An interpretation of radial basis function networks as zero-mean Gaussian process emulators in cluster space

An interpretation of radial basis function networks as zero-mean Gaussian process emulators in cluster space

Wim De Mulder[1], Geert Molenberghs*[2,1], and Geert Verbeke[1,2]

[1]I-BioStat, KU Leuven, Leuven, Belgium
[2]I-BioStat, Universiteit Hasselt, Hasselt, Belgium

Abstract

Emulators provide approximations to computationally expensive functions and are widely used in diverse domains, despite the ever increasing speed of computational devices. In this paper we establish a connection between two independently developed emulation methods: radial basis function networks and Gaussian process emulation. The methodological relationship is established by starting from the observation that the concept of correlation between random variables in Gaussian process emulation can be interpreted as a correlation function applied to points in input space. This correlation function is then extended to apply to clusters, i.e. to sets of points. It is then shown that the extended Gaussian process emulation method is equivalent to radial basis function networks, provided that the prior mean in Gaussian process emulation is chosen zero. This elegant connection might increase understanding of the principles of both types of emulation, and might act as a catalyst for mutually beneficial research in emulation domains that were hitherto considered independent.

Keywords— Radial basis function networks; cluster analysis; Gaussian process emulation; correlation function.

1 Introduction

Bertrand Russell once expressed the importance of approximations in science as follows: "Although this may seem a paradox, all exact science is dominated by the idea of approximation". The problem of approximative determination of a given quantity is one of the oldest challenges of mathematics. As a prime example, the formula for approximating the square root of a number

*Corresponding author. Email address: geert.molenberghs@uhasselt.be

is usually attributed to the Babylonians [1]. There can be several good reasons to be satisfied with an approximation to the function or quantity of interest. Emulators, also known as surrogate models, response surface models or metamodels, are a class of models that have been developed to provide approximations to models that are computationally too expensive to be used directly in a certain analysis [2]. Emulators are frequently used in, e.g., engineering, where simulation models are considered a key factor to understanding complex system behavior [3]. Simulation models are typically computationally expensive, requiring minutes if not hours or days to generate a single simulation outcome corresponding to a given input. If a certain study requires to obtain simulation outcomes for a very large number of inputs, it is convenient to approximate the simulation model with an emulator, which is supposed to generate an approximation to the output in the given input point very fast. We adopt the common practice to refer to the model that is approximated as the simulator.

Emulators are typically built in a purely data driven manner, i.e., their construction solely relies on input-output pairs obtained from the simulator. In fact, the rationale behind emulation can be summarized as follows. First, given a simulator $\nu : X \to Y$, with X and Y certain sets, apply the computationally expensive simulator to a limited number of carefully chosen points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in X$, obtaining outputs $\nu(\mathbf{x}_1), \ldots, \nu(\mathbf{x}_n)$. Secondly, use these input-output pairs to optimize the parameters of the emulator. Finally, use the computationally cheap emulator to perform an analysis for which many, i.e., much more than n, simulation outcomes are needed. We refer to the set of points $\top = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ as the training data set. The corresponding vector of output values is denoted as $N(\top) = [\nu(\mathbf{x}_1), \ldots, \nu(\mathbf{x}_n)]^T$. The simulators ν we consider in this work are arbitrary mappings from $\mathbb{R}^p$ to $\mathbb{R}$, for some $p \in \mathbb{N}$.

2 Overview of the paper

In this work we establish an interesting and elegant relationship between radial basis function networks and Gaussian process emulation, two emulation methods that have been developed independently and for which research still proceeds along separate lines. A brief review on radial basis function networks and on Gaussian process emulation is provided in Sections 3 and 4 respectively. In Section 5 we stress the fact that correlation functions, used in Gaussian process emulation, and radial functions, used in radial basis function networks, share similarities, and we consider a general class of functions that can serve as correlation function and, at the same time, as radial function. Section 6 introduces two metric spaces, one where points are the main entities and one where clusters play the main role, called point space and cluster space, respectively. This formalization is used in Section

7 to develop Gaussian process emulators in cluster space, by extending the notion of correlation in point space to correlation in cluster space. In Section 8 we then establish an elegant relationship between Gaussian process emulators in cluster space and radial basis function networks. One particular use of the established relationship is the determination of a confidence interval for radial basis function networks, which is discussed in Section 9. This particular use is then illustrated in Section 10.

