A Comparison of Some Simple and Complex Surrogate Models: Make Everything as Simple as Possible?

Wim De Mulder
and Geert Molenberghs
and Geert Verbeke
Leuven Biostatistics and
Statistical Bioinformatics Centre
KU Leuven and Hasselt University, Belgium
Email: wim.demulder@cs.kuleuven.be,
geert.molenberghs@uhasselt.be,
geert.verbeke@med.kuleuven.be

Bernhard Rengs
and Thomas Fent
Wittgenstein Centre (IIASA, VID/ÖAW, WU)
VID/ÖAW
Vienna, Austria
Email: bernhard.rengs@oeaw.ac.at,
thomas.fent@oeaw.ac.at

***Abstract*—In this paper, we compare three surrogate models on a highly irregular, yet real-world data set. The three methods strongly vary in mathematical sophistication and computational complexity. First, inverse distance weighting, a very intuitive method whose single parameter can be readily determined via basic, albeit nonlinear, optimization. Secondly, radial basis function networks, for which some parameters can be determined via simple matrix algebra, although determination of other parameters require cluster analysis and nonlinear optimization. Thirdly, Gaussian process emulation, a statistical technique having a technical mathematical formulation and where specification of parameter values rely on complex and time-consuming optimization. It comes as a complete surprise that inverse distance weighting performs best on our complex data set. Our work encourages to moderate the extreme optimism about and overly use of very advanced methods. The commonplace a priori assumption that simple, intuitive methods underperform on complex data sets is not always justified, and such methods still have their place in the current era of highly advanced computational and mathematical models.**

Keywords–Gaussian process emulation; Radial basis function networks; Inverse distance weighting; Agent-based models; Cluster analysis.

I. INTRODUCTION

Ideally, any model of a real-world phenomenon explains observed data very well, has a rigorous mathematical formulation and is practically useful in the sense that it is computationally cheap to run it as many times as desired on a sequence of given inputs. However, as any researcher is aware of, these widely acclaimed properties are often conflicting by their very nature. As just one example, complex biological phenomena such as cancer can often be accurately represented using mathematically sound signaling network models in which the model equations contain parameters that are mechanistically descriptive of direct or indirect interactions in the system [1]. Yet this comes at the price of computational intractability, as the combinatorial explosion in the number of possible network models, which defines the solution space, makes model inference problems of this type NP-hard [2]. Thus, the repeated application of such computer models, for example by running the model under different parameter settings to find the parameter values that ensure the closest fit to empirical data, is practically limited to much fewer runs than accurate empirical analysis or other research objectives require.

One popular solution adopted by researchers who have to apply computationally expensive computer models for varying values of a certain variable (typically a variable in input or parameter space) is to approximate the complex model by a simpler, fast-running surrogate model [3]. Surrogate models are a particular instance of approximation methods that mimic the behavior of the simulation model while being computationally cheaper to evaluate. They are typically constructed by applying the expensive model to a limited number of values of the variable of interest, and then using the obtained input-output pairs, known as the training data set, to find a suitable fast-running approximation.

Agent-based models, which are briefly reviewed in Section II-E, often belong to the class of computationally expensive methods as they require the repeated simulation of nonlinear interactions between basic entities, and this during consecutive time steps, until a stable state is reached. Consequently, a researcher will often rely on surrogate techniques in applications where the agent-based model is to be applied to a very large number of input points. In this work, we compare the performance of three well-established surrogate techniques that have been developed in the literature, namely inverse distance weighting, radial basis function networks and Gaussian process emulation. These methods are very different in nature, e.g., Gaussian process emulation is a statistical technique that also provides a measure for the uncertainty in the approximation, while inverse distance weighting and radial basis function networks have been developed by nonstatisticians. However, in the context of this paper we will especially emphasize their disparity from another perspective, namely in terms of their complexity. The term complexity is used here to cover conceptual complexity as well as computational complexity related both to optimization of the parameters of the surrogate model and to the calculation of the output of the surrogate technique for a given input point. We elaborate on this term below. If we imagine all surrogate models lying on a spectrum defined in terms of complexity, then, loosely speaking, Gaussian process emulation and inverse distance weighting are at opposite ends while radial basis function networks have their

 ISBN: 978-1-61208-501-2

place somewhere in the middle. Their very different positions in this spectrum makes it an interesting research question of how this exactly translates into performance.

The various methods that play a role in our work are reviewed in Section II. First, the three above surrogate techniques mentioned above. Second, agent-based models, which are used as the case study system to be approximated by the considered surrogate methods. Finally, cluster analysis, a dimensionality reduction technique that is essential in our implementations of Gaussian process emulation and radial basis function networks, due to the fact that our data set to train these models is very large. The training data set is described in Section III-A. Section III describes implementation details of the surrogate models, as well as of the cluster analysis that was applied to our training data set. Results are presented in Section IV and an extensive discussion follows in Section V. The final section concludes and suggests some future research questions.

