Input Partitioning to Mixture of Experts

Bin Tang, Malcolm I. Heywood, *Member, IEEE,* and Michael Shepherd

***Abstract*— Under the Mixtures of Experts architecture a method for 'designing' the number of experts and assigning local 'regions' of the input space to individual experts is investigated. Classification performance and transparency of the scheme is found to be significantly better than that using a standard Mixtures of Experts.**

***Index terms*—Mixtures of Experts, Self-Organizing Feature Map, Potential Function Clustering, Classification.**

I. Introduction

Divide and conquer has been shown to provide a particularly fruitful approach to machine learning in practice e.g. decision trees. Under this spirit, Jordan and Jacobs in [1, 2] proposed and studied a mixture of experts (MoE) model in which the divide and conquer paradigm was explicitly supported. Given a supervised learning context, the MoE approach uses several neural networks in parallel to provide a modular solution to the overall problem. However, rather than *a priori* assume a piecewise linear solution to the problem, an additional 'gate' network is used to associate input conditions with experts. This may result in multiple experts being 'mixed' or 'switched between' in order to provide the desired mapping. In addition, the scheme can be extended to provide multiple layers, thus supporting divide and conquer multiple times [1, 2].

In this work our interest lies on identifying what inputs to feed each expert. That is to say, without *a priori* knowledge, each expert and gate receives all inputs. Naturally, applying a transform to the original data, so reducing the initial dimensionality, can alleviate this problem and the raw count of inputs routed to each MoE 'component' (e.g. principal component analysis). This however does not change the fact that all inputs require forwarding to each 'component' of the MoE network. The interest of this work is to assess the significance of using a cluster based preprocessing step to partition the initial input space. Our hypothesis is that different partitions may be forwarded to different experts without reducing performance, thus supplying additional transparency regarding the solution formed. Moreover, such a scheme also explicitly determines the (single layer) MoE architecture – the number of experts, their inputs and inputs received by the gate network.

The approach followed here is therefore to first formulate a preprocessing scheme capable of introducing the necessary partitions, section II. Secondly, to compare MoE solutions formulated with and without the proposed preprocessing, section IV. Hereafter, MoE-I is used to denote the standard approach, and MoE-II used in the case of the pre-processed scheme. Section III is devoted to the detailed description of all the algorithms involved. In section V, we will further discuss the results and highlight potential future work.

II. Design Issues

The design issues are mainly concerned with the formation of the preprocessing module in MoE-II.

A. Primary clustering of input data

Preprocessing partitions the input data, and therefore provides hints regarding the number of expert networks necessary in the MoE and their relations to the input space. Kohonen's self-organizing feature map (SOM) is used for this purpose [3]. In particular, nodes in the SOM provide a topological organization of features drawn from the input space (i.e. "*spatial locations of the neurons in the lattice are indicative of intrinsic statistical features contained in the input patterns*" [3]). As a consequence, SOM nodes may themselves be grouped into regions/ partitions and employed as inputs to different expert networks of the MoE. That is to say, different experts are restricted to different partitions from the SOM. Finally, as the gate of the MoE is responsible for determining the expert mixture, the SOM node representing the 'mean' of each SOM partition is forwarded to the gate. Thus, for '*q*' partitions identified by the SOM, there are '*q*' experts and '*q*' inputs to the gate network. Note however that the objective of the process is to identify the significance of knowledge, gleaned from the data itself, to structure the inputs presented to the MoE. This is rather different from employing principal component analysis to the initial data; in which case each expert and gate still see the *same* input space. Moreover, such a transform does not necessarily represent a good pre-processing step for classification problems on its' own (c.f. most expressive features and most discriminating features [4]). Finally, the computational over head of PCA on some applications can preclude it's application.

Formulating partitions on SOM architectures has been of particular interest to the data-mining community [5].

Dalhousie University, Faculty of Computer Science. Halifax, Nova Scotia, B3H 1W5. Canada. http://www.cs.dal.ca

Moreover, by first using an SOM and then partitioning it, significant computational speedups are acknowledged [5], as well as reducing sensitivity to the selection of the SOM node parameter (number of nodes and topology of initial map). Thus the second level clustering will 'prune' superfluous SOM nodes (represent noise in the input data [6]) as well as identifying the regions forwarded to the MoE.

