Classifier Ensembles: Select Real-World Applications

Nikunj C. Oza

NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035-1000
oza@email.arc.nasa.gov

Kagan Tumer

Oregon State University
204, Rogers Hall
Corvallis, OR 97331
kagan.tumer@oregonstate.edu

Abstract

Broad classes of statistical classification algorithms have been developed and applied successfully to a wide range of real world domains. In general, ensuring that the particular classification algorithm matches the properties of the data is crucial in providing results that meet the needs of the particular application domain. One way in which the impact of this algorithm/application match can be alleviated is by using ensembles of classifiers, where a variety of classifiers (either different types of classifiers or different instantiations of the same classifier) are pooled before a final classification decision is made. Intuitively, classifier ensembles allow the different needs of a difficult problem to be handled by classifiers suited to those particular needs. Mathematically, classifier ensembles provide an extra degree of freedom in the classical bias/variance tradeoff, allowing solutions that would be difficult (if not impossible) to reach with only a single classifier. Because of these advantages, classifier ensembles have been applied to many difficult real world problems. In this paper, we survey select applications of ensemble methods to problems that have historically been most representative of the difficulties in classification. In particular, we survey applications of ensemble methods to remote sensing, person recognition, one vs. all recognition, and medicine.

Key words: Classifier ensembles, ensemble applications.
PACS:

1 Introduction

Classifying a set of inputs into one of many classes is one of the most basic statistical pattern recognition tasks. A classification task requires the construction of a statistical model that represents a mapping from input data (normally described by several features) to the appropriate outputs. This model is intended to approximate the true mapping from the inputs to the outputs, typically with the intent of generating predictions of outputs for new, previously unseen inputs. Applications of such classification or pattern recognition algorithms cover a broad array of domains, ranging from medical diagnosis, to fraud detection to remote sensing.

In general, there are two types of learning approaches for pattern recognition: supervised and unsupervised. In supervised learning, a set of training examples—examples with known output values—is used by a learning algorithm to create a classifier. For example, the classification task may be to learn to predict whether a particular credit card charge is legitimate or fraudulent. Each pattern (input) corresponds to a particular charge and is composed of multiple features (e.g., time of transaction, amount of transaction, average daily credit card balance, store in which transaction was made) as well as actual information on whether this charge was fraudulent (output). A learning algorithm uses the supplied examples to generate a classifier that approximates the mapping between each transaction and the legitimacy of that transaction. This classifier can then be used to predict whether a new transaction is legitimate or not.

An unsupervised classification task on the other hand consists of assigning a class to a training example without having a known target class. For example, different credit card transactions can be grouped together based on similarity. This information can then be used to screen transactions that may need to be reviewed by another algorithm or a human operator. This approach, also known as clustering provides value in finding correlations and similarities in large data sets where the actual class memberships of the training patterns are not known.

Many learning algorithms generate a single classifier (e.g., a decision tree or neural network) that can be used to make predictions for new examples. However, many decisions (e.g., initial model parameter settings) affect the performance of that classifier. Selecting the best available classifier is an option, but because the distribution over new examples that the classifier may encounter during operation may vary (slightly or significantly depending on the application), this approach does not provide the best solution in all cases. Furthermore, because many classifiers are generally tried before a single classifier is selected, this approach also discards valuable information by ignoring the

performance of all the other classifiers.

Classifier ensembles–also known as combiners or committees–are aggregations of several classifiers whose individual predictions are combined in some manner (e.g., averaging or voting) to form a final prediction. Because they use all the available classifier information, ensembles generally provide better and/or more robust solutions in most applications. As an example, consider the case where neural networks with different structure (or with just a different set of starting weights) were generated from a supplied training set. Now, an ensemble of these three classifiers can be formed by having each classifier provide a prediction for a given pattern and returning the class that gets the maximum number of votes. Many researchers have demonstrated that ensembles often outperform their *base* models (the component models of the ensemble) if the base models perform well on novel examples and tend to make errors on different examples (e.g., [12,81,52,114,117]).