II. Related methodology

This section describes the various methods that play a role in this paper.

A. Gaussian process emulation

Gaussian process (GP) emulation provides an approximation, called the emulator, to a mapping $\nu : \mathbb{R}^n \rightarrow \mathbb{R}^m$. The rationale behind the development of GP emulation was to provide fast-running approximations to slow-running computer models, such that tasks requiring many executions of ν (e.g., calibration) can be performed with respect to the emulator rather than by relying on ν, thereby ensuring that the considered task ends within a reasonable time. For our case study below it holds that $m = 1$.

The steps in constructing an emulator are as follows. In the first step, it is assumed that nothing is known about ν. The value $\nu(\mathbf{x})$ for any $\mathbf{x}$ is then modeled via a Gaussian distribution with mean $m(\mathbf{x}) = \sum_{i=1}^{q} \beta_i \, h_i(\mathbf{x})$, where β_i are unknown coefficients and where h_i represent linear regression functions. The covariance between $\nu(\mathbf{x})$ and $\nu(\mathbf{x}')$, with $\mathbf{x}$ and $\mathbf{x}'$ being arbitrary input vectors in $\mathbb{R}^n$, is modeled as $\mathrm{Cov}\Big(\nu(\mathbf{x}), \nu(\mathbf{x}') \,|\, \sigma^2\Big) = \sigma^2 \, c(\mathbf{x}, \mathbf{x}')$, where σ^2 denotes a constant variance parameter and where $c(\mathbf{x}, \mathbf{x}')$ denotes a function that models the correlation between $\nu(\mathbf{x})$ and $\nu(\mathbf{x}')$. We adopt the most common choice for c:

$$c(\mathbf{x}, \mathbf{x}') \quad = \quad \exp\Big[-\sum_i \Big((x_i - x_i')/\delta_i\Big)^2\Big] \qquad (1)$$

with x_i and x_i' as the ith component of $\mathbf{x}$ and $\mathbf{x}'$, respectively, and where the δ_i represent parameters that can be optimized via maximum likelihood [4]. In plain words, before the training data set is taken into account it is assumed that $\nu(\mathbf{x})$ can be well approximated as a linear combination of user-chosen regression functions h_i and that a Gaussian distribution is a good model to represent the uncertainty in this approximation. In the second step, training data $(\mathbf{x}_1, \nu(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, \nu(\mathbf{x}_n))$ are taken into account. The information encapsulated in the training data set is used to improve the initial approximation as well as the model for the uncertainty in this approximation. This is done via a Bayesian analysis. It can be shown that this results in adjusting the Gaussian distributions to Student's t-distributions [5]. The mean of the Student's t-distribution in $\mathbf{x}$ is then considered the best approximation to $\nu(\mathbf{x})$. Therefore, we refer to this mean as $\hat{\nu}(\mathbf{x})$. It is given by

$$\hat{\nu}(\mathbf{x}) \quad = \quad m(\mathbf{x}) + U^T(\mathbf{x}) A^{-1} ([\nu(\mathbf{x}_1), \ldots, \nu(\mathbf{x}_n)]^T - H\boldsymbol{\beta})$$

with

$$\begin{aligned} \boldsymbol{\beta} &= (\beta_1, \ldots, \beta_q)^T \\ H &= \begin{bmatrix} h_1(\mathbf{x}_1) & \ldots & h_q(\mathbf{x}_1) \\ \ldots & & \\ h_1(\mathbf{x}_n) & \ldots & h_q(\mathbf{x}_n) \end{bmatrix} \end{aligned}$$

and where $U(\mathbf{x})$ contains the correlations, as given by (1), between $\mathbf{x}$ and each of the training data points $\mathbf{x}_i$, and where A is the correlation matrix, containing the correlations between $\mathbf{x}_i$ and $\mathbf{x}_j$ for $i, j = 1, \ldots, n$. It is worth paying some attention to the role of $U(\mathbf{x})$. The ith element of this vector is $c(\mathbf{x}_i, \mathbf{x})$. From (1) it follows that the closer $\mathbf{x}$ to $\mathbf{x}_i$ in input space, the larger $c(\mathbf{x}_i, \mathbf{x})$. Therefore, the vector $U(\mathbf{x})$ is a weight vector, meaning that the closer $\mathbf{x}$ to $\mathbf{x}_i$, the higher the influence of the ith training data point in determining the approximation to $\nu(\mathbf{x})$.

