Expert Systems With Applications 42 (2015) 8484–8496

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Predicting long-term population dynamics with bagging and boosting of process-based models

Nikola Simidjievski[a,b,*], Ljupčo Todorovski[c], Sašo Džeroski[a,b]

[a] Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, Ljubljana 1000, Slovenia
[b] Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana 1000, Slovenia
[c] Faculty of Administration, University of Ljubljana, Gosarjeva ulica 5, Ljubljana 1000, Slovenia

ARTICLE INFO

Keywords:
Ensembles
Process-based modeling
Bagging
Boosting
Machine learning
Predictive modeling
Population dynamics

ABSTRACT

Process-based modeling is an approach to learning understandable, explanatory models of dynamic systems from domain knowledge and data. Although their utility has been proven on many tasks of modeling dynamic systems in various domains, their ability to accurately predict the future behavior of an observed system is limited. To address this limitation, we propose the use of a standard approach to improving the predictive performance of machine learning methods, i.e., the approach of learning ensemble models. Previous work on ensembles of process-based models has been limited to proof-of-principle experiments with a single ensemble method (bagging) and in the limited perspective of explaining the currently observed system behavior v.s. predicting future system behavior. In this paper, we design a general methodology for adapting ensemble methods to the context of process-based modeling. Using the methodology, we implement the two approaches bagging and boosting of process-based models. We perform an empirical evaluation of the implemented methods on three real-world modeling problems from the domain of population dynamics in aquatic ecosystems. The results of the empirical evaluation show that ensembles of process-based models can lead to long-term predictions of the population dynamics that are more accurate than the ones obtained with a single process-based model.

1. Introduction

Mathematical models are employed to provide an understanding of the laws that govern the behavior of dynamic systems. More specifically, such models are being utilized to recreate or simulate the behavior of dynamic systems under various conditions. This paper addresses the task of automated modeling of dynamic systems from time-series data and domain-specific modeling knowledge. The result of which is a process-based model that both explains the structure of the modeled system and allows for simulation of its behavior (Todorovski & Džeroski, 2007). For simulation, process-based models are transformed to systems of ordinary differential equations (ODEs), a widely accepted formalism for modeling dynamic systems. ODEs allow for long-term simulation of the system behavior given only its state at the initial time point and the time series corresponding to the input control variables. Models that allow for accurate long-term prediction of system behavior are typically hand-crafted by engineers and experts in the domain at hand. The process-based modeling approach allows for automated learning of such models in different domains (Atanasova, Recknagel, Todorovski, Džeroski, & Kompare, 2006a; Atanasova et al., 2006c; Taškova, Šilc, Atanasova, & Džeroski, 2012).

However, existing approaches to process-based modeling mostly focus on building descriptive models that explain the observed behavior of the system, however do not generalize well enough to predict future system behavior (Simidjievski, Todorovski, & Džeroski, 2015). Improving the predictive performance of process-based models is still a challenge. In this paper, we address this challenge by proposing ensembles of such models. This is a standard approach for improving the predictive performance of models in machine learning Dietterich (2000). An ensemble is a set of models, referred to as base models or ensemble constituents; its prediction is a combination of the predictions obtained with the individual ensemble constituents. They are usually employed in the context of supervised learning tasks of classification (Smith et al., 2015) and regression (King, Abrahams, & Ragsdale, 2014; Tay, Chui, Ong, & Ng, 2013) to address the problems of over-fitting, high dimensionality, or missing features in the training data, resulting in predictive performance gain as compared to that of a single model. While regression ensembles can be also used as models for time-series forecasting (Ma, Dai, & Liu, 2015), they have

[*] Corresponding author at: Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, Ljubljana 1000, Slovenia. Tel.: +386 1 477 3635.
E-mail addresses: nikola.simidjievski@ijs.si (N. Simidjievski), ljupco.todorovski@fu.uni-lj.si (L. Todorovski), saso.dzeroski@ijs.si (S. Džeroski).

http://dx.doi.org/10.1016/j.eswa.2015.07.004
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

so far been used to make short-term predictions, i.e., predict only the values of the system variables in immediate future and not the long-term system behavior.

The main motivation for the work presented in this paper is to improve the accuracy of long-term predictions of process-based models. To this end, our specific objective is to design and implement methods for learning ensembles of process-based models from data and knowledge. We conjecture that ensembles learned using the implemented methods will outperform single process-based models in terms of their accuracy/error of long-term prediction of system behavior. To test the validity of this conjecture, we perform an extensive evaluation of the implemented methods on the task of modeling and predicting population dynamics in aquatic ecosystems. The empirical evaluation allows us to decide between the ensemble methods of bagging (Breiman, 1996a) and boosting (Drucker, 1997; Freund, 1999) as well as design alternatives considered within this methods.

The remainder of the paper is organized as follows. In Section 2, we put our work in the context of related work on process-based modeling and ensemble learning. Section 3 provides an introduction to the process-based modeling paradigm and its specific implementation PRoBMoT (Čerepnalkoski, Taškova, Todorovski, Atanasova, & Džeroski, 2012) through an example modeling task from the domain of population dynamics in aquatic ecosystems. Section 4 presents the two methods we propose and their implementation as an extension to PRoBMoT for learning ensembles of process-based models. In Section 5, we present the experimental setup of the empirical framework for evaluating the developed methods on three tasks of modeling population dynamics in aquatic ecosystems. More specifically, we test the utility of ensembles of process-based models in the context of modeling three ecosystems: Lake Bled in Slovenia, Lake Kasumigaura in Japan, and Lake Zurich in Switzerland. Section 6 presents the results of the empirical evaluation and discusses them in the context of related research. Finally, Section 7 concludes this paper and suggests directions for further work.

2. Related work

The work presented in this paper extends the state-of-the-art of process-based modeling (Bridewell, Langley, Todorovski, & Džeroski, 2008; Todorovski & Džeroski, 2007). More specifically, it relates to previous work on process-based modeling that has proven successful for building descriptive models of population dynamics in a number of real-world aquatic ecosystems (Atanasova et al., 2006a, 2006c; Džeroski & Todorovski, 2003). These studies focus on descriptive models used to explain the observed behavior of the system at hand and not (so far) employed for predicting future (unobserved) system behavior. Simidjievski et al. (2015) present a proof-of-principle experiment and show that bagging process-based models can improve their predictive performance. However, this experiment is of limited scope since it does not consider alternative ensemble methods, such as boosting, which is considered here. The study presented there is purely empirical and does not tackle the methodological issues of learning ensembles that are being addressed in this paper.

The task of learning ensembles of process-based models has been also addressed by Bridewell, Asadi, Langley, and Todorovski (2005), who aim at integrating the ensemble constituents into a single (meta-level) process-based model that includes the structure fragments most frequently present in the constituents. The results show that the resulting meta-level model is more robust in terms of over-fitting to the observed data. Note however, that in the evaluation their proposed approach, the authors estimate the out-of-sample error by taking random sub-samples of the observed (training) time-series data. The ability of the meta-level model to predict system behavior outside the observed time span of the training data (i.e., future system behavior), which we focus on in this paper, has not been considered, far less evaluated.

On the other hand, our work is related to the long tradition of applying methods for learning ensembles to various predictive modeling tasks in different scientific and engineering domains. The ensemble methods that tackle the problem of time-series forecasting are most closely related. (Ma et al., 2015) present a method for pruning the set of ensemble constituents, in particular support-vector regression models, for the purpose of optimizing the size of an ensemble and its predictive performance. They apply the developed method to four tasks of forecasting financial time-series (stock indices) and show that the approach can reduce ensemble size while retaining reasonable levels of predictive accuracy. Similarly, Kourentzes, Barrow, and Crone (2014) also aim at short-term forecasting of financial time series using ensembles of neural networks. Their main contribution is the identification of the most suitable operator for aggregating the predictions of individual ensemble constituents. Note that all the forecasting tasks, considered in these two papers, are short-term since they aim at forecasting the next-time-point values of stock indices or retail prices. In contrast, our process-based models aim at long-term (typically one year) prediction that concern periods with potentially indefinite ranges of time points.

Finally, process-based modeling is a term frequently used in the realm of system analytics and modeling business processes. Recently, there has been interest in automatic building of enactment plans related to the same declarative specification of a given business process (Jiménez, Barba, del Valle, & Weber, 2013). Business process modeling formalisms follow the standard specification languages, such as BPMN (White, 2004), designed for the specific purpose of specifying declarative models of business processes that can be used for understanding, simulating and optimizing business processes. However, the business process models are quite different from the process-based models considered here that provide means for a mathematical formalization of the quantitative change of the observed system state through time. Our process-based models are thus based on an entirely different formalism that we will introduce in the next section.

3. Process-based modeling

Complex models are derived with the express purpose to recreate the observed behavior or simulate the subsequent states of a dynamic system under various conditions. Scientist and engineers often relate such models to the processes that govern the dynamics of the modeled system, and to the entities involved. These relations are commonly construed with equation-based specifications of the dynamics, and compiled into a set of ordinary differential and/or algebraic equations. The set of equations describes the change of the system's state over time and therefore is used to simulate past, present and future behavior of the system at hand. However, while equations offer a quantitative way of expressing models, they lack the ability to qualitatively express the structure of the modeled system in terms of interacting entities and processes.

