Comparing Knowledge Representation Forms in Empirical Model Building

Hao Wang
Leiden Institute of Advanced Computer Science
Leiden University
Leiden, The Netherlands
e-mail: h.wang@liacs.leidenuniv.nl

Ingo Schwab
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
e-mail:
Ingo.Schwab@HS-Karlsruhe.de

Michael Emmerich
Leiden Institute of Advanced Computer Science
Leiden University
Leiden, The Netherlands
e-mail:
m.t.m.emmerich@liacs.leidenuniv.nl

Abstract—**Empirical models in engineering practice often come from measurements of the machines but might also be generated from expensive simulations to build so-called surrogate models. From an abstract point of view can be seen as approximations of functions that map input variables to output variables. This paper describes and conceptually compares different function approximation techniques, with a focus on methods from machine learning, including Kriging Models, Gaussian Processes, Artificial Neural Networks, Radial Basis Functions, Random Forests, Functional Regression, and Symbolic Regression. These methods are compared on basis of different criteria, such as speed, number and type of parameters, uncertainty assessment, interpretability, and smoothness properties. Besides, a particular focus is to compare the different ways of how knowledge is represented in these models. Here we compare the families of functions used to build the model and which model components (structures, parameters) are provided by the user or learned from the available data. Although this paper is not about benchmarking, some numerical examples are provided that illustrate the typical behavior of the methods.**

Keywords-Function Approximation; Machine learning; Process Modelling; Model Formation; Symbolic Regression

I. INTRODUCTION

Contemporary engineering design is heavily based on computer simulations or work piece experiments. The results are used not only for design verification but, even more importantly, to adjust parameters of the system to have it meet given performance requirements. Unfortunately, accurate simulations or real life experiments with machines or work pieces are often very expensive. High-fidelity simulations can take with evaluation times as long as hours or even days per design, making design automation using conventional methods impractical. These and other problems can be alleviated by the development and employment of so-called surrogates that reliably represent the expensive, simulation- or experiment based model of the system or device of interest. They are often analytically tractable and can give answers in a much faster or less costly way. Once the model is set up, the parameter values which produce the desired output from the given input can be retrieved fully automatically and without any delay. More importantly, the models can inter- or extrapolate values and thereby predict response values for new input parameters. In this way empirical model building is a regression task. Methods already exist, but it is still subject to intense research. In particular advances were made in the field of machine learning where by the use of computers significantly more complex models can be studied than it was possible before the 'digital age'. This paper gives an overview over machine learning methods and classical methods that can be used in empirical model building. The goal is to compare them on basis of a wide range of mainly conceptual (qualitative) properties. In particular, we study how knowledge is represented with these methods and which knowledge can be introduced by the user or has to be learned by the method. We deliberately avoided an evaluation of the different algorithms on benchmark problems, as we think that such a benchmark is beyond the scope of a workshop paper as it would involve proper tuning/setting of model parameters in order to provide a fair comparison. Instead, we provide numerical examples in order to demonstrate typical behavior of certain model types.

This paper is organized as follows. In Section II, the most commonly exploited approximation models are briefly introduced, illustrating their corresponding capabilities and limitations. In addition, some intrinsic properties e.g., model uncertainty assessment are discussed in detail. In Section III, a numerical comparison is made through modelling the data sampled from a 2-D Rastrigin function. In Section IV, we compare the models in terms of their design principles as well as their underlying mathematical structures and parameters. Finally, Section V concludes the paper and points out the further researches beyond this work.

II. BACKGROUND AND RELATED WORK

In this section we first briefly describe the most commonly used function approximation models. Then, some important properties of the models: symbolic representation, model uncertainty assessment and universal function approximation capability are discussed.

A. Function Approximation Models

1) Kriging

Kriging is a stochastic interpolation/regression approach, which originates from earth science [1] and originally targets at problems in geo-statistics and mining. Note, that the

Kriging method is also termed as Gaussian Process Regression [2] in the statistical machine learning literature, although the latter is restricted to normal distributions and typically provides a Bayesian interpretation of the predictor. Kriging assumes that the observed input-output data is the realization of a random field, and based on this assumption estimates correlation parameters and then computes the best linear unbiased estimator to predict the output value for a given set of input variables. It offers a local assessment of the prediction uncertainty, known as the Kriging variance at any unobserved data point.

The Kriging method interpolates the output at unknown data sample by modeling the response values as a realization of a random process y, which is a sum of a function $\mu(\cdot)$ and a centered Gaussian random field ε [2].

$$y(x) = \mu(x) + \varepsilon(x) \quad (1)$$

Despite the fact that a spatial index is used (from R^n space) it is common in the literature to call the random field also "Gaussian Process" for the multi-dimension case. Moreover, unlike Gaussian Process Regression, the Kriging framework can also be used for non-Gaussian random fields.

