1996-10

Automated construction of a hierarchy of self-organized neural network classifiers

AUTOMATED CONSTRUCTION OF A HIERARCHY OF SELF-ORGANIZED NEURAL NETWORK CLASSIFIERS

Harald Ruda and Magnús Snorrason

October 1996

Technical Report CAS/CNS-96-028

Boston University Center for Adaptive Systems and
Department of Cognitive and Neural Systems
677 Beacon Street
Boston, MA 02215

AUTOMATED CONSTRUCTION OF A HIERARCHY OF SELF-ORGANIZED NEURAL NETWORK CLASSIFIERS

Harald Ruda and Magnús Snorrason

Charles River Analytics
55 Wheeler Street, Cambridge, MA 02138
mailto:hxr@cra.com

Abstract: This paper documents an effort to design and implement a neural network-based, automatic classification system which dynamically constructs and trains a decision tree. The system is a combination of neural network and decision tree technology. The decision tree is constructed to partition a large classification problem into smaller problems. The neural network modules then solve these smaller problems. We used a variant of the Fuzzy ARTMAP neural network which can be trained much more quickly than traditional neural networks. The research extends the concept of self-organization from within the neural network to the overall structure of the dynamically constructed decision hierarchy. The primary advantage is avoidance of manual tedium and subjective bias in constructing decision hierarchies. Additionally, removing the need for manual construction of the hierarchy opens up a large class of potential classification applications. When tested on data from real-world images, the automatically generated hierarchies performed slightly better than an intuitive (hand-built) hierarchy. Because the neural networks at the nodes of the decision hierarchy are solving smaller problems, generalization performance can really be improved if the number of features used to solve these problems is reduced. Algorithms for automatically selecting which features to use for each individual classification module were also implemented. We were able to achieve the same level of performance as in previous manual efforts, but in an efficient, automatic manner. The technology developed has great potential in a number of commercial areas, including data mining, pattern recognition, and intelligent interfaces for personal computer applications. Sample applications include: fraud detection, bankruptcy prediction, data mining agent, scalable object recognition system, email agent, resource librarian agent, and a decision aid agent.

Keywords: Classification, Clustering, Neural Networks, Decision Trees, Feature Selection

INTRODUCTION

Some types of clustering and classification problems remain difficult despite years of progress in the development of classifier algorithms. The current generation of neural network classifiers has shown very good performance and relative ease of use. The difficulties encountered become evident for situations characterized by:

1. Limited availability of training data
2. Non-stationary environment requiring categories to be flexible
3. Large set of potentially important features but no criteria for ranking features

To get around some of these problems, we have used a late generation neural network algorithm called Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds & Rosen, 1992). The Fuzzy ARTMAP network is capable of dealing well with the problem of small training sets. The Fuzzy ARTMAP network is also one of the most adaptive and flexible classifiers; unlike back-propagation algorithms it is capable of on-line learning. It is sensitive to the choice of features, however, and tends to perform less well when presented with a large set of features rather than just the optimal set. However, determining the globally optimal set for a given classification problem domain is as difficult as solving the classification problem itself. To some extent, this sensitivity to the number of features is an issue for all classifiers and is often referred to as the "curse of dimensionality." Back-propagation networks are sometimes able to successfully train with a superset of features, but this prolongs the already long training period even more and is therefore not useful in practice.

In recent work with an object recognition design for an automatic target recognition (ATR) system, we found that the curse of dimensionality can be circumvented by creating a multi-level decision hierarchy. Early-level decisions are made by one classifier-module which passes the responsibility of subsequent decisions to the appropriate next-level classifiers. By partitioning the problem in this way, each module can use a locally optimized feature set and a globally optimal set never needs to be determined. This approach is also very scalable, i.e., it is easily extensible to larger hierarchical structures. Two new problems are introduced:

4. This partitioning into a decision hierarchy has to be done by hand
5. The hierarchical structure is fixed, therefore limiting the overall adaptiveness of the system

Succinctly then, the problem addressed in this paper consists of coming up with a system which can be used when the circumstances are adverse (points 1, 2 and 3) and yet avoids the inflexibility of a static hierarchy (points 4 and 5). This type of system is very useful for both military and civilian applications. The military has frequent need

