# Multiclassifier Systems: Back to the Future

Joydeep Ghosh

Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
ghosh@ece.utexas.edu
http://www.lans.ece.utexas.edu/∼ghosh

**Abstract.** While a variety of multiple classifier systems have been studied since at least the late 1950's, this area came alive in the 90's with significant theoretical advances as well as numerous successful practical applications. This article argues that our current understanding of ensemble-type multiclassifier systems is now quite mature and exhorts the reader to consider a broader set of models and situations for further progress. Some of these scenarios have already been considered in classical pattern recognition literature, but revisiting them often leads to new insights and progress. As an example, we consider how to integrate multiple *clusterings*, a problem central to several emerging distributed data mining applications. We also revisit output space decomposition to show how this can lead to extraction of valuable domain knowledge in addition to improved classification accuracy.

## 1 A Brief History of Multilearner Systems

Multiple classifier systems are special cases of approaches that integrate several data-driven models for the *same* problem. A key goal is to obtain a better composite global model, with more accurate and reliable estimates or decisions. In addition, modular approaches often decompose a complex problem into subproblems for which the solutions obtained are simpler to understand, as well as to implement, manage and update.

Multilearner systems have have a rather long and interesting history. For example, Borda counts for combining multiple rankings are named after its 18th century French inventor, Jean-Charles de Borda. Early notable *systems* include Selfridge's Pandemonium [1], a model of human information processing involving multiple demons. Each demon was specialized for detecting specific features or classes. A head-demon (the combiner) would select the demon that "shouted the loudest", a scheme that is nowadays called a "winner-take-all" solution. Nilsson's committee machine [2] combined several linear two-class models to solve a multiclass problem.

A strong motivation for multilearner systems was voiced by Kanal in his classic 1974 paper [3]:

> "It is now recognized that the key to pattern recognition problems does not lie wholly in learning machines, statistical approaches, spatial, filtering,..., or in any other particular solution which has been vigorously

advocated by one or another group during the last one and a half decades as the solution to the pattern recognition problem. No single model exists for all pattern recognition problems and no single technique is applicable to all problems. Rather what we have is a bag of tools and a bag of problems."

This inspired much work in the late seventies on combining linguistic and statistical models, and on combining heuristic search with statistical pattern recognition. Subsequently, similar sentiments on the importance of multiple approaches were also voiced in the AI community, e.g., by Minsky [4]:

> " To solve really hard problems, we'll have to use several different representations.....It is time to stop arguing over which type of pattern-classification technique is best .....Instead we should work at a higher level of organization and discover how to build managerial systems to exploit the different virtues and evade the different limitations of each of these ways of comparing things."

In the 80's, integration of multiple data sources and/or learned models was considered in several disciplines, for example, the combining of estimators in econometrics [5] and evidences in rule-based systems. Especially noteworthy are consensus theoretic methods developed in statistics and management science, including how to produce a single probability distribution that summarizes multiple estimates from different Bayesian experts [6, 7]. The area of decision fusion and multi-sensor data fusion [8] has a rich literature from this era that can be useful for modern day multiclassifier problems as well. Multiple model systems are also encountered in some large engineering systems such as those that demand fault tolerance or employing control mechanisms that may need to function in different operating regimes [9]. In particular, multiple models for nonlinear control has a long tradition [10].

In the data analysis world, hybridization in a broader sense is seen in efforts to combine two or more of neural network, Bayesian, GA, fuzzy logic and knowledge-based systems. The goal is to incorporate diverse sources and forms of information and to exploit the somewhat complementary nature of different methodologies. Since in real-life applications, classification is often not a stand-alone problem but rather a part of a larger system involving optimization, explanation and evaluation of decisions, interaction with the environment, etc., such hybrid approaches will be increasingly relevant as we expand the scope of studies involving multiple classifiers.

## 2 Some Lessons from Multiclassifier Design

Figure 1 shows a generic diagram of the most popular type of multilearner systems studied in the past decade. While data ultimately originates from an underlying universal set $X$, each learner may receive somewhat different subsets of the data for "training" or parameter estimation (as in bagging and boosting), and

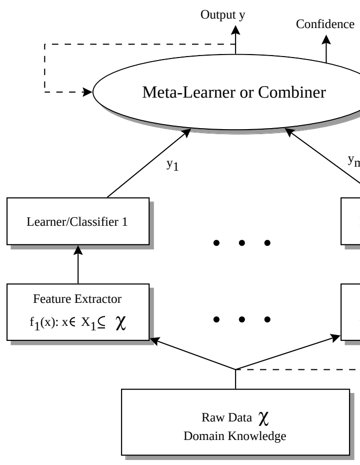

**Fig. 1.** Generic architecture of a multilearner system

may apply different feature extractors ($f$s) on the same raw data. Along with selection of training samples and feature extractors, one needs to decide how many and what types of learners to use, and finally, how to design the meta-learner. There are also larger issues of how to train the components given that they are part of a bigger system, and to estimate the overall gains achievable.