The centered Gaussian random field ε is completely defined by specifying the covariance function $\mathrm{k}(\cdot,\cdot)$, that solely depends on the relation between the input vectors.

$$k(x,x') = Cov[\varepsilon(x),\varepsilon(x')] = E[\varepsilon(x)\varepsilon(x')] \quad (2)$$

The user has to choose the structure of the covariance function, which must be positive definite. It normally depends on the similarity between the inputs, the parameters of the covariance function are learned from the available data, e.g., by maximum likelihood estimation.

When the function $\mu(\cdot)$ is assumed to be constant and unknown, the method is called Ordinary Kriging (OK). If $\mu(\cdot)$ is a functional regression model it is called Universal Kriging. In OK, the predictions are made based on the posterior distribution conditioning on the training set (X,y), which is shown in (1). (here: $\mathcal{N}(\mathrm{m},\mathrm{s}^2)$ denotes the normal distribution with mean m and variance s^2)

$$y^t | X, y, x^t \sim \mathcal{N}(m(x^t), s^2(x^t)) \quad (3)$$

Where x^{t} is target (unknown) input to make the prediction. The posterior mean function is the estimator and the variance can be used to compute uncertainty margins. For details, refer to [2].

2) Support Vector Machines

Support Vector Machines (SVMs) [3] [4] are supervised learning algorithms which are originally designed for classification tasks. The original SVM algorithm was introduced in Computational Learning Theory conference (COLT-92) by Boser, Guyon, Vapnik [4]. It is now well-established in machine learning. SVMs target at optimally solving non-linear classification tasks, using two techniques to achieve the goal: maximum-margin separation and kernel functions.

The simplest form, linear SVM construct a hyper plane or set of hyper planes in a high-dimensional space. Linear SVM can be considered as a perceptron but improve it by performing an optimal classification. Informally, a binary classification is achieved by finding two hyper planes. Both of them separate the data while the distance between them is maximized. The region bounded by these two hyper planes is called the margin. The motivation behind this approach is, in general, the larger the margin the lower the generalization error of the classifier. The corresponding linear classifier is known as the maximum margin classifier. It is also equivalent to the perceptron of optimal stability.

In order to extend the capability of linear SVMs, the well-known kernel trick [5] is applied to implicitly map the samples to be separated to the (normally) high-dimension feature space. This allows for a linear maximum-margin separation in the feature space even if the problem is not linearly separable in the input space.

3) Neural Networks

Artificial neural networks (ANNs) [6] [7] [8] is a broad family of statistical learning algorithms inspired by biological neural networks, whose structure is presented as systems of interconnected artificial neurons. The first artificial neuron, a simple linear classifier, was proposed in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits [8]. Each artificial neuron is modeled as a transformation of a linear combination of its input. Given the input to a neuron is x (not necessarily the input to the whole network), the output O from an arbitrary neuron is:

$$O = \varphi\left(\sum_{i=1}^{n} w_i \cdot x_i\right) \quad (4)$$

Note that $\mathrm{w_i}$ is the weight for the linear combination while $\varphi(\cdot)$ presents the transformation, or called activation function more commonly. There are various activation functions that have been applied. For example, linear, Heaviside step function, sigmoid and hyperbolic tangent.

There are many types of ANNs. If there are no cycles (including self-connections) in the network, then we could divide the network into layers, which is termed as feed forward neural network [9]. This type of ANNs are built by choosing the number of layers and the class of activation function for each neuron. Many well-known neural networks belong to this category, for example, multi-layer perceptron and radial basis function networks. If cycles or self-connections exist in the network, it is generally called recurrent neural network [10].

The feed forward neural network is of great interest here because any real-valued continuous function can be approximated arbitrarily close by a multi-layer perceptron with just one hidden layer according to the universal approximation theorem [11] [12] (see subchapter D of this article). Therefore, the feed forward network design is expected as good approximation method. However, due to the curse of dimensionality [13], it will be unavoidable that, even for smooth functions, the amount of data needed for precise approximation tends to grow exponentially with the dimensionality, unless more precise assumptions on the model class are made.

A variety of training techniques have been employed and tested. Among them, the back-propagation technique [14] is the most popular and influential. The general idea is to express the training error (loss) function as the squared difference between one true target and the approximated value, which is differentiable with respect to the weights. Consequently, we could compute the partial derivative of the error to each weight so that some efficient optimization techniques based on gradients can be applied.

