LINEAR AND QUADRATIC LOCAL MODELS FOR ICE-NETWORKS

Mark Schäfer and Werner Dilger

Chemnitz University of Technology
09107 Chemnitz, Germany
{mark.schaefer, werner.dilger}@informatik.tu-chemnitz.de

ABSTRACT

ICE-Networks are hybrid Neural Networks that are capable of fast initial learning and continuous learning, They have been developed for predicting the development of technical processes. ICE-Networks have a dynamic structure, they are built up starting from empty networks during the training process. This construction process is continued as long as the network is in use thus the network can yield an actual prognosis at any time.

An ICE-Network is a layered network consisting of four layers. The units of the first hidden layer are RBF-neurons, called *prototypes*, and combine subsets of input vectors into so called *local models* that are maintained in the units of the second hidden layer. The type of the local models can be predefined by the developer of the ICE-Network, they can be linear or of higher order. In this paper the preciseness of the prognosis made by linear and quadratic models and the efficiency of computing those models are compared.

1. INTRODUCTION

ICE-Networks were developed by M. Tagscherer in his PhD thesis [6]. The purpose of ICE-Networks was to predict the next few steps that will happen in a technical process, represented by the values of certain parameters. Therefore an ICE-Network should be used on-line as a component of a control system and for the same reason it should be capable of continuous learning in order to adapt itself to the ongoing process.

However, ICE-Networks can also be used off-line to find a function for a given set of time dependent parameter values that describe some unknown process. They adapt to this function on the basis of local models which they produce incrementally from the input values. The local models are constructed and continuously restructured during the training process until the whole set of local models satisfies a certain quality criterion.

In [6] (cf. also [3]) linear models are used, but the question is raised whether higher types of models would yield better results, i.e. adapt better to the underlying function. The question is, how much better the results would be and how this pays against the loss of efficiency. In this paper we try to give an answer to the question. We compared linear with quadratic models but did not deal with higher order models because they would be too inefficient. The result is that quadratic models are considerably better than linear ones but need eight times more computing time than those. This may be prohibitive for on-line use of the ICE-Networks at present, however, with ever faster processors it may soon come into reach.

In the rest of the paper we describe the structure, functioning and the training process of ICE-Networks (section 2). We then define the basic measure for the quality of local models that we use throughout the paper (section 3). In the following two sections the computation of the quality measure for the two types of models is described. Finally in section 6 we sum up the results and discuss some open problems.

2. ICE-NETWORKS

An ICE-Network is a layered Neural Network with four layers. The units of the input layer are fully connected to those of the first hidden layer. They deliver the whole input vector to each unit of that layer. The units of the first hidden layer are called *prototypes*. They are RBF-neurons with Gaussian type activation function which has the advantage over mostly used sigmoid type activation functions that the adaptation of the units has only local side effects (cf. [5]). The purpose of the prototypes is to combine the input vectors of a certain receptive field into a local model. For this end a group of prototypes is connected to exactly one unit in the second hidden layer. The units of this layer are called *local models*. The connections between the first and second hidden layer are bidirectional so that each group of prototypes connected with a certain local model can influence this unit and vice versa. The purpose of a local model is to maintain a model of the function to be adapted according to the input values of the receptive field that is represented by its attached prototypes. Finally the output layer consists of only one unit that combines the local models into one global model. Figure 1 shows the structure of the ICE-Network

ICE-Networks can be considered as hybrid networks. On the one hand they have the feature of self organizing maps ([2]) with the dynamic construction of the network during the training process. Similar approaches are the Growing Cell Structures [1] and the Neural Gases [4].

