Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Self-Selective Clustering of Training Data Using the Maximally-Receptive Classifier/Regression Bank

Ram Balasubramanian & M.A. El-Sharkawi
University of Washington
EE Dept.
Seattle, WA

R. J. Marks II
Baylor University
ECE Dept.
Waco, TX

Jae-Byung Jung
Scientific Fishery Systems
Anchorage, AK

R.T. Miyamoto, G.M. Andersen, & C.J. Eggen
Applied Physics Lab
Seattle, WA

W.L.J. Fox
BlueView Technologies
Seattle, WA

***Abstract*—A common approach to pattern classification problems is to train a bank of layered perceptrons or other classifiers by clustering the input training data and training each classifier with just the data from a specific cluster. There is no provision in such an approach, however, to assure the component layered perceptron is well suited to learn the training data cluster it is assigned. An alternate method of training, herein proposed, lets a layered perceptron in a classifier bank choose the cluster of inputs it processes on the basis of the perceptron's ability to successfully classify those inputs. During training, data is therefore processed only by the classifier in the bank that best classifies the data or, equivalently, to which the data is most receptive. This allows each classifier to learn a localized subset of data dictated by the classifier's own classification ability. Once each classifier in the bank is trained, a separate independent cluster pointer is trained to recognize to which cluster an input test pattern belongs. The cluster pointer is used in the test mode to identify which classifier in the bank will best classify the problem. The approach, also applicable to regression type problems, is illustrated through application on a simulated Gaussian data set and an active sonar test estimation problem. In both cases, the maximally receptive classifer/regression bank significantly outperforms a single layered perceptron trained on the same data,**

Keywords: ***Index Terms*—pattern classification, regression, clustering, classifier banks, domain expertise, active sonar classification.**

I. Introduction

Classification requirements of data sets can be too complex to be handled by a single classifier. A solution is to break the problem down into smaller pieces to distribute the problem to a bank of simpler classifiers. This type of general system is referred to as a multiple classifier system (MCS).

A common approach of a MCS classifier is to first divide the training data into sets by clustering. Each cluster of data is then used to train a classifier in a bank. In the test mode input data is first subjected to a clustering algorithm to determine in which cluster the data best resides. The corresponding component in the classifier bank is then used to perform the final classification. An alternate approach is to allow the bank of classifiers to perform self-clustering by letting only that classifier most receptive to a training pattern adapt. This maximally receptive classifier is updated using, for example error backpropagation, while the other classifiers remain unchanged. After repeated presentation of the training data, the process ideally converges and each training pattern is assigned to the classifier that produces minimum classification error. The training data assigned to the nth classifier thus belongs to the nth cluster. It remains to train a cluster pointer classifier on this data. Specifically, if an element to the nth cluster is input to the cluster pointer, the cluster pointer will indicate in some way, at its output, a pointer to the nth classifier.

II. Foundations

Much research is focused on cases where there is specific *a priori* information about the data leading directly to use in multiple classifiers. One example is where the distribution is known to be multi-modal. Consider, for example, a two class classification problem where data from both classes is concentrated in two far removed disjoint regions in the classification space. The overall classification problem is improved greatly by identifying to which of the two cluster data belongs and using two classifiers -one for each cluster. Another case for the use of a classifier bank occurs when different feature sets can be extracted from the raw data or when the feature scan be processed in different ways to generate pseudo-features [1], [2].

There is much interest in work on combining classifiers [3], [4]. This work, however, solves the problem of combining the outputs of multiple classifiers and assumes that disjoint set of trained classifiers is available. No concern is given to the coordinated training of the component classifiers.

A full MCS is one where a data set (with a single set of features) is fed to multiple classifiers and the output so the classifiers are somehow combined to produce a single output vector. The key elements to a MCS lie in how the subsets are chosen, in how the classifiers are trained, and is how the outputs are combined. The simplest solution is to pass the entire data set to randomly initialized classifier and to combine the outputs by am majority vote [5], [6]. This solution, though, does not present a clear advantage to a single classifier since each classifier is attempting to learn the entire space of data and therefore may reach approximately the same solution.

Another solution is to cluster the the data before classification and then pass each cluster of data to a different classifier [7], [8], [9], [10], [11], [12], [13]. The clustering can be done

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

using one of many well known clustering algorithms such as k- means clustering, ART, or a Kohonen *self organizing map* (SOM) [15]. This approach can be effective if the data successfully clusters and if the clusters give a less complex input-output mapping as a result of this clustering. There is no provision made, however, for assuring the component classifiers are well suited to handle the cluster assigned to it.

