Using Hierarchical Recurrent Neuro-Fuzzy Systems for Decision Making

Andreas Nürnberger

University of California at Berkeley
EECS, Computer Science Division
Berkeley, CA 94720-1770, USA
email: anuernb@eecs.berkeley.edu

Abstract
Over the last years, a number of methods have been proposed to automatically learn and optimize fuzzy rule bases from data. The obtained rule bases are usually robust and allow an interpretation even for data sets that contains imprecise or uncertain information. However, most of the proposed methods are still restricted to learn and/or optimize single layer feed-forward rule bases. The main disadvantages of this architecture are that the complexity of the rule base increases exponentially with the number of input and output variables and that the system is not able to store and reuse information of the past. Thus temporal dependencies have to be encoded in every data pattern. In this article we briefly discuss the advantages and disadvantages of hierarchical recurrent fuzzy systems that tackle these problems. Furthermore, we present a neuro-fuzzy model that has been designed to learn and optimize hierarchical recurrent fuzzy rule bases from data.

Introduction
In the beginning of the application of fuzzy systems (Zadeh 1965; Zadeh 1973) to real world problems, we were mainly confronted with simple rule bases that used just a few input variables to compute usually just a single output value. Meanwhile, the complexity of fuzzy systems that are used in decision making and control theory has increased drastically. Therefore, a number of methods have been proposed to automatically learn and optimize fuzzy rule bases from data.

Besides approaches that are based on decision trees (e.g. Yuan, and Shaw 1995; Chi, and Yan 1996; Wang et al. 1999), fuzzy cluster analysis (e.g. Bezdek et al. 1998; Höppner et al. 1999), genetic or evolutionary algorithms (e.g. Lee, and Takagi 1993; Cordon et al. 2001), neuro-fuzzy systems (e.g. Lin, and Lee 1996; Nauck, Klawonn, and Kruse 1997; Pedrycz 1998) have meanwhile proven their usability in practice. The obtained rule bases are usually robust and allow an interpretation even for data sets that contains imprecise or uncertain information. However, most of the proposed methods are still restricted to learn and/or optimize single layer feed-forward rule bases. The main disadvantages of this architecture is that the complexity of the rule base increases exponentially with the number of input and output variables – if we want to ensure a coverage of the considered data space – and that the system is not able to store and reuse information of the past. Thus temporal dependencies have to be encoded in every data pattern that is presented to the system, e.g. if feed-forward systems are applied to time series data, these data has to be preprocessed and recoded, so that the relevant dynamic information is available in each data set, which is presented to the system. In case of applications in system control, this is usually realized by additional use of the derivation or integral of the system state. If this information is not available, a vector of prior states has to be stored and used as additional system input. The same holds for fuzzy systems that should be applied to dynamic decision making problems. Unfortunately, it is usually hard to define an ‘optimal’ number of prior states that should be used and for continuously running systems additionally the time-delay between the used state information has to be determined. The use of prior state vectors leads again to a (sometimes drastically) increased number of inputs, which soon becomes intractable. Furthermore, by the increased number of inputs, the interpretability of the rule base suffers. Thus, these systems can not be used in applications where the decision making process has to be clearly visible.

In this article we discuss the advantages and disadvantages of hierarchical recurrent fuzzy systems that tackle these problems. Furthermore, we present a neuro-fuzzy model that has been designed to learn and optimize hierarchical recurrent fuzzy rule bases from data.

Hierarchical Fuzzy Systems
The development of hierarchical fuzzy systems was motivated by the dimensionality problem of single-stage fuzzy systems, i.e. fuzzy systems that support only a direct mapping from the input to the output variables. In these systems the total number of fuzzy rules increases exponentially with the number of input and output variables, since usually all combinations of input variables have to be considered to cover the complete input space. Therefore, also the total number of adjustable system parameters might increase exponentially, if individual fuzzy sets are assigned to each rule.
Hierarchical Recurrent Fuzzy Systems

