A Pattern Reordering Approach Based on Ambiguity Detection for Online Category Learning

Eric Granger, Yvon Savaria, *Senior Member*, *IEEE*, and Pierre Lavoie

Abstract—Pattern reordering is proposed as an alternative to sequential and batch processing for online category learning. Upon detecting that the categorization of a new input pattern is ambiguous, the input is postponed for a predefined time, after which it is reexamined and categorized for good. This approach is shown to improve the categorization performance over purely sequential processing, while yielding a shorter input response time, or latency, than batch processing. In order to examine the response time of processing schemes, the latency of a typical implementation is derived and compared to lower bounds. Gaussian and softmax models are derived from reject option theory and are considered for detecting ambiguity and triggering pattern postponement. The average latency and Rand Adjusted clustering score of reordered, sequential, and batch processing are compared through computer simulation using two unsupervised competitive learning neural networks and a radar pulse data set.

Index Terms—Ambiguity, online category learning, partitional clustering, pattern recognition, reject option.

1 Introduction

A number of pattern recognition applications involve high throughput category learning from continuous streams of input patterns. In pattern recognition literature, partitional clustering techniques [2], [9] such as online versions of the k-means [27] and leader [21] algorithms, are proposed for online category learning. Other algorithms include online versions of adaptive vector quantization (AVQ) [15], [18] (e.g., generalized Lloyd [26] and Linde-Buzo-Gray (LBG) [25] algorithms) in communication theory, and unsupervised competitive learning (UCL) [19], [20], [24] (e.g., standard UCL networks [1], [28] and Adaptive Resonance Theory (ART) networks [4]) in neural network theory. These algorithms learn input patterns autonomously on the fly, some without prior knowledge of the number and characteristics of the categories.

A common trait of these algorithms is that they process inputs sequentially. When an input pattern is presented, it is compared to the prototype of every category. The category with the best matching prototype is assigned to the input and the prototype of this category adapts to novel characteristics of the input. Over a succession of inputs, category prototypes become tuned to different parts of the input space and implicitly define inter-category decision boundaries. Since a final decision is taken upon presentation of every input, only information from prior input patterns is available. In addition, the fact that prototypes are stored rather than prior input patterns implies that learning—the consequence of each decision—is irreversible. Information is therefore lost in the process.

Limitations of such *sequential processing* are obvious, particularly if the structure of input data clusters tends to be scattered and/or overlapping. For instance, if upon its presentation, an input pattern lies close to a decision boundary separating two or more categories, the clusterer is forced to make a final decision, despite the ambiguity. Irreversible learning for such decisions yields category structures and decision boundaries that vary significantly according to the pattern presentation order.

The quality of results is generally improved with *batch processing*. This consists in accumulating one batch of patterns from the input stream, while the clusterer converges on a previously-accumulated batch of patterns. Batch processing can diminish or eliminate sensitivity to the input presentation order by allowing the clusterer to recover from early miss-assignments. However, it entails some delay for the accumulation of patterns into batches, and may require data buffering if the computational effort is variable.

In this paper, an alternative called *reordered processing* is proposed for online category learning. As a clusterer assigns categories to input patterns, it is granted the ability to postpone, or delay, category assignment and learning when it detects an ambiguity. Delayed patterns are queued, and their categorization is deferred for some fixed time. The overall effect is a modification to the order in which inputs are categorized. Aside from allowing to control the maximum response time, pattern postponement offers the opportunity to circumvent learning for ambiguous decisions.

The rest of this paper is organized as follows: In order to compare the response time of clustering systems, lower bounds on the latency required for online category learning are developed for sequential, batch, and reordered processing in Section 2. Practical issues are also discussed, and architectures are proposed. In Section 3, elements of reject option theory are briefly reviewed and developed for the detection of ambiguous decisions, as required by reordered processing. Finally, the experimental methodology, and proof-of-concept computer simulations are presented and discussed in Section 4. It is shown that the proportion of patterns that are declared ambiguous is a good indicator of poor clustering quality; and that a modification of the order in which input patterns are processed can indeed enhance clustering quality.

2 Latency in Online Category Learning

Online category learning of a continuous stream of patterns is performed by a clusterer (e.g., an online k-means algorithm) which is exploited through one of several data processing schemes. Let input patterns to be categorized arrive one at a time from some source that provides them at a rate that need not be constant. The *latency* L of the category learning is defined as the number of input patterns necessary before a final decision (category assignment) $\mathbf{y}(\mathbf{a})$ can be made for input $\mathbf{a}$.

