Advanced Engineering Informatics xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Model-Tree Ensembles for noise-tolerant system identification ☆

Darko Aleksovski [a,c,*], Juš Kocijan [b,d], Sašo Džeroski [a,c]

[a] Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[b] Department of Systems and Control, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[c] Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana, Slovenia
[d] University of Nova Gorica, Vipavska 13, Nova Gorica, Slovenia

ARTICLE INFO

Article history:
Received 29 August 2013
Received in revised form 9 June 2014
Accepted 26 July 2014
Available online xxxx

Keywords:
Decision tree ensemble
Fuzzified model tree
Nonlinear dynamic system identification

ABSTRACT

This paper addresses the task of identification of nonlinear dynamic systems from measured data. The discrete-time variant of this task is commonly reformulated as a regression problem. As tree ensembles have proven to be a successful predictive modeling approach, we investigate the use of tree ensembles for solving the regression problem. While different variants of tree ensembles have been proposed and used, they are mostly limited to using regression trees as base models. We introduce ensembles of fuzzified model trees with split attribute randomization and evaluate them for nonlinear dynamic system identification.

Models of dynamic systems which are built for control purposes are usually evaluated by a more stringent evaluation procedure using the output, i.e., simulation error. Taking this into account, we perform ensemble pruning to optimize the output error of the tree ensemble models. The proposed Model-Tree Ensemble method is empirically evaluated by using input–output data disturbed by noise. It is compared to representative state-of-the-art approaches, on one synthetic dataset with artificially introduced noise and one real-world noisy data set. The evaluation shows that the method is suitable for modeling dynamic systems and produces models with comparable output error performance to the other approaches. Also, the method is resilient to noise, as its performance does not deteriorate even when up to 20% of noise is added.

1. Introduction

In this paper, we address the task of identification of nonlinear dynamic systems from measured input–output data. In particular, we address the discrete-time variant of this task, which can be transformed into a regression problem of predicting the next state/output of the system from states and inputs in the recent past. Different regression approaches have been used for solving this task, including neural networks, support vector machines and Gaussian process regression. While most approaches to solving this task try to minimize the one-step prediction error, the learned models are typically evaluated in terms of their simulation (output) error.

We explore the use of tree ensemble methods for regression for modeling dynamic systems from measured data. We propose a novel approach for learning ensembles of model trees with randomized attribute selection and fuzzified splits. The approach includes an optimization step of ensemble pruning, which is based on the simulation (output) error criterion. We evaluate the performance of our Model-Tree Ensembles, comparing them to existing state-of-the-art methods used for system identification, focusing on their performance on noisy identification data.

The remainder of this section first introduces the task of discrete-time modeling of dynamic systems. It then discusses existing approaches to solving this task and some of their shortcomings. It next discusses tree ensemble approaches for regression that we plan to use for overcoming these deficiencies. The section concludes by laying out the aims of this paper and giving an outline of the remainder of the paper.

1.1. Discrete-time modeling of dynamic systems

The task of discrete-time modeling of nonlinear dynamic systems using measured input–output data is to find difference (recurrence) equations using the input variable (u) and output and system variable (y). These equations describe the system at a

☆ Handled by C.-H. Chen.

* Corresponding author at: Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia.
E-mail addresses: darko.aleksovski@ijs.si (D. Aleksovski), jus.kocijan@ijs.si (J. Kocijan), saso.dzeroski@ijs.si (S. Džeroski).

http://dx.doi.org/10.1016/j.aei.2014.07.008
1474-0346/

time instant k using past values of the input and output variables. Through the external dynamics approach [1] the modeling problem is reformulated as a regression task. The value of the system variable(s) at time instant $k, y(k)$, needs to be predicted from the lagged values of the input and system variable(s), $u(k-1), u(k-2), \ldots, u(k-n), y(k-1), y(k-2), \ldots, y(k-n)$ using a static function approximator.

The evaluation of the performance of a dynamic system's model is carried out according to the purpose of the model and often requires a stringent and purpose-specific evaluation. When evaluating a model using one-step-ahead prediction, as shown in Fig. 1(a), the predicted values for the system variable are compared to the measured values. On the other hand, the procedure of simulation, illustrated in Fig. 1(b), introduces one substantial difference: the one-step-ahead model predictions are fed back to the model to produce predictions for the more distant future.

While the first step of one-step-ahead prediction and simulation is the same, in simulation, the predicted value of the system variable y at time k ($\hat{y}(k)$ is fed back as input to the model, instead of a measured value ($y(k)$) at time $k+1$. Due to the realistic possibility of error accumulation in the case of an inaccurate model, divergence of the simulation predictions from the measured values may occur as we move further into the future. The cumulative error of simulation is referred to as the output error, while in the case of one-step-prediction the error is referred to as prediction error.

1.2. Methods for discrete-time modeling of dynamic systems

The task of discrete-time modeling of nonlinear dynamic systems from measured data can be approached using different modeling techniques. Over the last few decades, numerous different methods have emerged. The earlier approaches included for example the block-oriented Hammerstein and Wiener systems [2] and the semi-parametric Volterra method [3]. More recent approaches include the widely used basis-function approaches of Artificial Neural Networks [1] and fuzzy modeling, as well as the nonparametric approaches of kernel methods [4] and Gaussian Process models [5], to list just a few.

Existing approaches for dynamic systems identification can be classified according to the type of model they produce. Some approaches learn one global model describing the whole system, while other learn multiple-models. The following paragraphs describe the features that distinguish methods belonging to the two categories.

Methods that build one global model include, for example the well-known Artificial Neural Networks [1], Gaussian Process models [5], and support vector regression [4]. They learn one global model by using a nonlinear optimization procedure on all identification (training) data in a single optimization task. The learned model is valid in the whole operating region.

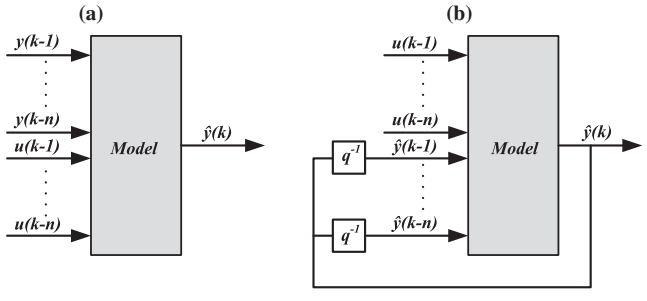

Fig. 1. (a) A model for prediction; (b) the model used for simulation; (q^{-1} is the backshift operator).

Artificial neural networks, which can be seen as universal approximators, are very powerful and flexible methods also used for modeling (nonlinear) dynamic systems [6]. Different architectures of neural networks exist, the most common ones being multilayer perceptron (MLP) networks and radial basis function (RBF) networks. However, in spite of their advantages, as for the other approaches in this class, their main disadvantages are the lack of transparency and curse of dimensionality [7].

Gaussian Process models are nonparametric, probabilistic black-box models that have been used for modeling dynamic systems [5]. One of their advantages is the measure of confidence for the predictions they provide, which helps in assessing the quality of the model prediction at each point. This approach is related to Support Vector Machines and especially to Relevance Vector Machines [5].

The multiple model approaches build several local models, each of them valid in a subregion of the whole operating region. They are also referred to as local model networks [8]. They include neuro-fuzzy approaches like the Adaptive Neuro Fuzzy Inference System – ANFIS [9], Local Linear Model Trees – Lolimot [1], and other methods based on the operating regime approach [10].

The determination of the subregion boundaries and the subset of features used for each subregion is frequently referred to as structure determination and is usually the first subproblem these methods address. The second subproblem is the identification of local model parameters. As expected, different solutions exist for these two subproblems, while some iterative methods even solve both subproblems at once [1]. Out of the plethora of possible solutions for the structure identification part, the most frequently used techniques are: grid partitioning, tree-based partitioning, fuzzy clustering, product space clustering, genetic algorithms, and partitioning based on prior knowledge [8]. The local model identification subproblem is solved by using either local or global least squares optimization methods, the choice of which has an effect on the accuracy and interpretability of the local models.

1.3. Challenges in system identification

1.3.1. Noisy data

In practice, the measured input–output data used for identification of the dynamic systems is disturbed by noise. The quality of the identified model is dependent on the resilience of the learning method to noisy data. Existing approaches for modeling from data handle the overfitting-to-noise issue using different strategies: probabilistic strategy, strategies including averaging of predictions, or pruning over-complex models. These meet with a variable degree of success and handling noise remains a challenging issue.

1.3.2. The curse of dimensionality

Modeling dynamic systems in some areas (such as aerospace engineering) requires high-dimensional models. Further more, discrete-time modeling of such systems, and in particular the introduction of lagged variables using the external dynamics approach, increases the dimensionality of the data used in the identification process. This can present a challenge for many modeling methods.

Existing approaches scale differently as data dimensionality (number of variables and number of data points) increases [7]. Artificial Neural Networks with their two most common architectures multilayer perceptron and radial basis functions have as disadvantages the curse of dimensionality and lack of transparency. One of the widely used fuzzy modeling approaches, ANFIS [9], also suffers from the dimensionality problem: the number of parameters that it needs to estimate is proportional to p^n where n is the number of input variables and p is the number of membership

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

functions assigned to each variable. The learning of Gaussian Process models [5] is limited to at most a few thousand data points.

1.3.3. Optimization of output error

Most approaches to modeling dynamic systems minimize prediction error during learning (parameter optimization or structure determination). However, the validation of the learned model is typically performed by simulation. This presents another challenge and raises the question whether it is possible to directly optimize the output error while learning, instead of optimizing the prediction error.

In this context, the works of Nelles [1] (cf. also [11]) conclude that a decrease of the prediction error does not necessarily lead to a decrease in the output error of the model, when using neural-networks as models. Similarly, Kocijan and Petelin [12] deal with the same question in the context of Gaussian Process models. They conclude that the direct optimization of output error is a much harder task as compared to the optimization of prediction error, since the optimization surface in the former case contains many local optima and its contour plots depict quite irregular shapes.