3 Brief review on radial basis function networks

A radial basis function network (RBFN) is an emulator that is composed of radial functions [4]. A radial function is defined as a function ϕ that is radially symmetric around some point $\boldsymbol{\mu}$ in input space. One commonly used type of radial functions are Gaussian functions [5], e.g.

$$\phi(\mathbf{x}, \boldsymbol{\mu}) \;=\; \exp\Big(-\gamma \,\big|\big|\mathbf{x} - \boldsymbol{\mu}\big|\big|^2\Big) \tag{1}$$

where $\gamma > 0$ is a parameter, and where $\big|\big|.\big|\big|$ denotes the Euclidean norm. The point $\boldsymbol{\mu} \in \mathbb{R}^p$ is conveniently called the center of the basis function.

The radial basis function approach relies on r radial functions with different centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_r$, where r is chosen by the user and which is typically much smaller than the training data set size n. One popular way to determine appropriate centers is by using a cluster analysis method, such as k-means [6]. Using k-means one can partition the training data set $\top$ into r disjoint clusters, where each cluster has an associated centroid that is considered a prototype of the cluster. It is intuitive to use these centroids as the centers of the basis functions.

A RBFN then approximates $\nu(\mathbf{x})$ as

$$\hat{\nu}(\mathbf{x}) \;=\; \sum_{j=1}^{r} w_j \, \phi(\mathbf{x}, \boldsymbol{\mu}_j) \tag{2}$$

where $w_1, \ldots, w_r$ are parameters. It is convenient to introduce the vector $W = [w_1, \ldots, w_r]^T$ and the $n \times r$ matrix Φ with elements $\Phi(i,j) = \phi(\mathbf{x}_i, \boldsymbol{\mu}_j)$. The parameter vector W is typically determined as the vector that minimizes the error function

$$E \;=\; \frac{1}{2} \sum_{k=1}^{n} \Big(\nu(\mathbf{x}) - \hat{\nu}(\mathbf{x})\Big)^2 \tag{3}$$

It is well known that an analytical expression for the optimal coefficient vector exists [7] and is given by

$$W \;=\; \Phi^{\dagger} \, N(\top) \tag{4}$$

where $\Phi^\dagger$ denotes the pseudo-inverse of Φ, provided that it exists.

A special case arises when $n = r$, such that there are n basis functions with centers $\boldsymbol{\mu}_j = \mathbf{x}_j$. In this case the optimal weight vector is given by $W = \Phi^{-1} N(\top)$, provided that Φ is non-singular.

RBFNs are especially well suited to handle very large training data sets since it relies on cluster analysis.

4 Brief review on Gaussian process emulation

A Gaussian process (GP) emulator is, as the name implies, an emulator. Over time researchers have developed variations on and extensions of the originally developed GP emulator. In this work we will restrict attention to the GP emulator as it was initially developed [8]. The essential idea behind GP emulation is to consider the output of ν in a given input $\mathbf{x}$ as the realization of a random variable $\zeta(\mathbf{x})$ in output space, which is justified by the fact that the output is unknown as long as the simulator has not been applied to the given input. This allows to model the output in $\mathbf{x}$ as a distribution, whose mean is then considered the best approximation for $\nu(\mathbf{x})$ and whose variance can be interpreted as a measure for the degree of uncertainty about the extent to which the mean agrees with the real output value $\nu(\mathbf{x})$.

A central concept in the formulation of GP emulation is the correlation matrix. The correlation matrix, denoted as A, is the matrix where each entry contains the correlation between two random variables in output space that are associated to inputs from the training data set. That is, entry $A(i, j)$ of this matrix equals the correlation between $\zeta(\mathbf{x}_i)$ and $\zeta(\mathbf{x}_j)$, corresponding to the inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ from the training data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Determining the correlation matrix requires the choice of a correlation function $c : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ that takes as input the two points from input space that are associated with the considered random variables, such that $A(i, j) = c(\mathbf{x}_i, \mathbf{x}_j)$. The Gaussian functions that are used in RBFNs are also commonly employed in GP emulation. Thus $c(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}, \mathbf{x}')$ is a typical choice.

