An Incremental Learning Algorithm with Confidence Estimation for Automated Identification of NDE Signals

Robi Polikar, *Member, IEEE*, Lalita Udpa, *Senior Member, IEEE*, Satish Udpa, *Fellow, IEEE*, and Vasant Honavar, *Member, IEEE*

***Abstract*—An incremental learning algorithm is introduced for learning new information from additional data that may later become available, after a classifier has already been trained using a previously available database. The proposed algorithm is capable of incrementally learning new information without forgetting previously acquired knowledge and without requiring access to the original database, even when new data include examples of previously unseen classes. Scenarios requiring such a learning algorithm are encountered often in nondestructive evaluation (NDE) in which large volumes of data are collected in batches over a period of time, and new defect types may become available in subsequent databases. The algorithm, named Learn++, takes advantage of synergistic generalization performance of an ensemble of classifiers in which each classifier is trained with a strategically chosen subset of the training databases that subsequently become available. The ensemble of classifiers then is combined through a weighted majority voting procedure. Learn++ is independent of the specific classifier(s) comprising the ensemble, and hence may be used with any supervised learning algorithm. The voting procedure also allows Learn++ to estimate the confidence in its own decision. We present the algorithm and its promising results on two separate ultrasonic weld inspection applications.**

I. Introduction

An increasing number of nondestructive evaluation (NDE) applications resort to pattern recognition-based automated-signal classification (ASC) systems for distinguishing signals generated by potentially harmful defects from those generated by benign discontinuities. The ASC systems are particularly useful in:

- accurate, consistent, and objective interpretation of ultrasonic, eddy current, magnetic flux leakage, acoustic emission, thermal or a variety of other NDE signals;
- applications calling for analysis of large volumes of data; and/or
- applications in which human factors may introduce significant errors.

Such NDE applications are numerous, including but are not limited to, defect identification in natural gas transmission pipelines [1], [2], aircraft engine and wheel components [3]–[5], nuclear power plant pipings and tubings [6], [7], artificial heart valves, highway bridge decks [8], optical components such as lenses of high-energy laser generators [9], or concrete sewer pipelines [10] just to name a few.

A rich collection of classification algorithms has been developed for a broad range of NDE applications. However, the success of all such algorithms depends heavily on the availability of an adequate and representative set of training examples, whose acquisition is often very expensive and time consuming. Consequently, it is not uncommon for the entire data to become available in small batches over a period of time. Furthermore, new defect types or other discontinuities may be discovered in subsequent data collection episodes. In such settings, it is necessary to update a previously trained classifier in an incremental fashion to accommodate new data (and new classes, if applicable) without compromising classification performance on preceding data. The ability of a classifier to learn under these constraints is known as incremental learning or cumulative learning. Formally, incremental learning assumes that the previously seen data are no longer available, and cumulative learning assumes that all data are cumulatively available. In general, however, the terms cumulative and incremental learning are often used interchangeably.

Scenarios requiring incremental learning arise often in NDE applications. For example, in nuclear power plants, ultrasonic and eddy current data are collected in batches from various tubings or pipings during different outage periods in which new types of defect or nondefect indications may be discovered in aging components in subsequent inspections. The ASC systems developed using previously collected databases then would become inadequate in successfully identifying new types of indications. Furthermore, even if no additional defect types are added to the database, certain applications may inherently need an ASC system capable of incremental learning. Gas transmission pipeline inspection is a good example. The network in the United States consists of over 2 million kilometers of

Manuscript received October 3, 2003; accepted April 23, 2004. This material is based upon work supported by the National Science Foundation under Grant No: ECS-0239090 for R. Polikar and Grant No: ITR-0219699 for V. Honavar.

R. Polikar is with the Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 (e-mail: polikar@rowan.edu).

L. Udpa and S. Udpa are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824.

V. Honavar is with the Department of Computer Science, Iowa State University, Ames, IA 50011.

gas pipelines, which are typically inspected by using magnetic flux leakage (MFL) techniques, generating 10 GB of data for every 100 km of pipeline [2]. The sheer volume of data generated in such an inspection inevitably requires an incremental learning algorithm, even if the entire data are available all at once. This is because current algorithms running on commercially available computers are simply not capable of analyzing such immense volumes of data at once due to memory and processor limitations.

Another issue that is of interest in using the ASC systems is the confidence of such systems in their own decisions. This issue is of particular interest to the NDE community [11] because ASC systems can make mistakes, by either missing an existing defect or incorrectly classifying a benign indication as a defect (false alarm). Both types of mistakes have dire consequences; missing defects can cause unpredicted and possibly catastrophic failure of the material, and a false alarm can cause unnecessary and premature part replacement, resulting in serious economic loss. An ASC system that can estimate its own confidence would be able to flag those cases in which the classification may be incorrect, so that such cases then can be further analyzed. Against this background, an algorithm that can:

- learn from new data without requiring access to previously used data,
- retain the formerly acquired knowledge,
- accommodate new classes, and
- estimate the confidence in its own classification

would be of significant interest in a number of NDE applications. In this paper, we present the Learn++ algorithm that satisfies these criteria by generating an ensemble of simple classifiers for each additional database, which are then combined using a weighted majority voting algorithm. An overview of incremental learning as well as ensemble approaches will be provided. Learn++ is then formally introduced along with suitable modifications and improvements for this work. We present the promising classification results and associated confidences estimated by Learn++ in incrementally learning ultrasonic weld inspection data for two different NDE applications.

Although the theoretical development of such an algorithm is more suitable, and therefore reserved for a journal on pattern recognition or knowledge management [12], the authors feel that this algorithm may be of specific interest to the audience of this journal as well as to the general NDE community. This is true in part because the algorithm was originally developed in response to the needs of two separate NDE problems on ultrasonic weld inspection for defect identification, but more importantly due to countless number of other related applications that may benefit from this algorithm.

II. Background

A. Incremental Learning

A learning algorithm is considered incremental if it can learn additional information from new data without having access to previously available data. This requires that the knowledge formerly acquired from previous data should be retained while new information is being learned, which raises the so-called stability-plasticity dilemma [13]; some information may have to be lost to learn new information, as learning new information will tend to overwrite formerly acquired knowledge. Thus, a completely stable classifier can preserve existing knowledge but cannot accommodate new information, but a completely plastic classifier can learn new information but cannot retain prior knowledge. The problem is further complicated when additional data introduce new classes. The challenge then is to design an algorithm that can acquire a maximum amount of new information with a minimum loss of prior knowledge by establishing a delicate balance between stability and plasticity.

The typical procedure followed in practice for learning new information from additional data involves discarding the existing classifier and retraining a new classifier using all data that have been accumulated thus far. However, this approach does not conform to the definition of incremental learning, as it causes all previously acquired knowledge to be lost, a phenomenon known as catastrophic forgetting (or interference) [14], [15]. Not conforming to the incremental learning definition aside, this approach is undesirable if retraining is computationally or financially costly, but more importantly it is unfeasible for prohibitively large databases or when the original dataset is lost, corrupted, discarded, or otherwise unavailable. Both scenarios are common in practice; many applications, such as gas transmission pipeline analysis, generate massive amounts of data that renders the use of entire data at once impossible. Furthermore, unavailability of prior data is also common in databases of restricted or confidential access, such as in medical and military applications in general, and the Electric Power Research Institute's (EPRI) NDE Level 3 inspector examination data in particular.

Therefore, several alternative approaches to incremental learning have been developed, including online learning algorithms that learn one instance at a time [16], [17], and partial memory and boundary analysis algorithms that memorize a carefully selected subset of extreme examples that lie along the decision boundaries [18]–[22]. However, such algorithms have limited applicability for realworld NDE problems due to restrictions on classifier type, number of classes that can be learned, or the amount of data that can be analyzed.