The basic approach to learning a continuous data stream is through *sequential processing*. Upon observation of each input pattern $\mathbf{a}$, a category is assigned to $\mathbf{a}$ without waiting for subsequent patterns. *Batch processing* consists in categorizing the input patterns $\{\mathbf{a}\}$ in fixed-size batches of k patterns. Once k successive patterns have been collected and stored, the following patterns are buffered while the k patterns are being learned. Batches of k patterns are learned iteratively, over several epochs, until convergence is attained, that is, until the prototypes remain constant for two epochs (complete presentations of data in the

- *E. Granger is with Integrated Systems Group, Mitel Networks, 98 Sweetland Ave., Ottawa, Ontario K1N 7T8, Canada. E-mail: eric.granger@rogers.com.*
- *Y. Savaria is with the Department of Electrical and Computer Engineering, École Polytechnique de Montréal, PO Box 6079, Station centre-ville, Montreal, Quebec H3C 3A7, Canada. E-mail: savaria@vlsi.polymtl.ca.*
- *P. Lavoie is with the Defence Research Establishment Ottawa, Department of National Defence, 3701 Carling Ave., Ottawa, Ontario K1A 0Z4, Canada. E-mail: pierre.lavoie@dreo.dnd.ca.*

Manuscript received 12 Feb. 2001; revised 13 Nov. 2001; accepted 16 July 2002.
Recommended for acceptance by C. Brodley.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 113609.

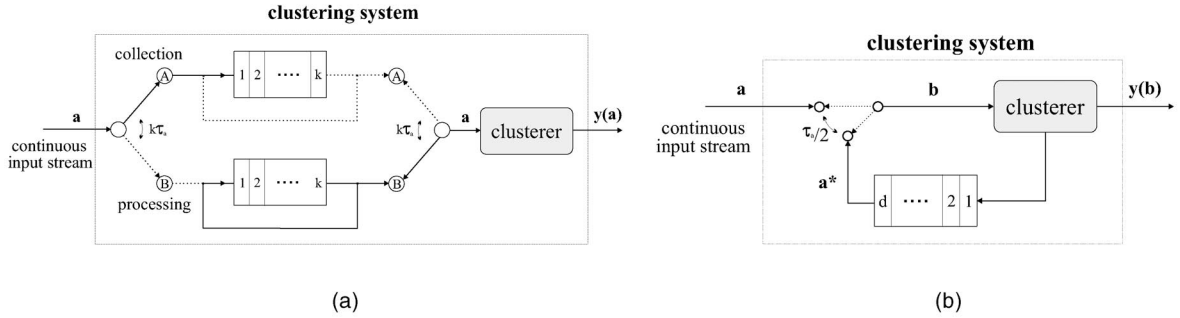

Fig. 1. Clustering system architectures. (a) Batch processing architecture and (b) reordered processing architecture.

batch). Categories are assigned once convergence has been detected, i.e., after the last one of these epochs. *Reordered processing* consists in postponing the learning of input patterns for which category assignment is deemed ambiguous. Each postponed pattern is queued, and following a fixed latency of d patterns, is categorized once and for all.

Assuming an infinitely fast clusterer, the minimum latency $L_{\min}$ for all three schemes must satisfy:

$$L_{\min} \geq 0. \tag{1}$$

This lower bound corresponds to $\mathbf{a}$ being the last pattern in a batch with batch processing, or $\mathbf{a}$ not being postponed with reordered processing. The maximum latency $L_{\max}$ must satisfy:

$$L_{\max} \geq \begin{cases} 0; & \text{sequential processing,} \\ k-1; & \text{batch processing,} \\ d; & \text{reordered processing,} \end{cases} \tag{2}$$

where k is the number of patterns per batch, and d is the user-defined queue size. This bound corresponds to a being the first pattern in a batch with batch processing, or $\mathbf{a}$ being postponed with reordered processing. Last, it is straightforward to show that the average latency $\overline{L}$ must satisfy:

$$\overline{L} \geq \begin{cases} 0; & \text{sequential processing,} \\ \frac{1}{k}\sum_{i=1}^{k}(k-i); & \text{batch processing,} \\ p_r(\mathbf{a}) \cdot d; & \text{reordered processing,} \end{cases} \tag{3}$$

where $p_r(\mathbf{a})$ is the pattern rejection or postponement rate. This rate depends on the data and the rejection criterion. Reordered processing constitutes a compromise between sequential processing and batch processing. Indeed if $d = (k-1)$ and $p_r(a) = 50\%$, then the lower bounds on $L_{\max}$ and $\overline{L}$ are equal for batch and reordered processing. If, on the other hand, $d = 0$, then reordered processing reduces to sequential processing.