We will make extensive use of the fact that the description of the concept of correlation in GP emulation allows an interpretation that is disconnected from the statistical meaning of correlation as an association measure between two random variables. Indeed, from the fact that c is a mapping from $\mathbb{R}^p \times \mathbb{R}^p$ to $\mathbb{R}$, it follows that the correlation function $c(\mathbf{x}, \mathbf{x}')$ does not make any explicit reference to the random variables $\zeta(\mathbf{x})$ and $\zeta(\mathbf{x}')$. Instead, it applies to points $\mathbf{x}$ and $\mathbf{x}'$ in input space.

The distributions of the random variables in output space are modeled as follows. In a first step the outputs are modeled via a Gaussian process [9]. A particular consequence is that each distribution in output space is considered

Gaussian. For a given input point $\mathbf{x}$, the mean $m(\mathbf{x}) = E[\zeta(\mathbf{x}) \,|\, \boldsymbol{\beta}]$ of the corresponding distribution in output space is modeled as a linear combination of q user-chosen regression functions h_j:

$$m(\mathbf{x}) \;=\; \sum_{j=1}^{q} \beta_j \, h_j(\mathbf{x}) \tag{5}$$

where $\boldsymbol{\beta}$ is a vector of parameters β_j. The variance of each Gaussian distribution is a parameter σ^2 that is constant over input space. Its value is estimated as

$$\hat{\sigma}^2 \;=\; \frac{N(\top)^T A^{-1} (I - HK)\, N(\top)}{n - q - 2} \tag{6}$$

By the assumption of Gaussian process, any finite collection of random variables in output space is considered to follow a multivariate Gaussian distribution. A special collection are the random variables associated to the training data set. Their joint Gaussian distribution has mean $[m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)]^T$ and covariance matrix $\sigma^2 A$. By introducing the matrix H with elements $H(i,j) = h_j(\mathbf{x}_i)$ the mean of this distribution can be conveniently written as

$$[m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)]^T \;=\; H\boldsymbol{\beta} \tag{7}$$

In a next step parameters are optimized and the Gaussian process is updated to a Student-t process [10] via a Bayesian analysis [11]. The mean of the Student's t-distribution associated with the input point $\mathbf{x}$ is then taken as the best approximation for $\nu(\mathbf{x})$ and is, therefore, denoted as $\hat{\nu}(\mathbf{x})$. It is given by [12]:

$$\hat{\nu}(\mathbf{x}) \;=\; m(\mathbf{x}) + \;<U(\mathbf{x}), A^{-1}(N(\top) - H\hat{\boldsymbol{\beta}})> \tag{8}$$

where $<.,.>$ denotes the Euclidean inner product and with $\hat{\boldsymbol{\beta}}$ an optimal value for $\boldsymbol{\beta}$, for example determined according to the maximum likelihood principle [12]. Furthermore, the vector $U(\mathbf{x})$ contains the correlations between $\zeta(\mathbf{x})$ and each $\zeta(\mathbf{x}_i)$:

$$U(\mathbf{x}) \;=\; [c(\mathbf{x}, \mathbf{x}_1), \ldots, c(\mathbf{x}, \mathbf{x}_n)]^T \tag{9}$$

We will refer to the quantity $\hat{\nu}(\mathbf{x})$ as the GP emulator. There also exists an explicit analytical expression for the posterior variance of the Student's t-distributions [12],

In this paper, we will pay particular attention to GP emulators for which the prior mean is zero, i.e. $q = 0$ and thus $H = 0$. We refer to these emulators as zero-mean GP emulators, and their description is a special case of (8), given by

$$\hat{\nu}(\mathbf{x}) \;=\; <U(\mathbf{x}), A^{-1} N(\top)> \tag{10}$$

It can be shown, using Bayesian principles, that the posterior variance in this special case is estimated as

$$V(\mathbf{x}) = \hat{\sigma}^2\Big[1 - U^T(\mathbf{x})A^{-1}U(\mathbf{x})\Big] \tag{11}$$

$$= \frac{N(\top)^T A^{-1}\, N(\top)\,\Big[1 - U^T(\mathbf{x})A^{-1}U(\mathbf{x})\Big]}{n-2} \tag{12}$$

where we used (6) with $q = 0$ and $H = 0$.

