MODEL-CATALOG COMPRESSION FOR RADAR TARGET RECOGNITION

Batuhan Uluğ and Stanley C. Ahalt

Department of Electrical Engineering
The Ohio State University
Columbus, OH 43210, USA

Richard A. Mitchell

Wright Laboratories, AARA
Dayton, OH 45433, USA

ABSTRACT

In many model-based Automatic Target Recognition (ATR) systems the size of the model catalog can be a critical factor in determining the viability of the system. In this paper we examine an ATR system which uses synthetic High Range Resolution (HRR) Radar data to determine how classification performance is affected by the compression of the HRR model catalog. For this purpose the data is preprocessed, clustered and classified using Nearest Neighbor and Radial Basis Function (RBF) classifiers. The effect of compression on classification performance is examined through simulations for both of these classification schemes. For the data in question we show that significant (100:1 or greater) compression can be achieved with little degradation in classification performance.

1. INTRODUCTION

Modern ATR systems typically have to cope with challenging scenario variations, such as target orientation, atmospheric conditions, and occlusion by terrain or clutter. Furthermore, many ATR systems are being designed to employ high dimensional data which became available with the advent of high-resolution sensors. Both of these factors result in the use of large target databases which stress storage and computational capabilities. This is true especially in mobile ATR applications where real-time operation is required with only limited computational resources. Therefore, a compression of the target database without severe degradation of recognition performance is highly desirable. In this paper we report the results of our studies related to one ATR database consisting of synthetic HRR radar data generated by the *XPatch* software package.

2. DATA DESCRIPTION AND PREPROCESSING

The HRR database consists of signatures of four different targets (the exact types of aircraft were unknown, but the targets were labeled) with the following characteristics :

- Azimuth angle : -25 to 25 degrees in 1 degree steps (51 settings)
- Elevation angle : -20 to 0 degrees in 1 degree steps (21 settings)
- 2 signatures at each azimuth/elevation setting corresponding to in-phase and quadrature components of the *vv*-polarization
- 1024 frequency samples per signature
- $51 \times 21 \times 2 \times 1024 \times 4 \approx 8.8$ Mb per target

To extract the relevant portion of the target signature, the radar data is preprocessed as follows :

1. The complex response is formed by combining the in-phase and quadrature components.
2. A Kaiser window is applied to the middle portion of 256 points of the data
3. An inverse FFT is performed on this 256 point data, and the resulting signal is shifted to center the range profile. The shifting operation is done in such a way that insures that the backscatter energy is centered in the range profile.
4. The absolute value (magnitude) of the resulting signal is calculated.
5. The middle 64 points of the range profile are extracted, as most of the significant energy is now contained in this region of the range profile.

While additional feature-extraction processing could be employed, we note that this simple feature extraction procedure reduces the data to the essential scattering features, and reduces the size of the signature by a factor of 32. The preprocessing also reduces the bandwidth and the resolution of the synthetic data to more realistic ATR levels. While the results reported here used only this simple procedure, we are currently considering additional signature feature extraction. For example, we could use parametric estimation techniques to further reduce the data to a set of parameters which describe the signature as a collection of complex exponentials. In addition, we have preliminary results which indicate that there is some benefit to applying logarithmic equalization to the preprocessed data.

3. CLUSTERING

Clustering the synthetic radar data constitutes the first step in the classification system we considered. The underlying assumption is that the codewords resulting from clustering can model the variability in target response as the look

This research was supported in part by the Advanced Research Projects Administration under Grant MDA972-93-1-0015, and by Wright Laboratories.