The formula for $\hat{\nu}(\mathbf{x})$ above shows that the Bayesian analysis adds a correction term to the prior mean $m(\mathbf{x})$ by taking into account the information encapsulated in the training data set. A crucial entity in the correction term is A^{-1}, the inverse of the correlation matrix. We found that the inversion operation for our very large training data set, described in Section III-A below, is computationally intractable. We resolved this by dividing the training data set into smaller subsets using cluster analysis (a method briefly reviewed in Section II-D) and then training an emulator for each small subset. Approximations generated by these emulators can then be combined into a final, unique approximation as outlined in Section III-C.

Values for the β_i and for σ^2 can be determined by optimization principles in Hilbert space, and analytical formulae are given in [5] and [6]. For a more detailed account on GP emulation we also refer to the cited works.

GP emulation is an advanced method, having a very sound mathematical basis, where Bayesian statistics, Hilbert space methods and approximation theory all play a role [7]. A thorough understanding thus requires knowledge of several advanced mathematical concepts. As so often happens, the conceptual complexity translates into computational intractability. In particular, optimizing the parameters δ_i, β_i and σ^2 can be very time-consuming. For example, each parameter δ_i corresponds to one input component and the presence of even a few input components often makes this multidimensional nonlinear optimization task computationally very hard. A final aspect of complexity is the computational complexity of evaluating $\hat{\nu}(\mathbf{x})$, which is dominated by the evaluation of the correction term, resulting in an overall computational complexity of $O(n^3)$.

B. Radial basis function networks

A radial basis function network (RBFN) relies on so-called basis functions ϕ_j that are radially symmetric around a chosen center $\boldsymbol{\mu}_j \in \mathbb{R}^n$. A common choice for ϕ_j is

$$\phi_j(\mathbf{x}) \quad = \quad \exp\left(-\left(\frac{||\mathbf{x} - \boldsymbol{\mu}_j||}{\rho}\right)^\tau\right) \qquad (2)$$

 ISBN: 978-1-61208-501-2

where $\rho > 0, \tau > 0$ are parameters and where $||.||$ represents the Euclidean norm. The parameter ρ is often called the width of the basis function and determining a value for it amounts to finding a compromise between locality and smoothness of $\hat{\nu}(\mathbf{x})$ given by (3) below [8].
The points $\boldsymbol{\mu}_j$ are called the centers of the basis functions and are typically determined by one of the heuristic methods described in [9]. One popular way, which we take here, is to use cluster analysis to determine the number of centers r and to specify a value for $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_r$, as outlined in Section II-D. These centers have an intuitive interpretation: they can be considered prototypical elements of the different parts of the input space. That is to say, cluster analysis divides the input space into smaller regions, ideally such that ν has a distinctive behavior in each region, and the distance from a given input point $\mathbf{x}$ to the jth region is calculated as $||\mathbf{x} - \boldsymbol{\mu}_j||$.
The value of ν in any $\mathbf{x}$ is then approximated as a shifted linear combination of the outputs of the basis functions:

$$\hat{\nu}(\mathbf{x}) \quad = \quad \gamma + \sum_{j=1}^{r} w_j \phi_j(\mathbf{x}) \tag{3}$$

where γ and $\mathbf{w} = [w_1, \ldots, w_r]^T$ are parameters that are typically determined as a least-squares solution to the minimization problem $\min \sum_{i=1}^{n} (\nu(\mathbf{x}_i) - \hat{\nu}(\mathbf{x}_i))^2$, as described in [9]. Notice that the centers $\boldsymbol{\mu}_j$ are obtained via unsupervised learning, while the other parameters involve supervised learning.
From a conceptual point of view, an RBFN is an artificial neural network where the centers $\boldsymbol{\mu}_j$ play the role of the nodes of the hidden layer [8]. The activation of the jth hidden node is determined by $\phi_j(\mathbf{x})$ and the closer an input $\mathbf{x}$ to $\boldsymbol{\mu}_j$, the more the jth hidden node is triggered. The output layer then combines the activations of all hidden nodes into one final value that is used as the approximation to $\nu(\mathbf{x})$. This is illustrated in Figure 1 where several output components are allowed.

Informally speaking, the complexity of RBFN is lower than that of GP emulation. A good conceptual understanding does not require more than a basic knowledge of artificial neural networks and of approximation theory. While an understanding of artificial neural networks helps to internalize a visual representation of a RBFN, some insight in approximation theory, e.g., polynomial approximation and Fourier series, is needed to appreciate that a complex, but reasonable, function can often be approximated in terms of simpler functions, provided that certain conditions are fulfilled. As for the computational complexity of optimizing the parameters, three stages are involved. First, centers are determined using, e.g., k-means. While the general version of k-means is NP-hard, there are numerous approximation algorithms, some of which run in time polynomial in n and r [10]. The second optimization task concerns optimizing a nonlinear function to determine ρ and τ. As only two parameters are involved, this task can often be completed in a very reasonable time. Some researchers prefer to associate a different ρ with each ϕ_j, resulting in the much harder task of optimizing a nonlinear function depending on $r + 1$ variables. Alternatively, there are researchers who simply assign a heuristic value to ρ, e.g., $\rho = d_{\max}/\sqrt{2r}$, with $d_{\max}$ being the maximum distance between all pairs of centers [11]. Finally, determining γ and $\mathbf{w}$ comes at almost no computational cost, as they are the solution of a least-squares problem which is essentially obtained by inverting a matrix,

