Fusion Algorithm Based on Fuzzy Neural Networks

Alexey Natekin
fortiss Gmbh
Guericksr. 25
Munich, Germany
natekin@fortiss.org

Alois Knoll
Technical University Munich
Boltzmannstr.3
Garching, Germany
knoll@in.tum.de

***Abstract*—The problem of optimal fusion of several predictive machine learning regression models is considered. The method of combining different predictive models based on additive fuzzy systems is presented. The framework of model fusion based on fuzzy neural networks is described and the appropriate algorithms are derived. Learning process justifications and the requirement of the separate fusion set are discussed. The presented models are supported with the real world application example of robotic hand control.**

***Index Terms*—machine learning; data fusion; model fusion; fuzzy associative memory; predictive ensembles**

I. Introduction

Data fusion is a wide branch of methods, used in a number of different practical applications [1]. In machine-learning the most common data fusion problem considered, is how to blend, or fuse, several predictive models. One of the most notable examples of such applications was described in the winning solution of the Netflix Grand Prize [2]. Authors carefully applied and tuned a large number of different predictive models and blended them together in the form of the linear combination. That same approach of model blending has become popular among practitioners, and currently is one of the most frequently used winning strategies in various online machine learning competitions like kaggle [3], [4].

However, linear blending has it's shortcomings. This model simplicity doesn't allow one to exploit the model specialization. In particular, if one has a set of very specialized local models, like the ones used in the Mixture of Experts (MoE) [5], a linear fusion will likely result in aggregating all of the expert's weaknesses together with their strengths thus, loosing most of the potential benefit of fusion. Despite this fact, when considering MoE as the single predictive model, it cannot compete with the strong ensemble models like GBMs [6], [7] or Random Forests [8], in neither accuracy nor their learning speed.

To exploit each of the local model's information, and make the blending process more interaction-specific, we adopt the model fusion approach, based on additive fuzzy systems [9]. We learn a fuzzy neural model, initialized by clustering with gaussian fuzzy patches, which builds atop the original predictor models. We treat the obtained predictive models as the final estimates of the consequent parameters in the fuzzy rules after what we optimize the gaussian parameters of the fuzzy rules obtained. Hence, unlike most of the other fuzzy neural fusion approaches, our fuzzy model behaves as if the consequents were fixed strong nonlinear models, learned with different processes.

In Section II, we describe the fuzzy neural network models. In Section III, we provide the learning algorithm for the fuzzy model fusion. In Section IV, the properties of the fuzzy model fusion are analyzed and the justifications about solving the arising issues are considered. Section V provides the application examples with the proposed fuzzy neural model on the robotic hand controller data. In Section VI, the results and conclusions are discussed.

II. Fuzzy neural networks

In this article only the regression problem will be considered. Suppose we are given the dataset $(x, y)_{i=1}^{N}$, where $x = (x_1, ..., x_d) \in R^d$ refers to the input variables and $y \in R$ to the response variable. The goal is to reconstruct the functional dependence $x \xrightarrow{f} y$ with the estimate $\widehat{f}(x, \theta)$, such that the squared error function is minimized. We will consider the estimate $\widehat{f}(x, \theta)$ to be the additive fuzzy system.

A. Additive fuzzy system model

Fuzzy system is a set of fuzzy "IF-THEN" rules that maps the input variables x to the response variable y. Additive fuzzy systems reconstruct the underlying functional dependence by covering the joint input-output distribution with fuzzy patches. Fuzzy patches form coordinate-wise fuzzy sets in the premise part of the fuzzy rules, and local regression models in the consequent part. Given G fuzzy rules, the i-th fuzzy rule is given in (1), $i = \overline{1..G}$.

$$\text{Rule}_i: \ \textbf{IF}\ x \text{ Is } A_i \ \textbf{THEN}\ \ y \text{ is } \widehat{f}_i(x), \tag{1}$$

- A_i is a fuzzy set defined on x, $i = \overline{1..G}$;
- $\widehat{f}_i(x)$ is a consequent model of the rule i.