B. Clustering on SOM

Properties of significance to the clustering problem context here include: 1) robustness in preference to outright optimality of the clusters, 2) the number of regions should be derived from the data, 3) provides crisp as opposed to fuzzy regions. To this end the Potential Function method is employed [7], section III.B. The result of this clustering process is an assignment of SOM nodes into regions fed to individual MoE experts or to the set of redundant nodes. Moreover, the Potential Function method as applied here defines a region 'center' point in terms of corresponding SOM node. Each instance of these nodes is forwarded to the gating network of the MoE to provide the 'global' view necessary to establish expert interaction.

III. LEARNING ALGORITHM DETAILS

The Mixture of Experts model (MoE) of [1], [2] is followed for application to a classification as opposed to a regression problem context. Algorithms for MoE-I and MoE-II are described in detail in the following subsections. All the programming is implemented under MATLAB 5.3 development environment [8].

A. MoE-I

MoE-I represents the standard MoE configuration as trained under a gradient descent-learning algorithm. An Expectation-Maximization algorithm has also been reported [1, 2], but given the interest in the preprocessing process in this work, it is not used here. The network is composed of K experts and one gating network. Each expert is composed of M input nodes and one output node. The gating network is composed of M input nodes, and K output nodes, such that there is a single output for every expert. Each expert and the gating network are fully connected, but in the case of this work are limited to a single layer. The whole network is trained under a supervised learning context. The detailed learning process is summarized as follows:

- Assign random weight values to all the links between input nodes and output nodes for all the experts, where w_{ki}, k = 1...K, i = 1...M; and for the gating network weights, a_{ik}, k = 1...K, i = 1...M.
- Present an input pattern, ($\boldsymbol{x}$, d), $\boldsymbol{x}$ is a vector of size M, d is the supposed target class ID 0 or 1.
- Each expert computes its output by:
  $v_k(t) = w_{ki}\, x_i(t)$;
  $y_k(t) = logsig\ (v_k(t))$;
- For the gating network, a softmax activation function is necessary:

$$g_k = \frac{\exp(u_k)}{\sum_{j=1}^{K}\exp(u_k)} \quad ; \quad k = 1, 2, ..., K$$

  where $u_k = a_k^T x$, k = 1, 2, ..., K.

- The *overall output* of the network combines gate and expert outputs,
  $y = \sum_{k=1}^{K} g_k y_k$; k = 1, 2, ..., K.
- According to the target value d, error e is defined as,
  $e = d - y$
- Posterior probability is defined as,

$$h_i = \frac{g_i y_i^d (1-y_i)^{1-d}}{\sum_{j=1} g_j y_j^d (1-y_j)^{1-d}}$$

- The weights w_{ki}, and a_{ik} are updated by the following rules:
  $w_i(t+1) = w_i(t) + \tau[w_i(t) - w_i(t-1)\ ] + \eta h_i(t) e_i(t) \boldsymbol{x_i}$
  $a_i(t+1) = a_i(t) + \tau[a_i(t) - a_i(t-1)] + \eta[h_i(t) - g_i(t)]\boldsymbol{x_i}$
  where η is the learning rate, τ is the momentum term constant.
- The updates on the weights stopped when certain stopping criteria is met.

B. MoE-II

1) Pre-processing module

a. Self-Organizing Feature Map

Kohonen's Self-Organizing Feature Map (SOM) algorithm is an unsupervised learning algorithm. The learning is detailed as follows:

- Assign random weight values for each node in the network, w_{ij} ;
- Upon presenting an input pattern $\boldsymbol{x}$, calculate the distance between $\boldsymbol{x}$ and each neuron j represented by the weight vector, $\boldsymbol{w}_j$, identifying the winning neuron as $\arg\min_j \left\{\left\| x - w_j \right\|\right\}$, where $\|.\|$ is the Euclidean norm.
- Adjust the weights of neighborhood of the winner neuron by:
  $w_{ij}(t+1) = w_{ij}(t) + \eta(t)K(j,t)\{x_i(t) - w_{ij}(t)\}$, where $\eta(t)$ is the learning rate at epoch t, and $K(j,t)$ is a suitable neighborhood function, in this case of a Gaussian nature;
- Repeat step 2-3 until convergence, in this case, when the absolute squared weight changes is smaller than 0.02 over 2500 epochs.

