Jul 1st, 12:00 AM

Optimal modularization of learning models in forecasting environmental variables

Dimitri P. Solomatine

Optimal modularization of learning models in forecasting environmental variables

Dimitri P. Solomatine

UNESCO-IHE Institute for Water Education, Delft, The Netherlands, sol at *ihe.nl*

Abstract:- Data-driven models based on the methods of machine learning have proven to be accurate tools in predicting various natural phenomena. Their accuracy can be however increased if several learning models are combined in an ensemble or a committee. Modular model is a particular type of a committee machine and is comprised of a set of specialized (local) models each of which is responsible for a particular region of the input space, and may be trained on a subset of the training set. This paper presents a number of approaches to building modular models. An issue of including a domain expert into the modelling process is also discussed, and the new algorithms in the class of model trees (piece-wise linear modular regression models) are presented. Comparison of the algorithms based on modular local modelling to the more traditional "global" learning models shows their higher accuracy and the transparency of the resulting models.

Keywords: local models, modular models, committees, neural networks, flood forecasting

1. INTRODUCTION

Natural phenomena are typically multi-stationary and are composed of a number of interacting processes. Their modelling assuming the existence of one single ("global") model handling all processes often suffers from inaccuracies. Various processes can be modeled separately by different models optimized to represent every single process. In case of data-driven models (for example, neural networks) the training set is split into a number of subsets, and separate models are trained on these subsets, or in other words, the input space can be divided into a number of subspaces or regions for each of which a separate specialized model is built. These models are called local, or expert models, or experts (not to confuse with the human domain experts). Such *modular model* (MM) can be also attributed to a class of *committee machines* (CM) [Haykin 1999]. In machine learning using several models instead of one is often termed "combining classifiers" [Kuncheva 2004].

This paper presents a number of ways to optimize the process of building modular models, and addresses the problem of incorporating more domain knowledge into such models. Comparative analysis of the modular models performance in solving water flow forecasting problems is reported as well.

2. METHODS OF PARTITIONING DATA AND COMBINING MODELS

In the process of building, training and using a MM, two decisions have to be made: (A) which module should receive which training pattern (partitioning problem), and (B) how the outputs of the modules should be combined to form the output of the final output of the system (combining problem). Accordingly, two decision units have to be built, or one unit performing both functions. Such unit is called an integrating unit, or in case of using ANN, a gating network. Note that the functioning of units A and B could be different during training and operation. Figure 1 illustrates the principle.

It is possible to classify the MMs with respect to the way partitioning/combining is performed:

- hard partitioning followed by training MMs, of which only one is used to predict the output for a new input vector;
- hard partitioning followed by training MMs which outputs are combined by "soft" weighting scheme (leading, for example, to *fuzzy committees*);
- statistically-driven soft partitioning, used in *mixtures*, stacked generalizers and *boosting*;
- no-split option leading to *ensembles*; the models are trained on the whole training set and their outputs are combined using a weighting scheme where the model weights typically depend on model accuracy;
- instance-based learning constructing a local approximation to the modeled function that applies well in the immediate neighborhood of the new query instance encountered (k-NN and local weighted regression methods).

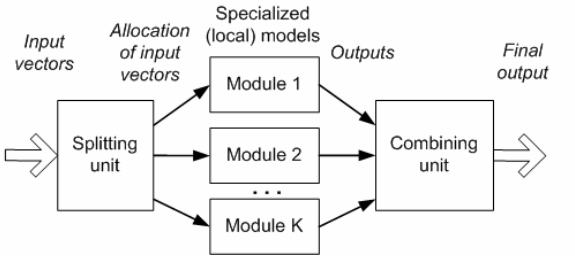

Figure 1. Modular modeling.

2.1 Statistical soft partitioning

Statistically-driven approaches with "soft" splits of input space are represented by *mixtures of experts* [Jacobs et al., 1991], stacked generalizers [Wolpert 1992], *bagging* [Breiman 1996] and *boosting* represented by a popular boosting algorithm *AdaBoost* [Freund and Schapire, 1997]. A new version of AdaBoost for regression is *AdaBoost.RT* by Solomatine and Shrestha [2004, 2006].

2.2 Hard partitioning

Training set is partitioned into subsets (the input space can be partitioned accordingly), and for each of them an individual local expert model is trained. In operation, the input vector is processed by one of the models and its output becomes the output of the overall models.