Pure feed-forward fuzzy system are not able to store and reuse information of the past. Thus temporal dependencies have to be encoded in every data pattern that is presented to the system. If they are applied to time series data, these data has to be preprocessed and recoded, so that the relevant dynamic information is available in each data set, which is presented to the feed-forward system. This problem led to the idea to store prior system states in internal variables of the fuzzy system (Gorrini, and Bersini 1994). To be able to store ‘information of the past’, the simple introduction of internal variables as in hierarchical rule structures is obviously not sufficient, unless the system information of prior system states (the values of output and inner variables of the fuzzy system) is reused as input. Therefore, all variables are now regarded as time dependent and thus the value of each variable depends on time t. Furthermore, recurrent links are introduced, which can be used to feed-back the output of variables of the previous time step. The propagation can still be done as in pure hierarchical systems, since the recurrence is hidden to the subsystems. However, initial values for the recurrent links have to be defined for the first propagation of the rule base. A simple example of a hierarchical recurrent rule base is depicted in Figure 2.
**A Hybrid Recurrent Neuro-Fuzzy System**

In (Nürnberger 2001) we have presented a neuro-fuzzy system that can be used to learn and optimize hierarchical recurrent fuzzy rule bases – as discussed in the previous section – from data. The structure is based on local Mamdani-like fuzzy system for function approximation (Mamdani, and Assilian 1975), that can be combined to an arbitrary hierarchical and recurrent rule base. The encoding of a fuzzy system in a neural network like architecture is motivated by the generic fuzzy perceptron (Nauck 1994), which was also used in our feed-forward neuro-fuzzy models for fuzzy control, approximation and classification (see, e.g. Nauck, Klawonn, and Kruse 1997; Nürnberger, Nauck, and Kruse 1999).

The learning algorithm proposed in (Nürnberger 2001) supports logistic or Gaussian-like fuzzy sets to define the membership functions of the antecedents and Gaussian-like fuzzy sets to define the consequents. However, the algorithm can easily be adapted to other (differentiable) membership functions.

In the following we briefly describe the structure of the model and the used learning methods.

**Model Structure**

The main idea of this model is to combine simple feed-forward fuzzy systems – which may consist of just one rule – to arbitrary hierarchical models. Therefore, the interpretability of every part is ensured before, during and after optimization. Backward connections between the models are realized by time-delayed feed-back links (see also the discussions in the previous sections). The interpretability of the fuzzy sets is ensured by the use of coupled weights in the consequents (fuzzy sets, which are assigned to the same linguistic terms share their parameters) and in the antecedents (layer two). Furthermore, constraints can be defined, which have to be observed by the learning method, e.g. that the fuzzy sets have to cover the considered state space. A possible structure is shown in Figure 3. A more formal definition is given in the following.

**Definition 1.** Let \( \text{Ant}(r) \) be the set of the fuzzy sets used in the antecedent and \( \text{Con}(r) \) be the set of the fuzzy sets used in the consequent of rule \( r \), then the recurrent neuro-fuzzy system is a fuzzy system with the following specifications:

(i) All fuzzy sets \( \mu_i^{(r)}(x_i) \in \text{Ant}(r) \) are defined either Gaussian-like or logistic fuzzy sets

\[
\mu_i^{(r)}(x_i) = e^{-\frac{(x_i-c_i)^2}{2\sigma_i^2}} \quad \text{or} \quad \mu_i^{(r)}(x_i) = \frac{1}{1+e^{-(x_i-d_i)^2}} ,
\]

where \( \mu_i^{(r)} \) is the \( j \)-th fuzzy set assigned to input \( x_i \).