A modular neural network, or a *mixture of experts* (MoE) [14], [16], [17], uses single layer perceptrons as experts to classify data and a gating net work to combine the results of the experts. The gating network is also a single layer perceptron [18] (note that both the expert and gating networks are linear: there is only a single node in the hidden layer and there is no activation function.) This MCS uses an *aposteriori* probability function that is back propagated through both the experts (component classifiers) and the gating network (cluster pointer). By adjusting the weights of both the experts and the gating network simultaneously, the MoE allows the clusters to adapt. The experts are linear and the region learned by each expert is largely determined by the initialization of the gating networks. It is possible to build a more complex MoE system by hierarchically arranging several MoE's in a tree-like structure. Since the gating networks separate the data linearly, however, the spawning of branches from the tree does not guarantee better classification. That is, if a high order declassifier is needed, linearly separating the input space into several branches (partitions) may not lead to linear class separation within the partitions. Without reliable stopping criteria, it is possible to spawn too many branches and over fit certain regions of the input space which have good class separation, due to other regions which have worse separation. A high order classifier, on the other hand, is able to use varying degrees of complexity in different regions of input space.

III. Self-Selective Maximally-Receptive Classifier/Regression Bank

We propose a new method of training which allows multiple classifiers to learn desired targets of disjoint subset of data without imposition of clustering heuristics. This is done by training each classifier in a classifier bank on only the training data it classifies most accurately, *i.e.* on the data to which the classifier is most receptive. In general, any classifier can be used. We investigate specifically use of the feed - forward multi-layered perceptron neural network (NN) [18]. Before training, each NN in the classifier bank will have its own unique error surface, assuming the weights of each are randomly initialized. If each NN is trained on the entire data set the results of each after training should be roughly equivalent since the objective of error backpropagation is to minimize the overall error. Although each NN may in fact approach different local minima, the overall combination will not be significantly better than a single NN [5]. On the other hand, by training each NN on only the data it best classifies, each NN will become an expert at classifying a specific subset of data. Instead of attempting to reduce error throughout the entire space, the error is reduced in its self-assigned cluster locally rather than globally. The resulting maximally-receptive classifier/regression bank will be referred to as the MaRC bank.

A. Algorithm Description

Let the training set contain N patterns and be defined by inputs, $X = [X_1, X_2, \cdots, X_N]$, $X_i \in R^{n_F}$, and by targets (desired outputs), $T = [T_1, T_2, \cdots, T_N]$, $T_i \in R^{n_O}$ where n_F is the number of features and n_O is the number of outputs. The data will be classified by M NN's labeled $NN_1, NN_2, \cdots NN_M$. The output of the jth NN for the ith pattern, X_i, is Y_{ij}. Define a cluster pointer, π_j, associated with NN_j,whose output for the ith pattern, X_i, is γ_{ij}, and whose target (which is based on the error of NN_j) is τ_{ij}. The purpose of the cluster pointer is to identify the regions of input space for which each NN has minimum error. Implementation of the MaRC bank consists of the following 3 steps.

1) *Training the NNs.* To train the NNs using on-line training [18], feedforward a single training pattern, X_i, to each of the NNs. Find the mean square error (mse) of each NN, E_{ij},
$$E_{ij} = ||Y_{ij} - T_{ij}||^2.$$
where $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, M$. Only backpropagate the NN for which E_{ij} is minimum and do not backpropagate another NN. This forces each pattern to be learned by only the classifier which classifies it best. One epoch is complete when this procedure is repeated for all patterns in the training set. The training can also be batch-mode [18].
2) *Training the Cluster Pointers.* The cluster pointers (referred to as π networks), which are also classifiers (in this case, feedforward multilayer perceptrons [18]), are trained to learn some function of the NN error. A simple value for the cluster pointer to learn is an indicator function. For each input X_i set the desired output τ_{ij} of the jth cluster pointer as follows
$$\tau_{ik} = \begin{cases} 1 & , \text{ if the } k\text{th mse is smallest} \\ 0 & , \text{ otherwise} \end{cases} \quad (1)$$
The actual output of the jth cluster pointer is γ_{ij} and the mse of the network is $\epsilon_{ij} = ||\gamma_{ij} - \tau_{ij}||^2$. To train the cluster pointers, simply backpropagate this error. Once the cluster pointers are trained, they will indicate which NN classifies a given pattern best
3) *Testing.* The last stage in classification is to present the system with new data. In this stage we assume that the desired output is not known. To determine the output, y, of a given test pattern, x, pass it through all the cluster pointers and find the maximum γ. Then the output y will be the output of the NN corresponding to maximum γ
$$y = Y_k; \text{if the } k\text{th mse is smallest.} \quad (2)$$

