Adaptive Neural Network Clustering of Web Users

A neural network based on adaptive resonance theory dynamically groups users based on their Web access patterns. A prefetching application of this clustering technique showed prediction accuracy as high as 97.78 percent.

Santosh K. Rangarajan
Vir V. Phoha
Kiran S. Balagani
Rastko R.Selmic
Louisiana Tech University

S.S. Iyengar
Louisiana State University

The degree of personalization that a Web site offers in presenting its services to users is an important attribute contributing to the site's popularity. Web server access logs contain substantial data about user access patterns. Properly exploited, the logs can reveal useful information about each user's interests in a site; but restructuring a site's structure to individual user interests increases the computation at the server to an impractical degree. One way to solve this problem is to group users on the basis of their Web interests and then organize the site's structure according to the needs of different groups. Two main difficulties inhibit this approach: the essentially infinite diversity of user interests and the change in these interests with time.

We have developed a clustering algorithm that groups users according to their Web access patterns.[1] The algorithm is based on the ART1 version[2,3] of *adaptive resonance theory*.[4] ART1 offers an unsupervised clustering approach that adapts to changes in users' access patterns over time without losing earlier information. It applies specifically to binary vectors. In our ART1-based algorithm, a prototype vector represents each user cluster by generalizing the URLs most frequently accessed by all cluster members.

We have compared our algorithm's performance with the traditional k-means clustering algorithm. Results showed that the ART1-based technique performed better in terms of intracluster distances. We also applied the technique in a prefetching scheme that predicts future user requests. Its prediction accuracy was as high as 97.78 percent.

ARCHITECTURE AND METHODOLOGY

Figure 1 illustrates the overall architecture of our method and its application to prefetching. From each client request recorded in the proxy server's Web log file, the feature extractor defines a feature vector. The ART1-based clustering algorithm uses the feature vector offline to determine the group to which the client belongs. The algorithm then returns an updated prototype vector of that group, and the prefetcher requests all URL objects represented by the prototype vector.

Preprocessing Web logs

We used NASA Web log files (http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html) to test our approach. The files contain HTTP requests to NASA Kennedy Space Center's Web server. We used the logs containing the HTTP requests from 1 July 1995 through 15 July 1995.

The form for the raw data from the log file is

```
< host name, timestamp, requested
  URL, HTTP reply code, bytes sent
  in reply >
```

The "host name" identifies the host making a request to the NASA Web server. We preprocessed the server log files, filtering them to capture access patterns for the 70 hosts whose requests constituted most of the Web log activity. We removed the remaining hosts because they did not generate enough requests to constitute a group.

Each host represents a large community of organizationally related users. For example, all requests

Published by the IEEE Computer Society

with the hostname www.latech.edu represent requests from the students and faculty of Louisiana Tech University.

We then cleaned the Web logs to retain the URLs that the 70 selected hosts requested most frequently. The frequency of hits to the 200 retained URLs ranged from 32 to 3,435 for 114,290 total hits.

Extracting feature vectors

The base vector $\mathbf{B} = \{URL_1, URL_2, \ldots URL_{200}\}$ represents the access pattern of the hosts. For each host H, the feature extractor forms a binary input pattern vector $\mathbf{P}_H$, which is an instance of the base vector. The pattern vector maps the access frequency of each base vector element, URL_i, to binary values. It is of the form $\mathbf{P}_H = \{P_1, P_2, \ldots P_{200}\}$ where $1 \le H \le 70$ and P_i is an element of $\mathbf{P}_H$ having a value of either 0 or 1.

Generated by the following procedure, the pattern vector is the input vector to our ART1-based clustering algorithm:

```
For each pattern vector P_H,
    H = 1 to 70
  For each element P_i in pattern
    vector P_H, i = 1 to 200
          { 1 if URL_i is requested
              by the host two or
              more times
    P_i = { 0 if URL_i is requested
              by the host less than
              two times
  End
End
```

where H stands for Host ID and i stands for URL ID.