4) Radial Basis Function Networks

Radial basis functions (RBFs) [15] are a special class of real-valued functions whose value only depends on the distance to the central point. In another word, any function h having the property $\mathrm{h}(x) = \mathrm{h}(||x||)$ is a radial basis function. A classical and commonly used type is the Gaussian:

$$\mathrm{h}(x) = \exp\left(-\frac{||x - c||}{\mathrm{r}^2}\right), \tag{5}$$

where c is the center and r^2 is its radius. Many other types are also available, e.g., multi-quadric. Please refer to [15] for more details. Radial basis functions are typically used for function approximations, in which multiple RBFs are linearly weighted as the predictor:

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{m} w_i \, h(||\boldsymbol{x} - \boldsymbol{c}_i||) \tag{6}$$

Note that m is the number of the RBFs, each associated with a different central point c_{i} and a combination weight $\mathrm{w_i}$. From the neural network perspective, the RBFs approximation method can be viewed as an artificial neural network, using radial basis function as activation function and thus is also called radial basis function network (RBFN) [16]. The RBFN is normally trained in two steps. Firstly, the central points of the RBF functions are chosen in an unsupervised manner. Usually, random sampling or k-mean clustering is used for this step. Secondly, note that the predictor $\hat{y}(\cdot)$ is differentiable with respect to the weights. Therefore, any well-established learning method involving derivatives can be employed to train the RBFN [17], e.g., using the back-propagation method [14] to find optimal weights.

RBFN is also closely related to the Kriging method we have described earlier. Giannakoglou [18] introduced an approach on how to employ RBF networks for exact interpolation in the sense that results for points in the training set are reproduced exactly. This kind of RBFN leads to the same equations as they are used in the prediction step of Simple Kriging [19].

5) Polynomials and Splines

As an extension to the linear model, polynomial regression models the relation between the response variable and the input variables as a polynomial, which represents a non-linear underlying assumption/knowledge. Therefore, it has been widely applied on the data involving nonlinear phenomena. The polynomial maps the input in R^n to high dimensional space R^m ($m > n$) by introducing interactions/correlations between input components. The free parameters (to estimate) in the polynomial regression is the coefficient of the polynomial plus the degree of the polynomial. Although polynomial regression models a non-linear relation, it is still linear to the unknown coefficients and the transformed input in R^m.

Second order polynomial functions are often used in real live scenarios. However, the use of polynomials as a global approximation only makes sense if the initial landscape, is unimodal, which is often the case in engineering problems.

The coefficients are usually determined by the (ordinary) least squares estimation method, which minimizes the variance of the estimators of the coefficients. The degree of the polynomial is usually specified by the user.

As an extension to simple polynomial functions, Splines [21], which are piecewise polynomials defined on disjoint domain partitions, are usually preferred for approximation purposes due to their ability of avoiding the instability from Runge's phenomenon [22] in high degree polynomial fitting. As for univariate regression of order n (the highest order of all the polynomials), each piecewise polynomial defined on disjoint interval is chosen such that the derivatives of each pair of connecting polynomial at the connection point should be equivalent up to order $n - 1$, which leads to the continuous and continuously differentiable of the whole spline curve.

The mostly applied splines are B-splines ('B' stands for basis), particularly cubic B-splines. The importance of B-splines is mainly due to the fact that any spline curve can be represented as a weighted linear combination of B-Splines. For each piecewise polynomial, the B-splines up to a certain order can be evaluated recursively by De Boor's algorithm [23]. Using B-splines, a function approximation task can be achieved by fitting a spline function composed of a weighted sum of B-splines, using the least-squares method.

6) Random Forests

Ensemble learning algorithms [26] [27] construct a set of classifiers and then label new data sets by taking a (weighted) vote of their predictions, e.g., random forest [28]. The main principle behind ensemble methods is that the prediction quality increases when a set of learning algorithms are grouped together. There are different types of algorithms for ensemble learning, i.e., boosting [29], bootstrap aggregation [30] or stacking [31].

Random forests are also a type of ensemble learning. They combine different decision tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The calculation of the resulting class (classification problem) or regression output is the most straightforward. For classification the majority of the class output of the decision trees are calculated. For a regression problem accordingly the mean of the trees are the result of the model.

The advantages of random forest is mostly its fast running speed, both in training and prediction/classification and easy interpretability. In addition, they are able to deal with unbalanced and missing data, which is quite common in the real application. Its weaknesses are that when used for regression they cannot predict beyond the range in the training data, and they may over-fit data sets that are particularly noisy.