2.3 Hard partitioning with soft combination of models (fuzzy committees)

Hard partitioning leads to a problem of compatibility at the boundaries between partitions. This issue calls for the introduction of smooth weighting schemes to combine outputs of the local expert models. One of the ways is of course use the statistical approaches mentioned above. It is however possible to combine hard and soft partitioning by introducing a more transparent combining scheme.

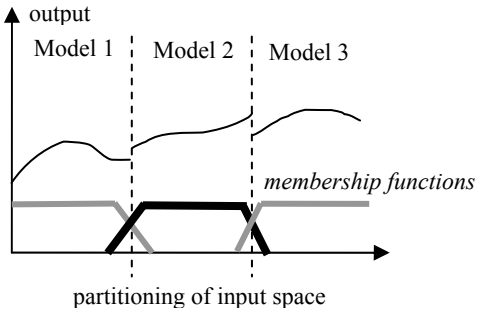

Figure. 2. MMs with hard partitioning have incompatibilities when switching between models. A solution is to assign models the membership functions and implementing a fuzzy committee.

This can be done of weighting the relevant model outputs for the input vectors that are close to the boundary. Such weighting can be also formulated with the use of fuzzy logic and the following framework of a *fuzzy committee* can be proposed (Fig. 2):

1. Perform hard partitioning of the training set into subsets and of the input space into regions.

2. Train the individual local models for each subset.

3. For each example, assign the values of membership functions (degree of belonging) corresponding to each model. The functions should be constant over the "central" part of the region, starting to decrease in the proximity of the region boundary, and decreasing to zero beyond the boundary; an option is to use a simple trapezoidal shape.

4. For the new input vector, calculate the output value as the combination of the outputs of all models weighted by the corresponding membership function values.

The weighted combination of global models is quite widely used, but the combination of local models is less common – see, e.g., [Kasabov and Song, 2002]. The presented framework would allow for combining the advantage of local modeling with the smooth combination of models at the boundaries between the regions. The shapes of the membership

functions have to be optimized, for example by minimizing the overall model error.

Note that the introduced principle of the fuzzy committee approach is to address the problem of "fitting" the local models and in this respect it differs from the combination of classifiers based on the fuzzy integral [Kuncheva 2004].

3. OPTIMIZATION OF PARTITIONING

In a number of method s "hard" splits of input space into regions is performed progressively, narrowing the regions of the input space. The result is a hierarchy, a tree (often a binary one) with splitting rules in non-terminal nodes and the expert models in leaves (Fig. 3).

3.1 Partitioning by greedy optimization

For solving numerical prediction (regression) problem, there is a number of methods that are based on the idea of building a MM. Each model could be of any type, for example linear regression or ANN. One is a regression tree by Breiman et al. [1984], where the input space is progressively partitioned into subsets by hyperplanes x_i=A (where x_i is one of the model inputs, and i and A are chosen by exhaustive search). A leave is associated with an average output value of the instances sorted down to it (zero-order model). Another method is MARS (multiple adaptive regression splines) by Friedman [1991] implemented as MARS software. A method, ideologically close to regression tree is a model tree, where leaves have linear (first-order) regression functions of the input variables. A method to build such trees is M5 by Quinlan [1992] implemented in Cubist software and, with some changes, in Weka software [Witten and Frank, 2000].

Note, that all these algorithms are *greedy* since after the decision about the partitioning into regions is made, it is not changed any more (regions however may be merged).

An approach used in regression and M5 trees is to minimize the intra-subset variation in the output values down each branch. In each node, the standard deviation of the output values for the examples reaching a node is taken as a measure of the error of this node and calculating the expected reduction in error as a result of testing each input attribute x_i and possible split values A. Such split attribute together with the split value that maximize the expected error reduction are chosen for each node. The splitting process will terminate if the output values of all the instances that reach the node vary only slightly or only a few instances remain.

After the initial tree has been grown, the regression models are generated (0- or 1-order). In M5 algorithm the 1-order (linear) models are, possibly, simplified and smoothed, and the tree is pruned. Wang and Witten [1997] reported M5′ algorithm, a version of the original M5 algorithm.

