Boosting Simplified Fuzzy Neural Networks

Alexey Natekin[1] and Alois Knoll[2]

[1] Fortiss GmbH, Guerickstr. 25, Munich, Germany
natekin@fortiss.org
[2] Technical University Munich, Boltzmannstr. 3, Garching, Germany
knoll@in.tum.de

Abstract. Fuzzy neural networks are a powerful machine learning technique, that can be used in a large number of applications. Proper learning of fuzzy neural networks requires a lot of computational effort and the fuzzy-rule designs of these networks suffer from the curse of dimensionality. To alleviate these problems, a simplified fuzzy neural network is presented. The proposed simplified network model can be efficiently initialized with considerably high predictive power. We propose the ensembling approach, thus, using the new simplified neural network models as the type of a general-purpose fuzzy base-learner. The new base-learner properties are analyzed and the practical results of the new algorithm are presented on the robotic hand controller application.

Keywords: neural network, gradient boosting, fuzzy neural network, neural ensemble, boosting, robotic control, machine learning.

1 Introduction

Fuzzy neural networks have found successful applications in a wide range of domains. The most commonly used models are the additive fuzzy system models, rule-based universal approximators [2,1]. Fuzzy rules are the core of the fuzzy neural models and can be interpreted as the local submodels, interacting with each other through the fuzzy inference design.

Fuzzy neural networks suffer from a number of problems. First of all, the desired number of the fuzzy rules is typically not known beforehand. This problem is very similar to the problem of choosing the number of neurons in a common neural network, which is typically approached by trial and error or by means of cross-validation. In fuzzy neural networks this problem can be efficiently circumvented with clustering procedures [1,3], used for rule-base estimation. Using Gaussian mixture models, one can efficiently initialize the rule-base in an unsupervised manner and to inspect the optimal number of rules by means of some information criteria, like Bayesian Information Criterion. However, mixture models in their turn greatly suffer from the curse of dimensionality, making them less applicable to a large portion of real-world datasets.

The second problem of the fuzzy neural networks is their learning speed, which is typically higher, when compared to other methods like common MLPs, SVMs or Random Forests. The fuzzy neural network model is simply more complex, which leads to increased learning time, in order to achieve a similar level of generalization accuracy.

We propose the solution to the described fuzzy neural network problems by designing a specific low-cost fuzzy neural network architecture, which will be used as a base-learner in the ensemble models. For this purpose we have developed a simplified fuzzy neural network model, efficiently initialized with k-means clustering and only one least-squares estimation. Under different hyperparameter settings, the proposed simplified network model can have different properties, which are addressed in this paper.

The ensembling approach, based on the gradient boosting algorithm [4] is introduced for fuzzy neural networks. Previous approaches to form fuzzy-based ensembles were mostly based on the bagging procedure, i.e. Fuzzy Random Forests [5]. In our approach, we want to fix the fuzzy base-learner complexity and take advantage of boosting for efficient learning of the details, not captured with other models in the ensemble. In this paper, we will provide both the algorithmic description of the proposed model, and the justifications about the choice of the hyperparameters with the resulting model properties.

In Section 2, we describe the fuzzy neural network models. In Section 3, we provide the basic gradient boosting algorithm description. In Section 4, the proposed fuzzy neural base-learner is presented and it's properties are analyzed. Section 5 provides the application example of the GBM ensembles with the proposed fuzzy base-learners on the robotic hand controller. In Section 6, the results and conclusions are discussed.

2 Fuzzy Neural Networks

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

2.1 Additive Fuzzy System Model

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

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

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

We will consider the additive model, where the fuzzy sets A_i are defined on R^d by cartesian product coordinate-wise: $A_i = A_{i1} \times A_{i2} \times \ ... \ \times A_{id}$. Each fuzzy set A_{ij}, defined on x_j, is in it's turn characterized with some membership function $\mu_{A_{ij}}(x_j) \in [0, 1]$, $i = \overline{1..G}, j = \overline{1..n}$;

There are different design choices for the membership functions to be used. 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}:

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

The consequent models $\widehat{f}_i(x)$ used can also be of different form and complexity. We will consider the commonly used linear regression consequents, as they are easy to estimate by solving LSE problem:

$$\widehat{f}_i(x) = \sum_{j=1}^{d} c_{ij} x_j + c_{i0}, i = \overline{1..G} \tag{3}$$

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.

