Fuzzy Rule Extraction by a Genetic Algorithm and Constrained Nonlinear Optimization of Membership Functions

Oliver Nelles, Martin Fischer, and Bernd Müller
Technical University of Darmstadt, Institute of Automatic Control
Laboratory of Control Engineering and Process Automation
Landgraf-Georg-Str. 4, D-64283 Darmstadt, Germany
Phone: +49/6151/16-4524, Fax: +49/6151/293445
E-mail: `nelles@irt1.rt.e-technik.th-darmstadt.de`

Abstract

In this paper we propose a new method for fuzzy rule extraction from data by a genetic algorithm and a fine tuning of the extracted membership functions by a constrained nonlinear optimization. This approach is able to select the most significant rules out of a set of all possible ones, that is it learns the rule structure by itself. The genetic algorithm does not limit the kind of operator and the number and form of the membership functions for the inputs. However, in order to utilize linear optimization techniques, singletons and center of gravity defuzzification are used on the output side. Since each rule premise may include a conjunction of a variable number of inputs (between one and the input dimension), the "curse of dimensionality" [1] can be overcome, that is the number of rules does not increase exponentially with the input dimension. This feature makes the proposed algorithm especially attractive for interpretation of high dimensional nonlinear mappings that are hard to visualize. The strategy followed by the nonlinear optimization of the fuzzy input membership functions focuses on a good interpretability rather than on best approximation performance. This will be demonstrated on a real world data example.

1. Introduction

In the last few years much attention has been paid to the construction of fuzzy systems that are able to learn. The main reason for the attractiveness of such systems is the combination of interpretable rules and adaptation to data. On one side, convergence and therefore speed of learning can be considerably improved by a priori knowledge in form of fuzzy rules [2]. The possibility to incorporate all a priori knowledge into a system is always a highly desirable feature. On the other hand, missing knowledge can be compensated by building a data driven model and coarse knowledge can be refined by adaptation.

There are two main directions of research on learning fuzzy systems, the neuro-fuzzy or fuzzy-neuro approaches that try to combine neural network architecture with fuzzy systems [2], [3] and the evolutionary optimization approaches using genetic algorithms (GA) or evolutionary strategies that advance the problem in a more direct manner [4]. Usually neuro-fuzzy systems are trained like neural networks by gradient based local optimization algorithms, e.g. backpropagation. Therefore, they often are limited to differentiable membership functions and fuzzy operators, like Gaussians and product operator for conjunction respectively. GAs and other global optimization techniques typically do not impose any restrictions of that kind. Thus, for the rule extraction method presented in this paper any kind of input membership function and any t-norm can be applied.

2. Classification of Learning Fuzzy Systems

It is very important to distinguish the existing neuro-fuzzy and GA-fuzzy systems by the kind of parameters that are subject to the optimization. It is suggested to classify all approaches in the following three categories:

1. *Structure learning:* The terms in the premise of each rule are learned. Optionally the number of rules is learned as well. This leads to the combinatorial optimization problem to choose a set of significant rules. Combinatorial optimization may be performed by a GA.

2. *Input membership function learning:* The membership functions can be described by functions with unknown parameters, e.g. Gaussians with position and width as free parameters. These parameters are optimized. This can be performed by global (e.g. GA) or local (e.g. backpropagation) nonlinear optimization.

3. *Output membership function learning:* If singletons are used on the output side, this reduces to the problem of finding the singletons' positions. These positions influence the output of the fuzzy system in a linear way. Therefore, this problem can be solved by a standard linear least-squares (LS) technique.

The complexity of the problems decreases from category 1 to 3. Of course all three strategies can be combined in various ways. Approach 3 is simple and most straight forward. It benefits from all advantages of linear optimization, that is convergence to the global minimum, very low computational cost and possible recursive formulation. However, its flexibility is limited, since all input membership functions are fixed. This restriction can be overcome by approach 2, that may be efficiently combined with approach 3. However, due to the nonlinear nature of this problem an iterative nonlinear optimization algorithm must be applied. If convergence to bad local minima should be avoided, GAs may be used. Furthermore, the search should be constrained in such a way, that interpretability is not lost, e.g. by moving membership functions out of the interval of physical meaning.