To characterize the latency of practical clustering systems, it is convenient to assume that inputs arrive at a regular interval τ_a. With batch processing, the number of batch epochs, δ, required to attain the state of convergence varies from one batch to the next according to the queue length k, and to the structure of the data in the batch. The computational effort is therefore variable and somewhat unpredictable. Detection of convergence requires at least one whole batch epoch of overhead and, thus, $\delta \geq 2$. To circumvent this delay, one can bound the number of batch epochs to a constant value across all batches, $\delta \geq 2$. A final decision $\mathbf{y}(\mathbf{a})$ can then be produced immediately after processing of $\mathbf{a}$ as part of the last epoch.

One possible batch processing architecture is shown in Fig. 1a. It consists of two identical fixed-length queues of k registers that operate concurrently, and a clusterer that can process each pattern within a fixed time τ_c. Each queue alternates between a collection (A) and a processing (B) mode. While one of the queues temporarily buffers k incoming patterns from the input stream (A), the other queue stores a batch of k patterns being learned by the clusterer (B). The processing rate must satisfy $\tau_c \leq \tau_a/\delta$, which imposes a processing rate constraint on the clusterer that is proportional to δ, and prevents overflows of the queue operating in collection mode. Once a queue is switched to the collection mode (A), a pattern $\mathbf{a}$ from the input stream is stored every τ_a seconds inside its registers. After $k\tau_a$ seconds, the registers are full, and the queue is switched into processing mode (B). In this mode, patterns are shifted right through the clusterer every τ_c seconds. A feedback loop allows repeated presentations. After δ epochs, the registers are reset prior to switching back to collection mode. The latency of the batch processing architecture of Fig. 1a is the sum of the delays incurred in both modes. In the best case, when $\mathbf{a}$ is the last of a batch,

$$L_{\min} = k, \tag{4}$$

and in the worst case, when $\mathbf{a}$ is the first of a batch,

$$L_{\max} = (k-1) + \left\lceil \frac{(\delta-1)k+1}{\delta} \right\rceil. \tag{5}$$

The average latency of this architecture is:

$$\overline{L} = \frac{1}{k}\sum_{i=1}^{k}\left\{ (k-i) + \left\lceil \frac{(\delta-1)k+i}{\delta} \right\rceil \right\}. \tag{6}$$

It is easily verified that (4), (5), and (6) satisfy, but do not meet the lower bounds (1), (2), and (3).

With reordered processing, a pattern having been postponed by d patterns is given priority over a pattern from the input stream. This ensures a fixed worst-case response time of d patterns, but leads to the accumulation of up to d incoming patterns. Also, a clusterer embedded within this system requires additional circuitry to detect ambiguity.

One possible reordered processing architecture is shown in Fig. 1b. It consists of a fixed-length queue composed of d registers, and a clusterer capable of detecting ambiguous category assignments. If an input pattern $\mathbf{a}$ is deemed ambiguous, it is diverted towards the queue and labeled $\mathbf{a}^*$. Shifting this pattern through the queue is equivalent to a fixed delay that changes the presentation order. Each pattern $\mathbf{b}$ that is processed by the clusterer is either a new input pattern, $\mathbf{b} = \mathbf{a}$, or a previously rejected and, therefore, delayed pattern, $\mathbf{b} = \mathbf{a}^*$. A simple way to schedule processing is to assume that the clusterer's processing time is partitioned into interleaved time slices: odd slices reserved for input patterns $\{\mathbf{a}\}$ and even slices reserved for previously-rejected patterns $\{\mathbf{a}^*\}$. To alternate between the two sources, the augmented clusterer must

be able to process a pattern $\mathbf{b}$ within a fixed time of τ_c that satisfies $\tau_c \leq \tau_a/2$. This speed constraint is similar to that of batch processing with $\delta = 2$. With this architecture, the lower bounds (1), (2), and (3) are met.

3 Ambiguity in Competitive Learning Assignments

With reordered processing, ambiguity is employed as a criterion for postponing pattern categorization. In statistical pattern recognition, the *reject option* [6], [7], [13] provides a framework for detecting ambiguous classifications. The Bayes decision procedure assigns one-of-N possible classes to input $\mathbf{a}$ using the maximum a posteriori probability decision rule, $J = \arg\max\{p(j \mid \mathbf{a}) : j = 1, 2, \ldots, N\}$, where $0 \leq p(j \mid \mathbf{a}) \leq 1$ and $\sum_{j=1}^{N} p(j \mid \mathbf{a}) = 1$. The *a posteriori* probability $p(j \mid \mathbf{a})$ that class j generated input $\mathbf{a}$ is computed according to the Bayes theorem [11], [14].

