to be published in:
Neural Processing Letters 10(3), 1999.

Knowledge Incorporation into Neural Networks From Fuzzy Rules

Yaochu Jin[†] (`yaochu.jin@alliedsignal.com`)
Dept. of Industrial Engineering
The State University of New Jersey
Piscataway
New Jersey, USA

Bernhard Sendhoff
(`bernhard.sendhoff@neuroinformatik.ruhr-uni-bochum.de`)
Institut für Neuroinformatik
Ruhr-Universität Bochum
D-44780 Bochum, Germany

Abstract. The incorporation of prior knowledge into neural networks can improve neural network learning in several respects, for example, a faster learning speed and better generalization ability. However, neural network learning is data driven and there is no general way to exploit knowledge which is not in the form of data input-output pairs. In this paper, we propose two approaches for incorporating knowledge into neural networks from fuzzy rules. These fuzzy rules are generated based on expert knowledge or intuition. In the first approach, information from the derivative of the fuzzy system is used to regularize the neural network learning, whereas in the second approach the fuzzy rules are used as a catalyst. Simulation studies show that both approaches increase the learning speed significantly.

1. Introduction

Conventional neural network learning algorithms are based solely on the available data examples. Knowledge about the system which is not in the form of input-output data pairs cannot usually be exploited. In real applications this restriction can be problematic. The data acquisition is usually expensive and/or the data is inherently noisy and its reliability therefore limited. Furthermore, if the whole system state cannot be measured and a reconstruction process has to be applied, the number of independent data pairs is reduced further. On the other hand, more general knowledge about the system is often available in linguistic form. The optimal exploitation of this knowledge for network learning can reduce the number of required data sets and speed up the learning process. Therefore, the transfer of knowledge, some-

---

[†] Also affiliated with the Dept. of Electrical Eng., Zhejiang University, Hangzhou, P.R. China.

times termed hints, which is not in the usual data format, into neural networks is currently receiving increasing attention.

Different forms of prior knowledge may be available. The most commonly used prior knowledge is called the invariance hint (Abu-Mostafa, 1993b). This hint suggests that $f(x) = f(x')$ for some x and x'. Another type of hint is the monotonicity hint (Sill and Abu-Mostafa, 1997), which implies that the function to be learned is monotonically increasing or decreasing within a certain range. In addition, the minimum Hamming distance (Al-Mashouq and Reed, 1991) and knowledge about the derivative of the input-output function of the system (for example, it is straightforward to know that if the input increases, the output increases too) can be used as prior knowledge. According to the different types of available knowledge, different cost functions have been suggested as a regularization term that is added to the standard error function of the neural network learning algorithm.

If we take a closer look at the learning process in humans, we find that it can also be helpful if several related tasks are learned at the same time. Hence, the second main approach to the incorporation of knowledge into neural networks is to let the neural network learn several related tasks simultaneously (Caruan, 1995). These additional tasks are called catalytic hints (Abu-Mostafa, 1993a).

Theoretical analysis of some simple problems has shown that adding extra knowledge or hints to the learning process improves the performance of neural networks after and during learning. Abu-Mostafa (1993a) and Barber (1996) have shown that hints or extra knowledge improve the generalization ability of neural networks.

The hints we mentioned above are not very general. A more general approach to using additional information besides data pairs for network learning would be to express this information in a fuzzy rule system (Zadeh, 1973) and to use the fuzzy system as a regularization term or a catalytic hint for network learning. In this way, a more general structure for knowledge representation, which is not based solely on data pairs, can be exploited for the adaptation of a neural network. The knowledge may come from an expert, from intuition or from some intuitive analysis of the system. The kinetics of a robot manipulator serves as an example in this work, since intuitive fuzzy rules are easily generated (Section 2) which can be exploited for network learning. The fuzzy system captures the general behaviour of the system (robot manipulator) and can be used as a regularization term (Section 3) and as a catalytic hint (Section 4) for network learning. In Section 5, the results from both methods will be presented and analyzed. We conclude the paper with a summary of our findings.