Both approaches 2 and 3 are the basis of most neuro-fuzzy or GA-fuzzy systems. Since the number of rules of a complete rule set is equal to

$$\prod_{i=1}^{n} m_i \tag{1}$$

where m_i is the number of membership functions for input i and n is the number of inputs, they underlie the "curse of dimensionality". Thus, those approaches become impractical for higher input dimensions for two reasons. First, the number of parameters to be estimated becomes too large (the more parameters the larger the estimation variance [5]). Second, even if the parameters could be estimated due to a huge amount of high quality data and fast computers the number of rules would be too large to allow an easy interpretation. Thus, for systems with many (more than two or three) inputs those approaches are not manageable. Therefore in this paper a new method for structure learning is developed, that overcomes these problems. It can be categorized as a combination of all the above mentioned approaches, whereas approach 1 and 3 are used in the genetic algorithm in order to learn the structure of the fuzzy system and approach 2 and 3 are applied afterwards for the tuning of the input and output membership functions.

For rule extraction all possible rules are coded in a binary string. The length of this binary string is equal to the number of all possible rules and selected rules are represented by setting the corresponding gene to "1" while not selected rules are symbolized by a "0". Thus, the number of "1"s in each binary string is equal to the number of selected rules. These binary strings are subject to optimization of the GA introduced in the following section.

3. Genetic Algorithms

Genetic algorithms [6] are probabilistic search methods that employ a search technique based on ideas from natural genetics and evolutionary principles. They were conceived by Holland [7] in 1973 and since then, they have emerged as general purpose, robust optimization and search techniques. In contrast to gradient based algorithms the GAs' strength is the global character that prevents them from being trapped in local optima, but of course GAs cannot guarantee to find the global optimum in finite time. However, usually at least a good local optimum can be discovered.

Structure learning can be formulated as the combinatorial optimization problem to find the significant rules out of a large given rule set. Since this problem can be naturally formulated in terms of binary strings, GAs seem to be perfectly suited. The rule set of all possible M rules is coded as shown in figure 1a. Since each bit has its own interpretation, i.e. it activates or deactivates a rule, crossover and mutation make physical sense. Mutation switches rules on or off (figure 1b). Crossover combines parts of the rule set of one individual with parts of the rule set of another (figure 1c). These are desired features that should lead to fast convergence of the algorithm.

For the examples presented in this paper a population size of 30 and a crossover probability of 0.9 was applied. The mutation rate was not determined for each bit, but for each individual. Each rule set on average was mutated with a probability of 0.2. The mutation probability can be calculated by dividing the individual mutation probability by the number of possible rules M.

The fitness of each individual was evaluated in the following way. First, the binary string was translated to the selected rules by finding the rule numbers that are set to "1". For these rules a least-squares (LS) optimization of the singletons is performed [8]. Then the normalized mean square error of this fuzzy system is evaluated. A penalty function that is proportional to the number of rules selected is added as well as another penalty function for singletons that have no physical meaning. The inverse of this loss function value is the fitness of the corresponding rule set. The penalty factor that determines how strong large rule sets should be penalized is chosen by the user. Large penalty factors will lead to small rule sets, while small penalty factors will lead to large rule sets. The penalty for the singletons is calculated in the following way: A range is given for every output by the minimum and maximum values of the corresponding output data. This range is expanded by a percentage factor set by the user. In this paper the factor 0.2 is used, so that the singletons are allowed to exceed the output range by ± 20 percent. Singletons that exceed these bounds lead to an additional penalty term. This penaltiy term is equal to the distance of the singletons to the violated range limit. This procedure controls the conformity of the learned structure with the given by the data.

Although an LS optimization for each fitness evaluation is time-consuming, this approach guarantees a linear optimized fuzzy rule set for each individual and therefore leads to fast convergence. Coding the singleton positions within the GA would ignore the information about the linear dependency of the output on these parameters. In order to further accelerate the fitness evaluation a maximum number of rules can be chosen by the user. Rule sets with a larger number of selected rules than this maximum are not LS optimized. Instead a fitness value of zero is returned. The GA will find a good or even the optimal rule set corresponding to the penalty value with a rule set size between one and the maximum number of rules.