It is an important strength of GP emulation that it provides a measure of the variance, since this allows to construct a confidence interval $CI(\mathbf{x})$ around any approximation $\hat{\nu}(\mathbf{x})$. For example, a 95% confidence interval is given by

$$CI(\mathbf{x}) = [\hat{\nu}(\mathbf{x}) - 2\,V(\mathbf{x})\,,\, \hat{\nu}(\mathbf{x}) + 2\,V(\mathbf{x})] \tag{13}$$

However, GP emulation has as severe limitation that it can only be applied to reasonably small training data sets, because the method relies on inverting the $n \times n$ correlation matrix A, an operation that is computationally infeasible for a large number of training data points n. Compare this to RBFN, which relies on the pseudo-inverse of the $n \times r$ matrix Φ with $r \ll n$, where r refers to the number of clusters.

5 Introduction of a general class of functions that can be used as correlation function and radial function

Consider any metric space of the form $(\mathbb{R}^p, d)$. The introduction of a metric space is motivated by the interpretation of the correlation function in GP emulation as a function that applies to points in input space, see Section 4. Now, it is seen that the function $c(\mathbf{x}, \mathbf{x}') = \exp\Big(-\gamma \Big|\Big|\mathbf{x} - \mathbf{x}'\Big|\Big|^2\Big)$ can be seen as the function $\Gamma(r) = \exp(-r^2)$ applied to the distance $r = d(\mathbf{x}, \mathbf{x}')$, where d is defined as $d(\mathbf{x}, \mathbf{x}') = \sqrt{\gamma}\,\Big|\Big|\mathbf{x} - \mathbf{x}'\Big|\Big|$. It is easily verified that this indeed defines a distance measure, given that γ is positive. Thus c can be written as $c(\mathbf{x}, \mathbf{x}') = \Gamma(d(\mathbf{x}, \mathbf{x}'))$.

From now on we will let Γ denote *any* function for which $\Gamma(d(\mathbf{x}, \mathbf{x}'))$ and $\Gamma(d(\mathbf{x}, \boldsymbol{\mu}))$ define a suitable correlation function in GP emulation and a suitable radial function in RBFN resp., for a chosen distance measure d. We use the following shorthand notation:

$$\varsigma(\mathbf{x}, \mathbf{x}') = \Gamma(d(\mathbf{x}, \mathbf{x}')) \tag{14}$$

6 Cluster space and point space

In Section 3 it was outlined how cluster analysis might play an important role in RBFN, namely to determine the centers of the radial functions. To formalize this role a little bit more, we impose a certain structure on the clusters. We consider clusters, determined by any clustering algorithm, $C_1, \ldots, C_r$ with each $C_i \subseteq \mathbb{R}^p$ for which the following holds:

$$\begin{aligned} &1. \quad \top = \bigcup_{i=1}^{r} C_i \\ &2. \quad C_i \cap C_j = \emptyset \quad \text{if } i \neq j \end{aligned}$$

The two conditions together imply that each training data point belongs to exactly one cluster. Notice that any point $\mathbf{x} \notin \top$ can also be considered a cluster, namely the cluster $\{\mathbf{x}\}$. It will turn out useful to think of the input space as the following set of clusters Δ:

$$\Delta \;=\; \bigcup_{i=1}^{r} C_i \;\cup\; \bigcup_{\mathbf{x} \in \mathbb{R}^p} \{\mathbf{x}\} \tag{15}$$

Furthermore, it is convenient to have a notion of distance between two given clusters. A distance measure between clusters can be defined in several ways and we will generally denote such a measure as d_c. Some measures can be found in [13] and [14]. This provides us with a metric space (Δ, d_c) to which we will refer as the cluster space. The metric space $(\mathbb{R}^p, d)$, which provides another view on the input space in terms of vectors consisting of p components, is then conveniently called the point space.