The degree of ambiguity regarding this decision rule can be measured in terms of the conditional error given input $\mathbf{a}$, $r_J(\mathbf{a}) = 1 - \max\{p(j \mid \mathbf{a}) : j = 1, 2, \ldots, N\}$ [6], [7], [10], [13]. Assignment of class J to input $\mathbf{a}$ is defined as *ambiguous* if $r_J(\mathbf{a})$ is greater than or equal to a *rejection threshold* γ, $\gamma \in \left(0, \frac{N-1}{N}\right]$. This criterion can be rewritten:

$$\left(\frac{1-\gamma}{\gamma}\right) \geq \frac{P_J \; p(\mathbf{a} \mid J)}{\sum_{j=1, j\neq J}^{N} P_j \; p(\mathbf{a} \mid j)}, \tag{7}$$

where P_j is the a priori probability of class j, with $0 \leq P_j \leq 1$ and $\sum_{j=1}^{N} P_j = 1$, and $p(\mathbf{a} \mid j)$ is the conditional probability density function (p.d.f.) of the input $\mathbf{a}$ given class j. Equation (7) defines a region in the input space where class assignments are considered to be ambiguous. The size of this region grows as γ is decreased. For a given γ, the size of the region also depends on the shape of $r_J(\mathbf{a})$ at the decision boundary of class J, and, hence, on the class distribution in the input space.

In practice, the probabilities required to implement the reject option are often unknown. For online clustering applications, these probabilities must be estimated from the clusterer's response to input patterns. A clusterer may be subjected to different environments, with more or less prior information on the underlying data structure. Equation (7) is now developed for two possible environments.

In the *Gaussian rejection model*, the data are assumed to be generated by sources with the same Gaussian noise and with equal a priori probabilities P_j, and all variables are assumed to be statistically independent and to have equal variance σ^2. Then, data clusters are modeled explicitly as hyperspherical normal distributions centered at mean vectors $\{\mu_j\}$. For this model, (7) becomes

$$\left(\frac{1-\gamma}{\gamma}\right) \geq \frac{\exp\left\{-\frac{||a-\mu_J||^2}{2\sigma^2}\right\}}{\sum_{j=1, j\neq J}^{N} \exp\left\{-\frac{||\mathbf{a}-\mu_j||^2}{2\sigma^2}\right\}}. \tag{8}$$

When implemented as part of a clusterer, μ_j is equal to the prototype vector of category j. The variance σ^2 is required, either from prior knowledge or from online estimation. The Euclidean distance $||\mathbf{a} - \mu_j||$ is a core component of the prototype matching function commonly used by clusterers when prototypes represent mean patterns.

In the *softmax rejection model*, no prior assumption is made regarding the underlying data structure, and data clusters are modeled implicitly. During online category learning, the prototype matching function provides the response strength for each category with respect to $\mathbf{a}$. The match strength ϕ_j between input and prototype can be interpreted as an estimate of the a posteriori probability $p(j \mid \mathbf{a})$. To ensure that the ϕ_j values are valid probabilities (i.e., sum up to 1 and range from 0 to 1), the *softmax activation function* [3], $y_j = \exp\{\phi_j\} / \sum_{k=1}^{N} \exp\{\phi_k\}$, from neural network literature can be applied. Assuming that y_j can be used as an estimate of $p(j \mid \mathbf{a})$, the rejection rule $r_J(\mathbf{a}) \cong (1 - y_J) \geq \gamma$ can then be expressed in a form similar to that of (7):

$$\left(\frac{1-\gamma}{\gamma}\right) \geq \frac{\exp\{\phi_J\}}{\sum_{j=1, j\neq J}^{N} \exp\{\phi_j\}}. \tag{9}$$

The reader is referred to [17] for details of the mathematical derivation of (7)-(9).

4 Simulation Results and Discussion

4.1 Experimental Methodology

Unsupervised competitive learning (UCL) [19], [20], [24], [28] neural networks were used as clusterers for computer simulation of sequential, batch, and reordered processing. In order to compare the performance of the clustering systems, simulations were repeated over several independent trials. Prior to each trial, input patterns were shuffled into a random presentation order. The patterns were then learned by the clustering system under test. After every trial, clustering quality was measured.