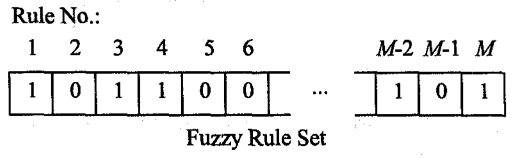

Figure 1a. Binary string for rule set coding.

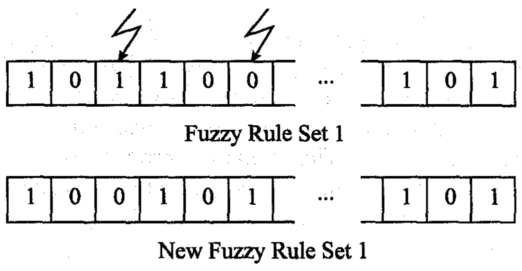

Figure 1b. Mutation operator.

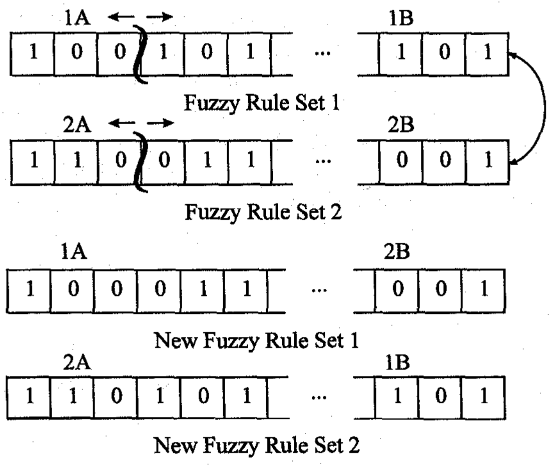

Figure 1c. Crossover operator.

4. Rule Extraction

For sake of simplicity, the rule extraction process will be demonstrated for a system with only two inputs x_1 and x_2, one output y and three membership functions for each input M_{11}, ..., M_{13} and M_{21}, ..., M_{23}, respectively. The set of all possible rules from which the extration process selects the significant ones contains 15 rules, that is 3 rules with x_1 in the premise only, 3 rules with x_2 in the premise only and 9 rules with all combinations of x_1 and x_2 in the premise. It is important to notice that such rules as (see figure 2)

IF $x_1 = M_{12}$ THEN $y = ?$ (rule 1)

cover the same area as the three rules

IF $x_1 = M_{12} \wedge x_2 = M_{21}$ THEN $y = ?$ (rule 2)

IF $x_1 = M_{12} \wedge x_2 = M_{22}$ THEN $y = ?$ (rule 3)

IF $x_1 = M_{12} \wedge x_2 = M_{23}$ THEN $y = ?$ (rule 4)

The singletons "?" will be optimized by the LS minimization. If the singletons of the rules 2-4 are about the same value, those three rules can be approximated by the first one. This is the mechanism to overcome the "curse of dimensionality". Generally speaking, one rule with just one term in the premise can cover the same area as m^{n-1} rules with n terms in the premise, where m is the number of membership functions for each input. Therefore, the number of rules required can be drastically reduced and this reduction effect increases exponentially with the input dimension n. Furthermore, a few rules with a small number of terms in the premise are much easier to interprete than many rules with complete premises.

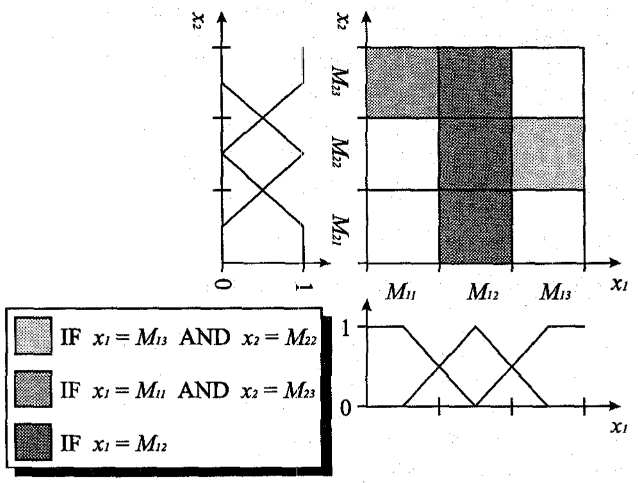

Figure 2. Three rules for a fuzzy system with two inputs.