Clustering users

The ART1 algorithm that we adopted for host clustering is a competitive neural net that consists of two subsystems:

- *attentional subsystem*, consisting of a comparison network layer F_1, a recognition layer F_2, and control gains G_1 and G_2. F_1 and F_2 are fully connected with top-down weights and bottom-up weights.
- *orientation subsystem*, consisting of the vigilance parameter ρ, which determines the mismatch allowed between the input pattern vectors and the weights connecting F_1 and F_2.

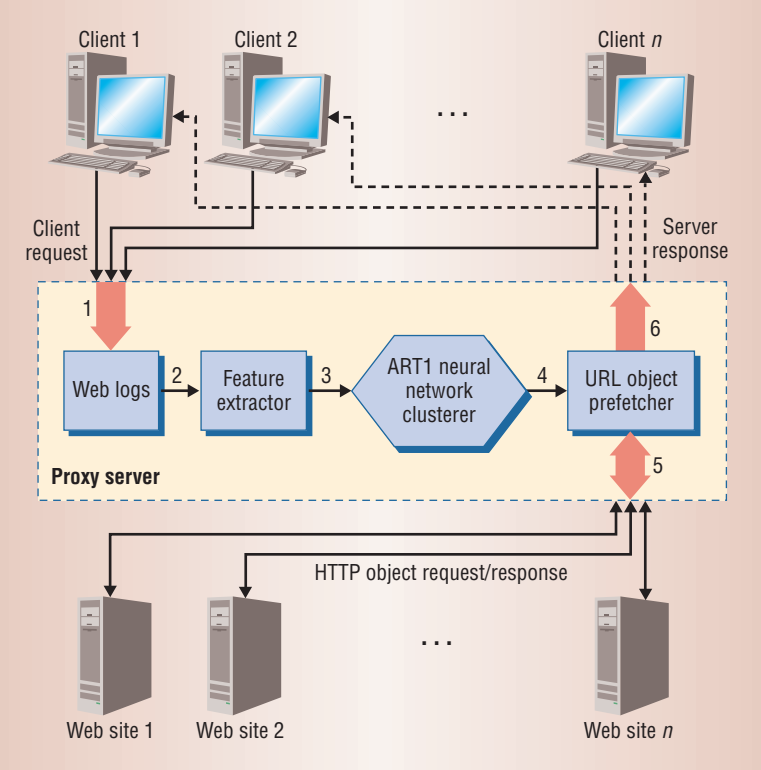

Figure 1. ART1-based clustering and prefetching architecture. (1) Each client request is recorded in the proxy server's Web log file; (2) the feature extractor extracts each client's feature vector, (3) which becomes the input to the offline ART1-based clusterer; (4) the clusterer identifies the group to which the client belongs and returns that group's prototype vector; (5) the prefetcher requests all URL objects that the prototype vector represents; and (6) the proxy server responds to the client with prefetched URL objects.

The input pattern vector $\mathbf{P}_H$ is presented at the F_1 neural network layer. The control gain G_1 is set to 0 to indicate that all nodes in F_2 are actively competing. Each input vector activates a node (winner) in the F_2 layer—specifically, the node with the highest value based on computing the product of the input pattern vector and the bottom-up weight vector. The F_2 layer then reads out the top-down expectation of the winning node to F_1, where the expectation is normalized over the input pattern vector and compared with the vigilance parameter ρ.

If the winner and input vector match within the tolerance allowed by the vigilance parameter, the ART1 algorithm sets the control gain G_2 to 0 and modifies the top-down weights corresponding to the winner. If a mismatch occurs, the control gains G_1 and G_2 are set to 1 to disable the current node and process the input on another uncommitted node. Once the network stabilizes, the top-down weights corresponding to each node in the F_2 layer represent the prototype vector for that node.