In some studies, incremental learning involves controlled modification of classifier weights [23], [24], or incrementally growing/pruning of classifier architecture [25]–[28]. This approach evaluates current performance of the classifier and adjusts the classifier architecture if and when the present architecture is not sufficient to represent the decision boundary being learned. One of the most successful implementations of this approach is ARTMAP [29]. However, ARTMAP has its own drawbacks, such as cluster proliferation, sensitivity of the performance to the selection of the algorithm parameters, to the noise levels in the training data, or to the order in which training data are presented. Various approaches have been suggested to overcome such difficulties [30]–[32], along with new algorithms, such as growing neural gas networks [33] and cell structures [34], [35]. A theoretical framework for the design and analysis of incremental learning algorithms is presented in [36].

B. Ensemble of Classifiers

Learn++, the proposed incremental learning algorithm, is based on the ensemble of classifiers approach. Ensemble approaches typically aim at improving the classifier accuracy on a complex classification problem through a divide-and-conquer approach. In essence, a group of simple classifiers is generated typically from bootstrapped, jackknifed, or bagged replicas of the training data (or by changing other parameters of the classifier), which then are combined through a classifier combination scheme, such as the weighted majority voting [37]. The ensemble generally is formed from weak classifiers to take advantage of their so-called instability [38]. This instability promotes diversity in the classifiers by forcing them to construct sufficiently different decision boundaries (classification rules) for minor modifications in their training datasets, which in turn causes each classifier to make different errors on any given instance. A strategic combination of these classifiers then eliminates the individual errors, generating a strong classifier. Formal definitions of weak and strong classifiers can be found in [39].

Learn++ is in part inspired by the AdaBoost (adaptive boosting) algorithm, one of the most successful implementations of the ensemble approach. Boosting creates a strong learner that can achieve an arbitrarily low error rate by combining a group of weak learners that can do barely better than random guessing [40], [41]. Ensemble approaches have drawn much interest and hence have been well researched. Such approaches, include but are not limited to, Wolpert's stacked generalization [42], and Jordan's hierarchical mixture of experts (HME) model [43], as well as Schapire's AdaBoost. Excellent reviews of various methods for combining classifiers can be found in [44], [45], and an overall review of the ensemble approaches can be found in [46]–[49].

Research in ensemble systems has predominantly concentrated on improving the generalization performance in complex problems. Feasibility of ensemble classifiers in incremental learning, however, has been largely unexplored. Learn++ was developed to close this gap by exploring the prospect of using an ensemble systems approach specifically for incremental learning [12].

III. Learn++ as an Ensemble Approach for Incremental Learning

A. The Learn++ Algorithm

In essence, Learn++ generates a set of classifiers (henceforth hypotheses) and combines them through weighted majority voting of the classes predicted by the individual hypotheses. The hypotheses are obtained by training a base classifier, typically a weak learner, using instances drawn from iteratively updated distributions of the training database. The distribution update rule used by Learn++ is strategically designed to accommodate additional datasets, in particular those featuring new classes. Each classifier added to the ensemble is trained using a set of examples drawn from a weighted distribution that gives higher weights to examples misclassified by the previous ensemble. The Learn++ algorithm is explained in detail below, and a block diagram appears in Fig. 1.

For each database D_k, $k = 1, \ldots, K$ that becomes available, the inputs to Learn++ are labeled training data $S_k = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, m_k\}$ where $\mathbf{x}_i$ and y_i are training instances and their correct classes, respectively; a weak-learning algorithm BaseClassifier; and an integer T_k, the maximum number of classifiers to be generated. For brevity we will drop the subscript k from all other internal variables. BaseClassifier can be any supervised algorithm that achieves at least 50% correct classification on S_k after being trained on a subset of S_k. This is a fairly mild requirement. In fact, for a two-class problem, this is the least that can be expected from a classifier.

At each iteration t, Learn++ first initializes a distribution $\mathbf{D_t}$, by normalizing a set of weights, $\mathbf{w_t}$, assigned to instances based on their individual classification by the current ensemble (Step 1):

$$\mathbf{D_t} = \mathbf{w}_t \Big/ \sum_{i=1}^{m} w_t(i). \tag{1}$$

Learn++ then dichotomizes S_k by drawing a training subset TR_t and a test subset TE_t according to $\mathbf{D_t}$ (Step 2). Unless there is prior reason to choose otherwise, $\mathbf{D_t}$ is initially set to be uniform, giving equal probability to each instance to be selected into TR_1. Learn++ then calls BaseClassifier to generate the t^{th} classifier, hypothesis h_t (Step 3). The error of h_t is computed on $S_k = TR_t + TE_t$ by adding the distribution weights of all misclassified instances (Step 4):

$$\varepsilon_t = \sum_{i:h_t(\mathbf{x}_i)\neq y_i} D_t(i) = \sum_{i=1}^{m_k} D_t(i)\,[|h_t(\mathbf{x}_i) \neq y_i|]\,, \tag{2}$$

where $[|\bullet|]$ is 1 if the predicate is true, and 0 otherwise. If $\varepsilon_t > 1/2$, the current h_t is discarded and a new h_t is generated from a fresh set of TR_t and TE_t. If $\varepsilon_t < 1/2$, then the normalized error β_t is computed as:

$$\beta_t = \varepsilon_t/(1 - \varepsilon_t),\ 0 < \beta_t < 1. \tag{3}$$

All hypotheses generated in the previous t iterations then are combined using weighted majority voting (Step 5) to construct the composite hypothesis H_t:

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(\mathbf{x})=y} \log\frac{1}{\beta_t}. \tag{4}$$

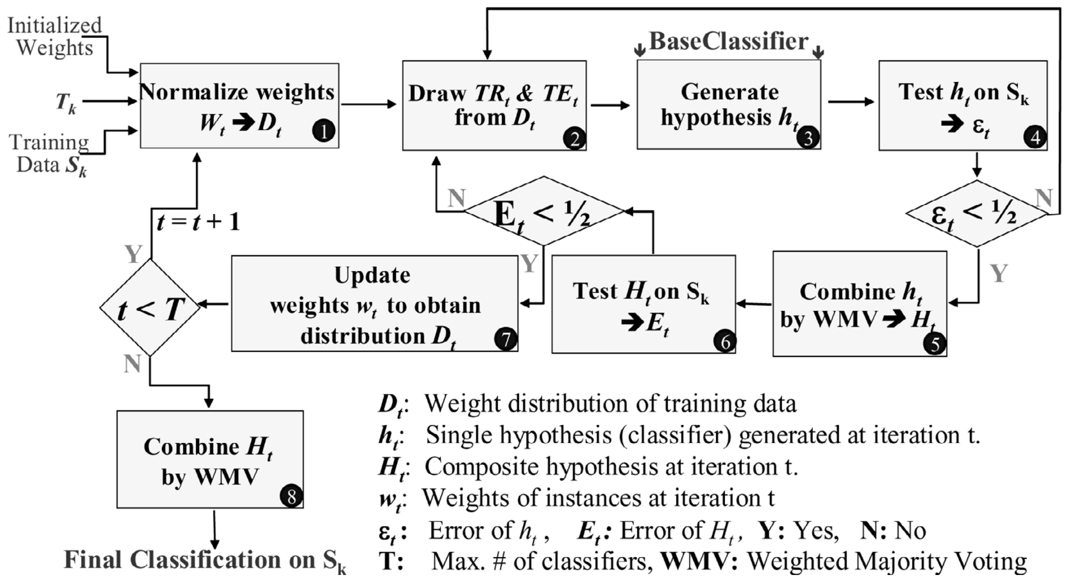

Initialized Weights
T_k
Training Data S_k
Normalize weights $W_t \Rightarrow D_t$ 1
Draw TR_t & TE_t from D_t 2
BaseClassifier
Generate hypothesis h_t 3
Test h_t on $S_k \Rightarrow \varepsilon_t$ 4
$\varepsilon_t < 1/2$
N
Y
Combine h_t by WMV $\Rightarrow H_t$ 5
Test H_t on $S_k \Rightarrow E_t$ 6
$E_t < 1/2$
N
Y
Update weights w_t to obtain distribution D_t 7
$t < T$
$t = t + 1$
Y
N
Combine H_t by WMV 8
Final Classification on S_k
D_t: Weight distribution of training data
h_t: Single hypothesis (classifier) generated at iteration t.
H_t: Composite hypothesis at iteration t.
w_t: Weights of instances at iteration t
ε_t: Error of h_t, E_t: Error of H_t, Y: Yes, N: No
T: Max. # of classifiers, WMV: Weighted Majority Voting

Fig. 1. The block diagram of the Learn++ algorithm for each database S_k that becomes available.