2. Knowledge Representation with Fuzzy Rules

In many areas of application *more general* knowledge about the behaviour of a system is available (i.e. from experts) which cannot be described by input–output data pairs. Fuzzy systems often are a more appropriate structure for the representation of this kind of knowledge. The IF-THEN rule base in fuzzy systems is believed to be more compatible with the way high level decision processes in humans are reached. Furthermore, the variables used in fuzzy rules are linguistic variables which are more appropriate to represent general knowledge (i.e. knowledge about the main elements rather than limited details). In this section we deduce several fuzzy rules for the simple robot manipulator with two links. From Figures 1 (a), (b) and 2 (a) and from “common sense” we can write down the following nine fuzzy rules, which we will use in the next sections to supplement the purely data based learning of the neural network.

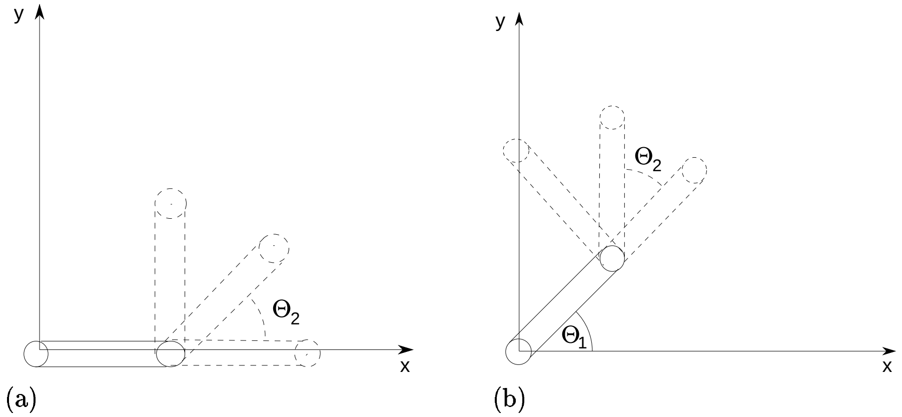
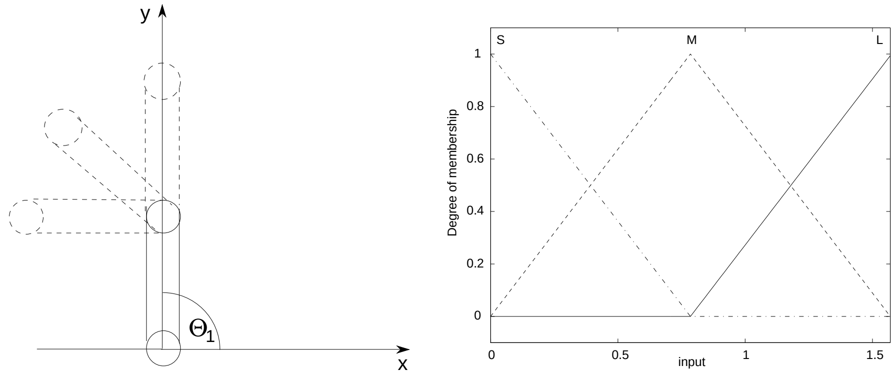


Figure 1. The fuzzy kinematics (a) Θ_1 `small` and (b) Θ_1 `medium`. The universe of discourse for the inputs is $[0, \frac{\pi}{2}]$.

1. **If** θ_1 is `small` and θ_2 is `small` , **then** y is `small`;
2. **If** θ_1 is `small` and θ_2 is `medium`, **then** y is `quite small`;
3. **If** θ_1 is `small` and θ_2 is `large`, **then** y is `medium`;
4. **If** θ_1 is `medium` and θ_2 is `small`, **then** y is `quite large`;
5. **If** θ_1 is `medium` and θ_2 is `medium`, **then** y is `large`;
6. **If** θ_1 is `medium` and θ_2 is `large`, **then** y is `quite large`;
7. **If** θ_1 is `large` and θ_2 is `small`, **then** y is `very large`;
8. **If** θ_1 is `large` and θ_2 is `medium`, **then** y is `large`;
9. **If** θ_1 is `large` and θ_2 is `large`, **then** y is `medium`

For the formulation of these fuzzy rules we used three linguistic terms (`small`, `medium` and `large`) for the two linguistic input variables



(a) (b)

Figure 2. (a) The fuzzy kinematics Θ_1 `large`. (b) Membership functions for the inputs

and six linguistic terms (`small`, `quite small`, `medium`, `quite large`, `large` and `very large`) for the linguistic output variables. We note that these fuzzy rules have been obtained solely from observation of the space relationships of the manipulators, without knowledge of the mathematical relations.