This work was performed under BMDO contract No. DASG60-95-C-0050 with the U.S. Army Strategic Defense Command, Huntsville, Alabama. The authors thank the Technical Monitor, Janice D. Pryor for her support and direction on this project.

for software that can either automatically classify targets, or aid human target recognition operators in the field. Such systems must be both flexible and robust in a variety of situations. A specific example of hierarchical classification for ATR is summarized in the following subsection. Applications of self-organizing hierarchical classification systems that are not specific to the military abound. Some applications for the techniques discussed in this paper can be found in the discussion.

This paper begins by describing the neural network used for the research into these problems. While this network has many attractive properties, it is important to keep in mind that it is not the only type of classifier that can be used with the algorithms for feature selection and automatic hierarchy construction. Many other neural network models, even some statistical algorithms, can be used instead. Then follows description of the algorithms developed and also a discussion of the results and other implications.

Adaptive Fuzzy ARTMAP

When the issue of classification using neural networks is brought up, the back-propagation (BP) neural network paradigm is often suggested. We have used BP neural networks with some success on pattern recognition problems, but we have found supervised learning Adaptive Resonance Theory, or ARTMAP (Carpenter et al., 1992) neural networks significantly more powerful. Some of the features that distinguish ARTMAP networks from other neural networks, such as BP, Kohonen (1984) maps, Radial Basis Function (RBF), and Hopfield networks are:

- Self-organizing architecture; nodes are added as required.
- Fast learning; input–output pairs learned in a single presentation.
- Stable learning; there are usually no changes in the weights after a few epochs.
- Incremental learning; additional training does not destroy what was previously learned.
- On-line learning; there is no need to distinguish between a training phase and a test phase.
- Immediate access to categories; performance requires only a single pass.
- Confidence level; degree of membership of the input vector in the category's template.
- Coarseness of classification is controlled via a single parameter, vigilance.

Fuzzy ARTMAP (Carpenter et al., 1992) is a supervised neural network classifier that learns to classify inputs by a fuzzy set of features, or a pattern of fuzzy membership values between 0 and 1 indicating the extent to which each feature is present. This way the ARTMAP's disadvantage of only processing binary patterns is removed without affecting the various advantages listed above. Fuzzy ARTMAP also differs from most other fuzzy pattern recognition algorithms in that it learns each input as it is received on-line, rather than by performing an off-line optimization of a criterion function.

Our Simplified Version

The full implementation as described in Carpenter et al. (1992) is rarely needed, as it is intended to associate an arbitrary output pattern with each input pattern. This association is a mapping between clusters of input patterns and clusters of output patterns, and these clusters can be formed with a controlled degree of "looseness" or coarseness, adjustable via the base-vigilance of the Fuzzy ART clustering mechanism. Since there is one Fuzzy ART for the inputs and one for the outputs, there are two such parameters. The base-vigilance for the inputs is generally set low, often at the minimum (0.0) in order to form the largest clusters possible. This is desirable because larger clusters usually provide better generalization.

The base-vigilance for output clustering is typically set very high, often at the maximum value (1.0) in order to be able to predict outputs precisely. When the base-vigilance of the output clustering mechanism is at the maximum value, it is possible to use a direct mapping from each input cluster to a set of output values instead of the more complicated output clustering and a map. It is also the case that an output can normally be designated with one number, such as output "1", "2", etc. In summary, the system implemented here is a complete Fuzzy ART input clustering mechanism (F1 and F2 layers in Figure 1 below) with a mapping for each cluster to a non-negative floating-point value (F3 layer).

