Safe and Interpretable Machine Learning: A Methodological Review

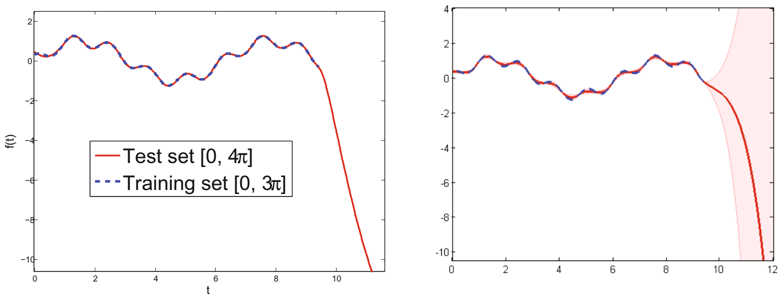
Clemens Otte

Abstract. When learning models from data, the interpretability of the resulting model is often mandatory. For example, safety-related applications for automation and control require that the correctness of the model must be ensured not only for the available data but for all possible input combinations. Thus, understanding what the model has learned and in particular how it will extrapolate to unseen data is a crucial concern. The paper discusses suitable learning methods for classification and regression. For classification problems, we review an approach based on an ensemble of nonlinear low-dimensional submodels, where each submodel is simple enough to be completely verified by domain experts. For regression problems, we review related approaches that try to achieve interpretability by using low-dimensional submodels (for instance, MARS and tree-growing methods). We compare them with symbolic regression, which is a different approach based on genetic algorithms. Finally, a novel approach is proposed for combining a symbolic regression model, which is shown to be easily interpretable, with a Gaussian Process. The combined model has an improved accuracy and provides error bounds in the sense that the deviation from the verified symbolic model is always kept below a defined limit.

1 Introduction

There is an increasing trend for using data-driven models (i.e. models learned from data) in monitoring and control applications, for instance in industry, healthcare or automotive electronics [9, 18]. The main reason usually is that analytical models derived from first principles are either unknown or suffer from insufficient accuracy. However, deploying data-driven models in applications where incorrect model outputs may have fatal consequences requires ensuring that the model is correct for all possible inputs. In practice, training data are almost always limited and may not

Clemens Otte
Siemens AG, Corporate Research and Technologies, 81739 Munich, Germany
e-mail: clemens.otte@siemens.com



(a) Single neural network (b) Ensemble of 50 networks

Fig. 1 Example of inappropriate extrapolation behavior. Left side: Neural network is tested outside the interval used for training. Right side: Uncertainty of the model is estimated by averaging the output of 50 neural networks. The uncertainty is large beyond the training data (shading shows mean ± 1 standard deviation)

represent all relevant operating conditions. Thus, it is crucial to understand what the model has learned and in particular how it will extrapolate to unseen data.

A simple example is shown in Fig. 1(a). A feed-forward network [2] with a single input, one hidden layer of nine neurons with sigmoidal activation function and a single output with linear activation has been trained on 1000 noise-free training samples drawn from a sum of two sines on the interval $[0, 3\pi]$. After training, it is tested on the larger interval $[0, 4\pi]$. While the test accuracy on the interval seen during training is very good, the extrapolation behavior is rather inappropriate. It is obvious that a linear combination of sigmoids cannot reproduce the periodicity of the true function. The key point here is that the model is tested in a range where the desired model behavior is not specified by training data. In other words, the model has learned an approximation to the unknown true function on the training data and beyond that data the approximation is poor.

A possible way for estimating the model uncertainty is to consider the average and standard deviation of the output of several models as shown in Fig. 1(b). In this example each of 50 neural networks was initialized with different random weights. They converged to slightly different solutions which are similar on the training data but have diverse extrapolation behavior. The resulting standard deviation is large in the region not specified by the training data, which might be used to alarm the user about the uncertainty. A further discussion of model averaging can be found in [5, 7].