7) *Symbolic Regression*

In the previous subsection, we assumed that we already know the structure of the output. After that we choose the best simplified model and fit the free parameters of the model. From this point of view Symbolic Regression is much more powerful. In this case, the function is composed of an arbitrary (but predefined) set of mathematical symbols, forming a valid expression of a parameterized function. Like other statistical and machine learning regression techniques symbolic regression also tries to fit observed experimental data. But unlike the well-known regression techniques in statistics and machine learning, symbolic regression is used to identify an analytical mathematical description and it has more degrees of freedom in building it.

A set of (basic) operators is predetermined (e.g., add, multiply, sin, cos) and the algorithm is mostly free in concatenating them. In contrast to the classical regression approaches which optimize the parameters of a prescribed structure, here the structure of the function is free and the algorithm optimizes the parameters and the structure.

There are different ways to represent the solutions in symbolic regression. For example, informal and formal grammars have been used in genetic programming to enhance the representation and the efficiency of a number of applications including symbolic regression.

Since symbolic regression operates on discrete representations of mathematical formulas, non-standard optimization methods are needed to fit the data. The main idea of the algorithm is to focus on searching promising areas of the target space while abandoning unpromising solutions (see [24] [25] for more details). In order to achieve this, symbolic regression uses the main mechanisms of Genetic and Evolutionary Algorithms. In particular, these are mutation, crossover and selection which are applied to an algebraic mathematical representation.

The representation is encoded in a tree. Both the parameters and the form of the equation are subject to the target space of all possible trees that representing mathematical expressions. The operations are nodes in the tree and can be mathematical operations such as additions (add), multiplications (mul), abs, exp, etc. The terminal values of the tree consist of the function's input variables and real numbers (constants). The input variables are realized by the values of the training dataset.

In symbolic regression, many initially random symbolic equations compete to model experimental data in the most promising way. Promising are those solutions possessing a good compromise between better prediction quality of the observed data and the length of the mathematical formula.

Mutation in a symbolic expression can change the mathematical type of formula in different ways. For example, a div is changed to an add, the arguments of an operation are replaced (e.g., change 2*x to 3*x), an operation is deleted (e.g., change 2*x+1 to 2*x), or an operation is added (e.g., change 2*x to 2*x+1).

The objective in symbolic regression, like in other machine learning and data mining algorithms, is to minimize the regression error on the training data. After an equation reaches a desired level of accuracy, the algorithm returns the best equation or a set of good solutions (the Pareto front). In many cases the solution reflects the underlying principles of the observed system.

B. *Symbolic vs Subsymbolic Representation*

In the perspective of symbolic representations, the previously described methods can be categorized according to whether they are using symbolic or subsymbolic model representations.

As Smolensky [32] noted, the term subsymbolic paradigm is intended to suggest symbolic representations that are built out of many smaller constituents: “Entities that are typically represented in the symbolic paradigm by symbols are typically represented in the subsymbolic paradigm by a large number of subsymbols” (p. 3).

The debate over symbolic versus subsymbolic representations of human cognition is this: Does the human cognitive system use symbols as a representation of knowledge? Or does it process knowledge in a distributed representation in a complex and meaningful way? E.g., in neural networks the knowledge is represented in the parameters of the model. It is not possible to determine the exact position of the knowledge.

From this point of view, the syntactic role of subsymbols can be described as the subsymbols participate in numerical computation. In contrast, a single discrete operation in the symbolic paradigm is often achieved in the subsymbolic paradigm by a large number of much finer-grained operations. One well known problem with subsymbolic networks which have undergone training is that they are extremely difficult to interpret and analyze. In [33], it is argued that it is the inexplicable nature of mature networks. Partially, it is due to the fact that subsymbolic knowledge representations cannot be interpreted by humans and that they are black box knowledge representations.

C. *Estimation Uncertainty*

Commonly, the mean square error (MSE) of the predictor, which measures the average of the squared error over the validation data set in the cross validation, is the most accessible error information from the models. It is used as an objective function (loss function) to facilitate the model fitting. For the regression model assuming homoscedasticity (constant noise variances), MSE is also an estimation of the uncertainty measure of the predicted values. When the noise variances are assumed to be non-constant, MSE gives no clue to the uncertainty measure of predicted values.

As a advantage of Kriging, it provides the MSE of the estimator or so-called Kriging variance as a built-in feature. It is of significant importance in machine learning as well as global optimization. It directly shows the regions where Kriging model might perform badly (high variance). The Kriging variance is determined by the relative location between the training data, the location of the input to predict as well as the covariance structure.

