SUMO - Supermodeling by combining imperfect models
Year 2 Report - Workpackage 7

Panče Panov, Jovan Tanevski, Nikola Simidjievski,
Ljupčo Todorovski, and Sašo Džeroski

October 19, 2012
## Contents

1 Background 7

2 State-of-the-Art 9

3 D7.1: Report on the generation of a diverse set of ODE models 12
  3.1 Methods that sample the instance space 14
  3.1.1 Bootstrap sampling for ODE ensembles 14
  3.1.2 Error-weighted sampling and boosting ODEs 16
  3.2 Methods that sample the feature space 17
  3.2.1 Random subspaces 17
  3.2.2 Random bagging 18
  3.3 Other methods 19

4 D7.2: Report on the selection of a complementary set of ODE models 20
  4.1 Distance measures on multivariate time series 21
  4.1.1 Non-elastic distance measures 23
  4.1.2 Elastic distance measures for time series 24
  4.1.3 Correlation-based distance measure 26
  4.1.4 Feature-based similarity measures 26
  4.2 Grouping behaviours of dynamic systems: Use of distance measures for hierarchical clustering 27
  4.2.1 Clustering different model structures, i.e., trajectories produced by simulating them 28
  4.2.2 Clustering observed behaviours (different datasets of measurements) 31
  4.3 Analysis of the clustering results 35

5 D7.3: Report on learning to interconnect ODE models 36
  5.1 Preliminary experiments with learning ensembles of ODEs 36
  5.1.1 Experiments with bootstrap sampling with replacement 38
  5.1.2 Experiments with bootstrap sampling w/out replacement 39
  5.1.3 Random window samples (and experiments) 42
  5.1.4 Experiments with boosting of ODEs 43
  5.2 Plans for further work 46

6 Conclusion 47
Context: Workpackage 7 and its structure

**WP7 Objectives.** The objective of WP7 is to develop methods for computational scientific discovery that can learn complete supermodels (ensembles of ODE models) of dynamical systems. The sub-objectives of this WP include the development of techniques for (semi)automated generation of constituent models, selection of an appropriate subset of models, and learning the form and coefficients of the interconnections among the models.

**WP7 Description of work.** WP7 will develop methods for learning complete supermodels. The supermodels are expected to be built in three phases: first generate diverse models, then select a set of complementary models, and finally learn the interconnections between the constituent models of an ensemble. These three phases form the three tasks that constitute WP7:

- **Task 7.1 Generate diverse set of ODE models.** To generate a diverse set of models, in this task, we adapt existing approaches from the area of ensemble learning. These include taking different subsamples of the data, taking projections of the data, and taking different learning algorithms (or randomized algorithms). Different subsets of domain knowledge may also be considered.

- **Task 7.2 Select a complementary set of ODE models.** Given a set of models, in this task, we use a measure of similarity between models to select models that are complementary. Different measures of similarity (or model performance/quality) can be considered. Besides the sum of squared errors and correlation, these can include the weighted sum of squared errors or robust statistical estimators.

- **Task 7.3 Learn to interconnect ODE models.** In learning the interconnections between the constituent models of the ensemble, in this task, we consider searching through the space of possible structural forms of the interconnections, coupled with parameter fitting for a selected functional form of the possible connections. For parameter fitting, we will use global optimization methods based on meta-heuristic approaches. The use of such parameter estimation methods is of crucial importance in supporting the use of different quality criteria, as well as avoiding local optima in search.
Deliverables in WP7. The progress of each task from WP7 is reported in the planned deliverables as follows:

- **D7.1 Report on the generation of a diverse set of ODE models.** This deliverable addresses task 7.1 and presents the progress on the task of generating a diverse set of ODE models. In the Annex 1 (Description of work) this deliverable was planned for month 18 of the duration of the project.

- **D7.2 Report on the selection of a complementary set of ODE models.** This deliverable addresses task 7.2 and presents the progress on the task of selecting a complementary set of ODE models. In the Annex 1 (Description of work), this deliverable was planned for month 27 of the duration of the project.

- **D7.3 Report on learning to interconnect ODE models.** This deliverable addresses task 7.1 and presents the progress on the task of learning to interconnect ODE models. In the Annex 1 (Description of work), this deliverable was planned for month 36 of the duration of the project.

The structure of the WP7 year 2 report. In the year 2 report for workpackage 7, we present the progress on all three tasks planned for workpackage 7 in Annex 1. The two introductory sections provides general background (Section 1) and state-of-the-art overview (Section 2) for the research within WP7. In Section 3, we present a revised version of the deliverable D7.1 submitted on 31 JUL 2012. In Section 4, we present deliverable D7.2 and the progress on the task of selecting a complementary set of ODE models. Finally, in Section 5, we present initial work on learning to interconnect ODE models organized in a preliminary version of deliverable D7.3. Updated and revised version of D7.3 will be provided for the Year 3 of the project.

Note that the deliverables of WP7 focus on learning ensembles of ODE models, i.e., supermodels. A related deliverable (Deliverable 1.6) is part of WP1, which is concerned with the fundamentals of supermodeling. This deliverable is concerned with solving the basic task of learning base ODE models and relies on the base learner ProBMoT for solving this task. Several extensions of ProBMoT are necessary for the successful learning of supermodels, e.g., the inclusion of capabilities to deal with weighted sum of squared errors as objective function, as well as capabilities to deal with multiple objective functions. Also, appropriate libraries of domain knowledge need to be developed for the learning of combination functions for the component models of a supermodel.
1 Background

Dynamic systems, the state of which changes over time, are ubiquitous in both science and engineering. Experts build models of dynamic systems to simulate and predict system behavior under various conditions. Models of dynamic systems typically take the form of ordinary differential equations (ODEs), where the rate of change of the values of the systems variables is expressed as a function of their current values. In the SUMO project, we focus on the task of constructing models of dynamic systems, i.e., the process of establishing models from observations and measurements of system behavior.

The major paradigm used when modeling dynamic systems is the approach of theoretical (knowledge-driven) modeling. A domain expert derives a proper model (ODE) structure, based on extensive knowledge about the system at hand, as well as domain-specific modeling knowledge. Measured data are then used to estimate the (constant) parameters in the model. In the alternative paradigm of empirical (data-driven) modeling, different model structures are explored to fit observed data in a trial and error process. This paradigm has been recently used to develop machine learning approaches to constructing ODE models from measured data.

Within the area of computational scientific discovery, equation discovery systems have emerged that use measured data to identify both the model structure and the values of the constant parameters in the model. They combine heuristic search methods with parameter estimation techniques: While the search methods explore the space of candidate model structures, the parameter estimation techniques find optimal values of a single structure and evaluate its fit against observations. The result of the evaluation in turn guides the search method towards a model with a good fit.

The most recent equation discovery approaches integrate the theoretical and empirical paradigms for modeling dynamic systems. Besides observed data, they take into account domain knowledge about the studied system: Process-based domain knowledge describes the basic types of processes that can take place in such a system and provides alternative modeling templates for each of them. The ability to take into account domain knowledge has been a key contributor to the success of these automated modeling approaches.

Machine learning approaches for equation discovery that learn ODE models from observed behaviors will be the major stepping stone for the proposed research. Another paradigm from machine learning that we draw upon in the SUMO project is the paradigm of ensemble learning. Ensemble methods create a set (ensemble) of diverse predictive models instead of a single one. Diverse models are typically obtained by applying a learning algorithm to different samples of the data. The individual models from the ensemble
(or their predictions) are combined by averaging or in a more complicated fashion, where the form and parameters of the combination function are also learned. Ensembles achieve high predictive performance, benefiting from the diversity of the individual models and outperforming them.

In machine learning, ensembles are typically employed in the context of predictive modeling in general and classification/regression in particular. They have also been considered in the context of predicting structured outputs, as well as other machine learning tasks, such as feature ranking and clustering. To the best of our knowledge, ensembles of (ODE) models of dynamic systems thus have not yet been considered in the machine learning literature. We will address them in WP7 of the SUMO project.

Ensemble predictions have been considered, however, for dynamic models of the climate system. Typically, a single climate model is taken and the diversity of predictions is achieved by perturbing the initial state from which the model is run or the constant parameters of the model: The resulting two types of ensembles are referred to as initial condition ensembles and perturbed physics ensembles, respectively. In contrast, the models within an ensemble generated by machine learning algorithm are completely different, both in their structure and parameters. Modest changes are considered in climate models due to the size of the models and the time complexity of their simulations.