Figure 2 illustrates the architecture for our ART1-based neural network for clustering user communities.[1] It consists of 200 nodes in the F_1 layer, with each node presented a 0 or 1 binary

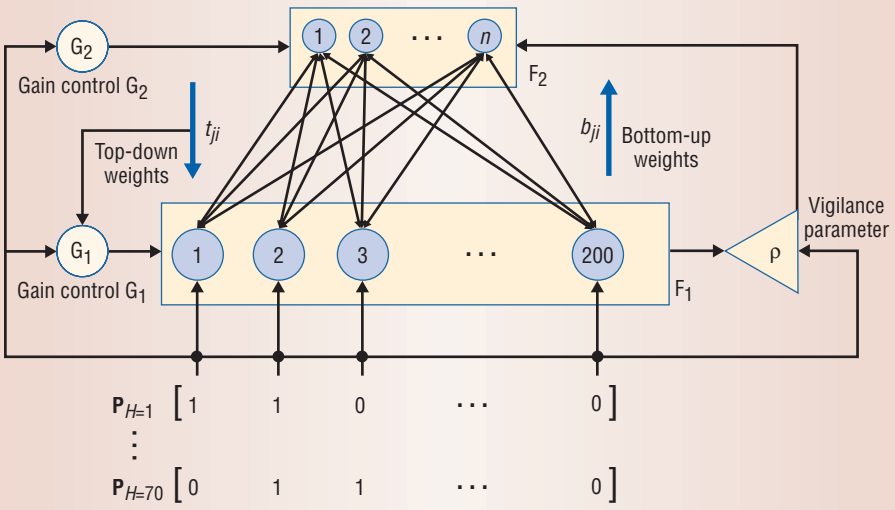

Figure 2. ART1-based clustering architecture. The pattern vector P_H, which represents the access patterns of the host H, is the input to the comparison layer F_1. The vigilance parameter ρ determines the degree of mismatch to be tolerated. The nodes at the recognition layer F_2 represent the clusters formed. Once the network stabilizes, the top-down weights corresponding to each node in F_2 represent the prototype vector for that node.

Web Data Mining and User Clustering

Clustering users based on their Web access patterns is an active area of research in Web usage mining. An influential paper on the application of data mining techniques to the Web proposed a generally accepted taxonomy that divides the research domain into Web content mining and Web usage mining.[1] Minos N. Garofalakis and colleagues reviewed popular data mining techniques and algorithms for discovering Web, hypertext, and hyperlink structure.[2]

Data clustering is a particular kind of data-mining problem. A generalization-based approach[3] that uses access patterns to generate hierarchical clustering of Web users combines attribute-oriented induction and the Birch (balanced iterative reducing and clustering using hierarchies)[4] method. Hierarchical clustering is a statistical method for finding clusters of identical data points.

Igor Cadez and colleagues[5] use first-order Markov models to cluster users according to the order in which they request Web pages. Georgias Paliouras and colleagues[6] analyze the performance of three clustering algorithms—autoclass, self-organizing maps, and cluster mining—for constructing community models for site users.

Our work in Web usage mining includes research in the automatic discovery of user access patterns from Web server data.[7] Other work from Louisiana Tech applies Dempster-Shafer's mass distribution concept and proposes a belief-function similarity measure.[8] With this approach, the clustering algorithm can handle uncertainty in Web users' navigation behavior.

Although all the methods described here succeed in grouping users according to their diverse Web interests, they lack the ability to adapt to changes in those interests over time.