While model averaging is a common approach for improving the prediction accuracy, there is no guarantee that the uncertainty estimation is correct. This is due to the fact that the ensemble members usually are not truly independent because, for example, they are trained on the same data or are based on the same type of model.

Other approaches for avoiding inappropriate extrapolation behavior include the following ones.

- Use of an additional model estimating the density in the input space and deactivating the prediction model in low-density regions. However, density estimation in higher-dimensional spaces is a challenge itself [7, 17].
- Constraining the model using a-priori knowledge. For example, if some input-output relations are known to be monotonic then monotonicity constraints can be introduced on the weights of a neural network [8].
- Limiting the allowed output range of the model, e.g. to the range given by the training data. This is of course just an ad hoc method.

None of these approaches can in general guarantee that unspecified behavior of the model does not occur. It is therefore preferable to use models possessing a level of interpretability that suits the safety requirements of the respective application. Usually, one has to weight interpretability against accuracy, so there is no magic formula. Instead, among the variety of methods the best one has to be chosen specific to the application. Some examples are given in this paper.

Structure of this Paper

The next section describes an approach for classification problems with an application to airbag control in automotive safety. Some basic ideas are then transferred to the learning of regression functions in Sect. 3. Using a benchmark data set (SARCOS inverse dynamics problem) several regression approaches are discussed. It is shown that a combination of symbolic regression with a Gaussian Process (GP) provides a good trade-off between interpretability and accuracy. In that combination, symbolic regression provides an analytical model and the GP improves the overall accuracy by learning the residuals of the analytical model. Section 4 concludes.

2 Safe and Interpretable Classification

In the following we review an approach based on an ensemble of nonlinear low-dimensional submodels where each submodel is simple enough to be completely verified by domain experts. The approach was firstly developed for binary classification problems [12] and then extended to multiple classes [13].

2.1 Ensembles of Low-Dimensional Submodels

The algorithm for learning an ensemble of low-dimensional submodels for binary classification problems is described in plain words below. For a formal description and extension to more than two classes we refer to [12, 13].

1. Given: Data set D with feature vectors in $\mathbb{R}^p$ of two classes (positive and negative class). Start with an empty set of selected models.
2. Consider all two-dimensional subspaces and learn a separation of both classes in each subspace, e.g. by a support-vector classifier, neural network or any other method. There is one important constraint for learning: The model must correctly

classify all negative samples in D, that is, false positives (i.e. false alarms) must not occur.
3. Select the best 2D-model (possibly involving expert knowledge in the selection process) and add the model to the set of selected models.
4. Remove all correctly classified positive samples from D.
5. Start again in step 2 to separate remaining positive samples until no positive instances remain or the maximum size of the set of selected models has been reached.

The reason for the no-false-alarm requirement in the second step is that it allows combining all selected models simply by logical-or. In other words a test sample will be assigned to the positive class if at least one of the models assigns it to that class. This greatly improves the interpretability of the ensemble as shown in the following example.

2.2 Example: Airbag Control

In this safety-relevant application the objective is to learn the deployment decision of an airbag system from crash test data. In each crash test the data from several sensors in the car (mostly acceleration sensors) were recorded and various features were extracted, resulting in a multivariate time-series sampled every 1 ms. In our experiment the data set includes 52 features. Each crash test either has a "nofire" or "fire" label. In case of a "fire" label, the crash has a desired point of time when the airbag has to be triggered, e.g. 24 ms after the first contact with the barrier. A "nofire" crash must not trigger the airbag at any time.

Since the time series of each crash comprises many samples that all share the same label, this is a typical example of a class of problems known as multiple-instance-learning [1]. For these problems it is sufficient to classify at least one sample of a "fire" crash correctly (within a certain time interval around the desired fire time). In contrast, no sample of a "nofire" crash may ever be misclassified as "fire".

Figure 2 shows three models learned by the algorithm. Note that the training data only cover some part of the input space. However, by using two-dimensional projections and visualizing the class boundary in the whole 2D-space, the extrapolation behavior can be verified. The features of each model are selected from the set of $\binom{52}{2} = 1326$ combinations of the original, physically interpretable features.