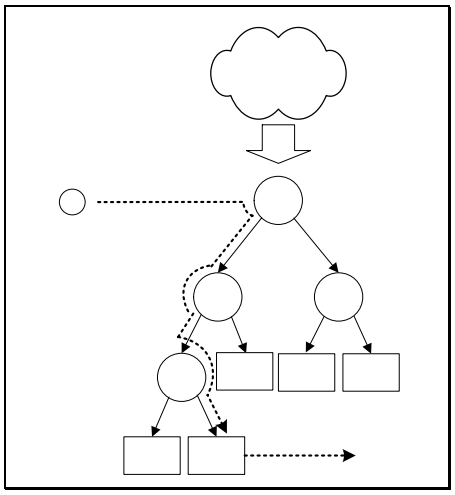

Fig. 3. Progressive partitioning of input space leading to local models.

3.2 Better ways to optimize

A number of researchers aimed at improving the predictive accuracy of a tree-based model, however they dealt mostly with decision trees for classification and with greedy approaches; see [Solomatine and Siek 2004a, 2006] for an overview. A rare exception of using a non-greedy approach for constructing decision trees is [Bennett 1997].

The problem of optimizing construction of tree-like regression methods like M5 tree is, however, addressed very little. Solomatine and Siek [2004a, 2006] presented a semi-greedy compromise approach where the top part of the tree is subject to the full-fledge optimization by exhaustive or randomized search, and the rest of the tree is built using fast greedy approach. The problem of building a MM is posed in a general way ensuring that the error of the resulting overall model is minimal among all possible configurations:

$$Find\ such\ M_{opt}\ that\ E(M_{opt}) = min,\ M \subset \{M_k\} \quad (1)$$

where M_{opt} is a model with optimal configuration, $\{M_k\}$ is a set of all possible model configurations, E

is the model error. For the purpose of this paper it will be assumed that M is a MM consisting of a number of individual models M_i.

The solution can be found by exhaustive search through the space of all models, but for practical problems the computational time would be too high. In order to avoid solving an overall hard optimization problem the generation of a tree can be done into two steps:

1. *Global optimisation*. Generate upper layers of the tree (from the 1st layer) by a global (multi-extremum) optimization algorithm (better-than-greedy), e.g. GA, or algorithms implemented in [GLOBE 2006];

2. *Greedy search*. Generate the rest of the tree (lower layers) by a faster greedy algorithm originally proposed by Quinlan [1992] and implemented in Weka software [Witten and Frank, 2000].

The layer up to which global optimization is applied could be different in different branches, but it would be reasonable to fix it at some value for all branches; in this case it will be denoted as L. This allows for a flexible trade-off between speed and optimality.

One of the implementations of the algorithmic approach presented above is the *M5opt* algorithm that builds M5 models in the leaves.

M5opt algorithm differs from M5 that it is able to build a linear model for the instances that reach a node, directly in the initial model tree which increases the model accuracy. A better version of pruning (called compacting) is implemented as well [Solomatine and Siek 2004a, 2006].

4. MODELS INVOLVING DOMAIN EXPERT

Incorporation of domain knowledge into the model building process is an important problem. A typical machine learning algorithm minimizes the training (cross-validation) error seeing it as the ultimate indicator of the algorithms performance, so is purely data-driven. Domain experts, however, may have additional consideration in judging the quality of the model, and want to have certain control over the decisions (A) and (B) on Fig. 1 and over the choice of models used in each unit. These models could be not only data-driven ones based on machine learning, but also process (behavioral) models based on the description of the underlying physical processes.

The challenge is to build hybrid models and to integrate the background domain knowledge into a machine learning algorithm by allowing the user to determine some important structural properties of the model based on the physical insight, and leaving more tedious tasks to machine learning.

A straightforward possibility to include a domain expert is to make it possible for him/her to construct the rules performing the hard splits of the input space, and, accordingly, of the training set. For various machine learning algorithms such possibility may have different forms. We developed and implemented a version of M5 algorithm, *M5flex*, that introduces a domain expert into making a decision about the splits at each node [Solomatine and Siek 2004b].

Some approaches give the user opportunity to choose the split attribute and value for each node, e.g. Ware et al. [2000] introduced visual decision tree (C4.5) construction using 2D polygons. Techniques for building interactively the model trees seem to be missing.

The proposed M5flex method enables the user to determine split attributes and values in some parts of important (top-most) nodes, and then the M5 machine learning algorithm takes care of the remainder of the model tree building. In the context of flood prediction, for example, the expert user can instruct the M5flex to separate the low flow and high flow conditions to be modeled separately. Such models can be more suitable for hydrological applications than ANNs or standard M5 model trees.

