ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 365-370

Training Activation Function in Parametric Classification

V. Colla†, L.M. Reyneri††, M. Sgarbi†
† Scuola Super. Sant'Anna, Pisa (I), †† Politecnico di Torino (I)
e.mail: colla@sssup.it, reyneri@polito.it, mirkosg@tin.it

Abstract. This work shows how to train the activation function in neuro-wavelet parametric modeling and how this improves performance in a number of modeling, classification and forecasting.

1. Introduction

Neuro-wavelet networks (**NWN**'s) [1, 2, 3] are widely used in modeling, forecasting and classification problems, because they are good approximators of strongly non-linear functions. Yet, it is mandatory to accurately select the network structure, in order both to obtain good performance and to reduce the number of free parameters, and consequently the size of training set.

An interesting solution to many applications comes from neuro-wavelet parametric modeling **NWPM** [4], which consists of describing a given problem in terms of a *parametric model* based on NWN's. A set of predefined parameters (weights and centers of the NWN) is computed (via training) from a collection of *numeric samples* of the problem. The extracted parameters have a strong relationship with the problem, of which they are a *compact representation*, and they allow either to forecast future samples, or to predict the behavior of the problem in different operating conditions, or to classify the samples.

Although it has been proven by several authors that any bounded function can be approximated with any given accuracy with a NWN, one of the requirements of NWPM is to use the *smallest possible number of parameters* which provide the desired accuracy. Therefore one key issue in NWPM is to find the optimal activation function (e.g. Gaussian, Wavelet, sigmoid, spline) which minimizes the number of parameters that provide a given accuracy).

The optimal function can seldom be derived analytically by examining the problem (that happens only in *structured neuro-wavelet networks*). Instead, in most cases the optimal function is unknown and therefore most users of NWPM tend to choose the activation function they are more familiar with. In other cases, an empirical search from a set of commonly used functions is performed.

Scope of this work is to show how the optimal activation function can be *trained*. Training activation function is not yet widely used, although an example is proposed in [4]. Training algorithm derives from neuro-fuzzy unification [1] and may significantly improve modeling performance.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 365-370

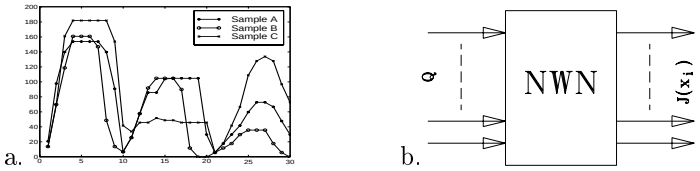

Figure 1: a) Examples of measured optoelectric characteristics of specimen. b) Block diagram of traditional neuro-fuzzy approaches

1.1. Case Study

We have been dealing with an application where a set of cylindrical specimen had to be classified according to their geometrical (diameter, thickness), electrical (conductivity) and optical (brightness) properties. Such properties had to be measured with a set of three very cheap sensors scanned over the specimen, which provided as many time-varying signals, as shown in fig. 1 (signals of the three sensors are plotted aside on the same plot, for three different samples).

The specimen had to be classified according to the shape $J(t)$ of signals. Due to the poor performance of available sensors, the relationship between the measured shape and specimen properties was difficult to express analytically.

We call: $\vec{\mathcal{Q}} \in \Re^4$, the *vector of geometrical and optoelectric properties of specimen*, associated with $J(t) : \Re \rightarrow \Re$, the *sensor signal* as a function of time t; $\vec{\mathcal{J}} \in \Re^{10}$, the *sensor vector* containing 10 samples $J(t_i)$ at regular intervals.

Scope of the problem was to filter out the unavoidable measurement noise and to compensate for sensor non-idealities, in order to provide a clean signal $J(t)$ to the classifier and to reduce misclassification.