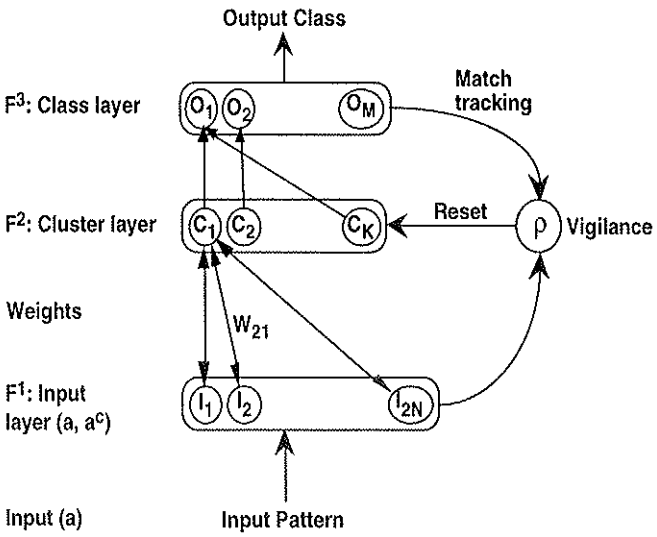

Figure 1: Simplified Fuzzy ARTMAP architecture

The version implemented is an Adaptive Fuzzy ARTMAP, developed (Ruda & Snorrason, 1995) to allow for true on-line learning without pre-processing the input features. Thus the usual Fuzzy ARTMAP restriction that input features are in the range from zero to one is relaxed; in fact, the range does not have to be known ahead of time. In fact, our adaptive preprocessing for Fuzzy ARTMAP completely eliminates the need for manual scaling of input values to the classifier. This is not only a convenience during training and evaluation, but also a necessity for true on-line operation.

Construction of Decision Hierarchies

For this research, certain constraints were set on the algorithms to be developed at an early stage. The basic idea was to use neural networks at the nodes of a hierarchical classifier; require learning to be relatively fast and not involve long searches for optimal parameters; and initially allow only batch processing.

The benefit of using neural networks at the nodes of the classification tree compared to traditional tree classifier algorithms is that more than one feature can be used to make the decision at each node . This allows for a more natural and flexible decision-making process. As a result, it is possible to build hierarchies that reflect meaningful choices at the nodes.

Classical hierarchical methods build the decision tree using a search through all available possibilities at each stage and optimizing based on those choices. But in order to make training of the hierarchical classifier reasonably fast it is important not to try every combination. At most a small number of possibilities should be attempted before finalizing the choice at any given node.

A further choice was made to limit the algorithms to batch learning. In other words, learning is done with all the training patterns available. This is a fairly common restriction on classification algorithms. Fuzzy ARTMAP has the ability to learn on-line, i.e., with patterns presented one at a time. On-line Fuzzy ARTMAP does not guarantee 100% correct performance on the training set as it does under batch learning, but we have found the performance to be quite good, especially for large training sets. We anticipate developing versions of the hierarchy construction algorithms which will have on-line capability as well.

Prune and Grow

The prune-and-grow algorithm for automatically constructing the classification hierarchy is a natural extension of the ARTMAP algorithm. Problems with classification occur when a clustering classifier, such as ARTMAP, creates some clusters that are relatively unpopular; i.e., it is not creating clusters that generalize well. Additionally, generalization is more difficult when there are too many clusters from which to choose.

The first method then consists of performing the classification using an ARTMAP and, where clusters are too numerous or too small, to remove selected clusters. When those clusters are removed, the patterns are re-classified using the pruned ARTMAP. For any clusters which now represent some of the patterns previously belonging to pruned clusters, the procedure is repeated recursively.

Tune and Descend

The other developed algorithm for automatically constructing the classification hierarchy is based on the use of unsupervised ART networks at each node. Thus, although the training patterns do have labels associated with them, these labels are not used for the training of the ART networks at the nodes. Different values of ART's vigilance parameter r are tried in order to find the value which maximizes a distribution requirement, such as Equation 1. The best value for the vigilance is then used, and the procedure is repeated recursively for each cluster that does not represent a sufficiently pure classification. In other words, for a homogeneity criterion of 100 %, the procedure stops when all the patterns classified into a given cluster have the same label.

Another way to view this algorithm is as a way to exploit the natural clustering tendencies in the data and not force any divisions. Each classifier must split the data into at least two groups. The questions are: (1) what value of the vigilance parameter should be used (in the case of ART classifiers) and (2) which features should be used. Some analysis is needed to find the value of the vigilance needed to add another cluster. It may be just the minimum match value, in which case the procedure resembles ARTMAP's "match tracking." Match tracking is about learning a specific pattern, whereas this procedure would increase the number of clusters, not knowing which patterns will be affected.