D. *Universal Function Approximators*

As mentioned earlier, multi-layer neural networks can be considered as universal function approximators. The formal statement of universal approximation theorem [12] states that neural nets with single hidden layer can approximate any function which is continuous on n-dimensional unit hypercube. In [11] Cybenko has showed that a continuous function on a compact set can be approximated by a piecewise constant function. And a piecewise constant function can be represented as a neural net as follows. For each region where the function is constant, use a neural net as an indicator function for that region. Then build a final layer with a single node, whose input linear combination is the sum of all the indicators, with a weight equal to the constant value of the corresponding region in the original piecewise constant function. With this idea every continuous function can be represented with a neural network.

While Cybenko's result is an approximation guarantee Kolmogorov [34] proved that that a neural network provides an equality. Additionally, with heterogeneous transfer functions it can be proved that only $O(n^2)$ nodes are needed. It should be mentioned that Cybenko's result, with using only one type of activation function, is more relevant to machine learning.

III. ILLUSTRATIONS

Despite all the discussions in this paper, the behavior of the models is still quite vague at this moment. Thus, a small illustration of the model behavior would be necessary. We try to fulfill this task by showing the capability of modelling methods on the well-known 2-D Rastrigin function, whose highly rugged response surface is depicted in Fig. 1. By drawing 1000 points using Latin hypercube sampling in $[-5,5]^2$, we build a Kriging, a polynomial regression and a RBFN model. In addition, SVMs with linear and polynomial kernels are also constructed. The polynomial regression and SVM with polynomial kernel look similar and generally capture the global quadratic structure of the function but smooth out the surface. The SVM with linear kernel is expected to be a 2-D plane. The RBFs performs even better than SVMs due to the fact that it also shows a "bumpy" surface compared to the real surface. The Kriging model both reproduces the global trend of the function and includes small fluctuations although they are too small in scales.

IV. MODEL PROPERTY COMPARISONS

In order to obtain a more accessible view of the similarities and differences among all the function approximation methods described in this paper, we summarize the major feature, characteristic and properties of these models in two perspectives.

On one hand, we summarize the properties of models which are implied by their corresponding design principles in Table 1. Those intrinsic properties includes whether the model is symbolic or subsymbolic, the uncertainty assessment and time complexity, etc.

The models presented can be classified by whether they are designed based on symbolic or subsymbolic representations. As discussed in section II.B, such property determines whether the model knowledge can be understood by human.

Uncertainty measurement is another important aspect, providing additional knowledge on the quality/confidence of the model. In this case, Kriging model is distinguishing because the exact mean square error is available for the predicted values, as depicted in section II.C.

These models also differ in the interpretability. Some models have clear and meaningful explanations, e.g., the linear relation between the output and the input in linear regression. However, models like multi-layer perceptron has no direct implications.

The training methods vary on these models due to corresponding underlying assumptions and model complexity. Commonly used methods include least square estimation, maximization likelihood, back-propagation, cross-validation and mathematical programming. In some models (e.g., SVM), an additional training method is needed for the additional hyper-parameters. For example in SVM, quadratic programming is used to find the model weights while the cross-validation could also be applied to fit the parameters in the kernel function.

Despite the theoretical elegance of some modeling algorithms, the time complexity is crucial in the real application, where some of them might be not computationally feasible on large dataset. Kriging takes very high $O(n^3)$ effort for the model training compared to $O(n)$ of linear regression.

On the other hand, in terms of the structure and parameters contained in the model, we are also interested in what kind underlying mathematical structures the models assume/built upon, what mathematical structures/parameter could be learned from the training data or should be fixed by the user. As a simple example, linear models assume a linear structure, where no parameters needed to provide by the users and the coefficients are learned from the data. In contrast, for symbolic regression and random forest, the tree structure is used. Such comparisons are listed in Table 2.

V. CONCLUSION

In this paper, we describe the most commonly used function approximation methods. The basic properties of them are discussed briefly. The meaning and effect of symbolic/subsymbolic representation is depicted. We also compare the ability of obtaining uncertainty measure for the models. We construct several models on the 2-D Rastrigin function and demonstrate the performance of approximation of these models. Finally, two tables are made to summarize and compare the essential conceptual properties of the models, where most of the important aspects are covered.

One finding is that already before fitting the model many decisions are made by the user. These decisions might restrict the capability of the models and ultimately this will unintendedly influence the prediction results. Universality properties do not practically solve these issues, as they are stated for a model whose size tends to infinity. A more meaningful approach to find models that can be used with confidence might be to use self-assessment of the error/model consistency provided by the method itself (Kriging method) or to build models that can be interpreted by humans (symbolic regression, random forests). The symbolic regression framework is particularly interesting because it also frees the user from the burden of deciding on a symbolic model representation a-priori, in cases where no 'natural' functional expression can be assumed.

