processes. There are many kinds of regression algorithms: parametric models, which are easy to interpret but may lack expressive power to model complex functions, *Regression Tree* based methods like *Random Forests* [3] or *Gradient Boosted Decision Trees*, which lack the advantage of interpretation [7] but have more expressive power. There are also more complex algorithms like *Neural Networks*, or *Extreme Learning Machines* [15], that are able to model very complex functions but are usually not easy to work with in practice. And last but not least there are kernel-based methods such as *Support Vector Machines*, *Radial Basis Functions* and *Kriging* [16]. These kernel based algorithms are flexible and easy to work with, but are computationally expensive.

E. Fromont et al. (Eds.): IDA 2015, LNCS 9385, pp. 310–321, 2015.
DOI: 10.1007/978-3-319-24465-5_27

Kriging is a stochastic interpolation/regression approach, which originates from geostatistics [16] and originally targets exploration problems in mining. It has been widely used in spatial interpolation and regression tasks. Note that the Kriging method is also called *Gaussian Process Regression* [21] in the machine learning literature. In addition to generating predictions, Kriging also provides the expected mean squared error of point estimates, so-called *Kriging variance*. The Kriging variance is of significant importance because it is typically used to measure the uncertainty of the predictions, but in this paper, it also serves to find an optimal weighting scheme to combine multiple independent Kriging models.

Notation. Through this paper, we shall use n, k, d to denote the size of the data points, the number of clusters and dimensionality, respectively.

A major problem with Kriging, is the complexity of training the model, which requires solving a dense linear system of size $n \times n$, which takes $O(n^3)$[1]. Even more time is required when estimating the hyper-parameters. The algorithm also takes $O(n^2)$ memory, which might be a bottleneck in big data sets as well. In this paper a new algorithm is proposed, *Optimally Weighted Cluster Kriging* (OWCK), as an answer to this complexity problem. Using a clustering method we divide the possible big data set into k much smaller data sets, on each data set a Kriging model is being trained and the results from each model are combined to create a predictor in an optimal way. Using this method we can achieve a theoretical speedup of k^2 (k is the number of clusters), and when executing on a parallel system with k CPU's, we can improve that further to almost k^3. The memory required by the algorithm is reduced by factor k^2.

In Sect. 2 a brief introduction to Kriging is given. Then, in Sect. 3 relevant research is reviewed and in Sects. 4 and 5, our method and the results from our experiments are presented and discussed. Finally, a conclusion is drawn and suggestions for further research are made.

2 Kriging

Kriging is based on the assumption that the function to be approximated is the realization of a Gaussian Random Field with known (or estimated) covariance structure [22]. Based on the this assumption, Kriging interpolates function values at unknown points from observed function values. Normally, the Kriging model is trained on some input vector $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and the corresponding target values $\mathbf{y} = \{y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n)\}$. Kriging estimates the output at unknown data samples by modeling the response values as a realization of a random process y, which is a sum of a mean function $\mu(\cdot)$ and a centered Gaussian process ε,

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

The centered Gaussian process ε is a stochastic process, which is completely defined by providing a prescribed covariance function $k(\cdot, \cdot)$ [21]:

[1] There are asymptotically faster algorithms for inverting a matrix. e.g. Strassen's $O(n^{2.807})$ and Stothers $O(n^{2.373})$.

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

A common choice of $k(\cdot, \cdot)$ is the Gaussian covariance (also known as squared exponential):

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

where θ_i's are called *hyper parameters*, which are either predetermined or estimated through model fitting. When the mean values $\mu(\cdot)$ are assumed to be constant but need to be estimated, the method is called *Ordinary Kriging* (OK). In OK, the joint distribution of the (uncertain) outcome $y^t(\mathbf{x}^t)$ at a target point $\mathbf{x}^t$ and the observations $\mathbf{y}$ is Gaussian. In addition, for given $\mathbf{X}$ and μ it holds:

$$\begin{bmatrix} y^t \\ \mathbf{y} \end{bmatrix} \Big| \mu, \mathbf{X}, \mathbf{x}^t \sim \boldsymbol{\mathcal{N}} \left(\mu \mathbf{1}_{n+1}, \begin{bmatrix} c & \mathbf{c}^T \\ \mathbf{c} & \boldsymbol{\Sigma} \end{bmatrix} \right), \tag{2}$$

where

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

and $\mathbf{1}_{n+1}$ represents a column vector of length $n+1$ that contains only 1's.