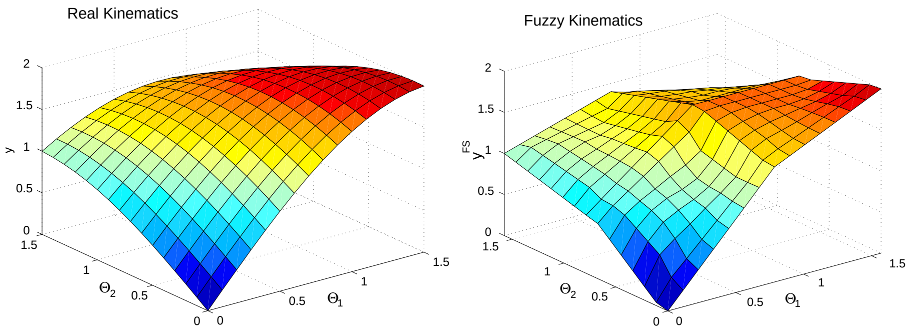
Before these fuzzy rules can be used in neural network learning, proper membership functions must be defined. The definition of the membership functions is based on knowledge from human experts. A straightforward choice is to use triangular membership functions as shown in Figure 2 (b). For the sake of simplicity, we use fuzzy singletons for the output variable; they are approximately set[1] as [0.0, 0.7, 1.0, 1.4, 1.7, 2.0] assuming that the length of each link is 1. Figure 3 (a) and (b) show the input-output surface described by the real system and by the fuzzy system.

3. Regularization with Fuzzy Derivative

In this section, we will use the information from the derivative of the fuzzy system to regularize the neural network learning. This *fuzzy derivative information* can be seen as additional knowledge to be learned by the neural network. Without the knowledge about the derivative, the neural network learns only the information at each sample point. If the derivative information at these points is given, the neural network can additionally learn information about the behaviour of the system

---

[1] The first value has to be zero in order to guarantee that the height is zero if both angles are zero.



(a) (b)

Figure 3. The kinematics of the real system (a) and the fuzzy system (b)

between sample points.

Regularization is a method for constrained learning of neural networks. Originally, regularization was used to improve the generalization ability of neural networks based on empirical or Bayesian methods (Bishop, 1995). In our work, we use the fuzzy derivative for regularization primarily to provide extra knowledge to the neural network. Similar to conventional regularization algorithms, we add an extra term to the quadratic error function:

$$J = E + \lambda \Omega \,. \tag{1}$$

E is the conventional quadratic error function:

$$E = (y^{NN} - y^t)^2, \tag{2}$$

where y^{NN} is the output of the neural network and y^t is the target output. In equation (1) λ is the regularization coefficient, which is usually much smaller than 1 and has to be set manually. Ω is the regularization term describing the difference between the partial derivatives of the neural network and the target partial derivatives calculated from the fuzzy rule system:

$$\Omega = \sum_{i=1}^{n} \left(\frac{\partial y^{NN}}{\partial x_i} - \frac{\partial y^{FS}}{\partial x_i} \right)^2 , \tag{3}$$

where i is the number of inputs.

In order to derive the learning algorithm, we assume the neural network is given in the following form:

$$y^{NN} = \sum_{j=1}^{H} v_j f(\sum_{i=1}^{n} w_{ij} x_i). \tag{4}$$

Equation (4) describes a standard feed-forward neural network with one hidden layer and H hidden nodes. The activation function for the output node is linear, whereas the activation function for the hidden neurons is given by the following logistic function:

$$f(z) = \frac{1}{1+\exp(-z)}. \tag{5}$$

Now we can derive the learning algorithm according to the cost function in equation (1) using the gradient method. The differentiation of equation (1) with respect to the output weights v_j, results in

$$\frac{\partial J}{\partial v_j} = \frac{\partial E}{\partial v_j} + \lambda \frac{\partial \Omega}{\partial v_j} \tag{6}$$