However, at least some nonlinear identification methods make use of the output error for model structure selection. An example is the Local Linear Model Trees – Lolimot method [1], which iteratively adds complexity to a tree structure. In each iteration, the method solves the parameter optimization problem using least squares estimation, evaluates the intermediate model using simulation, and tries to improve the structure by adding one more node to the tree. The author concludes that the structure search, a nonlinear optimization problem solved by a greedy hill-climbing approach, could benefit from directly estimating the simulation error. This approach is possible because: (a) the iterative nature of the approach means that after each iteration an intermediate solution, i.e., a fuzzy model tree, is ready to be used; and (b) the number of iterations, or total number of nodes in the tree is typically not large, so the time-consuming evaluation of the output error on the whole identification set does not increase the overall time complexity substantially.

1.4. Tree ensembles for regression and system identification

As outlined above, early research in discrete-time system identification focused on parametric and semi-parametric regression approaches. More recently, non-parametric approaches dominate, including Artificial Neural Networks, kernel methods, and Gaussian Process models. In this context, we propose the use of another non-parametric approach to regression, coming from the area of machine learning, namely tree ensembles.

As tree ensembles are a very successful predictive modeling approach, we propose the use of tree ensembles for regression for the task at hand. Ensembles of regression trees have been extensively used and different variants thereof explored (such as bagging and random forests). Bagging of model trees has also been considered.

Trees and tree ensembles have a number of properties that directly address some of the challenges mentioned in the previous section. On one hand, this includes the handling of noisy data by tree pruning. On the other hand, this includes the efficiency of tree induction. Finally, this includes the predictive power and efficiency of tree ensembles, such as random forests. The following subsections outline the advantages of using tree based methods and tree ensembles, and present the obstacles that need to be overcome for their successful application to system identification tasks.

1.4.1. Advantages of tree based methods

The issue of overfitting to noise in decision trees and tree ensembles is controlled by the depth of the individual tree or the trees in the ensemble [13]. Larger trees are more sensitive to noise and prone to overfitting, while smaller trees are less sensitive and less prone to overfitting. Tree pruning procedures can be used to reduce the depth of an overly large tree, and control the overfitting to noise.

Tree learning algorithms are robust and have the potential to scale well to large predictive modeling problems. The algorithms apply the divide-and-conquer principle, by splitting the available learning data into smaller subsets as the tree is constructed. Thus, the potentially complex optimization problem is broken down to several simpler optimization subproblems. Each optimization subproblem uses a proper subset of the whole set of identification data, so the identification procedure is simplified. This gives the tree learning algorithms the ability to efficiently handle a large number of training instances (identification points).

Tree based methods are not sensitive to irrelevant attributes/features included in the data. Recently they have been also extended to deal with multiple dependent/target variables (cf. e.g., [14]), making them suitable to handle (with appropriate modifications) the multi-output case of system identification.

1.4.2. Advantages of tree ensembles

Ensembles for regression, also called committees of predictors, are known to improve predictive accuracy. This is known in the field of neural-network ensembles [15] as well as tree-based ensembles [16]. Among the reasons for their success are the smoothing effect on individual model estimates and the reduction of variance of the ensemble [17], as compared to the one of the individual trees.

Motivated by the success of the bagging strategy, Breiman [18] concluded that adding randomness within the base learners can be beneficial for the tree ensembles. The randomness increases the diversity of the ensemble and allows for even more accurate predictions. The introduction of randomness into the tree building algorithm is achieved by randomly choosing a different subset of features in each internal node to be used as candidate split variables. The methodology is known as Random Forests [18] and has been shown to perform well as compared to many other methods: discriminant analysis, support vector machines, and neural networks. Such an ensemble method can also handle very large numbers of attributes/features, thus effectively dealing with the curse of dimensionality.

However, both bagging and Random Forests have been originally designed to use regression trees. Since bagging only manipulates the training data, and does not randomize the base learners, it can be used with any base learner, including algorithms that learn model trees. The only other ensemble method for regression that utilizes model trees, to our knowledge, is the semi-random model tree ensemble approach [19]. This approach modifies the base-level tree-learning algorithm to produce balanced trees: the number of points falling in each terminal node of the tree is approximately the same. This approach is thus not well-suited for dynamic systems, whose identification is performed on data that are not evenly distributed in the instance space. The partitioning of the instance space using semi-random model tree ensembles would be denser around the equilibrium points, as these regions contain more points than the out-of-equilibria regions. As a consequence, the critical out-of-equilibria regions would be covered by a small number of partitions, resulting in poor approximations.

1.4.2.1. Optimization of the ensemble. Several researchers using ensembles for regression have concluded that the prediction performance of the ensemble increases when the ensemble structure

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

is optimized. In the context of neural-network ensembles for regression, the work of Perrone and Cooper [20] uses weighted averaging of the base model predictions. The weights are optimized, such that the squared prediction error is minimized. The work of Aho et al. [21] (learning rule ensembles for regression) also includes an ensemble optimization step, which selects the best subset of rules for the ensemble and determines their respective weights.

A special case of weighting uses only zero/one weights. In this case ensemble optimization means reducing the number of models in the ensemble. This is called ensemble selection [22] or ensemble pruning [23,24] and has been shown to improve the overall performance of the ensemble.

1.4.3. Trees and tree ensembles for system identification

Besides their potential advantages, there are several possible obstacles to the use of trees and tree ensembles for regression for the task of system identification. They include the crisp decision boundaries and the extrapolation behavior. Also, these methods are aimed at optimizing prediction (and not simulation/output) error during learning.

The task of modeling dynamic systems requires a modeling procedure which produces a smooth function/mapping between the inputs and the outputs, i.e. a smooth fit. Tree-based models have crisp decision boundaries in the internal nodes, causing non-smooth responses. These discontinuities at the boundaries could increase the prediction error and potentially degrade the simulation performance. One possible approach to solving this problem is to use smoothing with the help of fuzzy set theory.

Several different tree learning algorithms have been introduced, which build upon the ideas of fuzzy set theory and produce smooth responses. Most of the related work considers smoothing regression tree predictions (constant predictions of the local models), and only some of it deals with smoothing the predictions of linear model trees. For example, some methods [25] start by learning a crisp tree, convert the splits from crisp to fuzzy, and optimize the fuzzy parameters. Other fuzzy tree methods learn directly both the tree structure and fuzzy parameters [26].

Regression trees (and ensembles thereof) have poor extrapolation behavior. For inputs falling outside the range of the identification data, predictions of the model are limited to constant values. One possible solution is the use of linear model trees instead of regression trees. Also, as outlined above, the existing tree and tree ensemble methods optimize the prediction error. We make an effort to optimize the simulation error during the learning of the ensemble model.

1.5. Aim and outline of the paper

In this paper, we set out to explore the use of tree ensemble methods for regression for the task of modeling dynamic systems from measured data. Aiming to exploit the above-mentioned strengths of tree ensemble methods and overcome their deficiencies in this context, we propose a novel approach for learning ensembles of model trees. The individual trees are built with randomized attribute selection and have fuzzified splits. The approach also includes an optimization step of ensemble pruning, which is based on the simulation (output) error criterion. We evaluate the performance of our Model-Tree Ensembles, comparing them to existing state-of-the-art methods used for system identification, focusing on their performance on noisy identification data.

The remainder of the paper is organized as follows. Section 2 presents the Model-Tree Ensembles (MTE) method for regression. This section describes the proposed MTE algorithm and illustrates the methodology on a simple static function approximation problem. Section 3 presents the experimental evaluation on noisy data, using two data sets, and discusses the empirical results. Section 4 presents further discussion of the suitability of the proposed method for dynamic system identification, states our conclusions and outlines some directions for further work.

2. Model-Tree Ensembles

In this section, we present our approach to learning fuzzified model tree ensembles, which includes randomized attribute selection and ensemble pruning. The Model-Tree Ensembles are illustrated on a simple example. A preliminary version of this approach was considered earlier by Aleksovski et al. [27], however it was evaluated using only prediction error (and not output error). Also, it did not optimize the ensemble structure by performing ensemble pruning, a procedure designed toward optimizing the output error.

Three key features distinguish our approach from other existing machine learning approaches. First, the split attribute randomization is unique, as described below. Second, after each tree is learned, the conditions in its internal nodes are fuzzified. Finally, a subset is selected from all the trees learned and only the selected trees form the final ensemble.

The remainder of this Section describes the ensemble construction, the model tree learning algorithm and the ensemble selection employed. Finally, it presents the computational complexity of the method and illustrates its properties on a simple modeling problem.

2.1. Ensemble construction

The ensemble construction procedure we use is based on the popular bagging approach. It creates M bootstrap replicates, i.e., random samples with replacement, of the identification set, which have an equal number of data points as the identification set. Using each of the M samples, a base learner algorithm is used to build a collection of M model trees: $f_1, f_2, \ldots, f_M$. Finally, the ensemble structure is optimized, using an ensemble selection procedure, as described in Section 2.3.

The final ensemble model is used for prediction by averaging the predictions of each of the base models. The pseudocode describing the ensemble construction and ensemble selection procedures is shown in the upper and lower parts of Table 1, respectively. The base learner algorithm used is a modification of the M5′ model tree algorithm [28] and is described in the next subsection.

2.2. Learning fuzzified model trees

Model trees are hierarchical structures of nodes, where the inner nodes consist of feature/attribute tests, while the terminal nodes consist of local models: functions of the features. The splits in the inner nodes of the tree perform a partitioning of the instance space into a set of partitions, for which local models are learned. Regression trees have constants in the terminal nodes, while model trees typically use linear models with a constant term. A simple tree is shown in Fig. 2(a), while Fig. 2(b) depicts the partitioning of the two-dimensional instance space represented by this tree.

The algorithm that learns a fuzzified model tree with split attribute randomization, to be used as a component model of the ensemble, is based on the M5′ model tree learning method. Its operation can be divided into three phases: tree growing phase, tree pruning phase and tree fuzzification. The pseudocode of the algorithm, including all three phases, is given in Tables 2 and 3. Note that the novel parts we introduce concern the randomized selection of attributes, and the fuzzification of model trees.

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

Table 1
Pseudocode for the Model-Tree Ensembles method.