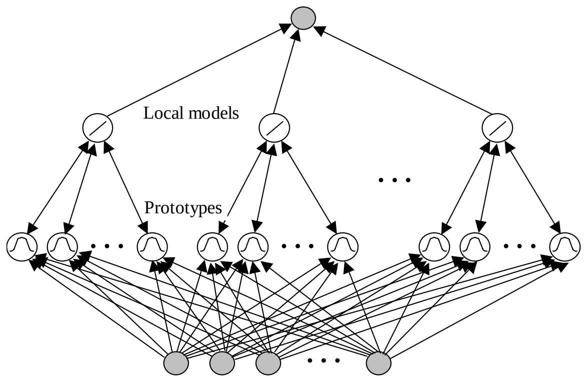

Figure 1: Structure of an ICE-Network

The learning process is supervised. At the beginning of the training process the two hidden layers are empty. With the first input vector a prototype and a local model are generated. The position of the prototype is equal to the input vector, the receptive field covers the whole domain of input vectors, and the local model is a constant whose value is equal to the expected target value for the respective input. When additional vectors are fed into the network it has to be decided if a new prototype must be generated and added to one of the groups of prototypes attached to a local model or if only one of the existing prototypes has to be adapted to the new value. This decision depends on the difference between the modeled and the target function. If the difference is less than some parameter α, only the prototype with the highest activation is moved a bit in the direction of the input vector. If it is greater than α, a new prototype is generated and added to one of the groups. It depends on the position of the vector in the input domain and on the receptive fields that have been discriminated so far to which local model the new prototype is added. In a beginning phase all new prototypes are put together in one local model. This local model is then no longer a constant value but some function of the target values that have been seen so far. The type of that function has been chosen before, in [6] it is a linear one.

In each step of the training process the receptive field of the prototypes is adjusted. This is controlled by a parameter λ. If two neighboring prototypes p_1 and p_2 have overlapping receptive fields then the activation of p_2 at the position of p_1 must not exceed λ. If it does, the receptive field of p_2 is reduced. This happens in most cases when a new prototype is introduced. Figure 2 illustrates the situation.

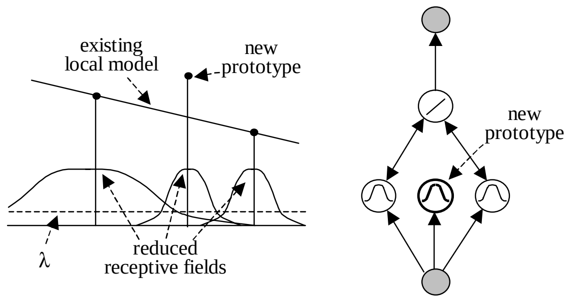

Figure 2: Reduction of receptive fields

When a new prototype is introduced it is checked whether the activation values of the already existing neighboring prototypes at the position of the new one is greater than λ. If not, the local model of these prototypes is split. This situation is illustrated in figure 2. New separated local models are generated and all three models are constant values equal to the respective target values, cf. figure 3.

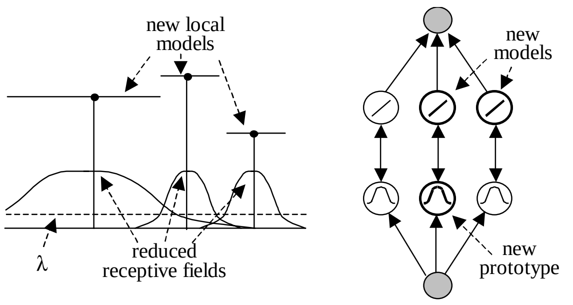

Figure 3: Generation of new local models

When additional prototypes are generated in the next steps of the training process, the local models again become linear, usually non-constant functions and adapt better to the target function. A number of different steps can be made that improve the adaptation, cf. [6].

Tagscherer also made some experiments with higher order local models. He showed that better results can be obtained with such models but he did not make an analytical investigation of this point.

3. QUALITY MEASURES FOR LOCAL MODELS

In order to analyse the quality of different local models we need a measure for the difference of two functions, namely the function represented by the local models and the target function. For this purpose the *mean square difference* (MSD) of two functions *f* and *g* is defined as follows:

Let $f, g: \Re^n \to \Re$ be two functions. The *mean square difference* of f and g in the lower and upper bounds $(a_1, a_2, \ldots, a_n)$ and $(e_1, e_2, \ldots, e_n)$ ($e_i > a_i$ for all i) $MSD_{f,g}$ is defined as

$$MSD_{f,g} := \prod_{i=1}^{n}(e_i - a_i)^{-1}\int_{a_1}^{e_1}\int_{a_2}^{e_2}\dots\int_{a_n}^{e_n}(g-f)^2(x_1,x_2,\dots,x_n)dx_n\dots dx_2 dx_1$$
$$MSD_f := MSD_{f,0} \tag{1}$$