**Ensembles.** The simplest meta-learner is the ensemble or combiner, where the output $y$ is determined solely by the outputs of the individual learners, each trying to solve the same classification or regression problem. The path indicated by the right-most dotted line in Fig. 1 is not used. In the past few years, a host of experimental results from both the neural network and machine learning communities show that combining the outputs of multiple models via voting, (weighted) averaging, order statistics, product rule, entropy, stacking etc., provides a statistically significant improvement in performance along with tighter confidence intervals [11, 12]. A notable study [13] shows that if perfect Bayesian classifiers are learned on distinct sets of extracted features, then a weighted product rule is the optimal combination scheme. The study used sensitivity analysis to show that the product rule is more susceptible to imperfections in the individual classifiers, and a sum (or median) rule turns out to be more reliable in practical situations. An extensive listing of both theoretical and practical works on ensembles or committees circa 1996 is in [14].

Analytical expressions have been developed for both regression [15, 16] and classification[17, 18], to quantitatively estimate the extra accuracy achieved by an ensemble. The seminal work of Hansen and Salamon [17] recognised that the unstable nature of certain neural networks was helpful for ensembles, though it was perhaps too optimistic in anticipating the amount of independence achievable among different networks given a shared training set. An extensive analysis of the gains provided by plurality voting was provided in this paper. In contrast, the analysis of [18] assumes that the individual classifiers provide estimates of the *a posteriori* class probabilities[1]. These estimates are combined by simple averaging. By focussing on the distributions of the *estimated a posteriori* probability functions about the true *a posteriori* probabilities, and how these distributions (and hence the decision boundaries obtained using Bayes rule) are affected when the averaging combiner is used, the following expression was derived under mild assumptions:

$$E_{model}^{ave} = \frac{1 + \delta(m-1)}{m} E_{model} \,, \tag{1}$$

where $E_{model}^{ave}$ and $E_{model}$ are the expected values of the *added* [2] classification error rate for the average-based combiner and for the individual classifiers respectively, $m$ is the number of classifiers combined, and $\delta$ is the average correlation of the errors (in estimating the *a posteriori* probabilities; not to be confused with classification error rate) among the individual classifiers. Note that if $\delta = 0$, signifying that the errors in approximating the *a posteriori* functions are uncorrelated, then the added error can be reduced by a factor of $m$ if an ensemble of $m$ classifiers is used. The Bayes error of course cannot be changed once the input feature space has been selected. In practice, $\delta$ tends to be closer to 1 than to 0 because of (partially) shared training sets and overlapping inductive biases [19], so the gains are not as much.

Dietterich [20] provides an accessible and informal reasoning, from statistical, computational and representational viewpoints, of why ensembles can improve results. Combining is primarily a way of reducing model variance, though in certain situations it also reduces bias. It works best when each learner is well trained, but different learners generalize in different ways, i.e., there is diversity in the ensemble [16].

The impact of diversity is quantified explicitly via the correlation measure $\delta$ in Eq. 1. Diversity may be induced through different presentations of the input data, as in bagging, variations in learner design, or by adding a penalty to the ouputs to encourage diversity. Interestingly, while pre-90's work mostly concentrated on design of the individual learners and the combination scheme, the most spectacular gains have come from smart input selection, as this has the most noticeable impact on diversity.[3] Another point of note is that ensembles are most popular in the machine learning and neural network communities, where the favorite classification models, namely decision trees and MLPs, are relatively unstable - hence leading to greater diversity! In constrast, ensembles are not that commonly applied, for example, to combine multiple robust logistic regression models or other more stable techniques. Finally, given adequately powerful constituent classifiers, there is a sweet spot of training data size for which diversity

[1] for example, using MLP and RBF based classifiers trained with "1-of-C" desired output vectors and a mean squared error or cross-entropy cost function.

[2] i.e., extra error due to imperfect classifiers, incurred in addition to the Bayes error.

[3] This has prompted studies on why bagging and boosting approaches work so well. Attempts to relate them to established statistical approaches are resulting in more sophisticated input selection techniques [21].

through input variation is most effective. For example, if there is too little data, the gains achieved via a bagged ensemble cannot compensate for the decrease in accuracy of individual models, each of which now sees an even smaller training set. On the other end, if the data set is extremely large and computation time is not an issue, even a single flexible classifier can be quite adequate.