Learn_ensemble(D)
Inputs: data set D
Outputs: an ensemble E

Create M bootstrap samples of D: $D_1, D_2, \ldots, D_M$
Build a tree using each of the M samples: T_i = **Learn_model_tree**(D_i)
Let E' = **Ensemble_selection**($\{T_1, T_2, \ldots, T_M\}$, D)
Return the ensemble E'

Ensemble_selection(E, D)
Inputs: ensemble E, consisting of trees $T_1, T_2, \ldots$, identification set D
Outputs: ensemble E'

Let err_{full} be the output error of E, obtained by simulation on data set D
Let $M = |E|$
Create M ensembles, $E_1, \ldots, E_M$ where $E_i = \{T_j | j \neq i\}$
Let err_i be the output (simulation) error of ensemble E_i
Let $err_{reduced} = min_{i=1,\ldots,M}(err_i)$ and $j = argmin_{i=1,\ldots,M}(err_i)$
If $(err_{full} > err_{reduced})$
  Return the ensemble **Ensemble_selection**(E_j, D)
Otherwise
  Return the ensemble E

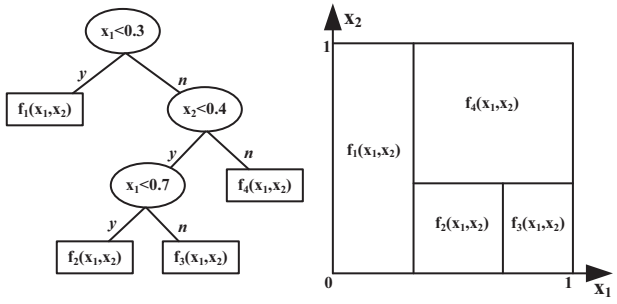

Fig. 2. A tree with 4 local models and the corresponding partitioning of the instance space. $\mathbf{x} = [x_1, x_2]$ is the two-dimensional feature vector.

Table 2
Pseudocode for learning a fuzzified model tree.

Learn_model_tree(D)
Inputs: an identification data set D
Outputs: a tree T

Let T be an empty tree
T = **Grow_tree**(D)
T = **Prune_tree**(T)
T = **Fuzzify_tree**(T)

Grow_tree(D)
Inputs: an identification data set D
Outputs: a pointer to the root node of the induced tree
If $|D| < N$
  Return a terminal node
If the st.dev. of the target var. on set D
is less than 5% of the st.dev. on the whole training set
  Return a terminal node
Let R be a random subset of features, that contains K elements
For each feature x in R
  For all possible split cut-points c
    Calculate the SDR of the split $s[x < c]$ using Eq. (1)
Let s be the split with maximum SDR
Split the set D into subsets D_l and D_r based on split s
Let T_l=**Grow_tree**(D_l), T_r=**Grow_tree**(D_r)
Return a tree with a root node with split s and subtrees T_l and T_r

Table 3
Pseudocode for learning a fuzzified model tree (cont.).

Prune_tree(T)
Inputs: a subtree T
Outputs: pruned subtree rooted at the same node as T

If the root node of T is not a terminal node
  Prune_tree(T → left_subtree)
  Prune_tree(T → right_subtree)
  Learn a linear model for the root node of T using linear regression
  Let err_{LM} be the error of the linear model
  If **Subtree_error**(T) > err_{LM} then
    Convert the root node of T to a terminal node

Subtree_error(T)
Inputs: a subtree T
Outputs: a numeric value – the error of the subtree model

Let $T_l = T \rightarrow$ left_subtree, $T_r = T \rightarrow$ right_subtree
Let $D_0 = T \rightarrow$ examples, $D_l = T_l \rightarrow$ examples, $D_r = T_r \rightarrow$ examples
If the root node of T is not a terminal node
  Return ($|D_l| *$ **Subtree_error**(T_l) + $|D_r| *$ **Subtree_error**(T_r))/ $|D_0|$
Otherwise
    Return the error of the linear model of the root node of T

series Fuzzify_tree (T)
Inputs: a subtree T
Outputs: fuzzified subtree rooted at the same node as T

If the root node of T is not a terminal node
  Let the split at the root node of T be: $s[x_j < c]$
  Let $D = T \rightarrow$ examples
  Let $[x_j^{min}, x_j^{max}]$ be the range of the data points of D in dimension j
  Using Eq. (2) calculate α, s.t. the split width is equal to $p_{overlap}|x_j^{max} - x_j^{min}|$
  Create a fuzzy split in the root node of T using Eq. (2): $\mu(x_j, c, \alpha)$
  Fuzzify_tree($T \rightarrow$ left_subtree)
  Fuzzify_tree($T \rightarrow$ right_subtree)

The **tree growing phase** is a recursive procedure which creates the initial tree structure. It determines whether to split the set of data, and if splitting is to be performed, also determines the split parameters. The optimal split is chosen from a set of candidate splits created in a randomized fashion as described below.

First, a random subset of all features is created, as in random forests [18]. Then, for each candidate attribute in this subset, an optimal cut-point is selected using the standard deviation reduction (SDR) score. Note that the size of the feature subset (K) is a function of the number of available features and this function is a parameter of the algorithm. When the size of the random subset of features is equal to the total number of available features, we eliminate the randomness in the split attribute selection and obtain bagging as a special case.

The candidate splits are evaluated using the SDR, i.e., standard deviation reduction [28] score

$$SDR = \sigma_D^2 - \frac{|D_l|}{|D|}\sigma_{D_l}^2 - \frac{|D_r|}{|D|}\sigma_{D_r}^2 \quad (1)$$

where D is the set of identification points falling in the current node, D_l and D_r are the left and right subsets of the identification data set created by the split, and σ_D^2 is the standard deviation of the target attribute in the set D. The split that maximizes this score is chosen for the current tree node. The data points are divided into two subsets, based on the chosen split, and the procedure is recursively applied on each subset.

The growing phase stops when either of the two stopping (pre-pruning) criteria are met. The first criterion is low variance of the target variable in a partition, as compared to the overall variance of that variable. The second is small number of identification points in a partition, i.e., less than the value of the algorithm parameter N, which could prevent reliable estimation of linear model coefficients.

The **tree pruning phase** employs a bottom-up post-pruning procedure that reduces the size of the tree. Overly large trees, built in the first phase, are prone to overfitting. Tree pruning reduces overfitting by reducing the tree size. The procedure starts by estimating linear models in all nodes of the tree, using linear

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

regression. The linear model in a node is estimated using only features found in the tests in the subtree rooted at that node. The linear model estimation procedure also includes an attribute removal part: attributes are dropped from a linear model when their effect is small, i.e., expected to increase the estimated error.

After estimating linear models in all tree nodes, the pruning procedure evaluates the prediction error of the linear model at each inner node of the tree with the prediction error of the subtree rooted at that node. The prediction error of the linear model learned for a tree node and the prediction error of the subtree below the node are calculated using only data points corresponding to the sub-partition that the node defines. The decision to replace the subtree with a terminal node, i.e., to prune the subtree, is made if the estimated (squared) error of the subtree is larger than or equal to the (squared) error of the linear model.

In the pruning phase, the decision to prune is based on evaluation of the prediction error instead of the simulation error (for dynamic system identification). The reason is that the missing data in the bootstrap sample prohibit the calculation of the simulation error. Also, note that as compared to the original implementation of the M5′ algorithm in WEKA [29], we modify two aspects of the algorithm. The first modification concerns the smoothing[1] procedure of M5′, which in our experience produces low performing models, and is turned off. The second modification concerns the building of linear models. We stop building a linear model, when a matrix singularity (within the computer's numerical precision) is detected in the LU matrix decomposition procedure of linear regression. The tree node is converted into a terminal node in this case.

2.2.1. Tree fuzzification

To smooth out the discontinuities that each split introduces in the model tree, we implement a split fuzzification procedure. A crisp split of the form $s[x_j < c]$, where x_j is the j-th feature, and c is a cut-point, is transformed to a fuzzy split by using a sigmoidal membership function $\mu(x_j, c, \alpha)$ with a fuzzy parameter α (the inverse split width):

$$\mu(x_j, c, \alpha) = \frac{1}{1 + \exp(-\alpha(x_j - c))} \tag{2}$$

The choice between sigmoidal, Gaussian or triangular membership functions [1] does not have an effect on the identification performance in the case when the function parameters are not optimized globally. So, for the sigmoidal membership function of Eq. (2), the value of α is calculated such that the overlap between the two subpartitions is equal to a predetermined percentage $p_{overlap}$ of the size of the partition in the dimension of the split attribute. Larger values for $p_{overlap}$ mean smoother transitions between the local models. The optimal value of $p_{overlap}$ is determined using (internal) cross-validation.

The prediction of a fuzzy tree with one fuzzy split $\mu(x_j, c, \alpha)$ and two local models f_{LM1} and f_{LM2} is calculated by using the following formula:

$$\hat{f}(\mathbf{x}) = \mu(x_j, c, \alpha)\hat{f}_{LM1}(\mathbf{x}) + (1 - \mu(x_j, c, \alpha))\hat{f}_{LM2}(\mathbf{x}) \tag{3}$$

The smoothing of the model tree response using fuzzification is performed after the crisp tree is built. An alternative to the efficient divide-and-conquer approach would be to take the fuzzification into account when estimating the linear models. However, this would decrease the efficiency of the M5′ model tree learning algorithm, as the number of data points that have to be considered when estimating a linear model is equal to the total number of identification points.

[1] Note that the smoothing procedure in M5′ is not related to the smoothing by fuzzyfication that we propose and describe below.

The utilization of linear models in the tree, instead of constants, solves the extrapolation problem, outlined in Section 1.4.3. The predictions of the model, when faced with inputs falling outside the range of the identification data, are no longer limited to constant values: they are linear functions of the input values. Also, by using the tree fuzzification phase, the problem of discontinuities in the model predictions is solved. The smooth switching between local models helps to create smooth simulation output of the model.

2.3. Ensemble selection

After the ensemble is built, it is optimized by using a greedy ensemble selection procedure. Trees that do not contribute to the accuracy of the ensemble are removed from the ensemble. A tree's contribution is evaluated by considering the output error of the reduced ensemble without the tree and comparing its performance to the current ensemble. By evaluating the output error of the ensemble on the identification data (instead of the prediction error), we aim to produce a more successful model of the dynamic system. In the next paragraph we describe the ensemble selection procedure in more detail.