The UCL neural networks selected for computer simulations are ART2A-E [12] and fuzzy ART [5]. ART2A-E is well suited for Gaussian type environments, where clusters are explicitly modeled as mean vectors. Fuzzy ART is appropriate for environments where clusters are modeled implicitly. Programs emulating these neural networks were written in the Matlab language for all three data processing schemes. For reordered processing, each network was granted the capacity to detect ambiguity. The Gaussian rejection model ((8)) was used in the ART2A-E network, whereas the softmax rejection model ((9)) was used in the fuzzy ART network. In the second case, $\phi_j = T_j$ [5], for $j = 1, 2, \ldots, N$, such that the vigilance test is passed, was substituted in (9).

The data set employed for computer simulations was collected in the field by the Defence Research Establishment Ottawa. It consists of radar pulses from 12 shipborne navigation radars. Fifty pulses were collected from each radar, with the exception of radars #7 (100 pulses) and #8 (200 pulses). The pulses were preprocessed to yield 800 patterns with 15 real-valued features. Data clusters in this set cannot be described by the same statistics—some are not Gaussian distributed and they sometimes overlap one another [16]. This Radar data set is representative of a continuous pulse stream intercepted by a radar electronic support measures (ESM) system. Within these systems, automatic sorting of pulses according to emitter is very challenging due to the density and complexity of signals encountered in modern environments [8]. For instance, at a given time, the intercepted pulses may be difficult to sort because clusters of emitter data often overlap for more than one parameter. Performance must often be compromised to allow for timely detection and avoidance of threats.

The Rand Adjusted similarity measure [22] was selected to assess clustering quality. This type of measure is known in pattern recognition literature as an external criterion index, and is used for evaluating the capacity to recover the true cluster structure [2], [9], [22]. A partition of n patterns into K groups defines a *clustering*. This can be represented as a set $A = \{a_1, a_2, \ldots, a_n\}$, where $a_h \in \{1, 2, \ldots, K\}$ is the category label assigned to pattern h. In our

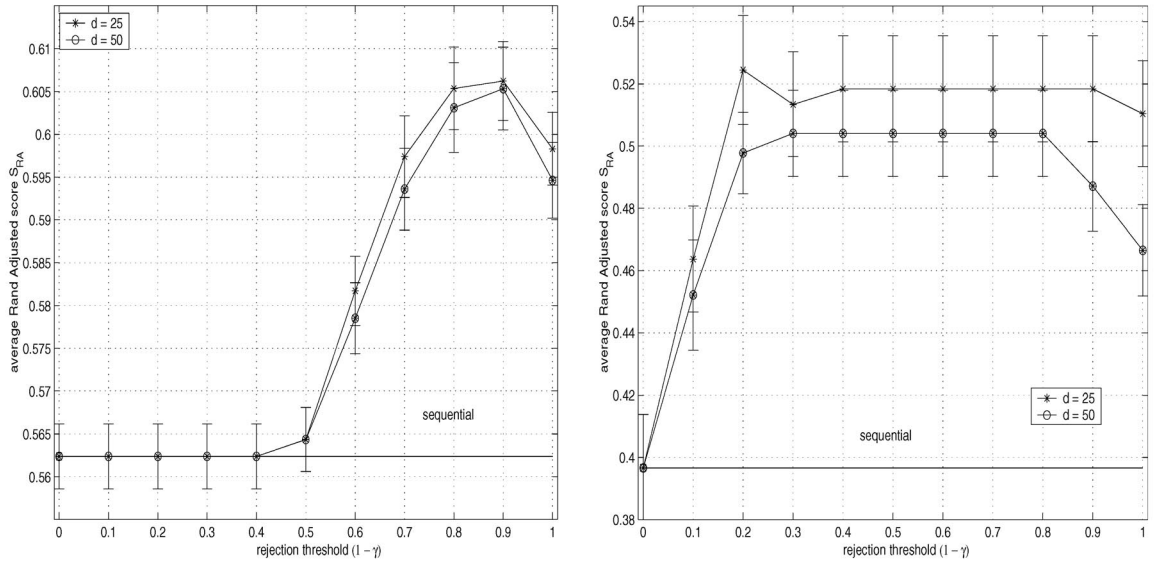

Fig. 2. Average Rand Adjusted scores $S_{RA}(A, R)$ versus rejection threshold γ for simulations with the Radar data set. Error bars show standard error. (a) ART2A-E on Radar data. (b) Fuzzy ART on Radar data.

context, the correct classification results are known for the data sets used, and their patterns are all accompanied by category labels. These labels provide a reference clustering, R, with which a clustering produced by computer simulation, A, may be compared. The Rand Adjusted measure now represents a *score*, $S_{RA}(A, R)$, that describes the quality of the clustering produced by the network. $S_{RA}(A, R)$ ranges from 0 to 1, where 0 denotes maximum dissimilarity (worst), and 1 denotes equivalence (best) [16].

