On the Nonlinearity of Pattern Classifiers

Aarnoud Hoekstra and Robert P.W. Duin

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology,
Lorentzweg 1, 2628CJ Delft, The Netherlands, E-mail: aarnoud@ph.tn.tudelft.nl

Abstract

This paper presents a novel approach to the analysis of the overtraining phenomenon in pattern classifiers. A nonlinearity measure $\mathcal{N}$ is introduced which relates the shape of the classification function to the generalization capability of a classifier. Experiments using the k-nearest neighbour rule, a neural classifier and the quadratic classifier show that the introduced measure $\mathcal{N}$ can be used to study the overtraining behaviour of a classifier. Moreover $\mathcal{N}$ shows to be a predictor for the local sensitivity of a classifier. Classifiers that have a small local sensitivity are shown to have a low nonlinearity, whereas an increased nonlinearity indicates an increase in local sensitivity.

1. Introduction

An important difference between neural network classifiers and most other classifiers is that they can implement almost any classification function, provided that sufficient hidden units are supplied [4, 5]. The final shape of this classification function in the feature space is also determined by the neuron transfer function and the weights of the neuron connections. This ability of a neural network has positive and negative effects: it is potentially a very general classifier as it can be adapted easily for various classification problems. On the other side, unless special cautions are built-in, it can adapt itself too in an undesired way to the data noise. This is the well known overtraining phenomenon [7], closely related to the peaking phenomenon in traditional pattern recognition [2, 6].

If a neural network is overtrained the classification function follows the data noise too strongly by which it looses generalization capabilities. Overtraining can be studied by inspecting an estimate of the generalization error during training. Until now, as far as the authors are aware of, there have been no studies published as to the relation of overtraining with the shape of the classification function. It is to be expected that an overtrained classifier shows a larger nonlinearity than an optimally trained one. It is the purpose of this paper to present a tool for studying the nonlinearity of classifiers. We will apply this to neural network feedforward classifiers and inspect the nonlinearity during training. In order to know how this nonlinearity measure has to be judged, results will also be compared with the nonlinearity of some traditional pattern recognition classifiers. As we are interested here in developing tools for understanding neural network classifiers and other ones, we will present results for artificial datasets only.

In the next section the nonlinearity measure is introduced, formally defined and an estimation procedure is presented. After that a number of experiments will be studied in section 3, for 2-dimensional problems, as this enables us to illustrate the nonlinearities with the classification functions themselves. Finally conclusions will be drawn in section 4.

2. A nonlinearity measure

2.1. Informal model

The nonlinearity of a classifier is of interest in relation with the data it has to classify. Nonlinearities that are not reflected into classification results are not of interest. This can be the case for the classifier behaviour in remote, empty areas of the data space or in a microscopic scale between the data points, but not influencing their classification. Any nonlinearity measure should thereby be dependent on the data set or on the data distribution that is classified.

Let us discuss the classifier S_1 in figure 1. What happens outside the data domain is not of importance. Moreover, the subtle behaviour of the classifier is also not important for the remote data points to the right and to the left. The classification of these points is not changed if the nonlinear classifier S_1 is replaced by the linear classifier S_2. For these points the definition of a linear functions applies: a linear combination of inputs (datapoints with the same classification) results in a linear combination of outputs (constant classification of all interpolated points). For points close to S_1 this does not always hold : all points between y_1 and y_2