In equation (6), the first term on the right side is the standard gradient term for neural network learning and will not be discussed further. For the second term, we derive:

$$\begin{aligned} \frac{\partial \Omega}{\partial v_j} &= \sum_{i=1}^{n} \left(\frac{\partial y^{NN}}{\partial x_i} - \frac{\partial y^{FS}}{\partial x_i} \right) \frac{\partial(\partial y^{NN}/\partial x_i)}{\partial v_j} \qquad (7) \\ &= \sum_{i=1}^{n} \left(\frac{\partial y^{NN}}{\partial x_i} - \frac{\partial y^{FS}}{\partial x_i} \right) \frac{\partial D}{\partial v_j}, \end{aligned}$$

where:

$$\frac{\partial D}{\partial v_j} = f'(.) w_{ij}. \tag{8}$$

Similarly, the learning algorithm for the input weights w_{ij} can be obtained:

$$\frac{\partial \Omega}{\partial w_{ij}} = \sum_{i=1}^{n} \left(\frac{\partial y^{NN}}{\partial x_i} - \frac{\partial y^{FS}}{\partial x_i} \right) \frac{\partial D}{\partial w_{ij}} \tag{9}$$

$$\frac{\partial D}{\partial w_{ij}} = v_j f'(.) + v_j w_{ij} f''(.) x_i, \tag{10}$$

where $f'(.)$ and $f''(.)$ are the first and second derivatives of the activation function, thus

$$f'(.) = f(1-f) \tag{11}$$

$$f''(.) = (1-2f)f(1-f). \tag{12}$$

In order to calculate the derivative error in equation (3), we have to calculate the derivative of the neural network with respect to its inputs x_i:

$$\begin{aligned}\frac{\partial y^{NN}}{\partial x_i} &= \sum_{j=1}^{H} v_j \frac{\partial f}{\partial x_i} \\ &= \sum_{j=1}^{H} v_j f'(.) w_{ij}.\end{aligned} \tag{13}$$

There are two different possibilities to calculate the derivative of the fuzzy system. The first approach is to use the analytical form of the fuzzy system and to apply the partial derivative to it.

The output of the fuzzy system described in Section 2 can be computed as follows:

$$y^{FS} = \frac{\sum_{i=1}^{9} y_i \; \min\left(A_i^1(\theta_1), A_i^2(\theta_2)\right)}{\sum_{i=1}^{9} \min(A_i^1(\theta_1), A_i^2(\theta_2))}, \tag{14}$$

where y_i is one of the six singletons specified in Section 2, $A_i^1(\theta_1)$ and $A_i^2(\theta_2)$ are one of the three membership functions for θ_1 and θ_2 respectively (see Figure 2 (b)). However, the analytical form of equation (14) is rather complicated especially when triangular fuzzy membership functions are used. An alternative approach is to calculate the derivative numerically:

$$\frac{\partial y^{FS}}{\partial x_i} \approx \frac{y^{FS}(x_i + \Delta x_i) - y^{FS}(x_i)}{\Delta x_i}, \tag{15}$$

where Δx_i should be selected appropriately.

In this section, we derived the regularized error function J for the neural network in equation (1) and its gradient in equations (9,10) incorporating the fuzzy derivative information. In Section 5, we will apply the proposed method to the problem of neural network learning of the manipulator relations introduced in the last section.

4. Fuzzy Systems As Catalysts

In the previous section, we introduced the fuzzy derivative as a regularization term for neural network learning. As a second approach to using fuzzy rules as additional knowledge for neural networks, we introduce the idea of catalysts. Catalytic hints were first proposed by Suddarth and Holden (1991) to improve the learning performance of feed-forward neural networks. Abu-Mostafa (1993a) has shown that by introducing catalytic hints into neural networks, the effective VC dimension of the problem will be reduced. The VC dimension (Blumer et al., 1989; Vapnik, 1995) indicates an upper bound for the number of examples needed by a learning process. The idea behind catalytic learning is shown in Figure 4. Here the output y of the output neuron is required to approximate the target output y^t. In addition, the output y_c of the augmented (catalyst) neuron is required to approximate the output, which is similar to the target output and which we will refer to as y_c^t. During neural network learning, these two output neurons are trained simultaneously. It has been shown that by adding the extra catalytic task the network can adapt the correct weights in the hidden layer for the original task represented by y^t more easily.