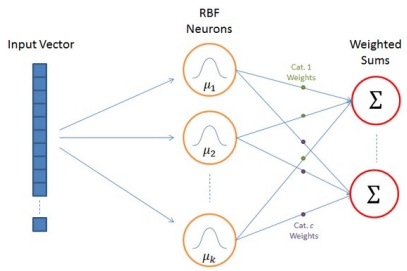

Figure 1. RBFN as artificial neural network [12]

and this matrix is only of dimension $(r + 1) \times (r + 1)$, i.e., only quadratic in the number of clusters. Since the essence of cluster analysis is dimensionality reduction, r is typically much smaller than n. It is seen from (3) that the computational complexity of evaluating $\hat{\nu}(\mathbf{x})$ is only $O(r^2)$.

C. Inverse distance weighting

Inverse distance weighting (IDW) is an approximation method in a metric space setting, originally developed by Shepard in the context of spatial analysis and geographic information systems [13]. Although the method is very simple, it is still used nowadays, for example to estimate spatial rainfall distribution, where the unknown spatial rainfall data is approximated from the known data of sites that are adjacent to the unknown site (see, e.g., [14] and [15]).
IDW approximates the unknown value of ν in a given point $\mathbf{x}$ as:

$$\begin{aligned} \hat{\nu}(\mathbf{x}) \quad &= \quad \sum_{i=1}^{n} \frac{w_i(\mathbf{x})}{\sum_{j=1}^{n} w_j(\mathbf{x})} \nu(\mathbf{x}_i) \qquad \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \ \forall i \quad (4) \\ &= \quad \nu(\mathbf{x}_i) \qquad\qquad\qquad\qquad \text{otherwise} \qquad\qquad (5) \end{aligned}$$

with

$$w_i(\mathbf{x}) \quad = \quad \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^{\alpha}}$$

where d is any metric and where α is a constant larger than zero.

The above formulation shows that IDW is very intuitive: obtain an approximation for the output in $\mathbf{x}$ as a weighted average of known outputs $\nu(\mathbf{x}_i)$, and give more weight to outputs that have been mapped from training data input points that are closer to $\mathbf{x}$. No advanced mathematical principle is involved. Optimization can be avoided by simply choosing $\alpha = 1$. Alternatively, α can be determined as the solution to the same minimization problem as in RBFN, namely $\min \sum_{i=1}^{n} (\nu(\mathbf{x}_i) - \hat{\nu}(\mathbf{x}_i))^2$. Although this is a nonlinear optimization task, there is only one parameter, α, involved. The computational complexity of evaluating $\hat{\nu}(\mathbf{x})$ is $O(n^2)$, which is higher than that of a RBFN, but provided that the number of training data points is not extremely large, this evaluation can be done very fast as only simple multiplications and additions are involved.

D. Cluster analysis

Cluster analysis is the unsupervised partitioning of a data set into groups, also called clusters, such that data elements which are members of the same group have a higher similarity

 ISBN: 978-1-61208-501-2

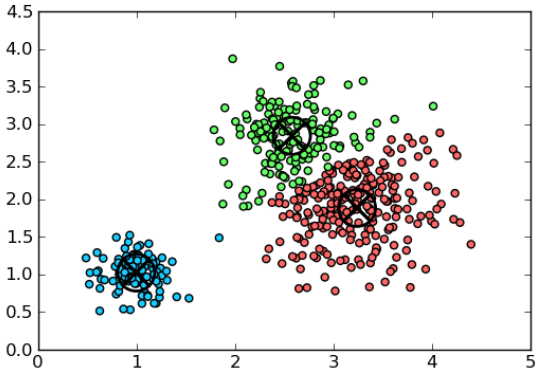

Figure 2. Illustration of k-means with $k = 3$ [18]