The selection procedure operates in a greedy fashion, reducing the ensemble size by one tree in each step, as shown in the lower part of Table 1. It stops when no improvement can be made to the performance of the ensemble. For dynamic systems, simulation on the identification data is performed and the evaluation of the performance of the ensemble is carried out by using the output error function.

After the ensemble selection procedure, assume that the resulting ensemble has M trees: $E = \{f_1, f_2, \ldots f_M\}$. The prediction of the ensemble is a uniformly weighted average of the model tree predictions:

$$\bar{f}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\hat{f}_i(\mathbf{x}) \tag{4}$$

2.4. Computational complexity

The computational complexity of the top-down growing procedure for a single tree is $O(P \cdot D \cdot logD)$, where D is the number od data points (examples) and P is the number of descriptive variables. The complexity of the bottom-up pruning procedure is $O(D \cdot P^2)$. This term accounts for the learning of the local linear models in all nodes of the tree. Note that the least-squares estimation of a linear model in a tree node is performed by using only variables found in the tests in the subtree rooted at that node. Finally, the complexity of the ensemble selection procedure depends on the square of the number of trees in the ensemble M (which is set to a constant value).

2.5. Illustrative example

To illustrate how ensembles of model trees are learnt, consider the regression problem of fitting the static nonlinear function

$$f(x) = \sin(2\pi x) + 2x \tag{5}$$

using 100 points (x, y), with x uniformly distributed in the interval $[0,1]$. Four fitting scenarios are presented. In the first scenario, one model tree is used to approximate the nonlinear function. The second fits an ensemble model composed of 50 model trees to the nonlinear function. The third and the fourth scenario fit one model tree and an ensemble of 50 model trees, respectively, to a noisy version of the data, where white noise with a standard deviation equal to 20% of the target variable deviation was added.

The results of the fits are shown in Fig. 3. The top parts of the figure present the approximator fit and the true nonlinear function,

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

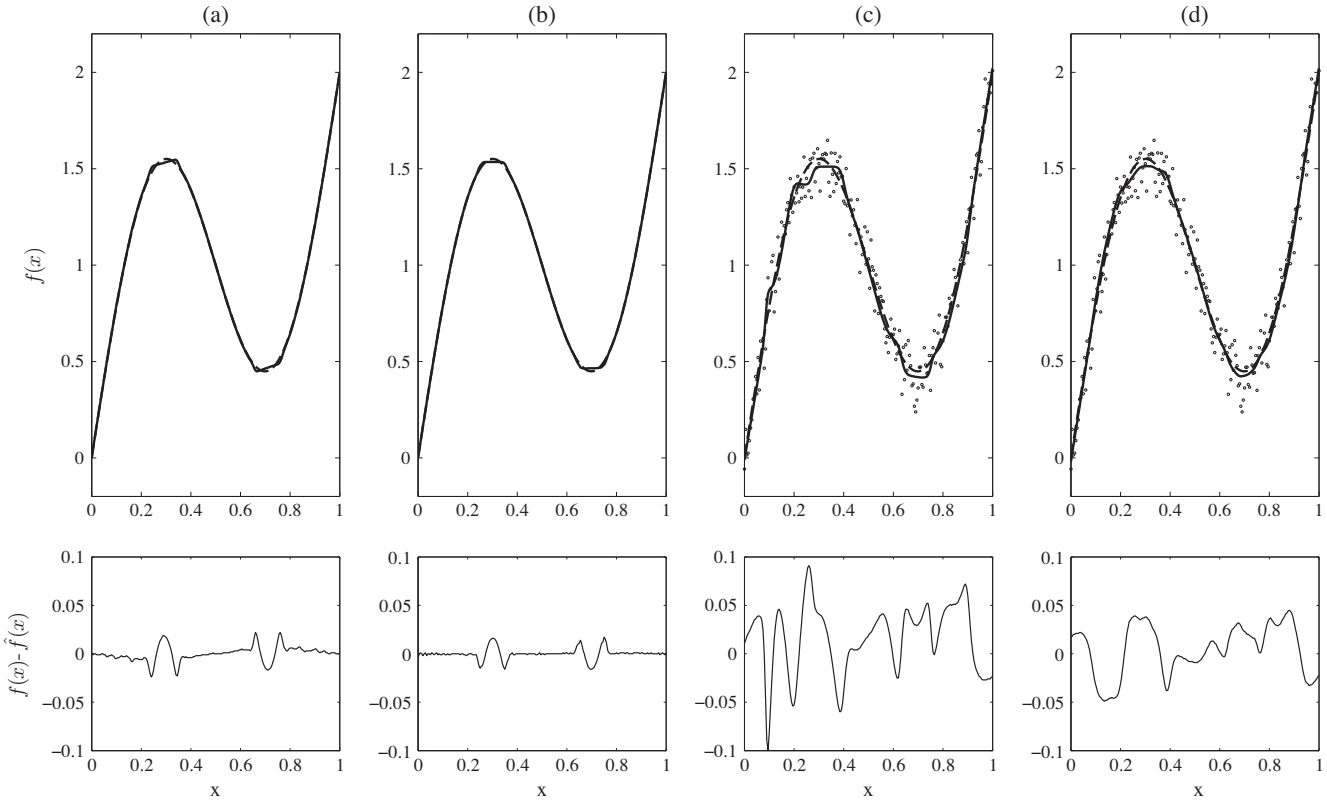

Fig. 3. Fitting the nonlinear function $f(x) = \sin(2\pi x) + 2x$ with (a) a model tree; (b) an ensemble; (c) a model tree from noisy data (20% noise) and (d) an ensemble from noisy data (20% noise). Dashed lines represent the function $f(x)$ we are trying to fit; solid lines represent the model predictions $\hat{f}(x)$; dots represent noisy function values. The lower panels show the approximation error $f(x) - \hat{f}(x)$.

while the bottom parts of the figure present the error of the fit: $f(x) - \hat{f}(x)$. Some properties of the model tree and the ensemble of model tree approximations can be clearly seen from Fig. 3. The single model tree with fuzzy splits, whose approximation is presented in Fig. 3(a), does not produce a good fit around the points $x = 0.3$ and $x = 0.7$, where the derivative of the function changes sign. The parts of the function where the derivative has the same sign (the intervals $(0, 0.3)$, $(0.3, 0.7)$, and $(0.7, 1)$) are approximated better, but not perfectly, as shown by the sloped error lines that can be seen in the bottom part of Fig. 3(a).

The ensemble of 50 model trees, using randomized splits and averaging of the individual tree predictions (Fig. 3(b)), performs better in the problematic regions. The error on the regions where the derivative changes sign is lowered, and the error lines corresponding to the three intervals are much closer to zero and straight, as depicted in bottom part of Fig. 3(b).

On the other hand, the fit to noisy data (Fig. 3(c) and (d)), is not as accurate as in (a) and (b), according to the mean-squared error of the fit: it increases by a factor of approximately 4. However, the ensemble fit improves over the fit of a single tree both for the parts where the derivative changes sign, as well as for the three mentioned intervals. The ensemble fit to the noisy data is mostly smooth and the model is not severely affected by the noisy identification data.

3. Experimental evaluation

3.1. Experimental questions

This section sets out to evaluate the performance of the Model-Tree Ensembles method proposed above and to compare it with some state-of-the-art methods for dynamic system identification. We are especially interested in its performance in the presence of noise. More specifically, the aim of this section is to address the three following experimental questions:

1. Is the MTE method resilient to noise?
2. How does the method compare to selected state-of-the-art methods when identifying dynamic systems from noisy data in controlled conditions?
3. How does the method perform on real-world noisy data?

Two data sets (case studies) are used in the experimental evaluation. The first one is a synthetic data set generated from a dynamic system model, to which noise has been artificially added. The second data set presents measured data for a semi-industrial pilot gas–liquid-separator plant.

In order to answer the first experimental question, the model tree ensembles method is evaluated on the synthetic data set with artificially added white noise. Besides the data without noise, we also consider data with three different noise levels: 5%, 10%, 20%. When introducing noise at level $n\%$, we add to each measured value x_i of the variable x noise, which is sampled from the distribution $N(0, n\sigma_x/100)$ where σ_x is the standard deviation of the variable x.

The second question is addressed by a comparison to representative methods, i.e., selected state-of-the-art approaches for modeling nonlinear dynamic systems. In particular, the method is compared to a multilayer perceptron neural network approach, a neuro-fuzzy approach, and a tree-based fuzzy approach. The third question is tackled by using a measured data set originating from a process engineering system.

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

3.2. Selected methods for comparison

We compare the MTE method with three methods that are well-established in the area of system identification: Neural Networks [4], ANFIS [9] and Lolimot [1]. From the parameter identification perspective, two of the methods utilize global optimization of the parameters: Artificial Neural Networks and ANFIS. On the other hand, Lolimot and the proposed Model-Tree Ensembles use local optimization of the local model parameters.

The models that the compared methods learn are of two types: a feed-forward neural-network model and a Takagi–Sugeno fuzzy model. Two different methods that build a Takagi–Sugeno model are compared, since the learning strategies are different. ANFIS uses a separate structure identification step and global optimization of the model parameters, while Lolimot uses an integrated structure identification and local parameter estimation approach. A brief overview of the properties of the methods is given in the following paragraphs.

We use feedforward Artificial Neural Networks (NN) [4], more specifically a multilayer perceptron with one hidden layer of neurons, trained by using a backpropagation procedure. The number of neurons in the hidden layer is the only parameter whose value needs selection. We use the Neural Network Toolbox implementation in Matlab. The network training is performed using Levenberg–Marquardt optimization.

The Adaptive network based fuzzy inference system (ANFIS) [9] is a hybrid neural-network approach, which builds a Takagi–Sugeno fuzzy model. ANFIS solves the parameter estimation problem by using a hybrid learning rule that combines the backpropagation gradient descent and the least-squares method. The structure identification task – determining the number of fuzzy rules and initial positioning of the fuzzy rule centers is handled by using different methods: grid partitioning of the instance space, fuzzy clustering, or a tree-based approach [30].

ANFIS suffers from the curse of dimensionality as the number of input dimension gets larger. In this work, we use the Matlab implementation of the ANFIS method (available in the Fuzzy Logic Toolbox). For the structure identification problem, we utilize the fuzzy *c*-means clustering method. We do not use the clustering method's automatic procedure of determining the number of clusters, since (in our experience), it produces sub-optimal models. Instead, we utilize a version of the clustering algorithm with a single tunable parameter: the number of fuzzy clusters.