The approach of process-based modeling of dynamic systems aims at constructing models which contain a high-level explanatory structure and a low-level mathematical formulation which allowing for making predictions. Process-based models integrate domain-specific modeling knowledge and data into explanatory models of the observed systems. A process-based model consists of two basic types of elements: entities and processes. Entities represent the state of the system. They incorporate the input variables (forcing terms of the system), state variables (the internal state of the system) and the constants related to the components of the modeled system. The entities are involved in complex interactions represented by the processes. The processes include specific details of how the entities interact in terms of equations and sub-processes.

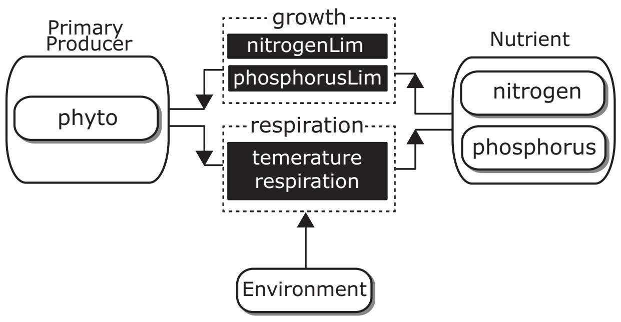


Fig. 1. Graphical representation of the relations (arrows and black boxes) between the entities (oval transparent boxes) in a simple lake ecosystem.

The task of process-based modeling, or learning process-based models from knowledge and data, can be specified, in terms of its inputs and outputs, as follows:

Input

– Measured data of the variables in the observed system.
– Domain-specific modeling knowledge.

Output

– Process-based model of the observed system.

The measured values of the observed variables are continuous, contiguous and may be non-uniformly distributed. We are interested in models that can predict the behavior of the system, i.e., how its state changes over time. Since, the state is represented by continuous variables, the task at hand resembles the task of regression.

Regression models, applied in the context of time-series data, are typically used for short-term prediction of the state at the next time point, based on the observed values of the current and previous states. In contrast, differential equations provide long-term predictions over many following time points, based only on the initial values of the target variables; no observations are necessary at the intermediate time points. Note also, that the use of modeling knowledge is an advantage of the process-based modeling approach over regression approaches since it improves model interpretability. The process-based models upgrade the purely empirical approaches, and strive at explaining how and why the dynamic system behaves under various conditions as opposed to just explaining/predicting how the measurements vary.

In the continuation of this section we are going to explain in more detail the process-based modeling paradigm through the prism of process-based models and the process of their learning from knowledge and data. To properly asses the relevant details of the paradigm, we are going to illustrate its use on a simple example of modeling population dynamics in an aquatic ecosystems.

3.1. Process-based models and modeling knowledge

Models of aquatic ecosystems are required for better understanding, prediction and management of such systems (Jørgensen & Bendoricchio, 2001). These models target the relations between entities, i.e., nutrients, primary producers, animals and environmental changes, that typically occur in aquatic ecosystems (Luenberger, 1979). Fig. 1 depicts a cyclic relationship involving a primary producer (phytoplankton, abbreviated as phyto) that grows by feeding on nutrients (nitrogen and phosphorous), the concentrations of which are influenced by the environment and the process of phytoplankton respiration (Atanasova, Todorovski, Džeroski, & Kompare, 2006b).

In order to model such a system using the process-based paradigm, we first need to formalize the modeling knowledge. Process-based modeling allows for a high-level representation of knowledge, cataloged in a domain-specific library of entity and process templates. The templates embody general properties of the interactions that govern the dynamics in the domain at hand and serve as recipes for establishing specific entities and processes observed in a given system.

Table 1 depicts an example of a simple library for modeling population dynamics in aquatic ecosystems. The first four declarations correspond to template entities organized in a hierarchy. The template entities of *EcosystemEntity* and *Environment* are at the top of the hierarchy; the first corresponds to the entities of the aquatic ecosystem, while the second to its environment. Down the hierarchy, the *EcosystemEntity* is then specialized into the two template entities of *PrimaryProducer* and *Nutrient*.

Each entity template may include constant and variable properties, which are inherited down the hierarchy. The variable properties (denoted `vars`) are those which change over time: the *EcosystemEntity* has as variable property its current concentration (denoted *conc*). Similarly, the *Environment* entity includes the *temperature* variable. The constant properties that do not change are denoted with `consts`, e.g., the template entity *PrimaryProducer* has the constant property *maxGrowthRate*. Finally, note that each variable property can have an aggregation function (denoted `aggregation`) that specifies how multiple influences on the variable (originating from different processes) are combined: the influences on the *conc* variable of *EcosystemEntity* are summed up, while the influences on the *growthRate* variable are multiplied.

The template processes are also organized in a hierarchy and specify which entities can interact and how these interactions govern the dynamics of entity variables. Highest in the hierarchy of aquatic ecosystem processes are the templates of Growth, Respiration and GrowthRate. The Respiration template specifies two differential equations that model the influence of the respiration process on the concentrations of the primary producer and the nutrients involved. Similarly, the Growth template encodes the influences of growth on the same concentrations. Additionally, Growth involves a subprocess GrowthRate, which implies that a GrowthRate must be specified for each nutrient involved in the process of growth. The hierarchy specifies two instances of the template process GrowhRate, MonodGrowthRate and ExpSaturatedGrowthRate, that correspond to two alternative models of growth limitation due to limited nutrient supply. Note therefore, the hierarchical structure of the process templates allows for the specification of modeling alternatives for an observed interaction between entities.

Once we have a library of process and entity templates, we can formulate a process-based model as a set of instances of the templates in the library. Table 2 presents a process-based model of the system depicted in Fig. 1: note the one-to-one correspondence between entities and process depicted in the system graphical presentation and the process-based model. Each entity and process instance incorporate the variables and the constants related to the corresponding template. For each variable property, its role in the observed system is specified. Exogenous variables represent input system variables that are not the subject of modeling (e.g., the environment temperature), while endogenous variable represent system state variables that are subject to modeling (e.g., the concentration of phytoplankton). For each endogenous variable, we have to provide its value at the initial time point. Finally, for the constant properties of the entities and processes, we have to specify their values.

Considering the mathematical formulation of the processes embodied in the library, we can compile this high-level representation of the interactions in the system into a system of algebraic and differential equations adequate for simulation. Table 3 provides the quantitative formulation of the process-based model presented

Table 1
Template entities and processes for modeling population dynamics in aquatic ecosystems. Here td(x) denotes the time derivative of x.

```
template entity EcosystemEntity{
   vars:conc{aggregation:sum;}}
template entity PrimaryProducer:EcosystemEntity{
   vars:growthRate{aggregation:product};
   consts:maxGrowthRate; }
template entity Nutrient:EcosystemEntity{consts:alpha;}
template entity Environment{vars:temperature;}
template process Growth(pp:PrimaryProducer,ns:Nutrients){
   processes:GrowthRate(pp,ns);
   equations:
     td(pp.conc) = pp.maxGrowthRate * pp.growthRate * pp.conc,
     td(ns.conc) = −n.alpha * pp.maxGrowthRate * pp.growthRate * pp.conc; }
template process Respiration(
pp:PrimaryProducer,ns:Nutrients,env:Environment){
   consts:respRate,refTemp,minTemp;
   equations:
     td(pp.conc) = −respRate * pp.conc * pp.conc
              *(env.temperature-minTemp)/(refTemp-minTemp),
     td(ns.conc) = respRate * pp.conc * pp.conc
              *(env.temperature-minTemp)/(refTemp-minTemp); }
template process GrowthRate(pp:PrimaryProducer,n:Nutrient){}
template process MonodGrowthRate:GrowthRate{
   consts:halfSaturation;
   equations:
     pp.growthRate = n.conc/(n.conc + halfSaturation);}
template process ExpSaturatedGrowthRate:GrowthRate{
   consts:saturationRate;
   equations:
     pp.growthRate = 1 − exp( − saturationRate * n.conc);}
```

Table 2
A process-based model of phytoplankton dynamics in the simple a lake ecosystem from Fig. 1, based on the template entities and processes from the library in Table 1.

```
//Entities
entity phyto:PrimaryProducer{
   vars:conc{role:endogenous; initial: 1.665;};
   consts:maxGrowthRate= 0.88;}
entity phos:Nutrient{
   vars:conc{role:exogenous;};}
entity nitro:Nutrient{
   vars:conc{role:exogenous;};}
entity env:Environment{
   vars:temp{role:exogenous;};}
//Processes
process growth(phyto, [phos, nitro]):Growth
   {processes:growthRate;}
process nitrogenLim(phyto,nitro):ExpSaturatedGrowthRate
   {consts:saturationRate=14.9;}
process phosophorousLim(phyto,phos):ExpSaturatedGrowthRate
   {consts:saturationRate=8.08;}
process respiration(phyto, [phos,nitro],env):Respiration
   {consts:respRate=0.036,minTemp=0.542,refTemp=17.4;}
```

Table 3
Ordinary differential equations obtained from the process-based model of phytoplankton dynamics presented in Table 2. Here td(x) denotes the time derivative of x.

$$\text{phyto.growthRate} = [1 - \exp(-8.08 * \text{phos.conc})] * [1 - \exp(-14.9 * \text{nitro.conc})]$$
$$\text{td}(\text{phyto.conc}) = 0.88 * \text{phyto.conc} * \text{phyto.growthRate} - \text{phyto.conc}^2 * 0.036 * \frac{\text{env.temp} - 0.542}{\text{env.temp} - 17.4}$$
$$\text{phyto.conc}(t_0) = 1.665$$

above. Since we are modeling just the concentration of phytoplankton (which is denoted as endogenous in Table 2) the system of equations consists of a single differential equation. This equation can be then simulated, thus generating a trajectory and utilizing it for further analysis. Fig. 2 shows the simulation of phytoplankton concentration obtained by using the process-based model. The *data* trajectory (represented by a dashed line) represents real measurements that can be used for a visual assessment of the process-based model performance.