My own "discovery" of ensembles was quite serendipitous. In the late 80's, I participated in the DARPA program of sonar signal classification using neural networks. It was soon evident that many different types of features, including Fourier, wavelet and autoregressive coefficients, could be extracted from a given preprocessed time series. Each type of features provided useful discriminatory information, but was not comprehensive by itself. At the same time, it was clear that networks with global hidden units such as sigmoidal functions, generalized quite differently from those with localized hidden units such as gaussian functions. So our solution was to use multiple feature sets as well as multiple classification models [11]. The results from individual classifiers were combined in several ways, including sum, entropy, majority vote and evidential reasoning based approaches [22]. This multiclassifier system gave the best results among all the solutions in this program and was subsequently selected for further development and deployment. Clearly, an ensemble approach was a good idea, and good ideas tend to pre-exist and get re-invented! Indeed, our subsequent literature study turned up much of the historical work described in the introduction, in addition to more contemporanous use of ensembles for other applications [23, 24]. By now, ensembles are being regarded among the most significant advances in pattern classification in the 90's, and the successful series of international workshops on Multiple Classifier Systems started in 2000 is a solid testimony to this fact [25].

**Modular Networks.** If the dotted line on the far right in Fig. 1 is used so that the combining action now depends on the current input, we obtain (soft) modular solutions. In these divide-and-conquer approaches, relatively simple learners get specialized in different parts of the input-output space during the training phase. This specialization enables the use of simpler as well as better models tailored to their localized domain of expertise. The total model is a (possibly soft) union of such simpler models. Techniques of modular learning include "mixtures-of-experts" (MOE), local linear regression, CART/MARS, adaptive subspace models, etc.[26–28]. Note that, in contrast to ensembles, the individual models do not need to perform well across all inputs, but only for their regions of expertise. Modular learning systems are based on the precept that learning a large number of simple local concepts is both easier and more useful than learning a single complex global concept. They are often found to learn concepts more effectively (better performance) and more efficiently (faster learning). Using simpler local models have several added advantages such as easier interpretability, better local tuning or adaptation, easier incorporation of prior knowledge, and less susceptibility to the curse of dimensionality.

Modular approaches till now have been largely confined to regression problems. A good beginning for classification applications is made in [29], where the authors dynamically select one classifier out of an ensemble based on the estimated accuracy in the neighborhood of the test point under consideration. However, unlike in mixtures-of-experts, the procedure does not specialize different classifiers for different regions of the input space during the training process. Rather, all classifiers are trained on the entire input space and selection of a specific model is only made during the test phase. Thus there is scope for further work in specializing classifiers based on soft decomposition of the input space.

**(Back to) The Future.** While there seems more scope for investigating modular multiclassifiers, overall I feel that the understanding of multilearner systems for *static* regression and classification is now quite mature. It is time to use this solid foundation to extend the multilearner framework to qualitatively new and more ambitious domains. There are several promising directions that can be taken. In the next sections, I examine two such directions: consensus clustering and output space decomposition. Both topics have been studied in the past to some extent, but recent application demands and theoretical progress makes it worthwhile to revisit them and expand their scope.

## 3 Combining Multiple Clusterings

Unlike classification problems, there are no well known approaches to combining multiple clusterings. This problem is more difficult than designing *classifier ensembles* since cluster labels are symbolic and so one must also solve a correspondence problem. In addition, the number and shape of clusters provided by the individual solutions may vary based on the clustering method as well as on the particular view of the data presented to that method. Moreover, the desired number of clusters is often not known in advance. In fact, the 'right' number of clusters in a data-set depends on the *scale* at which the data is inspected, and sometimes, equally valid (but substantially different) answers can be obtained for the same data [30].

**History.** A substantial body of largely theoretical work on *consensus classification* exists from the mid-80's and earlier [31]. These studies used the term 'classification' in a very general sense, encompassing partitions, dendrograms and $n$-trees as well. In consensus classification, a profile is a set of classifications which is sought to be integrated into a single consensus classification. A representative work is that of [32], who investigated techniques for strict consensus. Their approach is based on the construction of a lattice over the set of all partitionings by using a refinement relation. Such work on strict consensus works well for small data-sets with little noise and little diversity and obtains solution on a *different* level of resolution. The most prominent application of strict consensus is for the computational biology community to obtain phylogenetic trees [33]. A set of DNA sequences can be used to generate evolutionary trees using criteria such as maximum parsimony, but often one obtains several hundreds of trees with the same score function. In such cases, biologists look for the strict *consensus tree*, the 'infimum', which has lower resolution but is compatible with all the individual trees. Note that such consensus approaches are very domain specific. In particular, (i) they combine non-rooted but hierarchical clusterings, (ii) they use domain specific metrics (e.g. Robinson-Foulds distance) and evaluation criteria such as parsimony, specificity and density, and (iii) *strict* consensus is a requirement.

More general approaches to combining multiple clusterings have started to emerge recently. For example, in [34] a feasible approach to combining distributed agglomerative clusterings is introduced, motivated by distributed data mining scenarios. Another innovative approach is encountered in [35], where multiple, fine-grain $k$-means clusterings are used to determine a co-association matrix of patterns. This matrix represents a derived similarity measure that is then used by an MST algorithm for identifying arbitrary shaped clusters.