0-7803-2431-5/95 $4.00

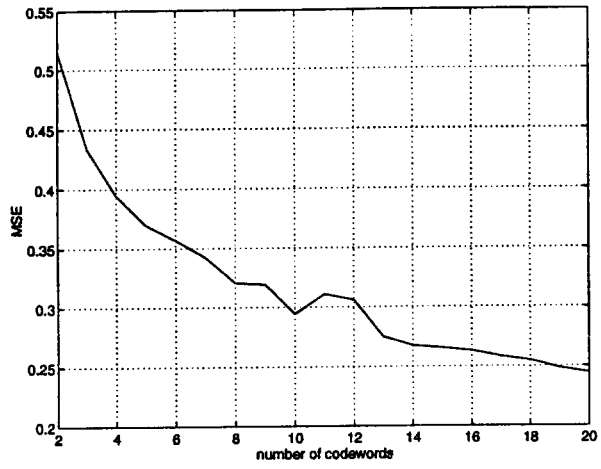

Figure 1: MSE index for k-means codewords of target 1

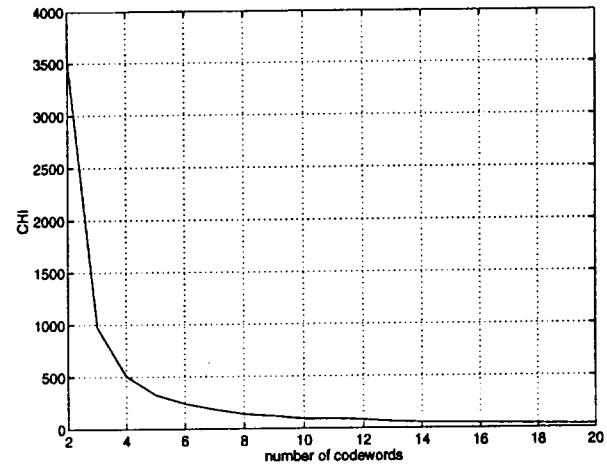

Figure 2: CHI index for k-means codewords of target 1

angle changes and thus, can be used to efficiently classify targets.

Clustering of the data can be done using the popular k-means algorithm [1, p. 24]. In this algorithm the cluster centers are initialized to randomly selected training data vectors. Clusters are then formed around these centers based on a minimum (typically Euclidean) distance criterion. Cluster centers are then replaced by the cluster sample means and the process is repeated until no further change in cluster assignments is observed from one iteration to the next. This algorithm has been found to work very well on our high dimensional data set.

In order to examine the validity of the resulting codewords, and as a means of estimating the number of codewords needed to represent the data well, we use two indices of partitional validity (IPV) [2], the mean square error (MSE) and the Calinski-Harabasz Index (CHI).

In a broad sense, the IPV's are used to measure the ability of the codewords (clusters) to represent the structure in the data. The mean square error is the simplest and most widely used of all the IPV's. It is given by :

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \|x_i - m_{L(x_i)}\|^2 \qquad (1)$$

where $\|.\|$ denotes the Euclidean norm, x_i is the i^{th} data vector, $m_{L(x_i)}$ is the codeword nearest the data vector x_i, and N is the number of data vectors. MSE measures the spherical compactness of the clusters.

The CHI can also be employed to determine cluster validity. This index measures the compactness of clusters divided by the isolation of clusters. It is given by Eq. 2,

$$CHI = \left(\frac{N-L}{L-1}\right)\frac{\sum_{i=1}^{N}\|x_i - m_{L(x_i)}\|^2}{\sum_{k=1}^{L} N_k\|m_k - m_0\|^2} \qquad (2)$$

where x_i is the i^{th} data vector, $m_{L(x_i)}$ is the codeword nearest to x_i, m_k is the k^{th} codeword vector, m_0 is the mean of all data vectors, N_k is the number of data vectors in the k^{th} cluster, N is the number of all data vectors, and L is the number of codewords.

Typical MSE and CHI vs. number of codewords (clusters) plots are shown in Figures 1 and 2. As expected both IPV's decrease as the number of codewords increases. Moreover, it appears that approximately 8 to 12 clusters should be adequate to represent the 1071 raw signatures in this aspect window since on the average both the MSE and the CHI tend to level off at about this number of codewords. Thus, these IPV's indicate that a significant reduction in the number of stored signatures is possible, but the critical test is to determine the effect of catalog compression on classification performance, an issue not definitively addressed in previous work by other authors [3].

4. CLASSIFICATION

For this study we examine the performance of two different classification techniques that naturally lend themselves to the utilization of cluster codewords.

4.1. Nearest Neighbor Classification

Nearest neighbor classification (NNR) is a widely used method which is a special case of the voting k-Nearest Neighbors (k-NN) algorithm [4, Ch. 7], with $k = 1$. In NNR classification an input data vector is assigned to the same class as that of the training data vector that it is closest to in some metric. We perform classification simulations on the HRR data using the NNR algorithm with the Euclidean distance as the metric. The probability of error of multiclass NNR classifiers is known to be less than twice that of a Bayesian classifier asymptotically, i.e. as the number of training data vectors approaches infinity. Hence, this experiment provides us with a measure that classification performances with compressed (clustered) training data sets can be compared against.