(ii) Each rule is a tuple

\[
R_r = (\mu_1^{(r)}, \ldots, \mu_n^{(r)}, \nu_1^{(r)}, \ldots, \nu_k^{(r)})
\]

and has the form

\[
R_r : \text{if } x_{i_1} \in \mu_{i_1}^{(r)} \text{ and } \ldots \text{ and } x_{i_n} \in \mu_{i_n}^{(r)} \text{ then } y_{k_1} \in \nu_{k_1}^{(r)} \text{ and } \ldots \text{ and } y_{k_k} \in \nu_{k_k}^{(r)} .
\]

(iii) The activation \( a_i(t) \) of a fuzzy rule at time \( t \) is computed by

\[
a_i(t) = \prod_{i,j} \mu_i^{(r)}(x_i) .
\]

(iv) The output of the system for each output variable \( y_k \) is computed by a weighted sum:

\[
y_k(t) = \frac{\sum_{r: \nu_k^{(r)} \in \text{Con}(r)} \text{area}(v_k^{(r)}, a_i(t)) \cdot \text{center}(v_k^{(r)}, a_i(t))}{\sum_{r: \nu_k^{(r)} \in \text{Con}(r)} \sum_{r: \nu_k^{(r)} \in \text{Con}(r)} \text{area}(v_k^{(r)}, a_i(t)) \cdot \text{center}(v_k^{(r)}, a_i(t))} .
\]
where area computes the area and center the center of gravity of the output fuzzy set $\hat{v}_k^{(i)}$ of rule $r$ for $y_k$ at time $t$ which is defined as

$$\hat{v}_k^{(i)}(y_k) = \min(v_i^{(k)}(y_k), a_i(t)).$$

(v) Two types of feedback connections are allowed:

a. Time-delayed feedback: Any output $y_k$ can be assigned to an input $x_i$ by defining $x_i(t) := y_k(t-1)$.

b. Hierarchical feedback: An output $y_k$ can be assigned to an input $x_i$ by defining $x_i(t) := y_k(t)$ if $x_i(t)$ is not used for the computation of $y_k(t)$ neither directly nor indirectly via another (hierarchical) input $x_j(t)$.

The feedback connections defined in (v) enable the definition of hierarchical fuzzy systems (b.) and the use of time-delayed inputs (a.). It should be noted, that only time-delayed feedback connections are considered as ‘real’ recurrent connections. Hierarchical feedback connections are simply a ‘recurrent’ definition of a hierarchical fuzzy rule base, and the condition defined in (b.) ensures that the rule base can be sorted topologically.

The used defuzzification method is a modified center of gravity approach (COG; see e.g. Kruse, Gebhardt, and Klawonn 1994). In the commonly used COG approaches the output fuzzy sets are merged prior to defuzzification. Therefore, overlapping areas of fuzzy sets are neglected. Here, the area is weighted by the number of occurrences, thus considering the number of ‘votes’ for a specific output.

**Learning Method for Parameter Optimization**

The considered learning problem is a function approximation problem and for each input vector (time series vector) the desired output vector is known. Therefore supervised learning methods can be used to optimize the parameters of the membership functions. The realized learning method is motivated by the real-time recurrent learning method that has already been applied successfully for the optimization of the weights in recurrent neural networks (Williams, and Zipser 1989). The idea is to propagate the error obtained at the output units back through the rules of the (hierarchical) fuzzy system and adapt the fuzzy sets accordingly. If feedback connections have to be considered, the fuzzy rule base is unfolded in time and the error is propagated back through time as described briefly below. A more detailed description of the learning algorithm is given in (Nürnberger 2001).

Let $E$ be the total error (the cost function to be minimized) of all output neurons $k$ over all time steps $t=0, ..., T$:

$$E = \sum_{t=0}^{T} E(t),$$

with

$$E(t) = \frac{1}{2} \sum_k (E_k(t))^2,$$

and

$$E_k(t) = \begin{cases} a_k(t) - y_k(t) & \text{if node } k \text{ has a target output } a_k \text{ at time } t, \\ 0 & \text{otherwise}, \end{cases}$$

be an error measure for output node $k$. Then the error gradient of error $E(t)$ for an arbitrary parameter $p$ of the considered fuzzy rule base is defined as

$$\frac{\partial E(t)}{\partial p} = -\sum_k E_k(t) \frac{\partial y_k(t)}{\partial p}.\quad (4)$$