By introducing a non-informative uniform prior distribution on μ, the *posterior* conditional distribution of y^t can be calculated by marginalizing μ out. Without any derivations, the posterior distribution for OK is again Gaussian [11]:

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

$$m(\mathbf{x}^t) = \left[\mathbf{c} + \left(\frac{1 - \mathbf{c}^T \boldsymbol{\Sigma}^{-1} \mathbf{1}_n}{\mathbf{1}_n^T \boldsymbol{\Sigma}^{-1} \mathbf{1}_n} \right) \mathbf{1}_n \right]^T \boldsymbol{\Sigma}^{-1} \mathbf{y} \tag{4}$$

$$s^2(\mathbf{x}^t) = c^2 - \mathbf{c}^T \boldsymbol{\Sigma}^{-1} \mathbf{c} + \frac{(1 - \mathbf{c}^T \boldsymbol{\Sigma}^{-1} \mathbf{1}_n)^2}{\mathbf{1}_n^T \boldsymbol{\Sigma}^{-1} \mathbf{1}_n} \tag{5}$$

The posterior mean function is used as the predictor while the posterior variance is the so-called Kriging variance.

3 Relevant Research

The high computational complexity of Kriging is not an unnoticed problem in the world of data analysis and modeling. Several modifications and algorithms are already proposed. One of the most intuitive and "simple" algorithms is *Nearest Neighbour Kriging* [8], where we train a Kriging model using only k neighbours of the point we want to predict. The disadvantage of such an algorithm is that we need to train a model for each record we want to predict. Another disadvantage is that the accuracy suffers greatly when not enough neighbours are being used.

Another modification to Kriging is the approximation of the covariance matrix with a sparse precision matrix by *Hartman, L. and Hössjer, O.* [12]. In this paper they use *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. Another algorithm, focused especially on data sets with scattered data points uses fast matrix-vector products to reduce the training complexity [18].

Several other attempts have been made to divide the Kriging model in sub-models [6,19], each solution for different domains. In [6], a *Bagging* [2] method is proposed to increase the robustness of the Kriging algorithm, rather than speeding up the algorithms training time. In [19], a partitioning method is introduced to separate the data points into local Kriging models and combine the different models using a distance metric.

While previously mentioned work has some similarities to what is proposed in this paper, the weighting and clustering methods being used in previous mentioned work seems to be far from optimal.

4 Optimally Weighted Cluster Kriging

As mentioned before, Kriging suffers from high computation time as the number of sampling points gets high. In this section, we will propose an algorithm that is suitable for efficiently processing big data sets while maintaining much of the ideas of Kriging. The basic idea of the new algorithm is to cluster the data set and build independent Kriging models for each part of the data. The value of an unknown point is predicted as a linear combination of the predictions from all the Kriging models. In addition, an optimal weights setting for the linear combination exists and is calculated. We therefore call the newly proposed algorithm *Optimally Weighted Cluster Kriging*.

4.1 Clustering the Data Set

In the first step, the input data samples $\mathbf{X}$ should be separated into several clusters (k clusters here), which can be represented by a set of tuples:

$$\{(\mathbf{X}_l, \mathbf{y}_l)\}_{l=1}^{k}, \quad [\mathbf{X}_1^T, \mathbf{X}_2^T, \ldots, \mathbf{X}_k^T]^T = \mathbf{X}, \quad [\mathbf{y}_1^T, \mathbf{y}_2^T, \ldots, \mathbf{y}_k^T]^T = \mathbf{y}$$

where $\mathbf{y}$ is again the vector containing all of the observed response values. Note that l is the label identifying the clusters. We also use $\{n_l\}_{l=1}^{k}$ to denote the size of each cluster. Within cluster l, all the input data samples can be represented as:

$$\mathbf{X}_l = [\mathbf{x}_{l_i}]_{i=1}^{n_l}, \quad \text{and} \quad \mathbf{y}_l = [y_{l_i}]_{i=1}^{n_l}.$$

In order to cluster the training data for our local models, a clustering algorithm that gives roughly equal sized parts is preferred. This lead us to choosing two algorithms: *K-means* clustering, and Random clustering. In Random clustering we divide the data points at random in k groups by assigning each data point to an alternating label, with *K-means* we use the K-means clustering algorithm to divide the training data into k ($< n$) clusters using the *Forgy* [9] method. k random data points are picked as the initial centroids of the clusters. Next, each

data point is assigned to the cluster with the least Euclidean distance, after assigning a data point to a cluster, the cluster's centroid is updated and the process is repeated several times. The algorithm minimizes the within-cluster sum of squares (Eq. 6):

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2, \tag{6}$$

where S is the set of clusters, n is the number of data points and μ_i is the mean of the points in S_i. Note that the within-cluster sum of squares takes only $O(nk)$ execution time.