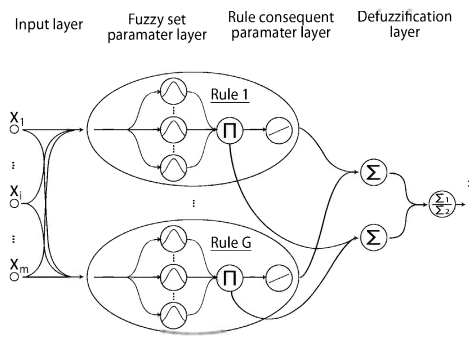
At last, the overall output of the network is calculated as the weighted average of consequents, with weights equal to normalized aggregated memberships. The overlapping fuzzy rules are fuzed with respect to their relative certainty. The overall functional model of the additive fuzzy model is given in (4):

$$\hat{y} = \frac{\sum_{i=1}^{G} \mu_{A_i} (\sum_{j=1}^{d} c_{ij} x_j + c_{i0})}{\sum_{i=1}^{G} \mu_{A_i}}, i = \overline{1..G}, j = \overline{1..d} \tag{4}$$

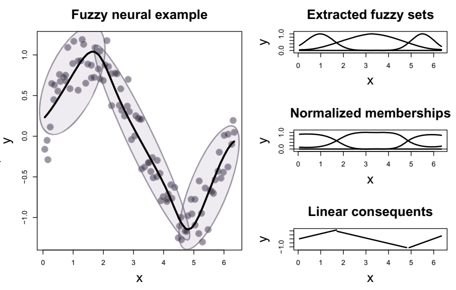
2.2 Fuzzy Neural Network Model

The (4) fuzzy function estimator can be represented in the form of a feedforward neural network 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 (1a).

For learning in these networks, hybrid learning algorithms are used. At each iteration, at first the fuzzy membership function parameters of the Gaussians m_{ij}, s_{ij} are optimized by a gradient descent step, $i = \overline{1..G}, j = \overline{1..d}$. Then, the consequent parameters c_{ij} are optimized by solving a single least squares linear regression problem, $i = \overline{1..G}, j = \overline{0..d}$. Due to the limited size of the article, we omit describing the details of the learning procedure. One can find the learning algorithms thoroughly described in [11].



(a) Neural network representation of the additive fuzzy system



(b) Example of the fitted fuzzy neural network

Fig. 1. Fuzzy neural network model

One of the most efficient and compact ways to approach fuzzy rule initialization is based on unsupervised clustering procedures. 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 initialize the fuzzy rule's membership functions.

Having the membership function parameters initialized, one can proceed to the hybrid learning. Demonstration of the initialization process is shown on Figure (1b). For the demonstration purposes, synthetic noisy $\sin(x)$ function was used as the data input.

3 Gradient Boosting

Gradient boosting machines, or simply GBMs, are a family of efficient ensemble models that can capture complex nonlinear function dependencies. This family of models has shown considerable success in not only various practical applications, but also in various machine-learning and data-mining challenges [9,10].

The GBM models are based on a constructive strategy of ensemble formation. The main idea is to add new models to the ensemble sequentially. The learning procedure consecutively fits new models to provide more accurate estimates. The base-learners used can be chosen in various ways, the most commonly used base-learner models are the decision trees and generalized additive models [6].

The principle idea behind the GBM algorithm is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble. We will consider the squared error loss, where the learning procedure would result in consecutive error-fitting.

The GBMs have some hyperparameters. The most important hyperparameter is the number of base-learners, or boosting iterations in the ensemble M. Larger numbers of M increase the model complexity and can lead to overfitting. Also, two regularization hyperparameters have been introduced. The first of them is the shrinkage λ, which penalizes the effect of each boosting iteration. The second regularization parameter is the bag fraction *bag*, which specifies the percentage of the data to be used at each iteration. The overall description of the GBM algorithm, used in this article is described in Algorithm (1). For more details about the GBMs we recommend the reader the [4,6,7] articles.

Algorithm 1. GBM Algorithm with squared error loss function

Inputs:

- input data $(x, y)_{i=1}^{N}$
- number of iterations M
- regularization parameters λ and bag
- choice of the base-leaner model $h(x, \theta)$
- base-leaner hyperparameters θ

Algorithm:

```
1: initialize $\widehat{f}_0$ with a constant
2: for $t = 1$ to $M$ do
3:     sample $Bag_t = bag \cdot N$ indices, used for fitting a base-learner
4:     compute the negative gradient $g_t(x) = \sum_{i \in Bag_t} (y_i - \sum_{j=0}^{t-1} \widehat{f}_j(x_i))$
5:     fit a new base-learner function $h(x, \theta)$ to the gradient $g_t$
6:     find the best gradient descent step-size $\rho_t$:
           $\rho_t = \arg\min_{\rho} \sum_{i \in Bag_t} [g_t(x_i) - \rho h(x_i, \theta)]^2$
7:     update the ensemble:
           $\widehat{f}_t \leftarrow \widehat{f}_{t-1} + \rho_t \lambda h(x, \theta)$
8: end for
```

4 Simplified Fuzzy Neural Base-Learner

We now aim at designing a simplified fuzzy neural network model, which will have two important properties. First of all, it is going to be a fuzzy non-linear model, which can capture complex non-linearities. And second, it will be very fast to initialize and learn, in order to be the base-learner, used for boosting thousands of neural network models in reasonable time.