Using Definition 1 we obtain

$$\frac{\partial y_k(t)}{\partial p} = \frac{\partial y_k(t)}{\partial f_p} + \sum_{i=1}^{n} \left[ \frac{\partial y_k(t)}{\partial a_i} \frac{\partial a_i(t)}{\partial p} + \sum_{i=1}^{n} \frac{\partial a_i(t)}{\partial x_i} \frac{\partial x_i(t)}{\partial p} \right],$$

where $\partial y_k(t)/\partial p$ is the error gradient of error $E(t)$ for a parameter $p$ in the fuzzy rule base. The figure illustrates the structure of the recurrent neuro-fuzzy system (using one time-delayed and one hierarchical feedback).
where the derivatives \( \frac{\partial x_i(t)}{\partial p} \) have to consider the derivations resulting from the input of connected subsystems or prior time-steps if a time delayed link is used. Thus, four cases have to be considered:

\[
\frac{\partial x_i(t)}{\partial p} = \begin{cases} 
\frac{\partial y_j(t)}{\partial p} & \text{If } \frac{\partial x_i(t)}{\partial p} \text{ is a hier. feedback, } y_j(t), \\
\frac{\partial y_j(t-1)}{\partial p} & \text{If } \frac{\partial x_i(t)}{\partial p} \text{ is a time del. feedb. } y_j(t-1), \\
\frac{\partial x_i(t-1)}{\partial p} & \text{If } \frac{\partial x_i(t)}{\partial p} \text{ is a self-feedback,} \\
0 & \text{If } \frac{\partial x_i(t)}{\partial p} \text{ is an external input.}
\end{cases}
\]

If \( x_i(t) \) is an external input to the system, then \( \frac{\partial x_i(t)}{\partial p} = 0 \) since the external input is independent from any parameter of the system (neglecting any external recurrences, which could not be handled by the system itself).

If \( x_i(t) \) is assigned to an internal variable of the system, three cases have to be considered:

- For a hierarchical feedback with \( x_i(t) := y_j(t) \), \( \frac{\partial x_i(t)}{\partial p} \) can be computed by (5) by replacing \( x_i(t) \) with the output variable \( y_j(t) \) of the preceding layer, since a hierarchical feedback is independent from \( x_i \).
- If \( x_i(t) \) is a self-feedback input, i.e. \( x_i(t) := x_i(t-1) \), then \( \frac{\partial x_i(t)}{\partial p} \) can be computed iteratively similar to RTRL learning (Williams, and Zipser 1989), by assuming \( \frac{\partial x_i(0)}{\partial p} = 0 \), since the initial state can be considered as independent from \( p \).
- If \( x_i(t) \) is a time-delayed feedback input, i.e. \( x_i(t) := y_j(t-1) \), then \( \frac{\partial x_i(t)}{\partial p} = \frac{\partial y_j(t-1)}{\partial p} \) and \( \frac{\partial y_j(t)}{\partial p} \) can be computed by use of Equation (5).

Finally, the parameters of the fuzzy sets are adapted according to

\[
p(t+1) = p(t) - \eta_p \frac{\partial E(t)}{\partial p},
\]

where \( \eta_p \) is the learning rate for a parameter \( p \).

### Implementation

The learning method discussed above was implemented in a software tool for the interactive design and computation of hierarchical recurrent fuzzy rule bases. To ensure a good interpretability of a learned rule base, the tool supports – besides shared weights as defined above – the use of simple constraints that has to be observed by the learning process’ and a simple pruning strategy: Rules are removed from the rule base if the support of an assigned fuzzy set is decreased below a threshold value during learning. (The influence of this rules on the output is very low and therefore the rule can usually be neglected.)

Furthermore, some modifications of the learning process have been implemented: Batch or on-line learning combined with a momentum term and a teacher forcing strategy (see, e.g. Williams, and Zipser 1989). A screenshot of the software implementation is shown in Figure 4.

Besides the manual definition of a rule base, the software implementation also provides a heuristic to learn a fuzzy rule base from data. This method is briefly described in the following.

### Rule Base Learning

The main idea of this approach is to learn the hierarchically structured rule base of local subsystems by use of rule templates. These templates enable the definition of the variables and the belonging time delays that should be used for the creation of the subsystems.