Our hypothesis is that using K-means clustering, the Kriging models trained on each cluster have a high accuracy due to the assumed "local" neighbourhood of the training data. With Random clustering, we hypothesize that each model is equally fitted, since each data set has roughly the same structure. Using random clustering we assume that the weighted combination will be more robust than normal Global Kriging.

4.2 Kriging Model on Clusters

The next step is to fit a Kriging model for each of the clusters. The procedure is the same as described in Sect. 3 except that each Kriging model has its own distinct data set. The Kriging formula for each cluster can simply be obtained by adding the cluster label l to the data set and all the parameters (e.g. μ) in Eq. 2. On cluster l, the joint distribution of the response-values to predict the observed data is:

$$\begin{bmatrix} y^t \\ \mathbf{y}_l \end{bmatrix} \Big| \mu_l, \mathbf{X}_l, \mathbf{x}^t \sim \boldsymbol{\mathcal{N}} \left(\mu_l \mathbf{1}_{n_l+1}, \begin{bmatrix} c_l & \mathbf{c}_l^T \\ \mathbf{c}_l & \boldsymbol{\Sigma}_l \end{bmatrix} \right) \tag{7}$$

$$c_l = k_l(\mathbf{x}^t, \mathbf{x}^t), \quad \mathbf{c}_{l_i} = k_l(\mathbf{x}^t, \mathbf{x}_{l_i}), \quad \boldsymbol{\Sigma}_{l_{ij}} = k_l(\mathbf{x}_{l_i}, \mathbf{x}_{l_j}).$$

Note that the covariance function is also indexed by l due to the fact that we might choose a different covariance function for each cluster. The predictive conditional distribution can be expressed as:

$$y^t | \mathbf{y}_l, \mathbf{X}_l, \mathbf{x}^t \sim \boldsymbol{\mathcal{N}} \left(m_l(\mathbf{x}^t), s_l^2(\mathbf{x}^t) \right) \tag{8}$$

The formula above is exactly the same as Eq. 3 except that the posterior mean and variance are labeled by l. We choose the Gaussian covariance function (Eq. 1) for all the Kriging models on the clusters. The hyper-parameters in the covariance function are fitted by Maximum Likelihood Estimation (MLE) [21]. In our algorithm, the *Cobyla* [20] algorithm is used to solve the maximum likelihood estimation task. The *nugget* parameter that specifies the amount of noise expected, is set to 0.01 in our experiments but can be set differently depending on the data.

4.3 Weighting Distribution

In order to combine multiple Kriging models, trained on different clusters of data, a straightforward method is to use a weighting distribution to model how much "trust" should be put on the prediction from each cluster. By using cluster indicator variable C, the weighting vector $\mathbf{w}$ can be written as:

$$\mathbf{w} = \{p(C = l) = w_l\}_{l=1}^{k}, \quad \sum_{l=1}^{k} w_l = 1, \quad w_l \geq 0,$$

which is non-negative and satisfies the normalization property. Unlike the weights in the Gaussian mixture model [13], the weighting distribution $\mathbf{w}$ depends on the target data sample $\mathbf{x}^t$. The optimal setting of $\mathbf{w}$ is discussed in the following sections.