## 3.1 Motivations for a Revisit

Why should the problem of integrating multiple clusterings be revisited? First, the works on strict consensus are narrow in scope. They were not meant for large datasets, and these approaches indeed do not scale well. Moreover, in presence of strong noise the results can be trivial, namely the supremum is the monolithic clustering (one cluster) and the infimum is the set of singletons. Another drawback is that the strict consensus is not at the same level of resolution as the original groupings. Second, there are several emerging applications that can benefit from cluster ensembles. We briefly describe three application scenarios below.

**Quality and Robustness**. Combining several clusterings can lead to *improved quality* and robustness of results. As compared to classification one often finds even more variability in clustering results for difficult data sets. This increased level of diversity means that the potential gains from employing ensembles is higher than that for classification problems of comparable difficulty. In the clustering context, diversity can be created in numerous ways, including: (i) using different features to represent the objects, (ii) varying the number and/or location of initial cluster centers in iterative algorithms such as $k$-means, (iii) varying the order of data presentation in on-line methods such as BIRCH, and (iv) using a portfolio of very different clustering algorithms.

A different but related motivation for using a cluster ensemble is to build a *robust*, diverse clustering portfolio that can perform well over a wide range of data-sets with little hand-tuning.

**Knowledge Reuse**. Another important consideration is the *reuse of existing clusterings*. In several applications, a variety of clusterings for the objects under consideration may already exist. For example, on the web, pages are categorized e.g., by Yahoo! (according to a manually-crafted taxonomy), by your Internet service provider (according to request patterns and frequencies) and by your personal bookmarks (according to your preferences). Can we reuse such pre-existing knowledge to create a single consolidated clustering? Knowledge reuse [36] in this context means that we exploit the information in the provided cluster labels *without* going back to the *original features* or the algorithms that were used to create the clusterings.

**Distributed Computing**. The ability to deal with clustering in a distributed fashion is becoming increasingly important since real applications nowadays often involve distributed databases. In several situations it may not be feasible to collect all the data into a single flat file, because of the computational, bandwidth and storage costs, or because of a variety of practical reasons including security, privacy, proprietary nature of data, need for fault tolerant distribution of data and services, real-time processing requirements, statutory constraints imposed by law, etc [37]. So, in a distributed computing scenario, each clusterer may have access to only some of the objects, or see only a limited number of features or attributes of each object. How can one perform distributed clustering and combining of results under such situations?

## 3.2 Cluster Ensembles: A Knowledge Reuse Framework

Clearly, there are many approaches and issues in combining multiple clusterings. In this section, we summarize some results from our recent work [38] on one specific formulation: the *combination* of multiple *partitionings* of the same underlying set of objects *without accessing the original features*. Since only cluster labels are available to the combiner, this is a framework for knowledge reuse [36].

If we number the $k$ clusters as $1, .., k$, then a given clustering can be denoted by a label vector $\lambda \in \mathbb{N}^n$, after imposing an arbitrary order on the $n$ objects that are being clustered. The first issue to address is how to evaluate a consensus clustering. Note that there is no ground truth to measure against. Intuitively, if there is no other apriori knowledge, then the best one can do is to extract the commonalities among the different clusterings. This suggests that mutual information, a symmetric measure that quantifies the statistical information shared between two distributions, is the natural measure of the consensus quality. Suppose there are two labelings $\lambda^{(a)}$ and $\lambda^{(b)}$. Let there be $k^{(a)}$ groups in $\lambda^{(a)}$ and $k^{(b)}$ groups in $\lambda^{(b)}$. Let $n^{(h)}$ be the number of objects in cluster $\mathcal{C}_h$ according to $\lambda^{(a)}$, and $n_\ell$ the number of objects in cluster $\mathcal{C}_\ell$ according to $\lambda^{(b)}$. Let $n_\ell^{(h)}$ denote the number of objects that are in cluster $h$ according to $\lambda^{(a)}$ as well as in group $\ell$ according to $\lambda^{(b)}$. Then, a [0,1]-normalized mutual information criterion $\phi^{(\mathrm{NMI})}$ is computed as follows [39]:

$$\phi^{(\mathrm{NMI})}(\lambda^{(a)}, \lambda^{(b)}) = \frac{2}{n} \sum_{\ell=1}^{k^{(a)}} \sum_{h=1}^{k^{(b)}} n_\ell^{(h)} \log_{k^{(a)} \cdot k^{(b)}} \left( \frac{n_\ell^{(h)} n}{n^{(h)} n_\ell} \right) \tag{2}$$

We propose that the optimal combined clustering be defined as the one that has maximal average mutual information with all individual labelings, given that the number of consensus clusters desired is $k$.