A related recent development arising in the context of climate modeling is the paradigm of supermodeling. A supermodel consists of a set of interconnected (coupled) models: The component models are simulated (integrated) simultaneously and information is exchanged between the models (through the interconnections) on a time-step basis. This is in contrast to simulating each of the models independently and then combining their predictions.

It is important to note here that classification and regression models (typically considered in ensemble learning) do not depend on time and can be considered to make predictions instantaneously, while the predictions of an ODE model are obtained by simulation algorithms that are iterative. The task of combining ODE models and their outputs is thus more complicated, and several alternative approaches can be considered, including the model coupling approach taken in supermodeling.

The practical relevance of learning models of dynamic systems, on one hand, and the success of ensemble methods in yielding better predictions, on the other hand, provides us with strong motivation to investigate the topic of learning ensemble models of dynamic systems. Additional motivation is provided by the recent developments in supermodeling, which highlight the importance, as well as the difficulty and complexity of the problem of combining ODE models. The task at hand is thus both important and challenging.
2 State-of-the-Art

The tasks addressed in WP7 of the SUMO project aim at developing approaches for learning ensemble models of dynamic systems. It is thus of interdisciplinary nature and the related work that we consider comes from several areas. First, these include machine learning [28] and two of its sub-areas, computational scientific discovery [16] and ensemble learning [30]. Second, these include ensembles for prediction of climate [8]. Finally, these include the recent paradigm of supermodeling [40].

From the area of computational scientific discovery, we will build upon existing equation discovery systems for learning ODE models of dynamic systems from observational data. After initial research on this topic, which used a completely empirical (data-driven) approach [11], further developments (starting with [13]) have tried to integrate the theoretical and empirical paradigms for modeling dynamic systems: Besides observed data, they take into account domain knowledge.

Recent machine learning systems for learning ODE models deal with the task of inductive process modeling and exploit process-based domain knowledge [15, 37, 5]. Such knowledge describes the basic types of processes that can take place in the studied system (and other systems from the same domain of study) and provides alternative modeling templates for each of them. The most recent process-based modeling tool (ProbMoT)[9], that we build upon within the WP7 of the SUMO project, uses a cleaner representation formalism based on entity and process templates, includes several local and global optimization methods for parameter estimation, and allows for the use of different objective functions (and not only the sum of squared errors between measurements and predictions). Equation discovery and inductive process modeling methods (for learning ODE models of dynamic systems), developed so far, deal with the task of finding a single model of an observed behavior and do not provide support for building ensembles.

Ensemble methods [30] are machine learning methods that construct a set of models and combine their outputs into a single prediction in order to achieve better predictive performance. The key to the predictive power of ensemble models, at least intuitively, is the diversity of the base models within the ensemble [24]. Procedurally, ensemble methods first learn a diverse set of models and then (learn how to) combine them. The number of models in an ensemble can often be quite large, e.g., an ensemble can contain hundreds of decision trees.

In learning ensembles, a diverse set of models can be generated by modifying the learning data and applying the same learning algorithm or by using the same data and changing the learning algorithm. To modify the data, the
most popular ensemble methods generate different samples of the training examples, as done in bagging [3], or take different subsets of the complete set of features, as done in random subspaces [22]. Other methods, such as bagged random subspaces [31] and random forests [4] sample both the examples and the features.

Ensembles are typically employed in the context of predictive modeling, for solving tasks such as classification, regression, and predicting structured outputs: While they have been also used for other machine learning tasks, such as feature ranking and clustering, ensembles of (ODE) models of dynamic systems have not yet been considered in the machine learning literature.

Ensemble predictions have been considered, however, for dynamic models of the climate system [8]. Typically, a single climate model is taken and the diversity of predictions is achieved by perturbing the initial state from which the model is run [36]. Alternatively, (some of) the constant parameters of the model can be perturbed [29]. The resulting two types of ensembles are referred to as initial condition ensembles and perturbed physics ensembles, respectively.

The changes made to the climate models in order to achieve diversity of prediction are modest. This is due to the large size of the models and the time complexity of their simulations. In contrast, the models within an ensemble for classification or regression, generated by a machine learning algorithm, can be completely different, both in their structure and parameters.

Ensembles for climate prediction have explored minor variations (perturbations) of initial conditions and model parameters and have not considered ODE models of different structure.

A related recent development arising in the context of climate modeling is the paradigm of supermodeling [40]. A supermodel consists of a set of interconnected (coupled) models: The component models are simulated (integrated) simultaneously and information is exchanged between the models (through the interconnections) on a time step basis. Note that a small set of component models of the supermodel is typically considered, which are provided by domain experts and are not learned [39].

The key insight for supermodeling in the context of climate models comes from nonlinear dynamics and concerns the synchronization of attractors: Attractors in the state space of a dynamic system consist of states that a trajectory will visit repeatedly and arbitrarily close in due time. If two systems with the same or similar attractors are connected, it often happens that the evolutions of both attractors synchronize [42]. In supermodeling, the goal is to choose (learn) the connections between the models so that the models fall into synchronization with each other, as well as with reality.
Supermodeling assumes that a small set of component models is given, which exchange information via coupling, and focuses on learning the coupling coefficients: The combination of models is specific to ODE models and different from the usual approaches of averaging.

In WP7 of the SUMO project, we combine the machine learning paradigms of equation discovery and ensemble learning with insights from (supermodeling of) dynamic systems and develop approaches for learning ensemble (ODE) models of dynamic systems. We develop methods for learning diverse sets of ODE models (Task 7.1) as well as methods to (learn how to) select, combine and interconnect the models (Tasks 7.2 and 7.3). The combination methods considered include the combination of model outputs, as well as model coupling.
3 D7.1: Report on the generation of a diverse set of ODE models

In learning ensembles, two major approaches are employed to obtain a diverse set of models [17]. In the first case, the diversity of models is achieved by modifying the learning data. In the second case, diverse models are learned by changing the learning algorithm.

The learning data can be modified in two straightforward ways. First, we can sample the training instances (rows) from the data table. This is, for example, the approach taken in bagging [3], where different bootstrap samples are taken from the dataset and given to the same learning algorithm to produce different models. Second, we can sample the features/attributes (columns): This is the approach taken in random subspaces [22].

More complicated ways of modifying the data also exist. For example, in boosting [34], the re-sampling of the dataset is error-proportional: Data points where high errors are made by the model are given higher weight, i.e., have higher probability. Panov and Dzeroski (2007) [31] propose bagged random subspaces, a combination of bootstrap sampling of the instance space and random subspaces of the feature space, which achieves performance very similar to that of random forests and is more general.

Ensemble methods which use different learning algorithms also use two major approaches for achieving diversity. First, a base learning algorithm can be used with different values of its parameters: Randomized learning algorithms are a special case, where the outcome of learning depends on a seed used for the randomization. Random forests [4] employ a combination of bootstrap sampling and a randomized tree-learning algorithm. Second, we can learn each base model with a completely different learning algorithm, as is typically the case in stacking [41].

We adapt the above algorithms for modifying the data to the case of learning models of dynamic systems. Here we need to sample instances and features from an observed behavior of a dynamic system. Selecting an instance sample would in this case correspond to selecting observations of all systems variables taken at a sample of time points. Selecting a feature sample would mean selecting a subset of the system variables. We use these adapted versions of sampling together with a base algorithm for learning ODE models of dynamic systems, such as ProBMoT [9]. In this way, we obtain variants of the approaches of bagging, random subspaces, bagged random subspaces and boosting for the task of learning ODE models. Since there are not many algorithms for learning ODE models of dynamic systems, we only consider the first option of using different learning algorithms, i.e., using ProBMoT.
with different parameter settings or a randomized version of ProBMoT.

Several complications may arise when using sampling of the data in the context of learning ODE models. The sampling of data instances results in behaviors that are not observed at equidistant time points, even if this has been the case with the initial behavior. Also, sampling of features results in problems of learning ODE models under limited observability, where measurements are not available for all, but only for some system variables. Finally, considering different weights of the prediction errors made for different time points requires the use of weighted (and not ordinary) sum of squared errors as an objective function when learning the parameters and structure of the ODE models.

Fortunately, our base learning method of choice (ProBMoT) can deal with all of the above complications, since it can use full simulation in evaluating models (as needed for the case of limited observability or observations at non-equidistant time points, i.e., for bagging and random subspaces), as well as with different objective functions (needed for error-weighted sampling and boosting). However, these complications do make the original problem of learning ODE models more difficult, both in terms of computational complexity and in terms of the quality of the learned models. What the exact impact is of these complications can be studied in the context of evaluation of ensemble approaches for learning ODE models.