The Local Linear Model Trees (Lolimot) [1] method is a multi-model approach which builds a fuzzy model tree. It solves the structure identification and parameter estimation problems in an integrated, iterative procedure. In each iteration, the method adds one local model to the tree structure and calculates the parameters of the model using local parameter estimation. It has been successfully used for identification of dynamic systems [1].

More specifically, Lolimot builds linear model trees with smooth transitions between the affine local models in the tree leafs. It fits a Gaussian basis function in the center of each leaf node, where the standard deviation vector is calculated based on the size of the partition in each dimension. The Lolimot algorithm does not perform pruning, but instead, the building of the tree is halted when the tree size increases beyond a predefined number of leaf nodes.

Using Lolimot for dynamic system identification requires determining the optimal lag for the premise part (variables included in split tests) and the optimal lag for the consequent part (variables used in linear models in the tree leafs). The best performing Lolimot tree is obtained by a trial-and-error procedure: running the method with different lag value combinations for the premise and consequent part and evaluating the obtained fit. Additional details about the method are given by Nelles [1].

The parameters of the MTE methodology are set and used as follows. The number of trees in the ensemble (M) is set to 50. Our preliminary experiments with different numbers of trees showed no improvement in accuracy by using larger values. Also, the value of the minimal number of identification points in a partition (N), is not crucial for the overall accuracy of the model being learned. From our experience, setting this parameter to its minimal value of 4, results in optimal model performance. On the other hand, the size of the random subset of feature attributes (K), used in the tree growing phase, has an effect on the overall performance of the method. The value of this parameter determines whether the ensemble approach is bagging or Random Forests.

After the optimal lag and optimal value for the parameter K have been found, the overlap width parameter $p_{overlap}$ is optimized. The values considered for this parameter range from 5% to 50%. After the parameter optimization step, the training of the ensembles is carried out, by learning 50 trees. Finally, the ensemble selection procedure is employed to reduce the number of trees in the ensemble. Its usefulness to the model-tree ensemble methodology is empirically demonstrated in Appendix A of the paper.

Our choice to use randomness in the split attribute selection, was due to the experience that it can produce models with more diverse predictions. A comparison of the MTE method with or without random split attribute selection (Forests of Model Trees (FMT) vs. bagging of Model Trees (BMT) respectively) is also shown in each of the tables in the Experimental results section (Section 3.6).

3.3. Experimental design

For each of the methods considered, the experimental evaluation required determining the optimal lag (order) of the variables,

Table 4
Method parameters and lag values considered and selected for the experimental evaluation.

	Parameter	pH neutralization		Gas–liquid separator	
		Values tried	Best value	Values tried	Best value
Neural Networks	Number of hidden neurons	1,2,...,10	6	1,2...,10	6
	lag	1,2,3,4	2	1,2,3	1
ANFIS	Number of fuzzy clusters (rules)	2,3...,10	3	2,3...,10	2
	Lag	1,2,3,4	2	1,2,3	1
Lolimot	Maximum num. local models	5,10,20,30,50	50	5,10,20,30,50	20
	Premise space lag	1,2,3,4	2	1,2,3	2
	Consequent space lag	1,2,3,4	2	1,2,3	2
Forest of Model Trees	Size of random subset of feature att. (K)	1,2,3,4	1	1,2,3	1
	$p_{overlap}$	5%,10%,15%,...,50%	25%	5%,10%,15%,...,50%	50%
	lag	1,2,3,4	2	1,2,3	1
Bagging of Model Trees	$p_{overlap}$	5%,10%,15%,...,50%	30%	5%,10%,15%,...,50%	40%
	lag	1,2,3,4	2	1,2,3	1

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

as well as optimal values for the method's parameters. This task involved evaluating the performance of the methods with different lag and parameter combinations on the identification data. The determination of the optimal lag and parameter combination was performed using 5-fold cross-validation, which evaluated the prediction error of the ensemble models. The cross-validation procedure included creating 5 train/test splits of the identification data and learning an ensemble model for each of the folds, as suggested by van der Laan et al. [31]. The lag and parameter combination that produced optimal (mean-squared) prediction error for the cross validation procedure was selected. The values of the lag and parameter combinations tested are shown in Table 4. After the optimal values for the lag and the method's parameters were determined, they were used to evaluate the performance of the method on the validation data.

For the identification using the synthetic input–output data, the methods were trained by using both the noiseless data and data with added white noise with mean zero and standard deviation of 5%, 10% and 20% of the standard deviation of the system variable. The noise was added only to the system variables in the identification data, i.e., to all corresponding lagged variables in the set of features as well as in the target variable of the identification set. No noise was added to the validation data.

The error measure reported is root relative mean-squared error (RRMSE), also known as normalized root mean squared error (NRMSE):

$$RRMSE = \frac{\sqrt{\sum (y_i - \hat{y}_i)^2}}{\sqrt{\sum (y_i - \bar{y})^2}} \tag{6}$$

To obtain more reliable estimates of the performance, each experiment was repeated 20 times,[2] using different random seeds for the randomization included in the methods. We report the mean and the standard deviation of the simulation (output), measured as RRMSE across the 20 runs.

It is worth noting here that Lolimot does not include any randomization and is a purely deterministic method. On the other hand, Neural networks, ANFIS and MTE use randomization. Neural networks use randomization to set the initial values of the neuron parameters. The structure determination of ANFIS, carried out by using *c*-means fuzzy clustering, uses random initial standard deviations of the fuzzy membership Gaussian functions. The MTE method uses randomization in the bootstrap sampling and split attribute selection.

3.4. Case study: pH neutralization

The control of alkalinity (pH) is common in biotechnological industries and chemical processes. The topic of this case study is the identification of the pH neutralization process, which exhibits severe nonlinear behavior [32,12]. What follows is a short description of the process itself, the equations governing the process and the synthetic data generated from this model of the pH neutralization process, which are used for the task of system identification.

The pH neutralization system, described in detail by Henson and Seborg [32], consists of an acid stream Q_1, a buffer stream Q_2, and a base stream Q_3 that are mixed in a tank T_1. Before mixing takes place, the acid stream Q_1 enters another tank T_2. The measured variable is the effluent pH, which is controlled by manipulating the flow rate of the base stream Q_3. The flow rates of the acid and buffer streams are taken to be constant. A schematic diagram of the system is shown in Fig. 4.

A model of this dynamic system is derived by Henson and Seborg [32], which contains the following state, input and output variables:

$$\mathbf{x} = [W_{a4}\ W_{b4}\ h_1]^T, \quad u = Q_3,\ y = pH \tag{7}$$

where W_{a4} and W_{b4} are the effluent reaction invariants and h_1 is the liquid level of tank T_1. Also, it is assumed for the state variable h_1 that a controller has already been designed to keep its level at a nominal value of $h_1' = 14$ cm by manipulating the exit flow rate Q_4. The state-space model obtained has the form:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u \tag{8}$$

$$c(\mathbf{x}, y) = 0 \tag{9}$$

where $f(\mathbf{x})$ and $g(\mathbf{x})$ are nonlinear functions of the state vector $\mathbf{x}$, while $c(\mathbf{x}, y)$ is a nonlinear function which is a part of the implicit output Eq. (9). The sampling time selected was 25 s, same as in the work of Kocijan and Petelin [12]. The input variable u changed its value every 500s, each time being set to a value generated by using a uniform random distribution. The input–output data used, shown in Fig. 5, consist of 320 data points for identification and 320 data points for validation.

The experimental procedure included determining the optimal method parameters, as well as the optimal order (lag) of the variables, whose value was in the range from 1 to 4. The parameter values considered are shown in Table 4. For example, for the Neural Networks method, we tried 10 different values for the number of hidden neurons parameter and 4 values for the lag, which gave a total of 40 combinations. Each of these was evaluated by using (internal) cross-validation on the identification data. The best performing combination of parameter values, also reported in Table 4, was selected.

To evaluate the resilience of the identification methods to noise, four different variants of the data were considered, where white noise was added to the output (system) variable only in the identification data. The standard deviations of the added white noise were 0%, 5%, 10% and 20% of the output variable's standard deviation. The bottom left panel in Fig. 5 shows the identification data with 20% noise. The validation data were not disturbed by white noise.

3.5. Case study: gas–liquid separator

The system being modeled in this case study is a unit for the separation of gas from liquid [33]. The separator unit is a semi-industrial process plant which belongs to a larger pilot plant. A scheme of the structure of the plant is given in Fig. 6.

The purpose of the modeled system is to capture flue gases under low pressure from the effluent channels using a water flow, cool the gases down, and supply them with increased pressure to other parts of the pilot plant. The flue gases coming from the effluent channels are absorbed by the water flow into the water circulation pipe through the injector I_1. The flow of water is generated by the water ring pump (P_1), whose speed is kept constant. The pump feeds the gas–water mixture into the tank T_1, where the gas is separated from the water. The accumulated gases in the tank form a kind of a pressurized gas 'cushion'. Due to this pressure, the flue gases are blown out from the tank into the neutralization unit, while on the other hand, the water is forced by the 'cushion' to circulate back to the reservoir. The water quantity in the circuit is constant.

[2] In machine learning, performance is standardly evaluated by cross-validation. A more reliable measure of the performance can be obtained by repeating the cross-validation procedure 10 times and reporting the mean and variance of the error measure. We selected a larger number of runs, to be sure that if a model built by using the current parameter combination had a possibility of divergent simulation, it would be detected.