We will consider the additive model, where the fuzzy sets A_i are defined on R^d by cartesian product: $A_i = A_{i1} \times \ ... \ \times A_{id}$. Each fuzzy set A_{ij}, defined on x_j, is characterized with some membership function $\mu_{A_{ij}}(x_j) \in [0, 1]$, $i = \overline{1..G}, j = \overline{1..n}$. In this paper we will focus on the Gaussian model of the membership function, parameterized with it's mean m_{ij} and standard deviations s_{ij}:

$$\mu_{A_{ij}}(x_j) = e^{-\frac{(x_j - m_{ij})^2}{2s_{ij}^2}}, i = \overline{1..G}, j = \overline{1..d} \tag{2}$$

978-1-4673-4544-6/13/$31.00 ©2013 IEEE

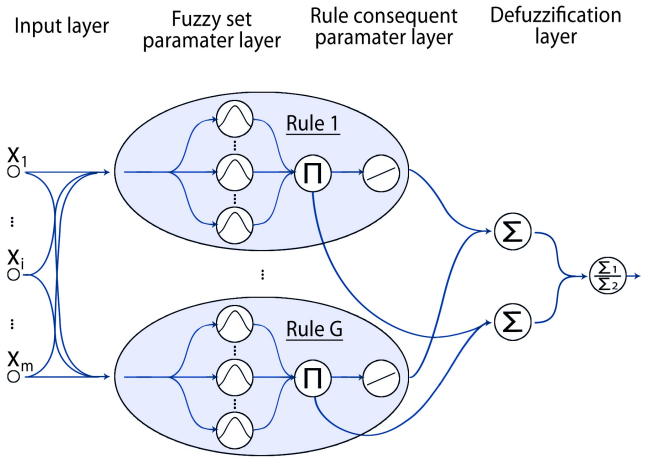

Fig. 1: Neural network representation of the additive fuzzy system. Each rule comprises of two parametric layers. The defuzzification layer is functional and parameter free.

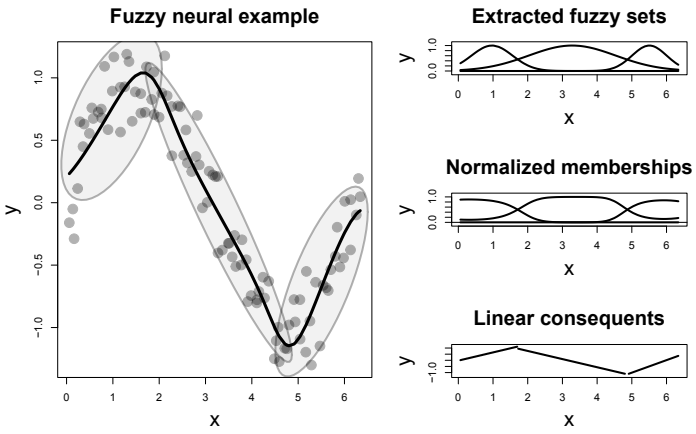

Fig. 2: Example of the fitting stages of the fuzzy neural network. The Gaussian patches in the joint input-output space are shown before marginalization.

The consequent models $\widehat{f}_i(x)$ used can be of different form and complexity. The most commonly used are the linear regression consequents, as they are easy to estimate.

After all the G fuzzy rules are fired, having calculated all the $\mu_{A_{ij}}$ and $\widehat{f}_i$, the aggregated memberships $\mu_{A_i} = \prod_{j=1}^{n} \mu_{A_{ij}}(x_j)$ are calculated. At last, the overall output of the network is calculated as the weighted average of consequents, with weights equal to normalized aggregated memberships. The overlapping fuzzy rules are fused with respect to their relative certainty. The resulting fuzzy model is given in (3):

$$\hat{y} = \frac{\sum_{i=1}^{G} \mu_{A_i} \widehat{f}_i}{\sum_{i=1}^{G} \mu_{A_i}}, i = \overline{1..G} \tag{3}$$

B. Fuzzy neural network model

The (3) fuzzy function estimator can be represented in the form of a feedforward neural network [13] with two parametric layers, containing the evaluation of the premise and consequent parts of fuzzy rules. The resulting neural network architecture is shown on Figure (1). From the purely functional perspective, this architecture corresponds to the first order Takagi-Sugeno-kang (TSK) model [11]. The only difference is that the membership functions are conveniently reorganized in the form of multidimensional clusters μ_{A_i}, each of which corresponds to the cartesian product of marginal membership functions $\mu_{A_{ij}}(x_j)$, $i = \overline{1..G}, j = \overline{1..d}$.