A final approach to generating diversity can only be considered in the context of learning ODE models from observed data and domain knowledge, the task addressed by our base learner ProBMoT. Namely, in a manner analogous to sampling instances and variables, we can sample from the given domain knowledge, i.e., only consider a randomly chosen subset of process templates from it. However, in some cases, this can make the task of learning an ODE model much harder (if not impossible). Thus, more sophisticated sampling methods may be needed in this case.

In the rest of this deliverable, the main focus is on the first approach to generating ensembles, where each ODE model in the ensemble is learned by repetitively applying the base learner on different data samples. We experiment here with four different data re-sampling strategies. Three of them use independent random samples of the whole training data set for learning each ensemble model (bagging), while one of them learn the next ensemble model on the data sample that depends on the errors of the models already included in the ensemble (boosting). Other approaches to generating ensembles are to be considered within the context of the experiments on learning ensembles of ODE models performed within Task 7.3.
3.1 Methods that sample the instance space

3.1.1 Bootstrap sampling for ODE ensembles

Bagging (Bootstrap sampling with aggregation) is an ensemble meta-algorithm developed by Breiman et al. which is one of the first and simplest ensemble techniques. This technique uses bootstrapping for manipulating the data instances, by sampling uniformly or with-replacements so-called bootstrap replicates from the training data. Each base model is learned from the bootstrap replicates and combined afterwards by averaging the output (regression) or voting (classification). This method successfully overcomes the overfitting problem, but it is not useful with linear models. Another advantage of Bagging is that it has no memory, and can be parallelized on different CPUs to handle different replicates which performance-wise is very useful.

Unlike the traditional method for data sampling, where part of the data is used for training the different models, which are combined into an ensemble afterwards (Figure 1); here we consider time point error weighting as a method for sampling.

![Figure 1: Classical bootstrap sampling with replacement.](image)

The classical approach of generating bootstrap samples for a building ensemble models for common classification task, includes generating samples from random data instances in the training set $T$. By uniformly sampling with replacements, it is very likely that some of the instances will repeat, and there
of generating m different training sets with size same as the original training set T. Furthermore, every sample is used in the learning algorithm, there of learning m different model classifiers which will participate in the ensemble. By sampling without replacements, m samples will be created with unique instances in each of them, there of creating smaller training sets which will be used in learning the model classifiers.

Figure 2: Creating samples by weighting time points.

On the other hand, our approach (Figure 2) differs in the sampling process from the classical approach. Mainly, for performing bagging, we choose uniformly random time points from the time series, and by increasing the weights \((w_1, w_2, \ldots, w_n)\) of those particular points in the evaluation process, we penalize (encourage) the errors made in the learning phase (Figure 3). As long for the bagging without replacements, we choose uniformly random time points which will participate with their error rates in the evaluation phase. Analogically, we created the window samples, but instead of choosing uniformly random time points form the series, we chose random starting point and every k following time points (where k is the size of the window). With this kind of set-up the parameters where refitted, and different models were generated. Every of the model simulations, was then averaged or weight-averaged (using the error rates as weights) into a descriptive ensemble model of the observed system.
For implementing the boosting technique, we slightly changed our approach. Firstly, in the first boosting iteration a model was fitted using the whole time series in the evaluation phase. In the following iterations, to every point of the time series a weight was assigned based on the errors reported in the previous iterations. The reweighting was done linearly, by incising every time point weight by the average amount of the normalized error values in the previous iterations for the particular time-point. After obtaining the models from every boosting iteration, an ensemble was created by averaging the simulation outputs of the models.

### Table 2. Time-point weighting

<table>
<thead>
<tr>
<th>$w_0$</th>
<th>$t_0$</th>
<th>$w_0 \cdot f(v_1, t_0)$</th>
<th>$w_0 \cdot f(v_2, t_0)$</th>
<th>$\ldots$</th>
<th>$w_0 \cdot f(v_m, t_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$t_1$</td>
<td>$w_1 \cdot f(v_1, t_1)$</td>
<td>$w_1 \cdot f(v_2, t_1)$</td>
<td>$\ldots$</td>
<td>$w_1 \cdot f(v_m, t_1)$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>$t_2$</td>
<td>$w_2 \cdot f(v_1, t_2)$</td>
<td>$w_2 \cdot f(v_2, t_2)$</td>
<td>$\ldots$</td>
<td>$w_2 \cdot f(v_m, t_2)$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\ddots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$w_n$</td>
<td>$t_n$</td>
<td>$w_n \cdot f(v_1, t_n)$</td>
<td>$w_n \cdot f(v_2, t_n)$</td>
<td>$\ldots$</td>
<td>$w_n \cdot f(v_m, t_n)$</td>
</tr>
</tbody>
</table>

$$\sum \min \left( \sum_{i=0}^{n} w_i \cdot f(v_i, t_i) \right) \quad \sum \min \left( \sum_{i=0}^{n} w_i \cdot f(v_i, t_i) \right) \quad \sum \min \left( \sum_{i=0}^{n} w_i \cdot f(v_i, t_i) \right)$$

**Figure 3: Time-point weighting.**

#### 3.1.2 Error-weighted sampling and boosting ODEs

Boosting [34] is a supervised ensemble learning technique which uses different history-based-weights for different instances of the training data for constructing the base models. Depending on the outcome of the past predictions/classifications this method decreases (for correct prediction/classification) or increases (for false prediction/classification) the weight value of every instance for the next iteration of training the model. This process can assure that the weak (with entropy different than 0.5 for binary classification) predictors/classifiers can focus on different instances, and thus creating one strong ensemble classifier. The most popular Boosting algorithm is AdaBoost (ADAptive BOOSTing) a meta-heuristic algorithm which was firstly introduced for binary-class problems. Adaboost successfully boost the performance of the weak classifiers/predictors by linearly combining them in a strong one. Even though the original AdaBoost doesn’t handle very well with a noisy data and less susceptible to the over-fitting problem than other algorithms, it is used in variety of pattern-recognition applications.

For implementing the boosting technique for the context of ODEs, we slightly changed our approach. Firstly, in the first boosting iteration a model was fitted using the whole time series in the evaluation phase. In the following iterations, to every point of the time series a weight was assigned based on the errors reported in the previous iterations. The reweighting was done linearly, by incising every time point weight by the average amount of the normalized error values in the previous iterations for the particular time-point. After obtaining the models from every boosting iteration, an ensemble was created by averaging the simulation outputs of the models.
3.2 Methods that sample the feature space

3.2.1 Random subspaces

The Random Subspaces (or Subset Splitting) method was designed by Ho, where the data feature space is randomly sampled in several sub-spaces and each model is learned over a different sub-space, which is a way of implementing the stochastic discrimination method. This method is useful in situations with a highly-dimensional (large number of attributes) data and small number of instances.

The traditional approach in subspace sampling (Figure 4) is randomly selecting different features from the feature space a in the training set T into sub-spaces and learning different models over each m of them. The resulting model predictions are then ensembled into one, using variety of techniques (e.g. averaging and weighted averaging).

![Figure 4: Random feature sampling.](image)

In random subspaces, we sample the feature space using random sampling without replacement. We propose here random sampling using feature weights, analogously to bootstrap sampling over the instance space. Analogously to assigning weights to instances in boosting, this would result in assigning weights to features. These weights would then be used in the error functions used in the process of learning ODEs (Figure 5). By assigning the weights ($\Omega_1, \Omega_2,..\Omega_m$) to the variables, we split the variable space into sev-
eral sub-spaces with different importance in the process of optimizing the objective functions in the parameter estimation phase. With this approach, we can easily distinguish several different models which will take part in the resulting ensemble.

<table>
<thead>
<tr>
<th>$\Omega_1$</th>
<th>$\Omega_2$</th>
<th>\ldots</th>
<th>$\Omega_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$</td>
<td>$v_2$</td>
<td>\ldots</td>
<td>$v_m$</td>
</tr>
<tr>
<td>$t_0$</td>
<td>$f(v_i,t_0)$</td>
<td>\ldots</td>
<td>$f(v_m,t_0)$</td>
</tr>
<tr>
<td>$t_1$</td>
<td>$f(v_1,t_1)$</td>
<td>\ldots</td>
<td>$f(v_m,t_1)$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$f(v_1,t_2)$</td>
<td>\ldots</td>
<td>$f(v_m,t_2)$</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

\[ f(v_i,t_n) \quad f(v_2,t_n) \quad \ldots \quad f(v_m,t_n) \]

\[ \min(\Omega_1 \times \sum_{i=1}^n f(v_1,t_i)) \quad \min(\Omega_2 \times \sum_{i=1}^n f(v_2,t_i)) \quad \ldots \quad \min(\Omega_m \times \sum_{i=1}^n f(v_m,t_i)) \]

Figure 5: Variable weighting.