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

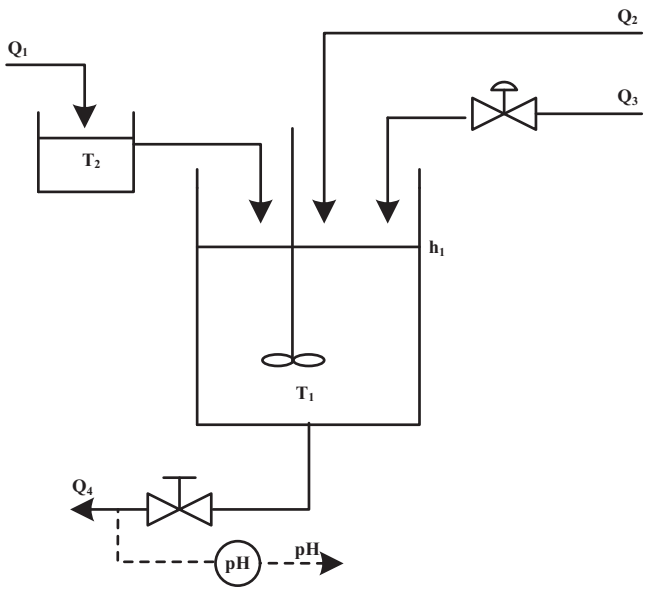

Fig. 4. A schematic diagram of the pH neutralization system.

The first-principles model of the system is a set of differential equations. The variable p_1 is the relative air pressure in the tank T_1, the variable h_1 is the liquid level of the tank T_1, while u_1 and u_2 are command signals for the valves V_1 and V_2 respectively. The differential equation for the air pressure variable p_1 has the form:

$$\frac{dp_1}{d} = f_a(h_1)[\alpha_1 + \alpha_2 p_1 + \alpha_3 p_1^2 + f_b(u_1)\sqrt{p_1} + f_c(u_2)\sqrt{(p_1 + \alpha_4 + \alpha_5 h_1)}] \quad (10)$$

where the values α_i are constants, while $f_a(h_1), f_b(u_1)$ and $f_c(u_2)$ are functions of the corresponding variables. $f_a(h_1)$ is a rational function of h_1, while $f_b(u_1)$ and $f_c(u_2)$ are the valve characteristics

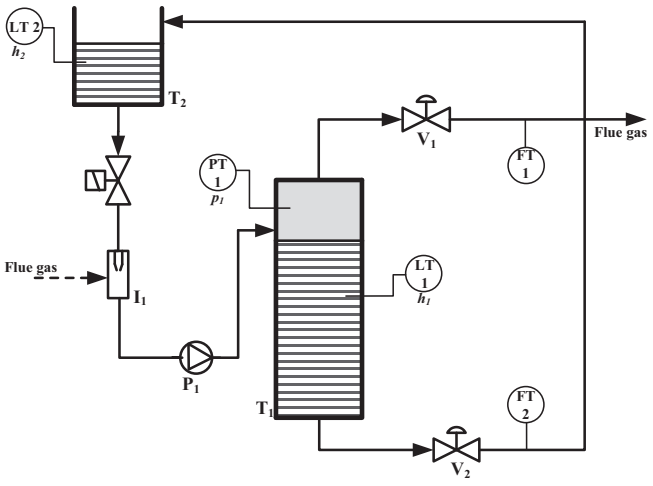

Fig. 6. A schematic diagram of the semi-industrial process plant.

(exponential functions of u_1 and u_2 respectively). The details of the model are given by Kocijan and Likar [33].

The aim of the system identification in this case study is to build a model for predicting the value of the pressure variable p_1, from lagged values of itself, as well as lagged values of the input variables. The sampling time selected was 20 s, same as in the work of Kocijan and Grancharova [34]. The identification and the validation data both consist of 733 input–output data points, shown in Fig. 7, and are disturbed by intrinsic measurement noise. The optimal lag was chosen by considering lag values from 1 to 3. For illustration, for a lag of 2, the system identification problem is transformed to the following regression problem:

$$p_1(k) = f(p_1(k-1), p_1(k-2), u_1(k-1), u_1(k-2), u_2(k-1), u_2(k-2), h_1(k-1), h_1(k-2)) \quad (11)$$

The experimental procedure was identical to the one for the previous case study (Section 3.4). Again, all possible parameter and lag

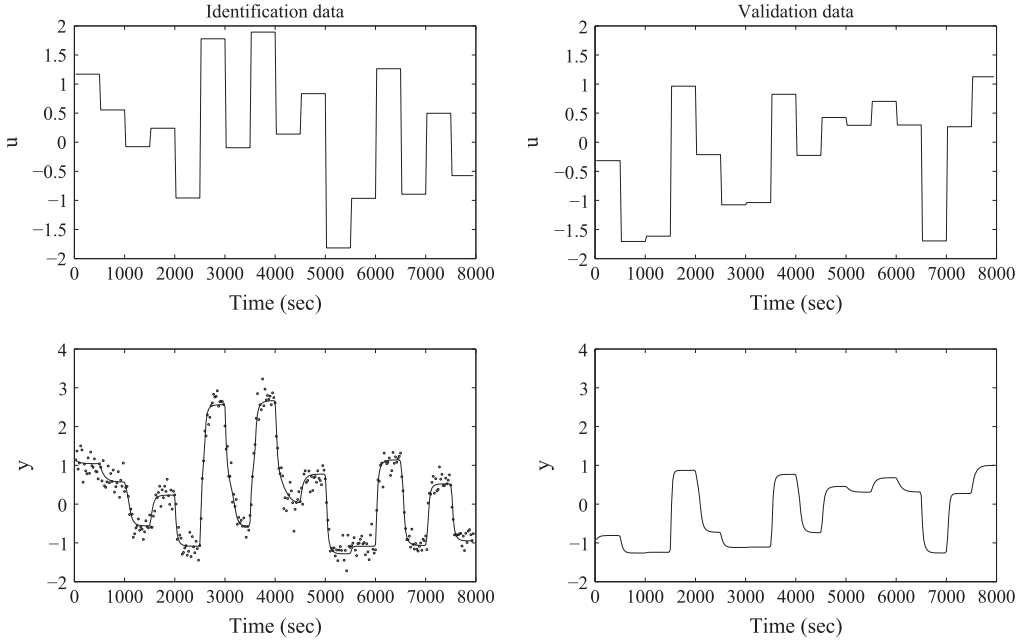

Fig. 5. Input–output data for identification of the pH system; detrended identification data (left) and detrended validation data (right). The bottom left panel shows both the non-noisy data (solid line) and the data with 20% noise (dots).

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

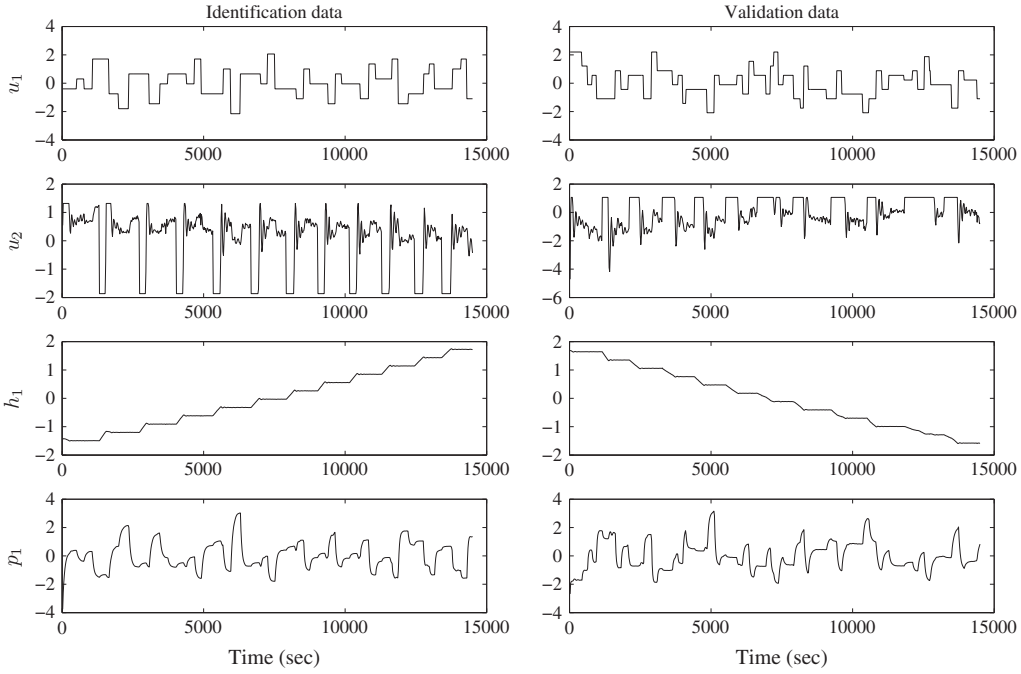

Fig. 7. Input–output data for identification of the gas–liquid separator system. Detrended identification data are shown in the left and detrended validation data in the right four panels.

Table 5
Performance of the methods on the pH data set in terms of simulation/output error (RRMSE) on the validation data.

	Noise 0%	Noise 5%	Noise 10%	Noise 20%
Neural networks	0.1498 ± 0.0712	0.1617 ± 0.0512	0.1617 ± 0.0478	0.1996 ± 0.0630
ANFIS	**0.0766 ± 0.0000**	**0.0843 ± 0.0000**	**0.1123 ± 0.0000**	0.1270 ± 0.0001
Lolimot	0.0834 ± 0.0000	0.1175 ± 0.0000	0.1203 ± 0.0000	**0.1180 ± 0.0000**
Forest of Model Trees	0.1085 ± 0.0111	0.1076 ± 0.0086	0.1141 ± 0.0084	0.1229 ± 0.0142
Bagging of Model Trees	0.1262 ± 0.0173	0.1109 ± 0.0088	0.1109 ± 0.0089	0.1353 ± 0.0154

Table 6
Performance of the system identification methods on the gas–liquid separator data set in terms of output error (RRMSE).

	Gas–liquid separator
Neural Networks	0.2106 ± 0.0402
ANFIS	**0.1622 ± 0.0014**
Lolimot	0.2368 ± 0.0000
Forest of Model Trees	0.1711 ± 0.0119
Bagging of Model Trees	0.2081 ± 0.0147

combinations were tested and the best performing combination, based on the cross-validation evaluation using the identification data, was selected. The selected parameters for each method are reported in Table 4, while the performance on the validation data is reported in Tables 5 and 6.

3.6. Experimental results

The results of the evaluation of the five methods, i.e., Neural Networks, ANFIS, Lolimot and the two variants of MTE (Forests of Model Trees and Bagging of Model Trees) on the pH data are shown in Table 5 and Fig. 8. The y axis of the figure depicts the output error of the methods, in terms of RRMSE, aggregated over the 20 runs. The curves display the mean error, while the error bars represent the standard deviations of the output error in terms of RRMSE.