Clearly, a function to evaluate the distribution requirement must be chosen. Following Auray et al. (1991) we have used the Daroczy entropy (Daroczy, 1970) which can be defined as:

$$H(s) = \frac{1}{2^{-\beta+1}-1}\left[\left(\sum_{i=1}^{m}\left(f(w_i/X_s)\right)^{\beta}\right)-1\right] \qquad \text{(Equation 1)}$$

where

$$f(w_i/X_s) = \frac{n^i_s + \lambda}{\sum_{i=1}^{m}\left(n^i_s + \lambda\right)} \qquad \text{(Equation 2)}$$

Where X_s represents all the patterns learned by node s; w_i, all the patterns in class number i (out of m classes); and n^i_s is the number of patterns belonging to class w_i at node s. The parameters were fixed at $\lambda = 1$, and $\beta = 0.5$. Other functions are, of course, possible, especially other types of entropy or uncertainty measures.

The tune-and-descend algorithm has the advantage of being easily extended to deal with situations where there are no labels available for the patterns. The only required change would be to use a distribution requirement insensitive to the correctness of classification. The distribution requirement would instead have to be purely sensitive to the "goodness" of the classes created. This type of extension for unlabeled patterns is not possible with the prune-and-grow algorithm.

Feature Selection

The main procedure for feature selection is a step-wise improvement algorithm that starts with a set of all available features and then removes one feature at a time. The process of removing a feature from the set actually consists of removing two features and then adding one back. It is important to add a feature after removing two in order to account for newly exposed beneficial correlations between features. When a feature is removed, it is chosen in order to maximize the evaluation of the feature set. Similarly, when a feature is added, the feature that maximizes the evaluation is the one chosen.

Clearly, it is important to choose the right method for evaluating feature sets. Some procedures for evaluating feature sets involve measuring the performance of the classifier. These are desirable since it is classifier performance that we ultimately want to improve. By implementing classifiers with an abstract interface, it is possible to use these functions to perform feature selection on any type of classifier, without needing to know about the internal workings of the classifier. Four methods for evaluating feature sets were implemented:

1. Train on part of a data set and test on the rest, reporting the percentage correct. This is a basic approach that utilizes the classifier directly. Using one cut of training and testing data makes this method somewhat dependent on the actual training and testing set chosen. Also, if the classifier is not fast, then this method can be quite slow.

2. Train and test on many different splits of training and testing data, reporting the average percentage correct. This method overcomes the main limitation of the first at the expense of computation time. For example, if 30 resamplings are used provide a statistically meaningful average of the performance for a given feature set, then this method will take 30 times longer than the first method.
3. Cross validation is a well-known approach to evaluating classifiers. If there are N patterns or groups of patterns, then the classifier is trained N times, each time leaving out the Nth pattern or group. Then the classifier is tested on a pattern or group on which it was not trained. The sum of the errors for all N conditions divided by N is the error rate. This method works well when the number of patterns or number of distinct groups is small. For larger data sets, the time it takes to perform the procedure grows at least $O(N^2)$.
4. We also developed a method that depends critically on the weight structure of the Fuzzy ARTMAP classifier. This method examines the weights of clusters and evaluates the amount of overlap between the coding of different classes. In general terms, large overlaps are considered less desirable than clean separations. This method may be extensible to other types of classifiers, but that has not yet been verified.

There are of course other methods of feature selection and means of evaluating feature sets (see Kittler (1986) for examples). However, these methods generally depend on knowing a great deal about the distribution of classes in feature-space. In most practical applications, those distributions are not known.

Results & Discussion

To evaluate the hierarchy construction and feature selection algorithms, we decided to use data sets for which we already had meaningful results achieved using manual methods. The data consisted of object features calculated from laser radar imagery. The objects were segmented, and a large number of features were calculated for each object using algorithms developed using the Khoros image processing system.