B. Visualization

The objective of the MaRC bank is to reduce the over- all error surface by focusing each individual classifier on specific areas of input space. In training the NNs, this objective is

easy to meet since the errors are known, and each pattern is, by definition, classified by the most receptive classifier. However, the true test of how well the MaRC bank is working is the degree to which the cluster pointers learn this error surface. That is, the π networks must be able to recognize, for a given input pattern, the NN for which the error is minimum. In order to gauge how well the π networks are doing, the clusters chosen by the cluster pointers should be compared to the ideal separating surface, if available. The ideal separating surface is only available, of course, if the distribution of the data is known *a priori*.

For the special case where the input space is two- dimensional ($i.e.$, $n_F = 2$) and the ideal classifier is known, it is relatively straightforward to visualize the separating surface traced by the ideal classifier and compare it to the π networks. The tracings of course need not overlap exactly, as long as the cluster pointers are choosing a reasonable NN. For example, if an input pattern falls between two NNs at a point where the error surfaces intersect, both NNs will classify the pattern equally well. The main objective is that one of these two NNs is chosen by the cluster pointers.

In order to visualize the performance of the MaRC bank we can use a two dimensional Kohonen SOM. This is done using the training set to create a SOM and then mapping the MaRC bank onto it. An $n \times m$ SOM in two dimensions has $n \times m$ neurons, each representing the mean of some subset of the original training data. Each pattern in the training set is associated with a particular neuron. The SOM is constructed in such a way that neurons which are close in the SOM contain patterns which are close in the original space. Also, patterns within an individual neuron are close in a Euclidean distance sense. If a large enough SOM is created, every neuron in the SOM can be associated with a particular classifier if the patterns that belong to the neuron are all chosen by one classifier. A SOM with many neurons clusters the data to some degree, but its main function in this application is to reduce the dimensionality of the space in order to facilitate visualization. The region chosen by each classifier can be traced on the SOM and the cluster pointer outputs can be traced onto the SOM and compared with the regions chosen by the classifiers.

IV. Results

The MaRC bank has been tested on two classification problems. The first uses simulated Gaussian data. Seven Gaussian clouds clouds in 2 dimensional space are generated. The data describes 2 classes with fairly high overlap between the classes. This data set is used since the ideal classifier (a maximum likelihood classifier) is known and, as such, can be compared to the MaRC bank results. The MaRC bank is then used to classify a complex, high dimensional sonar approximation problem, giving excellent results.

A. Gaussian Problem

This data set contains 1600 patterns, with 2 features and 2 outputs, divided randomly into 800 training patterns and 800 test patterns. Class a is comprised of 4 Gaussian clouds (each with 200 patterns) with relatively large variance, while class b is comprised of 3 Gaussian clouds with smaller variance (one of the clouds has 400 patterns, the other two have 200 each). Table I shows the means and variances of the clouds (the features were uncorrelated). Note that the second column represent the mean given to the random number generator, while the actual normalized mean is given in column 3. Figure 1a shows a scatter plot of the training points. The original data is normalized to the region [-1, 1] in both dimensions for training purposes, and the plots reflect this normalization. The table gives the original means and variances used to generate the data, along with the actual normalized means. The data was purposely constructed to have fairly severe overlap between the two classes. Since the clouds are Gaussian and have different shapes ($i.e.$ different covariance matrices), the ideal maximum likelihood classifier separates each cluster from every other cluster by a quadratic curve. These quadratic curves can be calculated since the density functions of the clouds are known a priori. These quadratic separating surfaces can then be combined to form an overall class separation surface. Thus, the ideal classifier can be found. This ideal classifier can be traced and compared to the surfaces produced by a single NN and by the MaRC bank.

TABLE I
Means, variances (var), normalized (norm) mean and number of patterns for each Gaussian cloud, C

C		mean		norm mean	var		Count
a	1	0.3	0.4	0.31	0.26	0.3	0.4	200
a	2	0.3	0.3	0.60	-0.46	0.3	0.3	200
a	3	0.5	0.2	0.05	-0.08	0.5	0.2	200
a	4	0.4	0.2	0.30	-0.59	0.3	0.1	200
b	1	0.3	0.1	0.35	-0.59	0.3	0.1	200
b	2	0.3	0.1	0.33	-0.29	0.3	0.1	400
b	3	0.4	0.2	-0.32	0.06	0.4	0.2	200

