

Overcome Neural Limitations for Real World Applications by providing Confidence Values for Network Prediction

Michael Tagscherer⁺, Lars Kindermann⁺,
Achim Lewandowski⁺ and Peter Protzel⁺⁺

⁺FORWISS - Bavarian Research Center for
Knowledge-Based Systems
Am Weichselgarten 7, 91058 Erlangen, Germany;
{tagscherer;kindermann;lewandowski}@forwiss.de

<http://www.forwiss.de/aknn/>

⁺⁺Chemnitz University of Technology,
Dept. of Electrical Engineering and Information
Technology,
Institute of Automation, 09107 Chemnitz, Germany
peter.protzel@e-technik.tu-chemnitz.de

Abstract

In this paper we present an incremental construction algorithm for continuous learning tasks and one of its special features - simultaneous learning of the target function and a confidence value for the system predictions. The basis of the hybrid system is a radial basis function (RBF) network layer. The second layer consists of local models. The two layers are closely combined with a strong interaction. The number of RBF-neurons and the number of local models have not to be determined in advance. This is one of the main advantages of the algorithm. Another advantage emphasized in this paper is the ability to learn the training data distribution simultaneously to the learning of the target function. The learned data set distribution can be used as a confidence value for a given network prediction. The development of the described approach is embedded in a larger project that is primarily concerned with system identification tasks for industrial control such as steel processing

1. Introduction

The approximation of nonlinear and time-variant functions are the main requirements for system identification tasks for industrial control such as steel processing [1]. While nonlinear function approximation is a well-known application for neural networks, the approximation of nonlinear functions that change over time poses many additional problems which are the focus of our current research.

1.1 Uncertain Network Predictions

One problem with system identification tasks especially with neural networks is, that most systems produce an output to a given input pattern regardless whether they have seen data from the current input space area or not. If the system has seen data from this input space area its prediction should be sufficient. However, if the system has not seen any training patterns from this input space sector or the seen patterns are quite different the generalization, of the neural net and thus its prediction might not be correct. However, for an operator of an industrial process the confidence of the learning system output values is very important. In case of less confident outputs, the operator has to be more careful. Thus the identification system should additionally produce a confidence value to its predictions.

1.2 Industrial Processes and Continuous Learning

Due to the time variance of most industrial processes, system identification, e.g. with neural networks has to be done continuously throughout the life-time of the process. We refer to this as *continuous learning*¹. One of the main problems of continuous learning tasks is the stability-plasticity dilemma. While the system should be able to follow changes of the time-varying function as quickly and accurately as possible (i.e. optimal plasticity), previously acquired information should not be "forgotten" (i.e. stability) [2]. In this case, the popular multi-layer perceptrons (MLPs) are not suited. Due to their global activation functions, global information can be forgotten while presenting and learning local information. Like mentioned above, MLPs always produce an output to a given input pattern whether they have seen data in this input space sector or not. No confidence value to their current output can be expected.

Another approach is to use localized units, e.g. radial basis function (RBF) networks [3]. Not only because they are much faster trainable, RBF-networks are in many cases a better choice than MLPs. Nevertheless, some difficulties have to be resolved when using RBF-networks, e.g. the optimal topology, the right placement of each neuron and the extension of each receptive field. RBF-networks which approximate the target function with the localized units have similar problems like MLPs to produce a confidence value.

1. A detailed description and more information can be found at our NIPS '98 *Continuous Learning* Workshop page: <http://www.forwiss.uni-erlangen.de/aknn/cont-learn/>

Instead of using localized units, another approach is to use a certain number of localized models. Each model is valid for a different region of the input-space. The main problem of these approaches is to find a good representation of the target function by different models, i.e. it is necessary to find a useful clustering of the input-space. Thus, the quality of the approach depends on the quality of the respective clustering method. Counterpropagation networks which use Kohonen networks for clustering are an example for these kind of approaches [4][5]. One disadvantage of those systems is the weak combination of the two different layers. First, the clustering must be nearly completed before useful models at the second layer can be adapted. Approaches like counterpropagation networks produce no confidence value to their current network output. Only the target function is approximated, not the data distribution.

