Predictive Adaptive Resonance Theory and Knowledge Discovery in Databases

Ah-Hwee Tan[1] and Hui-Shin Vivien Soon[2]

[1] Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613
ahhwee@krdl.org.sg
[2] Ngee Ann Polytechnic, 535 Clementi Road, Singapore 599489 shs@np.edu.sg

Abstract. This paper investigates the scalability of predictive Adaptive Resonance Theory (ART) networks for knowledge discovery in very large databases. Although predictive ART performs fast and incremental learning, the number of recognition categories or rules that it creates during learning may become substantially large and cause the learning speed to slow down. To tackle this problem, we introduce an on-line algorithm for evaluating and pruning categories during learning. Benchmark experiments on a large scale data set show that on-line pruning has been effective in reducing the number of the recognition categories and the time for convergence. Interestingly, the pruned networks also produce better predictive performance.

1 Introduction

One of the major challenges faced by the predictive modeling techniques is the efficiency and the scalability to very large databases. Gradient descent based neural network models require many learning iterations through the training data and are highly computational intensive. This paper presents an alternative approach to knowledge discovery using a predictive self-organizing neural network model, known as the Adaptive Resonance Associative Map (ARAM) [5].

Predictive self-organizing networks [1] perform fast incremental supervised learning of recognition categories (pattern classes) and multi-dimensional mappings of binary and analog patterns. When performing classification tasks, ARAM formulates recognition categories of the input and output pattern pairs. Unfortunately, for very large databases, the number of the recognition categories may become substantially large, causing the learning time to increase significantly. To tackle this problem, we introduce an on-line algorithm to estimate the *merit* or *confidence values* of the recognition categories of an ARAM network during learning. Each newly created category node is given a confidence value of 1. A *forgetting* process constantly reduces confidence values towards 0 at a specific interval. In conjunction, a *reinforcement* process increases confidence values towards 1s when correct predictions are produced by their respective recognition categories. The confidence values of the category nodes are then compared with a threshold parameter. Categories with confidence values below the threshold are removed from the network.

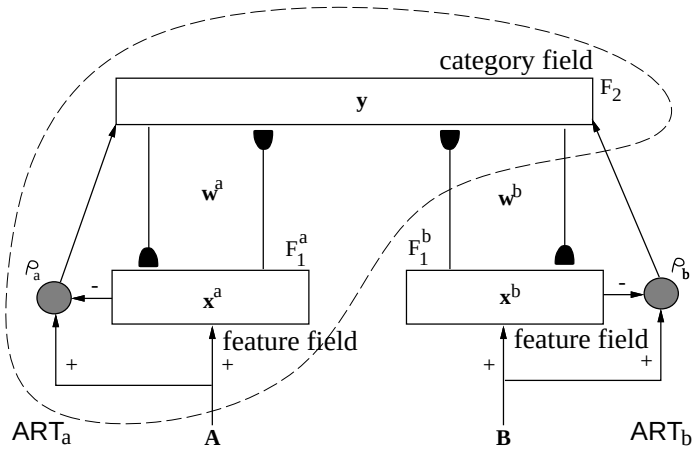

Fig. 1. The Adaptive Resonance Associative Map architecture.

2 Fuzzy ARAM

An ARAM network (Figure 1) consists of two input feature fields, F_1^a and F_1^b, connected by bi-directional and conditionable pathways to a category field F_2. For classification problems, F_1^a serves as the input field representing input activity vectors and F_1^b serves as the output field representing output class vectors.

Given a pair of input and output patterns, ARAM computes a choice function for each F_2 recognition category. The winning node that has the maximal choice function value then triggers a top-down priming on F_1^a and F_1^b to check if its associated input and output weight vectors satisfy the vigilance criteria in their respective modules. If so, under fast learning, the weight templates of the F_2 recognition category are modified towards their intersection with the input and output vector pair. Otherwise, the recognition category is reset and the system repeats to select another category node until a match is found. By synchronizing the unsupervised clustering of the input and output pattern sets, ARAM learns supervised mapping between the input and output patterns. As code stabilization is ensured by restricting learning to states where resonances are reached, fast learning in a real-time environment is feasible. Please refer to [5] for the detailed algorithm.

3 ARAM Complexity and Category Pruning

Let P be the number of the input and output training pattern pairs, Q be the number of the recognition categories created by an ARAM network, M be the number of the input attributes, and N be the number of the output classes. The time complexity per learning iteration of the ARAM algorithm is given by

$O(PQ(M+N))$. Since M and N are typically fixed for a specific knowledge discovery task, the complexity thus depends on P and Q. When Q is small, say around $\log(P)$, the time complexity is $O(P\log P)$. However, if Q is large, the complexity grows to P^2 in the worst case.