The clustering process can also be used in conjunction with the NNR classifier to directly divide the data vector space into decision regions. This technique, which we will refer to as NNRC is basically the same as the NNR algorithm except the training data set is replaced by a much smaller set of codewords from the clustering stage.

4.2. RBF Classification

Clustering results can also be used by more sophisticated classification algorithms such as the RBF network [5] instead of NNR. RBF networks can combine overlapping localized regions generated by simple kernel functions (typically Gaussian functions) to create complex decision regions. Bayesian *a posteriori* class probabilities can then be estimated when desired network outputs are 1 of L (one output unity, all others zero, L being the number of classes) and a squared error criterion is used [6]. The Bayesian *a posteriori* class probabilities can then be used to perform classification. The advantages of the RBF classifier are as follows :

- they are capable of forming arbitrarily complex decision boundaries
- they can approximate Bayesian classifiers when underlying class distributions are Gaussian mixtures
- they are better suited for certain classification problems than traditional Multi Layer Perceptron (MLP) neural networks with sigmoidal activation functions [1].
- they can be incrementally modified to accomodate new target scenarios
- they permit real-time implementation

The RBF neural network architecture consists of one hidden layer with nodes evaluating the kernel function on the input and one output layer, which simply forms a linear combination of the outputs of the first layer as given in eq. 3

$$f(x) = \sum_{k=1}^{M} a_k g(\|x - m_k\|) \tag{3}$$

where x is a real valued input vector, $g(.)$ is the kernel function, m_k and a_k are the center (mean), and output weight of the k^{th} node, respectively and M is the number of nodes. The kernel function $g(.)$ is a radially symmetric function with a single maximum at the origin and which drops off to zero as the distance from the origin increases. In this paper, we have employed Gaussian kernel functions of the form :

$$g(x) = e^{-\|x\|^2}. \tag{4}$$

The training of RBF networks involves three different problems :

1. Finding the number of nodes needed in the hidden layer
2. Finding the parameters ("means", "widths", etc.) associated with each hidden-layer node
3. Finding the weights to the output layer.

Training can be done in a completely supervised manner where an error measure such as the total squared error on the training set is defined at the output. In this case, training becomes a computationally intensive non-linear optimization problem which may yield high precision results, but is plagued by slow convergence and difficulties stemming from local minima. On the other hand, hybrid RBF training methods combining unsupervised and supervised learning are computationally more efficient and have also been shown to learn faster than MLP networks of comparable sizes using backpropagation [5]. Hence, we chose to employ the hybrid method for the training of the RBF classifier.

In hybrid RBF training one typically uses unsupervised learning for the first and second training stages and supervised learning for the third stage.

To find the necessary number of nodes one may resort to hierarchical clustering algorithms which heuristically determine the number of clusters present [7] or one can use a clustering method with a preassigned number of clusters for various numbers of clusters and then make a choice based on certain indications in IPV plots such as "knees" or "flat plateaus", depending on the properties of the IPV being used.

There are various approaches to the second problem of finding the parameters associated with each RBF node. In the case of Gaussian kernel functions in eq. 3, the parameters needed are the means, m_k associated with the Gaussian functions. The means can be taken to be the cluster means provided by a clustering algorithm. One may try to improve on these means by post-processing, but often this does not result in significant performance improvements.

The third problem of finding the weights to the output layer can be solved by a variety of well established approaches. For example, one can use the pseudo inverse [8] to find a least mean square solution to Eq. 5,

$$d = Fa \tag{5}$$

where F is the node matrix containing the outputs of all nodes to all data vectors in the training set, d is the desired output vector, and a is the output weight vector. However, if the database is large, or if one wants to use an on-line method as opposed to a batch approach, gradient-descent algorithms such as LMS [9], can also be employed.

For the results presented here the RBF network with Gaussian kernel functions was trained as follows :

- k-means clustering codewords were used as estimates of RBF centers (means)
- the output weights were calculated using the pseudo-inverse method.

Note that we are not directly interested in finding the number of nodes needed in the hidden layer, since our aim in this paper is to examine the performance of the RBF classifier as the compression factor (i.e. the number of codewords/nodes) is varied. However, as the results of the next section suggest, IPVs can be used to determine the number of required nodes.