3 Safe and Interpretable Regression

As explained above in the context of Fig. 1 it is difficult and often practically impossible to understand a neural network model on such a detailed level that it would be possible to make statements about its extrapolation behavior. The same holds for many other successful approaches like Support Vector Machines [16] and Gaussian Processes [14].

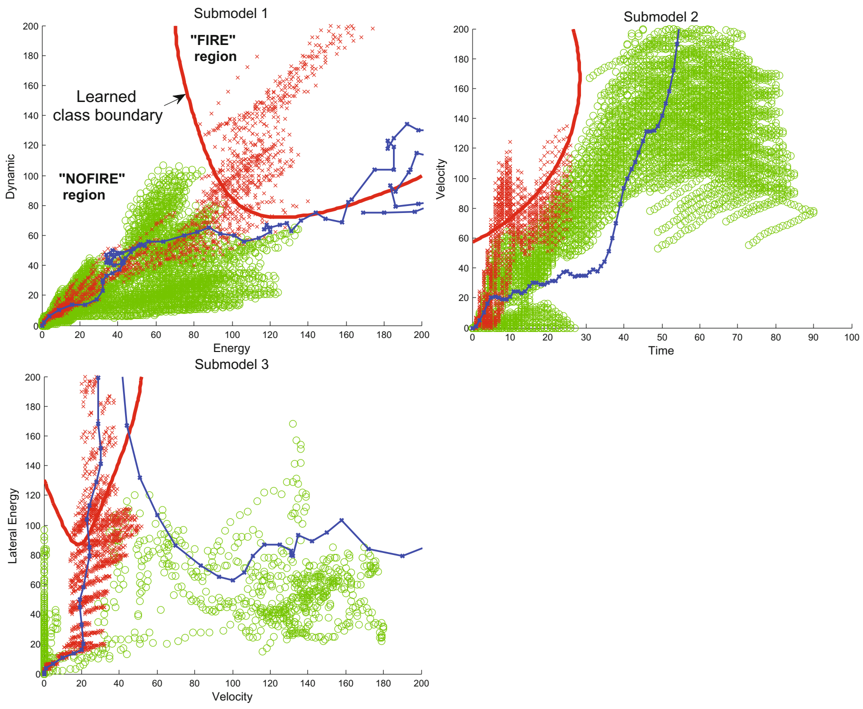


Fig. 2 Three airbag control submodels, each separating a certain “fire” region from a “nofire” region in a 2D-projection of the input space. Training samples from “nofire” crashes are marked as (green) circles. Samples from “fire” crashes are marked as (red) crosses. The thick line indicates the learned class boundary in each model. By explicitly visualizing the boundary in the complete input space of each model the extrapolation behavior can be verified. Note that large regions of the input space are not filled by data. Thus, manually checking the extrapolation behavior is mandatory. For illustration, the trajectory of a particular “fire” crash is shown in the respective 2D-projections. It starts in the origin and enters the “fire” region in the third model (second row, left) and some time later also in the first model (first row, left). A “fire” decision by one model is sufficient.

Traditionally, approaches like the **C**lassification **A**nd **R**egression **T**rees (CART) [4] or rule-based methods [11] are considered as being interpretable. There is of course a trade-off with accuracy. For example, a CART model only remains interpretable as long as the number of tree nodes is rather small, which means that the model is quite coarse. In contrast, Random Forests [3] provide the other extreme with a good reputation in terms of accuracy but without any kind of interpretability.

A possible compromise is to partition the input space similar to CART or rule learners but to use slightly more complex submodels in the different regions of the input space. MARS (multivariate adaptive regression splines) [6] and an approach called GUIDE (generalized unbiased interaction detection and estimation) [10] follow this strategy; both are discussed in an experiment below.

Symbolic regression [15] comes from a different direction and seeks for a symbolic representation (i.e. an equation) that best matches the given data. Details are given later on.