In summary, process-based models have several characteristics which make them very efficient for tasks of modeling dynamic systems. First, they provide a conceptual representation of the structure of the modeled system, depicting the high-level relations (processes) between the system components (entities). Second, they allow for the high-level process-based representation to be translated into a low-level mathematical formalism depicted as a set of differential and/or algebraic equations, facilitating simulation of the systems behavior. Finally, the library of domain-specific knowledge allows for

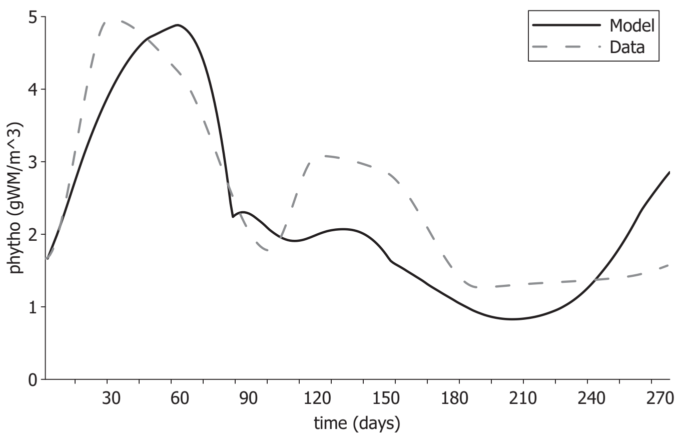


Fig. 2. Simulation of phytoplankton concentration dynamics (solid line) as modeled with the process-based model from Table 2 and its comparison to observed phytoplankton concentration (dashed line).

Algorithm 1 Outline of the generic algorithm for learning process-based models from knowledge and data.

```
1: function PBM($library, data, incompleteModel$)
2:     $components \leftarrow$ INSTANTIATE($library, incompleteModel$)
3:     for each ($structure \in$ ENUMERATE($components$,
       $incompleteModel$)
   do
4:         $modelEq \leftarrow$ COMPILETOEQUATION($strucutre$)
5:         $(model, error) \leftarrow$ PARAMETERESTIMATION($modelEq, data$)
6:         $modelList \leftarrow modelList \bigcup (model, error)$
7:     end for
8:     return SORT($modelList, error$)
9: end function
```

instantiation of a number of different building blocks for generating process-based models (Bridewell et al., 2008), which is particularly relevant for algorithms tackling the task of automated learning of process-based models from data.

3.2. Learning process-based models

Given the library of model fragments (template entities and processes), we can now formulate the task of learning process-based models from knowledge and data as a search task. Namely, given the specific entities in the observed system at hand, one can instantiate the template processes from the library into a set of specific processes that can be considered for inclusion in the model of the observed system. In turn, based on this set of specific model components, we can specify the search space of combinations thereof. Some of the combinations can be rejected as implausible, due to further modeling assumptions made by the user, such as the presence or absence of certain processes in the model.

The process-based modeling (PBM) algorithm, outlined in Algorithm 1, first takes as input *library* of domain-specific modeling knowledge, followed by *data* in form of time-series measurements of the observed dynamic system. The last input to the algorithm is an *incompleteModel* representing the modeling assumptions made by the modeler. First, the algorithm assembles all theoretically plausible model components by binding the entities of the observed system to the template processes from the library. Next, based on the incomplete model (taking into account the assumptions), the algorithm enumerates all the plausible candidate model structures. Each of these high-level structures is then compiled into a system of equations eligible for simulation. Before simulation, however, a parameter estimation task is being solved for the model structure at hand, to obtain values of the model parameters that best fit the observed data. After estimating the parameters for all candidate model structures, the algorithm outputs a sorted list of process-based models according to their error on training data, i.e., the discrepancy between the model simulation and the observed system behavior.

Most process-based modeling algorithms perform exhaustive search through a constrained space of candidate process-based structures, limiting the number of processes in the model (Bridewell et al., 2008). The more advanced approaches, such as Lagramge2.0 (Todorovski & Džeroski, 2007), combine the library of knowledge and the constraints into a grammar for enumerating plausible model structures. HIPM (Todorovski, Bridewell, Shiran, & Langley, 2005) allows for more sophisticated hierarchical constraints on the legal process combinations and tackles enumeration of model structures as a combinatorial problem. Last, ProBMoT (Čerepnalkoski et al., 2012) is a software platform for complete modeling, parameter estimation, and simulation of process-based models. It extends HIPM with explicit constraints (assumptions) for a particular domain at hand and employs a variety of meta-heuristic optimization methods. In this study, we use ProBMoT[1] as the base learning algorithm for learning constituents of ensembles of process-based models.

4. Ensembles of process-based models

Ensembles are commonly used for machine learning tasks, such as classification and regression (Maclin & Opitz, 1999; Rokach, 2010). However, a detailed layout of a methodology for learning ensembles of process-based models for predictive tasks has not been considered so far. Here we take note of the similarity between the tasks of process-based modeling and time-series regression, and we apply the idea of learning ensembles of regression models in the context of models of dynamic systems, that is, developing appropriate methods for learning ensembles of process-based models.

An ensemble is a set of models (referred to as base models or ensemble constituents) that is expected to have improved predictive performance compared to a single model. The idea behind ensembles is to maximize the overall predictive power by combing the predictions of the individual base models. The simplest form of an ensemble is a black-box. For a given "bag" of individual models, the resulting output is a combination of the individual predictions. For this reason, we first explore how an ensemble of process-based models can be simulated and how the resulting prediction can be interpreted. Next, we "open" the black-box, and investigate the different methods for learning the constituents of an ensemble, where we introduce a novel approach of learning ensembles of process-based models.

4.1. Simulating ensembles of PBMs

In order to simulate an ensemble, each base model needs to be simulated. The resulting ensemble output is a combination of the predictions of all individual base models. For obtaining a prediction for ensembles of process-based models, we use average, weighted average and weighted median as combining schemes, commonly used for regression tasks (Drucker, 1997).

In all cases, the real-valued predictions of the constituent models are combined per time-point, for each time-point separately. In the case of average, all base models participate in the resulting simulation equivalently. For weighted average and weighted median schemes, a confidence β is calculated for each of the base models based on their performance error. The base models with higher confidence will contribute more in the resulting ensemble simulation. The procedure for simulating an ensemble of PBMs is depicted in Algorithm 2.

The SIMULATEENSEMBLE() procedure takes as input: a set of process-based models denoted with *ensemble*, the library of domain

[1] Available at http://probmot.ijs.si.

Algorithm 2 Simulating ensembles of process-based models.

```
1: function SIMULATEENSEMBLE($ensemble, lib, D, scheme$) returns $\hat{y}_e$
2:     $Simulations \leftarrow \varnothing$        ▷ simulations from ensemble constituents
3:     $\hat{y}_e \leftarrow \varnothing$        ▷ $\hat{y}_e$: the resulting ensemble simulation
4:     for each $\{model, \beta\} \in ensemble$ do
5:         $\hat{y} \leftarrow$ SIMULATE($model, D$)
6:         if INRANGE($\hat{y}, lib$) then
7:             $Simulations \leftarrow Simulations \bigcup \{\hat{y}, \beta\}$
8:         else continue
9:         end if
10:    end for
11:    if $scheme = average$ then
12:        $\hat{y}_e \leftarrow$ AVERAGE($Simulations$)
13:    else if $scheme = weightedAverage$ then
14:        $\hat{y}_e \leftarrow$ WEIGHTEDAVERAGE($Simulations$)
15:    else
16:        $\hat{y}_e \leftarrow$ WEIGHTEDMEDIAN($Simulations$)
17:    end if
18: end function
```

knowledge *lib*, a data set *D* and a label *scheme* selecting the combination scheme used. The resulting prediction of the ensemble is a trajectory denoted with $\hat{y}_e$. First, each model from the set is simulated. The result of the prediction of a individual model for a data set *D* is a trajectory $\hat{y}$. Each model is accompanied with a confidence β, calculated based on the performance on a validation data set. We use this coefficient β in the weighted combining schemes. The pairs of trajectories and confidences $\{\hat{y}, \beta\}$ resulting from the simulation of the constituents in the ensemble is collected in the set *Simulations*.

