ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 331-336

Mixture of Experts and Local-Global Neural Networks

Mayte Fariñas and Carlos E. Pedreira

Catholic University of Rio de Janeiro PUC-Rio/DEE
CP.38063 RJ, Brazil, CEP 22452-970 e-mails: {mayte/pedreira}@ele.puc-rio.br

Abstract - In this paper we investigate mixture of experts problems in the context of Local-Global Neural Networks. This type of architecture was originaly conceived for functional approximation and interpolation problems. Numerical experiments are presented, showing quite nice solutions. Because of its local characteristics, this type of approach brings the advantage of improving interpretability.

1. Introduction

The Local-Global Neural Network (LGNN) architecture was introduced in [1] and [2]. This model was originaly proposed to approach functional approximation and interpolation problems. The main idea is to approximate the original function by a set of very simple approximation functions and by functions called activation-level functions. The considered algorithm shows an interesting capability to reconstruct the function from local estimates along domain of interest. In this paper we propose the use of LGNN model in the context of Mixture of Experts. Numerical examples are presented considering polynomials and Neural Networks experts.

2. Local-Global Neural Network Architecture

The central idea of LGNN model is to express the inter-output mapping by a piecewise structure. The network output is constituted by a combination of several pairs, each of those composed by an approximation function and by an activation-level function. The activation-level functions define the role of an associated approximation function, for each subset of the domain. Partial superposition of activation-level functions is allowed. In this way, the problem of approximation functions is approached by the specialization of neurons in each of the sectors of the domain. In other words, the neurons are formed by pairs of activation-level and approximation functions that emulate the generator function in different parts of the domain.

Let $\{x_i\}_1^n$ be the subset of the available data that is used for training. For algebraic and notational simplicity we will consider $x \in \Re$ (x subscript is omitted).

Generalization for $x \in \Re^n$ is straightforward. Let $x \in \Re$, the activation-level function is defined by:

$$B(x,\psi_B) = -\left[\frac{1}{1+\exp(d(x-h^{(1)}))} - \frac{1}{1+\exp(d(x-h^{(2)}))}\right],$$

where $\psi_B = (d, h^{(1)}, h^{(2)})^t$ is a vector of real parameters. Parameter d is related to the function declivity while parameters $h^{(1)}$ and $h^{(2)}$ delimit the domain sector where the associated activation-level function is more active.

The global approximation of a function can be approached through a partition of the function domain. In each region of the partition, the objective function can be locally approached by an approximation function. The degree of specialization in a region is given by the value of the activation-level function. Partial superposition of these functions may occur providing better quality of the intended mapping. The function g(x) that approximates the objective function can be expressed as:

$$g(x) = \sum_{j=1}^{m} B_j(x,\psi_{B_j})\kappa_j(x,\psi_{\kappa_j}) \qquad (1)$$

where $\psi_{\kappa j}$ is a vector of parameters associated to the approximation function κ_j, to be estimated. In the LGNN architecture (See Figure 1), each neuron is constituted by a pair {activation-level function; approximation function}. The input is connected to the nodes producing as their output the activation-level and approximation function, $B_j(x)$ and $\kappa_j(x)$ product. The parameters to be estimated are associated to neuron-pairs: 3 parameters for function B and the number of parameters of approximations functions (for example: 2 for linear case, 3 for quadratic). The output of the jth neuron is $B_j(x)\ \kappa_j(x)$, and the network output is given by equation 1.

If one defines, for each neuron, a vector of parameters $\Im^j = \left(d_j, h_j^1, h_j^2, \psi_{\kappa_i}\right)$. Then, the set of parameters to be estimated can be written as $\Im = (\Im^1,...,\Im^m)$. The vector $\Im$ could be estimated as a minimum argument of the mean square error (MSE). [1] and [3] a theorem that gives theoretical consistency to the proposed methodology is stated and proved. It guarantees that any L2-integrable function may be approximated by functions $\{g^m(x)\}$. They also present a heuristic to choose starting values in the parameters estimation procedure. This heuristic provides accelerated convergence, and the initial choice of the parameters $h_i^{(1)}$ and $h_i^{(2)}$, i=1,...,m may reflect a priori knowledge on the function domain.