This kind of information compression is the only way to handle high-dimensional problem with fuzzy systems, that relay on one-dimensional membership function.

For the case of n inputs, m membership functions per input (of course they are in general allowed to be different in shape and number for each input) the number of possible rules is equal to

$$\underbrace{\binom{n}{1}m^1}_{1\text{ term}} + \underbrace{\binom{n}{2}m^2}_{2\text{ terms}} + \ldots + \underbrace{\binom{n}{n}m^n}_{n\text{ terms}} \qquad (2)$$

Conventional neuro-fuzzy and GA-fuzzy systems only consider the last term of the sum in (2), that is dominant for $m>n$. With the proposed approach it is, in principle, possible for the GA to detect, if an input x_j is irrelevant, since then x_j will not appear in the optimized rule set, if the global optimium is found.

Since under some conditions a fuzzy system is equivalent to an RBF network [8] if all n terms appear in all premises, it is interesting to ask for an interpretation of premises with less than n terms from the neural network point of view. Indeed, in [9] so called Gaussian bar units are proposed for radial basis function networks to overcome the "curse of dimensionality". Those Gaussian bar units corespond to n 1-term premises as proposed in this paper from the fuzzy point of view. In [9] experimental results are given in order to proof, that this semilocal approach is much less sensible to the problem of "curse of dimensionality" than the pure local method.

However, the approach in [9] includes only 1 and n-term premises and gives no rule interpretation of the RBF network at all. The method presented here allows any number of terms in each rule premise ranging from 1 to n. This can be seen of a decreasing degree of locality, since a rule with 1 term in the premise covers $1/m$, a rule with 2 terms in the premise covers $1/m^2$ and a rule with n terms in the premise covers only $1/m^n$ of the input space. Therefore, the GA also controls the degree of localness of the fuzzy system for all input region separately.

5. Constrained Nonlinear Optimization

In order to tune the input membership functions, a nonlinear optimization problem has to be solved. The parameters of the optimization are the centers and widths (standard deviations) of the Gaussian membership functions. If the approximation quality is the only objective of optimization, the search space of the parameters to be optimized is not limited. Though the approximation quality for such an approach can expected to be quite good, the interpretability of the fuzzy rules may get lost. This is due to the fact, that if the range of the membership functions is not restricted, often widely overlapping membership functions give good numerical results. Fuzzy membership functions as shown in figure 7 lacking any physical interpretation and loosing locality are possible. To avoid this, different kinds of constraints should be put on the optimization: equality constraints, inequality constraints and parameter bounds. Another strategy is to add a penalty measure to the objective function. However, this normally reduces the efficiency of the optimization algorithm [10].

In order to efficiently solve this constrained nonlinear optimization problem in which the objective function and constraints may be nonlinear functions of the variables, a sequential quadratic programming (SQP) algorithm as presented in [11] is applied. It iteratively solves the Kunh-Tucker equations and builds up second order information for fast convergence.

6. Tuning of the Fuzzy Membership Functions

To increase the performance of the extracted rule set the following strategy is applied: Optimization of the input membership functions by a sequential quadratic programming (SQP) algorithm in which an LS optimization of the output membership functions is embedded. The objective function of the optimization is the normalized mean square error. To prevent a large overlap or even coincidental membership functions, different strategies were used:

1. *Minimum distance of membership functions:* The centers of the Gaussian membership functions must have a minimum distance to the centers of the adjoining membership functions.
2. *Parameter bounds:* The centers' values as well as the widths' values of the Gaussians of each membership function are limited within a given range.
3. *The sum of the optimized membership functions for each input should be around one:* The original normalized membership functions sum up to one. This is an appealing property that makes human interpretation easier. Thus the squared difference between one and the sum of the optimized membership functions is integrated. This value is a penalty which is weighted with a sum penalty factor and then added to the objective function.