7 Introducing cluster-based GP emulation

We now extend the GP emulation method by extending the notion of correlation in point space to correlation in cluster space. The extended GP emulator can then be seen as an emulator living in cluster space, and we will refer to it as a GP emulator in cluster space or, in short, a cluster-based GP emulator.

7.1 Introducing correlation between clusters

The purpose of this section is to define correlation between clusters, i.e., between members of (Δ, d_c), which extends correlation between points, i.e. between members of $(\mathbb{R}^p, d)$. In Section 5 we gave the following general definition for correlation between points: $\varsigma(\mathbf{x}, \mathbf{x}') = \Gamma(d(\mathbf{x}, \mathbf{x}'))$. It is then natural to define correlation between clusters C and C' from Δ as $\varsigma(C, C') = \Gamma(d_c(C, C'))$. We use the same notation ς to denote the correlation between

clusters, since the context will make it clear whether correlation between clusters or between points is considered. Although the given definition is intuitive, it will turn out useful to consider the distance d_c between certain transformations of C and C' instead of calculating this distance directly between C and C'. Therefore we introduce the function $G : \Delta \to \Delta$ and we define

$$\varsigma(C, C') \quad = \quad \Gamma(d_c(G(C), G(C'))) \tag{16}$$

7.2 Cluster-based GP emulation

Having at our disposal a correlation function between clusters, it becomes possible to extend some essential GP emulation concepts from point space to cluster space, namely the correlation matrix A and the vector $U(\mathbf{x})$. The correlation matrix has as natural counterpart in cluster space the matrix $\mathcal{A}$ with elements $\mathcal{A}(i,j) = \varsigma(C_i, C_j)$, where $\cup_i C_i = \top$. The vector $U(\mathbf{x})$ with components $\varsigma(\mathbf{x}, \mathbf{x}_i)$ is naturally extended to

$$\mathcal{U}(\mathbf{x}) \quad = \quad [\varsigma(\{\mathbf{x}\}, C_1), \ldots, \varsigma(\{\mathbf{x}\}, C_r)]^T \tag{17}$$

The zero-mean GP emulator, given by (10) can then be given a formulation in cluster space:

$$\hat{\nu}(\mathbf{x}) \; = < \mathcal{U}(\mathbf{x}), \mathcal{A}^{-1}\mathcal{N}(\top) > \tag{18}$$

where $\mathcal{N}(\top)$ is an as yet unspecified vector with r components. Since it is to be interpreted as the counterpart of $N(\top) = [\nu(\mathbf{x}_1), \ldots, \nu(\mathbf{x}_n)]^T$, it is clear that it should be determined by the true output values $\nu(\mathbf{x}_i)$. It is convenient to refer to this GP emulator as a cluster-based GP emulator, while the emulator described in (10) might be called a point-based emulator.

8 Interpretation of RBFN as a zero-mean cluster-based GP emulator

If we consider clusters C_i that have an associated center $\boldsymbol{\mu}_i$, we may take $G : \Delta \to \Delta$ as $G(C_i) = \{\boldsymbol{\mu}_i\}$ for $i = 1, \ldots, r$, and $G(\{\mathbf{x}\}) = \{\mathbf{x}\}$. For this G it holds that

$$\begin{aligned} \varsigma(\{\mathbf{x}\}, C_i) \quad &= \quad \Gamma(d_c(G(\{\mathbf{x}\}), G(C_i))) \\ &= \quad \Gamma(d_c(\{\mathbf{x}\}, \{\boldsymbol{\mu}_i\})) \end{aligned}$$