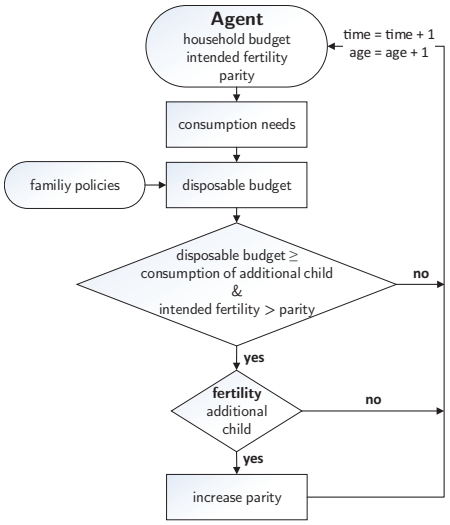

Figure 3. The decision making process in a household

than data elements which are members of different groups [16]. Similarity is typically expressed in terms of a user-defined distance measure, such as the commonly used Euclidean distance. Arguably the best known clustering algorithm is k-means [17], an iterative algorithm that not only determines clusters but also centers (sometimes called centroids) for the obtained clusters. Given the number of clusters, k, and a data set $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the algorithm chooses the clusters $C_i (i = 1, \ldots, k)$ such that $\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \boldsymbol{\mu}_i||^2$ is minimal, where $\boldsymbol{\mu}_i$ is the center of the ith cluster. The clusters C_i are simply subsets of the given data set with $C_i \cap C_j = \emptyset$ if $i \neq j$ and $\cup_{i=1}^{n} C_i = D$. Each center $\boldsymbol{\mu}_i$ readily follows from the construction of the clusters, as it is the average of all points belonging to C_i. An illustration is provided in Figure 2. We have used k-means for two purposes. First, due to the very large size of our training data set and the related instabilities in inverting the correlation matrix, direct application of GP emulation turned out to be infeasible. Thus we applied cluster analysis, thereby creating smaller training data sets, each of them being a subset of the original training data set. A separate emulator was then trained with each of these small data sets. Second, the number of clusters that was determined with k-means gave at once the number of basis functions in our RBFN, i.e. the parameter r in (3), and the cluster centers were used as the centers of the basis functions, i.e., the parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_r$ in (2). The result of the performed cluster analysis is outlined in Section III-B.

E. Agent-based models

An agent-based model (ABM) is a computational model that simulates the behavior and interactions of autonomous agents. A key feature is that population level phenomena are studied by explicitly modeling the interactions of the individuals in these populations [19][20]. The systems that emerge from such interactions are often complex and might show regularities that were not expected by researchers in the field who solely relied on their background knowledge about the characteristics of the lower-level entities to make predictions about the higher-level phenomena. ABMs are especially popular among sociologists who model social life as interactions among adaptive agents who influence one another in response to the influence they receive [21].

In previous work, we developed an agent-based model to analyze the effectiveness of family policies under different assumptions regarding the social structure of a society [22]. The agents represent the female partner in a household. They are heterogeneous with respect to age, household budget, parity, and intended fertility. A network of mutual links connects the agents to a small subset of the population to exchange fertility preferences. Figure 3 illustrates the decision making process in a household. The agents are endowed with a certain budget of time and money which they allocate to satisfy their own and their children's needs. We assume that the agent's and their children's consumption levels depend on the household budget but increase less than linearly with household budget. Thus, wealthier households have a higher savings rate. If the household's intended fertility exceeds the actual parity and the disposable budget suffices to cover the consumption needs of another child, the household is subject to the appropriate age-specific fertility. Hence the household is exposed to having an additional child depending on the outcome of a random process. If an additional child is born, other agents may update their intended fertility.

We considered two components of family policies: 1. the policymaker provides a fixed amount of money or monetary equivalent per child to each household and 2. a monetary or non-monetary benefit proportional to the household income is transferred to the household. Any policy mix greater than zero supports the household in covering the dependent children's consumption needs.

The output at the aggregate level that is produced by the ABM consists of the cohort fertility, the intended fertility, and the fertility gap. The inputs include the level of fixed and income-dependent family allowances, denoted by b^f and b^v, and parameters that determine the social structure of a society, such as a measure for the agents' level of homophily α, and the strength of positive and negative social influence, denoted by pr_3 and pr_4 resp. The results of the application of our ABM and the related sociological findings can be found in the cited work. The purpose of this paper is to generate a training data set by application of the ABM to selected input points and then using this data set to empirically compare GP emulation, RBFN and IDW. The training data set we created is described

 ISBN: 978-1-61208-501-2

in the next section.

III. IMPLEMENTATION

We describe our data set and the implementation details of the methods that are compared to each other.

A. Data set

The input variables of our ABM are given equidistant values from the input domain and the ABM is applied to generate the corresponding outputs. As input domain, we considered the variables b^f, b^v, α, pr_3 and p_4, a selection of the larger amount of variables that were used in the ABM. These five variables were found to have the greatest influence on the outcomes. On the output side, we restrict attention to one variable, namely cohort fertility. The ABM was applied to 10,732 vectors in the input domain, which gives a very large training data set. A test data set containing 500 input–output pairs was generated to compare the performance of the three methods under study.