In the future work, it would be interesting to investigate how to combine a quantitative performance assessment with the qualitative assessment of methods, on which our work focused. Also, symbolic regression can naturally be combined with some of the other machine learning techniques, for instance by learning the structure of kernel function.

ACKNOWLEDGMENT

This work was supported by the EU Project I-RAMP3 (Intelligent Network Devices for fast Ramp-up), project homepage http://www.i-ramp3.eu/. I-RAMP3 is co-financed by the European Commission DG Research under the 7th Framework Program.

This work was also supported by Netherlands Organization for Scientific Research (NWO) project PROMIMOOC.

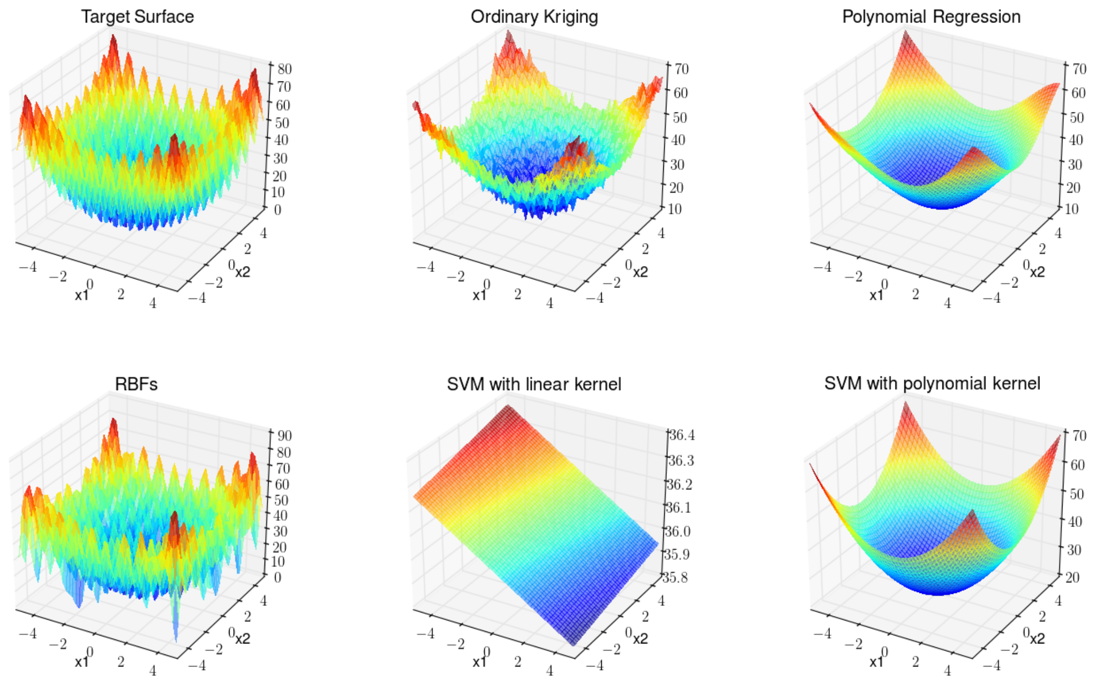

Figure 1 . On 2-D, the function approximations from 6 methods on Rastrigin function.

TABLE I. COMPARISON OF FUNCTION APPROXIMATION AND CLASSIFICATION MODELS BASED ON DIFFERENT PROPERTIES.