As was described in section 2, in the training process of an ICE-Network local models are split whenever the difference between the network prognosis and the target function $|y - \hat{y}|$ becomes greater than α. Thus after a finite number of steps for time invariant functions it holds $\forall \mathbf{x}|y(\mathbf{x}) - \hat{y}(\mathbf{x})| \le \alpha$. Therefore in this case we can give a simple estimation for the mean square difference:

$$\begin{aligned} MSD_{f,g} &\le \prod_{i=1}^{n}(e_i - a_i)^{-1}\int_{a_1}^{e_1}\int_{a_2}^{e_2}\dots\int_{a_n}^{e_n}(g-f)^2(x_1,x_2,\dots,x_n)dx_n\dots dx_2 dx_1 \\ &= \prod_{i=1}^{n}(e_i - a_i)^{-1}\prod_{i=1}^{n}(e_i - a_i)\cdot\alpha^2 \\ &= \alpha^2 \end{aligned} \tag{2}$$

For the investigation of local models it is sufficient to consider the difference function $f_d := y - \hat{y}$, or, more generally, the type of the difference function and its free parameters. The value of the difference function is always less than α and it can be shown that the MSD does not depend on the integration bounds. The question is now how the free parameters must be chosen such that the value of the difference function stays within the interval $[-\alpha;\alpha]$ throughout the whole domain, which is defined as $[-1;1]^n$. The set of tuples that satisfy this condition forms the parameter space Φ.

In the following discussion of the quality of linear and quadratic local models we do not use the mean square difference directly because it compares only two given functions. Rather we are interested in types of functions of which we do not know the exact parameter values in advance, therefore we can only compute the expectation value of the mean square difference $E(MSD)$ over all possible realizations of the function type. To each function type corresponds a point in the parameter space Φ and all points are assumed to be equally likely.

4. LINEAR MODELS

A linear model is a linear function $\hat{y} = \Re^n \to \Re$ with $\hat{y}(\mathbf{x}) = \mathbf{m}\cdot\mathbf{x'}$. For the local models we set $\hat{y}(\varsigma_i) = \rho_i$ for all i. Thus we have the following equation system

$$\begin{pmatrix} \varsigma_1^1 & \varsigma_2^1 & \cdots & \varsigma_n^1 & 1 \\ \varsigma_1^2 & \varsigma_2^2 & \cdots & \varsigma_n^2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \varsigma_1^p & \varsigma_2^p & \cdots & \varsigma_n^p & 1 \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n+1} \end{pmatrix} = \begin{pmatrix} \rho^1 \\ \rho^2 \\ \vdots \\ \rho^p \end{pmatrix} \tag{3}$$

For a satisfactory computation of a linear model $p = n + 1$ prototypes are sufficient. The equation system (3) has to be solved with $p = n + 1$ each time when any prototypes of the local model are to be modified.

For one-dimensional target functions the difference function is $f_d(x_1) = a_1x_1 + a_0$. The function is of degree 2 and dimension 1, thus the parameter space is $\Phi_2^1 \subseteq \Re \times \Re$ and it holds

$$\forall(a_1, a_0) \in \Phi_2^1 \forall x_1 \in [-1;1] - \alpha \le a_1 \cdot x_1 + a_0 \le \alpha$$

The parameter space is illustrated in figure 4.

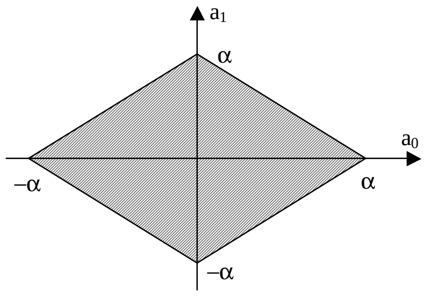

Figure 4: The parameter space Φ_2^1

Since with higher order difference functions it becomes more and more difficult to determine the parameter space, we generally transform the difference function in polar form and view it as a parameterized function depending on x_1 in the space (a_0, a_1). The borders of the parameter space have the equation $a_1x_1 + a_0 = \alpha$. For the representation with polar coordinates we need for each angle δ_0 the distance $l(\delta_0) > 0$ of the point of intersection of a line through the origin with angle δ_0 to the a_0-axis with the border of the parameter space. Between a_0, $a1$, and $l(\delta_0)$ the following relation holds:

$$\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = l(\delta_0)\cdot\begin{pmatrix} \cos(\delta_0) \\ \sin(\delta_0) \end{pmatrix}$$