Whatever distance measure d_c is chosen, it is natural that $d_c(\{\mathbf{x}\}, \{\boldsymbol{\mu}_i\}) = d(\mathbf{x}, \boldsymbol{\mu}_i)$, as d_c is an extension of the concept of distance between points to distance between clusters, i.e. sets of points. This implies that with this

choice of G we have that $\varsigma(\{\mathbf{x}\}, C_i) = \Gamma(d(\mathbf{x}, \boldsymbol{\mu}_i)) = \phi(\mathbf{x}, \boldsymbol{\mu}_i)$. It follows from this result and from (17) that

$$\mathcal{U}(\mathbf{x}) \quad = \quad [\phi(\mathbf{x}, \boldsymbol{\mu}_1), \ldots, \phi(\mathbf{x}, \boldsymbol{\mu}_r)]^T \tag{19}$$

and thus that a RBFN, as given by (2), can also be written as

$$\hat{\nu}(\mathbf{x}) \quad = \quad < \mathcal{U}(\mathbf{x}), W > \tag{20}$$

If we let $\mathcal{N}(\top)$ in the zero-mean cluster-based GP emulator (18) be determined by the equation $\mathcal{A}^{-1}\mathcal{N}(\top) = W$, we see that a RBFN is a zero-mean cluster-based GP emulator.

Furthermore, with W determined as $W = \Phi^{\dagger} N(\top)$, see (4), it follows that $\mathcal{A}^{-1}\mathcal{N}(\top) = \Phi^{\dagger} N(\top)$. The left-hand side concerns operations in cluster space, while the right-hand side refers to operations in point space. One interesting loose interpretation of this equation is that the pseudo-inverse of Φ is equivalent to an inverse correlation matrix in cluster space.

It is also interesting to notice that a zero-mean GP emulator is, at the same time, a RBFN. Letting $r = n$ it follows that each cluster contains exactly one training data point, such that $\mathbf{x}_i = \boldsymbol{\mu}_i$. It is easily seen that in this case we have that $\mathcal{U}(\mathbf{x}) = U(\mathbf{x}) = [\varsigma(\mathbf{x}, \mathbf{x}_1), \ldots, \varsigma(\mathbf{x}, \mathbf{x}_n)]^T$ and that, as we already know from Section 3, $W = \Phi^{-1} N(\top)$. Furthermore, in this case $\Phi(i, j) = \phi(\mathbf{x}_i, \boldsymbol{\mu}_j) = \phi(\mathbf{x}_i, \mathbf{x}_j) = \varsigma(\mathbf{x}_i, \mathbf{x}_j) = A(i, j)$, where the last two equalities follow from the results described in Section 5. Thus $\Phi = A$, and the RBFN, given by (20), then becomes $\hat{\nu}(\mathbf{x}) =< U(\mathbf{x}), A^{-1} N(\top) >$. This is the zero-mean GP emulator, given by (10).

Thus a RBFN is a zero-mean GP emulator in cluster space, while a zero-mean GP emulator in point space is an RBFN. In other words, RBFNs and zero-mean GP emulators are equivalent, provided that the class of zero-mean GP emulators is not restricted to point-based zero-mean GP emulators, but also includes cluster-based zero-mean GP emulators.

9 Discussion of the use of the established relationship

One of the strengths of Gaussian process emulation is that it not only provides an approximation to an unknown function, given a training data set, but that it also allows to determine a confidence interval around this approximation (see Section 4). However, GP emulation has the important disadvantage that it relies on inverting the correlation matrix A, which makes GP emulation, as considered in point space, infeasible for large training data sets. In a sense, the reverse holds for RBFNs. This method does not provide a confidence interval, but it is able to handle a very large number of training data points since it relies on cluster analysis.

The advantages of both methods can be combined by taking into account the established relationship between RBFN and GP emulation. Given a large training data set, an approximation to an unknown function in a point $\mathbf{x}$ can be obtained with the use of RBFN, as given by (2). Furthermore, a confidence interval can be derived by taking advantage of the established fact that a RBFN is a zero-mean cluster-based GP emulator. This is done by considering the confidence interval (13), which holds for GP emulation in point space, in cluster space:

$$CI(\mathbf{x}) \quad = \quad [\hat{\nu}(\mathbf{x}) - 2\,\mathcal{V}(\mathbf{x})\,,\, \hat{\nu}(\mathbf{x}) + 2\,\mathcal{V}(\mathbf{x})] \tag{21}$$

where $\mathcal{V}(\mathbf{x})$ denotes the posterior variance in cluster space, obtained by using the concepts developed in Section 7 in order to translate the posterior variance in point space, given by (12), to cluster space:

$$\mathcal{V}(\mathbf{x}) \quad = \quad \frac{\mathcal{N}(\top)^T \mathcal{A}^{-1} \mathcal{N}(\top) \left[1 - \mathcal{U}^T(\mathbf{x}) \mathcal{A}^{-1} \mathcal{U}(\mathbf{x})\right]}{r-2} \tag{22}$$

Notice that in translating (12) to (22) the number of training points n has been replaced by the number of clusters r, since in cluster space the basic entities are clusters.