H_t decides on the class that receives the highest total vote from individual hypotheses. This voting is less than democratic, however, as voting weights are based on the normalized errors β_t: hypotheses with lower normalized errors are given larger weights, so that the classes predicted by hypotheses with proven track records are weighted more heavily. The composite error E_t made by H_t then is computed as the sum of distribution weights of instances misclassified by H_t (Step 6) as:

$$E_t = \sum_{i:H_t(\mathbf{x}_i)\neq y_i} \mathbf{D}_t(i) = \sum_{i=1}^{m_k} \mathbf{D}_t(i)\,[|H_t(\mathbf{x}_i) \neq y_i|]\,. \tag{5}$$

If $E_t > 1/2$, a new h_t is generated using a new training subset. Otherwise, the composite normalized error is computed as:

$$B_t = E_t\big/(1 - E_t),\ 0 < B_t < 1. \tag{6}$$

The weights $w_t(i)$ then are updated to be used in computing the next distribution $\mathbf{D}_{t+1}$, which in turn is used in selecting the next training and testing subsets, TR_{t+1} and TE_{t+1}, respectively. The following distribution update rule then allows Learn++ to learn incrementally (Step 7):

$$\begin{aligned} w_{t+1}(i) &= w_t(i) \times B_t^{1-[|H_t(\mathbf{x}_i)\neq y_i|]} \\ &= w_t(i) \times \begin{cases} B_t, & \text{if } H_t(\mathbf{x_i}) = y_i, \\ 1, & \text{otherwise} \end{cases}, \end{aligned} \tag{7}$$

According to this rule, weights of instances correctly classified by the composite hypothesis H_t are reduced (since $0 < B_t < 1$), and the weights of misclassified instances are kept unchanged. After normalization (in Step 1 of iteration $t+1$), the probability of correctly classified instances being chosen into TR_{t+1} is reduced, and those of misclassified ones are effectively increased. Readers familiar with AdaBoost will notice the additional steps of creation of the composite hypothesis H_t in Learn++ as one of the main differences between the two algorithms. AdaBoost uses the previously created hypothesis h_t to update the weights, and Learn++ uses H_t and its performance on weight update. The focus of AdaBoost is only indirectly based on the performance of the ensemble, but more directly on the performance of the previously generated single hypothesis h_t; and Learn++ focuses on instances that are difficult to classify—instances that have not yet been properly learned—by the entire ensemble generated thus far. This is precisely what allows Learn++ to learn incrementally, especially when additional classes are introduced in the new data; concentrate on newly introduced instances, particularly those coming from previously unseen classes, as these are precisely the instances that have not been learned yet by the ensemble, and hence most difficult to classify.

After T_k hypotheses are generated for each database D_k, the final hypothesis H_{final} is obtained by combining all hypotheses that have been generated thus far using the weighted majority-voting rule (Step 8), which chooses the class that receives the highest total vote among all classifiers:

$$\begin{aligned} H_{\text{final}}(\mathbf{x}) &= \arg\max_{y\in Y} \sum_{k=1}^{K} \sum_{t:h_t(\mathbf{x})=y} \log\frac{1}{\beta_t}, \\ t &= 1, 2, \cdots, T_k. \end{aligned} \tag{8}$$

B. Dynamically Updating Voting Weights

We note that in the previously described algorithm, voting weights are determined—and fixed prior to testing—based on individual performances of hypotheses on their own training data subset. This weight-assigning rule does make sense, and indeed works quite well in practice [12]. However, because each classifier is trained only on a small subset of the entire training data, good performance on one subset does not ensure similar performance on field data. Therefore, a rule that dynamically estimates which hypotheses are likely to correctly classify each unlabeled field data and gives higher voting weights to those hypotheses should give better performance.

Statistical distance metrics, such as Mahalanobis distance, can be used to determine the distance of the unknown instance to the datasets used to train individual classifiers. Classifiers trained with datasets closer to the unknown instance then can be given larger weights. Note that this approach does not require the (previously used) training data to be saved but only the mean and covariance matrices, which are typically much smaller in size than the original data.

In this work, class-specific Mahalanobis distance is introduced as a modification to the original Learn++ voting weights. We first define TR_{tc} as a subset of TR_t (the training data used during the t^{th} iteration), where TR_{tc} includes only those instances of TR_t that belong to class c, that is:

$$TR_{tc} = \{\mathbf{x}_i \mid \mathbf{x}_i \in TR_t \ \& \ y_i = c\} \ni TR_t = \bigcup_{c=1}^{C} TR_{tc}, \tag{9}$$

where C is the total number of classes. We define the class-specific Mahalanobis distance of an unknown instance $\mathbf{x}$ to class-c training instances of t^{th} classifier as:

$$M_{tc} = (\mathbf{x} - \mathbf{m}_{tc})^T \mathbf{C}_{tc}^{-1} (\mathbf{x} - \mathbf{m}_{tc}), \quad c = 1, 2, \cdots, C, \tag{10}$$

where $\mathbf{m}_{tc}$ is the mean of TR_{tc}, and $\mathbf{C}_{tc}$ is the covariance matrix of TR_{tc}. The Mahalanobis distance-based weight MW_t of the t^{th} hypothesis then can be obtained as (or as a function of):

$$MW_t = \frac{1}{\min(M_{tc})}, \quad c = 1, 2, \cdots, C. \tag{11}$$

Eq. (10) and (11) find the minimum Mahalanobis distance between instance $\mathbf{x}$ and each one of the C data subsets TR_{tc}, and assigns the Mahalanobis weight of the t^{th} hypothesis as the reciprocal of this minimum Mahalanobis distance. Note that the class-specific Mahalanobis distance is dependent on the particular instance that is being classified. Therefore, this procedure updates the voting weights for each incoming test data instance, and hence provides a dynamic weight update rule.

The Mahalanobis distance metric implicitly assumes that the data is drawn from a Gaussian distribution, which in general may not be the case. However, in our empirical analysis of the algorithm on two different NDE datasets, Mahalanobis distance-based voting weights provided better results than voting weights based on the hypothesis performance on training data subsets.