4.4 Prediction Using All the Kriging Models

We will show how to make the overall prediction using all the Kriging models by combining the predictions from each of them. The background assumption is that the Gaussian Processes (behind the Kriging models) on the clusters are mutually independent from each other, which is reasonable due to our clustering procedure. Our approach is to first obtain the joint distribution of $y^t(\mathbf{x}^t)$ and $\mathbf{y}$. Actually, it is the joint distribution conditioning on $\mathbf{X}, \mathbf{x}^t$ and $\mathbf{y}$. We omit the conditional symbols in the following for simplicity. By applying the total probability with respect to the cluster indicator variable C, we have:

$$\begin{aligned} p(y^t, \mathbf{y}) &= \sum_{l=1}^{k} p(y^t, \mathbf{y}|C = l)p(C = l) \\ &= \sum_{l=1}^{k} p(y^t, \{\mathbf{y}_i\}_{i=1}^{k}|C = l)p(C = l) \\ &= \sum_{l=1}^{k} p(y^t, \mathbf{y}_l|\{\mathbf{y}_i\}_{i\neq 1}, C = l)p(\{\mathbf{y}_i\}_{i\neq l}|C = l)p(C = l) \end{aligned}$$

Due to the fact that $y^t, \mathbf{y}_l$ are conditionally independent of $\{\mathbf{y}_i\}_{i\neq l}$ given $C = l$, the equations above can be further simplified to:

$$\begin{aligned} p(y^t, \mathbf{y}) &= \sum_{l=1}^{k} p(y^t, \mathbf{y}_l|C = l)p(\{\mathbf{y}_i\}_{i\neq l}|C = l)w_l \\ &= \sum_{l=1}^{k} p(y^t, \mathbf{y}_l) \left(\prod_{i\neq l} p(\mathbf{y}_i) \right) w_l \end{aligned}$$

The product inside of the sum is again due to the independence. Now we perform the conditioning on $\mathbf{y}$ and omit the cluster indicator variable C:

$$
\begin{aligned}
p(y^t|\mathbf{y}) &= \frac{p(y^t, \mathbf{y})}{p(\mathbf{y})} \\
&= \sum_{l=1}^{k} p(y^t, \mathbf{y}_l) \frac{\prod_{i \neq l} p(\mathbf{y}_i)}{\prod_{i=l}^{k} p(\mathbf{y}_i)} w_l \\
&= \sum_{l=1}^{k} p(y^t|\mathbf{y}_l) w_l
\end{aligned} \tag{9}
$$

Note that each of the conditional distributions inside of the summation is exactly the same as the Gaussian conditional distribution obtained from the corresponding cluster. Finally the distribution of y^t conditioning on $\mathbf{y}$ can be obtained by combining Eqs. 8 and 9:

$$
y^t|\mathbf{y} \sim \boldsymbol{\mathcal{N}} \left(\sum_{l=1}^{k} w_l m_l(\mathbf{x}^t), \sum_{l=1}^{k} w_l^2 \sigma_l^2(\mathbf{x}^t) \right) \tag{10}
$$

Equation 10 suggests that the overall prediction is simply the weighted average of the prediction from each cluster while the prediction mean square error (variance) is also the weighted average of variance from each cluster, where the weight is squared.

4.5 Optimal Weighting

Equation 10 also suggests that an optimal weighting distribution exists and can be obtained by minimizing the variance of the weighted average. Thus, by putting all the variances into the diagonal of a matrix $\mathbf{Q} = \text{diag}(\sigma_1^2, \ldots, \sigma_k^2)$, the optimal weighting $\mathbf{w}^*$ is the solution to the following optimization problem:

$$
\text{minimize:} \quad \mathbf{w}^T \mathbf{Q} \mathbf{w} \quad \text{subject to:} \sum_{l=1}^{k} w_l = 1, \quad w_l \geq 0, \quad l = 1, \ldots, k.
$$

This quadratic programming problem can be immediately solved by the method of Lagrangian multipliers [1]. The optimal weights setting is:

$$
\mathbf{w}^* = \frac{\mathbf{Q}^{-1} \mathbf{1}_k}{\mathbf{1}_k^T \mathbf{Q}^{-1} \mathbf{1}_k}, \tag{11}
$$

Equation 11 gives the optimal setting of the weighting distribution for our approach. Note that matrix $\mathbf{Q}$ is invertible if and only if the conditional variances of the prediction (diagonal elements) are not zero, which is guaranteed because the prediction is not performed on any of the data samples.

4.6 Pseudo Code

In Algorithm 1 an outline of the algorithm is presented. Any algorithm can be used to create the clusters for training the models, though to gain maximal

speedup the clusters should be of the same size. In the prediction phase a method *OptimiseWeights* is used. This method uses the predicted mean squared errors from the Kriging models in order to find an optimal weighting distribution per prediction.