Methods	symbolic/ subsymbolic	uncertainty assessment	consistency check	interpretability	computational effort	training criterion/ method	exact interpolation	smoothness	Main purpose
Regression Models									
Linear Regression	subsymbolic	Mean Squared Error, global	Mean Squared Error	good	very small (O(n))	least squares	no	yes, differentiable	function approximation
Polynomial Regression	subsymbolic	Mean Squared Error, global	Mean Squared Error, parsimony	good	scalable (O(n^k)), k is degree of polynomial	least squares	no	yes, differentiable	function approximation
Symbolic Regression	symbolic	Mean Squared Error, global	parsimony (Occam's razor)	excellent	scalable/high, typically heuristics are used	least squares	no	yes, differentiable	function approximation,
Random Process Models									
Ordinary Kriging	subsymbolic	Mean Squared Error, local; conditional variance	Likelihood value	medium	high in number of training points (O(m^3)), fast prediction (O(m))	maximum likelihood, best linear unbiased prediction of a random process	yes	yes, continuous and for some correlation functiions differentiable	function approximation
Universal Kriging	subsymbolic	Mean Squared Error, local; conditional variance	Likelihood value	medium	high in number of training points (O(m^3)), fast prediction (O(m))	maximum likelihood, best linear unbiased prediction of a random process	yes	yes, continuous and for some correlation functiions differentiable	function approximation
Support Vector Machines									
SVN with linear kernel	subsymbolic, variable structure	Distances of training vectors to the separation plane.	penalty function value	medium	medium (quadratic programming)	minimization of distances to separating hyperplane	no	yes	Classification(binary)
SVN with RBF kernel	subsymbolic	Mean squared error	penalty function value	medium	medium (quadratic programming)	minimization of distances to separating hyperplane, after transformation	no	yes	Classification(binary)
SVN with polynomial Kerne	subsymbolic	Mean squared error	penalty function value	medium	medium (quadratic programming)	minimization of distances to separating hyperplane, after transformation	no	yes	Classification(binary)
Decision tree models									
Decision trees	structural	Global training errors	training error rates	good	fast	greedy split algorithm	no	discontinuous	Classification
Random Forests	structural	Global training errors	training error rates	good	fast-medium	randomized search	no	discontinuous	Classification
Rule Ensembles	structural	Distance from decision threshold (local), training error (global)	not applicable	good	depending on number and effort to compute rules, typically heuristics are used	error minimization (diverse methods, Michigan/Pittsburgh style)	no	discontinuous	Classification
Piecewise defined functions									
Splines	subsymbolic	no	not applicable	medium	medium, typically quadratic time complexity, prediction is very fast	smoothness maximization	no	optimal	function approximation
K Nearest Neighbors	subsymbolic	no	not applicable	good	very fast training and prediction	averaging, transduction	only for K=1	discontinuous	function approximation, binary class separation
Connectionist Models									
Radial Basis Functions	subsymbolic, variable structure+	Mean Squared Error	not applicable	intransparent	prediction in linear time, training time depends on number of RBF centers O(n^3)	least squares	only, if no. of basis functions equals or exceeds no. of training points	continuous, typically also differentiable	
Multilayer Perceptron	subsymbolic, variable structure+	Mean Squared Error	not applicable	intransparent	fast, training time depends on number of neurons	least squares, backpropagation	no	continuous, differentiable	

TABLE II. COMPARISON OF MODELS. ROLES OF USER, LEARNING COMPONENT AND FRAMEWORK

Methods/Model Components	Structures fixed by framework	Structures fixed by user	Structures learned	Parameters fitted from data	Parameters set by user
Regression Models					
Linear Regression	linear function	no	no	coefficients	no
Polynomial Regression	polynomial function	no	no	coefficients, degree of polynomial	degree of polynomial
Symbolic Regression	Use of expression trees	Detailed grammar of expression tree, terminal symbols, expressions used	Symbol set, Terminal Symbols, Structure of Function tree	Numerical constants in formula	no
Random Process Models					
Ordinary Kriging	Linear predictor	Correlation function type		Mean value, global variance, correlation hyperparameters	no
Universal Kriging	Linear predictor , regression function (free choice)	Type of regression function and type of the correlation function	no	parameters of regression function	no
Support Vector Machines					
SVM with linear kernel	linear hyperplane as separator	type of SVM (e.g. squared deviations, Tchebycheff)	no	support vectors	no
SVM with RBF kernel	RBF kernel	type of RBF Kernel, type of SVM (e.g. squared deviations, Tchebycheff)	no	kernel parameters	no
SVM with polynomial Kernel	polynomial function	type of SVM (e.g. squared deviations, Tchebycheff)	no	kernel parameters	no
Decision tree models					
Decision trees	tree structure, separation by hyperplanes	no	tree structure	thresholds, parameters of tree	no
Random Forests	set of decision trees, separation by hyperplanes	no	tree structures, number of trees		no
Rule Ensembles	way to combine rules (e.g. weighting)	rule set, rules are generated automatically in some cases	subset selection of rules, in some cases structure of rules	weights of rules	no
Piecewise defined functions					
Splines	Basis functions, way to superpose basis functions	Type of splines, type of basis functions	no	parameters of basis functions	no
K Nearest Neighbors	no	distance function, averaging function	no	no	Number of neighbors K
Connectionist Models					
Radial Basis Functions	Type of functions (positive definite basis functions based on distance)	kernel function type, number of kernel functions, method to select the RBF centers	no	weights of radial basis functions, smoothness parameters	number of RBF centers
Multilayer Perceptron	Superposition of activation functions	activation function type, topology of network (number of neurons, number of layers)	no, although some advanced neural networks have the capability to learn topology	weights of activation functions and smoothness parameters in activation functions	no

REFERENCES

[1] D. Krige, “A statistical approach to some basic mine valuation problems on the Witwatersrand,” Journal of Chemical, Metallurgical, and Mining Society of South Africa, vol. 52, no. 6, pp. 119-139, December 1951.