Since the dimension of $\mathcal{A}$ is only $r \times r$, compared to the $n \times n$ dimension of the correlation matrix A in GP emulation in point space, it is clear that very large training data sets should be considered in cluster space. In cluster space, we obtain an approximation by standard application of RBFN, while the established relationship between RBFN and GP emulation allows to construct a confidence interval $CI(\mathbf{x})$, given by (21), something which is lacking in the traditional description of RBFN.

10 Illustration of the use of the established relationship

The use of the established relationship, discussed in the previous section, is illustrated using an artificially created training data set. The considered true function is $\nu : \mathbb{R}^2 \to \mathbb{R}$ with $\nu(x, y) = x^2 + y^2$. The training input data consists of randomly generated points from 3 different multivariate normal distributions with means $(0, 0), (6, 0)$ and $(0, 6)$, and with each covariance matrix chosen as the identity matrix. From each distribution we generate 300 points, implying that the training data set contains 900 points. This is a relatively large training data set.

The purpose is to obtain an approximation to ν, which is presumably unknown, and a confidence interval around this approximation. Since the training data is relatively large, we consider the problem in cluster space. This means that we approximate ν with a RBFN (2) with ϕ chosen as in

(1) with $\gamma = 0.15$. Since our goal is merely to illustrate how the established relationship can be used in practical applications, we simply chose γ by trial and error. Furthermore, we choose $r = 50$, meaning that the training input data set is divided into 50 clusters, which are obtained by k-means. Next, we compute $\mathcal{A}$ and $\mathcal{U}$ as given in Section 7.2, and we obtain $\mathcal{N}(\top)$ from the relation $\mathcal{A}^{-1}\mathcal{N}(\top) = \Phi^{\dagger} N(\top)$ (see Section 8). From this we can compute the posterior variance in cluster space in any point $\mathbf{x}$, given by (22), and this then allows to compute the confidence interval $CI(\mathbf{x})$ given by (21).

For convenience, we display the results for a line segment in the input space $\mathbb{R}$. We choose the segment (x, y) with $x \in [0, 3]$ and $y = -x + 3$, because this line segment is well surrounded by training data input points. The result is shown in Fig. 1. Notice that the true function is completely within the confidence interval. It is also striking that the confidence interval is especially narrow around $x \in [2.5, 3]$, exactly where the approximation is very close to the true function.

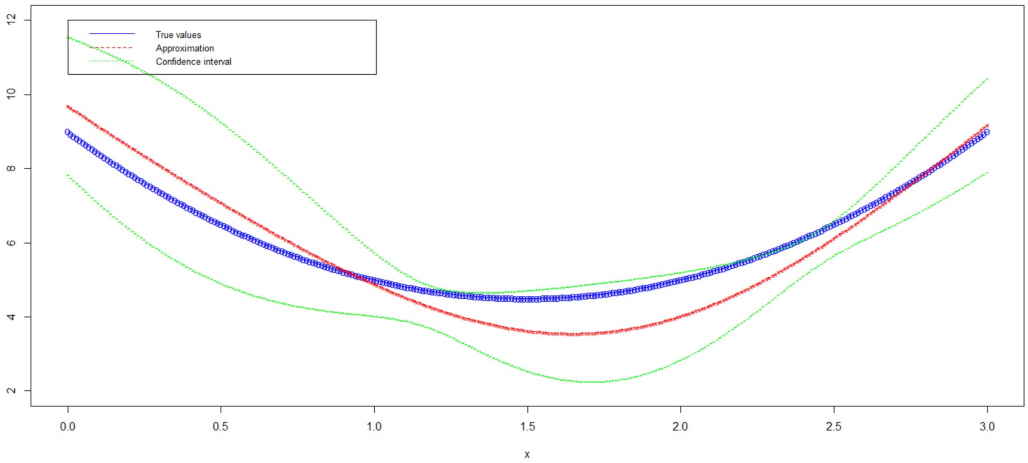

Figure 1: Approximation and confidence interval around approximation for $\nu(x, y) = x^2 + y^2$ with $x \in [0, 3]$ and $y = -x + 3$. This illustrates a zero-mean cluster-based GP emulator.