Manual Hierarchy Construction

The manually constructed hierarchy is shown in Figure 2 and mirrors the coarse-to-fine levels of specification used by the Air Force in these tasks (Detection, Classification, Recognition, Identification, Characterization). A classifier hierarchy was constructed that mirrors the hierarchy in this figure.

Automated Hierarchy Construction Results

Using the data for 299 truth-labeled objects in 25 different classes, we obtained the results shown in Table 1. Note that partially correct answers got no credit; i.e., if an object was correctly specified as a member of a non-leaf category, but incorrectly classified into a subclass or a leaf-node, the answer was counted as wrong. Many times, especially with the hierarchical algorithms, the wrong answer is very "close," so this measurement doesn't completely reflect the advantages of the hierarchical algorithms. For example, the classifier may correctly classify an M1 tank as a Target, Mobile, Armor, but then mistake it for an M2. For each classifier in the table, 50% of the objects were used for training and the rest for testing.

The single Fuzzy ARTMAP is the basic point of comparison. This is the normal method of using neural network classifiers, but,

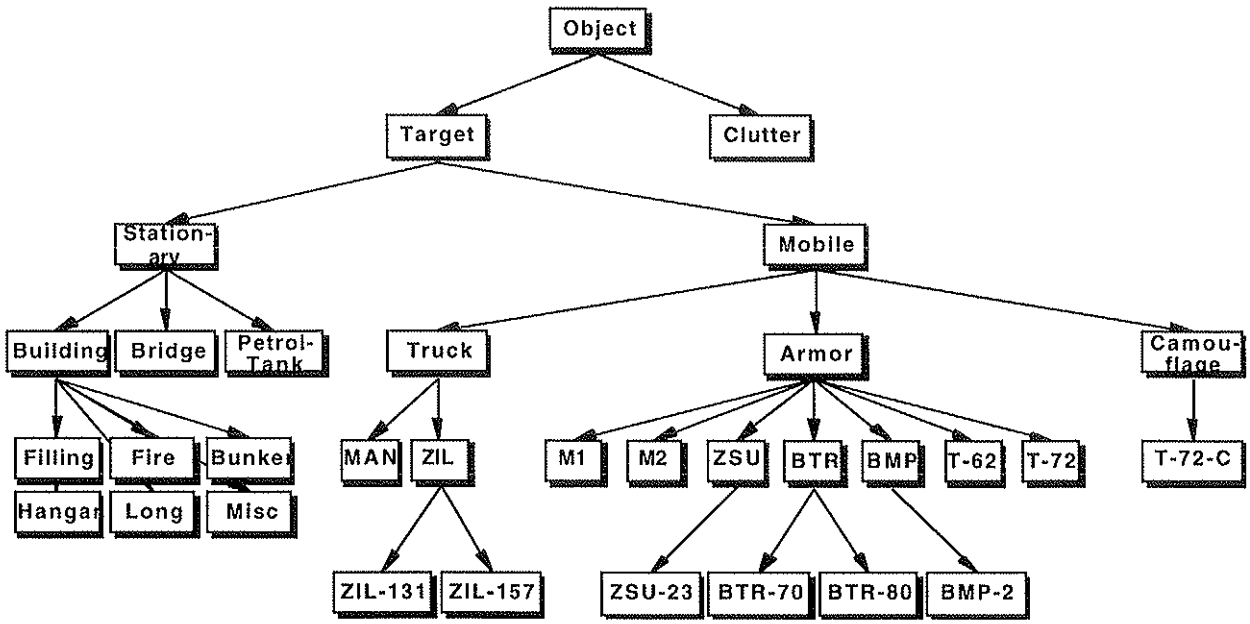

Figure 2: A hierarchical taxonomy of the objects in the ATR task

Classification Method	% Correct	Comments
Single Fuzzy ARTMAP	64.7	Single neural network
Predetermined Hierarchy	68.0	Manually specified
ARTMAP Hierarchy	70.0	Prune-and-grow
ART Hierarchy	52.7	Tune-and-descend

Table 1: Results for the automatic hierarchy construction

when the number of categories is large, it does not work as well. Using the predefined hierarchy does improve the percentage correct. The classification problem for this study was chosen to be difficult so that improvements would readily be noticed, therefore the performance numbers in all cases are not very high.