b. Clustering of the realized SOM

To partition the SOM into regions, we utilize a robust Potential Function clustering method proposed by Chiu [9], as follows:

- Take each SOM node w_j as potential cluster center, calculate its influence on all the other nodes w_i as

$$P_i = \sum_{i=1}^{N} \exp\left(-\alpha \left\| w_i - w_j \right\|^2\right),$$

where $\alpha = \frac{4}{\gamma_\alpha^2}$ and γ_α is a positive constant, a default value of 4 is used.

- Select the node with the highest potential as the first cluster center, $w_j(0)^*$ and its potential as $P_j(0)^*$, revise the potential of other nodes by subtracting w_j's potential as

$$P_i = P_i - P_j(0)^* \exp\left(-\beta \left\| w_i - w_j(0)^* \right\|^2\right) \text{ and}$$

$\beta = \frac{4}{\gamma_\beta^2}$; a good choice of γ_β is $1.5\gamma_\alpha$.

- Then we select the node with the highest potential as the second cluster center, revise the potentials of other nodes as in step 2.
- Repeat step 2 and 3, until some termination criteria met. In this case,

if $P_k^* > t_h\ P_j(0)^*$, accept $w_{.k}^*$ as cluster center and continue, (t_h is predefined upper cut threshold, $0<t_k<1$, here 0.5 is used);

else if $P_k^* < t_l\ P_j(0)^*$, reject the new cluster center and end the process, (t_l is predefined lower cut threshold, $0 < t_l < t_h$, here $t_l = 0.1$)

else let $d_{min} = \min\{\|w_{.k}^* - w_j(0)^*\|\}$, the minimum distance between current center to all the other previously determined centers.

if $d_{min}/\gamma_\alpha + P_k^* / P_j(0)^* \geq 1$, accept $w_{.k}^*$ as new cluster center and continue,

else set $P_k^* = 0$, reject $w_{.k}^*$, select the next highest potential as the new center and re-test.

As the result of step (b), SOM nodes are grouped into *q* partitions, each with a specific SOM node identified as the region centroid. That is to say, a natural consequence of the Potential Function method is that not only are partition centriods identified, but also nodes are assigned to regions on a nearest neighbour basis.

c. Calculate SOM weight factors

As indicated above, the centroid for each region is also forward to the gate in order to provide a "summary view" of the input space without having to forward a vector over the entire input dimension. To do so, the following normalization is employed,

- Calculate the inverse distance of node j in cluster i to its center c_i as

$$d'(w_{jci}, c_i) = \| w_{jci} - c_i \|^{-1}$$

- the weight factors are calculated as,

$$wf_{jc_i} = d'\left(w_{jc_i} - c_i\right) \Big/ \sum_{j=1 to |c_i|} d'\left(w_{jc_i} - c_i\right)$$

C. Changes to the MoE module

With the preprocessing module in the front end, we build *q* classifier experts, one for each SOM cluster. The number of SOM nodes in each partition is automatically determined in step (b) above, which is not necessarily equal for each partition. The gating network receives input from the SOM partition centroids, processed as per step (c) above. For any input pattern, ***x***, each SOM partition

- Forwards the Euclidean distance of each node to the corresponding expert network (no winner takes all rule) and,
- Forwards the value of the partition centroid to the gate network.

These activities are detailed as follows,

1. Present a pattern ***x*** to the network.
2. Calculate the distance between ***x*** and every SOM node, w_{jci}, c_i is the index for the cluster i, j is the node index within cluster c_i.
3. for expert *i*, the inputs are
$$d_{jci}(\boldsymbol{x}, w_{jci}) = \|\boldsymbol{x} - w_{jci}\|$$
4. for the gating network, its *q* inputs are the outputs from each cluster centers, which is calculated as,
$$gate_i(x) = \sum_{j=1}^{|c_i|} wf_{jc_i} \times d(x, w_{jc_i})$$

Once inputs for experts and gate are identified, operation of the MoE then follows that detailed for MoE-I as in section III.A.