Strategy 3 has turned out to be the most powerful one in terms of interpretation quality but also the most restrictive. To further increase the interpretation quality a penalty factor for singletons without physical meaning as explained in section 3 is used. In almost every combination of the different strategies this penalty factor leads to faster convergence of the optimization problem.

7. Real World Data

Figure 3 shows the relationship of exhaust-gas pressure dependent on the engine speed and the injection mass for a diesel engine in a lookup-table. The 320 (32x10) data points have been measured at a diesel engine test stand.

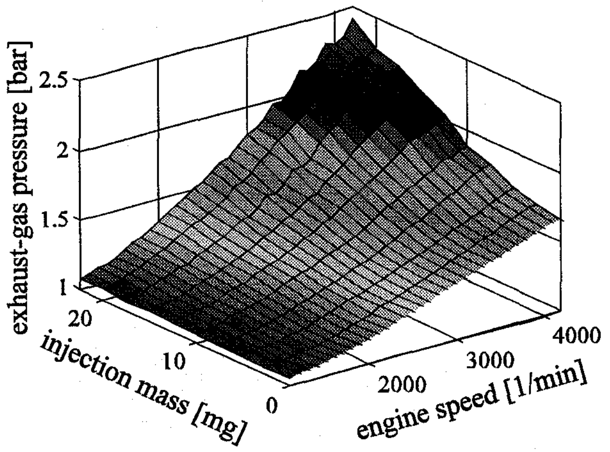

Figure 3. Measured lookup-table.

Since the relationship seem to be quite smooth only four (*very small, small, high, very high*) normalized Gaussian membership functions with considerable overlap were placed on each input axis. The resulting rule set contains 24 (4+4+16) possible rules. For a high penalty factor FUREGA leads to the following rule set of only four rules:

IF	speed = *small*	
THEN	exhaust-gas pressure = 1.113 bar	(rule 1)
IF	speed = *high*	
THEN	exhaust-gas pressure = 1.696 bar	(rule 2)
IF	speed = *very small* $\wedge$ injection = *very small*	
THEN	exhaust-gas pressure = 1.012 bar	(rule 3)
IF	speed = *very high* $\wedge$ injection = *very high*	
THEN	exhaust-gas pressure = 2.566 bar	(rule 4)

It has to be stated clearly, that this rule set is the best result obtained during a few runs of the GA and it could not always be reached. It is very easy to interpret and has the nice property that the more relevant input "engine speed" is used for all rules, while the less relevant input "injection mass" only appears in two rules in combination with the other input. Figure 4 shows the lookup-table generated by those four selected fuzzy rules with a normalized mean square error of 0.0243 and figure 5 shows the corresponding fuzzy membership functions for the input "engine speed". The tuning of the membership functions (see figure 7) without constraints leads to a normalized mean square error of 0.0018. The upper curve is the sum of all membership functions for this input. The corresponding approximation performance is shown in figure 6. By this optimization interpretability gets lost. For example the optimized singleton for rule 1 is 0.390 bar. This value has nothing to do with the output range of the lookup-table which varies approximately between 1 bar and 2.5 bar.

There are two more figures which show examplary the tuned membership functions of the input "engine speed". All constraints mentioned in section 6 were active. Only the sum penalty factor for strategy 3 was varied. Figure 8 shows the optimized membership functions for a sum penalty factor of 0.01. The singletons vary between 1.032 bar and 2.542 bar. Figure 9 shows the optimized membership functions for a larger sum penalty factor of 0.1. Here the singletons vary between 1.047 bar and 2.537 bar. The normalized mean square error was 0.0077 and 0.0144, respectively. This is a considerable improvement compared to the result of the GA.

As the range of the singletons and the Gaussian membership functions show, the results are easily interpretable. These results clearly show that one has to pay for increasing interpretability by decreasing performance. In this example nonlinear optimization without constraints could improve the approximation accuracy by a factor of 3.5, while improvement with constraints was between 1.8 and 1.3 dependent on the sum penalty factor (these numbers correspond to the normalized root mean square error ratios).