For recurrent systems it is usually necessary to be able to have different fuzzy partitions for input and output variables that are assigned to the same domain. Therefore the rule templates allow to define individual subsets of fuzzy sets for each domain. Thus, the rule learning algorithm can

---

1 For discussions concerning the use of constraints in neuro-fuzzy learning approaches, see e.g. Nauck, Klawonn, and Kruse 1997.
be restricted to choose only fuzzy sets of a specific subset during rule creation. In the realized software implementation subsets can be created by defining a simple filter pattern for every variable such that only fuzzy sets with matching linguistic terms are used in the learning process for a specific rule template. The templates can be defined using generic fuzzy rules. For example, the subsystem template

\[
\text{IF (} x_1(t-1) \text{ LIKE } '*2' \text{) AND (} x_2(t-1) \text{ LIKE } '*1' \text{) THEN (} y_1(t) \text{ LIKE } '*0' \text{)}
\]

is used to create rules like

\[
\begin{align*}
\text{IF (} x_1(t-1) \text{ IS small2) AND (} x_2(t-1) \text{ IS large1) THEN (} y_1(t) \text{ IS zero0)}\\
\text{IF (} x_1(t-1) \text{ IS zero2) AND (} x_2(t-1) \text{ IS large1) THEN (} y_1(t) \text{ IS large0).}
\end{align*}
\]

Based on the rule templates, the specific instances of the membership functions have to be selected for each rule. The used learning method is motivated by the heuristics implemented in the NEFPROX model (Nauck, and Kruse 1997). The algorithm requires an existing fuzzy partitioning for each considered domain. However, the rule base learning algorithm can be modified to create fuzzy sets and thus to avoid an initial partitioning of the domains (Nauck, and Kruse 1997).

The algorithm consists of two parts. During the first part (rule base learning), the algorithm creates rules iteratively based on the given training data. The algorithm selects for each variable the fuzzy set (of the subset) that obtains the highest membership degree for the given value and assigns it to the antecedent or consequent, respectively. Then the fuzzy rules are created based on the given templates. For fuzzy rules used to compute inner variables, the respective consequents are assigned randomly.

During the second part (consequent re-assignment), the algorithm re-assigns the consequents based on the obtained mean over all inputs for each output variable (considering a minimal rule activation) during a complete propagation through time, i.e. the fuzzy set that obtains the highest membership degree for the mean value is assigned to the consequent.

**Application Example**

In the following example we use the discussed neuro-fuzzy system to identify a dynamic process in order to be able to predict its future states. The example describes an application to a system identification problem in control theory (simple physical spring-mass model). However, an analogous problem in decision making is the identification of a dynamical process. The example describes an application to a system identification problem in control theory.

For the derivation of the model, we consider a mass on a spring. Let the height \( x = 0 \) denote the position of the mass \( m \) when it is at rest. Suppose the mass would be lifted to a certain height \( x(0) = x_0 \) and would then be dropped (i.e. the initial velocity is \( v(0) = 0 \)). Since the gravitational force on the mass \( m \) is the same for all heights \( x \), we may neglect it. The force exerted by the spring is governed by Hooke’s law, according to which the force of a spring is proportional to the change of the length of the spring and opposite to the direction of this change. This means \( F = -c \cdot \Delta l = -c \cdot x \), where \( c \) is a constant that depends on the spring. According to *Newton’s second law* \( F = m \cdot \alpha = m \cdot \ddot{x} \) this force causes an acceleration of the mass \( m \). Consequently, we obtain the differential equation

\[
\ddot{x} = -\frac{c}{m} x,
\]

with the initial values \( x(0) = x_0 \) and \( v(0) = \dot{x}(0) = 0 \).