11 Conclusion

By generalizing correlation between points to correlation between clusters, we were able to establish that RBFNs and zero-mean GP emulators are equivalent. This relationship was derived by restricting to certain kinds of GP emulators and RBFNs. In essence, our entire attention centered on the originally developed GP emulators and RBFNs, although a lot of diverse extensions of both methods have been developed. Nevertheless, there are two important contributions of our work. First, it increases insight into the

principles of both RBFN and GP emulation. For example, a RBFN can be considered as an artificial neural network [7], which has an intuitive representation as a diagram with interconnected nodes, and this mental picture might make interpretation of certain concepts in GP emulation easier. On the other hand, the fact that the pseudo-inverse of the matrix Φ in RBFN can be viewed as an inverse correlation matrix in cluster space might result in a deeper understanding of RBFNs. Increased understanding of and a broader perspective on both methods bring us to a second main contribution. The fact that both methods are directly related results in cross-fertilization. For example, a GP emulator not only provides a best approximation of the output of a given simulator in a given input point, but it also produces a measure of uncertainty of this approximation. Such a measure of uncertainty is absent in RBFN, and the relationship we have established can be used to introduce a confidence interval for a RBFN, as we have show in our work. In the other direction, GP emulation can also benefit from the derived connection. One of the main problems with GP emulation lies in numerical instabilities in inverting the correlation matrix (for a discussion, see, e.g., [15] and [16]). This problem becomes worse as the training data set size increases, since the number of entries of the correlation matrix A is quadratic in the number of training data points. However, the cluster-based GP emulator, which is equivalent to a RBFN, has a correlation matrix $\mathcal{A}$ whose dimension is only quadratic in the number of clusters.

References

[1] K.-G. Steffens, The history of approximation theory: from Euler to Bernstein, Birkhauser, 2006.

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

[3] J. Clymer, Simulation-based engineering of complex systems, Wiley-Interscience, 2009.

[4] M. Powell, Radial basis functions for multivariable interpolation: a review, in: IMA Conference on Algorithms for the Approximation of Functions and Data, 1985, pp. 143–167.

[5] B. Fornberg, E. L. N. Flyer, Stable computations with gaussian radial basis functions, SIAM Journal on Scientific Computing 33 (2) (2011) 869–892.

[6] A. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666.

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

[8] J. Oakley, A. O'Hagan, Bayesian inference for the uncertainty distribution of computer model outputs, Biometrika 89 (4) (2002) 769–784.

[9] C. Rasmussen, C. Williams, Gaussian processes for machine learning, MIT Press, 2006.

[10] B. Grigelionis, Student's t-distribution and related stochastic processes, Springer, 2012.

[11] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, D. Rubin, Bayesian data analysis, Chapman and Hall/CRC, 2013.

[12] A. O'Hagan, Bayesian analysis of computer code outputs: A tutorial, Reliability Engineering & System Safety 91 (10-11) (2006) 1290–1300.

[13] A. Gardner, J. Kanno, C. Duncan, R. Selmic, Measuring distance between unordered sets of different sizes, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[14] M. Goldberg, M. Hayvanovych, Measuring similarity between sets of overlapping clusters, in: Proceedings of the 2010 IEEE Second International Conference on Social Computing, 2010.

[15] J. Ramon, K. Driessens, On the numeric stability of gaussian processes regression for relational reinforcement learning, in: ICML-2004 Workshop on Relational Reinforcement Learning pages, 2004, pp. 10–14.

[16] P. Ranjan, R. Haynes, R. Karsten, A computationally stable approach to gaussian process interpolation of deterministic computer simulation data, Technometrics 53 (4) (2011) 366–378.