Algorithm 1. Optimally Weighted Cluster Kriging

Given: A data set X_{train} and X_{test} with records $x_1, \ldots, x_n$, a target attribute y and the number of clusters k

Initialization:

$Clusters = k - MEANS(X_{train}, k)$

for all $Cl_i \in Clusters$ **do**
  $Models[i] = Kriging.train(Cl_i)$
end for
$Predictions_{final} = []$
for all $x_i \in X_{test}$ **do**
  $Predictions = []$
  $MSE = []$
  for all $model_i \in Models$ **do**
    $Predictions[i], MSE[i] = model_i.predict(x_i)$
  end for
  $Weights = OptimiseWeights(MSE)$
  $Predictions_{final}[i] = WeightedSum(Predictions, Weights)$
end for
return $Predictions_{final}$

5 Experimental Setup and Results

We assessed the performance of *Optimally Weighted Cluster Kriging* (OWCK) on several known benchmark functions from the *DEAP* [10] Python package: *Rastrigin*, *Rosenbrock*, *Ackley*, *Himmelblau*, *H1*, *Schwefel*, *Schaffer*, and *Diffpow*. On these functions, Ordinary Kriging and OWCK were tested. The reason to use Ordinary Kriging is due to its applicability and simplicity. Both k-means and Random partitioning are used in the experiments. For each experiment the number of data samples ranges from 1000 to 10.000 records, in two and five dimensions. Five-fold cross validation is used on each of these runs, effectively using 4/5 of the data samples as training set and 1/5 as test set. The results shown are the averaged results from the five folds.

In the Tables 1 and 2 the R^2 scores of the functions in 2 dimensions are presented. The number of the clusters tested here are $4, 8, 16, 32, 64$. For each column, the entries are shaded in different levels of gray depending on their value. The best scores in each column are the most dark, the lower scores are more light.

In Tables 3 and 4 the results from the same algorithms on 5 dimensional functions are shown.

In Fig. 1 the execution time per algorithm is shown for the Rastrigin function. Note that 4/5 of the data was used for training and 1/5 for predicting in the

Table 1. Accuracy score of each algorithm (R^2) on the benchmark functions in 2 dimensions with a dataset of size 1.000.

	h1	Ackley	Himmel.	Diffpow	Rosenb.	Rast.	Schaffer	Schwefel
OK	0.254	0.920	1.000	1.000	1.000	0.547	0.352	0.170
4 Random	0.229	0.480	1.000	1.000	1.000	0.547	0.354	0.158
8 Random	0.216	0.433	1.000	0.803	1.000	0.546	0.354	0.136
16 Random	0.207	0.447	0.829	0.519	1.000	0.543	0.356	0.137
32 Random	0.213	0.465	0.709	0.531	0.996	0.540	0.355	0.132
64 Random	0.239	0.524	0.717	0.544	0.882	0.530	0.358	0.140
4 K-means	0.311	0.932	1.000	1.000	1.000	0.541	0.400	0.177
8 K-means	0.272	0.936	1.000	1.000	1.000	0.525	0.403	0.476
16 K-means	0.149	0.936	1.000	1.000	1.000	0.515	0.376	0.629
32 K-means	0.111	0.930	0.999	0.998	1.000	0.499	0.306	0.468
64 K-means	−1.364	0.895	1.000	0.997	1.000	−0.085	0.052	−0.054

Table 2. Accuracy score of each algorithm (R^2) on the benchmark functions in 2 dimensions with a dataset of size 5.000.

	h1	Ackley	Himmel.	Diffpow	Rosenb.	Rast.	Schaffer	Schwefel
OK	0.519	0.940	1.000	1.000	1.000	0.543	0.401	0.582
4 Random	0.453	0.929	1.000	1.000	1.000	0.543	0.352	0.196
8 Random	0.370	0.919	1.000	1.000	1.000	0.543	0.350	0.192
16 Random	0.297	0.589	1.000	1.000	1.000	0.543	0.350	0.171
32 Random	0.249	0.449	1.000	1.000	1.000	0.543	0.351	0.142
64 Random	0.229	0.455	0.887	0.578	1.000	0.542	0.351	0.135
4 K-means	0.528	0.942	1.000	1.000	1.000	0.561	0.403	0.549
8 K-means	0.580	0.944	1.000	1.000	1.000	0.540	0.407	0.956
16 K-means	0.528	0.945	1.000	1.000	1.000	0.577	0.410	0.718
32 K-means	0.550	0.945	1.000	1.000	1.000	0.664	0.410	0.938
64 K-means	0.561	0.942	1.000	1.000	1.000	−1.744	0.280	0.936