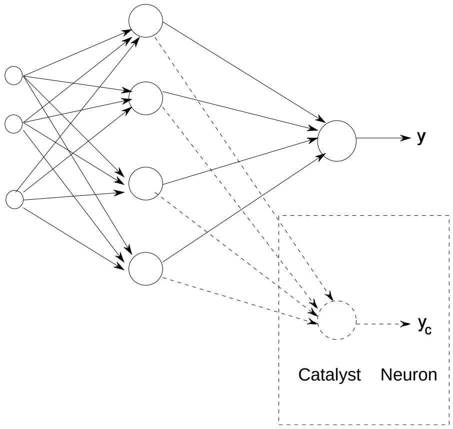


Figure 4. Catalyst learning.

Of course, the main problem is to find an appropriate, related task represented by y_c^t, which is then called the catalyst function. A possible choice for y_c^t is a simplified version of the original output target y^t. Fuzzy systems designed by experts which only capture the principles behind a certain task are therefore a sensible definition for y_c^t. In the next section, we will use the fuzzy system from Section 3 as a catalyst function for the neural network. This represents a second way to incorporate non data-based knowledge into the neural network and to support the standard network learning.

5. Simulation Results

In order to show the effectiveness of the two knowledge incorporation methods (Section 3 and 4), we implement these algorithms for learning a simple robotics task using neural networks. The robot considered in the following is a two-link planar manipulator as discussed in Section 2. The length of each link is set to 1 and the angle of each link varies between $[0, \frac{\pi}{2}]$. All samples are generated randomly. The neural network used is a feed-forward network with one hidden layer and 10 hidden nodes and is trained with standard back-propagation. The activation function for the hidden nodes is the logistic function and the output nodes are linear. In each run two data sets are randomly generated, one training set with 11 (insufficient[2] data) or 101 (sufficient data) samples, respectively, and a test set with 200 samples. The weights of the network with and without knowledge incorporation were initialized randomly in the interval $[-0.05, 0.05]$. All results presented in this section are the average of twenty different runs.

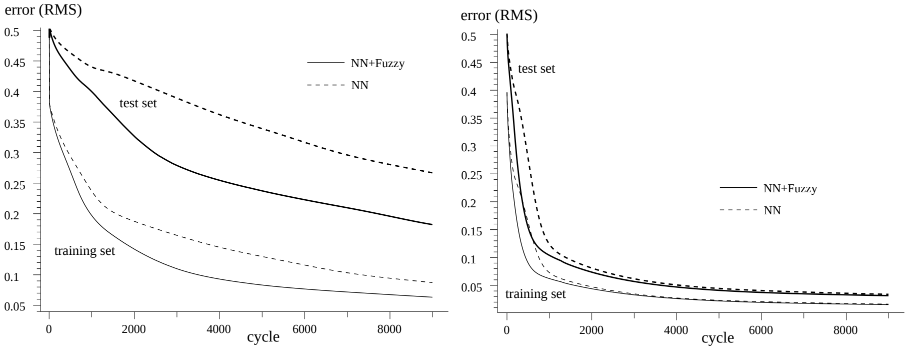
5.1. REGULARIZED LEARNING

Firstly, we consider the situation in which the available learning samples are insufficient. Eleven examples are generated for the training set and 9000 learning cycles are carried out. Figure 5 (a) shows the training and test errors with and without the fuzzy derivative regularization averaged over 20 runs. We observe that by incorporating knowledge from the fuzzy system the learning speed is significantly increased and the final error after 9000 learning cycles considerably reduced. The difference between the final error is larger on the test set than on the training set which corresponds well with the assumption that the incorporation of non-data based knowledge can enhance the generalization ability of the network.

If the available data is sufficient, as in Figure 5 (b) (101 samples), the final error is hardly reduced, however the effect on the learning speed remains. The learning speed increase is more evident in Figure 6, which shows the first 1500 cycles of Figure 5 (b). It takes, for example, on average ca. 300 cycles until the standard network reaches the same error (0.25) as the network–fuzzy system on the test set. In real-time applications the improvement of the learning speed of the neural network can be essential and even imperative if neural networks are to be employed at all.

---

[2] In the following the description of data sets as *small* or *insufficient* compared to *large* or *sufficient* implies that the data sets do not contain enough information about the system in order for the model to generalize well after learning, i.e. in order to reach a similar error on the test set as on the training set.



(a) (b)

Figure 5. Regularized learning: (a) Training and test error for 11 data samples averaged over 20 runs; (b) Training and test error for 101 data samples averaged over 20 runs.