To improve learning efficiency, we propose a method for evaluating and eliminating categories that are created by spurious cases during learning. Each category node j is associated with a confidence factor c_j, a real number between 0 and 1. For a newly committed node j, c_j equals 1. At a fixed interval, a *forgetting* process constantly causes c_j to decay towards 0. A *reinforcement* process increases c_j towards 1 whenever a correct prediction is made by the category node j during learning.

Confidence erosion: At a chosen interval, the confidence value of each recognition category depreciates towards 0 according to

$$c_J^{(\text{new})} = c_J^{(\text{old})} - \zeta c_J^{(\text{old})}, \tag{1}$$

where $\zeta \in [0,1]$ is the confidence decay parameter. The erosion process is self-scaling in the sense that the decay becomes smaller as c_J gets smaller.

Confidence reinforcement: When a category node J, chosen by the code competition process, makes a correct prediction, its confidence value c_J is increased towards 1 by

$$c_J^{(\text{new})} = c_J^{(\text{old})} + \eta(1 - c_J^{(\text{old})}), \tag{2}$$

where $\eta \in [0,1]$ is the confidence reinforcement parameter.

Category pruning: The computed confidence values are then compared with a threshold parameter $\tau \in [0,1]$. A category is removed from the ARAM network when its confidence value falls below τ.

Convergence criterion: After each training iteration, the pruned network is evaluated against the training set for its predictive performance. Training is stopped when the improvement on the training set performance (in terms of percents, compared with that obtained in the previous iteration) falls below a convergence threshold ξ.

4 Experiments

The adult data set [4] is one of the largest public domain data set. It contains 48,842 records, each characterized by six continuous attributes and eight nominal features. The task is to predict whether a person with a particular set of personal attributes draws a salary greater or less than US$50,000. The adult data set was noted as a hard domain with a good number of records and a mix of continuous and discrete features.

Fuzzy ARAM experiments used a standard set of parameter values: choice parameters $\alpha_a = \alpha_b = 0.1$, learning rates $\beta_a = \beta_b = 1.0$, and contribution parameter $\gamma = 1.0$. For on-line pruning, we used $\zeta = 0.005$ for confidence decay, $\eta = 0.5$ for confidence reinforcement, $\tau = 0.05$ for pruning threshold, and $\xi = 0.5$ for stopping criterion. For prediction, ARAM used K-max rule [5] with $K = 3$.

Table 1. Simulation results of fuzzy ARAM on the adult data set compared with those of KNN, C4.5, and NBTree. A '-' indicates that a value is not available for comparison. The results of ARAM were averaged over 10 simulations.

Methods	# Epochs	# Nodes/ Rules	Time (Secs)	Test Accuracy
1-Nearest-Neighbor	1	30162	-	78.6
3-Nearest-Neighbor	1	30162	-	79.7
C4.5	-	2213	-	84.6
NBTree	-	137	-	85.9
Fuzzy ARAM	11.5	3826	8226	81.0
+ On-line pruning	7.5	343	1125	84.1

As shown in Table 1, fuzzy ARAM created a total of 3,826 category nodes from the 30,152 training patterns, with a compression ratio of around 8. Each learning process took an average of 11.5 iterations to converge. One complete benchmark experiment, including training and testing, took 8373 seconds (more than two hours) using an Ultra-60 SUN SPARC machine. With on-line category pruning, fuzzy ARAM produced an average of 343 categories. The networks also converged faster in an average of 7.5 iterations. One complete benchmark involving 30,152 training cases and 15,060 test cases took about 1,125 seconds (less than 20 minutes). Comparing predictive performance, fuzzy ARAM obtained an accuracy of 81.0% on the test cases, about the same as those of K-Nearest-Neighbor. With on-line category pruning, the test set accuracy improved to 84.1%, roughly comparable to those obtained by C4.5 and NBTree [3].

References

1. Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., Rosen, D. B.: Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, **3** (1992) 698–713
2. Carpenter, G. A., Tan, A.-H.: Rule extraction: From neural architecture to symbolic representation. *Connection Science*, **7** (1995) 3–27
3. Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. *Proceedings, KDD-96* (1996)
4. Murphy, P. M., Aha, D. W.: UCI repository of machine learning databases [machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science (1992)
5. Tan, A.-H.: Adaptive Resonance Associative Map. *Neural Networks* **8** (1995) 437–446

This article was processed using the LaTeX macro package with LLNCS style