In contrast to the task of obtaining an output from a regression model, where the resulting prediction is a single point for a given input, the task of predicting with process-based models is far more challenging. The simulation of a process-based model takes as input the initial values of the endogenous variables and the complete trajectories of the exogenous (forcing) variables. As output, it produces complete trajectories of the endogenous variables. In a predictive scenario, this can often lead to divergent trajectories and disastrous predictive misperformance. For this reason, we examine the simulated values of each prediction $\hat{y}$ whether they satisfy the range of constraints given in the library of background knowledge (line 6 in Algorithm 2). If a value from a prediction is outside the range specified in the library, the whole trajectory of that particular model is discarded, i.e., is not taken into account when calculating the resulting ensemble prediction. In this paper, we use this kind of dynamic ensemble pruning as a standard technique when selecting the ensemble constituents and simulating the ensemble prediction. Finally, the valid simulations (*Simulations*) along with the respective confidence β_v are combined in the resulting ensemble prediction.

4.2. Learning the constituents of an ensemble of PBMs

Theoretically, ensemble methods consist of two main components: a technique for learning a set of candidate base models, and a combining scheme specifying how the base model predictions are being aggregated into an ensemble prediction. Previously, we demonstrated how the base models are combined into an ensemble of process-based models. Here we focus on the methods for learning the ensemble constituents.

Based on how the constituents are learned, the ensembles can be homogeneous or heterogeneous. In homogeneous ensembles, the base models are learned with the same learning algorithm, but using different samples of the training data, where the sampling variants include: sampling of data instances as in bagging (Breiman, 1996a)

Algorithm 3 Bagging process-based models.

```
1: procedure BAGGING($lib, \{D_T, D_V\}, incompleteModel, k$)
2: returns $Ensemble$
3:     $Ensemble \leftarrow \varnothing$        ▷ set of base models
4:     for $i = 1$ to $k$ do
5:         $D_S \leftarrow$ SAMPLE($D_T$)    ▷ randomly sample the training set $D_T$
6:         $modelList_i \leftarrow$ PROBMOT($lib, D_S, incompleteModel$)
7:         $bestModel_i \leftarrow$ RANK($modelList_i, D_V$)
8:         $\beta_i \leftarrow$ CONFIDENCE($bestModel_i, D_V$)
9:         $Ensemble \leftarrow Ensemble \bigcup (bestModel_i, \beta_i)$
10:    end for
11: end procedure
12:
13: function CONFIDENCE($model, D$) returns $\beta$
14:     let $\hat{y}$        ▷ simulated system variable $y$
15:     let $y$        ▷ measured system variable $y$
16:     $\hat{y} \leftarrow$ SIMULATE($model, D$)
17:     $maxDisc \leftarrow \sup(|y_t - \hat{y}_t|)^2$    ▷ calculate max discrepancy between measurements $y$ and simulation $\hat{y}$, where $t = 0..N$ and $N$ is number of time-points in D
18:     $\bar{L} \leftarrow \sum_{t=0}^{N} \frac{|y_t - \hat{y}_t|^2}{maxDisc}$        ▷ calculate average loss
19:     $\beta \leftarrow \frac{\bar{L}}{1 - \bar{L}}$        ▷ calculate confidence
20: end function
```

and boosting (Freund, 1999); sampling of data features/attributes as in random subspaces (Ho, 1998); or both as in random forests (Breiman, 2001). On the other hand, in heterogeneous ensembles, the candidate base models are learned using different learning algorithms, possibly together with a combination function as in stacking (Wolpert, 1992).

Bagging (Bootstrap aggregation), developed by Breiman (1996a), is one of the first and simplest ensemble learning methods. This method uses bootstrap sampling with aggregation. First, randomly sampled data instances, with replacements, from the training set are used to obtain bootstrap replicates. Next, each base model is learned from a different bootstrap replicate.

Boosting refers to a general approach for obtaining an accurate prediction by combining several less accurate ones learned on a different distributions of the training data. The AdaBoost algorithm, proposed by Freund and Schapire (1997), is an implementation of the boosting approach for the task of classification. AdaBoost works iteratively; it uses different distributions of the training data for learning the base models at each iteration. Depending on the outcome of the past iteration this method decreases (for correct classification)/increases (for incorrect classification) the weights of every instance, thus changing the distribution for the subsequent iteration of training the model. In this way, the individual weak predictors focus on different instances, and their combination is more robust. In a similar fashion, the implementation of Drucker (1997) successfully tackles the problem of combining regressors using AdaBoost.

In the reminder of this section, we will describe the process of generating ensembles of PBMs, and identify the key design principles for extending the two methods outlined above (bagging and Adaboost) to learn ensembles of PBMs.

4.2.1. Bagging of PBMs

The procedure for bagging process-based models is presented in Algorithm 3. The procedure BAGGING() takes four inputs: a library of domain knowledge *lib*, data consisting of training data D_T and validation data D_V, an incomplete model *incompleteModel*, and an integer k denoting how many base models are to be generated. The output is a set of process-based models denoted with *Ensemble*. Using PROBMOT()(line 6), we learn a set candidate base models from different random samples D_S of the training data D_T . The PROBMOT() procedure follows the algorithm design principles of the process-based modeling paradigm, and resembles Algorithm 1 in terms of inputs, outputs and flow.

The notable difference from bagging in the context of regression is that in our case the data instances have a temporal ordering, that has to be retained in each data sample. To achieve this, we implement sampling by retaining the order of the instances and introducing a weight for each instance (time-point), and provide it as part of the data. The weight corresponds to the number of times the instance has been selected in the process of sampling with replacement (SAMPLE() procedure). Instances that have not been selected (the ones with weight 0) are simply omitted from the sample.

To take into account the weights when learning a model from the sample, we employ the weighted root mean squared error (WRMSE) implemented in ProBMoT:

$$WRMSE(m) = \sqrt{\frac{\sum_{t=0}^{N} \omega_t * (y_t - \hat{y}_t)^2}{\sum_{t=0}^{n} \omega_t}}, \tag{1}$$

where y_t and $\hat{y}_t$ correspond to the *measured* and *simulated* values (simulating the base model m) of the system variable y at time point t, N denotes the number of instances in the data sample, and ω_t denote the weight of the data instance at time point t.

The output of a modeling task, when using ProBMoT, is a list of process-based models, which is afterwards sorted according to their performance (line 7 in Algorithm 3). Depending on the input in the procedure, the ranking can be based on the performance on a separate validation data set D_V, or on the training sample (if $D_V == D_T$). The highest ranked model from each modeling task i (out of k) denoted as $bestModel_i$, becomes an ensemble constituent in the output *Ensemble*.

Each ensemble constituent is paired with its own confidence β. The CONDIFENCE() function takes 2 inputs: the highest ranked model returned by ProBMoT and a data set D. Fist the *model* is simulated on the data set D resulting in a trajectory $\hat{y}$. Based on the error at each time point in the trajectory an average loss $\bar{L}$ is calculated for the *model* (line 18 in Algorithm 3). From this loss, a confidence measure β is calculated, where low values of β denote high confidence. The β coefficient is an indicator of the performance of the base model and is used in the process of simulating the ensemble, i.e., combining the simulations of the constituents into an overall ensemble prediction.

4.2.2. Boosting of PBMs

The procedure for boosting of process-based models is presented in Algorithm 4. In analogy with bagging, the BOOSTING() procedure, takes the same four inputs: a library of domain knowledge *lib*, data consisting of training data D_T and validation data D_V, incomplete model *incompleteModel*, and an integer k denoting how many base models are to be generated. In contrast to bagging, here we start with the complete training data set (instead of a sample). In addition we introduce the concept of weights for each data/time-point, which are recalculated after every boosting iteration, (line 10 in Algorithm 4) according to the error made by the model from the previous iteration at the respective time point. In the learning phase, we use the WRMSE objective function, presented in Eq. (1).

The REWEIGHT() function takes 3 inputs: the highest ranked model (denoted with *model*) from the previous iteration, a data set D, and the respective set of weights ω. While this function resembles the CONFIDENCE() function, there are important differences: here we consider a set of time-point wise weights and loss (rather than a single overall loss), and we calculate this on the training data (in contrast to validation data). First, the *model* is simulated on the data set D. Next, based on the error at each time point in the trajectory and the set of weights ω, the weighted average loss $\bar{L}$ is calculation. Finally, the set of weights is updated: the smaller the loss the more the weight is reduced – focusing on harder parts of the data set in the future iterations of the algorithm.