4.2 Correlation of Ambiguity with Clustering Quality

The clustering quality achieved can be examined as a function of the degree of ambiguity observed. Patterns from the Radar data set were categorized by the ART2A-E and fuzzy ART networks using reordered processing. Network parameters were fixed to values that produce a high Rand Adjusted score $S_{RA}(A, R)$ using fast learning and sequential processing. For a same randomly-selected presentation order, the rejection threshold γ was varied from trial to trial. Rejected patterns were counted, yet processed without postponement. After each trial, the score $S_{RA}(A, R)$, and the empirical rejection rate $\hat{p}_r$ were stored. The empirical rejection rate $\hat{p}_r$ is the ratio of the number of rejected patterns to the total number of patterns (the size of the data set). The linear correlation between score and respective rate was computed from a series of 1,000 independent presentation orders, with γ ranging from 0 to 1.

Results show that both clustering systems display a significant negative correlation between $\hat{p}_r$ and $S_{RA}(A, R)$, albeit in different ranges of values for the threshold γ. For a given value of γ, an increase in ambiguous category assignments corresponds to a decline of the clustering score. For example, when using fuzzy ART and softmax rejection, the peak negative correlation coefficient of about -0.65 is obtained when $\gamma = 0.95$, which corresponds to $\hat{p}_r \cong 14\%$.

4.3 Clustering Quality Obtained Using Reordered Processing

Having shown that the Gaussian and softmax rejection models detect ambiguous category assignments and, that, these assignments lead to poor clustering performance, the effect of delaying ambiguous assignments is now examined. ART2A-E and fuzzy ART parameters were fixed to provide high Rand Adjusted scores $S_{RA}(R, A)$ with fast learning and sequential processing. Rejected patterns were delayed by d patterns. Two different delay values, $d = 25$ and 50, were tried. After each trial, $S_{RA}(A, R)$ was stored. Average $S_{RA}(A, R)$ values were obtained from 20 independent presentation orders, with γ ranging from 0 to 1.

Fig. 2 shows the average $S_{RA}(A, R)$ as a function of γ. In both cases, reordered processing increases the clustering score over sequential processing alone. For example, if patterns are presented to the clustering system with fuzzy ART and $d = 25$, then reordered processing improves the score $S_{RA}(A, R)$ by about 30 percent over a wide range of γ values. In the example given above, scores of about $S_{RA}(A, R) = 0.52$ are obtained for $\gamma \in [0.2, 0.9]$, corresponding to $\hat{p}_r \cong 45\%$. For $0.8 \leq \gamma < 1.0$, the performance approaches that of sequential processing since $\hat{p}_r \to 0\%$. For γ values close to 0, the performance also declines because of the large number of rejections. Fuzzy ART using softmax rejection shows strong negative correlation between $\hat{p}_r$ and $S_{RA}(A, R)$ across a wide range of γ values, and appears to benefit from reordered processing more than ART2A-E.

4.4 Clustering Quality and Latency

Sequential, batch, and reordered processing are now compared in terms of clustering quality and latency. For a same randomly-selected presentation order, the queue lengths k and d were varied between five patterns and half the number of patterns in the data set, over successive trials. This range guarantees a minimum of two batches of k patterns for batch processing. With the batch architecture, the neural networks were left to converge for each batch of k patterns, until prototype weights were identical for two consecutive epochs. Network parameters were fixed a priori to provide high Rand Adjusted scores $S_{RA}(A, R)$ with batch processing when k is one half the data set size. With the reordered architecture, the network parameters were set a priori to provide high scores $S_{RA}(A, R)$ with sequential processing. The value of γ was set equal to 0.35 for ART2A-E and 0.05 for fuzzy ART. After