It can be observed from Fig. 8 that, apart from Neural Networks, there is no large difference between the compared methods. Neural Networks have the worst performance among all considered methods, as well as the largest deviation of the error measure. The performance of ANFIS and Lolimot is approximately the same, except for the 5% noise case, where ANFIS has lower error. FMT, on the other hand, performs worse than ANFIS for the 0% and 5% noise cases, while their performance is approximately the same for the 10% and 20% noise cases. It can be also observed that FMT is very resilient to noise, as its error increases only slightly with increased levels of noise. The variability of the error measures is almost zero for ANFIS and exactly zero for Lolimot, which is a purely deterministic method, while MTE shows small variations in the value of the error measure. However, this small variation does not affect the reliability of the ensemble models produced.

Below we give an illustration of the fit of the MTE models to the validation data. Fig. 9 shows the model predictions (simulations) for all noise cases (top), as well as the error of the fit $\hat{y} - y$ (bottom). It can be seen that the model learned in the presence of 20% noise in the identification data performs the worst. This is clearly visible for the time interval [0 s, 500 s], where the predicted and true values differ the most, as well as around the time points 2500 s, 4500 s, and 6500 s. However, it can be observed that the model simulation for the lower noise levels is smooth and closely approximates the output variable of the validation data. The only

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

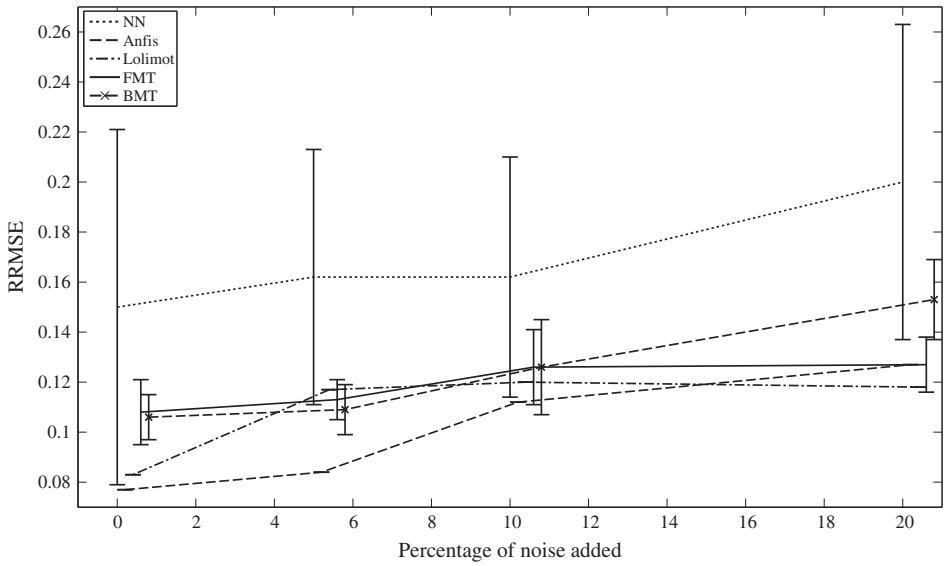

Fig. 8. Average output error (RRMSE) of the different methods on the pH validation data across 20 runs (error bars represent standard deviation). The models are learned using noisy identification data.

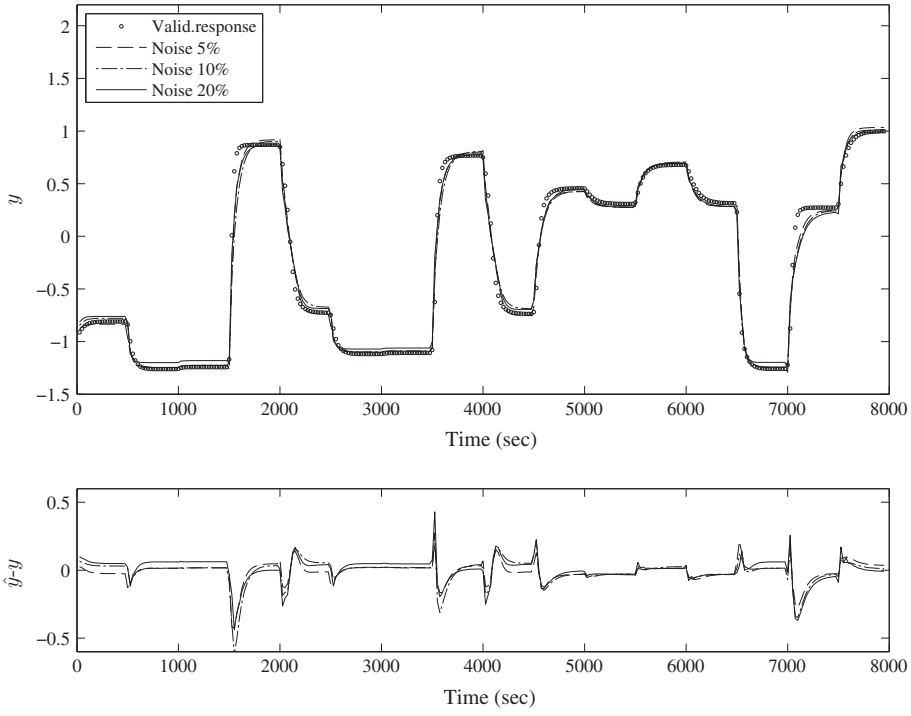

Fig. 9. Simulations (predictions) of the MTE models for the pH data at different noise levels: validation data and simulations of models learned from noisy data are shown at the top, simulation errors at the bottom panel.

exceptions are the out-of-equilibria points, around 1500 s, 3500 s, and 7000 s, where the model is not as accurate. This is due to the small number of identification data points in the out-of-equilibria regions.

The results of the evaluation of the different methods on the real-world gas–liquid-separator data are shown in Table 6. The table displays the mean and standard deviation of the simulation error (RRMSE), across the 20 runs. The globally optimized model of ANFIS, with only two rules, has the best simulation performance. The locally optimized MTE follows, showing only slightly worse performance.

On this case study, Neural networks and Lolimot show the worst performance. The variability of the error measure is close to zero for ANFIS, small for MTE, and quite large for the Neural Network models. The small variability of the simulation error on real-world data shows that MTEs are able to produce models with reliable simulation performance.

Fig. 10 illustrates the predictions (simulations) of the model that the FMT variant of the MTE method produces and their fit to the validation data. The model consists of 5 trees, while the total number of local linear models, i.e., total number of tree leaves is 36. The fit closely approximates the validation data, while the

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

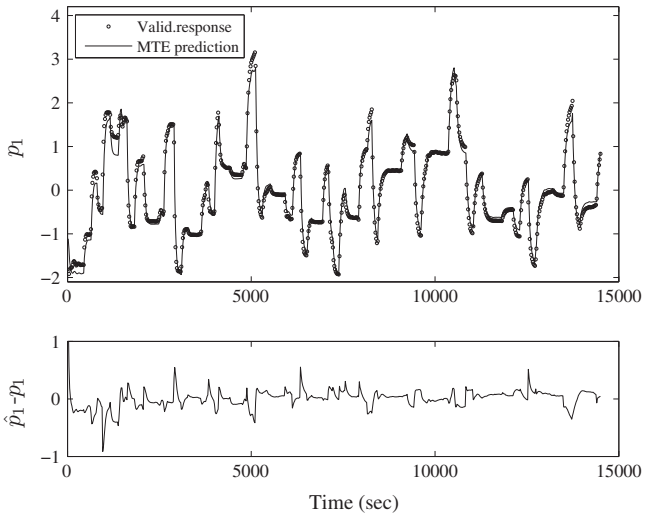

Fig. 10. Simulations (predictions) of the MTE model for the gas–liquid separator data: validation data and simulation of the model are shown at the top, simulation errors at the bottom panel.

predictions for the out-of-equilibria points, for example around time points 4000 or 5000, are not as accurate. The error of the MTE ensemble is shown in the bottom part of the figure.

The two variants of the MTE method – Forests of Model Trees and their special case of Bagging Model Trees show different performance. Tables 5 and 6 show that the performance of BMT is worse or equal to the performance of FMT. This shows that the attribute randomization parameter K, which determines whether the variant is FMT or BMT, needs selection.

The complexity of the models is shown in Table 4. It can be observed that ANFIS creates global models with very few local models: 3 for the pH data set and 2 for the gas–liquid separator data set. This is because the global optimization of the parameters produces accurate predictions using a small number of widely applicable local models. The neural networks method, which also optimizes the model parameters globally, requires quite a small model complexity – only 6 neurons. On the other hand, the methods which utilize local estimation of the local model parameters – Lolimot and MTE (the Forests of Model Trees variant), build global models containing more local models (Table 7).

Table 7
Information about the complexity of models learned by the five different methods.

		pH neutralization	Gas–liquid separator
Neural networks	Number of hidden neurons	6	6
	Lag	2	1
ANFIS	Number of fuzzy clusters (rules)	3	2
	Lag	2	1
Lolimot	Number of local models	45	20
	Premise space lag	2	2
	Consequent space lag	2	2
Forest of model trees	Number of trees after reduction (A)	6	5
	Average number of local models per tree (B)	7.5	7.2
	Total number of local models (A · B)	45	36
	Lag	2	1
Bagging of model trees	Number of trees after reduction (A)	6	9
	Average number of local models per tree (B)	11	15
	Total number of local models (A · B)	66	135
	Lag	2	1

4. Conclusions, discussion, and further work

4.1. Summary

We address the task of discrete-time modeling of nonlinear dynamic systems using measured data, which is typically converted into a regression problem. We investigate the use of tree ensembles for regression, a very successful predictive modeling approach, for this task. We consider existing tree ensemble approaches to regression, such as bagging of model trees, and propose the use of a novel approach of learning model tree ensembles tailored to the task of modeling dynamic systems. The latter learns random forests of fuzzified model trees and performs ensemble selection based on the output error measure.

We evaluate the performance of the tree ensemble methods and three state-of-the-art methods for system identification typically used in control engineering. We consider the predictive performance of the learned models and in particular their resilience to noise. For this purpose, we use one synthetic task without noise, one synthetic task with different levels of artificially introduced noise, and one real task of modeling nonlinear dynamic systems, all coming from the area of control engineering.