**Efficient Consensus Functions.** In [38], three efficient heuristics are proposed to solve the cluster ensemble problem. All algorithms approach the problem by first transforming the set of clusterings into a hypergraph representation. Simply put, each cluster is considered as a hyperedge connecting all its members (vertices). The hyperedges obtained from different clusterings are all added to a common graph, which thus has $n$ vertices and $\sum_{q=1}^{r} k^{(q)}$ hyperedges.

The simplest heuristic is to define a similarity measure between two objects as the fraction of clusterings in which these objects are in the same cluster. The resulting matrix of pairwise similarities can be used to recluster the objects using any reasonable similarity-based clustering algorithm. The second heuristic looks for a hyperedge separator that partitions the hypergraph into $k$ unconnected components of approximately the same size, using a suitable hypergraph partitioning package such as HMETIS. The idea behind the third heuristic is to group and collapse related hyperedges into $k$ meta-hyperedges. The hyperedges that are considered related for the purpose of collapsing are determined by a graph-based clustering of hyperedges. Finally, each object is assigned to the collapsed hyperedge in which it participates most strongly.

It turns out that the first and third approaches typically do better than the second. Since the third approach is much faster than the first, it is preferred. However, note that our objective function has an added advantage that it allows one to add a stage that selects the best consensus function without any supervision information, by simply selecting the one with the highest NMI. So, for the results reported later, we simply use this 'supra'-consensus function $\Gamma$, obtained by running *all three* algorithms, and selecting the one with the greatest score.

**Applications and Results.** For brevity, we just illustrate one application of cluster ensembles, namely how it can be used to boost quality of results by combining a set of clusterings obtained from partial views of the data. This scenario is motivated by certain distributed data mining situations in which it is not feasible to collect all the features at one central location. Results on a wider range of application scenarios and data sets can be found in [38].

For our experiments, we simulate such a scenario by running several clusterers, each having access to only a restricted, small subset of features. The clusterers find groups in their views/subspaces. In the combining stage, individual results are integrated to recover the full structure of the data (without access to any of the original features). Results are provided on two real data sets: (i) `PENDIG` from the UCI ML repository, is for pen-based recognition of handwritten digits from 16 spatial features. There are ten classes of roughly equal size corresponding to the digits 0 to 9. (ii) `YAHOO` represents 2340 documents from 20 news categories, and is available from `ftp://ftp.cs.umn.edu/dept/users/boley/`. After standard preprocessing, each document is represented by a 2903-dimensional vector. These two data sets are partitioned into 10 and 40 clusters respectively by each clustering algorithm. For this experiment, the individual clusterers are all graph-partitioning based (as they are quite robust and give comparable sized clusters), using a domain-appropriate similarity function, namely, Euclidean distance for `PENDIG` and cosine similarity for `YAHOO`. Table 1 summarizes the results, averaged over 10 runs. For example, in the `YAHOO` case, 20 clusterings were performed in 128-dimensions (occurrence frequencies of 128 randomly chosen words) each. The average quality amongst the results was 0.17 and the best quality was 0.21. Using the supra-consensus function to combine all 20 labelings results in a quality of 0.38, or 124% higher mutual information than the average individual clustering. The results indicate that, when processing on the all features is not possible but multiple, limited views exist, a cluster ensemble can significantly boost results compared to individual clusterings.

| data | subspace #dims | #models $r$ | **quality of consensus** $\phi^{(\mathrm{NMI})}(\kappa, \lambda)$ | max. individual quality $\max_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$ | ave. individual quality $\mathrm{avg}_q \phi^{(\mathrm{NMI})}(\kappa, \lambda^{(q)})$ |
|---|---|---|---|---|---|
| PENDIG | 4 | 10 | **0.59009** | 0.53197 | 0.44625 |
| YAHOO | 128 | 20 | **0.38167** | 0.21403 | 0.17075 |

**Table 1.** Effectiveness of consensus clustering for integrating multiple clusterings based on partial feature views.

## 4 Output Space Decomposition [40]

When one is faced with a $C > 2$ class problem, it is often preferable to break it down into multiple sub-problems, each involving less than $C$ classes or meta-classes, where a metaclass, $\Omega$, is formed by the union of two or more of the original classes. This approach entails a decomposition of the output space, i.e., the space of target classes, and has to deal with the issue of how to combine the answers from the sub-problems to yield a solution for the original $C$-class problem. A major motivation for this approach is that the sub-problems are typically much simpler to solve. In addition, feature selection/extraction can be tailored to each individual sub-problem. We shall see later that output space decomposition can also lead to extraction of valuable domain knowledge such as the relationships and natural hierarchies among different classes, the most discriminative features for a given pair of classes, etc.