IV. Experimental Results

A. Performance Measures

For both architectures, MoE-I and MoE-II, experiments are first conducted to identify best case parameter combinations. With these parameter settings, two thirds of the available data is randomly drawn and used for training, the remaining third is used as the test set. For each data set, 30 trials are made under different weight initializations, the mean and standard deviations of the error rates along with the parameter setting are reported in Table 1.

For MoE-I, the important parameters are the learning rate η, the momentum constant τ and the number of experts, |EXPS|, which largely establish the computation limits of the architecture. In the case of MoE-II, the parameters are the learning rate η, the momentum constant τ and the number of SOM nodes, |SOM|, and the number of the SOM partitions, |SOMC|. The latter parameters establish the computational limits of the architecture.

B. Data description.

Three benchmark data sets are taken to evaluate the architectures, each representative of a binary classification

task. They are C_HEART, IONO and BREAST and expressed as (attributes, patterns) pairs. C_HEART(13, 303), BREAST(9, 699), IONO(34, 351).

C. Results

Through all the experiments on different benchmark data sets and some man-made artificial data sets, we notice that keeping the learning rate low (0.1) is necessary for the computation to converge. A momentum term is also introduced such that, 1) the training error tends to accelerate descent in downhill direction on the error surface, 2), when downhill descent reaches a local or global minimum, the momentum term tends to have stabilizing effect; 3) it may also help to prevent the learning process from terminating in shallow local minimum on error surface [9]. The experiments here used values of 0.1 or 0.3.

From the results, it is apparent that MoE-II outperforms MoE-I for both C_HEART and IONO data sets, and is comparable with MoE-I for the BREAST data set. MoE-II has noticeably lower error rates on the C_HEART and IONO data, whereas error rates for the BREAST data are indistinguishable compared to that of MoE-I. It is noticeable that for all the data set, MoE-II has smaller standard deviation in its errors when compared to MoE-I.

TABLE I
TEST DATA ERROR OVER 30 RUNS

	CH	CH	BR	BR	IONO	IONO
	MoEI	MoEII	MoEI	MoEII	MoEI	MoEII
mean	0.232	0.204	0.037	0.042	0.107	0.058
std	0.039	0.029	0.014	0.006	0.034	0.033
η	0.01	0.01	0.01	0.01	0.01	0.01
τ	0.3	0.7	0.3	0.1	0.1	0.1
|EXP|	12		12		23	
|SOM|		72		90		72
|SOMC|		18		18		36
EP	400	400	400	300	200	400

Note: CH: C_HEART data set, BR: BREAST data set, IONO: IONO data set. EP: the epoch we stop training, and begin performance measure on test data.

During training sessions, MoE-I and MoE-II have drastically different behaviours. For MoE-I, the training curve starts with very deep descent during the early epochs, soon flattening out and remains almost the same until the end of the training session. The training error reaches low values at a rather early stage of training. For instance, for C_HEART data, under the best parameter combination, after 250 epochs, the training error nearly reaches zero. On the other hand, the behavior of test curve is rather random. It often reaches a low valley very quickly, but then continues to steeply climb. The starting point of the test curve is rather random, and the position when the valley occurs is also random. From our experiments, it is very difficult, if not impossible, to derive any heuristic rules to determine the position of low valley on test curve for better error rate values. It is fair to say, at least in the experiments and data sets reported here, that there is little correlation between the training curve and the test curve. This is well illustrated in Fig1. In contrast, for MoE-II, good correlation between the training and test curves is achieved. Both start with deep descent with similar speed and slope, and then follow a flattened long tail. The training curve remains reasonably low, near 0.1 for C_HEART, while the test curve remains nearly twice of that of training curve. The training curve and test curve of MoE-II show stable behavior in contrast to that of MoE-I. This is clearly illustrated in Fig2.