5. EXPERIMENTS

Some of the presented approaches were tested on the hydrological data sets of Sieve catchment (Italy) with the hourly rainfall and runoff [Solomatine and Dulal 2003], three hydrological data sets of Bagmati catchment (Nepal) with the daily rainfall and runoffs, and five widely used benchmark data sets (Autompg, Bodyfat, CPU, Friedman and Housing [Blake and Mertz 2006] were employed. Five methods were used: a global method (MLP ANN) and local modelling methods (M5′, M5opt and M5flex). M5flex which was used applied only for 6 hydrological data sets.

The problem associated with hydrological data sets is to predict runoff (Q_{t+i}) several hours ahead with respect to previous runoff ($Q_{t-\tau}$) and effective rainfall ($RE_{t-\tau}$). Before building a prediction model, it was necessary to analyze the physical characteristics of the catchment and then to select the input and output variables by analyzing the inter-dependencies

between variables and the lags τ using correlation analysis. Finally the following 3 models were built:

$$Q_{t+1} = f(RE_t, RE_{t-1}, RE_{t-2}, ..., RE_{t-5}, Q_t, Q_{t-1}, Q_{t-2})$$

$$Q_{t+3} = f(RE_t, RE_{t-1}, RE_{t-2}, RE_{t-3}, Q_t, Q_{t-1})$$

$$Q_{t+6} = f(RE_t, Q_t)$$

In Bagmati case study the following model was used:

$$Q_{t+1} = f(RE_t, RE_{t-1}, RE_{t-2}, Q_t, Q_{t-1}).$$

Global modeling with ANN. ANNs were built using *NeuroSolutions* [2006] and *NeuralMachine* software [2006]. For Sieve case study, the best network appeared to be a three-layered perceptron (MLP) with 18 hidden nodes and hyperbolic tangent as activation functions. The stopping criteria was either mean squared error in training reaching the threshold of 0.0001 or the number of epochs reaching 5000.

Local modelling with M5′ and M5opt. *M5′ model trees* were built based on default parameter values, employing pruning and smoothing; the same settings were also used in M5flex and M5opt experiments.

Local modelling with domain expert (M5flex). The user could to modify the split attributes and values in each node only in the first and the second level of model tree; this limitation was applied to reduce the complexity that the domain expert would face. The split values that were used in experiments were points around extreme values (minimum and maximum), mean and some trials were needed to find the best model tree.

Partitioning by experts, followed by global and local modeling. In Bagmati case study the data set was also separated into high flows and low flows with 300 m^3/s as division point. For each subset separate models of all the types mentioned above were built.

TABLE 1. PERFORMANCE OF M5′, M5FLEX, M5OPT AND ANN FOR HYDROLOGICAL DATA SETS (RMSE)

	Data Sets	ANN		M5'		M5flex		M5opt	
		Training	Verif..	Training	Verif.	Train	Verif	Train	Verif
Sieve	Q_{t+1}	7.946	8.476	4.550	3.612	4.547	3.037	4.614	3.110
	Q_{t+3}	15.176	12.762	13.089	13.674	15.150	11.806	14.228	11.816
	Q_{t+6}	28.897	20.233	26.469	22.894	28.681	19.705	28.596	19.353
Bagmati	All	97.335	158.405	93.607	152.759	93.625	121.548	99.260	145.453
	High	224.630	173.496	249.252	187.334	221.750	161.394	222.150	178.705
	Low	31.920	29.262	30.029	31.183	30.807	28.851	30.439	30.781

TABLE 2
SCORING MATRIX FOR ALL VERIFICATION DATA SETS

	ANN	M5'	M5opt
ANN	0	2.428	13.871
M5'	-2.428	0	13.914
M5opt	-13.871	-13.914	0
Total	-16.299	-11.487	**27.785**

6. RESULTS AND DISCUSSION

A so-called scoring matrix *SM* was used to present the results in a comparative way. This is a square matrix with the element $SM_{i,j}$ representing the average of relative performance of algorithm *i* compare to algorithm *j* with respect to all data sets used [Solomatine and Shrestha, 2004]

The overall experimental results are summarized in Table 1 and Table 2. M5opt model trees were most accurate on the mostdata sets, but for Bagmati-High, Bagmati-Low, Autompg, and Friedman data sets the best accuracy was given by ANN models.

The experiments dealing with all algorithms for Sieve and Bagmati data sets showed that M5flex model trees had the best accuracy on most of these data sets, except Sieve Q_{t+6} data set where M5opt model was a bit better.