In statistics, additive models (see e.g. chapter 9 in [7]) are often applied when the model shall be interpretable. The idea is to learn a multivariate function having p inputs as a sum of p univariate functions, that is, $y = \alpha + \sum_{i=1}^{p} f_i(x_i) + \varepsilon$. The advantage is that the univariate functions may be easier to interpret. The drawback, however, is that interactions between variables cannot be modeled.

In the experiment below we propose a different strategy which is similar to an additive model in the sense that two modular functions are added in the final model. The first function is used as an analytical model learned by symbolic regression; it is easily interpretable but has only moderate accuracy. Thus, a second function is learned on the residuals of the first function in order to improve the accuracy. The model of the second function is not interpretable; but by limiting its output to a defined range a worst-case guarantee can be given in the sense that the maximal deviation from the analytical model is always below a certain limit.

The objective of the following experiment is to compare different regression approaches in terms of their accuracy and interpretability. The results are discussed in Sect. 3.2.

3.1 Experiment: SARCOS Benchmark

In this example we consider learning the inverse dynamics of a seven degrees-of-freedom SARCOS robot arm[1]. The task is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. Following previous studies on this benchmark (see references in [14]) we only consider the mapping to the first of the seven torques, that is, we learn a function $f : \mathbb{R}^{21} \to \mathbb{R}$. There are 44,484 training examples and 4,449 test examples. All 21 inputs x_i have been standardized to have zero mean and standard deviation 1. The output y has zero mean. Results are given as *standardized mean squared error* (SMSE), which is the mean squared error on the test set divided by the variance of the target values in the test set. The normalization by the variance makes the error measure independent on the overall scale of the target.

While in a real safety-related application the *worst-case error* should be considered additionally to the mean error, here we solely use the SMSE to facilitate the comparison with previous studies.

All methods described in the following were applied to this benchmark. The resulting SMSE accuracy is summarized in Table 1.

Rigid-body-dynamics (RBD): This is a physics-based model derived from rigid-body-dynamics. The RBD result is taken from [14], p. 24. As shown in Table 1 the model accuracy is rather poor, which may be explained by the fact that the robot

[1] Named after the robotics company SARCOS. The benchmark data are publicly available from `http://www.gaussianprocess.org/gpml/data/`

is actuated hydraulically and is rather lightweight, so some rigid-body assumptions seem to be violated.

Linear Regression (LR): Linear model with 21 inputs and no intercept.

GUIDE [10]: Similar to the classical CART this is a recursive partitioning algorithm creating a regression tree. In CART a leaf of the regression tree contains just the mean output value of all training samples assigned to that leaf. In contrast, in GUIDE the leaves may contain linear models either with all inputs or with a subset of inputs. With the complexity of the submodels in the leaves being larger in GUIDE, the overall number of nodes can often be kept smaller in comparison to CART, which may improve the interpretability of the final model.

Note that switching between nodes may lead to discontinuities in the output, possibly causing problems in control applications. This holds for other approaches as well that use a hard partitioning of the input space, e.g. CART, MARS.

Three experiments were conducted. They differ in the type of linear models used in the terminal nodes (leaves). In the second experiment the minimum node size was raised; the parameter defines the minimum number of training samples that a node must have. Increasing this number reduces the number of nodes and helps in improving the interpretability. The lowest error is achieved in the third experiment, which is the value considered in Table 1. However, the model is not easy to interpret: it consists of 35 linear models each having 21 variables.

Experiment 1: Stepwise linear models with up to 5 variables. Using defaults (minimum node size = 889). Number of terminal nodes of final tree: 27, SMSE = 0.0411

Experiment 2: Stepwise linear models without limiting the number of variables (typically 11-17 variables entered the models in the leaves). Minimum node size set to 2000. Number of terminal nodes of final tree: 16, SMSE = 0.0403

Experiment 3: Multiple linear models, all with 21 variables, using defaults (minimum node size = 889). Number of terminal nodes of final tree: 35, SMSE = 0.0327

MARS: Multivariate adaptive regression splines [6] are an approach where a model is build as a linear combination of basis functions. The simplest basis function is a piecewise linear function (linear spline) $h(x_i)$ of one input variable; instead of linear splines cubic splines may also be used. Interactions between two inputs x_i and x_j are modeled by the product $h(x_i)g(x_j)$ of two univariate spline functions. The resulting product is considered as a basis function again. Higher-order interactions can be handled analogously by building the respective products. Usually the maximum interaction level is limited to aid in the interpretation of the final model. A key property of the basis functions is that they are zero over some part of their range, making it possible for them to operate locally.