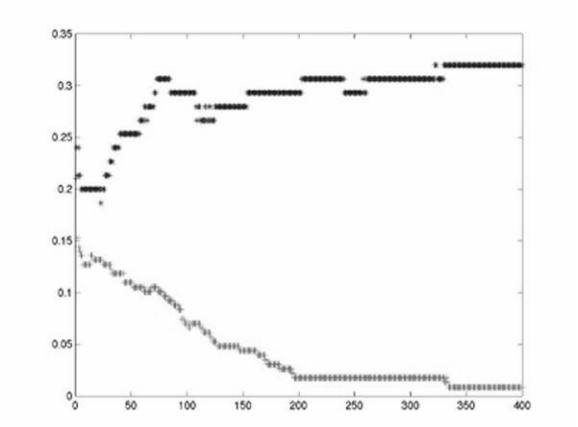

Fig 1. MoE-I for C_HEART, 19th run.
(upper curve is test error, lower curve is training error)

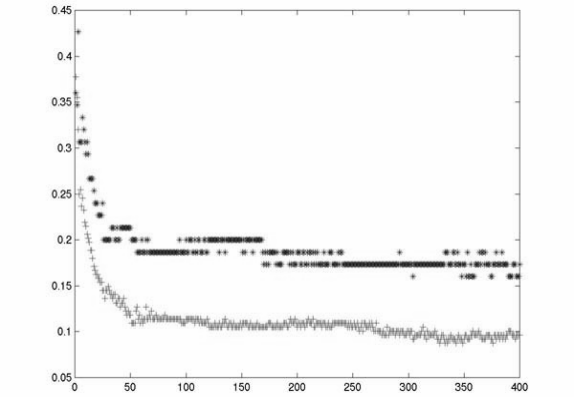

Fig 2. MoEII for C_HEART, 3rd run.
(upper curve is test error, lower curve is training error)

MoE-I alone therefore appears to be is a less robust architecture than MoE-II. Moreover, when searching for suitable combinations of learning parameters, it proved to be a rather harder task for MoE-I than for MoE-II. Little deviation from the "optimal" combination of training parameters would cause drastic changes in the final results. For MoE-II, a deviation from the "optimal" parameter setting would only cause minor degradation in the overall performance. This observation is also reflected in the sensitivity to MoE network nodes under both architectures. MoE-I becomes even less stable, resulting in further over fitting shown on the test curve. On the contrast, MoE-II is much more graceful it any deviation between test and training performance.

For the IONO data set, similar behavior patterns of the two architectures are observed. Under the best parameter settings, MoE-I shows strong randomness and less coupling between the training curve and the test curve. Fig 3 illustrates one common run. For MoE-II, the training curve and the test curve are strongly coupled, indicating that the learned network based on training data generalizes well on test data. One such example is shown in Fig 4.

Similarly, for BREAST data set, there is a tighter coupling between training curve and test curve with MoE-II, while there is more randomness with MoE-I. The trend is not as obvious as with the other two data sets, since both MoE-I and MoE-II reach indistinguishably low error rates.

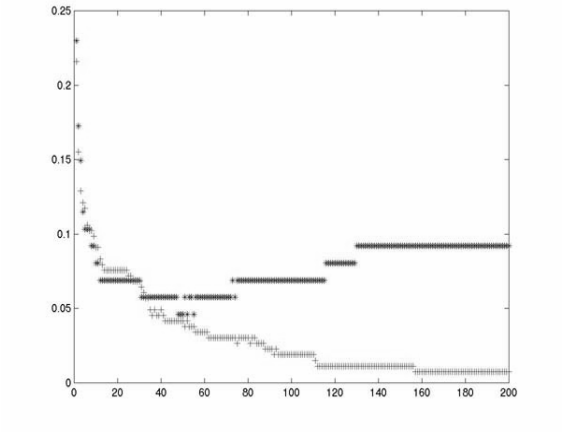

Fig 3. MoE-I for IONO, 24th run.
(upper curve is test error, lower curve is training error)