2. Parametric Modeling and Characterization

Most traditional approaches are based on the black-box approach shown in fig. 1, where a NWN predicts the elements of $\vec{\mathcal{J}}$ (namely, the samples of $J(t)$ at predefined points) as a function of input vector $\vec{\mathcal{Q}}$, or vice-versa.
At first, we drew some preliminary considerations on such approaches:

1. the number of network outputs equals the number of measured points. From fig. 1 it can be observed that all signals are relatively slowly varying, therefore statistical correlation of adjacent elements of $\vec{\mathcal{J}}$ approaches unity (≈ 0.93 for sensor 1) as well as, consequently, the correlation between weights of adjacent neurons.

2. Approximation errors can produce estimates of signals which are physically non plausible (for instance, a local increase instead of a decrease).

3. The positions where the signals are measured are not evenly distributed and often differ among different manufacturers, therefore signals cannot always be compared directly. Also the number of points may vary.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 365-370

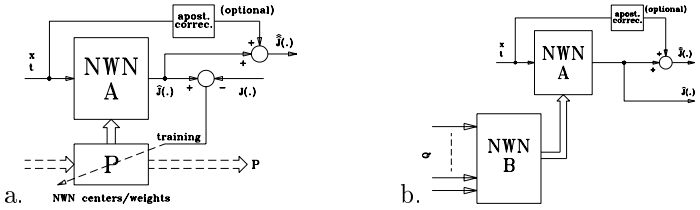

Figure 2: Block diagram of: a) NWPM; b) NWPM estimator/forecaster.

For all these and a few other reasons, we have decided to use a NWPM approach [4], which is composed of from one to three cascaded blocks (see fig. 2):

- A small NWN (A) is used as a *parametric model* of the signal, that is, as a function $J(t)$ of the single network input t. The collection of free parameters of A (weights, centers and biases) constitutes a vector $\vec{\mathcal{P}}$ which uniquely identifies an estimate $\hat{J}(t)$ of $J(t)$ and thus of vector $\vec{\mathcal{J}}$.

  As $\vec{\mathcal{P}}$ is the vector of the free parameters of A, it can be evaluated by simply *training* A, to reduce the output error $\|\hat{J}(t) - J(t)\|$.

- An optional *a-posteriori corrector*, to reduce estimation error. We noticed that each estimate $\hat{J}(t_i)$ of $J(t_i)$ always was slightly *biased* and the bias

$$\mathrm{E}\left(\hat{J}(t_i) - J(t_i)\right) = f(t_i)$$

  was a function of t_i. We have therefore measured and tabulated that particular function $f(t_i)$ and subtracted from the estimate (namely, $\hat{\hat{J}}(t_i) = \hat{J}(t_i) - f(t_i)$), as shown in fig. 2.a.

  The drawback of having an a-posteriori model corrector is twofold: an additional block is required, and the function $f(t_i)$ must be tabulated, therefore cannot be continuous, as would $\hat{J}(t)$ be. This corrector can be avoided by training the activation function of the NWN, as described in sect. 3..

- A larger NWN (B) is used as a *parameter estimator* which predicts the parameter vector $\vec{\mathcal{P}}$ (instead of $\vec{\mathcal{J}}$) as a function of input vector $\vec{\mathcal{Q}}$.

The NWPM can be used in either of the following fashions:

1. by training the network A alone, to get just a compact representation $\vec{\mathcal{P}}$ of the problem (to be used, for instance, in classification instead of $\vec{\mathcal{J}}$);
2. by supplying a representation $\vec{\mathcal{P}}$ to A, to reconstruct the original function;
3. by combining the two previous fashions: network A is first trained, then the parameter vector $\vec{\mathcal{P}}$ is used to reconstruct the original signal (for instance, to clean the measurements from noise);

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 365-370

Sensor	1^{st} layer		size	SRMSE		Improvement w.r.t. Gaussian	
	$F(z)$	neurons	of $\vec{\mathcal{P}}$	$\epsilon_{\rm av}$	$\epsilon_{\rm co}$	$\Delta\epsilon_{\rm av}/\epsilon_{\rm av}(\%)$	$\Delta\epsilon_{\rm co}/\epsilon_{\rm co}(\%)$
1	Gaussian	-	4	0.332	0.255	-	-
	trained	6	4	0.171	0.166	48	25
	trained	8	4	0.220	0.214	34	16
2	Gaussian	-	4	0.317	0.264	-	-
	trained	6	4	0.196	0.177	38	33
	trained	8	4	0.196	0.186	38	30
3	Gaussian	-	4	0.125	0.103	-	-
	trained	4	4	0.126	0.088	0	15
	trained	8	4	0.100	0.087	20	15

Table 1: Approximation error (SRMSE) of network A, with $H = 1$.