This type of problem suits a MCS well. If 3 NNs are used, each needs at least 2 hidden nodes (equivalent to a quadratic discriminant) to trace the clouds. Therefore, the MoE, with 3 experts, does not give optimal results (note that a tree structure may improve the MoE's results). The MoE does separate clouds, but is unable to isolate the clouds due to lack of complexity. The MaRC bank, on the other hand, can easily adapt to handle more complex problems by simply increasing the number of nodes in either the NNs or the π networks. For this problem, the MaRC bank was implemented in two ways. The NNs were fixed to have 1 hidden node while the π networks were set to have 2 hidden nodes in the first case and 3 hidden nodes in the second. A single NN was also trained on this Gaussian data. The architecture of the single NN was chosen after experimenting with several different NNs. The best performance was obtained using two hidden layers, each with 3 hidden nodes. More complex NNs tend to overfit this data set, while simpler NNs give inferior error rates. Table II compares the error rates between the single NN, the MoE, the MaRC bank with 2 hidden nodes, and the MaRC bank with 3 hidden nodes. The results of the single NN are almost

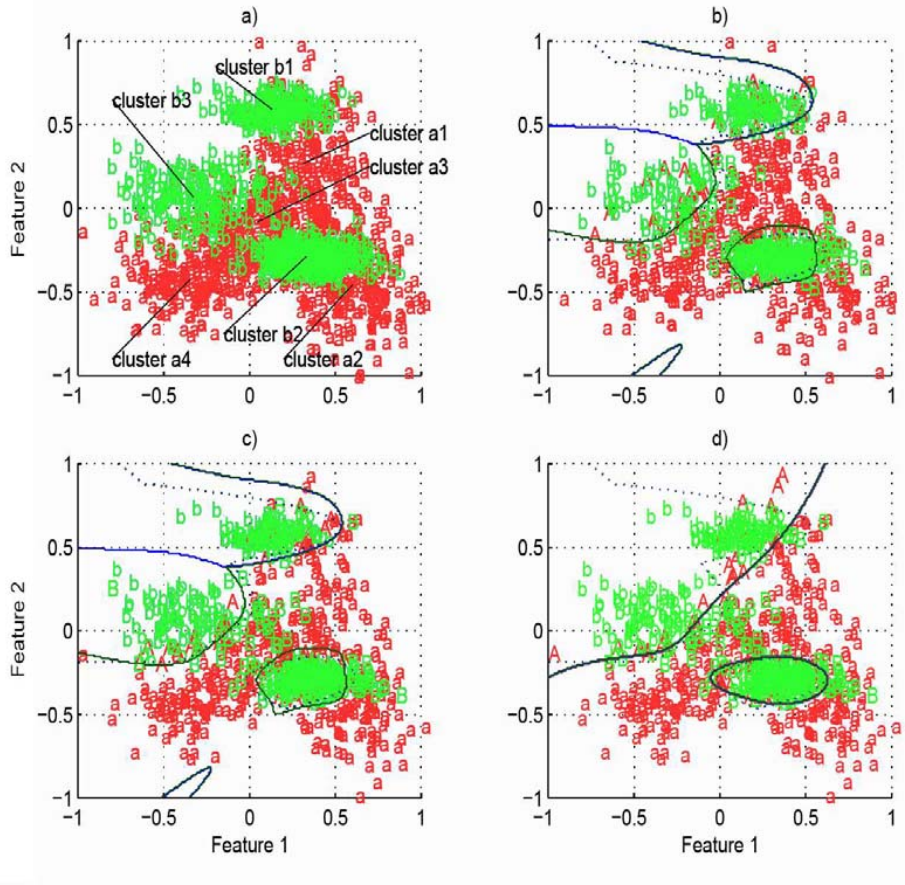

Fig. 1. a) A scatter plot of the Gaussian Data, normalized in both dimensions t o the range [-1,1]. The two classes are represented by the letters 'a' and 'b'; b) The training results with the π network trace (dotted line) sumperimposed on the ideal separating surface (solid line); c) The test results for the MaRC bank, lowercase letters 'a' and 'b' indicate correct classifications, while uppercase letters 'A' and 'B' indicate misclassifications; d) The test results for a single NN, dotted line is NN's separating surface.

as good as the MaRC bank, but are slightly worse. Note that the complexity of the single NN is very similar to the MaRC bank with 3 hidden nodes. Another interesting observation is that the MaRC bank is typically able to converge to its results within 1000 epochs, while the single NN took on the order of 30,000 epochs to converge. A visual interpretation of the error rates of the MaRC bank with 3 hidden nodes can be found in Figure 1b and 1c. Figure 1b shows the mapping of the π networks. When correctly classified, the two classes are noted by the letters 'a' and 'b', while incorrect classifications are noted by the capital letters, 'A' and 'B'. The dotted lines in the plot represent the ideal separating surface, generated from the means and variances of the clusters. The solid lines is the trace of the cluster pointers. The cluster pointer selects NN_1 for data in the upper left corer. NN_2 is selected for data in the middle left section, for feature 1 less than 0, feature 2 between approximately -0.1 and 0.5. The circle around cluster b2 also indicates a region for which NN_2 is chosen by the cluster pointers. The cluster pointers choose NN_3 for the remaining space (containing mostly patterns from class a).