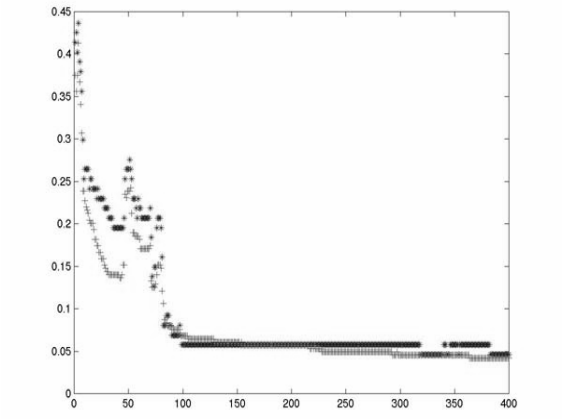

Fig 4. MoE-II for IONO, 30th run.
(upper curve is test error, lower curve is training error)

As indicated during the introduction, a major motivation of this work is to improve the transparency of the network following convergence. By scrutinizing the roles played by the experts in decision-making we are able to associate experts with classes for MoE-I and classes with partitioned inputs in the case of MoE-II. To do so, a count is made of the instances in which a gate value larger than some threshold value (25%, 60%) occurs. For different threshold values, the general trends are found to hold true. For instance, on MoE-II with CHEART data, experts have a clear role division in the classification task, Fig 5. Expert 5 is the decision maker for class 1 data with a minor influence from expert 7. Experts 8 and 13 mainly identify class 0 data, with a minor contribution from expert 9. This is an interesting result as it indicates that assigning local regions of the input space to each MoE expert has not resulted in complex expert combinations to solve the overall classification problem.

On the other hand, in MoE-I, Fig 6, the experts play complex roles in forming the decision function. For class 1, expert 1 is the major decision maker with minor help from expert 7. While, for class 0, the decision making task is distributed over many experts (expert 2, 3, 5, 7, 11). Also we notice that the discriminating ability of each expert is not as strong as that in MoE-II. For instance, for some of the decision makers of class 0 data, experts (5, 7, 11), they also participate in the identification for class 1 data in a non-negligible manner.

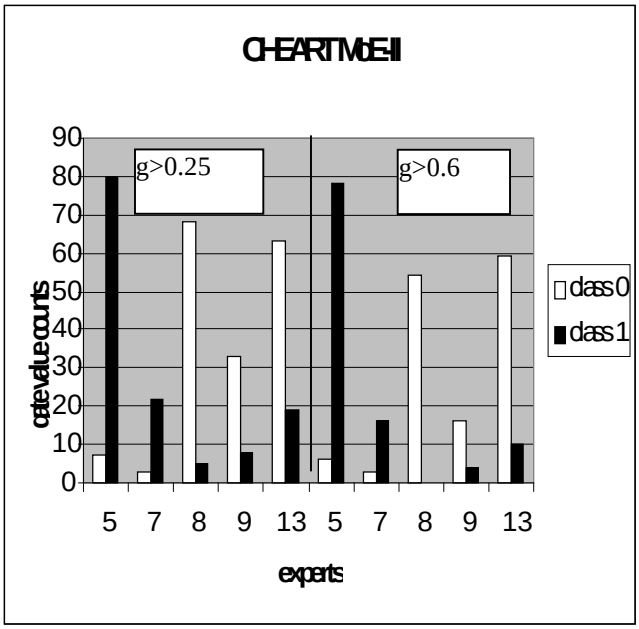

Fig 5. MoE-II gate plots for CHEART on run 3.
(unimportant experts, gate counts < 10 for both classes, are omitted)

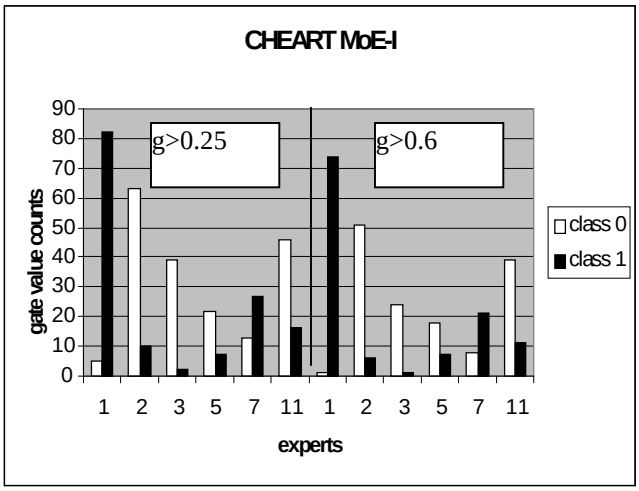

Fig 6. MoE-I gate plots for CHEART on run 19.
(unimportant experts, gate counts < 10 for both classes, are omitted)