We used the ARESLab[2] toolbox ver. 1.5.1 in Matlab. The maximum number of basis functions allowed to be included in the model was set to 21 (including the

[2] ARESLab obtainable from `http://www.cs.rtu.lv/jekabsons/`

intercept term) and the maximum interaction level was limited to 2 (only allowing pairwise products). The resulting model trained on the complete training set is shown below.

```
y = 21.51 +22.19*BF1 -25.44*BF2 +4.49*BF3 +5.88*BF4 +12.61*BF5 -8.45*BF6 +3.03*BF7 -14.14*BF8
+2.31*BF9 +1.63*BF10 +7.37*BF11 +2.69*BF12 -4.02*BF13 -3.20*BF14 -1.88*BF15 -2.37*BF16 -2.11*BF17
-4.43*BF18 +1.98*BF19 +1.60*BF20
```

where all basis functions are either univariate cubic splines or products of two univariate cubic splines. The model is rather difficult to interpret.

Gaussian Process (GP): A GP is a linear smoother using a weighted average of the stored training outputs **y** to predict the output for a test input. It can be seen as a linear combination of n kernel functions, each one centered on one of n training points [14],

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \quad \alpha = (K + \sigma_n^2 I)^{-1}\mathbf{y} \tag{1}$$

where $\mathbf{x}$ is a test input, $\mathbf{x}_i$ are the training inputs and α is a weight vector with kernel matrix K and a hyperparameter σ_n.

We used the squared exponential kernel where each input dimension is scaled by an individual factor, allowing the down-weighting of irrelevant inputs. The kernel function is given as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T D^{-2} (\mathbf{x} - \mathbf{x}')\right)$$

with diagonal matrix $D = \mathrm{diag}(\ell_1, \ldots, \ell_{21})$ containing the scaling factors of the input dimensions. In total 23 hyperparameters $(\ell_1, \ldots, \ell_{21}, \sigma_f, \sigma_n)$ were optimized during training.

Note that the GP cannot be trained on the complete training set as the kernel matrix would be too large to compute the inverse in Eq. (1). Thus, we randomly draw a subset of 4000 samples from the original training set and trained the GP on the reduced set. The GP was then applied to the test set and the total procedure was repeated ten times. Table 1 shows the average and standard deviation of the ten runs. The SMSE is in good accordance to the result reported in [14], p. 182. It is possible to slightly reduce the error further by replacing the random subsampling with more sophisticated methods. This has not been investigated in this paper, for details we refer to [14].

In terms of the interpretability of the model the GP has to be considered as a "black box" approach as the model of Eq. (1) is a linear combination of 4000 kernel functions.

Eureqa: Eureqa is a tool for symbolic regression[3], where the training data are used to search for an explicit symbolic relationship of the form $y = f(\mathbf{x})$. Based on genetic programming the search space of equations consisting of pre-specified "building

[3] `http://creativemachines.cornell.edu/eureqa`

blocks" (e.g. arithmetic operators, sin, exp) is explored to find an equation best matching the data [15]. We used the mean-squared error to guide the search on the complete training set and took the following model having the smallest training set error.

$$\begin{aligned}&\text{y = x2 + 25.31*x15 + 11.08*x1 + 11.08*x18 + 1.732*x21 + x1*x15 + x4*x21}\\&\text{- x11 - x2*x15 - x4*x18 - 1.732*x8*x9}\end{aligned} \tag{2}$$

Note that this representation is more compact and much more interpretable than the MARS model. Unlike the LR model the Eureqa model includes several terms with two interacting variables, so it is not surprising that the model achieves a better test set SMSE than the LR model as shown in Table 1.