References

1. R. Cooley, B. Mobasher, and J. Srivatsava, "Web Mining: Information and Pattern Discovery on the World Wide Web," *Proc. Int'l Conf. Tools with Artificial Intelligence* (ICTAI 97), IEEE CS Press, 1997, pp. 558-567.
2. M.N. Garofalakis et al., "Data Mining and the Web: Past, Present, and Future," *Proc. 2nd Int'l Workshop Web Information and Data Management*, ACM Press, 1999, pp. 43-47.
3. Y. Fu, K. Sandhu, and M. Shih, "Clustering of Web Users Based on Access Patterns," *Proc. Int'l Workshop on Web Usage Analysis and User Profiling* (WEBKDD 99), ACM Press, 1999; www.acm.org/sigkdd/proceedings/webkdd99/toconline.htm.
4. T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An Efficient Data Clustering Method for Very Large Databases," *Proc. ACM SIGMOD Conf. Management of Data*, ACM Press, 1996, pp. 103-114.
5. I. Cadez et al., "Visualization of Navigation Patterns on a Website Using Model-Based Clustering." tech. report MSR-TR-00-18, Microsoft Research, Mar. 2002.
6. G. Paliouras et al., "Clustering the Users of Large Web Sites into Communities," *Proc. 17th Int'l Conf. Machine Learning*, Morgan Kaufmann, 2000, pp. 719-726.
7. V.V. Phoha, S.S. Iyengar, and R. Kannan, "Faster Web Page Allocation with Neural Networks," *IEEE Internet Computing*, Nov./Dec. 2002, pp. 18-26.
8. Y. Xie and V.V. Phoha, "Web User Clustering from Access Log Using Belief Function," *Proc. 1st Int'l Conf. Knowledge Capture* (K-CAP 01), ACM Press, Oct. 2001, pp. 202-208.

```
Procedure: ART1-Based Prefetching
Preprocessing: Cluster the hosts using the ART1-based clustering algorithm. Each
  cluster is denoted by C_n, where n is the number of clusters formed. The clusters C_1,
  C_2, … C_k, … C_n are represented by prototype vectors. The prototype vector for the k^th
  cluster is of the form T_k = (t_k1, t_k2, ... t_k200), where t_k j=1...200 are the top-down weights
  corresponding to node k in layer F_2 of the network.
Input: Host-ID of the host that requests a URL.
Output: The array of prefetched_URLs[], which contains a list of URLs that are to be
  prefetched for the Host-ID.
Initialize count = 0
Step 1:  for n clusters formed using ART1-based clustering algorithm
         begin
Step 2:    if (Host-ID is a member of cluster C_k)
           begin
Step 3:      for j = 1 to 200 do   //size of prototype vector T_k representing cluster C_k
             begin
Step 4:        if (t_kj = 1)        //where t_kj is the j^th element of T_k
                     begin
                       prefetched_URLs [count]=URLi
                       count = count + 1
                          end-if- Step 4
             end-for- Step 3
           end-if- Step 2
         end-for- Step 1
Step 5: return prefetched_URLs []
Step 6: End ART1-Based Prefetching ()
```

Figure 3. ART1-based prefetching scheme.

value. The F_1 layer presents the pattern vector $\mathbf{P}_H$, which represents the access pattern of each host H. The F_2 layer consists of a variable number of nodes corresponding to the number of clusters. By clustering organizationally related Web users in binary vector space, our algorithm improves performance over clustering in larger quantized vector spaces.[1]

The "Web Data Mining and User Clustering" sidebar describes related clustering research.

Prefetching scheme

Prefetching is a technique, like Web caching, to reduce user-perceived latency. Most prefetching techniques predict requests for a single user. Such approaches can easily overload the network when the number of users is large. Our prefetching scheme uses the ART1-based algorithm for clustering large communities of organizationally related users.[1] When the algorithm stabilizes, the prototype vector that forms for each cluster gives a generalized representation of the URLs most frequently requested by all members (hosts) of that cluster.

Whenever a host connects to the server or a proxy, our prefetching strategy returns the URLs in the prototype vector for the cluster to which the host belongs. An immediate advantage of our approach is better network resource utilization by prefetching for a user community rather than a single user.

Figure 3 lists the prefetching scheme.

COMPARING ART1 AND K-MEANS CLUSTERING

To evaluate our clustering algorithm, we compared its performance with that of the k-means statistical clustering algorithm. The k-means algorithm clusters N data points into k disjoint subsets S_j. The geometric centroid of the data points represents the prototype vector for each subset.

For the ART1-based algorithm, we measured the quality of clusters obtained by varying the vigilance parameter's value. The quality measure is a function of the average distance between clusters (*intercluster* distance) and the average distance between members of each cluster (*intracluster* distance). Figure 4 shows the increase in the number of clusters with increased vigilance parameter values ranging from 0.3 to 0.5.

Next, we computed the average intercluster and intracluster distances for the clusters formed by varying the vigilance parameter. Figure 5a illustrates the variations in these distances for parameter values between 0.3 and 0.5.