B. Cluster analysis

In clustering the training data set containing 10,732 data points we performed trial and error to determine a suitable number of clusters. A highly desired objective is that the clusters are kept relatively small, since an emulator is to be trained for each cluster and numerical instabilities arise when the number of training data points that are used in constructing an emulator becomes large, as explained in Section II-A. In previous work [23], we found that numerical instabilities can be avoided by constraining the result of k-means such that the largest cluster has no more than about 500 training data points. The main characteristics of a suitable set of clusters that was obtained in this previous work are as follows: 34 clusters, largest cluster size equal to 500, smallest cluster size equal to 15.

C. Description of the implemented methods

We implemented 8 methods instantiated from the domains of IDW, GP emulation and RBFN.

The first two methods are based on GP emulation and cluster analysis. As outlined above, a GP emulator was trained for each of the 34 clusters, each containing a part of the full training data set. The first method, hereafter referred to as GP-closest, computes an approximation to $\nu(\mathbf{x})$ by finding the cluster that is closest to $\mathbf{x}$ and taking the output of the corresponding emulator in $\mathbf{x}$ as the sought-after value. The distance between a point and a cluster is hereby taken as the Euclidean distance between that point and the center of the given cluster. The second GP emulation method, which we call GP-average, determines the approximation as a weighted average of the outputs of all emulators in the given input point. Each weight is taken as an inverse measure of the distance between the given input point and the specified cluster. That is, we use the formulation of IDW, given by (4)-(5), choosing $\alpha = 1$ in (6), and where $\nu(\mathbf{x}_i)$ is replaced by the output of the emulator corresponding to the ith cluster.

The next five methods are RBFNs, with different values for the parameters ρ and τ in (2). The chosen values are displayed in Table I. To make reference to a specific method convenient, we labeled these methods RBFN-1,..., RBFN-5.

The 8th and last method is a simple IDW, as given by (4)-(5).

TABLE I. PARAMETER VALUES FOR THE DIFFERENT RBFNS.

Method	ρ	τ
RBFN-1	1	2
RBFN-2	1	1
RBFN-3	1	0.5
RBFN-4	2	2
RBFN-5	4	2

IV. RESULTS

We now discuss the measure that is used to evaluate our results and we analyze the outputs of this measure.

A. Evaluation measure

Given a test input point $\mathbf{x}$ with corresponding true output $\nu(\mathbf{x})$ and an approximation $\hat{\nu}(\mathbf{x})$ produced by one of the methods described in the previous section, we evaluate the quality of the approximation as the relative difference between $\nu(\mathbf{x})$ and $\hat{\nu}(\mathbf{x})$, as follows:

$$RD(\mathbf{x}) \quad = \quad \Big| \frac{\hat{\nu}(\mathbf{x}) - \nu(\mathbf{x})}{1/2(\hat{\nu}(\mathbf{x}) + \nu(\mathbf{x}))} \Big|$$

The average relative difference for a particular method, denoted ARD, is then the average of $RD(\mathbf{x})$ over all 500 test points.

B. Results and analysis

The ARD is calculated for each of the methods described in Section III-C and results are shown in Table II.

Let us first restrict attention to the RBFN methods. The results show that the choice of parameter values is of crucial importance for the performance. The ARD varies from 0.165 for RBFN-1 to 0.235 for RBFN-3. The fact that the best result is obtained for RBFN-1 where $\tau = 2$ supports the common practice to set τ at this value in applications. On the other hand, the parameter ρ should be part of the learning process, as results for $\tau = 2$ widely vary, from 0.165 to 0.227. However, the price to be paid for the resulting substantial rise in performance is high, as the determination of a suitable ρ amounts to a nonlinear optimization task.

Turning attention to a global comparison, a remarkable observation is that all ARDs are high, with at least a 14.6% difference, on average, between the approximations and the true outputs. Since the ARD is high for all considered methods and since these methods come from very different domains, varying in mathematical and computational complexity, it is conceivable that the training data set is highly complex, displaying many irregularities. This makes it even more interesting to compare the methods, as empirical comparisons are too often based on either artificially created data sets or real-world data sets that are well understood and not exceedingly complex. A multidimensional and exceptionally complex, but still real-world, data set as ours is useful to evaluate performance in a less typical situation than those appearing in the mainstream literature and to check the limits of surrogate modeling techniques.

However, the most striking observation is that IDW, by far the simplest of the considered techniques, is also by far the best method. The difference of the ARD of IDW and that of the second best performing, RBFN-4, is substantial. This is not only surprising because of the simplicity of IDW,

 ISBN: 978-1-61208-501-2

but also because the training data set is very complex. One would expect an irregular data set to call for more advanced techniques.

TABLE II. PERFORMANCE OF THE IMPLEMENTED METHODS ON THE TEST DATA SET.