Table 3. Accuracy score of each algorithm (R^2) on the benchmark functions in 5 dimensions with a dataset of size 5.000.

	h1	Ackley	Himmel.	Diffpow	Rosenb.	Rast.	Schaffer	Schwefel
OK	0.488	0.943	0.999	1.000	1.000	0.539	0.405	0.534
4 Random	0.537	0.942	0.997	0.999	0.999	0.539	0.405	0.438
8 Random	0.511	0.939	0.994	0.997	0.998	0.539	0.401	0.356
16 Random	0.464	0.936	0.987	0.995	0.995	0.538	0.396	0.185
32 Random	0.310	0.927	0.972	0.990	0.990	0.535	0.387	0.133
64 Random	0.175	0.225	0.902	0.980	0.981	0.467	0.201	0.111
4 K-means	0.557	0.942	0.997	0.999	0.999	0.539	0.404	0.634
8 K-means	0.530	0.940	0.995	0.998	0.997	0.538	0.402	0.706
16 K-means	0.467	0.936	0.987	0.995	0.994	0.535	0.392	0.450
32 K-means	0.359	0.924	0.963	0.987	0.981	0.514	0.323	0.318
64 K-means	0.324	0.820	0.912	0.965	0.949	0.399	0.257	0.237

Table 4. Accuracy score of each algorithm (R^2) on the benchmark functions in 5 dimensions with a dataset of size 10.000.

	h1	Ackley	Himmel.	Diffpow	Rosenb.	Rast.	Schaffer	Schwefel
OK	0.606	0.950	1.000	1.000	1.000	0.531	0.435	0.635
4 Random	0.561	0.950	0.999	0.999	1.000	0.529	0.428	0.171
8 Random	0.543	0.948	0.997	0.999	0.999	0.529	0.427	0.156
16 Random	0.516	0.947	0.994	0.997	0.998	0.528	0.425	0.218
32 Random	0.465	0.943	0.986	0.995	0.996	0.526	0.422	0.155
64 Random	0.367	0.935	0.971	0.990	0.991	0.523	0.416	0.109
4 K-means	0.619	0.950	0.999	1.000	1.000	0.529	0.427	0.554
8 K-means	0.547	0.949	0.997	0.999	0.999	0.530	0.428	0.801
16 K-means	0.524	0.948	0.993	0.998	0.997	0.528	0.421	0.668
32 K-means	0.433	0.940	0.979	0.994	0.990	0.523	0.365	0.621
64 K-means	0.361	0.878	0.948	0.980	0.970	0.444	0.340	0.374

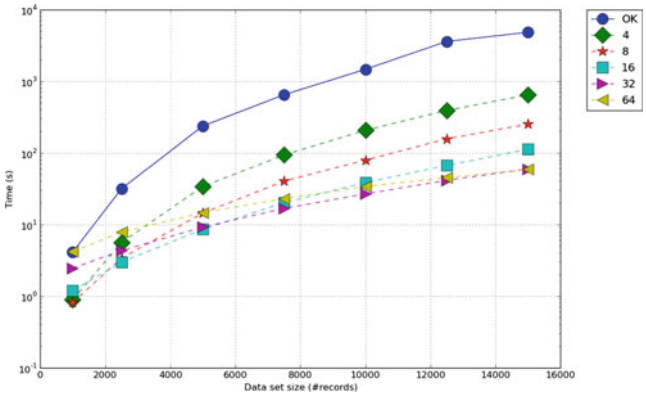

Fig. 1. Execution time per cluster size for the Rastrigin function in 5 dimensions. Using an Intel Core i7-4910MQ CPU 2.90 GHz, 8 cores and 32 GB of main memory. Running on one core.

Figure. To gain maximum benefit from the OWCK algorithm, we can process each cluster on a different thread (on a different CPU core), speeding up the algorithm linearly in the amount of cores.