8. Conclusions

A new approach of learning fuzzy systems was developed. The key feature of this method is the fuzzy rule extraction by a genetic algorithm, called FUREGA. It is combined with a linear least-squares optimization of the singletons on the output side. It was shown that the "curse of dimensionality" can be overcome if rules with fewer terms in the premise are included. FUREGA leads to very small rule sets that are easy to interpret. This interpretability could be kept after nonlinear optimization of the fuzzy membership functions if constraints were applied. It has been shown that choosing the constraints and penalty factors allows a trade-off between performance and interpretability. The performance of FUREGA was demonstrated for a measured lookup-table for diesel engines.

References

[1] Bellman R.E.: "Adaptive Control Processes", Princeton University Press, 1961

[2] Brown M., Harris C.: "Neurofuzzy Adaptive Modelling and Control", Prentice Hall, 1994

[3] Ayoubi M.: "Rule Extraction for Fault Diagnosis with a Neuro-Fuzzy Structure and Application to a Turbocharger", Second European Congress on Fuzzy and Intelligent Techniques, EUFIT, 1994

[4] Trift P.: "Fuzzy Logic Synthesis with Genetic Algorithms", Forth International Conference on Genetic Algorithms, 1991

[5] Isermann R.: "Identifikation dynamischer Systeme I", 2. ed., Springer, 1992 (in German)

[6] Goldberg D.E.: "Genetic Algorithms in Search, Optimization and Machine Learning", Reading Massachusetts, Addison-Wesley, 1989

[7] Holland J.H.: "Adaptive in Natural and Artificial Systems", Ann Arbor, University of Michigan Press, 1973

[8] Kecman V., Pfeiffer B.M.: "Learning Fuzzy Rules Equals Radial Basis Function Neural Network Training", IEEE World Congress on Computational Intelligence, 1994

[9] Hartman E., Keeler J.D.: "Predicting the Future: Advantages of Semilocal Units", Neural Computation 3, pp. 566-578, 1991

[10] Gill P.E.: "Practical Optimization", Academic Press, 1988

[11] Grace A.: "MATLAB Optimization Toolbox User's Guide", The MATHWORKS Inc., 1994

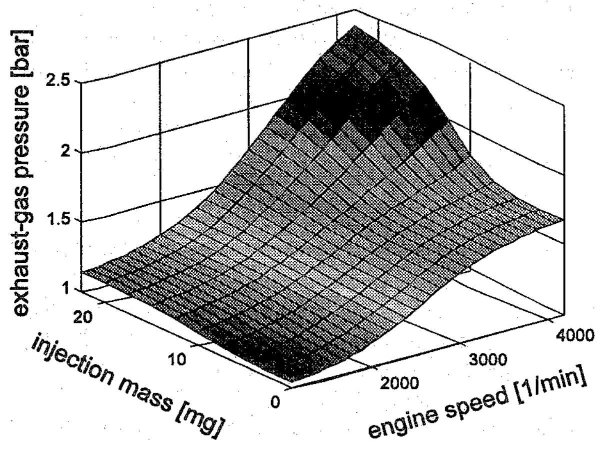

Figure 4. Approximation with four rules.

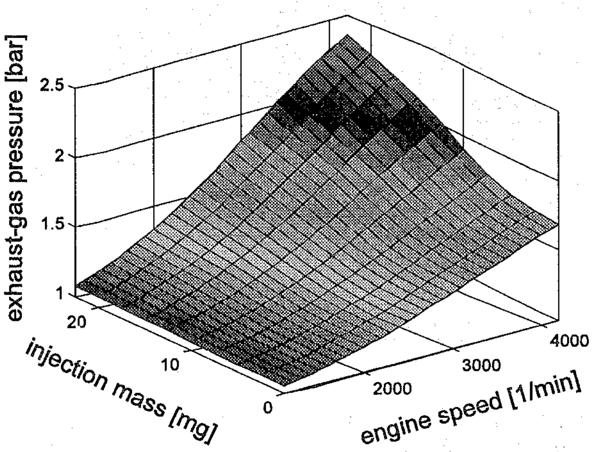

Figure 6. Approximation with four rules after nonlinear otimization without constraints.