Algorithm 4 Boosting process-based models.

```
1:  procedure BOOSTING($lib$, $\{D_T, D_V\}$, $incompleteModel$, $k$)
2:  returns $Ensemble$
3:      $Ensemble \leftarrow \emptyset$                        ▷ set of base models
4:      $\omega_t \leftarrow 1$          ▷ $\omega_t$ is the weight of time point $t$, where $t = 0..N$
                                          and $N$ is the number of measurements in $D_T$
5:      for $i = 1$ to $k$ do
6:          $modelList_i \leftarrow$ PROBMOT($lib$, $\{D_T, \omega\}$, $incompleteModel$)
7:          $bestModel_i \leftarrow$ RANK($modelList_i$, $D_V$)
8:          $\beta_i \leftarrow$CONFIDENCE($bestModel_i$, $D_V$)
9:          $Ensemble \leftarrow Ensemble \bigcup (bestModel_i, \beta_i)$
10:         $\omega \leftarrow$ REWEIGHT($model_i$, $D_T$, $\omega$)
11:     end for
12: end procedure
13:
14: function REWEIGHT($model$, $D$, $\omega$)   returns $\omega$
15:     let $\hat{y}$                          ▷ simulated system variable $y$
16:     let $y$                                ▷ measured system variable $y$
17:     $\hat{y} \leftarrow$SIMULATE($model$, $D$)
18:     $maxDisc \leftarrow \sup(|y_t - \hat{y}_t|)^2$   ▷ calculate max discrepancy between measurements $y$ and simulation
                                          $\hat{y}$, where $t = 0..N$ and $N$ is number of time-points in D
19:     $L_t \leftarrow \frac{|y_t - \hat{y}_t|^2}{maxDisc}$      ▷ calculate square loss at each time point $t$, where $t = 0..N$
20:     $\bar{L} \leftarrow \sum_{t=0}^{N} L_t * \frac{\omega_t}{\sum_{t=0}^{N} \omega_t}$   ▷ calculate weighted average loss, according to the weights
21:
22:     $\omega_t \leftarrow \omega_t * \left[\frac{\bar{L}}{1-\bar{L}}\right]^{1-L_t}$, $t = 0..N$   ▷ update weights
23:     $\omega \leftarrow$ NORMALIZE($\omega$, $N$)        ▷ normalize weights to $N$
24: end function
```

The output of the BOOSTING() procedure is a set of pairs (process-based models and their respective confidences) denoted with *Ensemble*. The highest ranked process-based model from each boosting iteration is considered as an ensemble constituent, for which a confidence is calculated. As in bagging, the ranking can be based on the performance of the process-based model on a separate validation data set D_V, or on the training sample (if $D_V == D_T$).

5. Experimental setup

In this section, we present the setup of the experiments used to evaluate the predictive performance of the ensembles of process-based models. The aim of the evaluation presented is three-tiered. First, we want to identify the ensemble method that leads to best long-term predictions of process-based models. In particular, we perform a comparative analysis of the predictive accuracies of the models obtained with bagging and boosting to the predictive accuracy of a single model. These experiments will confirm our hypothesis that ensembles of process-based models improve over the predictive performance of single models. Second, for each of the methods considered, we identify the optimal design decisions in terms of choosing the ensemble constituents, combining their predictions and choosing the ensemble size. In the following sections, we first introduce the data sets to be used in the experiments, then briefly describe the two other ProBMoT inputs, i.e., the library of modeling knowledge and incomplete models, and finally define the performance metrics used to asses the process-based models and ensembles thereof.

5.1. Data sets

In the experiments, we use fifteen data sets from the domain of aquatic ecosystems, in particular, modeling food-web dynamics in the three lakes of Bled in Slovenia, Kasumigaura in Japan, and Zurich in Switzerland. The original data sets comprise monthly measurements in the seven year periods from 1986 to 1992 for the Lake Kasumigaura (Atanasova et al., 2006a) and in the period from 1996 to 2002 for Lake Bled (Atanasova et al., 2006c) and Lake Zurich (Dietzel, Mieleitner, Kardaetz, & Reichert, 2013). To obtain daily values, the measurements were interpolated and daily samples were taken from the interpolation.

For each aquatic ecosystem, we split the original multi-year data sets into seven single-year data sets. Five of these were used (one at a time) for training the base-level models, one was used for validating the models in the process of selecting the ensemble constituents, and one was used to measure the predictive performance of the learned process-based models and ensembles thereof. Thus, we perform fifteen learning experiments; in each, we take a single-year training data set, learn a model using the train and the validation data sets, and test its predictive performance on the test data set. In the tables reporting the experimental results, we label the experiments with the labels B1–B5, K1–K5 and Z1–Z5 corresponding to the training data set using in the experiment, where, e.g., K3 denotes the Lake Kasumigaura data set for the third year (i.e., 1988).

5.2. ProBMoT inputs and parameter settings

In our experiments, we use the library of domain-specific knowledge for modeling population dynamics in aquatic ecosystems, based on the previous work by Atanasova et al. (2006b). To reduce the computational complexity of the experiments performed in this paper, we used a simplified version of the library, where we omitted some of the alternatives for modeling individual processes. The simplified version of the library and the incomplete models, lead to 320 candidate model structures (as opposed to 18144 with the whole library) for Lake Kasumigarua and 128 candidates (as opposed to 27216 with the whole library) for the other two lakes.

We use the same structure of the population dynamics model for all three aquatic ecosystems. It includes a single ordinary differential equation for a system variable representing the phytoplankton biomass (measured as *chlorophyll-a* in Lake Kasumigaura). The exogenous variables include the concentration of zooplankton *Daphnia hyalina* (where available, i.e., for Bled and Zurich only), the dissolved inorganic nutrients of nitrogen, phosphorus, and silica (ammonia in Kasumigaura), as well as two input variables representing the environmental influence of water temperature and global solar radiation (light).

ProBMoT uses the Differential Evolution (Storn & Price, 1997) method for parameter estimation with population size 50, strategy *rand/1/bin*, differential weight (F) and crossover probability (Cr) of 0.6. The limit on the number of evaluations of the objective function is one thousand per parameter. For simulating the ODEs, we used the CVODE simulator with absolute and relative tolerances set to 10^{-3}.

5.3. Evaluation metrics

To evaluate the predictive performance of a given model m, we use the relative root mean squared error (*ReRMSE*) (Breiman, 1984), defined as:

$$ReRMSE(m) = \sqrt{\frac{\sum_{t=0}^{N}(y_t - \hat{y}_t)^2}{\sum_{t=0}^{N}(\bar{y} - \hat{y}_t)^2}}, \quad (2)$$

where N denotes the number of measurements in the test data set, y_t and $\hat{y}_t$ correspond to the *measured* and *predicted*[2] value of the system variable y at time point t, and $\bar{y}$ denotes the mean value of y in the test data set. Note that the usual root mean squared error is considered here relative to the standard deviation of the system variable in the test data, thus allowing us to compare the errors of models for system variables measured on different scales.

We observe and compare the predictive performance (*ReRMSE*) of the models learned using different algorithms on the 15 data sets. To assess the statistical significance of the differences in performances among models obtained with different algorithms, we follow the recommendation by Demšar (2006) and use the corrected (Iman & Davenport, 1980) Friedman test (Friedman, 1940), followed by the post-hoc Nemenyi test (Nemenyi, 1963). The Nemenyi test is used to explain where the significant differences come from (from which pairs of algorithms) by computing the critical distance between the algorithm ranks at a given significance level; in our case, we set the significance level threshold at 95% (i.e., $p = 0.05$). The results of the Friedman–Nemenyi tests are depicted in average rank diagrams, such as the ones in Fig. 3.

Finally, to explore the correlation between ensemble performance and the diversity of the ensemble constituents, we measure the diversity of the constituents of the ensemble e as the average pairwise distance between the simulations of the constituents:

$$Diversity(e) = \frac{1}{\binom{|e|}{2}} \sum_{\{m_1,m_2\}\subset e} \sqrt{\frac{\sum_{t=0}^{N}(y_{1,t} - y_{2,t})^2}{N}}, \quad (3)$$

where $|e|$ denotes the number of models in the ensemble, N the number of measurements in the data set, m_1 and m_2 two models from e, and $y_{1,t}$ and $y_{2,t}$ the simulated states of these two models at time point t. To assess the performance improvement of the ensemble e over a single model m, we calculate:

$$Improvement(e, m) = -\frac{ReRMSE(e) - ReRMSE(m)}{ReRMSE(m)}. \quad (4)$$

We draw a scatter plot that depicts the correlation between ensemble diversity and performance improvement and calculate the Pearson correlation coefficient between them.

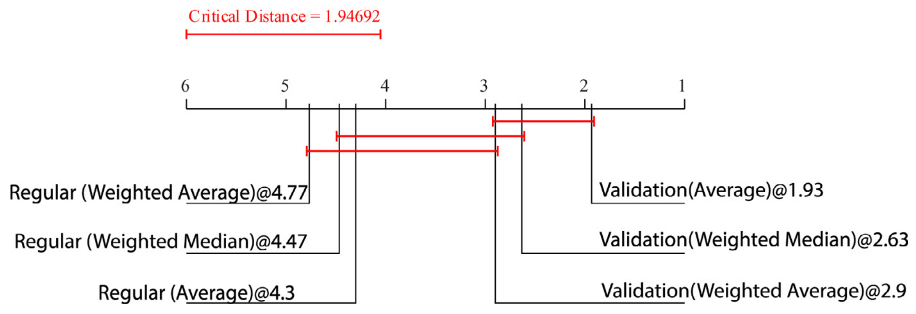
6. Results

In this section, we present and discuss the results of the empirical evaluation. In particular, we first identify the most suitable design decisions for individual ensemble methods. Then, we test the validity of our central hypothesis that ensembles of process-based models outperform the single process-based models. Finally, we investigate whether the performance improvement is related to the diversity of the predictions obtained with the ensemble constituents. Finally, we discuss the results in the context of the related machine learning research on ensembles.