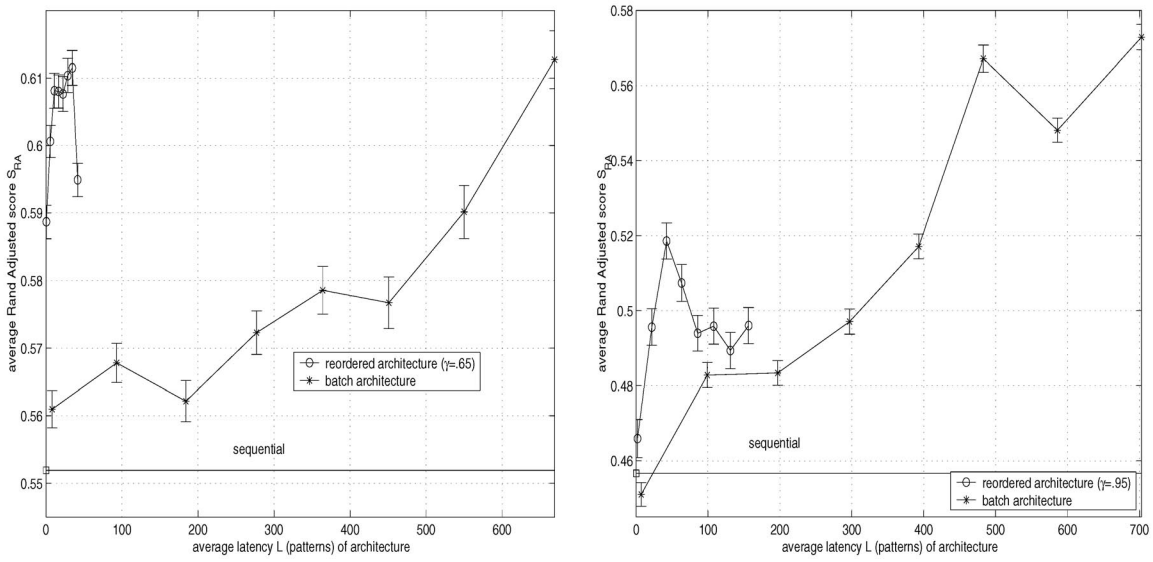

Fig. 3. Average Rand Adjusted scores $S_{RA}(A,R)$ versus average latency $\overline{L}$ of the batch and reordered processing architectures for simulations with the Radar data set. Error bars show the standard error. The standard error for values of $\hat{p}_r$ always ranges from 0 percent to 2 percent. The batch architecture requires between two and five epochs for convergence on each batch of patterns. (a) AT2A-E on Radar data and (b) fuzzy ART on Radar data.

each trial, the score $S_{RA}(A,R)$ and the average latency $\overline{L}$ of the processing architectures were stored. The empirical rejection rate, $\hat{p}_r$, was used as a fixed estimate of the variable rejection rate $p_r(\mathbf{a})$, and was substituted into (3). Simulations were repeated 20 times for every k and d value in order to yield representative results.

The average scores $S_{RA}(A,R)$ as a function of the average latency $\overline{L}$ for batch and reordered processing are shown in Fig. 3. Both achieve significantly higher scores than sequential processing. However, reordered processing requires a much lower average latency than batch processing. For instance, using fuzzy ART and reordered processing with $\gamma = 0.05$ yields a score of $S_{RA}(A,R) \cong 0.52$ for $\hat{p}_r \cong 32\%$, $d = 118$ and an average latency of $\overline{L} \cong 40$ patterns. By comparison, batch processing yields a comparable level of performance for $k = 231$ and an average latency of $\overline{L} \cong 400$ patterns, 10 times that of reordered processing. Results also reveal that the peak performance obtained with reordered processing often occurs for relatively small values of d, indicating that it is suitable for high speed applications. Beyond this peak, increasing d does not necessarily yield a higher score $S_{RA}(A,R)$. This may be due to our emulation of an infinite data stream with a fixed-size data set. Indeed, if an input $\mathbf{a}$ among the last d patterns of a data set is rejected, then it is postponed to the end of the data set, that is, for less than d patterns. The reader is referred to [17] for simulation results obtained on Gaussian distributed data and on the standard Iris data set.

5 Conclusion

A clustering system applied to online category learning would traditionally consist of a clusterer that processes input patterns sequentially. Despite the fast response time obtained when using this data processing scheme, the quality and consistency of the results remain an issue. By contrast, batch processing yields higher clustering quality, but incurs longer delays. In this paper, an alternate approach has been proposed. It consists in postponing for a fixed time the learning of patterns for which category assignments are ambiguous. The reject option from statistical pattern recognition literature has been applied to the detection of ambiguous category assignments. Reordered processing alters the original pattern presentation order and, therefore, introduces latency in the system. This latency has been defined and used to compare sequential, batch, and reordered processing.

Computer simulations performed by presenting patterns from a radar pulse data set to ART2A-E and fuzzy ART neural networks with sequential, batch and reordered processing show that 1) the number of category assignments detected as ambiguous is correlated with poor clustering quality, 2) reordered processing improves clustering quality over sequential processing, and 3) it offers an attractive alternative to batch processing for trading off clustering quality versus response time.