TABLE II
ERROR RATES FOR THE GAUSSIAN PROBLEM.

Classifier	Training	Testing
Single NN	14.12%	15.25%
MoE	18.30%	22.25%
MaRC bank (2 nodes)	19.75%	21.00%
MaRC bank (3 nodes)	13.75%	15.00%

Figure 1c shows the test results, using the π networks to trace the space again. Compare these test results with that of the single NN, in Figure 1d. Note that the separating surface of the single NN is unable to trace the ideal separating surface as closely as the MaRC bank. This results in some misclassification in the region (0.1, 0.3), where several patterns from class a are misclassified. The MaRC bank, on the other hand is able to correctly classify these patterns.

B. Sonar Data

The goal of the sonar approximation problem is to generate a map of the *signal to interference ratios* (SIRs) for an active sonar on a grid of hypothesized target locations. A data set was generated using a standard underwater acoustic modeling tool based on a ray tracing approximation [19]. For a given hypothesized target location, the signal level is calculated from the dominant rays connecting the target and sonar locations. The interference level is the sum of ambient noise and reverberation (backscatter from interfaces and volume inhomogeneities) levels at the time of the sonar return corresponding to the two-way travel time of the rays used for the signal level calculation.

A single SIR 'map' represents a vertical slice of the ocean, and consists of 13 pixels in depth and 30 pixels in range, with each pixel representing the SIR for a hypothesized target location. A total of 4000 SIR maps were generated by randomly varying five input parameters (sonar and environment descriptors). Half of these maps were used for training, and the other half were used for testing.

Figure 2 shows a typical pixel map. The 30 range pixels represent approximately 6 km, while the 13 depth pixels represent 200 m. A surface ship located at the top left of the pixel map deploys an active monostatic sonar at a specified depth, transmits an acoustic pulse, and processes the resulting signal received on its hydrophones. The SIR map is used to identify regions in range and depth where potential targets may or may not be detected. In general, high SIR corresponds to higher probability of detection, while low SIR corresponds to lower probability of detection. The nature of acoustic propagation in the ocean leads to non-uniform coverage in range and depth. The objective of the neural net approximation is to accurately and quickly predict regions of good/poor detection capability.

The biggest challenge with this data set is the high dimensionality (390) of the output space coupled with sparse training data. Also, the complexity of acoustic propagation in the ocean can cause one or more 'hot spots' to appear in the pixel map. These hot spots are areas of high SIR which appear in isolated regions of the ocean. The problem is that hot spots are not well represented in the data. A hot spot may appear at a certain region for a given set of inputs, but a slight change in one of the input parameters may cause the hot spot to move to a completely different region, or to disappear altogether. As a result, the training set may not contain a smooth input to output mapping, and thus hot spots tend to become blurred (or completely ignored) when a single NN is trained on the data. Aside from the problem of hot spots, a single neural network is unable to learn the space without memorizing the training patterns just because of the complexity of acoustic propagation.

Referring to the sample SIR map in Figure 2, several features can be identified. The areas of high SIR (hot spots) are in the bottom left (the location of the sonar) and in 3 diagonal stripes, all beginning near the ocean surface at ranges 5, 14, and 21. Also, there is a 'shadow zone' (area with low SIR) located in the top right. The well trained classifier should pick out the 3 diagonal hot spots as well as the shadow zone.

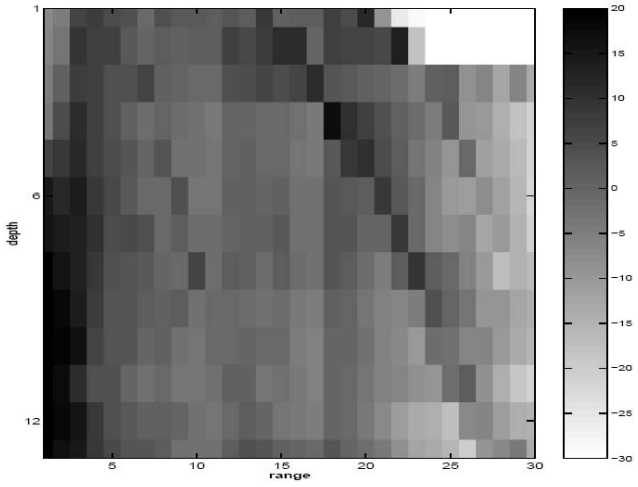

Fig. 2. Pixel map of a single SIR output. See text for detailed description.