The k-means algorithm partitions a given data set into k clusters. For our comparison, we selected the same numbers of clusters corresponding to each variant in the ART1 results. Figure 5b shows the variations in the average intercluster and intracluster distances for the different values of k.

Figure 6a compares the variation in average intercluster distances for the two algorithms as the

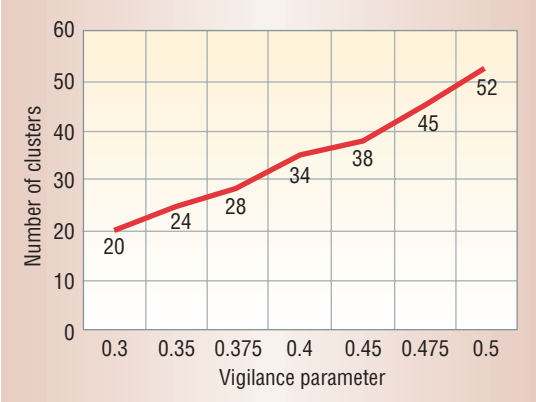

Figure 4. Evaluating cluster quality. Increasing the vigilance parameter of the ART1-based clustering technique increases the number of clusters.

number of clusters increases. Both algorithms show distances varying at a steady rate, indicating little difference in their performance in terms of intercluster distance.

In Figure 6b, however, the average intracluster distances using the k-means algorithm decrease from 24.20 to 12.67 as the number of clusters increases, while the intracluster distances using the ART1-based algorithm increase only slightly, from 18.04 to 20.45. The ART1-based results are quite uniform compared to the k-means algo-

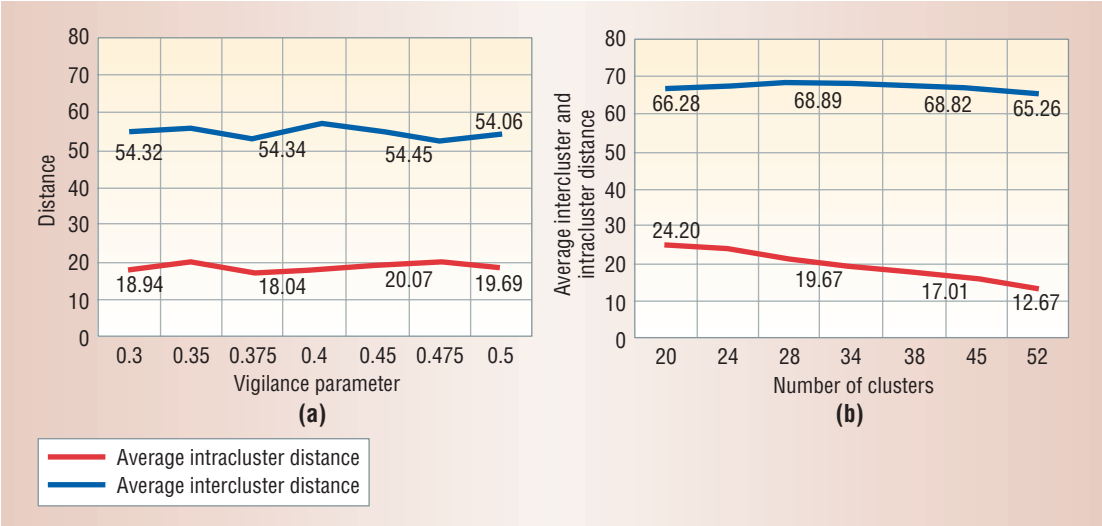

Figure 5. Variations in average intercluster and intracluster distances. (a) The value obtained by varying the vigilance parameter between 0.3 and 0.5 for the ART1-based algorithm and (b) the corresponding number of clusters obtained using the k-means algorithm to partition the data.