To efficiently initialize the fuzzy rule-base, one can apply unsupervised clustering procedures [12]. The joint input-output $x \times y$ space is covered with cluster patches in an unsupervised way. Then, the obtained clusters are marginalized over y, in order to get cluster projections on x and extract the fuzzy rule's membership functions parameters.

Having the initial estimate of the membership function parameters, one can proceed to learning the resulting neural network model. Demonstration of the initialization process steps is shown on Figure (2). For the demonstration purposes, synthetic noisy $\sin(x)$ function was used as the data input. The final smoothed estimate of the local linear consequents is plotted on the first picture together with the fuzzy patches. At first the two-dimensional Gaussian patches are fitted to the data and marginalized over y in order to extract functions of x only. To fit the local linear consequents, the membership functions are normalized.

III. Fuzzy model fusion

The main difference between the fuzzy model fusion and the fuzzy neural model described above is that in our scenario the consequent functions $\widehat{f}_i(x)$ are considered to be already learned and fixed. Moreover, we assume that one doesn't have access to the consequent function parameters and can only build a model atop the consequents. This might be the case of both simplifying the learning task with submodels built independently, or using models of different types(categorical or continuous) built on different subspaces. Another motivation for such design option would be to use domain-specific models to match different types of data simultaneously, for example sensor reading and camera image data. This is the most critical difference, when comparing this method to other fuzzy neural fusion algorithms.

There are different ways of combining a set of predictive models. One can consider stacking procedures [14], fusion through linear formulas or portfolio-based linear fusion [15]. These approaches can increase the overall accuracy of the system, but they are typically considered for models of the same type [16], [17]. However if the models come from very different functional families, like a linear regression model and a random forest, stacking of these models will not allow one to benefit from this dramatic difference. In this same example, linear regression can work significantly better in the farther regions of data distributions, including extrapolation, whereas

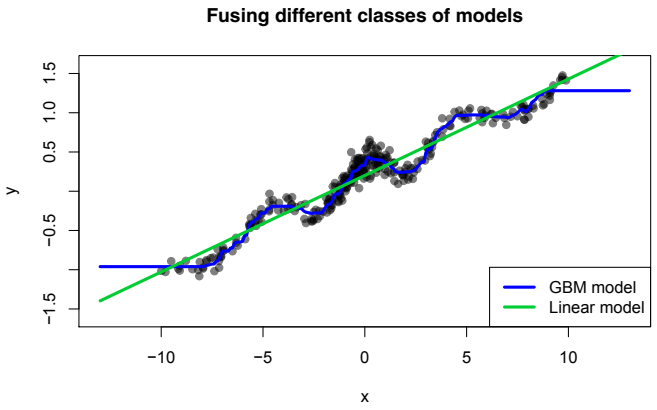

Fig. 3: Artificial data example of fitting two different classes of models: linear regression and a GBM.

the tree-based ensembles will give constant extrapolative predictions, whereas being very precise in the dense regions of data.

To illustrate our motivation to use a diverse set of consequent models which form the ensemble, consider the following scenario: a nonlinear function with two predictive models considered: a linear regression and a GBM. The dataset with the two fitted models is shown on (3). We know that by design, the tree-based GBM model cannot extrapolate, as it consists of piecewise constant functions. However, the GBM ensemble can efficiently catch the principal dynamics of the function, reconstructing most of the original nonlinear dependencies. Linear regression model, on the contrary, can't catch the nonlinear dynamics, however it is well generalizing across the whole dataset. If we stack these models linearly, we would have to loose either the nonlinear part in the center or the values in the tails of the joint distribution.