### 3.2.2 Random bagging

Random Forests [4] was developed as a hybrid ensemble learning technique from the well-established bootstrapping and random sub-spaces methods. Random Forests uses decision trees as base models, which are learned using bootstrap replicates constructed from the training data, and different nodes from the trees are trained on a different data feature space. Random Forests is one of the most accurate algorithms in existence, it runs smoothly on large and complex datasets and variations of the original can handle even unbalanced and unlabeled data. However, due to the use of decision trees, this method can overfit some of the datasets, especially in cases with noisy data in classification and regression tasks.

Random Forests rely on a randomization of the decision tree induction algorithm in order to subsample the feature space. Different subsamples are considered in each node of the decision tree. Panov and Džeroski [31] combine bootstrap sampling of the instance space and feature subsampling in a more direct fashion, achieving similar effects to those of Random Forests. The approach they propose, however, has the advantage of being applicable to arbitrary learning algorithms (without requiring their randomization).

The two approaches of bootstrap sampling of the instance space and (bootstrap) sampling of the feature space can thus be combined into a hy-
brid approach. Here we can assign weights to variables and/or time points simultaneously. This is illustrated in Figure 6.

<table>
<thead>
<tr>
<th>(\Omega_1)</th>
<th>(\Omega_2)</th>
<th>(\Omega_m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_1)</td>
<td>(v_2)</td>
<td>(v_m)</td>
</tr>
<tr>
<td>(w_0)</td>
<td>(w_0 \cdot f(v_1, t_0))</td>
<td>(w_0 \cdot f(v_m, t_0))</td>
</tr>
<tr>
<td>(w_1)</td>
<td>(w_1 \cdot f(v_1, t_1))</td>
<td>(w_1 \cdot f(v_m, t_n))</td>
</tr>
<tr>
<td>(w_2)</td>
<td>(w_2 \cdot f(v_1, t_2))</td>
<td>(w_2 \cdot f(v_m, t_n))</td>
</tr>
</tbody>
</table>

\[
\sum_{i=1}^{n} w_i \cdot f(v_i, t_i) \\
\sum_{i=0}^{m} \Omega_i \\
\]

Figure 6: Time-point and variable weighting.

3.3 Other methods

Based on the above discussion, we propose a generalized sampling procedure. The procedure assigns weights to both the instances (time-points), i.e., \((w_1, w_2, \ldots, w_n)\), and features (system variables), i.e., \((\Omega_1, \Omega_2, \ldots, \Omega_m)\). In random sampling without replacement, the weights can only take the values of 0 and 1. In bootstrap sampling, they can take non-negative integer values, so that the \(w\)'s sum to \(n\) and the \(\Omega\)'s sum to \(m\), respectively. Without loss of generality, non-negative random weights can be assigned for both the \(w\)'s and \(\Omega\)'s, so that each of these sum to one. This is a more general sampling procedure as compared to all of the special cases considered above.

Another approach to generate diverse set of ODE models is to vary the input parameters of the base learner ProBMoT. An input parameter that has a strong influence on the resulting models is the library of domain knowledge, consisting of template entities and processes considered as model components. Sampling the set of template entities and processes can be a source of diversity in the ensemble. This sampling should take special care of retaining the hierarchical structure of the templates in the library so that the sampled library can be still used as a legal input for the base learner.
4 D7.2: Report on the selection of a complementary set of ODE models

The key to the predictive power of ensemble models, at least intuitively, is the diversity of the base models within the ensemble [24]. Ensemble methods first learn a diverse set of models and then (learn how to) combine them. In this Task 7.2 of the SUMO project, we first consider how to measure the diversity of the ODE models within an ensemble.

Note first that most of the published literature on learning ensembles, including survey articles [10, 17], considers ensembles of classifiers, i.e., predictors of discrete variables. For classifiers, a variety of diversity measures exist [24]. These can be divided into pairwise measures, that assess the differences in the predictions between a pair of classifiers from the ensemble, then average across all pairs, and non-pairwise measures, that simultaneously take into account the predictions of all classifiers in the ensemble.

We first consider pairwise measures of difference (or conversely, similarity) for ODE models, which, when simulated, serve as predictors of numeric system variables. These measures focus on similarity between a pair of simulated model behaviors. Each behavior is a tuple of time series, where each tuple member correspond to a system variable. In this context, we consider distance or similarity measures on time series. In this deliverable, we sum up or average the measures calculated on individual system variables to obtain the aggregated distance/similarity measure on an ODE model level. In principle, other aggregation functions [20], such as ordered weighted aggregation, can be applied.

For behaviors observed at the same series of equidistant time points one can apply distances on real-valued vectors, such as the Euclidean distance and weighted variants thereof or the correlation (cosine) similarity [27]. While the Euclidean distance is non-elastic, i.e., it does not allow for different baselines or scales, correlation does. However, neither of them allows for phase shifts along the time dimension and frequency changes. On the other hand, the elastic distance based on dynamic time-warping [33] accounts for both and, in addition, allows for comparing time series observed for different lengths of time or at different time points. Another approach to address the issues of different lengths, time points, and scale is to embed the time-series into a lower-dimension feature space, leading to feature-based distance measures.

We illustrate the utility of the implemented distance measures in the second part of this section. We perform clustering of the ODE models obtained in the process of automated modeling of the phytoplankton growth in Lake Bled in Slovenia. In turn, we use the clusters to select candidate ODE models.
to be included an ensemble; several very simple selection methods are con-
sidered in the experiments. Further experiments are to be conducted within
Task 7.2.

4.1 Distance measures on multivariate time series

The data being considered can be datasets of measurements from an exper-
iment performed under different conditions or a simulation of the behavior
of the variables of a dynamical model given different parameters or different
model structures, under the assumption that the same set of variables are
observed in each case. The data we would like to represent is a tuple of
time series which can be measurements or simulations of multiple variables
of a dynamical system. Each variable can be represented as a time-series.

An example of measured data for 3 variables of an ecosystem can be seen
plotted in Figure 7.

Each time-series \( X \), defined as a sequence of real values sampled at \( n \) spe-
cific time points \( X = [x_1, x_2, x_3, \ldots, x_n] \), can be rescaled by normalization or
standardization. Standardization (see Equation 1) is performed by dividing
each point \( X_i \) by the standard deviation (\( \sigma \)) of the values in the time series
after subtracting the mean (\( \mu \)), resulting in values which have zero mean and
The plot of the data represented in Figure 1 after standardization can be seen in Figure 3.

Figure 3 Normalized data for 3 variables of the Lake Bled ecosystem

Rescaling of the data is recommended in order to make the calculation of the distance between time series invariant to amplitudes, value shifting or different scales. In Figure 8 and 9 we present the standardized and normalized data from Figure 7.

In order to represent the data in a formal way which can be later used to perform clustering, we describe each instance (dataset or model simulation) with two attributes the first attribute is an ID of the instance while the second attribute is a relational attribute which points to a different table that contains the time series for the specific instance. Each instance in the table that contains the time series represents one measured behavior or one observed variable. Each instance has as many attributes as time points for which we have measurements or simulation values. Given the description each row in the table represents one time series from the dataset or the model simulation.

Figure 8: Standardized data for 3 variables of the Lake Bled ecosystem.

Normalization (see Equation 2) refers to rescaling by the minimum and the range of the values to make all values lie in the range $[0, 1]$.

$$x_i = \frac{x_i - \mu}{\sigma}$$ (1)

Rescaling of the data is recommended in order to make the calculation of the distance between time series invariant to amplitudes, value shifting or different scales. In Figure 8 and 9 we present the standardized and normalized data from Figure 7.

$$x_i = \frac{x_i - \min(x_k, k = 1..n)}{\max(x_k, k = 1..n) - \min(x_k, k = 1..n)}$$ (2)
The plot of the data represented in Figure 1 after standardization can be seen in Figure 3.

Figure 3: Normalized data for 3 variables of the Lake Bled ecosystem.