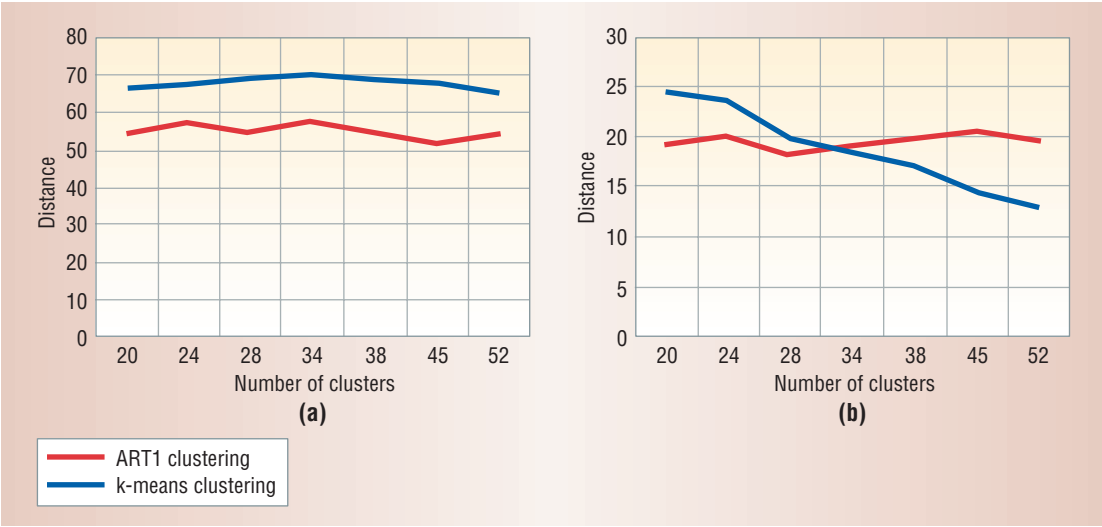

Figure 6. Comparison of ART1 and k-means algorithms. (a) Variations in the average intercluster distance and (b) variations in the average intracluster distance. The ART1 results are quite uniform compared to the k-means algorithm.

Table 1. Results of ART1-based prefetching scheme. Each row represents a host cluster, whose members are listed in column one and for whom our algorithm prefetched the number of URLs listed in column 2. "Requested URLs" lists each host and the number of URLs during the period for which we made predictions.

Members	Number of URLs Prefetched	Requested URLs		Hits	Accuracy (%)
0, 2, 5, 7, 8, 13, 14, 58	45	0	173	44	97.778
		2	189	43	95.556
		5	150	42	93.330
		7	160	44	97.778
		8	192	42	93.330
		13	200	43	95.556
		14	200	44	97.778
		58	122	39	86.666
3, 4, 10, 11, 18	65	3	200	60	92.300
		4	200	56	86.150
		10	168	56	86.150
		11	200	57	87.690
		18	13	8	12.300
6, 12, 15, 16	39	6	168	37	94.871
		12	181	37	94.871
		15	–	–	–
		16	126	32	82.050
1, 9, 67	38	1	–	–	–
		9	20	35	92.120
		67	13	34	89.470

rithm. The uniformity indicates clustering stability, which is an important attribute of high-quality clusters.

PREFETCHING RESULTS

We used two parameters to assess our prefetching scheme's performance:

- *hits*, the number of URLs requested from the prefetched URLs; and
- *accuracy*, the ratio of hits to the number of prefetched URLs.

To verify our prefetching scheme's accuracy, we prefetched the URLs for each host and compared predicted URLs with the NASA access logs over the next 13 days.

Table 1 presents the results obtained by assigning a value of 0.38 to the ART1-based algorithm's vigilance parameter. The prediction accuracy ranges from 82.05 to 97.78 percent. A deviation occurred in three cases, in which the hosts had not requested any of the prefetched URLs. Excluding these three cases, the average prediction accuracy of our scheme is 92.3 percent.

These results are very high. By comparison, Li Fan and colleagues[5] achieved prediction accuracies ranging from 40 to 73 percent with a prefetching approach that uses a prediction-by-partial-matching algorithm to reduce Web latency. Evangelos Markatos and Catherine Chronaki[6] used a top-10 prefetching approach that accurately predicted 60 percent of future requests. Ton Sau Loon and Vaduvur Bharghavan[7] achieved 50 to 75 percent accuracies in an approach based on user profiles. The profiles characterized each user's access patterns in a weighted, directed graph in which the nodes represented URLs, the edges represented access paths, a node's weight represented the frequency of access to URLs, and an edge's weight represented the access frequency of one URL after another.