The discussed examples represent only a short and nonexhaustive overview of different approaches and their characteristics with respect to continuous learning and the calculation of confidence values for network predictions.

In the following section we will describe the Incremental Clustering and Evaluation (ICE) algorithm. The ICE-algorithm combines advantages of RBF networks and localized models to a hybrid system.

2. ICE-Hybrid System and Training Algorithm

ICE is an incremental construction algorithm of a hybrid system for continuous learning tasks [6]. The foundation of the hybrid system is a radial basis function (RBF) network layer. The two tasks of this layer are to approximate the data distribution and the clustering of the input space in combination with the second layer. The second layer consists of local models which are used to approximate the target function. Noteworthy is that the number of RBF-neurons, the number of models and the size of their receptive fields are determined incrementally during the presentation of the training patterns. Another advantage of ICE is that the target function and the training data distribution can be learned simultaneously.

2.3 New Aspects

In contrast to methods like counterpropagation-networks, the ICE-model layer and the RBF cluster layer are closely combined. For example, the RBF-layer uses information from the model-layer

to decide if new RBF-neurons are needed and the model layer uses the RBF-neurons as the base for its models. This strong interaction produces a useful network output, that is already available during the initial learning phase. This is very important for industrial applications, e.g. plants can be put into operation much faster.

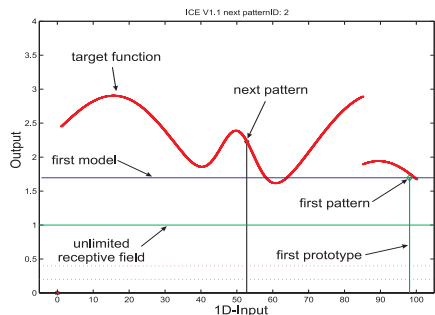
The combination of a strong interaction between the different ICE-layers and the usage of more than one prototype (specialized RBF-Neurons, cf. 2.4 Adaptive Hybrid Network) per model can result in a better representation of the past training patterns within each model. This makes a clustering with less models possible. For an operator a clustering with less models is easier to survey and to handle. With regard to industrial control tasks, such as steel processing, a better acceptance of such systems can be expected. Another positive effect is that no storage of the past training patterns is needed, because the prototypes are a representation of the seen patterns.

2.4 Adaptive Hybrid Network

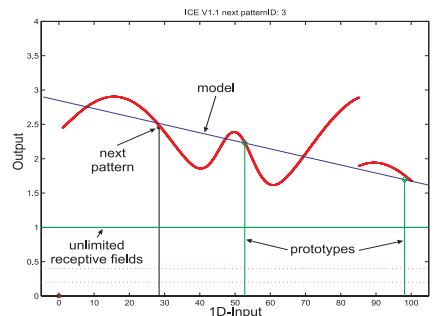
The main idea of the ICE-algorithm is that the individual models *compete* with each other. In this way they reduce their receptive fields in a mutual interaction. A new model will be created if the quality of the active model is not sufficient. The base of each model are n specialized RBF-neurons. In the following we call this neurons *prototypes*. Specialized means that they have additional properties, e.g. an offset parameter at the center. This parameter is used by the models as supporting pillars. Like mentioned above, each ICE-model can have more than one prototype, e.g. in the case of linear models a goal of the ICE-algorithm can be to create $i+1$ prototypes per model where i is the input-space dimension.

The initial ICE-network configuration starts from *scratch*. No prototypes and no models exist. The prototypes and the models, respectively, will be created depending on the presented training patterns and the current stage of the hybrid system. The first prototype will be generated at the position of the first input pattern. Because of no competing models or competing prototypes respectively, the receptive field can be *infinite*. The prototype offset is set to the value of the first target value. In the next step, this prototype, especially the combination of center and offset, is the base or supporting pillar of the first model (cf. Fig. 1a).

While presenting new training patterns, new prototypes are created. The center and offset location



(a) After presentation of the first pattern.