The MaRC bank was constructed with five NNs, each with three hidden layers. The number of nodes in each NN is[1] $[5-12-12-30-390]$.The π networks each have one hidden layer with 20 nodes. Visualization of the clusters in 390-space is only possible by looking at individual pixel maps. However, by mapping the outputs onto a SOM we can trace the clusters formed by the NNs. This allows us to see whether the clusters contain patterns which are close to one another in Kohonen space. Also, by mapping the cluster pointers onto the same SOM, we can evaluate how well the cluster pointers are learning the errors of the NNs. Figure 3 shows the results of mapping the sonar output onto a 10×10 SOM. Each color in both a) and b) can be mapped to a cluster. Figure 3a shows the division of the training data as chosen by the NNs. This represents the target for the cluster pointers. Figure 3b shows the actual trace of the cluster pointers. Note that the cluster pointers choose the correct NN for all but 10 of the neurons. And that the missed neurons occur mostly in transition regions.

The MaRC bank was directly compared to both a single NN and to another MCS, which clustered the data with an ART neural network [15] and then classified each cluster with a NN (referred to herein as the pre-clustering technique). The architecture of the single NN was chosen by trying several different NNs and choosing the best, a [5-12-12-30] NN. Smaller NNs completely miss certain patterns, while bigger NNs memorize the training data and do not outperform the [5-12-12-30] network. The NNs in the pre-clustering technique are of the same complexity [12-12-30] as the NNs in the MaRC bank. The performance of the three classifiers is compared in two ways. First, the overall mse is found and compared, and second, the actual SIR maps from individual patterns are plotted and compared. Both are performance measure are effectively the same from the classifier's point of

[1]Meaning 5 inputs, 12 neurons in the first and second hidden layer and 30 on the next and 390 output nodes.

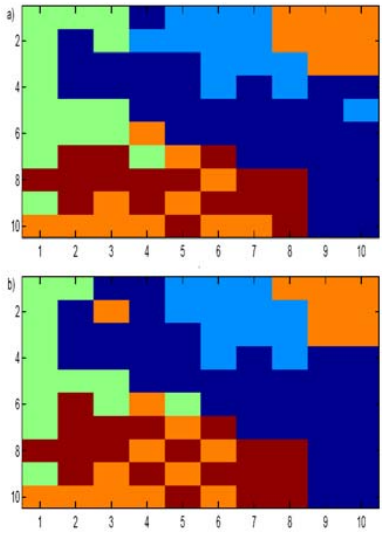

Fig. 3. Mapping the MaRC bank onto a SOM. The colours navy blue, sky blue, green, orange, and brown indicate that the corresponding neuron belongs to NN_1, NN_2,NN_3,NN_4 and NN_5, respectively. In a) the color indicates that the corresponding NN has minimum error (desired network selection). b) shows the choice of the cluster pointers. Note that the selection of the cluster pointers match very closely to the NN selection.

TABLE III
MEAN SQUARE ERRORS FOR SONAR DATA

Classifier	Training	Testing
Single NN	0.0333	0.0402
per-clustering	0.0187	0.0328
MaRC bank	10.0158	0.0293

view, but the high dimensionality of the output causes there to be a distortion between them. The human eye perceives a good pixel map that may actually have high mse because of a shift of one or two 'very bad' pixels. As a result, the mse is useful as an overall performance indicator, but observing the individual SIR maps manually can be much more informative. Table III shows the mse results of the three classification techniques for the same training and testing sets. Notice that the MaRC bank has lower mse for both training and testing.

Finally, Figure 4 shows the actual pixel maps for eight patterns. The first four patterns belong to a cluster (as chosen by the MaRC bank) containing patterns with one or two downward bending rays. The last four patterns belong to a cluster whose patterns have a downward bending ray followed by an upward bending ray. Pre-clustering the patterns gives better results than a single NN, but the hot spots are still not as crisp as they are in the MaRC bank. Take, for example, the SIR maps in columns 2, 3, and 4. In each case, there are two distinct diagonal hot spots shown by the target and in the second and fourth column there is a horizontal hot spot. The MaRC bank identifies these hot spots (albeit somewhat less distinctly than the model), whereas the pre-clustering and single NN both blur the hot spots together. This slight difference does not give significantly worse mse's, but the difference in SIR maps can be very important to the ship's captain. The bottom four SIR maps also show hot spots are more accurately represented by the MaRC bank. Although the pre-clustering captures the hot spots fairly well, it still blurs the edges more than the MaRC bank. data before classifying.