Algorithms that group organizationally related users can extract valuable domain access information, and the prefetching application of our ART1-based neural network for user clustering based on HTTP request patterns shows its usefulness. However, neural networks such as ART1 can only capture—not utilize—the inherent self-similar properties of the World Wide Web. We are currently developing adaptive prediction systems that use statistical, neural, and Bayesian learning paradigms that can capitalize on the self-similarity of Web requests. ■

Acknowledgments

S.S. Iyengar is supported in part by National Science Foundation grants ITR-0312632 and IIS-0329738.

References

1. S.K. Rangarajan, *Unsupervised Learning Techniques for Web Domain Clustering and Its Application for Prefetching*, master's thesis, Louisiana Tech Univ., 2002.
2. B. Moore, "ART1 and Pattern Clustering," *Proc. 1988 Connectionist Models Summer School*, Morgan Kaufmann, 1998, pp. 174-185.
3. L.G. Heins and D.R. Tauritz, "Adaptive Resonance Theory (ART): An Introduction," internal report 95-35, Dept. of Computer Science, Leiden University, 1995; www.cs.brandeis.edu/~cs113/docs/other/heins_tauritz.pdf.
4. G.A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics and Image Processing*, vol. 37, 1987, pp. 54-115.
5. L. Fan, P. Cao, and Q. Jacobson, "Web Prefetching between Low-Bandwidth Clients and Proxies: Potential and Performance," *Proc. Joint Int'l Conf. Measurement and Modeling of Computer Systems* (Sigmetrics 99), ACM Press, 1999, pp. 178-187.
6. E.P. Markatos and C.E. Chronaki, "A Top-10 Approach to Prefetching on the Web," *Proc. 8th Ann. Conf. Internet Society* (INET 98), Internet Society, 1998; www.isoc.org/inet98/proceedings/.
7. T.S. Loon and V. Bharghavan, "Alleviating the Latency and Bandwidth problems in WWW Browsing," *Proc. USENIX Symp. Internet Technologies and Systems* (USITS 97), Usenix, 1997; www.usenix.org/publications/library/proceedings/usits97/tong.html.

Santosh K. Rangarajan *is working with the Jacobs Engineering Group to develop corporate Web infrastructure applications. His research interests include the design and development of database-driven applications and the administration of application servers. Rangarajan received an MS in computer science from Louisiana Tech University. Contact him at santosh_kumarr@hotmail.com.*

Vir V. Phoha *is an associate professor of computer science at Louisiana Tech University in Ruston. His research interests include Web caching, Web mining, network and Internet security, intelligent networks, and nonlinear systems. Phoha received a PhD in computer science from Texas Tech University. He is a senior member of the IEEE and a member of the ACM. Contact him at phoha@latech.edu.*

Kiran S. Balagani *is a graduate student in computer science at Louisiana Tech University and a research assistant in its Anomaly Detection and Mitigation Laboratory. His research interests include artificial neural networks, Web mining, computer and network security, and steganography. Balagani received a BS in computer science from Dayananda Sagar College of Engineering, Bangalore, India. Contact him at kiranbalagani@hotmail.com.*

Rastko R. Selmic *is an assistant professor of electrical engineering at Louisiana Tech University. His research interests include nonlinear control, adaptive control, neural networks, and backlash compensation using intelligent control tools. Selmic received a PhD in electrical engineering from the University of Texas, Arlington. Contact him at rselmic@latech.edu.*

S.S. Iyengar *is department chair and the Roy Paul Daniels Professor of Computer Science at Louisiana State University in Baton Rouge. His research interests include high-performance parallel and distributed algorithms and data structures for image processing and pattern recognition. Iyengar received a PhD in engineering from Mississippi State University. He is a Fellow of the IEEE and a member of the ACM and the American Association for the Advancement of Science. Contact him at iyengar@bit.csc.lsu.edu.*