From this we can compute the distance function as

$$l(\delta_0) = \frac{\alpha}{\sin(\delta_0)x_1 + \cos(\delta_0)}$$

Now the borders of the parameter space can be easily determined by computing the minimum of x_1. In order to do this the derivation of the distance function is used:

$$l'(\delta_0) = -\frac{\alpha\cdot\sin(\delta_0)}{(\sin(\delta_0)x_1 + \cos(\delta_0))^2}$$

This function has nowhere the value zero, therefore only the borders $x_1 = -1$ and $x_1 = 1$ have to be considered. Thus we get the region shown in figure 4. The probability density in this space is constant, namely the reciprocal of the area $\frac{1}{2\alpha^2}$. Therefore the expectation value $E(MSD_{f_d})$ is

$$E(MSD_{f_d}) = \int_{-\alpha}^{\alpha} \int_{-(\alpha-a_0)}^{\alpha-a_0} \frac{1}{2\alpha^2} \cdot \frac{1}{2} \int_{-1}^{1} (a_1x_1 + a_0)^2 \, dx_1 da_1 da_0$$
$$= \frac{4}{9}\alpha^2$$

For two-dimensional target functions the difference function is

$$f_d(x_1, x_2) = a_2x_2 + a_1x_1 + a_0 .$$

The parameterized equation of the border is

$$a_2x_2 + a_1x_1 + a_0 = \alpha \qquad (4)$$

Therefore for $l(\delta_0,\delta_1)$ we have

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = l \cdot \begin{pmatrix} \cos(\delta_0)\sin(\delta_1) \\ \sin(\delta_0)\cos(\delta_1) \\ \cos(\delta_1) \end{pmatrix} \qquad (5)$$

From (4) and (5) we can compute

$$l = \frac{\alpha}{\cos(\delta_1)x_2 + \sin(\delta_0)\cos(\delta_1)x_1 + \cos(\delta_0)\sin(\delta_1)} \qquad (6)$$

From (6) the partial derivation with respect to x_1 and x_2 can be determined:

$$\frac{\partial l}{\partial x_1} = -\frac{\alpha \sin(\delta_0)\cos(\delta_1)}{\cos(\delta_1)x_2 + \sin(\delta_0)\cos(\delta_1)x_1 + \cos(\delta_0)\sin(\delta_1)}$$

$$\frac{\partial l}{\partial x_2} = -\frac{\alpha \cos(\delta_1)}{\cos(\delta_1)x_2 + \sin(\delta_0)\cos(\delta_1)x_1 + \cos(\delta_0)\sin(\delta_1)}$$

Again, as in the case of one-dimensional functions, both functions do nowhere become zero, thus there exist no local minima and it is sufficient to consider the border of the region. In this case it is even sufficient to consider the corners because on the other parts of the border the partial derivations disappear. Figure 5 shows the parameter space. Its volume is $\frac{4}{3}\alpha^3$, hence the probability density $\frac{3}{4\alpha^3}$. The expectation value of the mean square difference is

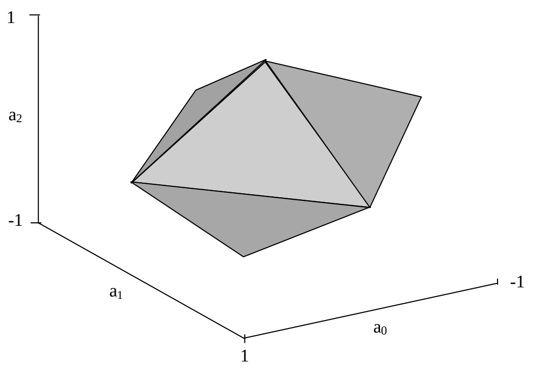

Figure 5: The parameter space Φ_3^2

$$E(MSD_{f_d}) = \int_{-\alpha}^{\alpha} \int_{-(\alpha-a_0)}^{\alpha-a_0} \int_{-(\alpha-a_0-a_1)}^{\alpha-a_0-a_1} \frac{3}{4\alpha^3} \cdot \frac{1}{4}$$
$$\int_{-1}^{1}\int_{-1}^{1} (a_2x_2 + a_1x_1 + a_0)^2 \, dx_2 dx_1 da_2 da_1 da_0$$
$$= \frac{48}{5}\alpha^2$$