The unclear division of roles that experts played in MoE-I compared to that in MoE-II is interesting. MoE-I resulted in the distribution of the decision function for all the data over all the experts on an instance-by-instance base (case of multiple experts contributing to both classes). On the other hand, in MoE-II, experts and gating network base their judgments on preprocessed, de-noised, abstracted features rather than the raw data. Since the experts are directly linked to specific (SOM) feature regions, there appears to be a better ability for identifying relationships between payoff and most applicable expert-input partition. This speculation is supported by the strong discriminating ability exhibited by the experts in MoE-II.

V. Conclusion

Experiments are conducted with the Mixture of Experts architecture, but under different pre-processing conditions, in this case designed to identify partitions regarding the association of inputs to experts in the MoE. Such a scheme provides: 1) better generalization ability, 2) robust and stable to parameter selection, and 3) makes the contribution between experts clearer.

Factors contributing to this conclusion are, 1) pre-processing module divides the input space into partitions according to the underlying probability distribution of the data, 2) partitioning determines the number of experts in MoE, (same as the number of regions in the SOM), 3) given any pattern, due to the preprocessing, each expert is selectively given stronger or weaker pre-processed signals, this leading to a clearer distinction between expert "responsibilities", 4) noise in the input space is reduced within the preprocessing module, therefore, each expert only handles the de-noised, preprocessed signals, 5) gating network still receives inputs representative of the 'global problem'.

This work can be extended in the following directions. Firstly, we are naturally interested in the use of hierarchical MoE architectures and the significance of hierarchical (SOM) partitioning for determining the MoE architecture. Secondly, extensions to the case of multi-class classification problems are to be verified. Thirdly, so far all the data sets we have deal with a relatively low dimension of input data, from 9 to 34. The eventual aim will be to test MoE-II on data with high dimensionally, for instance text classification problems, which is a multi-class classification problem on high dimensional data.

VI. Reference

[1] M. I. Jordan and R. A. Jacobs. 1994., Hierarchical Mixtures of Experts and the EM Algorithm. Neural Computation, vol 6, pp181-214.

[2] M. I. Jordan and R. A. Jacobs. 1995., Modular and Hierarchical Learning Systems. in M.A.Arbib, ed., The Handbook of Brain Theory and Neural Networks, pp579-53, Cambridge, MA:MIT Press.

[3] T. Kohonen . 1990., The Self-Organizing Map. Proceedings of the IEEE . Vol 78. No.9. p1464-1480.

[4] Swets D., Weng J.J., "Using discriminant eigenfeatures for image retrieval," IEEE Transactions on Pattern Analysis and Machine Intelligence. 18(8), pp 831-835, Aug. 1996.

[5] B. Fritzke 1993. Kohonen Feature Maps and Growing Cell Structures - a Performance Comparison. Advances in Neural Information Processing Systems 5. p123-170

[6] J. Vesanto and E. Alhoniemi. 2000. Clustering of the Self-Organizing Map. IEEE Transactions on Neural Networks. Vol 11. No.3 P 586-600.

[7] R. N. Devé and R. Krishnapuram . 1997. Robust Clustering Methods: A United View. IEEE Transactions on Fuzzy Systems. Vol 5. No. 2. P270-293.

[8] H. Demuth and M. Beale. Matlab -Neural Network Toolbox, Users Guide 4.0; http://www.mathworks.com

[9] S. L. Chiu. 1994. Fuzzy Model Identification Based on Cluster Estimation. Journal of Intelligent and Fuzzy Systems. Vol.2 p267-278.