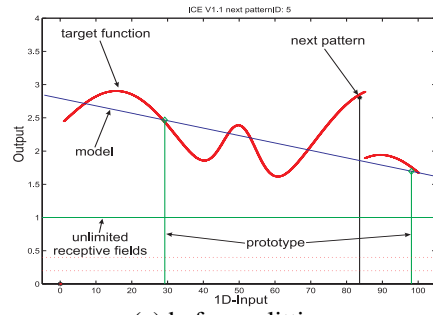
(b) After presentation of the second pattern.

Figure 1: (a) First prototype (with infinite receptive field) and the first model. (b) First model after presentation of the second training pattern. The model uses the two prototypes as supporting pillar.

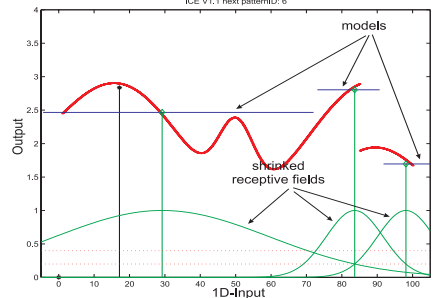
are determined by the same procedure as for the first pattern. If enough prototypes (e.g. in the case of linear models $i+1$ prototypes are needed, where i is the input space dimension) are generated the model can be adapted (cf. Fig. 1b).

As long as the model quality is sufficient, the active prototype (prototype with highest activation) moves into the direction of the current training pattern (in a similar way like Kohonen networks).

During the lifetime of the system or the training, respectively, it can be necessary to extend the network with further models. The first step is the extension of the network with a new *competing* prototype. In the same way as described for the first prototype, the center and its offset are determined. The only difference is that the prototype receptive fields of competing models have to be shrunk mutually. The shrinking process is comparable with the Dynamic Decay Adjustment algorithm (DDA) [7]. At the center of competing ICE-prototypes (prototype of a competing model) an upper limit activation is permitted. Using this criterion, all receptive fields can be determined in a straight forward way. The shrinking of the receptive field is followed by the adaptation of the new model.



(a) before splitting



(b) after splitting

Figure 2: Generation of new models. The old model is splitted and the receptive fields of competing models are shrunk.

The borders of the models are determined by the activations of their prototypes (cf. Fig. 2). In [6] a more detailed description of the algorithm is given.

In the following section a short overview of achieved one dimensional (1D), 2D and 11D results is presented.

3. Experimental results

The 1D and 2D experiments are done with nonlinear noncontinuous functions with 10% noise. For the high dimensional experiments 11D data from a project internal benchmark are used (cf. acknowledgement).

3.5 Uniform distributed data

In the first experiment 200 noisy patterns of a 1D function are presented. Only *one* network adaption step is done after the presentation of each pattern.

In Fig. 3 it can be seen that even with noisy training patterns a good approximation of the nonlinear noncontinuous target function is found. Remarkable is that the training patterns are *presented only once*. Prototypes are generated over the complete input space. This is because of the uniform distributed data selection in the interval from 0 to 100. Furthermore it can be seen that the positioning of

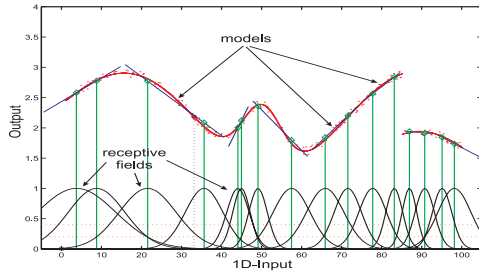


Figure 3: Approximation of a nonlinear noncontinuous function with linear models. Result after 200 randomly distributed training patterns (10% noise). Remarkable: the training patterns are presented only once.