If we consider the model in (4) and it's demonstration on Figure (1b), we can denote that the model tends to fit fuzzy patches with local linear models, slightly smoothing the models with respect to their memberships. An obvious way to simplify the model will be to cut the consequent model to the constant terms c_{i0} only, $i = \overline{1..G}$.

This model will still remain smooth and continuous due to taking memberships into account. In hybrid learning, to estimate the parameters c_{ij}, one has to solve a LSE problem for a very large matrix, having dimensions $n \times (d+1)G$, $i = \overline{1..G}$, $j = \overline{0..d}$. Using only the intercept parameters in each rule's consequent will dramatically reduce the size of this matrix to $n \times G$. This particular simplification is equivalent to the Takagi-Sugeno model of the 0-th order, but we aim at simplifying the model even further.

The next problem is the initialization of the fuzzy system. Mixture-based clustering, currently considered in the model, is a computationally expensive procedure. Moreover, inferring the number of clusters is also a problematic task.

To deal with these issues, one can consider the k-means clustering as an example of a simple mixture model, where the Gaussian clusters have all equal covariance structure of the form $\Sigma = vI$. This spherical simplification of the model significantly decreases the initialization time. The parameter $v > 0$ is held out as a hyperparameter that describes the uncertainty of the fuzzy rules.

Exploitation of the k-means clustering also allows us to use larger number of fuzzy rules, which in the clustering context is equal to the number of clusters. The larger the number of rules, the more complicated patterns one can capture. This particular design choice allows us to reduce the number of fuzzy rule parameters from $G \times d \times 2$ to $F \times d + 1$, having only the Gaussian center parameters and the only instance of v considered.

The last constraint that we haven't touched upon yet is the curse of dimensionality: the geometrical structure, captured by the clustering algorithm vanishes with the dimensionality of data increased. We will use a simple heuristic, used in Random Forests: for each of the base-learners in the ensemble we will randomly sample $p = \lceil \sqrt{d} \rceil$ dimensions, used for fitting. We constrain ourselves with the dimensionality p and number of clusters G for each of our base-learners.

To summarize, the simplified fuzzy neural (SFN) model is given in (5). The fitting procedure of one SFN base-learner is organized in the Algorithm (2).

$$\begin{gathered}\mu_{A_{ij}} = e^{-\frac{(x_j - m_{ij})^2}{V}}, \mu_{A_i} = \prod_{j=1}^{n} \mu_{A_{ij}}(x_j), \\ \hat{y} = \frac{\sum\limits_{i=1}^{G} \mu_{A_i}(c_i)}{\sum\limits_{i=1}^{G} \mu_{A_i}}, i = \overline{1..G}, j \in R^p \subset R^d\end{gathered} \tag{5}$$

4.1 SFN Properties

The hyperparameter v defines the uncertainty of the whole fuzzy model, behaving like a fuzzy membership modifier, either sharpening or smoothing the membership values. However, there is one important effect, connected with this parameter. To illustrate it we define the average winning normalized membership *awnm* as (6).

$$awnm = \frac{1}{N} \sum_{j=1}^{N} \max_{i \in 1..G} \frac{\mu_{A_i}(x_j)}{\sum\limits_{k=1}^{G} \mu_{A_k}(x_j)} \tag{6}$$

Algorithm 2. Fitting a simplified fuzzy base-learner

Inputs:

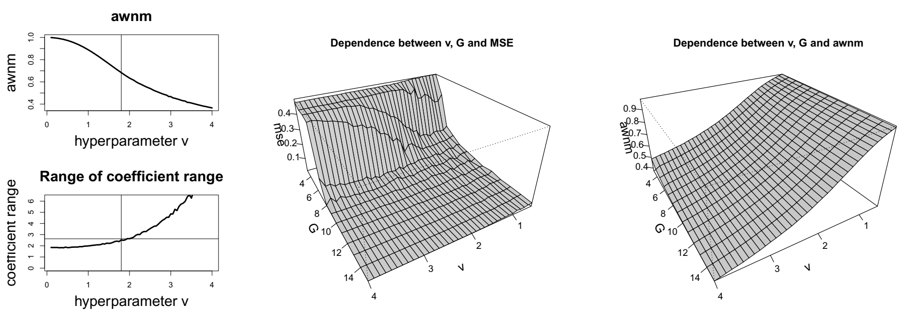
- input data $(x, y)_{i=1}^{N}$
- number of fuzzy rules $\sim$ clusters G
- number of dimensions $p < d$, used for fitting. default value: $p = \lceil\sqrt{d}\rceil$
- uncertainty hyperparameter $v > 0$

Algorithm:

```
1: sample $p$ dimensions of $x$, used for fitting $x^p$
2: apply k-means clustering with $G$ clusters to the joint $(x^p, y)$ distribution
3: marginalize the $y$ variable from the obtained cluster centers:
       $(m_{ij}, m_{iy}) \rightarrow (m_{ij}),\ i = \overline{1..G},\ j = \overline{1..p}$
4: calculate the average cluster standard deviations:
       $\sigma^2 = \frac{1}{pN} \sum_{i=1}^{G} \sum_{j=1}^{p} \sigma_{ij}^2, \sigma_{ij}^2 = \frac{1}{N-1} \sum_{k \in Cluster_i} (x_{kj} - m_{ij})^2$
5: initialize the parameter $V = \frac{v}{\sigma^2}$
6: estimate the $c_i$ parameters by fitting a linear regression model to normalized memberships $\mu_{A_i}$, $i = \overline{1..G}$
```

When the value of *awnm* is near to 1, the smooth borders between rules are nearly diminished and the SFN model has a sharp discontinuous form, similar to the decision trees. On the contrary, when the *awnm* is near to $\frac{1}{G}$, constructive membership information vanishes and the coefficients c_i become unstable. Visualization of these effects for fixed number of rules $G = 20$ on the same simulated $\sin(x)$ data is given on Figure (2).

From Figure (2a) we can see that starting from $awnm = 0.7$ the coefficients become less stable. If we consider the dependence between G, *awnm* and MSE error, we can see that from some number G the model stabilizes and converges



(a) *awnm* and coef.range

(b) joint G and v dependencies with *awnm* and MSE

Fig. 2. SFN hyperparameter dependencies

to the same output, regarding complexity. And the *awnm* has a similar behavior all across the hyperparameter grid. The plot with the described dependencies is given on Figure (2b). In future works, parametrization based on *awnm* will be considered, as this statistic is more descriptive than paramater v.

4.2 SFN's Connection to Other Base-Learners

SFNs have one important common property with decision trees: they extrapolate the data with constant values. This means that the extrapolation with the SFN model will most likely result in exactly the coefficient value of the closest fuzzy rule, on the border of the observed data., just like a decision tree. This can be considered both a shortcoming and a feature.

Another property of the SFNs is that if the boosted model is considered to be additive, they can be used as a substitute for spline base-learners. Together with the variety of continuity in shape, this makes the SFNs a general purpose base-learner, lacking only the purely linear effects.

It feasible to interpolate a linear function with the SFNs, unlike the decision trees, but the linear effects can be intuitively added to the model. Thus, forming a model that captures nonlinear SFN effects, accounted after a linear fit.

5 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 [8]. 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. Combined, this data was used to design a robotic arm control system. The machine learning task was to reconstruct the hand's position and orientation from the EMG channels and then to use it online as the robotic hand control. In our application we will consider a slightly altered experiment setup than the originally described one. We will focus on predicting the 3D hand positions only.

For each of the hand position coordinates, the data comprises 8 pre-processed signal features and 27,161 observations. Originally, the controller was first trained on all of the available data and then tested live, without knowing the correct labels, as the hand-tracking device was not available. In our case, we train the models on one half the available data and validate it on the other part. The train/test separation is organized sequentially: the first 100 points are used for training, the following 100 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 and the test set consists of 13,600 points.

The model performance metric to be used will be the root mean squared error (RMSE), evaluated for each of the position variables y_i, $i = 1, 2, 3$ and the compiled, 3D error metric:

$$\begin{aligned} RMSE_i &= \sqrt{\sum_{j=1}^{N} \frac{1}{N} (y_{ij} - \widehat{y_{ij}})^2} \\ M3DE &= \sum_{i=1}^{N} \frac{1}{N} \sqrt{(y_{1i} - \widehat{y_{1i}})^2 + (y_{2i} - \widehat{y_{2i}})^2 + (y_{3i} - \widehat{y_{3i}})^2} \end{aligned} \quad (7)$$

We consider 3 SFN models with different number of rules G. The regularization parameters shrinkage $\lambda = 0.01$ and $bag = 0.5$ were chosen beforehand because they are a sensible guess of GBM model's detalization. The optimal number of base-learners was chosen by cross-validation as $M = 1000$, number of dimensions used $p = 4$. We have also cross-validated the v parameter and for this application selected it equal to $v = 2$.