Method	*ARD*
GP-closest	0.212
GP-average	0.293
RBFN-1	0.165
RBFN-2	0.197
RBFN-3	0.235
RBFN-4	0.190
RBFN-5	0.227
IDW	0.146

V. DISCUSSION

Above, we have compared three surrogate models on a highly complex, irregular real-world data set. The three methods strongly vary in mathematical sophistication and computational complexity. IDW is a very simple and intuitive method, requiring only limited optimization efforts, and the resulting approximation function can be quickly evaluated on a given input point. RBFNs require to choose basis functions and to determine suitable centers for these functions. The centers of the basis functions are typically obtained by applying k-means, an algorithm that in many practical applications turns out to converge very fast. If this is not the case, one may decide to rely on one of the approximation algorithms that have been developed that run in polynomial time. Determination of the parameters ρ and τ associated to the basis functions require nonlinear optimization. Although nonlinear optimization is known to be computationally hard, the fact that only two parameters are involved ensures that this task can typically still be fulfilled in a very reasonable time. Some researchers prefer to increase flexibility by associating a different ρ to each basis function, which drastically increases computational complexity. At the other extreme, it is also possible to assign ρ a fixed, heuristic value. The remaining parameters to be optimized in RBFNs are the weights in the linear combination of the basis functions, which can be determined quickly via the inversion of a matrix of, for most applications, small dimensions. GP emulation is a statistical technique and has a deep and refined mathematical foundation. Optimization is typically very complex and time-consuming: determination of the parameters δ_i calls for nonlinear optimization of a likelihood function, while the posterior mean of the generated Student's t-distribution, to be used as approximation for the output in a given input point, requires inverting a correlation matrix having dimensions that are quadratic in the number of training data points.
If we vaguely define the overall complexity of a method as the sum of its conceptual complexity, the computational complexity of optimizing the parameters involved and the computational complexity of calculating the approximate output for a given input point, the above discussion shows that the arrow of increasing complexity runs along IDW, RBFN and GP .

One is inclined to assume that the most advanced method will perform best. In fact, it has become commonplace among researchers to select a method from the set of advanced approximation or machine learning techniques whenever confronted with a complex data set (as an illustration, we refer to [24] and [25]). Furthermore, when experimentally comparing machine learning algorithms, it is typically advanced methods that are given a role in this process, leaving out simple techniques under the a priori assumption that these will underperform anyway (see, e.g., [26] and [27]). And new machine learning algorithms appearing in the literature are almost always a sophistication of an already sophisticated algorithm (examples are [28] and [29]). That is, *most advanced* and *best* are used almost interchangeably.

To many researchers it will thus come as a complete surprise that a very simple method like IDW performed better than two much more advanced approximation methods on our complex training data set generated by an ABM.

VI. CONCLUSION AND FUTURE WORK

Although it might be objected that the three considered surrogate models were applied to only one data set, our answer is that the purpose of our work is obviously *not* to demonstrate that simpler methods are better than more advanced ones. Rather, we have shown that *there are* complex problems for which simple methods work better than much more advanced techniques.

Therefore, our work is meant as a cautionary note that the meaning of the term *best approximation method* is still that it performs best on a given application. Although intuition says that a more advanced technique will often deliver better results, there is no strict guarantee that this will be the case for a specific problem at hand. Indeed, methods having a sophisticated mathematical foundation always rely on assumptions about the given training data set and the function to be approximated. It is all too often ignored that these assumptions might not hold in practice or too quickly assumed that deviations from mathematical assumptions are small enough, and that the considered method will still perform properly.

Our findings should not be seen as a discouragement to apply advanced methods. Rather, we interpret our results as an invitation to always evaluate a very simple method on a given data set alongside one or more advanced techniques. This comes at almost no cost, as simple methods like IDW can be easily implemented and are computationally very cheap. It is also important to notice that IDW does not make any assumption about the given data set, thus applying it in a setting where it is strictly speaking not applicable (as is often the case for advanced methods) is not an issue. When such simple methods do perform better than other, more sophisticated methods, their return on investment is therefore incredibly high. As often attributed to Einstein: "everything should be made as simple as possible ..." This is, of course, not to say that advanced methods are useless. In many situations a more complex problem cannot be handled by simple intuition and then a more advanced method should be applied. Or, as Einstein continues, "... but not simpler".

Possible future research questions include:

- Can we improve the RBFNs by associating a different ρ with each basis function to the extent that these RBFNs become better than IDW?
- What specific characteristics of the data set are responsible for the fact that IDW performs better than GP emulation and RBFN?

 ISBN: 978-1-61208-501-2

- Are the obtained results sensitive to the number of clusters?