3. Local-Global Neural Networks and Mixture of Experts

The idea of using a mixture of experts for achieving a complex mapping function, based on a “divide and conquer” strategy, was proposed by [4]. The motivation for the development of this model is twofold: first, the ideas of [5], viewing competitive adaptation in unsupervised learning as an attempt to fit a mixture of simple probability distributions into a set of data points; and the ideas developed in [6] using a similar modular architecture but a different cost function.

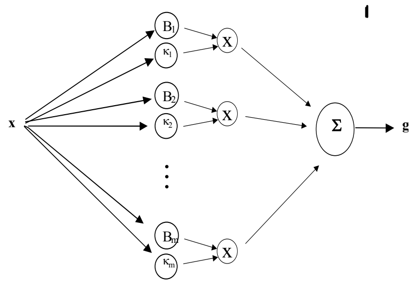

Figure 1 – The proposed architecture

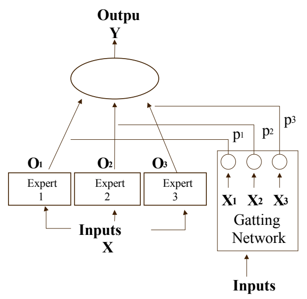

Figure 2 – Multi-experts architecture

In general terms, the problem of mixture of experts, can be presented as in [4]: Let us assume that the set of training cases can be naturally divided into different subgroups that correspond to distinct subtasks according to knowledge of the problem. The interference (fitting errors) can be reduced by using a system composed by some experts and a gating network, that decides which expert should be used for each training case. Each expert can be a neural network or any other model and all experts receive the same inputs and have the same number of output. The gating network is also a Feedforward Neural Network and could receive another type of inputs. The output of this network gives us the probabilities of selecting each expert.
In training phase one of the probabilities p_i approaches one while all the others are vanished. In this paper, we propose to use as outputs of the gating network, activation-level functions, as used in [1] and [2], instead of the probability concept. Using activation-level functions provides a smooth transition between experts. In the following section preliminary numerical results are presented. In order to evaluate the operation of the method and analyze how the division of the expert's domain is made, simple examples are used.

3. Numerical Results

In this section, two types of numerical experiments are presented. We first simulate experiments with 2 and 3 experts, where it is considered that the outcome of each expert can be fitted to a linear or a quadratic function. After that, we present a more complex experiment, where a neural network is considered as one of the experts.
Training data were generated without any noise. In experiments 1 (2 and 3 experts, 1 and 2-degree polynomials) the first figure (Figure 3a, 4a) represents the hypothetical situation that generated the data, in terms of approximation and activation-levels functions. The training data (x,f(x)) are generated in accordance with equation (1). Levenberg Marquardt algorithm was used for optimization purposes. Data used in the generalization stage had been generated from a uniform distribution in the considered interval, using about 40% of the total of data in the training stage. We consider the MSE (Mean Square Error) and MAPE (Mean Percentile Absolute Error) as error measures.

Experiments with Polynomials Experts.

For the following examples, the functions used to generate mixture are represented in Figure 3-a. The training set is formed from this situation in *x*=[0:0.1:6] in Example 1 and *x*=[0:.1:10] in Example 2. The third experiment is composed by 3 experts, where all are polynomials. Figure 4a shows the theoretical shape used to generate the data, following equation 1 in x=[0:0.1:30]. Table 1 presents the results obtained for all these examples.

Table 1. Results for a Mixture of polynomial experts

Ex	No.Experts	Iterations	Training		Generalization	
			MSE	MAPE	MSE	MAPE
1	2	136	0.0406	0.1731	0.0566	0.2052
2	2	75	0.0377	0.5198	0.0348	0.5327
3	3	133	0.0110	0.6586	0.0265	1.2868

The results for 3-expert examples, using linear and quadratic experts are excellent. The algorithm converges in relatively few iterations with pretty good training errors. The values obtained in the generalization phase can be considered good.