The straightforward extension to the fusion framework would be to use local-linear fusion models, smoothly switching one another within the fuzzy neural model. It is a nonlinear model, which not only can be efficiently learned, but also pertains the tractability as a grey box model. In this example it would result in using the linear model only in the farther regions of data, whereas exploiting the GBM nonlinearities in the central region. This would let us stay assured about the generalizing properties of the linear model, not loosing the benefits of the strong nonlinear learners.

To proceed with the fusion model, operating on the external consequent models $\widehat{f}$, we must make one important transformation. In order to maintain the model stability, the fuzzy rules should be learned on the joint distribution of the noise terms $\epsilon = \widehat{f} - E(\widehat{f})$ and $E(\widehat{f})$, where $E(\widehat{f})$ is the mean prediction over the whole set of predictors. This transformation is essential because the high correlation of both the consequents $\widehat{f}$ among each other, and their joint correlation with y may lead to both unstable and unreliable clustering results in this joint space. It is important to note that the consequent layer in this type of model will still be using the true values of $\widehat{f}$, concentrating the fuzzy rule learning on the differences between models.

To initialize the appropriate fuzzy rules, we consider the complete joint space of $\epsilon \times E(\widehat{f}) \times y$. The option of considering the complete $x \times \epsilon \times E(\widehat{f}) \times y$ space in order to extract more discriminative information from the complete distribution is also feasible, however it becomes more prone to the curse of dimensionality. Afterwards, given the number of rules G, we apply the Gaussian Mixture Model [10], fitted to the data with the Expectation-Maximization algorithm. The choice of the number of Gaussians G, which corresponds to the number of fuzzy rules in the model, can be done by either trial-and-error, or by means of Bayesian Information Criterion, which is well suited for this type of model.

To proceed with learning of the whole model, one can apply the hybrid learning algorithm. At each iteration, at first, the membership function parameters m_{ij}, s_{ij} are modified by means of gradient descent. Afterwards, given the new parameter estimates, the normalized membership values are calculated $W = \{w_1, \ \dots \ , w_G\}$:

$$w_i(x) = \frac{\mu_{A_i}(x)}{\sum_{j=1}^{G} \mu_{A_j}(x)} \tag{4}$$

Given processed k predictive models $\widehat{f}^{N\times k} = \{\widehat{f}_1, \ \dots \ , \widehat{f}_k\}$ to fuse and the normalized membership matrix $W^{N\times G}$, their column-wise Kronecker product $(\widehat{f} \otimes W)^{N\times Gk}$ can be used to fit the local linear coefficients. One high-level design difference with the conventional fuzzy neural networks is that we use fixed $\widehat{f}_i$ instead of the input variables x. However the linear parameters are still needed to adjust each Rule's fusion of models.

IV. Model fusion design

A. Fusion set

When building a machine learning model, one typically separates the data into training and validation sets (unless the test set is considered). When fitting a nonlinear fusion model, based on different types of models, one has to consider the chance that some of these models suffer from the overfitting. In the perfect situation, fusion algorithm would simply discard these models with nearly-zero weights in the ensemble. However if only the training set is considered, the valuable information about the model interactions will be omitted and the overfitted models will receive inappropriately high weights in each of the fuzzy rules.

To overcome this problem, one has to separate the data into three distinct non-overlapping sets: the external nonlinear consequent-fitting training set, the fuzzy neural fusion set, and the final validation set. This will lead to the tradeoff between the training set sizes, whether to invest some of the training points into the fusion process or not.

B. Reference fusion

One design advantage of fuzzy model fusion is that we can embed information from several features x for building the corresponding membership functions in the fuzzy fusion stage. For example, if one has a MIMO system, one can use estimates of the other variables as references for fusing some particular response variable. This idea can be a perspective approach to feature engineering in the model fusion, having relevant information embedded into the model from different sources selectively. A practical example could be to use some nonlinear projectors like the autoassociative neural networks, as the feature proxy for a more reliable fuzzy neural fusion.

V. Application example

We will consider building a regression model to map the EMG signals to the robotic hand controller, in a similar manner as described in [18]. The data was provided by the TUM Roboroterhalle machine learning laboratory. 8 surface EMG electrodes, positioned on the hand, were used to record the muscular activity of the person performing different hand movements. These movements were then visually tracked to gather the actual spatial positions of the hand. The machine learning task was to reconstruct the hand's position and orientation from the EMG channels. For simplicity, we will consider predicting only the first positional target variable.