4.1.1 Non-elastic distance measures

The most widely used distance measure for vectors is the Euclidean distance. Let $X_i$ and $Y_i$ be two $n$-dimensional vectors. The Euclidean distance between these two vectors can be calculated as:

$$d_{euclid}(X_i, Y_i) = \sqrt{\sum_{j=1}^{n} (x_{ij} - y_{ij})^2}$$  \hspace{1cm} (3)

A generalization of the Euclidean distance is the Minkowski distance which can be calculated as:

$$d_{mink}((X_i, Y_i)) = \left( \sum_{j=1}^{n} (x_{ij} - y_{ij})^p \right)^{\frac{1}{p}}$$  \hspace{1cm} (4)

where $p$ is a positive integer. The most typically used values for $p$ in the Minkowski are $p = 1$ which corresponds to the Manhattan distance:

$$d_{manh}((X_i, Y_i)) = \sum_{j=1}^{n} |x_{ij} - y_{ij}|$$  \hspace{1cm} (5)

and $p = 2$ which corresponds to the previously described Euclidean distance. When $p = \infty$, the metric corresponds to the Maximum metric (Chebyshev
distance) which can be calculated as:

\[ d_{cheb}(\{X_i, Y_i\}) = \max_j |x_{ij} - y_{ij}|, j = 1..n \]  \tag{6} 

4.1.2 Elastic distance measures for time series

While the Euclidean distance and its generalizations are easy to compute they cannot handle cases where the time series which are being compared are time shifted or have different lengths. In order to deal with this problem, several other 'elastic' measures have been proposed. Elastic methods for comparing time series are derived from the methods used for string comparison where the distance between two strings can be defined as the minimal number of changes (edits) needed for one string can become identical to another. In the domain of vectors of real values, given two \( N \)-dimensional vectors, these methods try to align them in the best possible way that will minimize the required number of edit operations for the vectors to match or the required number of edit operations to minimize the distance between them. The elastic measures, as opposed to the other described metrics, have the ability to align and calculate the distance between time series with different lengths.

Let \( X_i \) and \( Y_i \) be vectors of real values representing time-series with lengths \( p \) and \( q \), respectively. The Edit Distance on Real sequences (EDR) \[6\] is based on the Levenstein distance metric \[26\] used for string matching. It calculates the minimum number of edit operations needed to change one time-series into another. In EDR, two elements match if the absolute distance between them is smaller than a predefined tolerance. If the elements match, there is no penalty added to the final distance, otherwise a penalty of 1 is added to the final distance. In the case of performing an operation such as insertion or deletion as in the Levenstein distance, a penalty of 1 is added to the final distance. The EDR between two vectors can be recursively defined as follows:

\[
d_{edr}(X_i^p, Y_i^q) = \min \begin{cases} 
d_{edr}(X_i^{p-1}, Y_i^{q-1}) + \text{subcost} \\
d_{edr}(X_i^{p-1}, Y_i^q) + 1 \\
d_{edr}(X_i^p, Y_i^{q-1}) + 1 
\end{cases} \tag{7}
\]

where the subcost is 0 if \( |x_{ip} - y_{iq}| < \varepsilon \) and 1 otherwise.

Another string distance measure which has been considered for adaptation to vectors of real values is the Longest Common Subsequence (LCSS) measure \[21\]. The extension to time-series \( LCSS_{\varepsilon, \delta}(X_i^p, Y_i^q) \) is defined in
as follows:

$$\begin{align*}
\min \left\{ \begin{array}{ll}
0 & \text{if } p < 1 \text{ or } q < 1 \\
1 + \text{LCSS}_{\varepsilon, \delta}(X^{p-1}_i, Y^{q-1}_i) & \text{if } |x_{ip} - y_{iq}| < \varepsilon \text{ and } |p - q| < \delta \\
\max \left\{ \text{LCSS}_{\varepsilon, \delta}(X^{p-1}_i, Y^{q}_i), \text{LCSS}_{\varepsilon, \delta}(X^{p}_i, Y^{q-1}_i) \right\} & \text{otherwise}
\end{array} \right. 
\end{align*}$$

$$\text{max} \left\{ \text{LCSS}_{\varepsilon, \delta}(X^{p-1}_i, Y^{q}_i), \text{LCSS}_{\varepsilon, \delta}(X^{p}_i, Y^{q-1}_i) \right\}$$

where the match is rewarded with 1 and no reward is given for insertion or deletion. The main idea behind this approach is that two time-series are similar to each other if they contain a long subsequence that is identical in both of them. The LCSS calculates the length of this sequence given that gaps in the sequences may appear.

In order to make the LCSS a distance measure, the following transformation is performed:

$$D_{\text{lcss}} = 1 - \frac{\text{LCSS}_{\varepsilon, \delta}(X^{p}_i, Y^{q}_i)}{\min \{p, q\}}$$

The Dynamic Time Warping (DTW) [33] distance, differs from the previous approaches in that it minimizes the required number of edit operations needed for minimizing the distance. The cost of a match is equal to the distance between the elements. On the other hand, the insert and delete operations become operations of replicating the previous value and the cost of the replication is equal to the distance between the replicated element and the currently observed element from the other time-series. The DTW distance between two vectors can be defined as follows:

$$d_{\text{dtw}}(X^{p}_i, Y^{q}_i) = |x_{ip} - y_{iq}| + \min \left\{ \begin{array}{l}
d_{\text{dtw}}(X^{p-1}_i, Y^{q-1}_i) \\
d_{\text{dtw}}(X^{p-1}_i, Y^{q}_i) \\
d_{\text{dtw}}(X^{p}_i, Y^{q-1}_i)
\end{array} \right. \quad (10)$$

The shortcoming of the previously described elastic measures is that neither of them can be considered as a distance metric. The Edit Distance with Real Penalty (ERP) [7], on the other hand has been designed as a variant of the Manhattan distance that can handle local time shifting. The ERP uses a real penalty between non replicated elements and a constant penalty for computing the distance when one of the elements is replicated. The ERP can be defined as follows:

$$d_{\text{erp}}(X^{p}_i, Y^{q}_i) = \min \left\{ \begin{array}{l}
d_{\text{erp}}(X^{p-1}_i, Y^{q-1}_i) + |x_{ip} - y_{iq}| \\
d_{\text{erp}}(X^{p-1}_i, Y^{q}_i) + |x_{ip} - g| \\
d_{\text{erp}}(X^{p}_i, Y^{q-1}_i) + |g - y_{iq}|
\end{array} \right. \quad (11)$$

where $g$ is a constant which is usually set to $g = 0$, thus giving intuitive geometric interpretation.
Each of the described distance measures is defined on two real vectors. In order to extend the measures to be applicable to multivariate time-series that represent behaviors of dynamical system, an aggregation (sum, mean ...) over the distances between the corresponding variables of each behavior can be calculated and for example used as a similarity measure needed for the task of clustering.

### 4.1.3 Correlation-based distance measure

The correlation between two trajectories can also be used to extract information about the similarity. Let \( X_i \) and \( Y_i \) be two \( n \)-dimensional vectors. The Pearson’s correlation coefficient can be calculated as:

\[
\rho = \frac{\sum_{j=1}^{n} (x_{ij} - \mu_{X_i})(y_{ij} - \mu_{Y_i})}{\sqrt{\sum_{j=1}^{n} (x_{ij} - \mu_{X_i})^2 \sum_{j=1}^{n} (y_{ij} - \mu_{Y_i})^2}}
\]

where \( \mu_{X_i} \) and \( \mu_{Y_i} \) are the means of \( X_i \) and \( Y_i \) respectively. If the two time series are maximally correlated or anti-correlated the correlations will be 1 and \(-1\) respectively. If the time series are uncorrelated the correlation coefficient will be close to 0. In order to use the Pearson correlation as a distance measure a simple transformation can be performed:

\[
d_{\text{pear}} = 1 - \rho
\]

Other transformations can also be devised [27, 19] in order to convert the correlation coefficient into a distance measure.

### 4.1.4 Feature-based similarity measures

A different approach to comparing time-series data is the feature based approach. The positive side of these approaches is that they convert the problem into lower dimensional space. On the other hand, the feature based approaches are dependent on the specific domain of the problem or the application and, as opposed to the approaches that consider whole time series, cannot be used for wide range of problems.

For example the authors in [38] consider representing the time series as a set of features that are commonly used to describe biological signaling networks combined with a set of statistical and oscillation related features. They compare different simulation trajectories produced by a single model using different parameter values which allow for emergence of different behaviors.