In this article we provide a summary of the leading ensemble methods and provide a discussion of their application to four broad classes of real world classification problems. In Section 2, we present a motivating example, followed by the bias/variance tradeoff and Bayesian interpretation of classifier ensembles. In Section 3 we present the background on the most commonly used ensemble techniques, including basic averaging, weighted averaging, bagging, boosting and order statistics ensembles. In Section 4, we present four different domains where classifier ensembles have been successfully applied. These domains are remote sensing, person recognition (face,fingerprint), one-vs-all classification (which covers many types of fault and intrusion detection), and medicine. Finally, in Section 5 we discuss the impact of these results and highlight future research and application directions for classifier ensembles.

2 Motivation and Mathematical Insight

2.1 Motivating Example

To intuitively show the impact of ensembles, let us define h_1, h_2, h_3 to be the three neural networks in the previous example and consider a new example x. If all three networks always agree, then whenever $h_1(x)$ is incorrect, $h_2(x)$ and $h_3(x)$ will also be incorrect, so that the incorrect class will get the majority of the votes and the ensemble will also be incorrect—having an ensemble of three networks provides no benefit over only one network. On the other hand, if the networks tend to make errors on different examples, then when $h_1(x)$ is incorrect, $h_2(x)$ and $h_3(x)$ may be correct, so that the ensemble will return the correct class by majority vote. More precisely, if an ensemble has

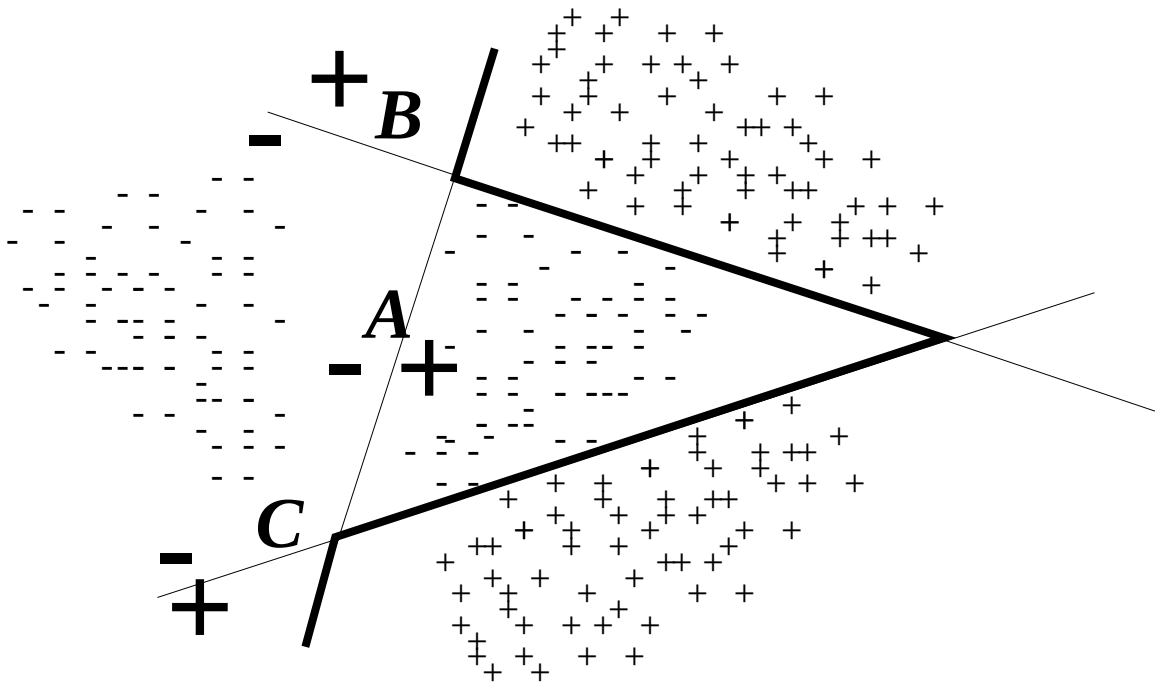

Fig. 1. An ensemble of linear classifiers. Each line A, B, and C is a linear classifier. The boldface line is the ensemble that classifies new examples by returning the majority vote of A, B, and C.