Acknowledgments

This research was supported in part by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche du Québec and the Natural Sciences and Engineering Research Council of Canada. This research was carried out while E. Granger was a scientist at the Defence Research Establishment Ottawa.

References

[1] S.C. Ahalt, A.K. Krishnamurthy, P. Chen, and D.E. Melton, "Competitive Learning Algorithms for Vector Quantization," *Neural Networks,* vol. 3, pp. 277-290, 1990.

[2] M.R. Anderberg, *Cluster Analysis for Applications*. London: Academic Press, 1973.

[3] J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," *Neurocomputing—Algorithms, Architectures and Applications,* F. Fogelman-Soulié and J. Hérault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989.

[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, pp. 54-115, 1987.

[5] G.A. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks,* vol. 4, no. 6, pp. 759-771, 1991.

[6] C.K. Chow, "An Optimum Character Recognition System Using Decision Functions," *IRE Trans. Electronic Computers,* vol. 6, pp. 247-254, 1957.

[7] C.K. Chow, "On Optimum Recognition Error and Reject Tradeoff," *IEEE Trans. Information Theory,* vol. 16, pp. 41-46, 1970.
[8] C.L. Davies and P. Hollands, "Automatic Processing for ESM," *Proc. IEE,* vol. 129, no. 3, pp. 164-171, June 1982.
[9] R.C. Dubes and A.K. Jain, *Algorithms for Clustering Data.* Prentice Hall, 1988.
[10] B. Dubuisson and M. Masson, "A Statistical Decison Rule with Incomplete Knowledge About Classes," *Pattern Recognition,* vol. 26, no. 1, pp. 155-165, 1993.
[11] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis,* John Wiley and Sons, 1973.
[12] T. Frank, K.-F. Kraiss, and T. Kuhlan, "Comparative Analysis of Fuzzy ART and ART-2A Network Clustering Performance," *IEEE Trans. Neural Networks,* vol. 9, no. 3, pp. 544-559, 1998.
[13] K. Fukunaga and D.L. Kessell, "Application of Optimal Error-Reject Functions," *IEEE Trans. Information Theory,* pp. 814-817, Nov. 1972.
[14] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* 2nd ed., Academic Press, 1990.
[15] A. Gersho, "On the Structure of Vector Quantizers," *IEEE Trans. Information Theory,* vol. 28, no. 2, pp. 157-166, 1982.
[16] E. Granger, Y. Savaria, P. Lavoie, and M.-A. Cantin, "A Comparison of Self-Organizing Neural Networks for Fast Clustering of Radar Pulses," *Signal Processing,* vol. 64, no. 3, pp. 249-269, 1998.
[17] E. Granger, Y. Savaria, and P. Lavoie, "A Pattern Reordering Approach Based on Ambiguity Detection for On-Line Category Learning," Département de génie électrique, École Polytechnique de Montréal, EPM/RT-01/02, 40 pages, Sept. 2001.
[18] R.M. Gray, "Vector Quantization," *IEEE ASSP Magazine,* vol. 1, no. 2, pp. 4-29, 1984.
[19] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Detectors," *Biological Cybernetics,* vol. 23, pp. 121-134, 1976.
[20] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding: II. Feedback, Oscillation, Olfaction, and Illusions," *Biological Cybernetics,* vol. 23, pp. 187-207, 1976.
[21] J.A. Hartigan, *Clustering Algorithms.* NY: Wiley, 1975.
[22] L. Hubert and P. Arabie, "Comparing Partitions," *J. Classification,* vol. 2, pp. 193-218, 1985.
[23] M.E. Hellman, "The Nearest Neighbor Classification Rule with Reject Option," *IEEE Trans. Systems Science and Cybernetics,* vol. 6, no. 3, pp. 179-185, 1970.
[24] T. Kohonen, *Self-Organization and Associative Memory.* Third ed., Springer-Verlag, 1989.
[25] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.,* vol. 28, no. 1, pp. 84-95, 1980.
[26] S.P. Lloyd, "Least-Square Quantization in PCM," *IEEE Trans. Information Theory,* vol. 28, no. 2, pp. 129-137, 1982.
[27] J. MacQueen, "Some Methods for Classification Analysis of Multivariate Observations," *Proc. Fifth Berkley Symp. Math. Statistics and Probability,* pp. 281-297, 1967.
[28] D.E. Rumelhart and D. Zipser, "Feature Discovery by Competitive Learning," *Cognitive Science,* vol. 9, pp. 75-112, 1985.