A certain drawback of symbolic regression is that it is computationally intensive; we spent about 60 hours search time on a 16 CPU cores computer. After ≈ 35 hours there were only little improvements so the search could have been stopped earlier. Given the much better interpretability of Eq. (2) in comparison to the GUIDE and MARS models, the higher search time may be acceptable.

Eureqa + GP: Here we propose a new approach where a Gaussian Process (GP) is used to learn the residuals of the Eureqa model. The basic idea is to take the Eureqa model as an easily verifiable analytical model and to improve the accuracy by a Gaussian Process. Thus, the overall model has the form $y = f(\mathbf{x}) + r(\mathbf{x})$ where f is the Eureqa equation and r is the GP model. The model r was trained on $n = 4000$ randomly drawn residuals $y^{tar} - f(\mathbf{x})$ where y^{tar} are the target output values of the training set. The experiment was repeated 10 times to avoid a sampling bias. The number n controls the accuracy of the GP residual model as in the case of the pure GP. A much smaller number of training samples, e.g. $n = 2000$, would yield a coarser model with less accuracy.

Note that in terms of safety little is gained unless the amount of correction r is limited. Otherwise, the output of the overall model may be arbitrarily wrong because it is hardly possible to fully verify the GP model as explained in the GP section above. Therefore, the output r is limited, giving the overall model $y = f(\mathbf{x}) + r^*(\mathbf{x})$ with $r^*(\mathbf{x}) = \max(\min(r(\mathbf{x}), \gamma), -\gamma)$. The limit $\gamma \in \mathbb{R}$ provides a worst-case guarantee on the error of the overall model. When choosing the limit, there is a trade-off between possible accuracy gains and the worst-case guarantee.

In the SARCOS training set the output is in the range $[-108.5, 107.7]$. We therefore evaluated the following limits $\gamma \in \{5, 10, 15, \infty\}$. For example, $\gamma = 10$ means that the maximal possible deviation from the analytical model is ± 10, which is less than ten percent of the maximal output values. We also tested the unlimited case, which is denoted as $\pm\infty$.

As shown in Table 1 the tightest limit ± 5 already improves the accuracy considerably in comparison to the pure Eureqa model. The error is reduced further when the limits are relaxed and at ± 15 the same accuracy as of the pure GP is achieved. Results show average and standard deviation over 10 runs.

Table 1 Results on the inverse dynamics problem. The error is given as standardized-mean-squared error (SMSE) on the test set. The third column refers to the interpretability of the resulting model.

Method	SMSE	Interpretability
RBD	0.104	very high
LR	0.075	high
GUIDE	0.033	moderate
MARS	0.059	moderate
GP	0.020 ± 0.001	very low
Eureqa	0.062	very high
Eureqa + GP (± 5)	0.028 ± 0.000	high
Eureqa + GP (± 10)	0.021 ± 0.001	high
Eureqa + GP (± 15)	0.020 ± 0.001	(high)
Eureqa + GP ($\pm \infty$)	0.020 ± 0.001	very low

3.2 Discussion

The best accuracy in this benchmark is achieved by Gaussian Processes (GP). However, as seen in Eq. (1) a GP consists of in our case 4000 kernel functions centered on the respective training points, so the model is not interpretable. In principle, a GP is able to additionally provide confidence estimates; but these may be wrong, for example, if the model is slightly misspecified. Therefore, it is problematic to apply a pure GP in safety-related applications.

The best interpretability among the data-driven approaches is achieved by the symbolic regression model of Eureqa. A domain expert should be able to verify the learned equation. In terms of accuracy the Eureqa model is better than the linear regression model (LR) but worse than the more complex models of MARS or GUIDE. However, the latter two are difficult to interpret.

The physics-based RBD model shows the worst accuracy on the test data, possibly due to violations of rigid-body assumptions. This clearly shows the improvements that are possible by using data-driven models.