We omit the derivation of the expectation value for three-dimensional functions. The result is

$$\frac{133}{45}\alpha^2$$

5. QUADRATIC MODELS

A quadratic target function $\hat{y}$ has the form

$$\hat{y}(\mathbf{x}) = m_{12}x_1^2 + m_{22}x_2^2 + \ldots + m_{n2}x_n^2$$
$$+ m_{11}x_1 + m_{21}x_2 + \ldots + m_{n1}x_n + m$$
$$= \begin{pmatrix} m_{12} \\ \vdots \\ m_{n2} \\ m_{11} \\ \vdots \\ m_{n1} \\ m \end{pmatrix} \cdot \begin{pmatrix} x_1^2 \\ \vdots \\ x_n^2 \\ x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix}$$

This results in an equation system that corresponds to (3), however with a $(2n + 1) \times (2n + 1)$ – matrix. A one-dimensional difference function now has the form

$$f_d := a_1x_1 + a_2x_1^2 + a_0$$

In order to determine the parameter space the function is transformed in polar form. The parameterized equation for the border is

$$a_2 x_1^2 + a_1 x_1 + a_0 = \alpha$$

and we have

$$l(\delta_0, \delta_1) = \frac{\alpha}{\cos(\delta_1)x_1^2 + \sin(\delta_0)\cos(\delta_1)x_1 + \cos(\delta_0)\sin(\delta_1)}$$

The derivation with respect to x_1 is

$$\frac{dl}{dx_1} = -\frac{\alpha(2\cos(\delta_0)x_1 + \sin(\delta_0)\cos(\delta_1)}{\cos(\delta_1)x_1^2 + \sin(\delta_0)\cos(\delta_1)x_1 + \cos(\delta_0)\sin(\delta_1)}$$

This function has a minimum at $x_1 = -\frac{1}{2}\sin(\delta_0)\cos(\delta_1)$. The volume of the parameter space can be determined only numerically. The value is approximately $3.73313\alpha^3$ and the expectation value is approximately $0.2429\alpha^2$.

For two-dimensional and three-dimensional target functions we get by a similar computation for the expectation value the values $E(MSD) = 0.32691\alpha^2$ and $E(MSD) = 0.6603\alpha^2$ respectively.

6. RESULTS AND OPEN PROBLEMS

In the following table the results for the value $E(MSD)$ in the different cases are summarized.

E(MSD)	one-dimen-sional	two-dimen-sional	three-dimen-sional
linear	$0.4444\alpha^2$	$9.6\alpha^2$	$2.9556\alpha^2$
quadratic	$0.2492\alpha^2$	$0.3269\alpha^2$	$0.6603\alpha^2$

Obviously quadratic models are considerably better than linear ones. On the other hand, they require eight times the computation time of linear models. This is an important criterion for the investigation of time variant functions but not for that of time invariant functions which can be done off-line. However, it is not known how often the computation has to be performed on the average so this value does not tell us too much, it may be the case that in practice the reqiured time is less than eight times.

The approach we have made is well suited to compute the approximation quality of linear and quadratic models. However, if we try to compute this property for higher order models or even completely other types of models we get unbound parameter spaces for which the expectation value for the mean square difference can not be computed. For this kind of models a more general approach is required.

In our approach we have assumed that all instances of a function type are equally likely. This need not be the case. The probability values result from the training process, therefore an empirical investigation or a more precise analysis could lead to a refined assumption about the values.

7. REFERENCES

[1] Fritzke, B., Growing cell structures – s self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9): 1441 – 1460, 1994.

[2] Kohonen, T., *Self-Organization and Associative Memory* ^{3}, Springer Series in Information Sciences, Springer, Heidelberg, 1989.

[3] Lewandowski, A., Tagscherer, M., Kindermann, L., Protzel, P., Improving the fit of locally weighted regression models. *Proc. of the Sixth Int. Conf. on Neural Information Processing* (ICONIP ’99), 371 – 374, Perth, 1999.

[4] Martinetz, T, Schulten, K., A neural-gas network learns topologies. In Kohonen et al. (eds.): *Proc. Int. Conf. on Artificial Neural Networks*, Espo, vol. 1, 397 – 402, North-Holland, Amsterdam, 1991.

[5] Poggio, T., Girosi, F., Networks for Approximation and Learning. *Proc. of the IEEE*, 78, 1481 – 1497, 1990.

[6] Tagscherer, M., *Dynamische Neuronale Netzarchitektur für Kontinuierliches Lernen*. PhD thesis, Chemnitz University of Technology (http://archiv.tu-chemnitz.de/pub/2001/0072/data/tagscherer.pdf), 2000.