Experiment with MLP experts

Problems that involve more complex functional representations need more sophisticated experts. For example, to simulate the function:

$$f(x) = \begin{cases} (x+2)^2 & x \in [-2,0] \\ sin(x) + sin(2x) + sin(6x) + 4 & x \in [0,3] \end{cases}$$

it is reasonable to use two experts, one to pick-up the quadratic behavior and a neural network, for the interval where the function has a more complex functional form. We use here a MLP with one hidden layer and 5 or 6 neurons (MLP(5) and MLP(6)). The starting solution is obtained by fitting each expert separately. Table 2 summarizes the obtained results in this in case. The set of points (x, f(x)) with x=[-2:0.05:3] was considered as training set.

Table 2. Results for the Mixture of two experts [Pol(1) and MLP(n)]

		Training		Generalization	
Neurons	Iterations	MSE	MAPE	MSE	MAPE
5	173	0.0387	4.2958	0.0493	5.3675
6	67	0.0021	1.1919	0.0026	1.7870

We remark that it is not possible to improve the training results without losing generalization capacity. With 5 neurons we lose a little in fitting at the end of the interval (Figure 5a); this indicates that a modification in the architecture is needed. With 6 neurons an excellent fitting is obtained after 127 iterations. It is interesting to observe that the method is able to allocate each expert to its interval, obtaining a mixture in the intersection of intervals that smooth the transition from the linear expert to the MLP expert.

In this last experiment we deal with two MLP neural networks as experts. Each MLP expert has one hidden layer. The data had been generated from the function:

$f(x)=\left\{\begin{matrix}(x+2.5)(x-2)(x-5) & x\in[-3,5] \\ (x-5)(x-8.5)(x-13.5)(x-15.5)-0.4 & x\in[5,15]\end{matrix}\right\}$ where each interval corresponds to a network with 5 and 11 neurons in the hidden layer respectively. The heuristic proposed in [1] was used to find an appropriated starting solution. The results are illustrated in figure 6. The obtained errors has MAPE equals to 0.08% and 0.1% for training and generalization stages.

4. Final Remarks

In this article we focused the mixture of experts problems with the ideas of Local Global Neural Networks. Numerical experiment results showed quite satisfactory solutions, emphasizing the potentiality of the considered method. This type of approach brings the advantage of improving the ability of interpretation since the location of the activation-level functions can indicate changes in the model. The use of the method in more complex problems and with real data is under investigation. A study on the identifiability of the model is being lead and the results are expected improve the robustness of estimation.

References

1. Fariñas, M. e Pedreira, C.E(2002) “New Neural-Network Based Approach for Function Approximation” Submitted to IEEE Tr. on NN (06/2002-Paper 379)
2. Pedreira, C.E, Fariñas, M. and Pedroza, L.C. “Local-Global Neural Networks for Interpolation” in Proc. ICANNGA’2001 Czech Republic. Ed. NY: Springer Computer Science. April, 2001
3. Fariñas, M. e Pedreira, C.E(2002) “Missing Data Interpolation By Using Local-Global Neural Networks” Int. Jr. of Engineering Intelligent Systems. June 2002.
4. Jacobs, R.A., Jordan, M.I,. Nowlan, S.J and Hinton,G.E. (1991). Adaptative Mixture of local Expert Neural Computation, vol. 3, pp. 79-87
5. Nowlan, S.J.(1990). Maximum likelihood competitive learning Advances in Neural Information. Processing Systems, vol.2, pp.574-582, San Mateo, CA:Morgan Kaufmann.
6. Jacobs, R.A(1990). Task Decomposition Trough Computation in a Modular Connectionist Architecture. Ph.D. Thesis, University of Massachusetts.
7. Nowlan, S.J & Hinton (1991). Evaluation of Adaptative Mixture of Competing Experts. Advances in Neural Information. Processing Systems, vol.3, pp.774-780, San Mateo, CA: Morgan Kaufmann.