To make the model evaluation fair, we will compare the SFN model with other popular machine learning techniques: SVMs, ANFIS, Random Forests and tree-based GBMs. For each of these models the optimal hyperparameters were chosen by the 5-fold cross-validation, applied to the grid-search. The algorithm accuracy comparisons are given in Table (1).

Table 1. Machine learning algorithm accuracy

Method	$RMSE_1$	$RMSE_2$	$RMSE_3$	$M3DE$
Boosted SFN, G=10, v=2	0.073	0.057	0.078	0.087
Boosted SFN, G=20, v=2	0.069	0.054	0.071	0.084
Boosted SFN, G=30, v=2	0.065	0.052	0.067	0.081
GBM, trees	0.063	0.054	0.066	0.081
Random Forests	0.062	0.054	0.067	0.081
Support Vector Machine	0.076	0.069	0.084	0.100
ANFIS	0.074	0.069	0.089	0.103
Linear Regression	0.100	0.087	0.095	0.136

From Table (1) we can see that the new base-learner can efficiently compete with other machine learning techniques, delivering the same high level of accuracy. And the accuracy level is significantly higher than the accuracy of a single ANFIS, trained with all the necessary parameters and no simplifications. At last, the although the accuracy is nearly the same as the GBM and RF, the predictions of the new model are much smoother.

The high SFN performance can be explained by accurately catching nonlinear interactions. RF and GBM also exploit the interaction structure of the data in the ensemble, but due to decision tree limitations in approximating continuous functions, Boosted SFNs outperform these methods.

To summarize, the developed SFN model allowed us not only to design a competitive machine learning algorithm, but also to efficiently exploit all of the available information and build an accurate predictive model.

6 Conclusion

We have developed a new type of base-learner, which is not only interesting from a theoretical perspective, but has shown considerable success in practice, compared to other machine learning algorithms. The new model allows to have a fast fit of smooth interactions between variables, like the currently used generalized additive base-learners, but in multiple dimensions. Varying the hyperparameter v, one can also achieve different model behavior. Besides generalizing the model to the classification task, a straightforward extension to the algorithm would be to estimate the optimal parameter v with several gradient steps.

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 thank anonymous reviewers for their valuable comments.

References

1. Gan, M.-T., Hanmandlu, M., Tan, A.H.: From a Gaussian mixture model to additive fuzzy systems. Trans. Fuz Sys. 13(3), 303–316 (2005)
2. Kosko, B.: Fuzzy Systems as Universal Approximator. B.s. IEEE Transactions on Computers 43, 1329–1333 (1994)
3. George, E., Tsekouras, H.S.: A hierarchical fuzzy-clustering approach to fuzzy modeling. Fuzzy Sets and Systems 150, 245–266 (2005)
4. Friedman, J.: Greedy Boosting Approximation: A Gradient Boosting Machine. The Annals of Statistics 29, 1189–1232 (2001)
5. Bonissone, P., Cadenas, J.M., Garrido, M.C., Diaz-Valladares, A.R.: A fuzzy random forest. International Journal of Approximate Reasoning 51, 729–747 (2010)
6. Hothorn, T., Buhlmann, P., Kneib, T., Schmid, M., Hofner, B.: Model-based boosting 2.0. Journal of Machine Learning Research 11, 2109–2113 (2010)
7. Natekin, A., Knoll, A.: Gradient Boosting Machines, A Tutorial. Frontiers in Neurorobotics (2013), doi:10.3389
8. Vogel, J., Castellini, C., van der Smagt, P.: EMG-based teleoperation and manipulation with the DLR LWR-III. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS (2011)
9. Johnson, R., Zhang, T.: Learning Nonlinear Functions Using Regularized Greedy Forest. arXiv:1109.0887 (2012)
10. Bissacco, A., Yang, M.-H., Soatto, S.: Fast Human Pose Estimation using Appearance and Motion via Multi-Dimensional Boosting Regression. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2007 (2007)
11. Jang, J.-S.R.: ANFIS: Adaptive-Network-based Fuzzy Inference Systems. IEEE Transactions on Systems, Man and Cybernetics 23, 665–685 (1993)