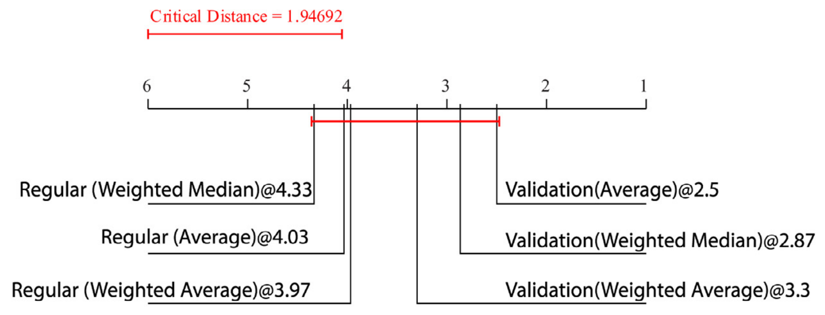
6.1. Learning ensembles of process-based models

The first design decision in an algorithm for learning ensembles of PBMs is related to the way of choosing the ensemble constituents. In each iteration of learning an ensemble, we select a single ensemble constituent, i.e., the highest ranked model from the ProBMoT output. Note that the latter represents a list of models, ranked with respect

[2] Predictions are obtained by simulating the model m on a test set.



(a) Bagging



(b) Boosting

Fig. 3. Comparison of the average ranks of different methods for selecting (and combining) ensemble constituents in terms of the predictive model performance averaged over the fifteen data sets in the case of bagging (top) and boosting (bottom). The labels describe the methods and give their average ranks.

to their performance on the training set. We refer to this base-line method for choosing ensemble constituents as *regular*. Alternatively, to avoid overfiting, we can re-rank the ProBMoT output, i.e., list of models, according to their performance on a separate validation data set. We refer to this selecting method as *validation*.

Fig. 3 summarizes the comparison between the performance of the regular and the validation method for choosing the base models to be included in the ensemble built in 50 iterations. The upper diagram depicts the results of the Friedman-Nemenyi test in the case of bagging, while the lower diagram depicts the results for the case of boosting. In both cases, choosing the base models based on their ranking on a separate validation set leads to superior ensemble performance. In the case of bagging, the superiority is statistically significant (note that the critical distance is smaller than the difference between the best validation rank and the best regular rank), while in the case of boosting it is not statistically significant.

We conjectured earlier that choosing ensemble constituents based on their performance on the training data set leads to ensembles that overfit the training data. Fig. 4 confirms the validity of this conjecture: in both cases (bagging and boosting), ensembles built using the regular selection method outperform the ones built with the validation selection method. This demonstrates a clear case of overfitting — while being superior on the training data, the regular selection method leads to ensembles with poor predictive performance.

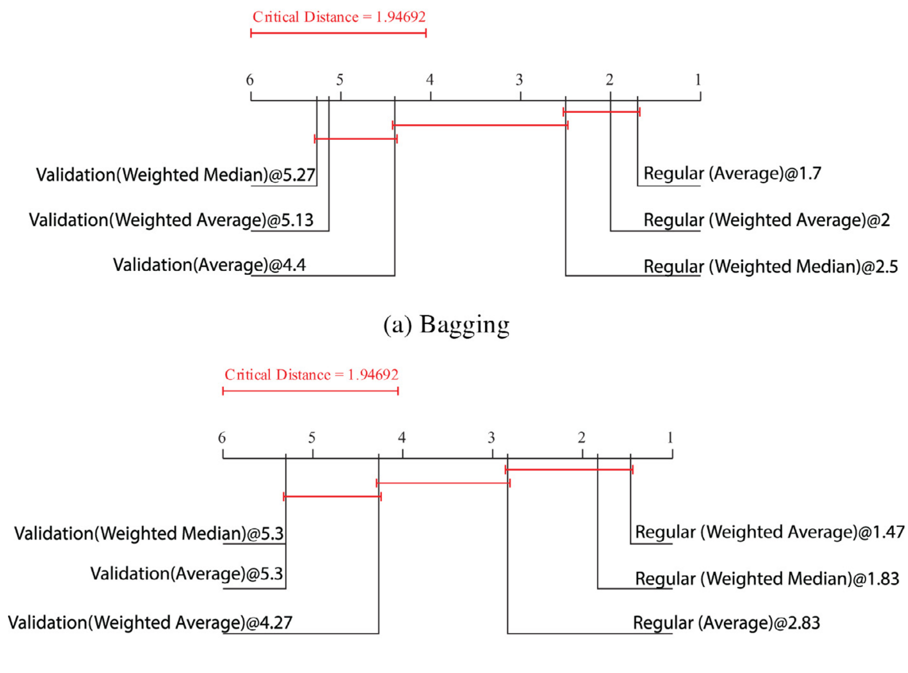
Next, we focus on the design decision concerning the most appropriate method for combining the simulations of the base models in the ensemble. We compare the performance of three methods commonly used in learning ensembles of regression models: average, weighted average, and weighted median (Breiman, 1984; Drucker, 1997). Fig. 5 depicts the comparison of the average ranks of the three methods for combining the base-model simulations in the case of bagging and boosting with 50 iterations in each case. In both cases, the simplest method, i.e. *average*, outperforms the other two. In the case of bagging, the observed difference between the average and the weighted average methods is statistically significant, while the other differences are not significant. Given this, and following the parsimony principle, we can conclude that the most appropriate method for combining the simulations of the ensemble constituents is the simple average.

For all experiments so far, we learned ensembles of fixed size, always consisting of 50 base models. In the last series of experiments, we aim at making a decision on the optimal number of iterations for learning ensembles of process-based models. To this end, we compare the predictive performance of ensembles consisting of 5, 10, 25 and 50 base models. Fig. 6 summarizes the results of the comparison for both bagging and boosting. The Friedman-Nemenyi diagram shows that the ensemble built in 25 iterations leads to the best performance in both cases. Note however, that the observed differences in performance are not statistically significant.

In summary, based on the presented results we make the following design decisions related to learning ensembles of process-based models: We choose the ensemble constituents based on their performance on a separate validation data set, we combine the base-model predictions (simulations) using simple, unweighted, average and perform 25 iterations of adding base-models to the ensemble. In all the further experiments, we used these algorithm settings for learning ensembles.

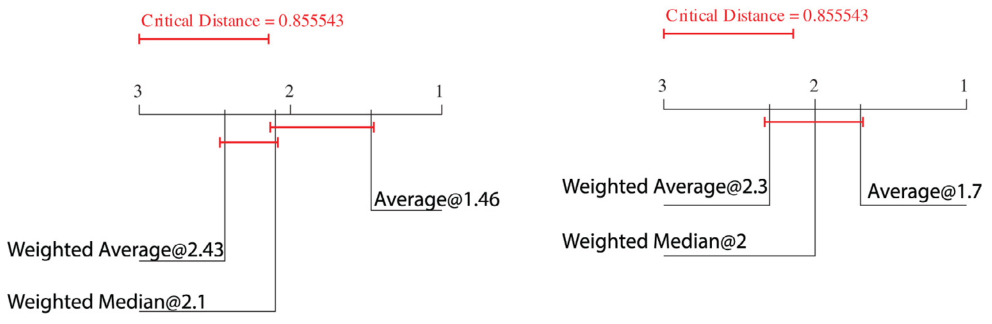
6.2. Ensemble performance and diversity

Having made the design decisions for learning ensembles of process-based models, we necessary now focus on testing our central hypothesis that ensembles improve the predictive performance of process-based models. To this end, we compare the predictive performance of the ensembles with the one of a single process-based model learned on the whole training data set and chosen based on its performance on the separate validation data set. Fig. 7 depicts the



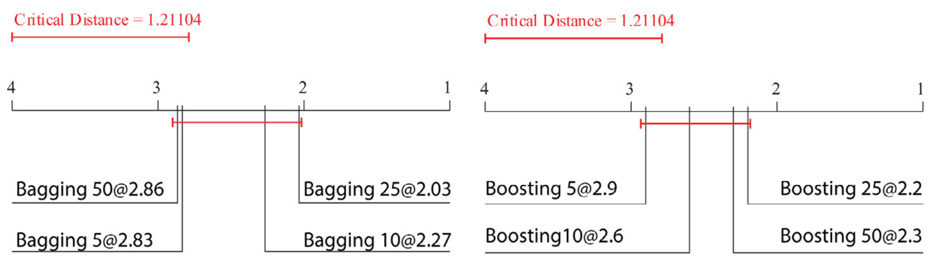
(b) Boosting

Fig. 4. Comparison of the average ranks of different methods for selecting (and combining) ensemble constituents in terms of the descriptive model performance averaged over the fifteen data sets in the case of bagging (top) and boosting (bottom). The labels describe the methods and give their average ranks.



(a) Bagging (b) Boosting

Fig. 5. Comparison of the average ranks of the three methods for combining the simulations of base models (average, weighted average, and weighted median) in terms of predictive performance averaged over the fifteen data sets in the case of bagging (left-hand side) and boosting (right-hand side).



(a) Bagging (b) Boosting

Fig. 6. Comparison of the average ranks of ensembles that include 5, 10, 25, and 50 base models in terms of predictive performance averaged over the fifteen experimental data sets.

comparison of the average ranks of a single model as well as the bagging and boosting ensembles, averaged over fifteen data sets.

The results of the Friedman-Nemenyi test show that both ensemble methods outperform the single process-based models. More importantly, the bagged ensembles *significantly* outperform the single models. These results support the central hypothesis of our paper that the ensembles improve the predictive performance of process-based models.