C. Estimating the Confidence of Learn++ Classification

An additional benefit of the Learn++ algorithm is that the inherent voting mechanism hints at a simple procedure for determining the confidence of the algorithm in its own decision, particularly when the new data does not contain new classes. Intuitively, a vast majority of hypotheses agreeing on a given instance can be interpreted as the algorithm having more confidence in its own decision, compared to a decision made by a mere majority. Let us assume that a total of T hypotheses are generated in K training sessions for a C-class problem. For any given instance $\mathbf{x}$, the final classification is class c, if the total vote class c receives:

$$\xi_c = \sum_{t: h_t(\mathbf{x}) = c} \Psi_t, \ t = 1, \cdots, T; \ c = 1, \cdots, C, \tag{12}$$

is maximum, where Ψ_t denotes the voting weight of the t^{th} hypothesis h_t, whether we use static weights or dynamically updated voting weights. Normalizing the votes received by each class:

$$\gamma_c = \xi_c \Big/ \sum_{c=1}^{C} \xi_c, \tag{13}$$

allows us to interpret γ_c as a measure of confidence on a 0 to 1 scale. We note that γ_c values do not represent the accuracy of the results, nor are they formally related to the statistical definition of confidence intervals determined through hypothesis testing. The γ_c merely represents a measure of the confidence of the algorithm in its own decision. However, under a reasonable set of assumptions, γ_c can be interpreted as the probability that the input pattern belongs to class c [50]. Keeping this in mind, we can heuristically define the following confidence ranges: $0.9 < \gamma_c < 0.6$ very low, $0.6 < \gamma_c < 0.7$ low, $0.7 < \gamma_c < 0.8$ medium, $0.8 < \gamma_c < 0.9$ high, and $0.9 < \gamma_c < 1$ very high confidence. This procedure can flag the misclassified instances by assigning them lower confidence. As will be discussed in Section IV, this procedure produced promising and noteworthy trends and outcomes. A theoretical framework on how combining classifiers improve classification confidence can be found in [51].

IV. Results

The original Learn++ algorithm (with static voting weights) was evaluated on a number of real world and benchmark databases, and Learn++ was able to learn the new information from the new data, even when new classes were introduced with subsequent databases. The results of these experiments with the static voting weights can be found in [12], [52], [53] for various other benchmark and real world databases. For this work, we developed and evaluated the modified Learn++ algorithm with dynamically updated voting weights, on a three-class and a four-class ultrasonic weld inspection problem. For both applications, the task was identifying the type of defects or discontinuities in or around the welding region from ultrasonic A-scans. In each case, the algorithm was trained incrementally in three training sessions in which Learn++

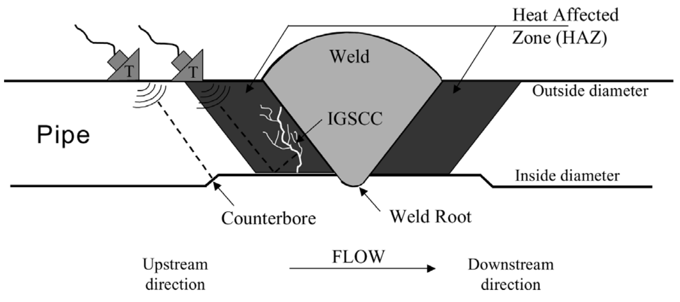


Fig. 2. Ultrasonic testing of nuclear power plant pipes.

was provided with a new database in each session. In the three-class problem, no new classes were introduced with new data, but an additional class was introduced with each database in the four-class problem.

A. Results on the Three-Class Problem

Welding regions often are susceptible to various kinds of defects due to imperfections introduced into the material during the welding process. In nuclear power plant pipes, for example, such defects manifest themselves in the form of intergranular stress corrosion crackings (IGSCC), usually in an area immediately neighboring the welding region, known as the heat-affected zone. Such cracks can be detected by using ultrasonic (or eddy current) techniques. However, also in the vicinity of this zone, there are often other type of reflectors or discontinuities, including counterbores and weld roots, which are considered as geometric reflectors. Counterbores and weldroots do not pose any threat to the structural integrity of the pipe; however, they often generate signals that are very similar to those generated by cracks, making the defect identification a very challenging task. The cross section in Fig. 2 conceptually illustrates the ultrasonic testing procedure using 1 MHz ultrasonic pulse-echo contact transducers, used to generate the first database analyzed in this study. Fig. 3 illustrates typical signals from each type of reflector.

The goal of the classification algorithm is the identification of three different types of indicators, namely, crack, counterbore, and weld root from the ultrasonic A-scans. Three training databases $S_1 \sim S_3$ of 300 A-scans each, and a validation database, TEST, of 487 A-scans were acquired from the above described system. Discrete wavelet transform (DWT) coefficients were computed for each 256-point A-scan to be used as feature vectors. During each of the three training sessions, only one of the training databases was made available to the algorithm to test the incremental learning capability of Learn++. The weak BaseClassifier was a single hidden layer MLP of 30 nodes, with a relatively large mean square error goal of 0.1. We emphasize that any supervised classifier can be used as a weak learner by keeping its architecture small and its error goal high, with respect to the complexity of the classification problem. In this application, each weak MLP—used as a base classifier—obtained around 65% classification performance on its own training data.

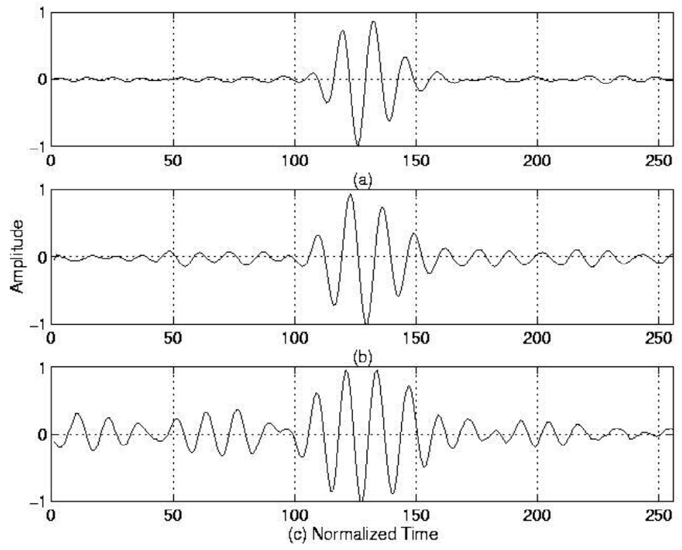


Fig. 3. Sample signals from (a) crack, (b) counterbore, and (c) weld root.

TABLE I
Classification Performance of Learn++ on the Three-Class Weld Inspection Database.

	TS_1 (6)	TS_2 (10)	TS_3 (14)
S_1	95.70%	95%	94.30%
S_2	—	95%	95.30%
S_3	—	—	95.10%
TEST	81.90%	91.70%	95.60%

Table I presents the results in which rows indicate the classification performance of Learn++ on each of the databases after each training session (TS_k, $k = 1, 2, 3$). The numbers in parentheses indicate the number of weak classifiers generated in each training session. Originally, T_k was set to 20 for each database. The generalization performance typically reaches a plateau after a certain number of classifiers; therefore, those late classifiers not contributing much to the performance were later removed, truncating the cardinality of the ensemble to the number of classifiers indicated in Table I. A more accurate way of determining the precise number of classifiers is typically to use an additional validation set, if available, to determine when the algorithm should be stopped.

The first three rows indicate the performance of the algorithm on each training set S_k after the k^{th} training session was completed, whereas the last row provides the generalization performance of the algorithm on the TEST dataset after each session. We note that the performance on the validation dataset TEST improved steadily as new databases were introduced, demonstrating the incremental learning ability of the algorithm. Also, the algorithm was able to maintain its classification performance on the previous datasets after training with additional datasets. This shows that previously acquired knowledge was not lost.

As a comparison, the classification performance of a single strong learner with two hidden layers of 30 and 7 nodes, respectively, and an error goal of 0.0005 was also about 95%, although the entire training database (900 instances) was used to train this classifier. Therefore, we conclude that Learn++, by only seeing a fraction of the training database at a time in an incremental fashion, can perform as good as (if not better than) a strong learner that is trained with the entire data at once.