REFERENCES

[1] E. Molinelli et al., "Perturbation biology: inferring signaling networks in cellular systems," PLoS Computational Biology, vol. 9, 2013.

[2] V. Chandrasekaran, N. Sebro, and P. Harsha, "Complexity of inference in graphical models ," in Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, 2008.

[3] S. Koziel and L. Leifsson, Eds., Surrogate-based modeling and optimization. Springer, 2013.

[4] I. Andrianakis and P. G. Challenor, "The effect of the nugget on Gaussian process emulators of computer models," Computational Statistics and Data Analysis, vol. 56, 2012, pp. 4215–4228.

[5] A. O'Hagan, "Bayesian analysis of computer code outputs: A tutorial," Reliability Engineering & System Safety, vol. 91, 2006, pp. 1290–1300.

[6] J. Oakley and A. O'Hagan, "Bayesian inference for the uncertainty distribution of computer model outputs," Biometrika, vol. 89, 2002, pp. 769–784.

[7] W. De Mulder, G. Molenberghs, and G. Verbeke, "A mathematical review of the standard Gaussian process emulator," *Submitted to:* International Statistical Review.

[8] N. Benoudjit and M. Verleysen, "On the kernel widths in radial-basis function networks," Neural Processing Letters, vol. 18, 2003, pp. 139–154.

[9] C. Bishop, Ed., Neural networks for pattern recognition. Clarendon Press, 1996.

[10] M. Song and S. Rajasekaran, "Fast k-means algorithms with constant approximation," in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[11] S. Haykin, Ed., Neural networks: a comprehensive foundation. Prentice-Hall, 1999.

[12] C. McCormick, "Radial basis function network (RBFN) tutorial," http://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/, 2013.

[13] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in Proceedings of the 1968 ACM National Conference.

[14] P. Goovaerts, "Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall," Journal of Hydrology, vol. 228, 2000, pp. 113–129.

[15] F. Chen and C. Liu, "Estimation of the spatial rainfall distribution using inverse distance weighting (IDW) in the middle of Taiwan," Paddy and Water Environment, vol. 10, 2012, pp. 209–222.

[16] A. Jain, M. Murty, and P. Flynn, "Data clustering: a review," ACM Computing Surveys, vol. 31, 1999, pp. 264–323.

[17] A. Jain and R. Dubes, Eds., Algorithms for clustering data. Prentice Hall College Div, 1988.

[18] M. Pacula, "K-means clustering example," http://blog.mpacula.com/2011/04/27/k-means-clustering-example-python/.

[19] N. Gilbert, Ed., Agent-based models: quantitative applications in the social sciences. SAGE Publications, Inc, 2007.

[20] F. C. Billari, T. Fent, A. Prskawetz, and J. Scheffran, Eds., Agent–Based Compuational Modelling: Applications in Demography, Social, Economic, and Environmental Sciences, ser. Contributions to Economics. Springer, 2006.

[21] M. Macy and R. Willer, "From factors to factors: computational sociology and agent-based modeling," Annual Review of Sociology, vol. 28, 2002, pp. 143–166.

[22] T. Fent, B. Aparicio Diaz, and A. Prskawetz, "Family policies in the context of low fertility and social structure," Demographic Research, vol. 29, 2013, pp. 963–998.

[23] W. De Mulder, B. Rengs, G. Molenberghs, T. Fent, and G. Verbeke, "Statistical emulation applied to a very large data set generated by an agent-based model," in Proceedings of the Seventh International Conference on Advances in System Simulation.

[24] A. Khazaee, A. Ebrahimzadeh, and A. Babajani-Feremi, "Application of advanced machine learning methods on resting-state fMRI network for identification of mild cognitive impairment and Alzheimers disease," Brain Imaging and Behavior, 2015, pp. 1–19.

[25] J. Behmann, A. Mahlein, T. Rumpf, and L. Plümer, "A review of advanced machine learning methods for the detection of biotic stress in precision crop protection," Precision Agriculture, vol. 16, 2015, pp. 239–260.

[26] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five learning algorithms for practical IP traffic flow classification," ACM SIGCOMM Computer Communication Review, vol. 36, 2006, pp. 7–15.

[27] A. Morton, E. Marzban, G. Giannoulis, A. Patel, R. Aparasu, and I. Kakadiaris, "A comparison of supervised machine learning techniques for predicting short-term in-hospital length of stay among diabetic patients," in 13th International Conference on Machine Learning and Applications.

[28] I. Rojas et al., "A new radial basis function networks structure: application to time series prediction," in Proceedings of the International Joint Conference on Neural Networks, 2000, pp. 449–454.

[29] X. Hong and S. Billings, "Dual-orthogonal radial basis function networks for nonlinear time series prediction," Neural Networks, vol. 11, 1998, pp. 479–493.

 ISBN: 978-1-61208-501-2