It is clear that automatic classification using Fuzzy ARTMAPs (prune-and-grow) is slightly better than the classification using the manually constructed hierarchy. Unfortunately the tune-and-descend algorithm, using unsupervised ART networks, did not do well at all. Not only is the performance significantly lower than with the first algorithm, but the number of clusters used to reach that performance is much larger (generally more than twice as many), indicating a problem with generalization. It is possible that choosing a different type of entropy distribution function for evaluating the clustering would improve the performance of this algorithm.

In summary, the automatic construction of the hierarchical classifier succeeded in replacing the hand-crafted classifier hierarchy when the prune-and-grow algorithm was used. This is very encouraging, as further improvements, especially choosing the features at each node using the feature selection methods already discussed, could enhance the performance of automatically constructed hierarchies. The ability to customize the classifier itself at each node is another benefit of using hierarchical classifiers.

Visualization of Automatic Hierarchy Construction

To provide a sense of the dynamic nature of the hierarchy construction, this section contains some figures of the system "in action". The hierarchy constructed in these pictures is automatically generated according to the prune-and-grow algorithm. Remember that this is just one of many possible shapes that the hierarchy can take, depending on the ordering of the input data. Figure 3 shows the hierarchy in the beginning stages. By Figure 4 the hierarchy has grown, but is still not finalized.

In Figure 5 the final structure is shown. Notice that the empty boxes denote clusters which are homogenous and all patterns coded in that cluster belong to the same class. Clusters with numbers are the ones that had to be "subdivided".

Manual Feature Selection

In Caglayan, Snorrason, & Ruda (1996), a two-level classification tree was manually constructed for target identification. The top-level classifier determined whether an object is a target or something else. If it is not a target, then it is ignored. If it is a target, then it is

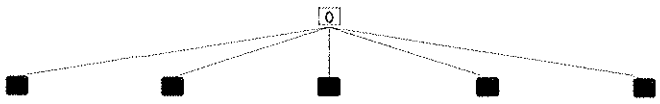

Figure 3: Early stage of hierarchical construction

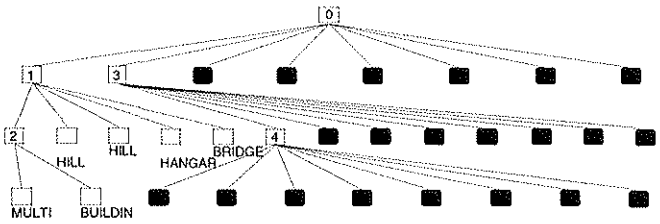

Figure 4: Intermediate stage of hierarchical construction

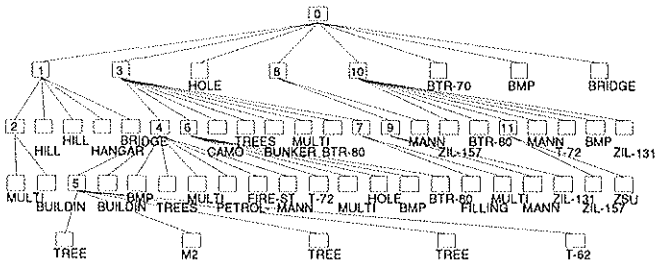

Figure 5: Hierarchical construction has been completed

passed to the lower level classifier which determines the type of the target is a tank. There were not enough data in this particular data set to construct more elaborate hierarchies.

By using shell scripts and a few small programs written to aid in the task, the features to be used for the top-level classifier were selected by hand. The data set was not large, containing a total of 44 objects of which 20 were targets. The data set was split into two equal parts, and features were selected based on performance on the test set. A total of 56 features were initially available. Using the manual method of feature selection, this number was whittled down to five features. This small set of five features yielded 100 % correct classification of the test set when a network was trained on the training set.

Feature Selection Results

Automatic feature selection using the straightforward method of splitting the data set into two parts was used (method 1 described in the section on Feature Selection). The automatic method was set up to duplicate the previous manual effort as closely as possible. The automatic method also came up with a set of five features which yielded 100% correct classification of the test set. Except for one feature, the set of features was identical to the one produced manually. This was most definitely a success in terms of automating the feature selection process.