4.2. Comparison to regression tree ensemble methods

The task of modeling dynamic systems requires a modeling procedure which produces a smooth fit and a reasonable extrapolation behavior. Regression trees and ensembles thereof do not meet these two criteria [27] and are thus not appropriate for the task at hand: Both fail on the extrapolation front, with ensembles producing smoother fits than individual regression trees.

Instead, we propose the use of ensembles of model trees with fuzzified splits. Here already the base models produce much smoother fits and have better extrapolation capabilities. The ensembles (random forests) of such trees produce better behavior still, which is further improved by performing ensemble selection (guided by the output error as a measure).

4.3. Noise resilience

The proposed Model-Tree Ensembles method demonstrates useful noise-tolerance properties, even in quite noisy scenarios. The experimental evaluation confirms the beneficial noise resilience properties of the method by adding a large amount of noise to the identification data for the pH problem, which was as high as 20%. The output error of the model increases only slightly as the noise level increases.

4.4. Comparison to system identification methods

The experimental evaluation revealed that the number of rules (local models) is smaller when the method at hand utilizes global optimization for its parameters, as compared to the methods which only utilize local optimization, as the proposed MTE method does. However, the globally optimal approaches use nonlinear optimization procedure, which is more computationally complex

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

Table 8
Output error results of the ensemble methods with and without ensemble selection.

	FMT with ensemble selection	FMT without ensemble selection	BMT with ensemble selection	BMT without ensemble selection
pH noise 0%	**0.1085 ± 0.0111**	0.1493 ± 0.0128	**0.1262 ± 0.0173**	0.1747 ± 0.0138
pH noise 5%	**0.1076 ± 0.0086**	0.1360 ± 0.0056	**0.1109 ± 0.0088**	0.1458 ± 0.0069
pH noise 10%	**0.1141 ± 0.0084**	0.1493 ± 0.0079	**0.1109 ± 0.0089**	0.1556 ± 0.0057
pH noise 20%	**0.1229 ± 0.0142**	0.1754 ± 0.0119	**0.1353 ± 0.0154**	0.2019 ± 0.0125
Gas–liquid separator	**0.1711 ± 0.0119**	0.1786 ± 0.0058	0.2081 ± 0.0147	**0.1957 ± 0.0105**

than the linear optimization used in the locally optimal approaches.

Our proposed Model-Tree Ensembles have comparable predictive performance to the models learned by state-of-the-art methods for system identification typically used in control engineering. Having in mind that the Model-Tree Ensembles is a method that includes randomization, the empirical analysis showed that the variance of the error measure is low and the method demonstrates reliable performance. The computing time of our approach is comparable to that of the state-of-the-art methods on the small problems considered [27], with the potential to scale much better to large modeling problems.

The MTE method has the potential to scale well, as it can handle data sets that are large along both dimensions, i.e., the number of features and the number of examples (identification points). Decision trees as a base method are among the most efficient machine learning methods in terms of handling a large number of examples. Random forests, which take random subsets of the set of features at each step of building the tree, can handle very large numbers of features. This translates to handling either a large number of system/input variables, a large time lag, or both.

4.5. Further work

We intend to pursue several directions for further work, including the further development of the model tree ensemble methodology, a more extensive evaluation and practical applications thereof. In terms of the latter, we plan to apply our methodology to additional problems/tasks of modeling dynamic systems, especially to larger problems. This would allow us to investigate and demonstrate the scalability of our approach.

As further work, we would also like to extend our method for the multi-input multi-output (MIMO) case. This would increase the applicability of the methodology and would produce dynamic system models of smaller sizes as compared to learning several single-output models. Related approaches for learning multi-output model trees already exist [14]: These could be extended to include fuzzified splits and be used as base model learners within an ensemble setting similar to the one considered here.

Acknowledgements

We would like to gratefully acknowledge the financial support of the following institutions: The Slovene Human Resources Development and Scholarship Fund, the Slovenian Research Agency (Grants P2-0001 and P2-0103), the European Commission (Grants ICT-2010-266722 and ICT-2011-287713), and the Operation No. OP13.1.1.2.02.0005 financed by the European Regional Development Fund (85%) and the Ministry of Education, Science, and Sport of Slovenia (15%). We would also like to thank Oliver Nelles for providing the source code for the Lolimot method.

Appendix A. Results without ensemble selection

To demonstrate the utility of ensemble selection, we provide a comparison of the FMT and BMT performance with and without ensemble selection, shown in Table 8. The results without ensemble selection present the performance of the FMT and BMT ensembles with 50 model trees each. On the other hand, the results with ensemble selection show the performance of the ensembles after the reduction of the number of trees.

The results in Table 8 show that the ensemble selection stage is beneficial in most cases. The benefit is especially evident in the noisy cases. Apart from the improved performance, the final ensemble models resulting from ensemble selection have lower complexity as compared to the full-size ensembles.

References

[1] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer, 2001.
[2] F. Giri, E.-W. Bai, Block-Oriented Nonlinear System Identification, Lecture Notes in Control and Information Sciences, Springer, 2010.
[3] R. Haber, H. Unbehauen, Structure identification of nonlinear dynamic systems–a survey on input/output approaches, Automatica 26 (1990) 651–677.
[4] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and other Kernel-based Learning Methods, Cambridge University Press, 2000.
[5] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.
[6] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE T. Neural Networ. 1 (1990) 4–27.
[7] K. Ažman, J. Kocijan, Dynamical systems identification using Gaussian process models with incorporated local models, Eng. Appl. Artif. Intel. 24 (2011) 398–408.
[8] R. Murray-Smith, T. Johansen (Eds.), Multiple Model Approaches to Modelling and Control, Taylor & Francis, 1997.
[9] J.-S.R. Jang, C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing–A Computational Approach to Learning and Machine Intelligence, Prentice Hall, 1997.
[10] T.A. Johansen, B.A. Foss, Operating regime based process modeling and identification, Comput. Chem. Eng. 21 (1997) 159–176.
[11] O. Nelles, On training radial basis function networks as series-parallel and parallel models for identification of nonlinear dynamic systems, in: Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, IEEE, 1995, pp. 4609–4614.
[12] J. Kocijan, D. Petelin, Output-error model training for Gaussian process models, in: A. Dobnikar, U. Lotrič, B. Šter (Eds.), Adaptive and Natural Computing Algorithms, Lecture Notes in Computer Science, vol. 6594, Springer, 2011, pp. 312–321.
[13] M.R. Segal, Machine Learning Benchmarks and Random Forest Regression, Technical Report, Center for Bioinformatics and Molecular Biostatistics, University of California, 2004.
[14] A. Appice, S. Džeroski, Stepwise induction of multi-target model trees, in: Proceedings of the 18th European Conference on Machine Learning, Springer, 2007, pp. 502–509.
[15] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: Advances in Neural Information Processing Systems, pp. 231–238.
[16] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[17] Y. Grandvalet, Bagging equalizes influence, Mach. Learn. 55 (2004) 251–270.
[18] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[19] B. Pfahringer, Semi-random model tree ensembles: an effective and scalable regression method, in: D. Wang, M. Reynolds (Eds.), AI 2011: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 7106, Springer, 2011, pp. 231–240.
[20] M.P. Perrone, L.N. Cooper, When Networks Disagree: Ensemble Methods for Hybrid Neural Networks, Technical Report, Institute for Brain and Neural Systems, Brown University, 1993.

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008

[21] T. Aho, B. Ženko, S. Džeroski, T. Elomaa, Multi-target regression with rule ensembles, J. Mach. Learn. Res. 13 (2012) 2367–2407.
[22] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the 21st International Conference on Machine Learning, ACM, 2004, p. 18.
[23] G. Tsoumakas, I. Partalas, I. Vlahavas, An ensemble pruning primer, in: O. Okun, G. Valentini (Eds.), Applications of Supervised and Unsupervised Ensemble Methods, Studies in Computational Intelligence, vol. 245, Springer, 2009, pp. 1–13.
[24] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Taylor & Francis, pp. 119–134.
[25] A. Suarez, J. Lutsko, Globally optimal fuzzy decision trees for classification and regression, IEEE T. Pattern Anal. 21 (1999) 1297–1311.
[26] C. Olaru, L. Wehenkel, A complete fuzzy decision tree technique, Fuzzy Set. Syst. 138 (2003) 221–254.
[27] D. Aleksovski, J. Kocijan, S. Džeroski, Model tree ensembles for modeling dynamic systems, in: J. Fürnkranz, E. Hüllermeier, T. Higuchi (Eds.), Discovery Science, Lecture Notes in Computer Science, vol. 8140, 2013, pp. 17–32.
[28] Y. Wang, I.H. Witten, Induction of model trees for predicting continuous classes, in: M. van Someren, G. Widmer (Eds.), Poster Papers of the 9th European Conference on Machine Learning (ECML 97), Lecture Notes in Computer Science, vol. 1224, Springer, 1997, pp. 128–137.
[29] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. 11 (2009) 10–18.
[30] J.-S.R. Jang, Structure determination in fuzzy modeling: a fuzzy CART approach, in: Proceedings of the Third IEEE Conference on Fuzzy Systems, IEEE, 1994, pp. 480–485.
[31] M.J. van der Laan, S.E. Sinisi, M.L. Petersen, Cross-validated bagged learning, Technical Report 182, Division of Biostatistics, University of California, Berkeley, 2005.
[32] M.A. Henson, D.E. Seborg, Adaptive nonlinear control of a pH neutralization process, IEEE T. Contr. Syst. T. 2 (1994) 169–182.
[33] J. Kocijan, B. Likar, Gas–liquid separator modelling and simulation with Gaussian-process models, Simulat. Model. Pract. Theory 16 (2008) 910–922.
[34] J. Kocijan, A. Grancharova, Gaussian process modelling case study with multiple outputs, C.R. Acad. Bulg. Sci. 63 (2010) 601–607.

Please cite this article in press as: D. Aleksovski et al., Model-Tree Ensembles for noise-tolerant system identification, Adv. Eng. Informat. (2014), http://dx.doi.org/10.1016/j.aei.2014.07.008