The algorithm's confidence in each decision was computed as described earlier. Table II lists a representative subset of the classification results and confidence levels on the validation dataset after each training session. A number of interesting observations can be made from Table II, which is divided into four sections. The first section shows those instances that were originally misclassified after the first training session but were correctly classified after subsequent sessions. There were 66 such cases (out of 487 in the TEST dataset). Many of these were originally misclassified with rather strong confidences. During the next two training sessions, however, not only their classifications were corrected but also the confidence on the classification improved as well.

The second section of Table II shows those cases consistently classified correctly, but on which the confidence steadily improved with training. A majority of the instances (396 out of 487) belonged to this case. These cases demonstrate that seeing additional data improves the confidence of the algorithm in its own decision, when the decision is indeed correct. This is a very satisfying outcome, as we would intuitively expect improved confidence with additional training. The third section shows examples of those cases that were still misclassified at the end of three training sessions, but the classification confidence decreased with additional training. In these cases (a total of 21 such instances), the confidence was very high after the first training session and decreased to very low after the third training session. These cases demonstrate that, with additional training, the algorithm can determine those cases in which it is making an error. This is also a very satisfying outcome as the algorithm can flag those instances that it is probably making an error by assigning a low confidence to their classification. The fourth section shows the only four instances in which the algorithm either increased its confidence in misclassification or decreased its confidence in correct classification. Such cases are undesired outcomes that are considered as isolated instances (noisy samples or outliers) because there were only four such cases in the entire database.

B. Results on a Four-Class Problem Introducing New Classes

The second database had four types of defects, namely, crack, porosity, slag, and lack of fusion (LOF), all of which appeared in the welding region. Among these, cracks and LOFs pose the most serious threat as they eventually can cause structural damages if remedial actions are not taken.

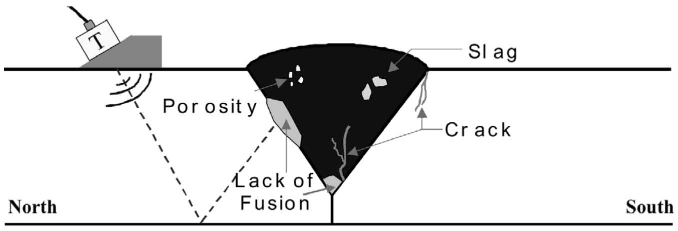


Fig. 4. Ultrasonic weld inspection for identification of cracks, porosity, slag, and LOF.

Fig. 4 conceptually illustrates the testing procedure for this application in which the goal of the classification algorithm is the identification of four different types of defects from the discrete wavelet transform coefficients of the ultrasonic A-scans. A total of 156 C-scans were obtained, each consisting over 12,000 A-scans.

Of the C-scans, 106 were randomly selected to be used for training, and 50 were selected to be used for validation. From the C-scans reserved for training, 2200 A-scans, each 512-points long, were randomly selected for training and 800 were selected for testing (from regions of interest; see Figs. 5–7). The DWT coefficients were computed for each A-scan to be used as feature vectors. The training instances were further divided into three subsets to simulate three different databases that become available at different times. Furthermore, additional classes were added to subsequent datasets to test the incremental learning ability of the algorithm on new classes. Table III shows the distribution of the instances in various datasets, and Fig. 8 illustrates typical signals from these four classes.

As seen in Table III, the first training dataset S_1 had instances only from crack and LOF, but S_2 and S_3 added instances from slag and porosities, respectively. The 800 test signals were never shown to the algorithm during training. The weak learner used to generate individual hypotheses was a single hidden layer MLP with 50 hidden layer nodes. The mean square error goals of all MLPs were preset to a value of 0.02 to prevent overfitting and to ensure a weak learning algorithm.

Results summarized in Table IV indicate that Learn++ was able to correctly classify 99.2% of training instances in S_1, but only 57% of the test instances by combining 8 hypotheses. This performance is not surprising as S_1 had instances only from two classes, but TEST had instances from all four classes. After the next training session, using instances only from S_2, the algorithm was able to correctly classify 89.2% of instances in S_1 and 86.5% of instances in S_2. The performance on TEST dataset improved to 70.5%. After the final training session using instances from S_3, the algorithm correctly classified 88.2%, 88.1%, and 91.2% of instances in S_1, S_2, and S_3, respectively. The classification performance on TEST dataset increased to 83.8%, demonstrating the incremental learning capability of the Learn++ algorithm.

As a performance comparison, the same database also was used to train and test a single strong learner. Among various architectures tried, a $149 \times 40 \times 12 \times 4$ two hid-

TABLE II
SAMPLE CONFIDENCES ON THE TEST DATASET FOR EACH TRAINING SESSION.

		Training session 1		Training session 2		Training session 3	
Instance No.	True Class	Class	Confidence	Class	Confidence	Class	Confidence
Misclassification Corrected with Improved Confidence (66 such cases)							
25	Crack	Cbore	0.69	Crack	0.91	Crack	0.96
144	Crack	Cbore	0.54	Crack	0.86	Crack	0.91
177	Cbore	Crack	0.81	Crack	0.55	Cbore	0.71
267	Cbore	Crack	0.52	Cbore	0.86	Cbore	0.96
308	Root	Cbore	0.69	Root	0.47	Root	0.87
354	Root	Crack	0.87	Crack	0.64	Root	0.79
438	Crack	Cbore	0.57	Crack	0.76	Crack	0.92
Improved Confidence in Correct Classification (396 such cases)							
67	Cbore	Cbore	0.66	Cbore	0.94	Cbore	0.96
313	Crack	Crack	0.6	Crack	0.73	Crack	0.88
321	Cbore	Cbore	0.47	Cbore	0.87	Cbore	0.93
404	Root	Root	0.59	Root	0.73	Root	0.96
Reduced Confidence in Misclassification (21 such cases)							
261	Crack	Cbore	1	Cbore	1	Cbore	0.54
440	Cbore	Root	1	Root	0.85	Root	0.66
456	Cbore	Root	0.65	Cbore	0.52	Crack	0.55
Utterly Confused Classifier (4 such cases)							
3	Root	Root	0.49	Crack	0.55	Crack	0.59
45	Crack	Crack	0.78	Crack	0.61	Crack	0.53
78	Cbore	Crack	0.67	Cbore	0.52	Crack	0.56
93	Root	Root	0.94	Crack	0.58	Crack	0.58

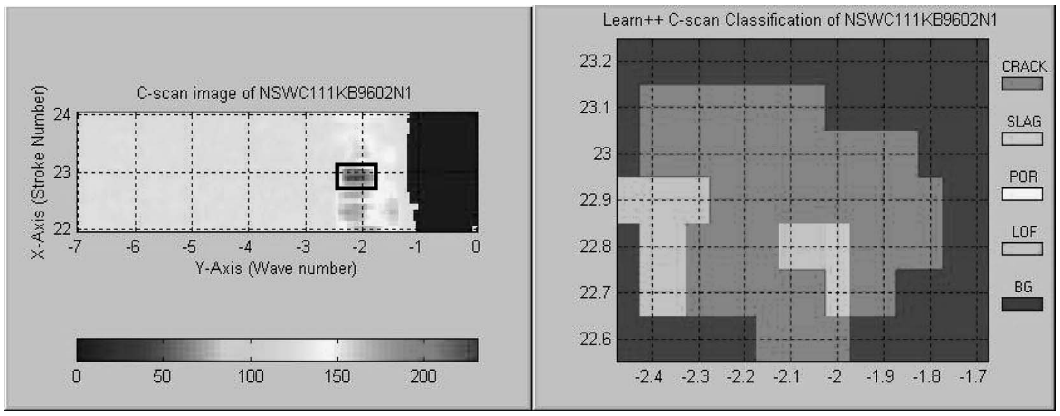


Fig. 5. Original C-scan and Learn++ classification, correct class: Crack.