**Brief History.** Both the Pandemonium (1958) and the Learning Machine (1965) models mentioned in the introduction involve output space decomposition. Nilsson's machine trained one linear discriminant function per class. For a given test input, the class with the highest discriminant value was assigned as the predicted label. This machine partitions the input space using hyperplanes that all intersect at a point. This approach has since been extended to include quadratic discriminants and kernel discriminants. Note that they all involve $C$ two-class problems with a simple "max" combination function, and are attractive when the individual classifiers are simple. Even with the advent of more general classifiers, this methodology is helpful if either $C$ is large or the class boundaries are quite complex. For example, it has been shown that in several cases, building $C$ MLP-based models, one each for discriminating a specific class from all the rest, outperforms a single complex model for discriminating all the classes simultaneously [41].

In the 70's and 80's, there were some works in the pattern recognition community that used multiclassifiers arranged in tandem or as a tree-structured hierarchy, and exploited output space decomposition[42]. One interesting technique was to decompose a $C$-class problem into $\binom{C}{2}$ two-class problems, one for each unique pair of classes. A naive way of labeling a test sample is to let each of the $\binom{C}{2}$ vote for the more likely of its two classes, and then sum up all the votes to determine the winner. A more sophisticated approach [43] iteratively re-estimates the overall posterior probabilities from the $\binom{C}{2}$ pairwise posterior probabilities. We have used this technique to address a 14 feature, 26-class digit recognition problem and a 14 feature, 11-class remote sensing problem [44]. What was most remarkable was that, in both applications, certain pairs of classes could be distinguished by only using 2 or 3 features, even though the entire $C$-class problem needed all the input features. But a drawback of this approach is that $O(C^2)$ classifiers are needed, which becomes impractical if the number of classes is in the tens or more.

In the machine learning community, a seminal work involving output space decomposition is called error-correcting output coding [45]. Each member of an ensemble solves a 2-class problem obtained by partitioning the original classes into two groups (meta-classes) either randomly or based on error correcting codes. A simple voting scheme is then used by the ensemble to label a test input. This technique has the advantage that the number of classifiers required can be varied and need not be quadratic in $C$. It also retains another advantage of output decomposition, namely that feature selection can be tailored for each 2-class problem [46]. However, the output partitioning, being random, fails to detect or exploit any natural affinities among different classes. Consequently, some of the two-class problems may be quite complicated as very different classes can get grouped together.

**Prognosis.** A very promising direction to explore is the design of multiple classifiers organized as a hierarchy. A good example of the benefits of this approach is provided in [47], where a hierarchical topic taxonomy is used to effectively organize large text databases. Recently, we introduced a hierarchical technique to recursively decompose the output space into a binary tree [48]. The resulting architecture is illustrated in Fig. 2. It requires that only $C-1$ two-(meta) class problems need to be solved.

The hierarchical ensemble is built using a top-down approach based on a generalized associative modular learning approach. Initially, a randomly selected class is fully associated with a metaclass (to seed the partitioning) while all other classes are equally associated with both metaclasses. An EM type iterative procedure is used to update the metaclass parameters for the current class associations, and then update the associations based on the new metaclass parameter values. Note that the selection of features that best discriminate the resulting meta-classes can be concurrently updated as well. Using ideas from deterministic annealing, a temperature parameter can be used to slowly converge the associations to hard partitions in order to induce specialization and decoupling among the modules. The overall approach produces class groupings that are more natural in the sense of being highly likely to conform well with human domain experts' opinions.

The hierarchical ensemble technique described above was applied to the important remote sensing problem of determining the types of land-cover from $\sim$200 dimensional hyperspectral images. These are typically about $\sim$10 class

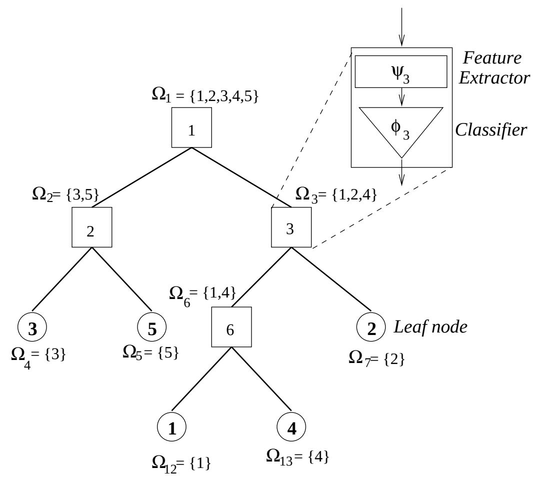

**Fig. 2.** Hierarchical decomposition of a 5 class problem into four 2-(meta)class problems. All the feature extractors ($\psi$s) as well as the classifiers ($\phi$s) are automatically determined as part of the decomposition process.

problems. In coastal regions, we found that the first split would invariably separate land classes from water classes, the land classes would get further split between uplands and wetlands, and so on. Equally important, custom discriminatory features were obtained for the sub-problems, adding to our domain knowledge.