6 Conclusions and Further Research

In this paper a novel algorithm which creates multiple small Kriging models and combines these models using an optimally weighting method is presented. It is shown that the Optimally Weighted Cluster Kriging model outperforms Ordinary Kriging in both execution time and accuracy. The number of clusters that should be used depends a lot on the size of the dataset and partly on the nature of the dataset as well. When a dataset with n points is split into k clusters of roughly equal size, our method reduces both the required execution time and

memory by factor k^2. Moreover, the algorithm can be parallelized, providing yet another speedup factor that is linear in the number of workers. In practice, it pushes the limits of applicability of Kriging from thousands to millions of data points.

For further research several extensions and generalizations can be made to improve the algorithms accuracy. One could look for example for different clustering methods. An interesting candidate is *Mixture of Gaussians* [17], where each point has a probability to belong to a certain cluster. Using these probabilities we may derive new weighting schemes that might perform better. The method used for comparing the different Kriging algorithms can be expanded by using a comparison framework as proposed in [5], as well as comparison with a few other recent Kriging variations [4,14]. Moreover, the nugget of the Kriging model plays an important role in training the model and has a big effect on the accuracy of the models, in further research we should exploit a way to optimize the nugget parameter for both the Ordinary Kriging model as well as for the cluster models.

References

1. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
2. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
3. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
4. Bui, T.D., Turner, R.E.: Tree-structured Gaussian process approximations. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2213–2221. Curran Associates Inc. (2014). http://papers.nips.cc/paper/5459-tree-structured-gaussian-process-approximations.pdf
5. Chalupka, K., Williams, C., Murray, I.: A Framework for Evaluating Approximation Methods for Gaussian Process Regression, pp. 1–18 (2012). arXiv preprint arXiv:1205.6326
6. Chen, T., Ren, J.: Bagging for Gaussian process regression. Neurocomputing **72** (7–9), 1605–1610 (2009)
7. D'Ambrosio, A., Aria, M., Siciliano, R.: Accurate tree-based missing data imputation and data fusion within the statistical learning paradigm. J. Classif. **29**(2), 227–258 (2012)
8. Emmerich, M.: Single-and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels. Ph.D. thesis, FB Informatik, TU Dortmund (2005)
9. Forgy, E.W.: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics **21**, 768–769 (1965)
10. Fortin, F., Michel, F., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)
11. Ginsbourger, D., Le Riche, R., Carraro, L.: Kriging is well-suited to parallelize optimization. In: Tenne, Y., Goh, C.-K. (eds.) Computational Intel. in Expensive Opti. Prob. ALO, vol. 2, pp. 131–162. Springer, Heidelberg (2010)
12. Hartman, L., Hössjer, O.: Fast kriging of large data sets with Gaussian Markov random fields. Comput. Stat. Data Analy. **52**(5), 2331–2349 (2008)

13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer, New York (2001)
14. Hensman, J., Sheffield, U., Fusi, N., Lawrence, N.: Gaussian processes for big data. In: Proceedings of UAI, vol. 29, pp. 282–290 (2013)
15. Huang, G., Wang, D.H., Lan, Y.: Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. **2**(2), 107–122 (2011)
16. Krige, D.G.: A statistical approach to some basic mine valuation problems on the Witwatersrand. J. Chem. Metall. Min. Soc. S. Af. **52**(6), 119–139 (1951)
17. Lindsay, B.: Mixture models: theory, geometry, and applications. In: Conference Board of the Mathematical Sciences: NSF-CBMS Regional Conference Series in Probability and Statistics, Institute of Mathematical Statistics (1995)
18. Memarsadeghi, N., Raykar, V.C., Duraiswami, R., Mount, D.M.: Efficient kriging via fast matrix-vector products. In: 2008 IEEE, Aerospace Conference, pp. 1–7. IEEE (2008)
19. Nguyen-Tuong, D., Seeger, M., Peters, J.: Model learning with local Gaussian process regression. Adv. Rob. **23**(15), 2015–2034 (2009)
20. Powell, M.J.D.: A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Gomez, S., Hennart, J.-P. (eds.) Advances in Optimization and Numerical Analysis, pp. 51–67. Springer, Boston (1994)
21. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. Adaptative Computation and Machine Learning Series. University Press Group Limited, New Era Estate (2006)
22. Stein, M.L.: Interpolation of Spatial Data: Some Theory for Kriging. Springer Science & Business Media, New York (1999)