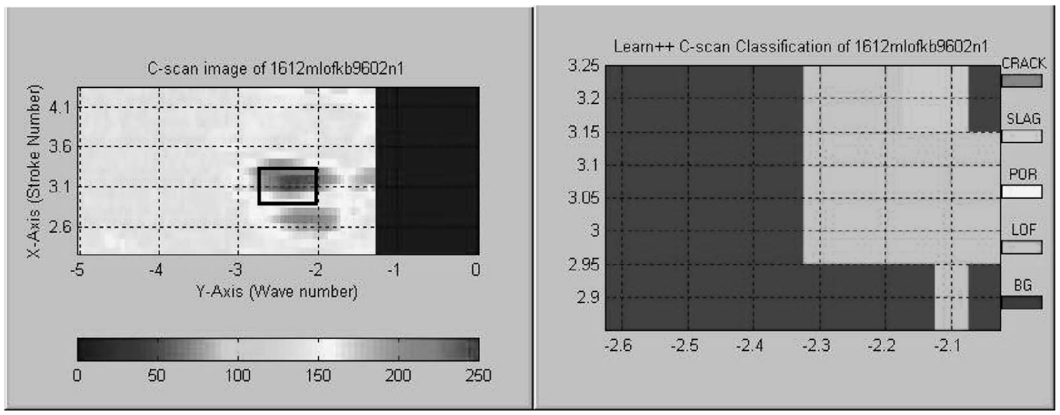


Fig. 6. Original C-scan and Learn++ classification, correct class: LOF.

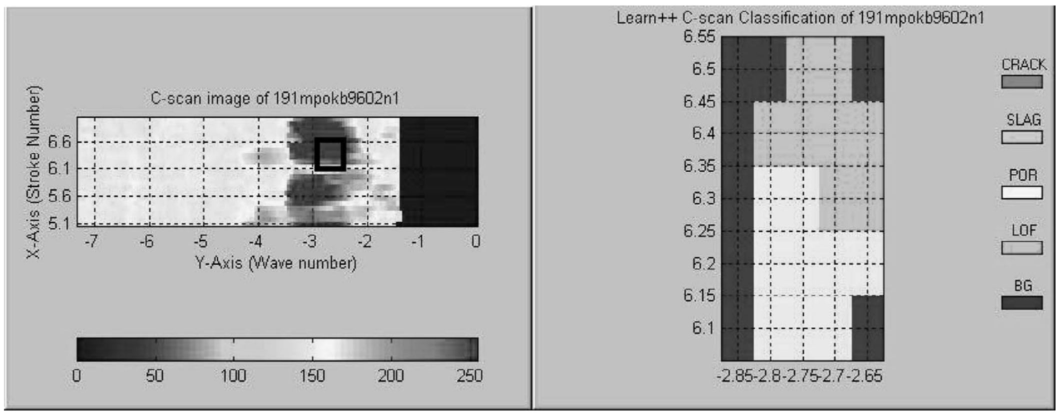


Fig. 7. Original C-scan and Learn++ classification, correct class: Porosity.

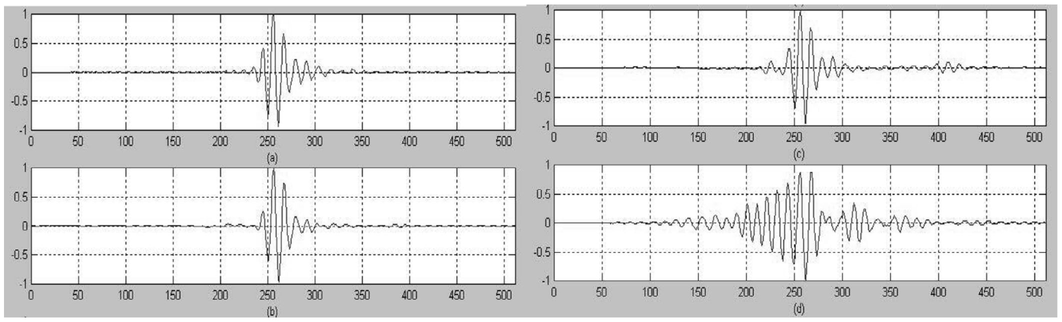


Fig. 8. Typical A-scans (a) crack, (b) LOF, (c) slag, (d) porosity.

TABLE III
Distribution of Weld Inspection Signals.

	Crack	LOF	Slag	Porosity
S_1	300	300	0	0
S_2	150	300	150	0
S_3	200	250	250	300
TEST	200	300	200	100

TABLE IV
Classification Performance of Learn++ on the Four-Class Problem.

Inc. Train→ ↓ Dataset	Training 1 (8)	Training 2 (27)	Training 3 (43)
S_1	99.20%	89.20%	88.20%
S_2	—	86.50%	88.10%
S_3	—	—	96.40%
TEST	57.00%	70.50%	83.80%

den layer MLP with an error goal of 0.001 provided the best test performance, which was about 75%. The original Learn++ algorithm that did not used dynamically updated voting weights also was evaluated on this database, and its performance was found to be less than 80% after a similar three session training procedure that introduced third and fourth classes in second and third training sessions [52].

Finally, Learn++ was tested on the entire set of C-scan images. On each C-scan, a previously identified rectangular region was selected and classified by Learn++, creating a classification image of the selected rectangular region. Median filtering then was applied to the classification image to remove isolated pixels, producing the final classification image. The final C-scan classification was determined according to a simple majority of its individual A-scan classifications. Figs. 5–7 illustrate examples of raw C-scans and their respective classification images. Table V presents the classification performance of Learn++ compared to that of the strong learner trained on the entire training dataset.

The C-scans indicated with an unknown (Unk.) classification in Table V refer to those cases in which an approximately equal number of A-scans in the selected region had conflicting classifications.

V. Discussion and Cconclusions

We introduced Learn++, an incremental learning algorithm for supervised neural networks that uses an ensemble of classifiers for learning new data. The feasibility of the approach has been validated on two real-world databases of ultrasonic weld inspection signals. Learn++ shows very promising results in learning new data, in both scenarios

TABLE V
Comparison of Learn++ and Strong Learner on C-scans of Weld Inspection Data.

	No. of C-scans (training)	No. of C-scans missed/Unk. (training)	Classification performance	No. of C-scans (validation)	No. of C-scans missed/Unk. (validation)	Classification performance
Strong learner	106	8/1	92.40%	50	11/2	77.10%
Learn++	106	1/0	99.10%	50	7/4	84.80%

in which the new data may or may not include previously unseen classes. Both of these scenarios occur commonly in nondestructive testing; therefore, the algorithm can be of benefit in a broad spectrum of NDE applications.

Although we have implemented Learn++ using MLPs as base classifiers, the algorithm itself is independent on the choice of a particular classification algorithm, and it is able to work well on all supervised classifiers whose weakness (instability) can be controlled. In particular, most supervised neural network classifiers can be used as a base classifier as their weakness can be easily controlled through network size and/or error goal. Results demonstrating such classifier independence on a number of other applications were presented in [52].

The algorithm has additional desirable traits. It is intuitive and simple to implement, and it is applicable to a diverse set of NDE and other real world automated identification and characterization problems. It can be trained very quickly, without falling into overfitting problems because using weak base classifiers avoids lengthy training sessions that are mostly spent on fine tuning the decision boundary, which itself may be—and typically is—influenced by noise. The algorithm also is capable of estimating its own confidence on individual classifications in which it typically indicates high confidence on correctly classified instances and low confidence on misclassified instances after several training sessions. Furthermore, the confidence on correctly classified instances tend to increase and the confidence on misclassified instances tend to decrease as new data become available to the algorithm, a very satisfying and comforting property.