For each of the hand position coordinates, the data comprises 8 pre-processed signal features and 27,161 observations. The training, fusion and validation set separation is organized sequentially: the first 100 points are used for training, the following 40 points for fusion and the next 60 points for validation; the next 100 points are used for training again and so on. As a consequence, the training set consists of 13,561 points, the fusion set of 5440 points and the test set consists of 8160 points. The predictive models considered were the tree-based Gradient Boosting Machine (GBM), Random Forests, Support Vector Regression (SVR)R and Linear regression. Each of the model's hyperparameters was chosen by means of 5-fold cross-validation.

The result of fitting the models is presented on Figure (4). For this application we considered using $G = 10$ fuzzy rules, as we found it optimal for our application. It is important to note once again that even though the consequent models were fixed, they could form smooth nonlinear combinations with negative weights, which eventually resulted in fixing some of the unfitted regions. However, even though the fusion is carried out in a smooth and continuous manner, the resulting model fusion still retains both noise artifacts and the model-specific noncontinuous behavior(due to decision tree ensembles). To summarize, each model's accuracy on the test set is given in Table. 1, together with the 1 standard deviation estimate, obtained from 20 model runs (on the same data). The performance metric used was the RMSE. We have also fitted a linear GLM-based model fusion, which resulted in successful accuracy improvement too. But the fuzzy neural-based fusion provided significantly more accurate results.

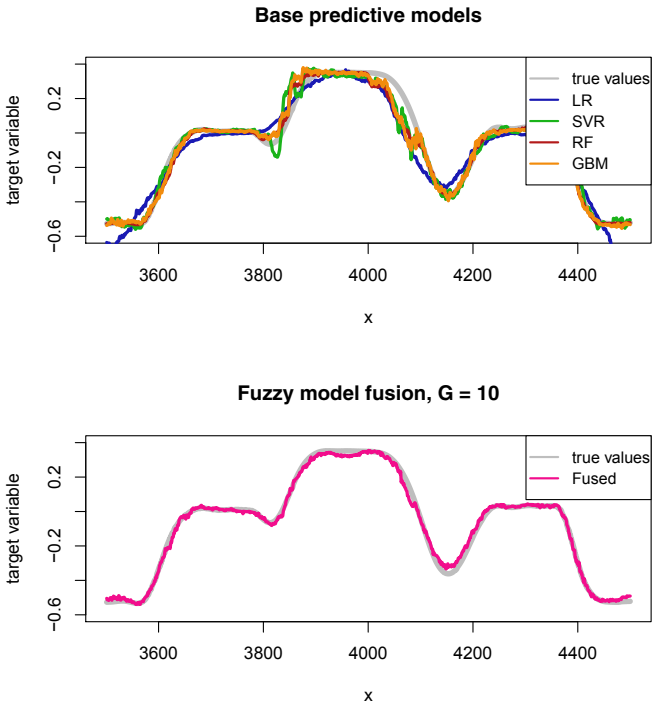

Fig. 4: The fuzzy fusion results.

TABLE I: Machine learning algorithm accuracy

Method	$RMSE$
Fuzzy fusion, G=10	0.027 ± 0.02
Linear fusion, GLM	0.055 ± 0.02
GBM, trees	0.064 ± 0.03
Random Forests	0.064 ± 0.03
Support Vector Machine	0.080 ± 0.07
Linear Regression	0.099 ± 0.00

VI. Conclusion

In the application example we have shown that one can benefit from more sophisticated strategies of model fusion. Fuzzy model fusion provided significant boost in accuracy, when compared to both initial methods and to the linear fusion. It is important to note that this fusion technique worked properly only with the separate fusion set taken into account. Both the RF and the GBM models are already very strong on their own, which lead to the overly high weights for both of these models.