In our last set of experiments, we explore the relation of the observed significant improvement to the diversity of the simulations of the ensemble constituents. We first measure the relative improvement of the performance obtained by using an ensemble instead of a

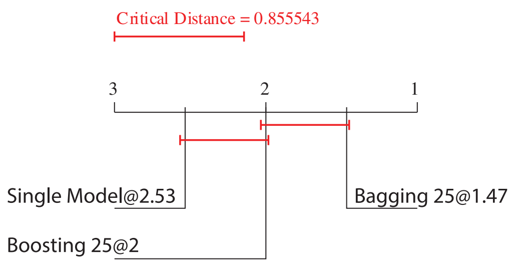


Fig. 7. Comparison of the average ranks of the single model and the ensembles (learned by using bagging and boosting) in terms of predictive performance averaged over the fifteen data sets.

Table 4
Diversity of the base models and the percentage of the relative improvement of the ensemble error over the error of the single model for the fifteen data sets. The Pearson correlation coefficient *r* between the improvement and the diversity is also given.

Case	Bagging		Boosting	
	Diversity	Improvement	Diversity	Improvement
B1	0.354	12.27%	0.342	12.97%
B2	0.561	4.48%	0.729	16.69%
B3	0.230	9.24%	0.477	11.77%
B4	0.617	17.45%	0.919	14.12%
B5	0.270	6.81%	0.408	24.29%
K1	1.010	1.04%	0.827	−24.32%
K2	1.030	35.06%	1.319	25.38%
K3	0.543	9.45%	0.656	3.59%
K4	0.598	1.02%	0.759	9.64%
K5	0.605	−1.53%	0.737	−7.97%
Z1	0.089	0.69%	0.411	−13.70%
Z2	0.234	7.23%	0.585	6.25%
Z3	0.223	5.72%	0.317	−1.01%
Z4	0.125	−2.33%	0.157	−4.21%
Z5	0.285	32.69%	0.702	15.16%
r	0.274		0.261	

single model. Then, we measure the diversity of base models in the ensemble. Finally, we analyze the correlation between the two.

Table 4 and Fig. 8 summarize the results of these experiments. The results presented in Table 4 confirm our previous finding: bagging outperforms single models in all but two data sets, K5 and Z4. Note that the loss of performance for these two data sets is minor (below 3%). On the other hand, the gain in performance can be substantial and reach up to 17% for Lake Bled, 35% for Lake Kasumigaura and 33% for Lake Zurich.

In the case of boosting, the improvement over the single model is less substantial, and more importantly, less consistent. The boosting method outperforms the single model for the majority of the training data sets (up to 25% for the case of Lake Bled; up to 26% for Lake Kasumigaura; and 16% for Lake Zurich). However for the remaining five data sets (K1, K5, Z1, Z3 and Z4) it under-performs. Note that two of these (K5 and Z4) are the same as the ones where bagging under-performs. For the remaining three cases, bagging makes a modest (up to 6%) improvement over the single model. This confirms that bagging is a better method for learning ensembles of process-based models.

Finally, we observe a varying degree of diversity between ensemble constituents for different data sets—diversity varies from 0.125 to 1.030. The scatter plots in Fig. 8 show weak positive correlation between ensemble diversity and relative improvement of performance. While measured Pearson correlation coefficient of 0.274 for bagging, and 0.261 for boosting, is neither high nor significant, the positive correlation is in line with the implicit assumption that ensembles improve predictive performance by exploiting the diversity of their constituents (Kuncheva & Whitaker, 2003).

6.3. Discussion

The results presented in this paper confirm our main hypothesis that ensembles of process-based models yield a significant gain in predictive performance when compared to a single model. Based on the performed empirical evaluation, we identified the main design decisions that need to be made when learning such ensembles by using bagging and boosting as underlying methods. In this context, it is very important that one uses a separate validation data set in addition to the training one when learning the base models included in the ensemble. The optimal ensembles of PBMs consist of relatively low numbers of constituent models, ranging between 10 and 25 for bagging and 25–50 for boosting (for both methods, the best performing ensembles comprised 25 constituents). For combining the simulation of the constituent process-based models, the results showed that the simplest combining scheme, i.e averaging, provides the most satisfying results both in terms of predictive accuracy and computational complexity.

The process-based models, when simulated in a predictive setting, can often produce divergent simulations, i.e., simulations where the systems variables leave their plausible ranges. Therefore, when simulating ensembles of PBMs, we explicitly handle this kind of behavior of the base models. We use the provided domain knowledge on system variable ranges to discard the invalid behaviors from the

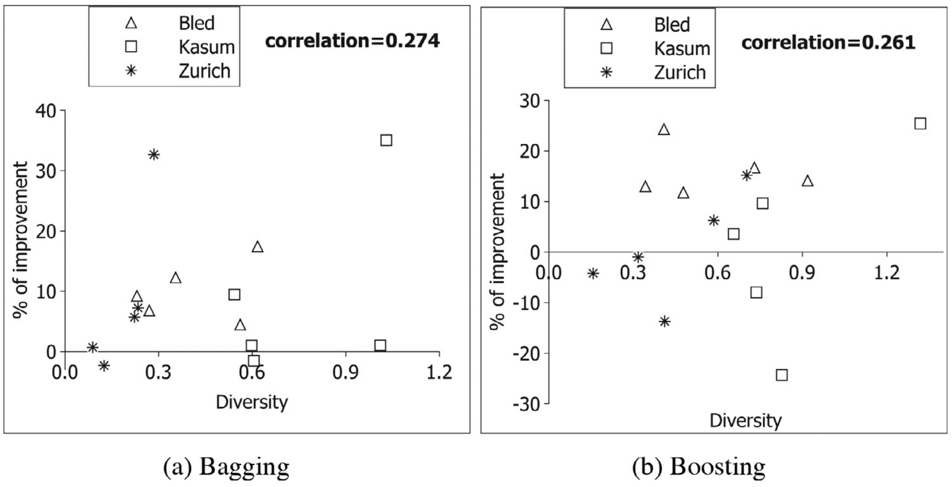


Fig. 8. Scatter plots depicting the correlation between the diversity of the base-model predictions and the relative error improvement between a single model and an ensemble for fifteen data sets.

resulting ensemble prediction. This can be viewed as a dynamic form of ensemble pruning.

Finally, our major conclusion is that both methods for learning ensembles of process-based models, following the design outlined above, outperform single models. More importantly, bagged process-based models provide a significant performance gain over a single model. Note that the improvement of performance over the single model is positively related to the diversity of the ensemble constituents — the higher the diversity, the greater the improvement.

However, in our case, the correlation between the diversity and the performance gain is very weak. Breiman (1996a) states that bagging can improve predictive performance when the ensemble is composed of models whose predictions vary sufficiently. The base models obtained in our approach have only modest diversity, which may limit the full predictive potential of the ensembles. One reason for this may be the lower number of model structures considered by ProBMoT, which used a simplified library of domain knowledge. This can be overcome by using the original library, which leads to tens of thousands of model structures.

In addition, the study of Joshi, Agarwal, and Kumar (2002) points out that the performance of boosting is correlated with the performance of the base learner, in our case, ProBMoT. This means that, when a single model obtained with ProBMoT, exhibits very good predictive performance, the ensembles exhibit similar or worse performance. This may be one reason for the behavior seen in Table 4 for the five highlighted experimental data sets (K1, K5, Z1, Z3 and Z4). Further investigations are required to determine whether increased diversity could lead to better performance.

7. Conclusion

In this paper, we address the task of learning ensembles of process-based models by designing, implementing and evaluating appropriate methodology. The developed methodology is general enough to allow for adapting different ensemble methods to the particular context of learning process-based models. This methodology is the main contribution of our paper, since it extends the scope of current process-based modeling approaches to the task of learning ensembles of process-based models. While the methodology has been only used in the limited context of adapting bagging and boosting, we can easily extend it towards other methods for learning homogeneous ensembles.

The second contribution of our paper is the extension of the scope of ensemble methods to process-based modeling. While previous proof-of-concept experiments have been performed for specific types of ensembles (Simidjievski et al., 2015), this is the first paper to provide a detailed layout of a methodology for building various types of ensembles of process-based models. This contribution is also important in the wider context of ensembles for time-series forecasting (Kourentzes et al., 2014; Ma et al., 2015; Tay et al., 2013). While forecasting ensembles have a narrow focus on short-term prediction tasks, where the value of the time series at the next time point is predicted, ensembles of process-based models provide accurate long-term predictions over many future time points. In contrast to Bridewell et al. (2005), who build ensembles that explain observed (albeit long-term) system behavior, the methods presented here provide accurate predictions of the unobserved future system behavior.

Note also that the results of the performed experimental evaluation confirm our central conjecture that ensembles provide much more accurate predictions of future concentrations of species in an aquatic ecosystems than a single process-based model. While single models struggle with achieving the performance of the baseline predictor (that predicts constant species concentrations at the level of their average), the ensembles of process-based models lead to accurate predictions of population dynamics over a long prediction periods, e.g., one season (year) in advance. When compared to previous results obtained in the domain of population dynamics (Atanasova et al., 2006a, 2006c), this is also a non-trivial improvement of predictive performance over the state-of-the-art models of population dynamics. These results are consistent over experiments with data from three real-world aquatic ecosystems: Lake Bled in Slovenia, Lake Kasumigaura in Japan, and Lake Zurich in Switzerland. This is the third important contribution of our paper, which mainly contributes to the domain of ecological modeling.