The main drawback of Learn++ is the computational burden due to the overhead added by computing multiple hypotheses and saving all classifier parameters for these hypotheses. Because each classifier is a weak learner, it has fewer parameters than its strong counterpart; therefore, it can be trained much faster. However, because the parameters of a large number of hypotheses may need to be saved, its space complexity can be high, although ever increasing storage capacities should reduce the significance of this drawback. An additional overhead in using Mahalanobis distance-based voting weights is the computation of inverse of covariance matrices. For most practical applications, the additional computational overhead is not significant. For a very large number of features, however, this may be rather costly, in which case the user needs to weigh the added computational burden against the performance improvement over the original version of Learn++.

Learn++ has two key components, both of which can be improved. The first one is the selection of the subsequent training dataset, which is determined by the distribution update rule. This rule can be optimized to allow faster learning and reduced computational complexity. A learning rate parameter, similar to that of gradient descent type optimization algorithms, is being considered to control the rate at which distribution weights are updated.

The second key component is the procedure by which hypotheses are combined. We described two approaches for Learn++ using weighted majority voting in which the voting weights can be determined either by training data performance of hypotheses, or dynamically through the class-specific Mahalanobis distances. A combination of the two may provide better overall performance. Furthermore, a second level of classifier(s), similar to those used in the hierarchical mixture of classifiers, may prove to be effective in optimally identifying such weights. Work is currently underway to address these issues.

We have shown how the inherent voting scheme can be used to determine the confidence of the algorithm in its own individual classifications on applications that do not introduce new classes. Similar approaches are currently being investigated for estimating the true field performance of the algorithm along with its confidence intervals, as compared to those measures obtained through hypothesis testing, even for those cases that do introduce new classes.

References

[1] J. Haueisen, J. R. Unger, T. Beuker, and M. E. Bellemann, "Evaluation of inverse algorithms in the analysis of magnetic flux leakage data," *IEEE Trans. Magn.*, vol. 38, pp. 1481–1488, May 2002.

[2] M. Afzal and S. Udpa, "Advanced signal processing of magnetic flux leakage data obtained from seamless gas pipeline," *NDT & E Int.*, vol. 35, no. 7, pp. 449–457, 2002.

[3] K. Allweins, G. Gierelt, H. J. Krause, and M. Kreutzbruck, "Defect detection in thick aircraft samples based on HTS SQUID-magnetometry and pattern recognition," *IEEE Trans. Appl. Superconduct.*, vol. 13, pp. 809–814, June 2003.

[4] D. Donskoy, A. Sutin, and A. Ekimov, "Nonlinear acoustic interaction on contact interfaces and its use for nondestructive testing," *NDT & E Int.*, vol. 34, pp. 231–238, June 2001.

[5] E. A. Nawapak, L. Udpa, and J. Chao, "Morphological processing for crack detection in eddy current images of jet engine disks," in *Review of Progress in Quantitative Nondestructive Evaluation.* vol. 18, D. O. Thompson and D. E. Chimenti, Eds. New York: Plenum, 1999, pp. 751–758.

[6] R. Polikar, L. Udpa, S. S. Udpa, and T. Taylor, "Frequency invariant classification of ultrasonic weld inspection signals," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 45, pp. 614–625, May 1998.

[7] P. Ramuhalli, L. Udpa, and S. S. Udpa, "Automated signal classification systems for ultrasonic weld inspection signals," *Mater. Eval.*, vol. 58, pp. 65–69, Jan. 2000.

[8] U. B. Halabe, A. Vasudevan, H. V. GangaRao, P. Klinkhachorn, and G. L. Shives, "Nondestructive evaluation of fiber reinforced polymer bridge decks using digital infrared imaging," in *Proc. IEEE 35th Southeastern Symp. System Theory*, Mar. 2003, pp. 372–375.

[9] A. W. Meyer and J. V. Candy, "Iterative processing of ultrasonic measurements to characterize flaws in critical optical components," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 49, pp. 1124–1138, Aug. 2002.

[10] S. Mandayam, K. Jahan, and D. B. Cleary, "Ultrasound inspection of wastewater concrete pipelines—signal processing and defect characterization," in *Review of Progress in Quantitative Nondestructive Evaluation.* vol. 20, D. O. Thompson, Ed. New York: AIP Press, 2001, pp. 1210–1217.

[11] P. Ramuhalli, L. Udpa, and S. S. Udpa, "A signal classification network that computes its own reliability," in *Review of Progress in Quantitative Nondestructive Evaluation.* vol. 18, D. O. Thompson and D. E. Chimenti, Eds. New York: Plenum, 1999, pp. 857–864.

[12] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst. Man and Cybernetics (C)*, vol. 31, pp. 497–508, Nov. 2001.

[13] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms and architectures," *Neural Networks*, vol. 1, pp. 17–61, Jan. 1988.

[14] M. McCloskey and N. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *The Psychology of Learning and Motivation.* vol. 24, G. H. Bower, Ed. San Diego: Academic, 1989, pp. 109–164.

[15] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognitive Sci.*, vol. 3, no. 4, 1999, pp. 128–135.

[16] D. P. Helmbold, S. Panizza, and M. K. Warmuth, "Direct and indirect algorithms for on-line learning of disjunctions," *Theor. Comput. Sci.*, vol. 284, pp. 109–142, 2002.

[17] S. Nieto-Sanchez, E. Triantaphyllou, J. Chen, and T. W. Liao, "Incremental learning algorithm for constructing Boolean functions from positive and negative examples," *Comput. Operations Res.*, vol. 29, no. 12, pp. 1677–1700, 2002.

[18] S. Lange and G. Grieser, "On the power of incremental learning," *Theor. Comput. Sci.*, vol. 288, no. 2, pp. 277–307, 2002.

[19] M. A. Maloof and R. S. Michalski, "Incremental learning with partial instance memory," in *Foundations of Intelligent Systems, Lecture Notes in Artificial Intelligence.* vol. 2366, Berlin: Springer-Verlag, 2002, pp. 16–26.

[20] J. Sancho, W. Pierson, B. Ulug, A. Figueiras-Visal, and S. Ahalt, "Class separability estimation and incremental learning using boundary methods," *Neurocomputing*, vol. 35, pp. 3–26, 2000.

[21] S. Vijayakumar and H. Ogawa, "RKHS-based functional analysis for exact incremental learning," *Neurocomputing*, vol. 29, pp. 85–113, 1999.

[22] P. Mitra, C. A. Murthy, and S. K. Pal, "Data condensation in large databases by incremental learning with support vector machines," *Int. Conf. Pattern Recognition*, 2000, pp. 708–711.

[23] L. Fu, H. H. Hsu, and J. C. Principe, "Incremental backpropagation learning networks," *IEEE Trans. Neural Networks*, vol. 7, no. 3, pp. 757–762, 1996.

[24] K. Yamaguchi, N. Yamaguchi, and N. Ishii, "An incremental learning method with retrieving of interfered patterns," *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1351–1365, 1999.

[25] D. Obradovic, "On-line training of recurrent neural networks with continuous topology adaptation," *IEEE Trans. Neural Networks*, vol. 7, pp. 222–228, Jan. 1996.

[26] N. B. Karayiannis and G. W. Mi, "Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Trans. Neural Networks*, vol. 8, no. 6, pp. 1492–1506, 1997.

[27] J. Ghosh and A. C. Nag, "Knowledge enhancement and reuse with radial basis function networks," *Proc. Int. Joint Conf. Neural Networks*, 2002, pp. 1322–1327.

[28] R. Parekh, J. Yang, and V. Honavar, "Constructive neural network learning algorithms for multi-category pattern classification," *IEEE Tran. Neural Networks*, vol. 11, no. 2, pp. 436–451, 2000.

[29] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 698–713, 1992.

[30] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, no. 5, pp. 881–897, 1996.