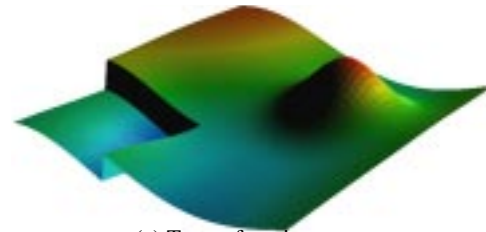
the prototypes and the size of the receptive fields depend on the model error or the nonlinearity of the target function, respectively. Similar results are achieved with 2D input space patterns. The target function is highly nonlinear and noncontinuous (cf. Fig. 4a). In Fig. 4b the result after 2500 training patterns is given. To get a smoother coverage, it is obvious that more and *smaller* linear models are needed to fit the nonlinear target function. This can be controlled with a model accuracy parameter, Fig. 4c shows the generated prototype layer of the example in Fig. 4b. It can be seen that in areas with strong nonlinearities, e.g. the hill in the right corner or the area of the noncontinuous step, the input space resolution with different prototypes is higher. Higher resolution means small receptive fields. Another 2D example is given in Fig. 5. Here the same training patterns are used as in the experiment in Fig. 4. One difference is that the linear models are merged together. The mixture of the linear models is done in a very simple way. It is the sum of all linear models weighted with their highest prototype activation at the current position. Obviously, there are a lot more ways to do a mixture of the linear models

In the presented experiments the training data has a uniform distribution. In the following section, some results with nonuniform dataset distributions will be shown.

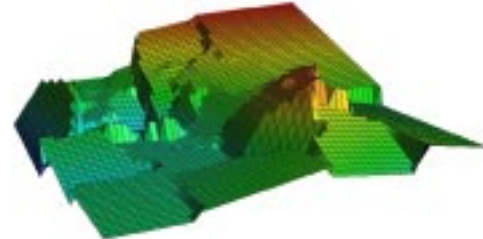
3.6 Nonuniform data distribution

In this paper a nonuniform data distribution means that only data from certain areas of the input space are selected for the network training. The areas can be connected or separated.

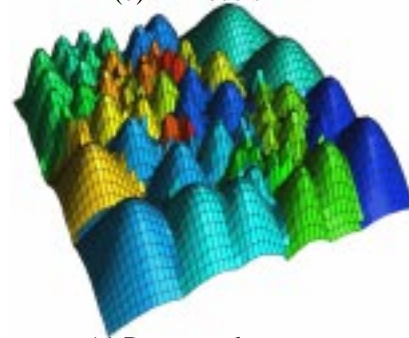
The following experiment uses the same 1D target function as in the experiment in Fig. 3. Instead of covering the complete input space interval ([0:100]), now the training data is presented only



(a) Target function



(b) 24 models



(c) Prototype layer

Figure 4: In this example 2500 pattern of the target function Fig. 4a are used for ICE training (10% noise, the training patterns are *presented only once*). (b) shows the result with an upper limit of 24 generated models and a very high model accuracy. The resulting prototypes are shown in (c).

within the two intervals 0 to 45 and 70 to 100. In the interval 45 to 70 no training patterns are presented.

Recall that the ICE-algorithm creates prototypes only in areas where training data is presented. Therefore, in the current experiment only within the two intervals [0:45] and [70:100] prototypes are generated (cf. Fig. 6) and a good approximation of the target function is found. In areas where no data was seen the activations of prototypes are low. The prediction accuracy can be bad but does not have to. The activations of the prototypes can be used to detect insecure areas where the prediction of the system may fail.

Similar results can be observed in the next 2D example. In contrast to the example in Fig. 4 which represents an uniform data distribution, we present training patterns with a nonuniform data distribu-