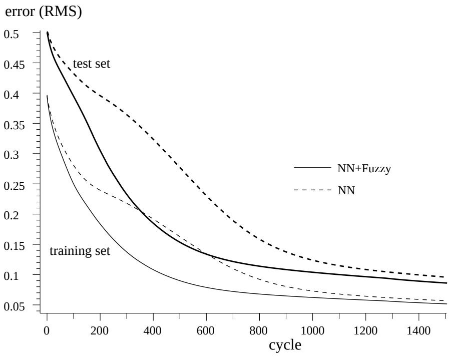
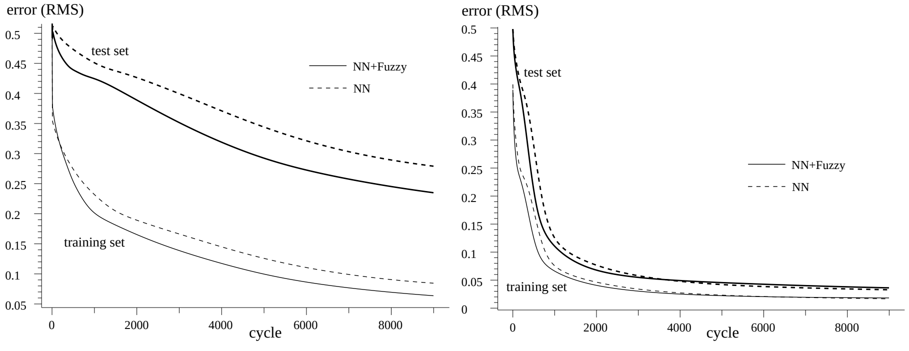


Figure 6. Regularized learning. The training and the test error for 101 samples during the first 1500 learning cycles (the first 1500 cycles of Figure 5).

5.2. CATALYZED LEARNING

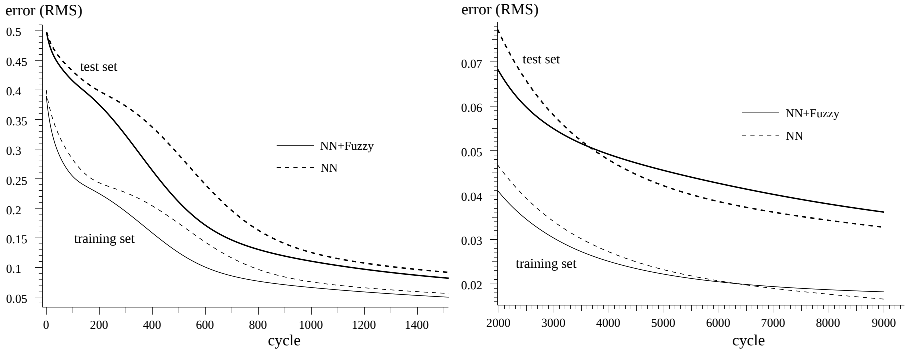
In this section, we will show that fuzzy knowledge can be used as a catalyst function in neural network learning. An output node and the corresponding weights are added to the neural network as shown in Figure 4. This node is removed after the network training. Similar to the regularized learning, two cases are considered. In the first case 11 examples and in the second 101 examples are generated randomly to train the *task neuron* of the neural network. The same number of examples (identical input) are generated by the fuzzy rule system in order to train the *catalyst neuron.* The training and test results are shown in Figure 7 (a) and (b) again averaged over 20 runs. The results are similar to those for the regularization technique. For the small data set, the network with knowledge incorporation performs better, both with respect to learning speed and the final error. Again the decrease



(a) (b)

Figure 7. Catalyzed learning: (a) Training and test error for 11 data samples averaged over 20 runs; (b) Training and test error for 101 data samples averaged over 20 runs.

of the error on the test set indicates improvement of the generalization ability of the network. In the case of the larger data set, Figure 7 (b), the findings are similar to those in the last section. Figure 8 (a) highlights the increase of the learning speed of the network for catalyzed learning. Closer observation of the curves at later cycles, as shown in Figure 8 (b), reveals that the error curves for both the training and the test data sets for the network with and without knowledge incorporation intersect after 6500 cycles and 3500 cycles, respectively. Since the effect



(a) (b)