The real payoff of such extracted knowledge will come as more and more of the earth's surface get mapped. Currently, hyperspectral classification is done based on training and test samples drawn from the same image, since even a single image generates megabytes of data. In a given image, not all classes are present, and there are different mixtures of classes in different images. Due to such changing compositions, when one attempts to apply a classfier to a new image, the results are often much weaker since a fundamental assumption of data-driven modeling, namely that the training and test samples are drawn uniformly from a common distribution, is violated. The extracted domain knowledge from hierarchical ensembles suggests a way out of this dilemma. If one organizes this information properly, then, given data from a new area, one can quickly estimate what features to extract and examine, and what classes to anticipate, even with very little data from this new region. This can greatly reduce the amount of data (both labelled and unlabelled) needed, with little compromise in the results. It also provides a nice platform for integrating semi-supervised learning ideas into a multiclassifier framework, yet another topic worth exploring!

## 5 Concluding Remarks

Besides the two directions described in this article, there are several more venues worthy of exploration for scholars interested in multilearner systems. For example, virtually all the work on ensembles and mixture of experts to date is on static problems, where the output is solely a function of the current input. Suppose we are interested in classifying variable length sequences of vectors, for example, those representing gene expressions or acoustic signals. Does temporal information have to be dealt with at the pre-processing step since our ensembles cannot handle them directly, or can memory mechanisms be incorporated into the meta-learner? What if one member of an ensemble is ready to make a decision after observing a subsequence while others want to see more of the sequence?

Another interesting situation arises when one has to solve a series of different but related classification tasks, either simultaneously or over time, rather than a single task. This issue is becoming increasingly relevant due to the ever increasing and continual generation of data and problems thanks to the growing Internet, advances in the human genome project, evolving financial markets, etc. Some advances on this topic have already been made under categories such as 'life-long learning', 'learning to learn', and 'knowledge reuse' [49, 36]. From a multiclassifier point of view, in this case the individual members of an "ensemble" may not be trying to solve the same task or be trained simultaneously, both *significant departures* from the traditional framework. Clearly there are enough venues that can be fruitfully explored, and I look forward to a continued stream of exciting research on multiclassifier systems in the near future.

**Acknowledgements:** Parts of this article reflect joint work with Kagan Tumer, Shailesh Kumar and Alexander Strehl, and I thank them for their inputs. This research was supported by the NSF Grant ECS-9900353, and by Intel and IBM.

## References

1. Selfridge, O.G.: Pandemonium: a paradigm for learning. Proc. of Symp. held at the National Physical Lab. (1958) 513–526
2. Nilsson, N.J.: Learning Machines: Foundations of Trainable Pattern-Classifying Systems. McGraw Hill, NY (1965)
3. Kanal, L.: Patterns in pattern recognition. IEEE Trans. Information Theory **IT-20** (1974) 697–722
4. Minsky, M.: Logical versus analogical or symbolic versus connectionist or neat versus scruffy. AI Magazine **12** (1991) 34–51
5. Granger, C.W.J.: Combining forecasts–twenty years later. Journal of Forecasting **8** (1989) 167–173
6. French, S.: Group consensus probability distributions: A critical survey. In Bernardo et al., J.M., ed.: Bayesian Statistics 2. North Holland, New York (1985)
7. Benediktsson, J.A., Swain, P.H.: Consensus theoretic classification methods. IEEE Transactions on Systems, Man, and Cybernetics **22** (1992) 688–704