4. by letting network B predict the parameter vector $\vec{\mathcal{P}}$ instead of $\vec{\mathcal{J}}$ from the input vector $\vec{\mathcal{Q}}$. Signal $\hat{J}(t)$ can then be reconstructed from $\vec{\mathcal{P}}$.

In all cases $\vec{\mathcal{P}}$ is the main input or output of NWPM. Advantages and drawbacks of NWPM are described in [4].

2.1. Performance Index

As a performance index for all NWN's, we adopt the *Standardized Root Mean Square Error* (**SRMSE**):

$$\epsilon = \sqrt{\frac{\sum_{p=1}^{P}\sum_{j=1}^{M}(y_j^p - \hat{y}_j^p)^2}{\sum_{p=1}^{P}\sum_{j=1}^{M}(y_j^p - \overline{y})^2}} \qquad \text{where} \qquad \overline{y} = \frac{1}{PM}\sum_{p=1}^{P}\sum_{j=1}^{M} y_j^p \tag{1}$$

where P and M are, respectively, the number of samples in the training (or validation) set and the number of network outputs; y_j^p is the j-th component of the p-th output vector $\vec{Y}^p$ in the training (or validation) set, while $\hat{y}_j^p$ is the corresponding network estimate.

Table 1 lists the SRMSE of network A both alone ($\epsilon_{\rm av}$) and followed by the a-posteriori corrector ($\epsilon_{\rm co}$), for the case study, with Gaussian activation function (the commonly used function which empirically provides best performance).

3. Training Activation Function

The need for an a-posteriori model corrector indicates that the *shape* of $\hat{J}(t)$ is far from optimal and definitely needs improvements. The performance of different networks A mainly depend on how well their activation function fits the signal $J(t)$. See [4] on how to choose the best NWN for NWPM.

In practice, none of the traditional functions is optimal and no other commonly used function would be significantly better under this respect. The only way to improve model accuracy would be to increase the number of hidden neurons, but this would increase the number of free parameters (size of $\vec{\mathcal{P}}$), thus reducing the effectiveness of NWPM approach.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 365-370

A solution to the problem would be to design an ad-hoc activation function to minimize $\|f(t_i)\|$ and therefore the average modeling error. Direct minimization is not trivial, therefore we tried the new idea of "training" the activation function, as described below.

By using the terminology and symbols proposed in [1], we first selected, for A, a two-layers $1 \times H \times 1$ WRBF-0($F(z)$)-0(lin) (in the old terminology, an MLP with H hidden units with $F(z)$, plus 1 output neuron with linear activation, respectively), for a total of $3H+1$ parameters:

$$y(x) = \mathcal{H}_0^{\text{lin}}\left(\mathcal{H}_0^{F(z)}\left(x; \vec{C}^1, \vec{W}^1, 0\right); 0, \vec{W}^4, \Theta^4\right) \tag{2}$$

where the activation function $F(z)$ is defined by another two-layer $1 \times N \times 1$ WRBF-0(tanh)-0(lin) which is known to be a universal approximator:

$$F(z) = \mathcal{H}_0^{\text{lin}}\left(\mathcal{H}_0^{\text{tanh}}\left(z; \vec{C}^2, \vec{W}^2, \vec{0}\right); \vec{0}, \vec{W}^3, \Theta^3\right) \tag{3}$$

By substituting (3) in (2) we get a four-layers $1 \times H \times (NH) \times H \times 1$ WRBF-0(lin)-0(tanh)-0(lin)-0(lin), where input and output layers (1^{st} and 4^{th}) coincide with the MLP, while the inner layers (2^{nd} and 3^{rd}) implement its activation function.