In [35] the authors consider the use of simulation based numerical calculation of the Lyapunov exponents as features of oscillatory and chaotic
dynamical systems. They try to estimate the parameters based on the maximization of the feature values, which requires comparison between models with different parameter values. A representation based on the features extracted by a discrete Fourier transformation [1, 12] and features extracted by wavelet transformation [32] of time series have also been considered in the literature.

4.2 Grouping behaviours of dynamic systems: Use of distance measures for hierarchical clustering

Hierarchical clustering methods differ by the approach used for creating clusters from data points. They can be divided into two groups: group of divisive approaches following a top-down method of creating clusters and a group of agglomerative approaches which use the bottom-up method for creating clusters.

The agglomerative approach starts by putting every object in a separate cluster. In the next steps it iteratively merges clusters until all objects belong in one cluster or other stopping criteria are met. The resulting hierarchy and the different clusters produced in each iteration can be illustrated using a dendrogram (Figure 10).

![Figure 10: A dendrogram illustrating hierarchical clustering of objects.](image)

In the dendrogram, the root node represents the final cluster to which all objects belong. The leaves of the tree represent the initial single object clusters. All internal nodes are the result of the iterative merging of similar clusters. The merging is based on the similarity between the clusters which in turn depends on the distance between the objects belonging to the different clusters. The heights of the branches in the dendrogram represent the distances between the merged clusters.

In each iteration, the most similar clusters are merged. The measure of similarity between the clusters depends on the linkage technique being used.
Commonly used linkages between two sets of data are the single-linkage and the complete-linkage techniques. The single-linkage technique measures the similarity between two clusters \( c_1 \) and \( c_2 \) as the similarity of the closest pair \((a,b)\) of data such as:

\[
D(c_1, c_2) = \min\{d(a, b) : a \in c_1, b \in c_2\} \quad (14)
\]

On the other hand the complete-linkage algorithm measures the similarity between two clusters as the similarity of the farthest pair of data:

\[
D(c_1, c_2) = \max\{d(a, b) : a \in c_1, b \in c_2\} \quad (15)
\]

A similar but computationally more expensive linkage algorithm is the average-linkage algorithm which as a measure of similarity between two clusters takes the average distance between all pairs of elements, one from each cluster:

\[
D(c_1, c_2) = \frac{1}{|c_1||c_2|} \sum_{a \in c_1} \sum_{b \in c_2} d(a, b) \quad (16)
\]

### 4.2.1 Clustering different model structures, i.e., trajectories produced by simulating them

As our first example we consider the task of clustering different model structures obtained in the process of automated data-driven process based modeling with ProBMoT [9] of the aquatic ecosystem of Lake Bled. The models in question are ordinary differential equations (ODEs). These describe the trajectories of phytoplankton biomass, which are the time series that we cluster.

The task of modeling the system includes a search through space of possible model structures defined in a conceptual model of the system using fragments of possible processes defined in a library of model fragments. In the process of modeling, the concentration of phytoplankton in the lake for the year 1999 was selected as a target variable that needed to be modeled. Every candidate structure that is obtained in the structure identification phase is then fitted to a set of real measurements. In the fitting process every parameter of the structure is estimated using a meta-heuristic parameter estimation method, namely the differential ant-stigmergy algorithm [23].

Since each model structure is fitted to the same data, we cannot expect a large range of different behaviors. We chose to perform agglomerative hierarchical clustering on 1000 trajectories obtained by simulation of different randomly chosen ODE model structures. The simulated trajectory of the best model from the randomly selected model structures can be seen in Figure
11. The Euclidean distance and DTW were chosen as representatives of non-elastic and elastic distance measures to be used in the process of clustering. Complete-linkage was used for merging the different clusters at each level.

By looking at the dendrograms we decided to cut the hierarchy at the level where 10 clusters are formed. Each of the 10 clusters contains examples that are similar to each other but different from the examples contained in the other clusters. By selecting one example from each obtained cluster we created an ensemble. The prediction of the ensemble is obtained by aggregating the simulated trajectories, in particular their average is taken.

We created ensembles from both hierarchies obtained by clustering with the different distance measures. Additionally, for the purpose of comparison, we created an ensemble from the 10 best model structures from the space of randomly sampled model structures. We compared the fit of the ensembles obtained in this manner to the measured data and the best sampled model. The trajectories from the measured data and the trajectories of the best model and the ensembles can be seen in Figure 12.

The trajectories seem to be very close to one another but since for the process of fitting the models in the process of automated modeling the sum of squared errors between the simulated trajectory and the measurements...
We compared the fit of the ensembles obtained in this manner to the measured data and the best sampled model. The trajectories from the measured data and the trajectories of the best model and the ensembles can be seen in Figure 6.

The trajectories seem to be very close to one another but since for the process of fitting the models in the process of automated modeling the sum of squared errors between the simulated trajectory and the measurements was used, we also compare the goodness of fit using this metric. The results can be seen in Table 1.

From the results, it can be seen that combining the trajectories from the best models produces better fit than combining the trajectories of the diverse model. Neither ensemble managed to produce better fit than the best individual model from the sample of 1000 trajectories. On the other hand, a significant improvement of the fit can be observed when multiple trajectories are combined into an ensemble, especially when combining diverse models. The performance of the ensemble built by combining the best models is worse than the other ensembles as a result of the trajectories having very similar

<table>
<thead>
<tr>
<th>Model type</th>
<th>SSE</th>
<th>Average SSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensemble (best models)</td>
<td>97.8252</td>
<td>103.2462</td>
</tr>
<tr>
<td>Ensemble (clustering with non-elastic measure)</td>
<td>137.0748</td>
<td>244.2723</td>
</tr>
<tr>
<td>Ensemble (clustering with elastic measure)</td>
<td>128.3303</td>
<td>196.0615</td>
</tr>
<tr>
<td>Best model (automated modeling)</td>
<td>84.97481</td>
<td>N.A</td>
</tr>
</tbody>
</table>

Figure 12: Measured data and trajectories of the best model found by automated modeling, ensemble using clustering by using elastic measures, ensemble using clustering by using non-elastic measures and an ensemble of the best 9 models.
behaviors. By looking at the obtained clusters it can be seen that in the case of clustering with the use of elastic measure, all of the 10 best models belong to the same cluster. In the case where the clustering was performed based on non-elastic measures 8 out of 10 best models belonged to the same cluster.

4.2.2 Clustering observed behaviours (different datasets of measurements)

For the next example, we consider analyzing eight measurements datasets for the lake Bled aquatic ecosystem. The available data contains separate measurements for 8 consecutive years (1995-2002). For each year the data consists of measurements for 30 different variables that represent different environmental conditions, different nutrients, different populations of producer species and different consumer species [2]. By performing agglomerative hierarchical clustering we obtained the dendrograms shown in Figure 13. Different distance measures were used as measures of similarity between the separate variables in the datasets: Manhattan distance, Euclidean distance, DTW, EDR, LCSS and ERP. The measure of similarity between the different datasets was obtained by summing the distances between the corresponding variables. A complete-linkage has been used during the merging of the separate clusters.

It can be seen the data being analyzed are very complex. However, several conclusions can be drawn from observing the dendrograms. The use of different distance measures in the process of clustering captures different aspects of the data. It can be clearly seen that the data from years ?95 and ?96 are very similar and they always form a cluster, no matter which distance measure is used for comparison, which may be indicative that the data from these years the best potential candidate for considering if dealing with modeling a predictable system. The same can be observed for the data from years ?00 and ?02. Only in the dendrogram obtained by using ERP as a distance measure the two datasets are not combined in a same cluster. However, looking at the ERP dendrogram it can be seen that it is highly unbalanced which shows that ERP as a measure did not perform well on the data. Looking at the dendrograms produced when a non-elastic measure is used at one hand and the dendrogram produced when an elastic measure is used at other it can be concluded that there is no significant time shifting in the data and for further analysis performed on the same data, in order to reduce the complexity of the calculation when comparing data, a non-elastic measure can be used as a measure of similarity between the data.

Exactly the same clustering is produced when during the clustering as a distance measure the Manhattan distance and the LCSS distance are used.
managed to produce better fit that the best individual model from the sample of 1000 trajectories. On the other hand, a significant improvement of the fit can be observed when multiple trajectories are combined into an ensemble, especially when combining diverse models. The performance of the ensemble built by combining the best models is worse than the other ensembles as a result of the trajectories having very similar behaviors. By looking at the obtained clusters it can be seen that in the case of clustering with the use of elastic measure, all of the 10 best models belong to the same cluster. In the case where the clustering was performed based on non-elastic measures 8 out of 10 best models belonged to the same cluster.