8. Luo, R.C., Kay, M.G.: Multisensor integration and fusion in intelligent systems. IEEE Transactions on Systems, Man, and Cybernetics **19** (1989) 901–931
9. Narendra, K., Balakrishnan, J., Ciliz, K.: Adaptation and learning using multiple models, switching and tuning. IEEE Control Systems Magazine (1995) 37–51
10. Murray-Smith, R., Johansen, T.A.: Multiple Model Approaches to Modelling and Control. Taylor and Francis, UK (1997)
11. Ghosh, J., Deuser, L., Beck, S.: A neural network based hybrid system for detection, characterization and classification of short-duration oceanic signals. IEEE Jl. of Ocean Engineering **17** (1992) 351–363
12. Sharkey, A.: On combining artificial neural networks. Connection Science **8** (1996) 299–314
13. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Trans. Pattern Analysis and Machine Intelligence **20** (1998) 226–239
14. Tumer, K.: Linear and order statistics combiners for reliable pattern classification. PhD thesis, Dept. of ECE, Univ. of Texas at Austin, Dec. (1996)
15. Perrone, M.P.: Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization. PhD thesis, Brown University (1993)
16. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation and active learning. In G. Tesauro, D.T., Leen, T., eds.: Advances in Neural Information Processing Systems-7. (1995) 231–238
17. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 993–1000
18. Tumer, K., Ghosh, J.: Analysis of decision boundaries in linearly combined neural classifiers. Pattern Recognition **29** (1996) 341–348
19. Tumer, K., Ghosh, J.: Ensemble correlation and error reduction in ensemble classifiers. Connection Science **8** (1996) 385–404
20. Dietterich, T.G.: Ensemble methods in machine learning. In Kittler, J., Roli, F., eds.: Multiple Classifier Systems. LNCS Vol. 1857, Springer (2001) 1–15
21. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Technical Report Dept. of Statistics, Stanford University. (1998)
22. Ghosh, J., Beck, S., Chu, C.: Evidence combination techniques for robust classification of short-duration oceanic signals. In: SPIE Conf. on Adaptive and Learning Systems, SPIE Proc. Vol. 1706. (1992) 266–276
23. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man and Cybernetics **22** (1992) 418–435
24. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (1994) 66–76
25. Kittler, J., Roli (Eds.), F.: Multiple Classifier Systems. LNCS Vol. 1857, Springer (2001)
26. Jordan, M., Jacobs, R.: Hierarchical mixture of experts and the EM algorithm. Neural Computation **6** (1994) 181–214
27. Holmstrom, L., Koistinen, P., Laaksonen, J., Oja, E.: Neural and statistical classifiers - taxonomy and two case studies. IEEE Transactions on Neural Networks **8** (1997) 5–17
28. Ramamurti, V., Ghosh, J.: Structurally adaptive modular networks for nonstationary environments. IEEE Transactions on Neural Networks **10** (1999) 152–60

29. Woods, K., Kegelmeyer, W.P., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 405–410
30. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, New Jersey (1988)
31. Barthelemy, J.P., Laclerc, B., Monjardet, B.: On the use of ordered sets in problems of comparison and consensus of classifications. Jl. of Classification **3** (1986) 225–256
32. Neumann, D.A., Norton, V.T.: Clustering and isolation in the consensus problem for partitions. Journal of Classification **3** (1986) 281–298
33. Kim, J., Warnow, T.: Tutorial on phylogenetic tree estimation. In: Intelligent Systems for Molecular Biology, Heidelberg. (1999)
34. Johnson, E., Kargupta, H.: Collective, hierarchical clustering from distributed, heterogeneous data. In Zaki, M., Ho, C., eds.: Large-Scale Parallel KDD Systems. Volume 1759 of LNCS., Springer-Verlag (1999) 221–244
35. Fred, A.L.N., Jain, A.K.: Data clustering using evidence accumulation. In: Proc. ICPR. (2002) to appear
36. Bollacker, K.D., Ghosh, J.: Effective supra-classifiers for knowledge base construction. Pattern Recognition Letters **20(11-13)** (1999) 1347–52
37. Kargupta, H., Chan, P., eds.: Advances in Distributed and Parallel Knowledge Discovery. AAAI/MIT Press, Cambridge, MA (2000)
38. Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining partitionings. In: Proceedings of AAAI 2002, Edmonton, Canada, AAAI (2002) in press.
39. Strehl, A., Ghosh, J., Mooney, R.: Impact of similarity measures on web-page clustering. In: Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin, AAAI (2000) 58–64
40. Kumar, S.: Modular learning through output space decomposition. PhD thesis, Dept. of ECE, Univ. of Texas at Austin, Dec. (2000)
41. Anand, R., Methrotra, K., Mohan, C.K., Ranka, S.: Efficient classification for multiclass problems using modular neural networks. IEEE Transactions on Neural Networks **6** (1995) 117–125
42. Berenstein, C., Kanal, L.N., Lavine, D.: Consensus rules. In Kanal, L.N., Lemmer, J.F., eds.: Uncertainty in Artificial Intelligence. North Holland, New York (1986)
43. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In Jordan, M.I., Kearns, M.J., Solla, S.A., eds.: Advances in Neural Information Processing Systems. Volume 10., The MIT Press (1998)
44. Kumar, S., Crawford, M.M., Ghosh, J.: A versatile framework for labelling imagery with a large number of classes. In: Proc. IJCNN. (1999)
45. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Jl. of Artificial Intelligence Research **2** (1995) 263–286
46. Ricci, F., Aha, D.: Extending local learners with errorcorrecting output codes. Technical report, Technical Report 9701-08, IRST (1997)
47. Chakrabarti, S., Dom, B., Agrawal, R., Raghavan, P.: Scalable feature selection, classication and signature generation for organizing large text databases into hierarchical topic taxonomies. VLDB Journal **7** (1998) 163–178
48. Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis", Pattern Analysis and Applications, spl. Issue on Fusion of Multiple Classifiers **5** (2002) In Press
49. Thrun, S., Pratt, L.: Learning To Learn. Kluwer Academic, Norwell, MA (1997)