Automatic feature selection is a very powerful tool since all the work is performed by the program. Indeed, while using this tool we realized that the previous result of 100% correct classification does not correspond to the true generalization ability of the classifier.

Instead of using just one split of the data into training and testing sets, we found that one should use a larger number of random splits to evaluate feature sets (method 2 described in the section on Feature Selection). The best feature set produced using this method yields about 89% correct on classification. This would not have been feasible had we been restricted to a manual selection of features; the automatic feature selection capability, however, allowed us to rapidly explore options. Note that this is very much like Monte Carlo methods used in physics and other computational sciences to estimate the validity of models.

Potential Applications

One of the main benefits of this research is to open up new domains of application for hierarchical classifiers. Previously, hierarchical classifiers could only be used in situations that allowed for simple decisions at each node (such as used in traditional tree classifiers like ID3, C4.5, etc.) or that allowed the hierarchy to be specified by hand (as described above). With the hierarchical classifier system designed in this project, the domain of applicability expands to include all situations that require both complex decisions at each node and dynamic construction of the hierarchy.

We determined a number of potential commercial applications that build directly on this research. The applications fall in the general areas of data mining, pattern recognition, and intelligent interfaces for personal computer applications. Specific applications include: fraud detection, bankruptcy prediction, data mining agent, scalable object recognition system, email agent, resource librarian agent, and a decision aid agent.

Conclusions

In the research effort, we showed the feasibility of automating classification tasks formerly performed with manual guidance. Algorithms were developed as part of this effort to develop new technology that can be used in a variety of domains. In particular, our study:

- Showed proof of feasibility for the automatic construction of a hierarchy of self-organized neural network classifiers.
- Successfully demonstrated a system which can automatically construct a hierarchy that recursively partitions a complex classification problem into simpler problems. Complex classification problems have many output classes or complex internal structures; for example, a sample target recognition task with 25 output classes was split into a hierarchy with eleven classifiers five levels deep (see Figure 5).
- Successfully demonstrated the ability to enhance classifier performance by automatically choosing a small number of features (e.g., ~5) from a large set of possible features available as input.
- Implemented these designs in C++. The implementation is portable across computer architectures and across operating systems.
- Validated the implementation using real data with results equal to or better than manual efforts. Table 1 above shows the results for hierarchy construction.
- Established a direct foundation for further research and development efforts.

In summary, we have demonstrated the feasibility of our approach to automating the task of constructing large-scale classification systems. Furthermore, we feel that this approach of self-organizing hierarchies can be used to automate even more aspects of the design and construction of classification systems. Systems designed with this technology will be more flexible and easier to train and retrain.

References

Auray, J.-P., Duru, G., Terrenoire, M., Tounissoux, D., & Zighed, A. (1991). "Segmentation and classification. An application to patients' risk estimation". In Duru & Paelinck (Eds.), **Econometrics of Health Care**. Kluwer Academic Publishers.

Caglayan, A. K., Snorrason, M., & Ruda, H. (1996). Multi-Level AutonomousTargetIdentification(FinalReportR93131):Charles River Analytics (February 6, 1996).

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). "Fuzzy ARTMAP: Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps". *IEEE Transactions on Neural Networks,* 3(5), 493-504.

Daroczy, D. (1970). "Generalized Information Functions". *Information and Control,* 16, 16.

Kittler, J. (1986). "Feature Selection and Extraction". In Young & Fu (Eds.), **Handbook of Pattern Recognition and Image Processing**. New York, NY: Academic Press.

Kohonen, T. (1984). **Self-Organization and Associative Memory**. Berlin: Springer Verlag.

Laird, P., & Saul, R. (1994). "Automated feature extraction for supervised learning". *IEEE World Congress on Computational Intelligence,* Orlando, FL.

Ruda, H., & Snorrason, M. (1995). "Adaptive Preprocessing for On-Line Learning with Adaptive Resonance Theory (ART) Networks". **IEEE Workshop: Neural Networks for Signal Processing V**, Cambridge, MA.