5.2. Analysis of datasets of measurements

For the next example, we consider analyzing eight measurements datasets for the lake Bled aquatic ecosystem. The available data contains separate measurements for 8 consecutive years (1995-2002). For each year the data consists of measurements for 30 different variables that represent different environmental conditions, different nutrients, different populations of producer species and different consumer species [16]. By performing agglomerative hierarchical clustering we obtained the dendrograms shown in Figure 4.

Different distance measures were used as measures of similarity between the separate variables in the datasets: Manhattan distance, Euclidean distance, DTW, EDR, LCSS and ERP. The measure of similarity between the different datasets was obtained by summing the distances between the corresponding variables. A complete-linkage has been used during the merging of the separate clusters.

Figure 13: Dendrograms representing the hierarchical clustering of the datasets. Different distance measures were used in the process of clustering: a) Manhattan distance, b) Euclidean distance, c) DTW, d) EDR, e) LCSS, f) ERP.

Similar clusters are obtained when Euclidean distance, DTW and EDR are used. In Figure 14 the hierarchy produced by the Manhattan distance and LCSS together with the trajectories of 3 variables can be seen. In Figure 15 we show one hierarchy (EDR) of the other group of similar hierarchies.
Figure 14: Hierarchy of datasets of measurements obtained by clustering using Manhattan distance.
Figure 15: Hierarchy of datasets of measurements obtained by clustering using EDR distance.
4.3 Analysis of the clustering results

In this section, we presented a method for agglomerative hierarchical clustering which uses different distance measures for clustering analysis of univariate and multivariate time-series. We showed the possible uses of the method in two examples: (1) clustering of simulations of dynamical models that are structurally different in order to build ensemble of trajectories which will improve the goodness of fit to the measured data and, (2) analysis of datasets of measurements of aquatic ecosystem.

We concluded that simulated trajectories of dynamical behavior can be clustered and combined to improve the goodness of fit. We also concluded that building ensembles of models might improve the fit to the data. Significant improvement can be achieved when constructing ensembles from diverse models as opposed to the construction of ensembles of similar models.

For the example of analysis of different datasets of measurements we concluded that the different distance measures used for clustering captured different aspects of the dynamics. This is reflected in the hierarchy of the obtained clusters. The conclusions of the analysis can be further referred by taking into account that the data is used for different tasks.

As further work, given these preliminary findings, we can consider analyzing the predictive behavior of an ensemble of model structures. Different model structures can be learned on a certain dataset. We can then test the predictive abilities of a certain model or ensemble of models on a similar dataset. Furthermore, given simulations of fitted model structures of a dynamical system in which more than one variable is being modeled, we can further analyze the possibility of combining diverse behaviors in order to improve the goodness of fit or analyze the predictive abilities of the ensembles of trajectories of multiple variables.
5 D7.3: Report on learning to interconnect ODE models

To evaluate the performance of the methods for generating a diverse set of ODE models (presented in the Deliverable 7.1) and methods for measuring the diversity and selecting a complementary set of ODE models (presented in the Deliverable 7.2), we have to integrate them into approaches for learning ensembles. These approaches are based on complex combinations and interactions between the three basic components for generating, selecting, and combining the components of the ensembles. Note that we can employ different methods for coupling or interconnecting the component ODE models into supermodels.

In this deliverable, we first present the results of some preliminary experiments with learning ensembles of ODE models. The preliminary experiments are based on approaches that take ad-hoc combinations of the components presented in D7.1 and D7.2 and use averaging to aggregate the component model simulations into the simulation of the supermodel ensemble. In the second part of the deliverable, we outline the plans for further work within the Task 7.3.

5.1 Preliminary experiments with learning ensembles of ODEs

We performed several experiments to evaluate and compare the performance of the ensembles obtained with different methods for sampling data instances. We consider the following four sampling methods: (1) simple bootstrap sampling with replacements (Bagging); (2) simple bootstrap sampling without replacements (random samples); (3) continuous-time bootstrap sampling without replacements (window samples, i.e., samples in continuous temporal windows); and (4) error-dependent sampling as used in boosting [34].

Each model in the ensemble is obtained by re-fitting the values of the constant parameters of an ODE model of phytoplankton growth in the Lake Bled. The structure of the model, obtained by using the automated modeling tool ProBMoT[9], is as follows:
\[
\frac{d{phyto}}{dt} = {phyto} \cdot 0.55 \cdot \frac{ps}{ps + 0.033} \cdot \frac{silica^2}{silica^2 + 0.02} \cdot \frac{no}{no + 0.073} \\
\cdot 1.11^{temp-18.5} \cdot \frac{light}{140} \cdot e^{(\frac{light}{140})} - {phyto} \cdot 0.092 \cdot \frac{temp - 0.4}{19.4 - 1.5} \\
- {phyto} \cdot \frac{0.09}{10} \cdot 1.11^{temp-15.3} - daph \cdot 10^{-5} \cdot \frac{temp}{15 - 3.4} \\
\cdot \frac{{phyto}}{phyto + 2.1} \cdot {phyto} \cdot 0.37
\]

The data samples are taken from the eight data sets of data corresponding to the eight years of measurements in the period from 1995 to 2002. In each data set, a number of variables are being measured: water temperature, light intensity, concentrations of different nutrients, and the phytoplankton concentration (\(phyto\)) that is considered to be a target modeling variable. Figure 16 depicts the data on phytoplankon concentration.

Figure 16: Phytoplankton growth measurements for years 1995–2002.

In Figure 17 we present the overall process of sampling data and estimating the ODE model parameters on each data sample. In the diagram, \(v_1\sim v_m\) denote the variables of the systems of ODE (there of having a multivariate problem), and \(f(v, t)\) is the objective function of variable \(v\) in time-point \(t\).

For this task, we are using multi-objective differential evolution algorithm, introduced in Opt4j\(^1\), a framework for applying methods for arbitrary

\(^1\)http://opt4j.sourceforge.net/
After obtaining the model structure from the ProBMoT, we perform additional parameter estimation over the model structure over different samples of the data, therefore obtaining several model candidates. The classical approach of parameter estimation is presented in Table 1, where \( v_1 \) … \( v_m \) are the variables of the systems of ODE (there of having a multivariate problem) , and \( f(v, t) \) is the objective function of variable \( v \) in time-point \( t \).

Table 1. Parameter estimation by minimizing objective function

| \( t_0 \) | \( f(v_1, t_0) \) | \( f(v_1, t_0) \) | \( f(v_m, t_0) \) |
| \( t_1 \) | \( f(v_1, t_1) \) | \( f(v_2, t_1) \) | \( f(v_m, t_1) \) |
| \( t_2 \) | \( f(v_1, t_2) \) | \( f(v_2, t_2) \) | \( f(v_m, t_2) \) |
| \[ \sum_{i=0}^{n} f(v_1, t_i) \] | \[ \sum_{i=0}^{n} f(v_2, t_i) \] | \[ \sum_{i=0}^{n} f(v_m, t_i) \] |

Figure 17: Parameter estimation by minimizing objective function.

5.1.1 Experiments with bootstrap sampling with replacement

For this experiment we also started with the best model of phytoplankton growth obtained by ProBMoT, and performed the bagging method with replacements. The parameters were refitted for every bootstrap sample of the data (50 samples in total) using the differential evolution algorithm, with 100 generations and 50 agents, and sum of squared errors as an objective function. In Figure 18 are presented the trajectories of every sample. The individual sample simulations were aggregated using two different methods: simple average and weighted average. The weights in the weighted average method were calculated using minmax normalization in range 0-1 of the reported errors by sample. The error values were rescaled to values between 0 and 1.

\[
x_i = \frac{x_i - \min_{k=1..n} x_k}{\max_{k=1..n} x_k - \min_{k=1..n} x_k} \tag{17}
\]

From this (Figure 19), we can conclude that the bagging method clearly outperforms the single best model reported by the ProBMoT tool. As long as
the different averaging techniques, we can see imperceptibly small difference in favor of the weighted average. The difference is small, due to the many samples with great imperfection weights vis-a-vis five-six respectable models, which in the end results in a simple average.