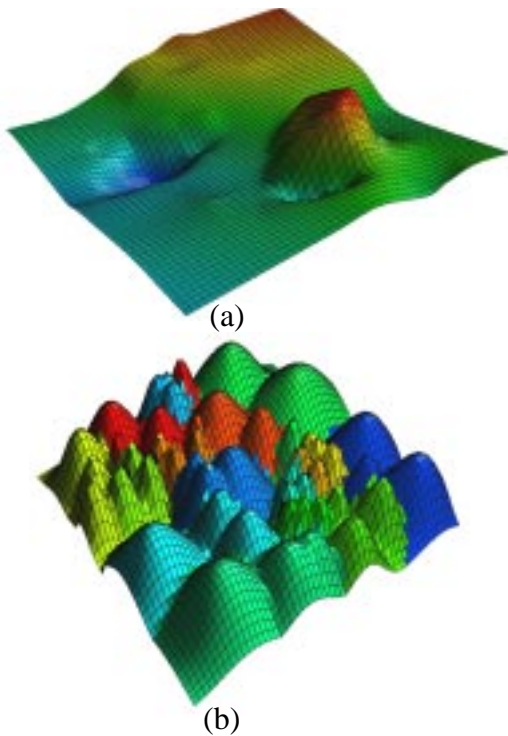


Figure 5: (a) ICE function (mixture of the linear models, grayscaleing corresponds to the prediction error); (b) the receptive fields of the ICE-prototypes. The training patterns are presented only once.

tion. The training patterns are selected from a ring around the center (dots in Fig. 7a). The surface of the learned function is shown in Fig. 7b. The surface grayscaleing corresponds to the prediction error (difference between the target value and the predicted value). In Fig. 7c it can be seen that the prototypes are arranged like a circle. The prototype ring corresponds to the training data distribution.

From Fig. 7b and its view from above in Fig. 8a it can be observed that in areas where training patterns were presented (area around the white ring in Fig. 8a) the error is very low (dark areas around the white ring in Fig. 8a). In the other areas it could be higher. The correlation of the prediction error and the prototype activation is printed in Fig. 8b. If the activation is low the error can be very high, because no training patterns were seen. But if the activation increases the error decreases. So the activation of the prototypes can be used directly as a confidence value for the ICE-output. The reason is that the prototypes represent the training data distribution.

In all 1D- and 2D-experiments useful network outputs are produced already during the initial learn-

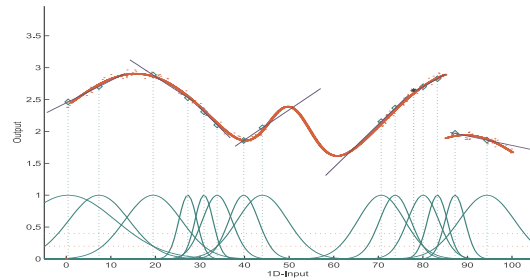


Figure 6: Approximation of a nonlinear noncontinuous function with linear models. In contrast to the example in Fig. 3, training data is presented only within the two intervals $[0:45]$ and $[70:100]$.

ing phase. Remarkable is that the training patterns are “presented only once”. No further training data storage is necessary. Even with few training patterns a good approximation of the target function with the linear models is found. In further experiments with up to 11-dimensional input-space patterns, comparable results to the 1D- and 2D-examples presented above are achieved. Comparable means that already after the first training pattern useful outputs are available. At this point it should be recalled that the linear models approximate the target function and *not* the RBF-Neurons. Compared to the method (special MLP approach) with the smallest RMS error the RMS error of ICE was slightly higher. However the special MLP approach needs the first 20000 patterns for training (the whole data set consists of 100000 data points). In contrast to that the ICE is able to predict useful output after the first pattern is presented. Furthermore ICE does not need multiple training data presentations. Only one network adaption step is necessary with each training pattern.

4. Conclusion

In this paper we have presented an incremental learning algorithm for classification and function approximation problems. With the adaptive prototypes the algorithm can be used directly for continuous learning tasks. One advantage of the ICE-algorithm is that no network topology parameter has to be determined in advance, i.e. the network starts from “scratch”. All parameters, e.g. the number of RBF-neurons in the first layer of the hybrid system or the number of localized models in the second layer, are adjusted autonomously based on the training data presented. Another advantage of the algorithm is the ability to learn the target function and the training data distribution simultaneously. The prototype activations can be used directly as a