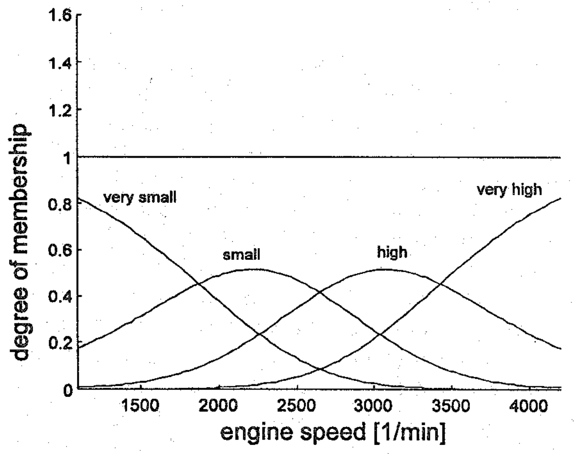

Figure 5. Membership functions for the input "engine speed" before optimization.

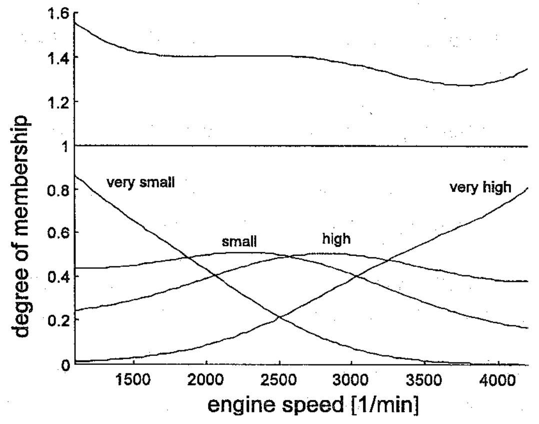

Figure 7. Membership functions for the input "engine speed" after nonlinear optimization without constraints.

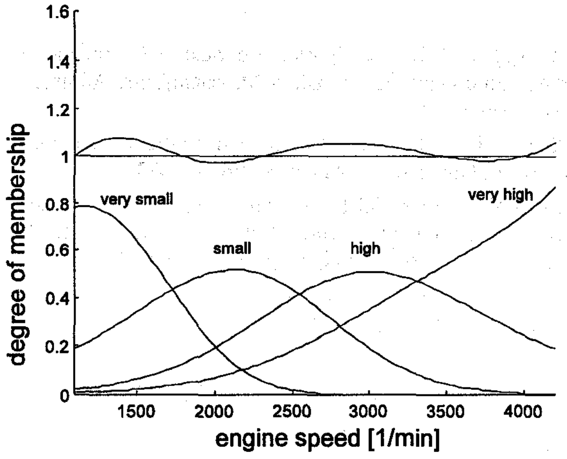

Figure 8. Membership functions for the input "engine speed" after nonlinear optimization with constraints and a sum penalty factor of 0.01.

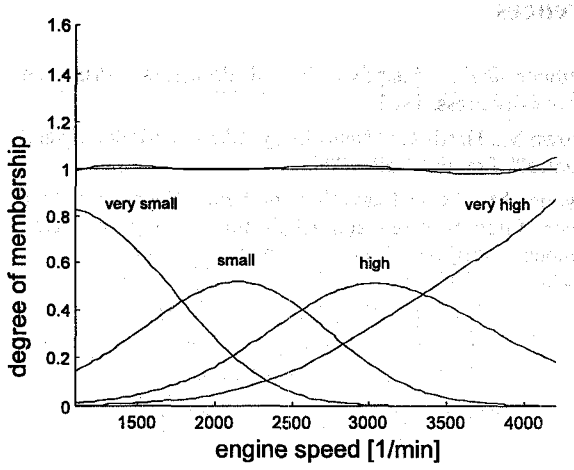

Figure 9. Membership functions for the input "engine speed" after nonlinear optimization with constraints and a sum penalty factor of 0.1.