Several directions for further work can be followed. First, note that the validity of the results presented in this paper is limited to the particular domain of modeling population dynamics in aquatic ecosystems. An immediate continuation of the work presented here is to investigate the generality of the results across various domains and modeling tasks: both the superior performance of ensembles of process-based models (as compared to individual models) and the optimal settings and design decisions for the algorithms for learning them need to be verified for other domains and datasets. Next, following ideas from Bridewell et al. (2005), we intend to explore methods for incorporating the structure of the ensemble constituents into a single process-based model with good predictive performance.

Here we limit our attention to a single type of ensembles, where the diversity in the ensemble is obtained by learning ensemble constituents from different samples of the training data. In future, we can extend this narrow scope by considering other methods for generating ensemble constituents that rely on sampling the model variables or sampling the model components/templates in the library of modeling knowledge. The first method directly relates to the standard ensemble method of random subspaces (Ho, 1998). The second method would take different samples of entity and process templates from the library when learning individual models with the extra benefit of reducing the computational complexity of the individual learning tasks due to the reduced complexity of the search space.

Finally, we intend to extend our methodology towards learning interactive ensembles of models of dynamic systems, referred to as super-models (Mirchev, Duane, Tang, & Kocarev, 2012; van den Berge, Selten, Wiegerinck, & Duane, 2011). In contrast to ensembles, where the base models are learned and simulated independently and combined afterwards, within super-models, the base models can share and interchange information both during the learning and the simulation phase. In this context, one can learn the constituent models, their interconnections or both.

Acknowledgments

We thank Panče Panov and Dragi Kocev for the helpful discussions and comments on the work presented in this paper. We also thank Nataša Atanasovska for the knowledge and early work on the systems of Lake Bled and Lake Kasumigaura, along with Anne Dietzel for data from and knowledge about Lake Zurich. We would also like to acknowledge the financial support of the Slovenian Research Agency, via the grant P2-0103 and a young researcher grant to the first author, as well as the European Commission, via the grants ICT-2013-612944 MAESTRA and ICT-2013-604102 HBP.

References

Atanasova, N., Recknagel, F., Todorovski, L., Džeroski, S., & Kompare, B. (2006a). Computational assemblage of ordinary differential equations for chlorophyll-a using a lake process equation library and measured data of Lake Kasumigaura. In F. Recknagel (Ed.), *Ecological Informatics* (pp. 409–427). Springer.

Atanasova, N., Todorovski, L., Džeroski, S., & Kompare, B. (2006b). Constructing a library of domain knowledge for automated modelling of aquatic ecosystems. *Ecological Modelling, 194*(13), 14–36.

Atanasova, N., Todorovski, L., Džeroski, S., Remec, R., Recknagel, F., & Kompare, B. (2006c). Automated modelling of a food web in Lake Bled using measured data and a library of domain knowledge. *Ecological Modelling, 194*(1-3), 37–48.

Breiman, L. (1984). *Classification and regression trees*. Chapman & Hall.

Breiman, L. (1996a). Bagging predictors. *Machine Learning, 24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. doi:10.1023/A:1010933404324.

Bridewell, W., Asadi, N. B., Langley, P., & Todorovski, L. (2005). Reducing overfitting in process model induction. In *Proceedings of the 22nd international conference on machine learning (ICML '05)* (pp. 81–88). ACM Press.

Bridewell, W., Langley, P. W., Todorovski, L., & Džeroski, S. (2008). Inductive process modeling. *Machine Learning, 71*, 1–32.

Čerepnalkoski, D., Taškova, K., Todorovski, L., Atanasova, N., & Džeroski, S. (2012). The influence of parameter fitting methods on model structure selection in automated modeling of aquatic ecosystems. *Ecological Modelling, 245*, 136–165.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Proceedings of the 1st international workshop on multiple classifier systems* (pp. 1–15). Springer.

Dietzel, A., Mieleitner, J., Kardaetz, S., & Reichert, P. (2013). Effects of changes in the driving forces on water quality and plankton dynamics in three Swiss lakes long-term simulations with BELAMO. *Freshwater Biology, 58*(1), 10–35. doi:10.1111/fwb.12031.

Drucker, H. (1997). Improving regressors using boosting techniques. In *Proceedings of the 14th international conference on machine learning (ICML'97)* (pp. 107–115). Morgan Kaufmann Publishers Inc.

Džeroski, S., & Todorovski, L. (2003). Learning population dynamics models from data and domain knowledge. *Ecological Modelling, 170*, 129–140.

Freund, Y. (1999). An adaptive version of the boost by majority algorithm. In *Proceedings of the 12th annual conference on computational learning theory (COLT'99)* (pp. 102–113). ACM Press.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*(1), 86–92. doi:10.2307/2235971.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832–844.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics - Theory and Methods, 9*(6), 571–595. doi:10.1080/03610928008827904.

Jiménez, A., Barba, I., del Valle, C., & Weber, B. (2013). Optbpplanner: automatic generation of optimized business process enactment plans. In H. Linger, J. Fisher, A. Barnden, C. Barry, M. Lang, & C. Schneider (Eds.), *Building sustainable information systems* (pp. 429–442). US: Springer. doi:10.1007/978-1-4614-7540-8_33.

Jørgensen, S. E., & Bendoricchio, G. (2001). *Fundamentals of ecological modelling*: vol. 21. Elsevier.

Joshi, M. V., Agarwal, R. C., & Kumar, V. (2002). Predicting rare classes: can boosting make any weak learner strong? In *Proceedings of the 8th ACM sigkdd international conference on knowledge discovery and data mining (KDD'02)* (pp. 297–306). ACM Press. doi:10.1145/775047.775092.

King, M. A., Abrahams, A. S., & Ragsdale, C. T. (2014). Ensemble methods for advanced skier days prediction. *Expert Systems with Applications, 41*, 1176–1188.

Kourentzes, N., Barrow, D. K., & Crone, S. F. (2014). Neural network ensemble operators for time series forecasting. *Expert Systems with Applications, 41*(9), 4235–4244. http://dx.doi.org/10.1016/j.eswa.2013.12.011.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning, 51*(2), 181–207.

Luenberger, D. (1979). *Introduction to dynamic systems: theory, models, and applications*. Wiley.

Ma, Z., Dai, Q., & Liu, N. (2015). Several novel evaluation measures for rank-based ensemble pruning with applications to time series prediction. *Expert Systems with Applications, 42*(1), 280–292. http://dx.doi.org/10.1016/j.eswa.2014.07.049.

Maclin, R., & Opitz, D. (1999). Popular ensemble methods: an empirical study. *Journal of Artificial Intelligence Research, 11*, 169–198.

Mirchev, M., Duane, G. S., Tang, W. K., & Kocarev, L. (2012). Improved modeling by coupling imperfect models. *Communications in Nonlinear Science and Numerical Simulation, 17*(7), 2741–2751.

Nemenyi, P. B. (1963). *Distribution-free Multiple Comparisons* Ph.D. thesis. Princeton University.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review, 33*(1-2), 1–39. doi:10.1007/s10462-009-9124-7.

Simidjievski, N., Todorovski, L., & Džeroski, S. (2015). Learning ensembles of population dynamics models and their application to modelling aquatic ecosystems. *Ecological Modelling, 306*(0), 305–317. http://dx.doi.org/10.1016/j.ecolmodel.2014.08.019.

Smith, D., Dutta, R., Hellicar, A., Bishop-Hurley, G., Rawnsley, R., Henry, D., et al. (2015). Bag of class posteriors, a new multivariate time series classifier applied to animal behaviour identification. *Expert Systems with Applications, 42*(7), 3774–3784. http://dx.doi.org/10.1016/j.eswa.2014.11.033.

Storn, R., & Price, K. (1997). Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359. doi:10.1023/A:1008202821328.

Taškova, K., Šilc, J., Atanasova, N., & Džeroski, S. (2012). Parameter estimation in a nonlinear dynamic model of an aquatic ecosystem with meta-heuristic optimization. *Ecological Modelling, 226*, 36–61.

Tay, W.-L., Chui, C.-K., Ong, S.-H., & Ng, A. C.-M. (2013). Ensemble-based regression analysis of multimodal medical data for osteopenia diagnosis. *Expert Systems with Applications, 40*(2), 811–819.

Todorovski, L., Bridewell, W., Shiran, O., & Langley, P. W. (2005). Inducing hierarchical process models in dynamic domains. In *Proceedings of the 20th national conference on artificial intelligence (NAI'05)* (pp. 892–897). AAAI Press.

Todorovski, L., & Džeroski, S. (2007). Integrating domain knowledge in equation discovery. In S. Dzeroski, & L. Todorovski (Eds.), *Computational discovery of scientific knowledge*. In *Lecture Notes in Computer Science: vol. 4660* (pp. 69–97). Springer.

van den Berge, L. A., Selten, F. M., Wiegerinck, W., & Duane, G. S. (2011). A multi-model ensemble method that combines imperfect models through learning. *Earth System Dynamics, 2*(1), 161–177.

White, S. A. (2004). *Process modeling notations and workflow patterns. Technical report*. IBM Corporation.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.