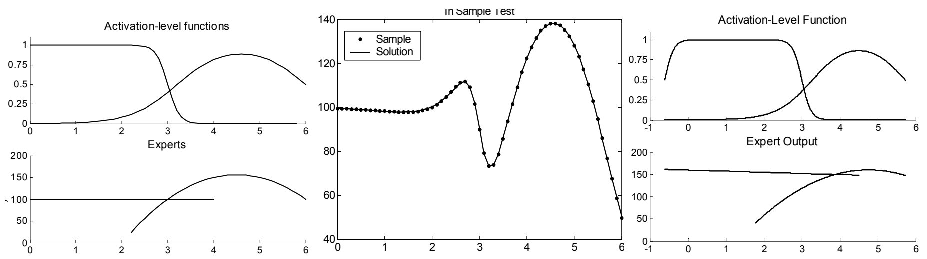

Figure 3- Mixture of 2 Experts. Example 1. a)Theoretical shape. b) in Sample test c) Shape solution.

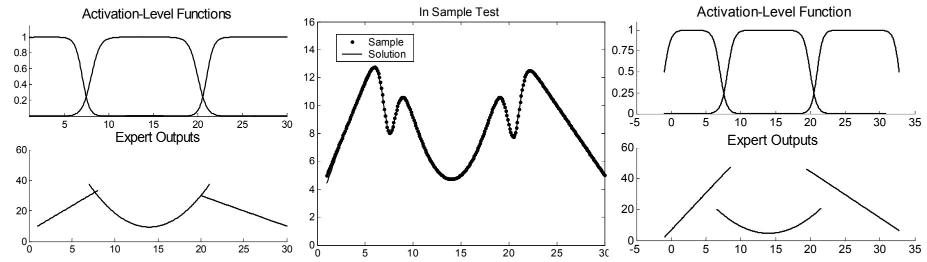

Figure 4- Mixture of 2 Experts. Example 3. a)Theoretical shape. b) in Sample test c) Shape solution.

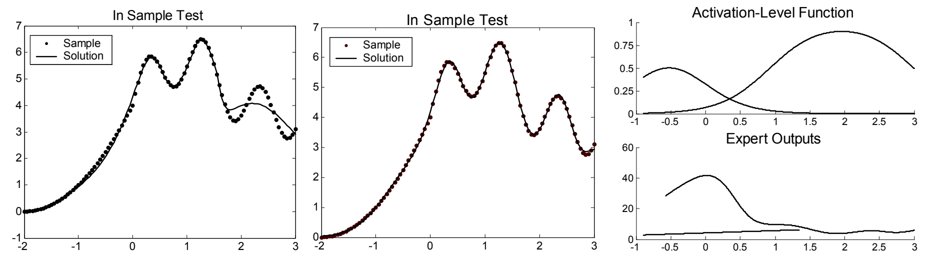

Figure 5- Mixture of polynomial and MLP(n). a, b: in Sample test MLP(5) and MLP(6). c) Shape solution for MLP(6)

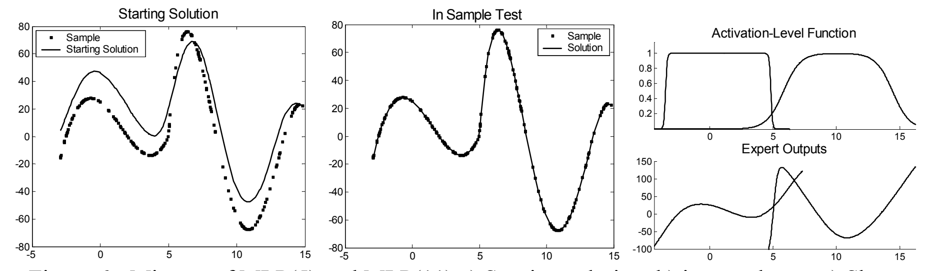

Figure 6 - Mixture of MLP(5) and MLP(11) a) Starting solution. b) in sample test c) Shape solution