However, the method has several shortcomings. It greatly suffers from the curse of dimensionality, which affects both the unsupervised initialization stage, and the fuzzy membership function parameter learning. Also, the procedure can be sensitive to the number of fuzzy rules considered, which currently requires some trial and error process. Therefore in order to use the model in high-dimensional tasks with large pools of predictors, these problems should be dealt with.

The above mentioned problem with the curse of dimensionality can be partially neglected with more constrained

rule models and the exploitation of either subspace clustering, or some projection pursuit stage before clustering. Anther approach to dealing with this problem would be to make model fusion a hierarchical process. One potential solution to the problem of choosing the number of clusters can be to use some more sophisticated ensemble model of fusion, which will trade the trial and error process with the higher computational cost, like for example, boosting the simple fuzzy fusion networks.

ACKNOWLEDGMENT

The authors would like to thank Prof. Patrick van der Smagt, Jorn Vogel and Justin Bayer from TUM Roboterhalle for providing the robotic control data. The authors also want to thank the anonymous reviewers for their valuable feedback.

REFERENCES

[1] James Llinas, Martin E. Liggins, David L. Hall. Handbook of Multisensor Data Fusion: Theory and Practice, Second Edition. CRC Press, 2008

[2] Toscher A., Jahrer M., and R. Bell.: The BigChaos Solution to the Net ix Grand Prize. Tech report, (September 2009)

[3] Wu K. et al., A Two-Stage Ensemble of Diverse Models for Advertisement Ranking in KDD Cup 2012. KDD Cup 2012

[4] Caruana R. et al., Ensemble selection from libraries of models. In Proceedings of the twenty-first international conference on Machine learning (ICML '04). ACM, New York, NY, USA, 18-.

[5] Yuksel, S.E. Twenty Years of Mixture of Experts. Neural Networks and Learning Systems, IEEE Transactions on, 23, 8 (August 2012), 1177 - 1193

[6] Natekin, A, Knoll, A. Gradient Boosting Machines, A Tutorial. Frontiers in Neurorobotics, 2013, doi:10.3389

[7] Johnson, R., Zhang, T., Learning Nonlinear Functions Using Regularized Greedy Forest. arXiv:1109.0887, 2012

[8] Leo Breiman Statistics and Leo Breiman, Random Forests, Machine Learning. 2001, 5–32

[9] Kosko B.: Fuzzy Systems as Universal Approximator. B.s IEEE Transactions on Computers, 1994, 43, 1329-1333

[10] Ming-Tao Gan, M. Hanmandlu, and Ai Hui Tan.: From a Gaussian mixture model to additive fuzzy systems. Trans. Fuz Sys. 13, 3 (June 2005), 303-316

[11] Babak Rezaee and M.H. Fazel Zarandi. 2010. Data-driven fuzzy modeling for Takagi-Sugeno-Kang fuzzy system. Inf. Sci. 180, 2 (January 2010), 241-255.

[12] George E. Tsekouras et al., A hierarchical fuzzy-clustering approach to fuzzy modeling. Fuzzy Sets and Systems, 2005, 150, 245-266

[13] Jang, J.-S. R. ANFIS: Adaptive-Network-based Fuzzy Inference Systems IEEE Transactions on Systems, Man and Cybernetics, 1993, 23, 665-685

[14] Joseph Sill, Gabor Takacs, Lester Mackey, David Lin, Feature-Weighted Linear Stacking. arXiv:0911.0460, 2009

[15] Shamsoddini A., Trinder J.C., Neural Network Fusion for Regression Problems. International Journal of Machine Learning and Computing, Vol. 2, No. 4, August 2012

[16] Youzhu Ling; Xiaoguang Xu; Lina Shen; Jingmeng Liu, Multi sensor data fusion method based on fuzzy neural network. Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on , vol., no., pp.153,158, 13-16 July 2008

[17] Cheng-Jian Lin, Shyi-Shiun Kuo, and Chun-Cheng Peng, Multiple Functional Neural Fuzzy Networks Fusion Using Fuzzy Integral. International Journal of Fuzzy Systems, Vol. 14, No. 3, September 2012

[18] Vogel, J.; Castellini, C.; van der Smagt, P.: EMG-based teleoperation and manipulation with the DLR LWR-III. Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, 2011