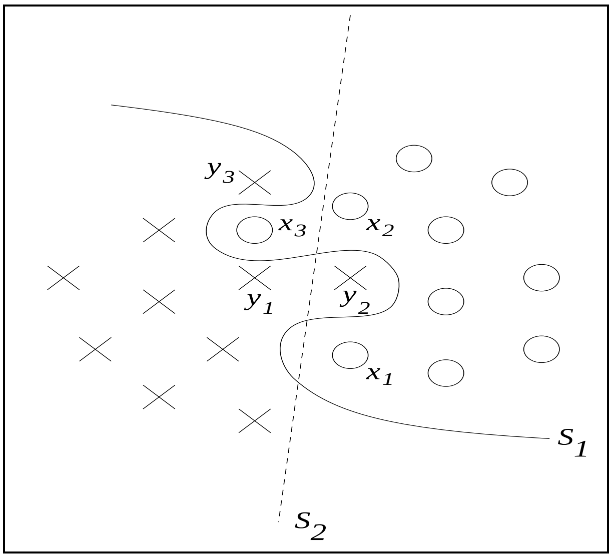

Figure 1. A highly nonlinear and a linear decision function on a set of objects

share the same classification, but fractions of the identically classified points between y_1 and y_3 and between x_1 and x_2 have different classifications. Note that the order of the decision is not of interest. Consider the following example. Assume all the labeled objects in figure 1 are not present, ie. no x_1,x_2 etc. If the B nonlinearities of both classifiers would be calculated they yield the same results, namely zero. The fact that S_1 is highly nonlinear is not detected since the objects do not observe the nonlinearities. Only if more objects are supplied the nonlinearities might appear.

Based on these observations we will investigate in this paper the usefulness of the following concept of classifier nonlinearity: The nonlinearity $\mathcal{N}$ of a classifier with respect to a dataset is the probability that an arbitrary point, uniformly and linearly interpolated between two arbitrary points in the dataset with the same classification shares this classification. This defines the nonlinearity with respect to a dataset, but is independent of the true labels of the datapoints. As a result, this definition is also independent of the performance of the classifier. In the remainder of this section we will present a more formal definition and the way the nonlinearity $\mathcal{N}$ is estimated.

2.2. Formal model

In this section the formal definition of $\mathcal{N}$ is introduced. First the nonlinearity of a classifier $\mathcal{S}$ with respect to two objects is defined which is then extended to a whole set of objects. Hence resulting in a definition of $\mathcal{N}$ of a classifier with respect to a set of objects. The nonlinearity of a classifier with respect to two objects is calculated according to the following definition.

Definition 1 *(Nonlinearity of $\mathcal{S}$ with respect to two objects)*
Let $\mathcal{L}$ be a set of objects x, ie. $\mathcal{L} = \{x_1, x_2, \ldots, x_N\}$, and let $\mathcal{W}$ be a set of labels w, ie. $\mathcal{W} = \{w_1, w_2, \ldots, w_M\}$. Given a classifier $\mathcal{S}(x)$ which assigns to each object $x_i \in \mathcal{L}$ a label $w_j \in \mathcal{W}$, define subsets $\mathcal{L}^i$ such that $\forall x \in \mathcal{L}^i : \mathcal{S}(x) = w_i \in \mathcal{W}$. The nonlinearity of $\mathcal{S}$ with respect to $\bar{x} \in \mathcal{L}^i \times \mathcal{L}^i$, with $\bar{x}$ being a pair (x_k, x_l) and $x_k, x_l \in \mathcal{L}^i$, is:

$$n(\bar{x}) = n((x_k, x_l)) = \frac{1}{||x_k - x_l||} \int_0^1 \mathcal{S}_\Delta(\alpha x_k + (1-\alpha)x_l, x_k) d\alpha, \quad (1)$$

where $\bar{x} \in \mathcal{L}^i \times \mathcal{L}^i$ and $\mathcal{S}_\Delta$ is defined as:

$$\mathcal{S}_\Delta(a, b) = \begin{cases} 0 & \text{if} \quad \mathcal{S}(a) \equiv \mathcal{S}(b) \\ 1 & \text{otherwise} \end{cases} . \quad (2)$$

□

The $\mathcal{S}_\Delta$ ensures that the way the classlabels are represented does not effect the definition of $n(\bar{x})$. By having $\mathcal{S}_\Delta$ classlabels can be represented by numbers or whatever one desires. Note that $\bigcup_{i=1}^M \mathcal{L}^i = \mathcal{L}$ and $\bigcap_{i=1}^M \mathcal{L}^i = \emptyset$, otherwise the classifier $\mathcal{S}$ would assign more than one label to an object at the same time. Having defined the nonlinearity for two objects it can be extended to the $\mathcal{N}$ of the whole set of objects $\mathcal{L}$. $\mathcal{N}(\mathcal{L}, \mathcal{S})$, the nonlinearity of classifier $\mathcal{S}$ with respect to the set $\mathcal{L}$, is defined as:

Definition 2 *(Nonlinearity of $\mathcal{S}$ with respect to a set of objects $\mathcal{L}$)*
Let $n(\bar{x})$ be the function stated in equation 1, let $\mathcal{L}$ be a set of objects and let $\mathcal{S}$ be a classifier which assigns a label to every $x \in \mathcal{L}$ then $\mathcal{N}(\mathcal{L}, \mathcal{S})$ is defined as:

$$\mathcal{N}(\mathcal{L}, \mathcal{S}) = \frac{1}{\sum_{i=1}^M |\mathcal{L}^i \times \mathcal{L}^i|} \sum_{i=1}^M \sum_{\bar{x} \in \mathcal{L}^i \times \mathcal{L}^i} n(\bar{x}). \quad (3)$$

□

The $\mathcal{N}$ of the set $\mathcal{L}$ is computed by calculating the individual $n(\bar{x})$ contributions of each pair in $\mathcal{L}$ having the same label. This sum is then normalized with respect to the number of pairs in each class.

2.3. The approximation algorithm

Since $n(\bar{x})$ in definition 1 is stated in terms of an integral calculation, for each $n(\bar{x})$ the integral over α has to be calculated. For M classes with in each class

N objects, the number of $n(\bar{x})$ calculations would be $M * N^2$. This is only the number of integrals to be computed. Since the computation of an integral also takes a number of steps, depending on the accuracy desired, the total number of computations becomes enormous. Moreover if $\mathcal{N}$ should be known at every time step when training a classifier it is useful to construct an algorithm that takes less computation time and still yields the correct results. In order to achieve this, an algorithm was constructed which uses a Monte-Carlo estimate to approximate $\mathcal{N}(\mathcal{L}, \mathcal{S})$.

First all the objects in the dataset are classified according to the classifier. This computation step can not be avoided in our implementation. Then n classified objects x_k are randomly drawn using sampling with replacement. Given this set of classified objects a corresponding x_l with the same classification is randomly generated. This ensures valid pairs $\bar{x} = (x_k, x_l)$. For each of the pairs (x_k, x_l) randomly an α between 0 and 1 is drawn. This α is used to create a new object y, created using $\alpha x_k + (1-\alpha)x_l$, which is a linear interpolation between x_k and x_l. Now the classifier $\mathcal{S}$ is used to classify this object y. If the classifier assigns the object to another class than that of x_k (or x_l), this contributes to the nonlinearity. The nonlinearity $\mathcal{N}$ is now found by dividing the number of nonlinearity contributions by n (the number of pairs drawn). Note that when n is large enough, the algorithm approximates $\mathcal{N}(\mathcal{L}, \mathcal{S})$ asymptotically. Below the NONLIN algorithm is given which which the nonlinearity $\mathcal{N}(\mathcal{L}, \mathcal{S})$ can be approximated.

The NONLIN algorithm is implemented as follows:

Step 1. Classify all objects according to the classifier $\mathcal{S}$.

Step 2. Draw randomly n objects x_k from the data set $\mathcal{L}$.

Step 3. Find for each x_k a corresponding x_l with the same classification label, ie. create an $\bar{x}$.

Step 4. Generate n random α's where $0 \leq \alpha \leq 1$ (for each pair an α is drawn).

Step 5. Compare classification $\mathcal{S}(\alpha x_k + (1-\alpha)x_l)$ with $\mathcal{S}(x_k)$, and count as nonlinearity if they are not classified in the same class. Do this for all pairs $\bar{x}$.

Step 6. $\mathcal{N}$ is now determined by the number of nonlinearity counts divided by n.

3. Experiments

To investigate the nonlinearity behaviour of pattern classifiers, several experiments were conducted using the nonlinearity measure $\mathcal{N}$. The experiments reported

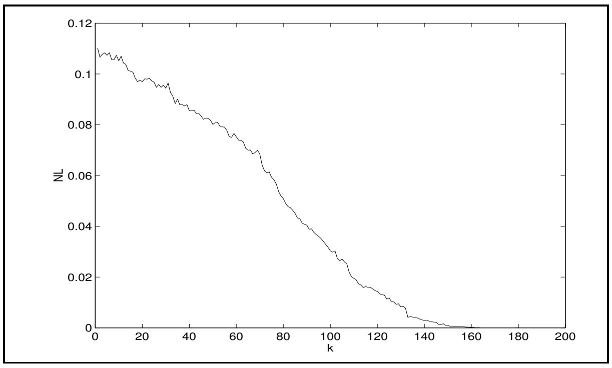

Figure 2. The nonlinearity of the k-nearest neighbour rule for different k on the dataset, NL denotes the nonlinearity