Figure 8. Catalyzed learning. The training and the test error for 101 samples during the (a) first 1500 learning cycles (the first 1500 cycles of Figure 7) and (b) during learning cycles 2000–9000.

also occurs on the test data, it is unlikely that overtraining of the feed-forward neural network is responsible. An alternative explanation is that, due to its *fuzzy* nature, the additional knowledge itself is not error free. Therefore, after a certain number of cycles, the network performance depends on the accuracy of the additional knowledge. If this is indeed the case, then a possible extension would be to reduce the influence of the additional knowledge with increasing number of cycles. In this case, the additional knowledge would be exploited at the start of the training process as a means of speeding up the adaptation and it would be removed in later periods when the fine tuning of the network should depend solely on the assumed, more accurate data. Of course, for insufficient data sets (Figure 7 (a)) this effect cannot occur.

6. Conclusion

In this paper, two feasible approaches to knowledge incorporation into neural networks with the help of fuzzy rules have been suggested and applied to the task of modeling a two-link robot manipulator. The fuzzy rules can act as a bridge to transfer knowledge into examples which can be exploited by neural networks during learning. The first approach uses information from the derivative of the fuzzy system as a regularization term for the network learning. In addition, the fuzzy rule system, which is generated based on expert knowledge, was successfully used as a catalytic function in neural network learning. In both cases the learning speed of the network was increased considerably. As expected, the quantitative increase depends on how much information is given to the network in each cycle, i.e. how large and efficient the data set is which is used for learning. The final error of the network was decreased significantly for small data sets. The decrease on the test set, in particular, indicates that the ability of the network to generalize was enhanced by exploiting the additional knowledge. This kind of knowledge exploitation has also been reported using non-fuzzy based hints by Abu-Mostafa 1993a. In the case of the large data set and catalyzed learning a slight increase in the error of the network was observed, which might be due to the inherent error in the fuzzy system.

Knowledge incorporation by adding a regularization term performed better than catalyst learning both with respect to the quantitative decrease of the error and to the increase of the learning speed. At the same time the error increase at later learning cycles only occurred for the catalyst learning. This effect will be the subject of future research together with the application of the method to real world data sets in order to determine whether the improved performance of the regularization method holds in general.

Acknowledgements

The authors would like to thank W. von Seelen, M. Kreutz and C. Igel for stimulating discussions on the subject and for proofreading the manuscript. The project is supported by the German Ministry of research under grant LEONET 01IB802C4.

References

Abu-Mostafa, Y. S.: 1993a, 'Hints and the VC dimension'. *Neural Computation* **5**(2), 278–288.

Abu-Mostafa, Y. S.: 1993b, 'A method for learning from hints'. In: S. Hanson, J. Cowan, and C. Giles (eds.): *Advances in Neural Information Processing Systems*, Vol. 5. pp. 73–80.

Al-Mashouq, K. A. and I. S. Reed: 1991, 'Including hints in training neural nets'. *Neural Computation* **3**(3), 418–427.

Barber, D. and D. Saad: 1996, 'Does extra knowledge improve generalization?'. *Neural Computation* **8**(1), 202–214.

Bishop, C. M.: 1995, *Neural Networks for Pattern Recognition.* Oxford University Press.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. Warmuth: 1989, 'Learnability and the VC dimension'. *Journal of the ACM* **36**, 929–965.

Caruan, R.: 1995, 'Learning many related tasks at the same time with backpropagation'. In: G. Tesauro, D. Touretzky, and T. Leen (eds.): *Advances in Neural Information Processing Systems*, Vol. 7. pp. 657–664.

Sill, J. and Y. S. Abu-Mostafa: 1997, 'Monotonicity hints'. In: M. Mozar, M. Jordan, and T. Petsche (eds.): *Advances in Neural Information Processing Systems*, Vol. 9. pp. 634–640.

Suddarth, S. and A. Holden: 1991, 'Symbolic neural systems and the use of hints for developing complex systems'. *International Journal of Man-Machine Studies* **35**, 291–311.

Vapnik, V. N.: 1995, *The nature of statistical learning theory.* New York: Springer.

Zadeh, L.: 1973, 'Outline of a new approach to the analysis of complex systems and decision processes'. *IEEE Trans. on Systems, Man and Cybernetics* **3**, 28–44.