This second order differential equation can be transformed into a system of two coupled first order differential equations by introducing an intermediary variable \( v \):

\[
\dot{x} = v \quad \text{and} \quad \dot{v} = -\frac{c}{m} x. \tag{8}
\]

As system parameters the values \( c=40, m=1 \) and \( x_0=1 \) were used. A data set was generated by simulating the system for a period of 20 sec. with a (fixed step size \( \Delta t=0.1 \)) explicit fourth-order Runge-Kutta method and storing the values \( x(t) \) and \( v(t) \). To obtain a rule base that can be interpreted with respect to each variable, two rule templates – one for the computation of \( x \) and one for the computation of \( v \) – were defined for the rule base learning method. The domains \( x \) and \( v \) were partitioned by three Gaussian-like fuzzy sets (neg, zero, pos), while for each subsystem and delay independent fuzzy sets were defined with similar initial parameters. The use of independent fuzzy sets for output and (time-delayed) input is necessary to allow each system to scale between the output and input values. The rule base learning method yielded the following rules:

\[
\begin{align*}
\text{IF (} x(t-1) \text{ IS neg2 AND } v(t-1) \text{ IS neg1) THEN (} x(t) \text{ IS neg0) }\\
\text{IF (} x(t-1) \text{ IS neg2 AND } v(t-1) \text{ IS zero1) THEN (} x(t) \text{ IS neg0) }\\
\text{IF (} x(t-1) \text{ IS neg2 AND } v(t-1) \text{ IS pos1) THEN (} x(t) \text{ IS zero0) }\\
\text{IF (} x(t-1) \text{ IS zero2 AND } v(t-1) \text{ IS neg1) THEN (} x(t) \text{ IS neg0) }\\
\text{IF (} x(t-1) \text{ IS zero2 AND } v(t-1) \text{ IS zero1) THEN (} x(t) \text{ IS zero0) }\\
\text{IF (} x(t-1) \text{ IS zero2 AND } v(t-1) \text{ IS pos1) THEN (} x(t) \text{ IS pos0) }\\
\text{IF (} x(t-1) \text{ IS pos2 AND } v(t-1) \text{ IS neg1) THEN (} x(t) \text{ IS neg0) }\\
\text{IF (} x(t-1) \text{ IS pos2 AND } v(t-1) \text{ IS zero1) THEN (} x(t) \text{ IS zero0) }\\
\text{IF (} x(t-1) \text{ IS pos2 AND } v(t-1) \text{ IS pos1) THEN (} x(t) \text{ IS pos0) }\\
\text{IF (} x(t-1) \text{ IS neg1 AND } v(t-1) \text{ IS neg2) THEN (} v(t) \text{ IS zero0) }\\
\text{IF (} x(t-1) \text{ IS neg1 AND } v(t-1) \text{ IS zero2) THEN (} v(t) \text{ IS pos0) }\\
\text{IF (} x(t-1) \text{ IS neg1 AND } v(t-1) \text{ IS pos2) THEN (} v(t) \text{ IS pos0) }\\
\text{IF (} x(t-1) \text{ IS zero1 AND } v(t-1) \text{ IS neg2) THEN (} v(t) \text{ IS neg0) }\\
\text{IF (} x(t-1) \text{ IS zero1 AND } v(t-1) \text{ IS zero2) THEN (} v(t) \text{ IS zero0) }\\
\text{IF (} x(t-1) \text{ IS zero1 AND } v(t-1) \text{ IS pos2) THEN (} v(t) \text{ IS pos0) }\\
\text{IF (} x(t-1) \text{ IS pos1 AND } v(t-1) \text{ IS neg2) THEN (} v(t) \text{ IS neg0) }\\
\text{IF (} x(t-1) \text{ IS pos1 AND } v(t-1) \text{ IS zero2) THEN (} v(t) \text{ IS zero0) }\\
\text{IF (} x(t-1) \text{ IS pos1 AND } v(t-1) \text{ IS pos2) THEN (} v(t) \text{ IS pos0) }
\end{align*}
\]

If we interpret this rule base, we see that the learned rule base of the subsystems for the computation of \( x \) and \( v \) are similar to the rule base of a simple integrator (adder). So, we can suppose, that \( x \) is computed by integration of \( v \), and \( v \) by integration of (negated) \( x \). If we compare this interpretation to Equation (8) we see, that this behavior corresponds to the physical model.
After rule base learning, the rule base was optimized for 4000 training cycles (complete propagations through time) using $x$ and $v$ as training data. Figure 6 depicts the learning progress. The learning terminated with a summed square error of $E = 12.872$, which is caused by the deviation from the frequency and amplitude of the physical system. In Figure 5 the position $x$ and velocity $v$ of the trained system and the validation data over time is shown ($x$ and $v$ given).