[31] F. H. Hamker, "Life-long learning cell structures—continuously learning without catastrophic interference," *Neural Networks*, vol. 14, no. 4-5, pp. 551–573, 2000.

[32] G. C. Anagnostopoulos and M. Georgiopoulos, "Category regions as new geometrical concepts in Fuzzy-ART and Fuzzy-ARTMAP," *Neural Networks*, vol. 15, no. 10, pp. 1205–1221, 2002.

[33] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems.* G. Tesauro, D. Touretzky, and T. Keen, Eds. Cambridge, MA: MIT Press, 1995, pp. 625–632.

[34] D. Heinke and F. H. Hamker, "Comparing neural networks: A benchmark on growing neural gas, growing cell structures, and fuzzy ARTMAP," *IEEE Trans. Neural Networks*, vol. 9, no. 6, pp. 1279–1291, 1998.

[35] F. H. Hamker, "Life-long learning cell structures—Continuously learning without catastrophic interference," *Neural Networks*, vol. 14, pp. 551–572, 2001.

[36] D. Caragea, A. Silvescu, and V. Honavar, "Analysis and synthesis of agents that learn from distributed dynamic data sources," in *Emerging Neural Architectures Based on Neuroscience.* S. Wermter, J. Austin, and D. Willshaw, Eds. ser. Lecture Notes in Artificial Intelligence, vol. 2036, Berlin: Springer-Verlag, 2001, pp. 547–559.

[37] N. Littlestone and M. Warmuth, "Weighted majority algorithm," *Information Comput.*, vol. 108, pp. 212–261, 1994.

[38] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization," *Mach. Learning*, vol. 40, no. 2, pp. 1–19, 2000.

[39] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory.* Cambridge, MA: MIT Press, 1994.

[40] Y. Freund and R. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," *Comput. Syst. Sci.*, vol. 57, no. 1, pp. 119–139, 1997.

[41] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margins: A new explanation for the effectiveness of voting methods," *Ann. Stat.*, vol. 26, no. 5, pp. 1651–1686, 1998.

[42] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[43] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, no. 2, pp. 181–214, 1994.

[44] C. Ji and S. Ma, "Performance and efficiency: Recent advances in supervised learning," *Proc. IEEE*, vol. 87, no. 9, pp. 1519–1535, 1999.

[45] T. G. Dietterich, "Machine learning research," *AI Magazine*, vol. 18, no. 4, pp. 97–136, 1997.

[46] ——, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop on Multiple Classifier Systems*, 2000, pp. 1–15.

[47] T. K. Ho, "Data complexity analysis for classifier combination," in *Proc. 2nd Int. Workshop on Multiple Classifier Systems, LNCS*, 2001, pp. 53–67.

[48] J. Ghosh, "Multiclassifier systems: Back to the future," in *Proc. 3rd Int. Workshop on Multiple Classifier Systems*, 2002, pp. 1–15.

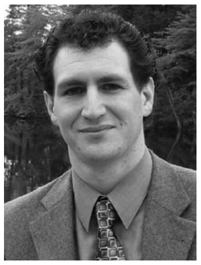
[49] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 4–37, Jan. 2000.

[50] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* New York: Wiley, 2001.

[51] R. E. Schapire, Y. Freund, P. Bartlett, and W. Sun Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Stat.*, vol. 26, no. 5, pp. 1651–1686, 1998.

[52] R. Polikar, J. Byorick, S. Krause, A. Marino, and M. Moreton, "Learn++: A classifier independent incremental learning algorithm for supervised neural networks," in *Proc. Int. Joint Conf. Neural Networks*, 2002, pp. 1742–1747.

[53] R. Polikar, "Algorithms for enhancing pattern separability, optimum feature selection an incremental learning with applications to gas sensing electronic nose systems," Ph.D. dissertation, Iowa State University, Ames, 2000.



Robi Polikar (S'92–M'01) received his B.S. degree in electronics and communications engineering from Istanbul Technical University, Istanbul, Turkey, in 1993, M.S. and Ph.D. degrees, both co-majors in biomedical engineering and electrical engineering, from Iowa State University, Ames, Iowa, in 1995 and in 2000, respectively. He is currently an assistant professor with the Department of Electrical and Computer Engineering at Rowan University, Glassboro, NJ.

His current research interests include signal processing, pattern recognition, neural systems, machine learning, and computational models of learning, with applications to biomedical engineering and imaging, chemical sensing, nondestructive evaluation and testing. He also teaches upper level undergraduate and graduate courses in wavelet theory, pattern recognition, neural networks and biomedical systems and devices at Rowan University.

He is a member of IEEE, American Society for Engineering Education (ASEE), Tau Beta Pi, and Eta Kappa Nu. His current work is funded primarily through National Science Foundation (NSF)'s Career program and National Institutes of Health (NIH)'s Collaborative Research in Computational Neuroscience program.

Lalita Udpa (S'84–M'86–SM'92) received her Ph.D. degree in electrical engineering from Colorado State University, Fort Collins, CO, in 1986. She joined the Department of Electrical and Computer Engineering at Iowa State University, Ames, Iowa, as an assistant professor where she served from 1990–2001. Since 2002, she has been with Michigan State University, East Lansing, Michigan, where she is currently a Professor in the Department of Electrical and Computer Engineering.

Dr. Udpa works primarily in the broad areas of nondestructive evaluation (NDE), signal processing, and data fusion. Her research interests also include development of computational models for the forward problem in electromagnetic NDE, signal processing, pattern recognition, and learning algorithms for NDE data.

Dr. Udpa is an associate technical editor of the American Society of Nondestructive Testing Journals on Materials Evaluation and Research Techniques in NDE.

Satish S. Udpa (S'82–M'82–SM'91–F'03) began serving as the Chairperson of the Department of Electrical and Computer Engineering at Michigan State University, East Lansing, Michigan, in August 2001. Prior to coming to Michigan State, Dr. Udpa was the Whitney Professor of Electrical and Computer Engineering at Iowa State University, Ames, Iowa, and Associate Chairperson for Research and Graduate Studies.

Dr. Udpa received a B.Tech. degree in electrical engineering and a Postgraduate Diploma from J.N.T. University, Hyderabad, India. He earned his master's and doctoral degrees in electrical engineering from Colorado State University, Fort Collins, CO.

He is a Fellow of the IEEE, a Fellow of the American Society for Nondestructive Testing as well as a Fellow of the Indian Society for Nondestructive Testing.

Dr. Udpa holds three patents and has published more than 180 journal articles, book chapters, and research reports. His research interests include nondestructive evaluation, biomedical signal processing, electromagnetics, signal and image processing, and pattern recognition.

Vasant Honavar (S'84–M'90) received his Ph.D. in computer science and cognitive science from the University of Wisconsin, Madison in 1990. He founded and directs the Artificial Intelligence Research Laboratory at Iowa State University (ISU), Ames, Iowa, where he is currently a professor of Computer Science and Professor and Chair of the Graduate Program in Bioinformatics and Computational Biology. He is also a member of the Lawrence E. Baker Center for Bioinformatics and Biological Statistics.

Dr. Honavar's research and teaching interests include artificial intelligence, machine learning, bioinformatics and computational biology, distributed intelligent information networks, intrusion detection, neural and evolutionary computing, and data mining.

He has published over 100 research articles in refereed journals, conferences, and books, and has co-edited 6 books. He is a co-editor-in-chief of the Journal of Cognitive Systems Research and a member of the Editorial Board of the *Machine Learning Journal*. His research is funded in part by National Science Foundation (NSF), NIH, the Carver Foundation, and Pioneer Hi-Bred. Prof. Honavar is a member of Association for Computing Machinery (ACM), American Association for Artificial Intelligence (AAAI), IEEE, and the New York Academy of Sciences.