The proposed combination of the Eureqa model with a GP greatly improves the accuracy. Limiting the possible contribution of the GP to a certain range gives a worst-case guarantee in the sense that the maximal deviation from the verified analytical model is always below that limit. Thus, the proposed approach may be very interesting for safety-related applications where these kinds of guarantees are mandatory.

The combination scheme can also be translated to other methods. For example, instead of using a GP, the residuals could be learned by support vector regression [16], which may be beneficial due to the automatic selection of support vectors. On the other hand, the verified model could be a rule-based or tree-based model instead of the symbolic regression equation as long as the model is truly interpretable.

4 Conclusions

Understanding what a model has learned and in particular how it will extrapolate to unseen data becomes a crucial concern if the model correctness must be ensured not only for the available data but for all possible input combinations. The paper has reviewed an approach for learning a classifier for safety-related applications. The basic idea in that case is to solve the problem by splitting it into several low-dimensional subproblems and to visualize the decision boundary of the classifier in the whole low-dimensional input space of the respective submodels.

For regression problems similar concepts exist. MARS and the tree-growing approach GUIDE use submodels which are valid only in certain regions of the input space. They have been compared together with other regression methods on the SARCOS data benchmark. In that application both MARS and GUIDE provide only moderate interpretability due to a large number of submodels. An excellent interpretability is achieved by symbolic regression; however, this comes with the price of a reduced accuracy. A good trade-off is provided by combining the symbolic model with a Gaussian Process that learns the residuals of the symbolic model. The output of the Gaussian Process model, that is, the amount of correction applied to the symbolic model is limited to a certain range. The combined model has an improved accuracy and provides error bounds in the sense that the deviation from the verified symbolic model is always kept below a defined limit.

References

[1] Andrews, S., Tsochantaridis, I., Hofmann, T.: Support vector machines for multiple-instance learning. In: Advances in Neural Information Processing Systems, vol. 15, pp. 561–568. MIT Press, Cambridge (2003)

[2] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)

[3] Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)

[4] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Statistics/Probability Series. Wadsworth Publishing Company, Belmont (1984)

[5] Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)

[6] Friedman, J.H.: Multivariate Adaptive Regression Splines. The Annals of Statistics 19(1), 1–67 (1991)

[7] Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The elements of statistical learning: data mining, inference and prediction. Springer, New York (2009), http://www-stat.stanford.edu/~tibs/ElemStatLearn

[8] Lang, B.: Monotonic Multi-layer Perceptron Networks as Universal Approximators. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 31–37. Springer, Heidelberg (2005), doi:10.1007/11550907

[9] Lisboa, P.J.G.: Industrial use of safety-related artificial neural networks. Contract research report 327/2001. Liverpool John Moores University (2001)

[10] Loh, W.: Regression by parts: Fitting visually interpretable models with guide. In: Chen, C., Härdle, W., Unwin, A. (eds.) Handbook of Computational Statistics, pp. 447–468 (2008)

[11] Mitra, S., Hayashi, Y.: Neuro-fuzzy rule generation: Survey in soft computing framework. IEEE Transact. Neural Networks, 748–768 (2000)
[12] Nusser, S., Otte, C., Hauptmann, W.: Interpretable ensembles of local models for safety-related applications. In: Proceedings of 16th European Symposium on Artificial Neural Networks (ESANN 2008), Brugge, Belgium, pp. 301–306 (2008)
[13] Nusser, S., Otte, C., Hauptmann, W., Kruse, R.: Learning verifiable ensembles for classification problems with high safety requirements. In: Wang, L.S.L., Hong, T.P. (eds.) Intelligent Soft Computation and Evolving Data Mining: Integrating Advanced Technology, pp. 405–431. IGI Global (2009)
[14] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
[15] Schmidt, M., Lipson, H.: Distilling Free-Form Natural Laws from Experimental Data. Science 324(5923), 81–85 (2009)
[16] Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
[17] Silverman, B.: Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). Chapman and Hall/CRC (1986)
[18] Taylor, B.J. (ed.): Methods and Procedures for the Verification and Validation of Artificial Neural Networks. Springer (2005)