in this section are all two class 2-dimensional classification problems. These types of problems enable an easy illustration of the nonlinearity behaviour. The nonlinearity $\mathcal{N}$ is, however, not restricted to these kind of problems. All experiments were done using the Matlab neural network toolbox [1] and our, publically available, pattern recognition toolbox [3]. In the experiments the behaviour of three classifiers was studied, the *normal based quadratic classifier*, the *k-nearest neighbour classifier* and a *neural network classifier* trained with the Levenberg-Marquardt learning rule. Furthermore the classifiers were trained on different datasets in order to investigate their specific features such as overtraining. The quadratic classifier is included in the experiments since it yields a stable nonlinearity and can therefore serve as a reference.

3.1. Nearest neighbour classifier

Figure 3 visualises the dataset used to investigate the behaviour for the k-nearest neighbour rule, both classes are denoted by '*' and '+'. This dataset is perfectly separable, however it requires a highly nonlinear decision function to do it. Two experiments were done with the k-nearest neighbour rule. One involves the computation of the optimal k for this dataset using a leave-one-out estimation and the resulting nonlinearity. The other is the calculation of the nonlinearity for different k.

It turned out that the the optimal k is equal to 1 resulting in a classifier having a small local sensitivity. This could be expected due to the fact that the classes do not overlap but are closely to each other and the sample size used to train the classifier is large enough. Figure 3 depicts the 1-nearest neighbour decision function on the dataset. The classifier has zero apparent error, a small generalization error of 0.2% on a test set of 400 samples per class, and an $\mathcal{N}$ of 0.11 indicating that its nonlinearity is high.

The other experiment involved the determination of the nonlinearity of the k-nearest neighbour rule for different k. The result is depicted in figure 2, k was chosen in the range from 1 (optimal) to 200 (the number of samples/class) in steps of 1. For each k the nonlinearity was calculated. For sufficient accuracy the number of data pairs used in the algorithm was set to $100,000$. As can be seen the nonlinearity decreases for larger k, this means that the classifier starts as a highly nonlinear one and gradually becomes linear. It can be concluded that the larger the local sensitivity of the classifier the more linear the classifier. However the local sensitivity is determined by the dataset and therefore it also determines the nonlinearity of the classifier. Generally for a nearest neighbour classifier, the larger the local sensitivity the smaller the nonlinearity.

3.2. Neural network classifier

It is well known that neural classifiers suffer from small sample size problems. Training a classifier on too few samples will generally result in classifiers having a poor generalization capability. Such classifiers have adapted too much to the noise in the dataset and are therefore expected to have an increased nonlinearity. In figure 4 the dataset is visualised on which a neural network was trained, the classes are denoted by '*' and '+'. The dataset consists of two overlapping classes that require at least a quadratic decision function to be separated. In order to study the nonlinearity behaviour of a neural network properly, a small training set of 40 samples per class was used. To obtain overtraining in the neural network, the Levenberg-Marquardt learning rule was chosen. This learning rule combined with a sufficient number of hidden units and a small number of examples leads relatively fast to an overtrained network. The overtraining is desirable since the nonlinearity of the classifier during the whole learning process is of interest and not only the optimal case. However the number of hidden units is difficult to determine beforehand. Therefore several neural networks with different sizes of the hidden layer were trained and investigated. The number of hidden units was chosen to be 2, 8 and 20. For each of these neural networks their nonlinearity was computed on $10,000$ data pairs, also each network was trained for a fixed number of epochs which was set to 150.

Figure 5 depicts the nonlinearities of the different networks. Each network is denoted by $2-X-1$ indicating that it has 2 inputs, X hidden units and 1 output. The nonlinearity for each network starts small and then almost monotonically increases to a certain level where it remains more or less stable. However not in all situations this occurs, for instance for the $2-8-1$ network, the nonlinearity still increases and does not