5. SIMULATIONS AND RESULTS

Our underlying premise is that the codewords resulting from the clustering stage can be used efficiently to classify targets. To validate this assumption we perform a set of simulations. This experiments will also allow us to compare the classification performances of the NNR, NNRC and RBF classifiers.

For the classification simulations conducted we divided the data into two sets, allocating $\frac{8}{9}^{th}$ to training and $\frac{1}{9}^{th}$ to testing. This is done by randomly sampling in the azimuth/elevation angle space. Twenty Monte Carlo simulations were performed in this way. Estimates of the probability of error and the confusion matrix were calculated as sample averages along with the sample standard deviation of the probability of error estimate.

As expected the NNR classifier has the lowest probability of error, $P_e = 0.0064$. The resulting classification performances for the NNRC classifier with data vectors *arbitrarily* selected from the training data set as codewords (NNRC-A), the NNRC classifier with k-means supplied codewords (NNRC-K), and the RBF classifier are depicted in Fig. 3. This figure shows how the probability of error for the three classifiers varies as the database is compressed by factors from about 50 to 500. The error bars around estimated P_e values in Fig. 3 mark one (estimated) standard deviation above and below the estimated P_e values.

We see that the RBF classifier outperforms the NNRC-K classifier, which outperforms the NNRC-A classifier. The performance improvement with RBF over NNRC-K is more pronounced at higher compression factors because the RBF classifier makes better use of the codewords. As the number of codewords used approaches the compression levels (i.e. about 10 codewords, compression factor= 107.1) suggested by the IPV plots the NNRC-K classifier performance approaches that of the RBF classifier and both algorithms perform quite close to NNR. It is interesting to note that the RBF classifier can attain a given level of probability of error at about twice the compression factor as that of the NNRC-K classifier. We also observe that as the compression factor increases, all three classifiers tend to exhibit larger variances in their probability of error estimates.

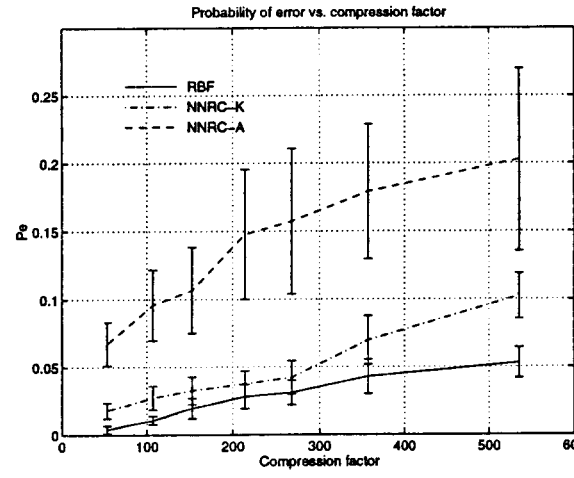

Figure 3: Effect of compression on classification performance

6. CONCLUSIONS

We have presented results which clearly show that a HRR model-based classifier can be constructed with a compressed model catalog without suffering significant performance degradation at compression rates around 100. We examined two efficient classification schemes (NNR and RBF) which use synthetic HRR radar data for ATR applications. The superiority of the RBF classifier, especially at high compression rates is demonstrated. The results are also found to be in agreement with the implications of the IPV plots obtained for the data.

7. REFERENCES

[1] D. R. Hush and B. G. Horne, "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, pp. 8–39, January 1993.

[2] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, no. 2, pp. 159–179, June 1985.

[3] J. S. Baras and S. I. Wolk, "Model based automatic target recognition from high range resolution radar target returns," in *Proceedings of the SPIE International Symposium on Intelligent Information Systems*, Apr. 1994.

[4] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. San Diego, CA: Academic Press, 2 ed., 1990.

[5] J. Moody and C. Darken, "Fast Learning in Networks Of Locally Tuned Processing Units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[6] M. D. Richard and R. P. Lippmann, "Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities," *Neural Computation*, pp. 461–483, 1991.

[7] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, no. 4, pp. 595–603, Apr. 1992.

[8] A. Albert, *Regression and the Moore-Penrose Pseudoinverse*. San Diego, CA: Academic Press, 1972.

[9] B. Widrow and M. A. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, September 1990.