To evaluate the performance of the presented approach for inner variables, in following runs either $x$ or $v$ was presented to the system for learning, thus the other variable was considered as unknown. The simulation run using just $x$ for training yielded the best approximation of the amplitudes for $x$ and $v$, while the system which was trained by use of $v$ fits the frequencies (see Figure 5). Both approaches achieved an appropriate approximation of the physical model. The performances of the resulting models are similar to the model that was trained using both variables ($x$ and $v$) as known output value for learning.

It has to be emphasized, that the only input to the system are the initial values $x(0)=1$ and $v(0)=0$. The dynamic behavior results from the reuse of the output signals as input (time-delayed feedback). Even during learning, the system used its own output as input and just the error signal $E_i(t)$ – the difference between the system output and training data – for learning.

Discussion

In the preceding section we have discussed the application of the proposed neuro-fuzzy system to a simple dynamical system to show the principle usability for the approximation of dynamic systems. The analysis covered identification of hierarchical as well as recurrent rule structures.

In the same way, the model can be applied to decision making problems:

- If prior knowledge is available, this can be used to define an initial rule base.
- Using rule templates, existing training data can be used to learn (additional) fuzzy rules.
- The membership functions used to define an existing or learned fuzzy rule base can be optimized using available training data.
- The obtained fuzzy rule base can be analyzed to obtain information about the analyzed data set or to justify the decision process.

However, for the application to decision making problems we have to consider, that the proposed model is based on local subsystems and the inner variables are defuzzified before used in a succeeding layer (subsystem). Thus, we neglect information about the specificity or vagueness of an attribute in the inference process. Furthermore, we can not forward alternative solutions to succeeding layers and we therefore have to avoid contradicting rules (e.g. implementing alternative decisions).

To further improve the usability of the discussed system, the structure of the neuro-fuzzy model could be extended so that also symbolic attributes can be used directly. Currently, these attributes have to encoded as real values (if an order can be defined) or as a binary vector. Furthermore, the model should be extended to support the use of patterns with missing values. Here, approaches similar to the methods discussed in (Nauck, and Kruse 1999) might be appropriate. These aspects will be considered in future work.

Conclusions

In this article we have discussed hierarchical recurrent fuzzy systems that tackle the complexity problems of pure feed-forward architectures and enable the approximation of dynamic processes by storing information of prior system states internally.
Furthermore, we have discussed a neuro-fuzzy model that can be used to optimize hierarchical recurrent fuzzy rule bases. The model allows the use of prior knowledge in form of fuzzy rules or rule templates, e.g. to define an appropriate initial solution or to enhance the learning performance. The learning methods ensures the interpretability of the optimized rule base by use of shared weights (fuzzy sets) and constraints.

The proposed neuro-fuzzy system can be a valuable tool to (interactively) optimize arbitrary hierarchical recurrent fuzzy rule bases. However, we have to be aware that even if we use neuro-fuzzy techniques for optimization, we still apply a fuzzy system as a matter of fact. In general, fuzzy systems cannot be meant for outperforming other approximation approaches. This is mainly prevented by the usually small numbers of linguistic terms that are shared by all rules to ensure an interpretable solution. This trade-off between (often quantitative) methods that achieve good performance and (often qualitative) models that give insights into a specific domain or problem was formulated as the principle of the incompatibility of precision and meaningfulness by Zadeh (Zadeh 1973). The benefit gained by using a fuzzy system lies in interpretability and readability of the rule base. This is widely considered more important than the ‘last percent’ of increase in performance.

Acknowledgements

The work presented in this article was partially supported by BTexact Technologies, Adastral Park, Martlesham, UK.

References