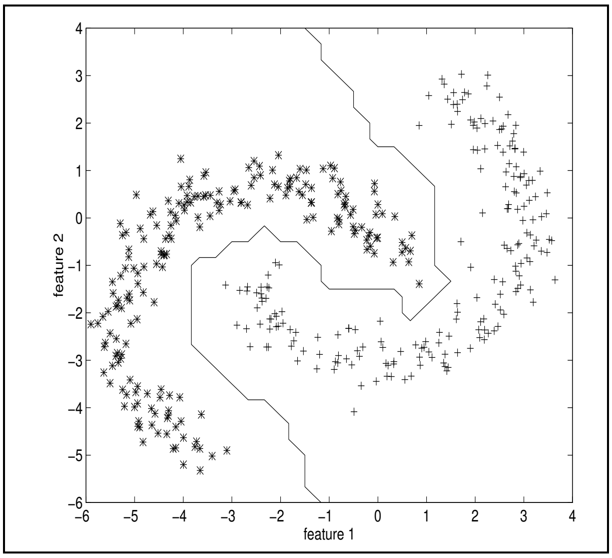

Figure 3. The decision function implemented by the optimal $k = 1$ nearest neighbour rule.

reach a stable point. When these curves are compared with the error curves (see right column of figure 5) for the networks, it turns out that the nonlinearity increases with increasing generalization error. Moreover if the test set error remains almost constant the nonlinearity also remains stable. Hence the nonlinearity is an indicator for the network's local sensitivity. A small local sensitivity results in a small training error, in general zero, and a large nonlinearity. Figure 4 depicts the $2-20-1$ network's decision function and the optimal decision function using a quadratic classifier. This network is clearly in an severly overtrained situation which implies a high nonlinearity. Compared to the "optimal" nonlinearity it is about 6 times higher. The network with 20 hidden units has far too much flexibility. For instance the $2-2-1$ neural network has just enough flexibility since its nonlinearity is almost equal to that of the quadratic classifier.

Concluding, training a neural network classifier results in a classifier that starts as a linear one and gradually increases to a highly nonlinear one. Due to a decreasing local sensitivity in the neural network, ie. it becomes overtrained, the nonlinearity increases. Experiments show that if the generalization error stabilizes the nonlinearity also reaches a stable point. However this behaviour strongly depends on initial conditions, such as sample sizes, learning algorithms and network sizes. The larger a neural network the more flexibility it has and the higher its nonlinearity will be if no precautions are taken.

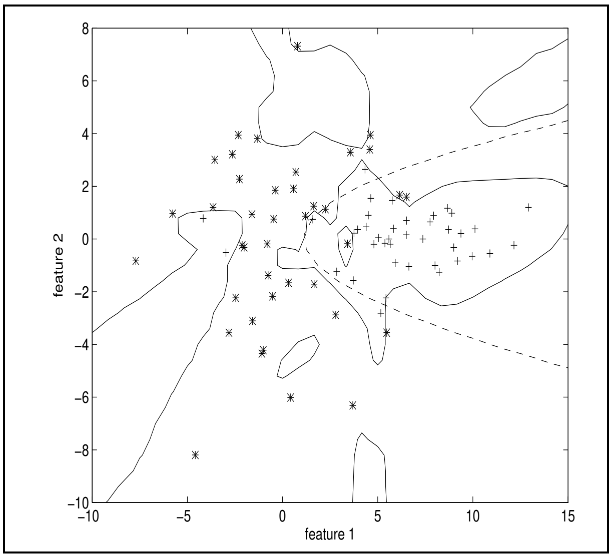

Figure 4. The decision function implemented by the $2-20-1$ neural network, solid line, and the optimal decision function implemented by the quadratic classifier, dashed line.

4. Conclusions

Here we will discusse the possible use of the introduced nonlinearity measure for neural network research and classification applications.

We have shown experimentally that during the training of a neural network classifier the resulting classification function has an almost monotonically increasing nonlinearity. It starts as a linear classifier and then adapt itself to the nonlinear dataset properties and finally to the dataset noise. The training procedure, of course, influences this behaviour strongly. If no precautions, however, are made this is the general behaviour.

The nonlinearity appears to be a good measure for comparing traditional pattern classification techniques with neural networks. Parametric classifiers yield a stable nonlinearity that can be used as a reference. Nonparametric techniques like the k-nearest neighbour rule and Parzen based classifiers have a nonlinearity that depends on a neighbourhood parameter that determines the local sensitivity of the rule. The dependency of the nonlinearity on the local sensitivity might be compared with the training effort put into a neural network classifier. As the local sensitivity has an optimal value, this indicates that there also might be an optimum training effort for a neural network classifier. This depends, of course, on the training rule and especially on the network criterion. For the traditional mean square error criterion, that was used in our experiments, this shows that a search for the global optimum is not desirable. This global optimum corresponds with a too large nonlinearity adapted to the noise and thereby with a too small local sensitivity. Once more this illustrates the strength and weakness of a neural network: it is very flexible and yet this flexibility should be restricted firmly in order to stay away from the noise.