5.1.2 Experiments with bootstrap sampling w/out replacement

For this experiment we used the same model structure obtained by the ProBMoT, and refitted the parameters with DE with 100 generations and 50 agents. For the training data we used 25 bootstrap samples with capacity of 50% of the whole amount. After the obtaining the results, we performed simple average over the simulations of the individual models (Figure 20).

We can conclude from Figure 21, the bagging method again outperforms the best model generated by the ProBMoT. As long as the sample replacement is considered, we can see that the method which uses replacements is slightly better that the method without replacements. Even though, the models were fitted with different samples, the outcomes are nearly identical.
For this experiment we used the best model structure obtained by ProBMoT, and performed the bagging method with replacements. The parameters were refitted for every bootstrap sample of the data (50 samples in total) using the differential evolution algorithm \([DE]\), with 100 generations and 50 agents, and sum of squared errors as an objective function.

Figure 18: Simulations of 50 bootstraps.

Figure 19: Average v.s. Weighted Average.
We can conclude from Figure 7, the bagging method again outperforms the best model generated by the ProBMoT. As long as the sample replacement is considered, we can see that the method which uses replacements is slightly better than the method without replacements. Even though the models were fitted with different samples, the outcomes are nearly identical.

Figure 20: Bootstrap sampling without replacements.

Figure 21: Comparison of bootstrap sampling with and without replacement.
5.1.3 Random window samples (and experiments)

Similarly as in the previous experiments, the same model structure and parameter estimation algorithm for refitting were used. The bootstrap sample windows (10 in total) were generated by using different continuous chunks of data for every sample, starting at a random point. Two different experiments were performed using 30% and 70% of the data, respectively; after which the sample simulations were averaged (Figure 22 and 23).

70% of the whole data can be considered as an upper threshold of the length of the sample, because choosing bigger can cause an overfit. Analogously, 30% can be considered as a lower threshold of the length of training data, because choosing smaller amounts cannot provide sufficient information.

To evaluate the performance of the boosting, we performed an experiment where the first boosting iteration was chosen as a baseline (with weight values 1), and for every following boosting iteration (10 in total) the weights were increased based on the past experience, i.e., the errors made by the previous model in every time point of the series. The reweighting was done linearly, by increasing the penalty of every weight by minmax\[0 \text{-} 1\] normalized value of the previous errors (Figure 10) or average of all previous errors (Figure 11), according to the sum of squared errors measure. The rest of the setup was same as in the previous examples.
length of the sample, because choosing bigger can cause an overfit. Analogously, 30% can be considered as a lower threshold of the length of training data, because choosing smaller amounts cannot provide sufficient information.

Figure 23: 70% (Bootstraps, up), (Averages, down).

5.1.4 Experiments with boosting of ODEs

To evaluate the performance of the boosting, we performed an experiment where the first boosting iteration was chosen as a baseline (with weight values
1), and for every following boosting iteration (10 in total) the weights were increased based on the past experience, i.e., the errors made by the previous model in every time point of the series. The re-weighting was done linearly, by increasing the penalty of every weight by minmax[0-1] normalized value of the previous errors (Figure 24) or average of all previous errors (Figure 25), according to the sum of squared errors measure. The rest of the set-up was same as in the previous examples.

In general, it can be clearly seen (Figure 26), that the boosting method outperforms the best model reported by the ProBMoT. Furthermore, the choice of a method for the re-weighting procedure is not significant. Even though, the method which re-weights with the average of all past experience is slightly better, computationally is more expensive than the other method.

Figure 24: Boosting (reweight from previous).
In general, it can be clearly seen (Fig. 12), that the boosting method outperforms the best model reported by the ProBMoT. Furthermore, the choice of a method for the reweighting procedure is not significant. Even though, the method which reweights with the average of all past experience is slightly better, computationally is more expensive than the other method.

Figure 25: Boosting (reweight from average of all previous).

Figure 26: Comparison of the two boosting approaches.
5.2 Plans for further work

The main limitation of the preliminary experiments presented above is that they all employ parameter estimation of a single ODE structure as a base learner instead of learning the structure and parameters of an ODE model with ProBMoT. This is due to the fact that the current ProBMoT implementation cannot be directly used in some of the learning ensembles approaches used in the experiments. In the remainder of this deliverable, we will identify the ProBMoT limitations and outline the plans for the further developments needed for learning ensembles of ODEs. We will also summarize the planned research within Task 7.3.

Most of the methods for generating a diverse set of ODE models presented in D7.1 can be implemented as a preprocessing step employed before running the base learner ProBMoT with the particular data or knowledge library sample at input. Note however, that the generalized sampling procedure requires adjustments of the objective function of ProBMoT used to evaluate the candidate ODE models. The ProBMoT objective function is based on squared error, i.e., the sum of squared differences between points in the observed trajectory provided as learning data points in the trajectory obtained with simulating the candidate model. All variables and time points in the trajectories have equal contribution to the error (i.e., the value of the objective function). The generalized sampling procedure introduces weights on examples and variables; to take these into account, we have to implement a new objective function based on weighted squared errors in ProBMoT.

Integrating the methods for measuring diversity of the generated ODE models presented in D7.2 into approaches to learning ensembles provides ample room for exploration. One can use the pair-wise similarity of the generated ODE models to filter out the ones that lead to highly correlated behaviors. Or we can use the presented clustering methods to select representatives of the generated clusters as component models of the ensemble. Scenarios like these can be compared to the basic scenario of using the whole set of generated diverse ODE models.

Finally, note the narrow focus of the preliminary experiments with the single method for combining the ODE models in the ensemble by averaging their simulations. In further experiments, we will follow the paradigm of supermodeling and tight ODE models coupling, where we simulate the component models of the ensemble simultaneously and combine their simulations on a time-step basis. To this end, we will develop new libraries of domain knowledge for ProBMoT, similar to the one presented in the D1.6. We will then learn supermodels, by learning both the component ODE models and the structure/parameters of the coupling functions.
6 Conclusion

The objective of WP7 is to develop methods for computational scientific discovery that can learn complete supermodels (ensembles of ODE models) of dynamical systems. The supermodels are expected to be built in three phases: generate diverse models, select a set of complementary models, and learn the interconnections between the constituent models of an ensemble. While this decomposition allows for independent development of the different components of the approaches to learning ensembles of ODE models, we can evaluate their performance only when we integrate them into complete approaches to learning ensembles. Thus, the issues related to finding working combinations of the methods developed and proposed within D7.1 and D7.2 are still open and are to be resolved within the task T7.3.

Task 7.1 is concerned with generating a diverse set of ODE models. To generate a diverse set of models, we adapt existing sampling approaches from the area of ensemble learning. The specificities of the task of subsampling an observed behavior of a dynamic system are taken into account. After different subsamples are generated, the base learner ProBMoT can be applied to learn different ODE models from them. The approaches adapted include subsampling the instance space and subsampling the feature space. Along the first dimension, we consider the selection of random sub-intervals of the observation period. We adapt bootstrap sampling and error-weighted sampling for the case of time-series data. Preliminary experiments with learning ODE models (same structure, different parameters) have been conducted with some of these adaptations within Task 7.3.

Along the second dimension, we consider random sub-sampling of the variable space, as in random subspaces. We consider combining it with bootstrap sampling of the instance space. Moreover, we propose a generalized sub-sampling approach, which simultaneously subsamples both the instance and the feature space. Finally, we also suggest to consider a procedure for sampling the template entities and processes from the library of background knowledge used by ProBMoT as a way to generate a set of diverse ODE models. Further experiments within Tasks 7.3 are needed to evaluate the performance of these methods in the context of learning supermodels.

Task 7.2 is concerned with selecting a set of complementary ODE models. To address this task, we first attempt to define the notion of diversity for ODE models. While model diversity has been extensively studied in the context of ensemble models, the bulk of work has considered the task of classification. For regression, co-variance is used as a measure of similarity/diversity of constituent models of an ensemble. To the best of our knowledge, the notion of diversity has not been studied in the context of ensembles of ODE models.
Our definition of diversity is based on similarity measures between dynamic system behaviors, i.e., multi-variate time series. These are based on similarity measures for single time-series. The similarities of corresponding time-series pairs in two behaviors are aggregated into an overall similarity. We consider a range of similarity measures for time-series, which we have collected and implemented. We explore their use in clustering of ODE models and observed behaviors of dynamic systems. These clustering experiments are important in the context of further work within Task 7.3, where a set of appropriate models has to be chosen for inclusion in the ensemble. After clustering a larger set of models, representative models from each cluster could be chosen for combination, for example.
References


50