- *Learning the error surface.* As presented, the cluster pointers are only identifying the classifier with minimum error (i.e. the maximally-receptive classifier) for a given pattern. By training the cluster pointers to learn the actual error of each classifier, the MaRC bank will be able to more accurately assign patterns. It will also allow areas of high error to be identified.
- *Aggregating the classifier's decision.* In the generic MaRC bank, the output of the most receptive classifier is used as the output of the system. Alternatively, the out- puts of all the classifiers can be combined through, for example, a simple weighted average, or a more sophisticated technique, such as fuzzy integration [20], [21].
- *Invertibility.* Another useful feature of the MaRC bank is that it is a fully invertible system. In fact, if the component classifiers are invertible, the MaRC bank can be inverted internally. That is, there is no need to run an external optimization loop to invert the system. An internally invertible system can be inverted very quickly and efficiently and is needed for applications which specify a de- sired output and require a corresponding input. To invert the MaRC bank, first invert the output through every classifier. Then feed-forward the input of each classifier to its respective cluster pointer. Choose the input corresponding to the maximum cluster pointer output [22].

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under grant ECCS 0801440, the Office of Naval Research, under Grant No. N00014-96-1-0246, Budget No. 61-6936, and the Boeing Airplane Company. Technical suggestions by Professors James Luby at the University of Washington Applied Physics Laboratory and Michael Healy at the Boeing Airplane Company are gratefully acknowledged.

REFERENCES

[1] O.K. Ersoy and D. Hong, "Parallel, self-organizing,hierarchical neural networks," IEEE Transactions on Neural Networks, vol 1,no.2,pp.167-178, June 1990
[2] J.A. Benediktsson, R.Svinsson, O.K. Ersoy, and P.H. Swain "Parallel consensual neural networks," IEEE Transactions on Neural Networks, vol.8, no.1, pp.54-64, January1997
[3] J. Kittler, M. Heatef, R.P.W. Dulin, and J. Matas, "On combining classifiers," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.20, no.3, pp.226-238, Mar.1998
[4] G.S. Ng and H. Singh, "Democracy in pattern classifications Combinations of votes from various pattern classifiers," Artificia Intelligence in Engineering, vol.12, pp. 189-204, 1998
[5] L.K. Hansen and P. Salamon "Neural networks ensembles," IEEE Transactions on Pattern Analysis and Machine Intelligence, no.10, pp.993-1001, Oct.1990
[6] R. Maclin and J.W. Shavlik, "Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks," Int'l Joint Conf. on Artificial Intelligence, 1995 pp.524-530