M5flex showed the best score on all hydrologic data sets, with M5opt following closely. The reason for high performance of M5flex is that it uses additional domain knowledge for determining the best split attributes and values. M5flex and M5opt also are better predictors of the flow peak values.

Some results of using yet another committee machine model, *AdaBoost.RT*, for the same data sets are presented by Shrestha and Solomatine [2006]. The overall performance of this method tested on the 5 data sets (Sieve Q_{t+3}, Sieve Q_{t+6}, Auto-Mpg, CPU and Housing,) was the highest if compared to M5′,

Bagging, ANN and AdaBoost.R2. On some of the sets, M5', however, was the most accurate method.

It can be noted that in a number of cases the methods have very close performance so that it is not possible to speak about the statistically significant superiority of one method over another.

7. CONCLUSION

The following can be concluded.

- the modular models allow for building the accurate specialized models that can capture the details of the processes characteristic attributed to certain regions of the input space;
- local models often can be made simpler than global ones, and thus more transparent, understandable and reproducible by decision makers. An example is M5 model tree where local models are linear;
- M5opt, an optimized version of M5 model tree algorithm, showed the higher performance;
- in the problems of forecasting natural phenomena, the incorporation of domain knowledge into modular modeling allows for more accurate account for details about the process (behavior) of the modeled system. Such models are typically trusted more than purely data-driven models.

8. REFERENCES

Bennett, K.P.. Global tree optimization: a non-greedy decision tree algorithm. *Journal of Computing Science and Statistics, 26*, 156-160, 1994.

Blake, C.L., and Mertz, C.J. UCI Repository of machine learning databases. *www.ics.uci.edu/ ~mlearn/MLRepository.html*, 2006.

Breiman, L. Stacked regressor. *Machine Learning*, 24(1), 49-64, 1996.

Freund, Y., and Schapire, R. A decision-theoretic generalisation of on-line learning and an application of boosting. *J. of Computer and System Sci.*, 55 (1), 119-139, 1997.

Friedman, J.H. Multivariate adaptive regression splines. *Annals of Statistics, 19*, 1-141, 1991.

Haykin, S. *Neural networks*. Second edition, Prentice-Hall, 1999.

GLOBE: global and evolutionary optimization tool. *http://www.data-machine.com*, 2006.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E.. Adaptive mixtures of local experts. *Neural Computation, 3*, 79-87, 1991.

Kasabov, N.K. and Song, Q. DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time Series Prediction. *IEEE Trans. Fuzzy Systems*, 2, 144-154, 2002.

Kuncheva, L.I. *Combining Pattern Classifiers*. Wiley, NJ, 2004.

NeuroSolutions software. *http://www.nd.com*, 2006.

NeuralMachine software: a neural network tool. *http://www.data-machine.com*, 2006.

Quinlan, J.R. Learning with continuous classes. Proc. 5th Australian Joint Conf. on Artificial Intelligence, Adams, Sterling (eds.), World Scientific, Singapore, 343-348, 1992.

Shrestha D.L. and Solomatine, D.P. Experiments with AdaBoost.RT, an Improved Boosting Scheme for Regression. *Neural Computation,* 18(7), 2006.

Solomatine, D.P. and Shrestha, D.L. AdaBoost.RT: a boosting algorithm for regression problems. *Int. Joint. Conf on Neural Networks*, Budapest, Hungary, 2004.

Solomatine, D.P. and Siek, M.B. Semi-optimal hierarchical regression models and ANNs. *Int. Joint. Conf on Neural Networks*, Budapest, Hungary, 2004a.

Solomatine, D.P. and Siek, M.B. Flexible and optimal M5 model trees with applications to flow predictions. *Proc. 6th Int. Conference on Hydroinformatics*, World Scientific, Singapore, 2004b.

Solomatine, D.P. and Siek, M.B. Modular Learning Models in Forecasting Natural Phenomena. *Neural Networks*, March, 2006.

Wang, Y., and Witten, I.H. Induction of model trees for predicting continuous classes. Proceedings of the European Conference on Machine Learning, Prague, 128-137, 1997.

Ware, M., Frank, E., Holmes, G., Hall, M. and Witten, I.H. Interactive machine learning - letting users build classifiers. *Int. J. on Human-Computer Studies*, 2000.

Witten, I.H. and Frank, E. *Data Mining*. Morgan Kaufmann Publishers, 2000.

Wolpert, D.H. Stacked generalization. *Neural Networks*, 5(2), 241-259, 1992.