All the above depends on the size of the training set. If this size is too small there is no transition between generalizable nonlinearity and noise adaptation in the training of a neural network classifier. If the training set size is large enough then this transition shows a flat region in the grow of the nonlinearity, corresponding with a local sensitivity that has still not yet reached the single datapoints. As a result the classification function is not yet determined by the noise in the dataset.

In conclusion it can be stated that we have developed a tool for studying the nonlinearity of classification functions in general that is particularly of importance for understanding the possibilities of neural network classifiers.

5. Acknowledgements

This work was partially supported by the Foundation of Computer Science in the Netherlands (SION) and the Dutch Organization for Scientific Research (NWO).

References

[1] Demuth, H. and Beale, M. *Neural Network toolbox for use with Matlab.* The Mathworks Inc., 1993.

[2] Duin, R.P.W. Superlearning capabilities of neural networks ? In *Proceedings of the 8^{th} Scandinavian Conference on Image Analysis*, pages 547–554. Norwegian Society for Image Processing and Pattern Recognition, May 25-28 1993.

[3] Duin, R.P.W. *PRTOOLS a Matlab toolbox for pattern recognition.* Pattern Recognition group, Delft University of Technology, October 1995.

[4] Funahashi, K-I. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[5] Hornik, K. Multilayer feedforward networks are universal approximators. *Neural networks*, 2:359–366, 1989.

[6] Kanal, L.N. and Krishnaiah, P.R. *Handbook of statistics 2: Classification pattern recognition and reduction of dimensionality.* North-Holland, 1982.

[7] Wolpert, D.H., editor. *The mathematics of generalization.* proceedings of the SFI/CNLS workshop on formal approaches to supervised learning, Novemer 1994.

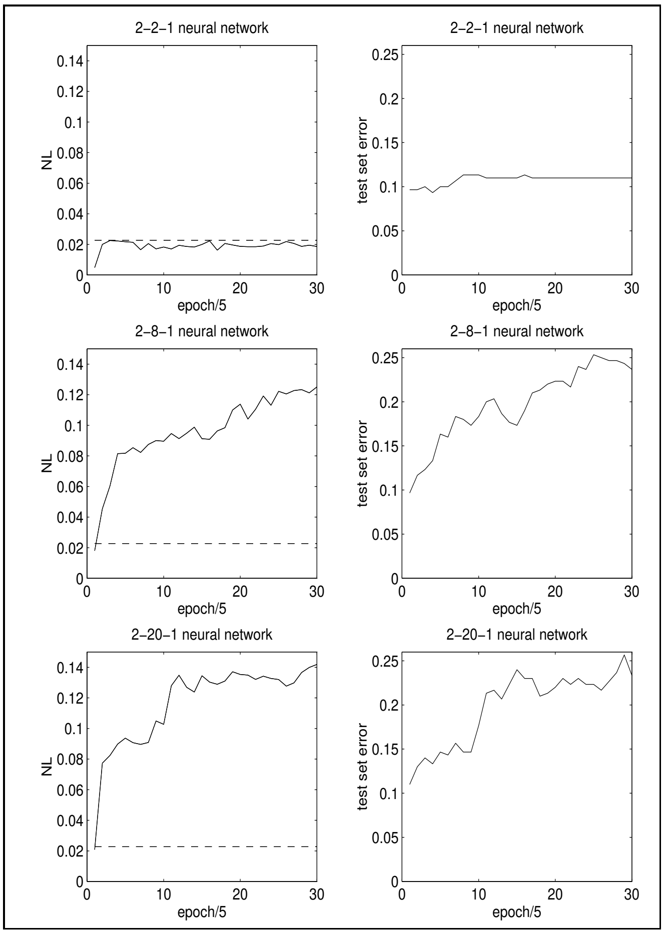

Figure 5. The nonlinearities for different neural networks, left column, trained using the Levenberg-Marquardt training rule. The right column depicts the generalization error of the trained neural networks.