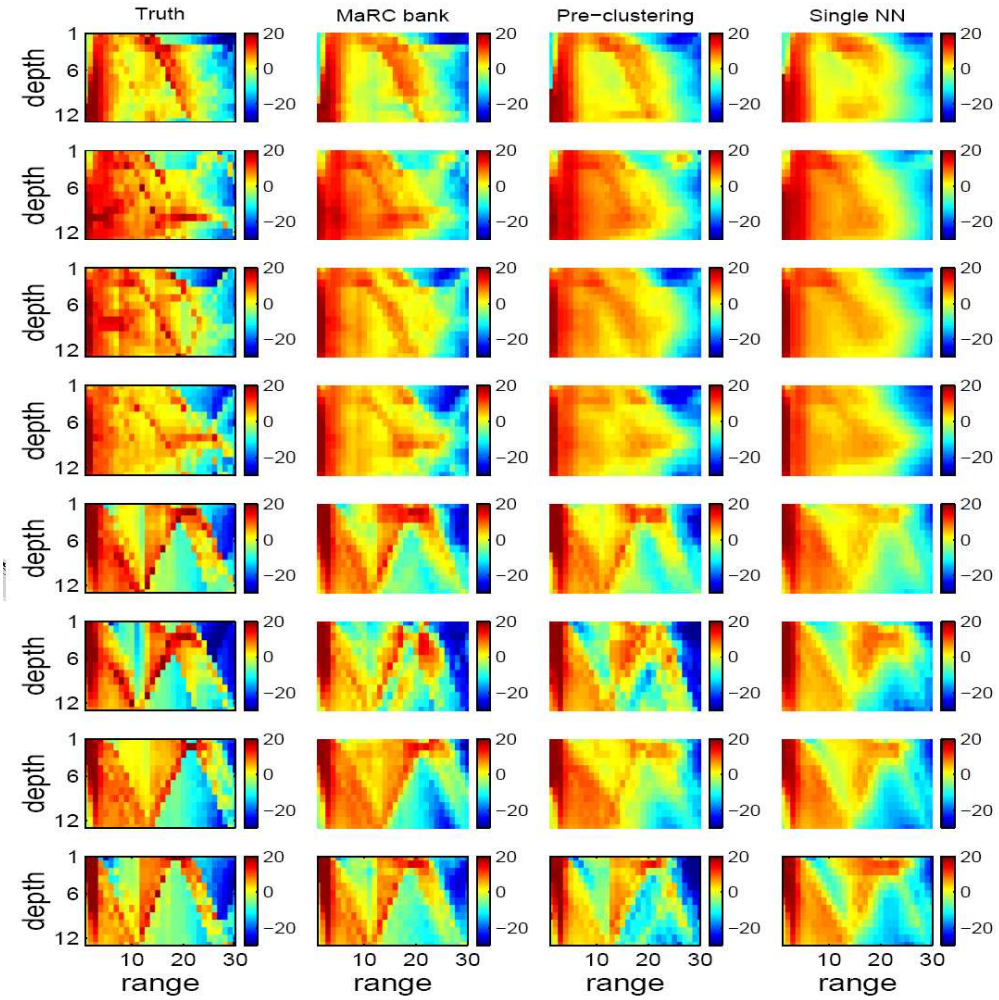

Fig. 4. Test Results for Sonar Data; First four rows are from cluster 2, last four rows are from cluster 3.

[7] A. Kehagias and V. Petridis "Predictive modular neural networks for time series classification," Neural Networks, vol.10, no.1,pp.31-49,1997

[8] J. Buhmann and H. Kuhnel, "Unsupervised and supervised at a clustering with competitive neural networks," International Joint Conference on Neural Networks, vol.4, pp.796-801, 1992

[9] P. Loonis, E.H. Zahzah, and J.P. Bonnefoy "Multi-classifier neural network fusion versus Dempster-Shafer's orthogonal rule," IEEE Conference on Neural Networks, Perth, Australia, 1995 pp.2166-2165 AUGUST,1995

[10] N. Chowdhury, C.A. Murthy, and S.K. Pal "Cluster detection using neural networks," IEEE Conference on Neural Networks, Perth, Australia, 1995,pp.2166-2170

[11] C.D. Wann and S.C.A. Thomopoulos, "Clustering with self organizing neural networks," Proceeding of the World Congress on Neural Networks, 1993, vol.2, pp.545-548

[12] H. Raafat and M.A.A. Rashwan, "At reestructured neural network," IEEE International Conference on Document Analysis and Recognition, Oct.1993, pp.939-942

[13] Y.Bennani, "A modular and hybrid connectionist system for speaker identification," Neural Computation,vol.7,pp.791798,1995

[14] R.A.Jacobs, M.I.Jordan, S.J.Nowlan, G.E.Hinton "Adaptive mixtures of local experts," Neural Computation, vol.3, pp.79-87, 1991

[15] Patrick K. Simpson, **Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations**, Pergamon (1990).

[16] R.A. Jacobs and M.I.Jordan, "Hierarchical mixtures of expert and the em algorithm," Neural Computation, vol.6, p.181,1994

[17] E. Alpaydin and M.I. Jordan "Local linear perceptrons for classification," IEEE Transactions on Neural Networks, vol.7, no.3, pp.788-792,May 1996

[18] Russell D. Reed and Robert J. Marks II, **Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks**, MIT Press (1999).

[19] F.B. Jensen, W.A. Kuperman, M.B. Porter, and H.Schmidt, "Computational Ocean Acoustics", American Institute of Physics, NewYork, 1994

[20] Y. Lu and F. Yamaoka "Fuzzy integration of classification results," Pattern Recognition, vol.30, no.11, pp.1877-1891, 1997

[21] J.M. Keller, P. Gader, H. Tahani, J. Chiang, and M. Mohamed, "Advances in fuzzy integration for pattern recognition," Fuzzy Sets and Systems, pp.273-283, Mar.1994

[22] C.A. Jenson, R.D. Reed, R.J. Marks II, M.A. El-Sharkawi, J.B. Jung, R.T. Miyamoto, G.M. Anderson, and C.J. Eggen "Inversion of feedforward neural networks: algorithms and applications," Proceeding of the IEEE, vol.87, no.9, pp.1536-1549 Sept.1999