This four-layer NWN can be trained using a traditional backpropagation rule [1, 2], with the same training set used for network A. Once trained, the parameters in (3) (namely, 2^{nd} and 3^{rd} layer) are *frozen and they are identical for all signals*, as they define the shape of a *fixed but trained* activation function. The only parameters which remain free for each profile are the $3H+1$ parameters in (2) (namely, 1^{st} and 4^{th} layer) which compose vector $\vec{\mathcal{P}}$.

It must be pointed out that, although that appears to be a traditional four-layer network, in practice the two inner layers are trained apart and, once trained, their weights are frozen and they are not part of the compact representation $\vec{\mathcal{P}}$. Only the two outer layers are part of the NWPM approach and their weights are the elements of $\vec{\mathcal{P}}$.

Training of the activation function takes place in a slightly uncommon way: the input and output layers are trained with the nominal values of η, while the two inner layers are trained with much smaller training coefficients (namely, K times lower, where K is the size of the training set). The first specimen data are applied to the network, which is trained for 200 epochs, then next specimen is applied for another 200 epochs, etc., until all training specimen have been applied. This process (on the whole training set) is then repeated a few times. By doing so, the two outer layers can learn the shape of each profile in less than 200 epochs, while the two inner layers require all the samples in the training set (with 200 epochs each) to train.

As a consequence of such training, the inner layers slowly learn from *all* the actual profiles the optimal shape which reduces the average approximation error to a minimum. After a sufficiently long training (as many as $\approx 4 \cdot 10^5$ epochs, but each epoch is very fast to compute, therefore the whole training can last just a few minutes), the shape of the activation function has been

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 365-370

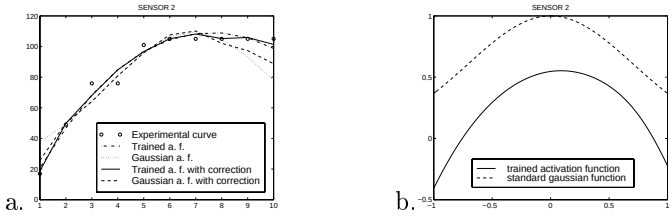

Figure 3: a) Comparison of estimated signals for sensor 2. b) Trained $F(z)$.

learned and is then *frozen forever*, to be later used to associate each profile with is compact representation $\vec{\mathcal{P}}$. For what convergence regards, the two inner layers do have such a small learning coefficient that their convergence is always guaranteed. Remember that this is the major difference between a 2-layers NWN with trainable activation functions and a 4-layers NWN.

4. Performance and conclusions

Table 1 shows that modeling error with the trained $F(z)$ reduces by about 35% (average, without a-posteriori corrector). Furthermore the a-posteriori corrector has less effects on performance, therefore it can be removed. Figure 3.a compares measured and predicted signals for sensor 2, while fig. 3.b compares the Gaussian and the trained activation functions.

This paper has proposed a method to successfully train the activation function in a class of neuro-wavelet parametric modeling problems. Training has significantly improved the modeling and classification performance for the proposed case study and for some other industrial applications not reported here.

Acknowledgments

This work has been supported by Italian project MADESS II "Architectures and VLSI devices for embedded neuro-fuzzy...", contract ASI-ARS 98.

References

[1] L.M. Reyneri, "Unification of Neural and Wavelet Networks and Fuzzy Systems", in *IEEE Trans. on Neural Networks*, Vol. 10, no. 4, July 1999, pp. 801-814.

[2] S. Haykin, "Neural Networks: A Comprehensive Foundation", *Mc Millan College Publishing Company*, New York, 1994.

[3] Q. Zhang: "Using Wavelet Network in Non-parametric Estimation," *IEEE Transactions on Neural Networks*, Vol. 8, No. 2, pp. 227-236, March 1997.

[4] V. Colla, L.M. Reyneri, M. Sgarbi, "Neuro-Wavelet Characterization of Jominy Profiles of Steels", in *Journal of Integrated Computer-Aided Engineering*, 1999, John Wiley & Sons, New York (NY). (to be printed)