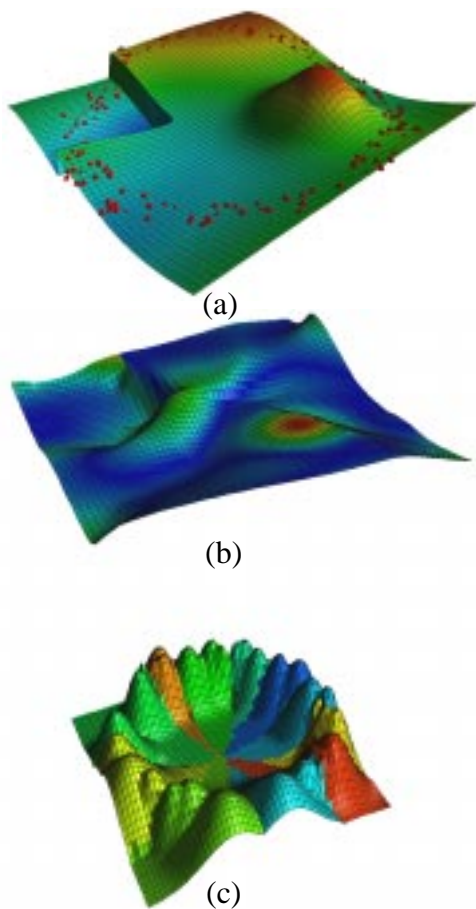


Figure 7: (a) Target function with 200 noisy training pattern (dots). (b) ICE function (mixture of the linear models; the grayscaleing corresponds to the prediction error). (c) ICE prototypes arranged as a ring (confirm the shape of the training pattern distribution in (a)).

confidence value of the network output, which is increasingly important in real world applications that strongly rely on the performance gained by neural networks. Furthermore, useful network outputs are already available when the second pattern is presented and during the whole initial learning phase. This is very important for industrial applications, e.g. plants can be put into operation much faster.

Acknowledgement

This research was sponsored by the Federal Ministry of Education, Science, Research and Technology under grant number 01 IN 505 B.

References

[1] Martinetz T., Protzel P., Gramckow O., and Sörgel G.. Neural network control for steel rolling mills. in B. Kappen and S. Giele, editors, Neural

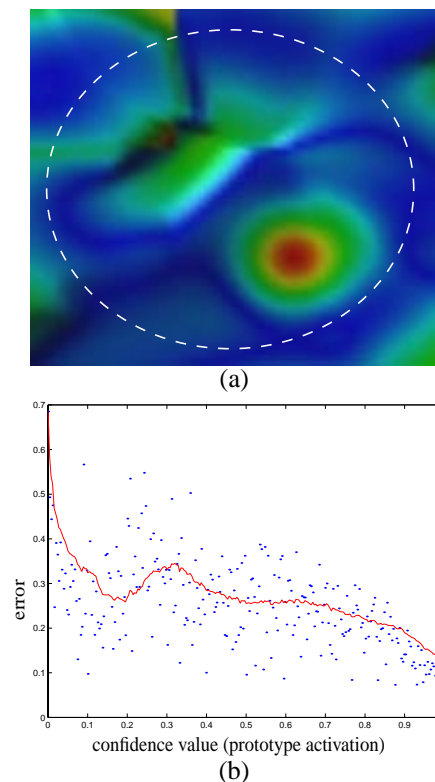


Figure 8: (a) view from above of Fig. 7b (dark=small error; light=large error). (b) correlation of prototype activation and prediction error.

- Networks: artificial intelligence and industrial application, Berlin, 1995, Springer-Verlag.
- [2] Carpenter, G., Grossberg, S.: A massively parallel architecture for a selforganizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, pp. 54-115., 1987
- [3] Poggio, T., Girosi, F.: Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks. *Science* 247, pp. 978-982, 1990.
- [4] Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59-69, 1982.
- [5] Hecht-Nielsen, R.: Counterpropagation networks. *Proc. of the IEEE First International Conference on Neural Networks*, Vol. 2, pp. 19-32, San Diego, CA 1987.
- [6] Tagscherer, M.: ICE - an Incremental Hybrid System for Continuous Learning, *Proceedings of the 8th International Conference on Artificial Neural Networks (ICANN'98)*, Skövde, Schweden, Sep 1998, pp. 597-602.
- [7] Berthold, M., Diamond J.: Boosting the Performance of RBF Networks with Dynamic Decay Adjustment. *Advances in Neural Information Processing Systems 7*, Morgan Kaufmann Publishers, San Mateo, CA, 1994.