[2] C. E. Rasmussen, and C. K. I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning): The MIT Press, 2005.

[3] C. Cortes, and V. Vapnik, “Support-vector networks,” Machine learning, vol. 20, no. 3, pp. 273-297, 1995.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in Proceedings of the fifth annual workshop on Computational learning theory, Pittsburgh, Pennsylvania, USA, 1992, pp. 144-152.

[5] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” The annals of statistics, pp. 1171-1220, 2008.

[6] C. M. Bishop, Neural networks for pattern recognition: Oxford university press, 1995.

[7] D. O. Hebb, The organization of behavior: A neuropsychological theory: Psychology Press, 2005.

[8] W. S. McCulloch, and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115-133, 1943.

[9] S. Haykin, Neural Networks: A Comprehensive Foundation: Prentice Hall PTR, 1998.

[10] S. Hochreiter, and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[11] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Mathematics of control, signals and systems, vol. 2, no. 4, pp. 303-314, 1989.

[12] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” Neural networks, vol. 4, no. 2, pp. 251-257, 1991.

[13] E. Novak, Deterministic and stochastic error bounds in numerical analysis: Springer-Verlag Berlin, 1988.

[14] A. E. Bryson, W. F. Denham, and S. E. Dreyfus, “Optimal programming problems with inequality constraints,” AIAA journal, vol. 1, no. 11, pp. 2544-2550, 1963.

[15] M. D. Buhmann, Radial basis functions: theory and implementations: Cambridge university press, 2003.

[16] D. S. Broomhead, and D. Lowe, “Multivariable Functional Interpolation and Adaptive Networks,” Complex Systems vol. 2, pp. 321-355, 1988.

[17] F. Schwenker, H. A. Kestler, and G. Palm, “Three learning phases for radial-basis-function networks,” Neural networks, vol. 14, no. 4, pp. 439-458, 2001.

[18] K. Giannakoglou, “Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence,” Progress in Aerospace Sciences, vol. 38, no. 1, pp. 43-76, 2002.

[19] M. Emmerich, “Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels,” IEEE Transactions on Evolutionary Computation, vol. 10, no. 4, pp. 421-439, August 2006.

[20] L. Magee, “Nonlocal behavior in polynomial regressions,” The American Statistician, vol. 52, no. 1, pp. 20-22, 1998.

[21] K. L. Judd, Numerical methods in economics: The MIT Press, 1998.

[22] C. Runge, “Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten,” Zeitschrift für Mathematik und Physik, vol. 46, no. 224-243, pp. 20, 1901.

[23] C. De Boor, “On calculating with B-splines,” Journal of Approximation Theory, vol. 6, no. 1, pp. 50-62, 1972.

[24] K. J. Holyoak, J. H. Holland, and J. H. Holland, Induction: Processes of inference, learning, and discovery: The MIT Press, 1989.

[25] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection: The MIT Press, 1992.

[26] Z.-H. Zhou, Ensemble methods: foundations and algorithms: CRC Press, 2012.

[27] R. Maclin, and D. Opitz, “Popular ensemble methods: An empirical study,” Journal of Artificial Intelligence Research, 1999.

[28] L. Breiman, “Random forests,” Machine learning, vol. 45, no. 1, pp. 5-32, 2001.

[29] L. Breiman, “Arcing classifier (with discussion and a rejoinder by the author),” The annals of statistics, vol. 26, no. 3, pp. 801-849, 1998.

[30] L. Breiman, “Bagging predictors,” Machine learning, vol. 24, no. 2, pp. 123-140, 1996.

[31] L. Breiman, “Stacked regressions,” Machine learning, vol. 24, no. 1, pp. 49-64, 1996.

[32] P. Smolensky, “On the proper treatment of connectionism,” Behavioral and brain sciences, vol. 11, no. 01, pp. 1-23, 1988.

[33] D. A. Robinson, “Implications of neural networks for how we think about brain function,” Behavioral and brain sciences, vol. 15, no. 04, pp. 644-655, 1992.

[34] V. M. Tikhomirov, "On the Representation of Continuous Functions of Several Variables as Superpositions of Continuous Functions of one Variable and Addition," Selected Works of A. N. Kolmogorov, Mathematics and Its Applications (Soviet Series) V. M. Tikhomirov, ed., pp. 383-387: Springer Netherlands, 1991.