M base models having an error rate $\epsilon < 1/2$ and if the base models' errors are independent, then the probability that the ensemble makes an error is the probability that more than $M/2$ base models misclassify the example. This is precisely $P(B > M/2)$, where B is a $Binomial(M, \epsilon)$ random variable. In our three-network example, if all the networks have an error rate of 0.3 and make independent errors, then the probability that the ensemble misclassifies a new example is 0.21. Even better than base models that make independent errors would be base models that are somewhat anti-correlated. For example, if no two networks make a mistake on the same example, then the ensemble's performance will be perfect because if one network misclassifies an example, then the remaining two networks will correct the error.

We can see the intuition behind this point graphically in Figure 1. The goal of the learning problem depicted in the figure is to separate the positive examples ('+') from the negative examples ('−'). The figure depicts an ensemble of three linear classifiers. For example, line C classifies examples above it as negative examples and examples below it as positive examples. Note that none of the three lines separates the positive and negative examples perfectly. For example, line C misclassifies all the positive examples in the top half of the figure. Indeed, no straight line can separate the positive examples from the negative examples. However, the ensemble of three lines, where each line gets one vote, correctly classifies all the examples—for every example, at least two of the three linear classifiers correctly classifies it, so the majority is always

correct. This is the result of having three very different linear classifiers in the ensemble. This example clearly depicts the need to have base models whose errors are not highly correlated. If all the linear classifiers make mistakes on the same examples (for example if the ensemble consisted of three copies of line A), then a majority vote over the lines would also make mistakes on the same examples, yielding no performance improvement.

Another way of explaining the superior performance of the ensemble is that the class of ensemble models has greater expressive power than the class of individual base models. We pointed out earlier that, in the figure, no straight line can separate the positive examples from the negative examples. Our ensemble is a piecewise linear classifier (the bold line), which is able to perfectly separate the positive and negative examples. This is because the class of piecewise linear classifiers has more expressive power than the class of single linear classifiers.

2.2 Bias vs. Variance

The intuition that we have just described has been formalized [109,113]. Ensemble learning can be justified in terms of the *bias* and *variance* of the learned model [39,118]. The bias of a model is the difference between the true function that generated the data and the "average" function returned by the learning algorithm (averaged over all possible training sets). The variance of the model is the variance over the possible training sets of the function returned by the learning algorithm.

Intuitively, the simpler the space of models, the higher the bias and the lower the variance. Simpler models may be too simple to fit the variations present in the data, so systematic errors may remain regardless of the amount of training performed—this is bias. On the other hand, simpler model spaces are smaller than more complicated model spaces; therefore, variations in the training set are less likely to introduce significant changes in the returned model; therefore, the variance will be lower. An extreme example of this is a learning algorithm that always returns a particular model regardless of the training set presented as input. This will clearly have a high bias because it does not attempt to fit the data at all. However, it will have low variance (zero, in fact) because there is only one model, so there will be no variation over different training sets.

2.3 Bayesian Interpretation

Ensemble learning can also be seen as a tractable approximation to full Bayesian learning [11,26,50]. In full Bayesian learning, the final learned model is a mix-

ture of a very large set of models—typically all models in a given family (e.g., all decision trees up to a certain depth). If we are interested in predicting some quantity Y, and we have a set of models h_m ($m \in \{1, 2, \ldots, M\}$ for some presumably large M) and a training set T, then the final learned model is

$$P(Y|T) = \sum_{m=1}^{M} P(Y|h_m)P(h_m|T) = \sum_{m=1}^{M} P(Y|h_m)\frac{P(T|h_m)P(h_m)}{P(T)}.$$

Full Bayesian learning combines the explanatory power of all the models ($P(Y|h_m)$) weighted by the posterior probability of the models given the training set ($P(h_m|T)$). However, full Bayesian learning is intractable because it uses a very large (possibly infinite) set of models. Ensembles can be seen as approximating full Bayesian learning by using a mixture of a small set of the models having the highest posterior probabilities ($P(h_m|T)$) or highest likelihoods ($P(T|h_m)$). Ensemble learning lies between traditional machine learning with single models and full Bayesian learning in that it uses an intermediate number of models.

3 Ensemble Methods

The concept of ensembles appeared in the classification literature as early as 1965 [79], and has subsequently been studied in several forms, including *stacking* [117], *boosting* [24,25,36,37], *bagging* [12,13] model averaging [83] and forecast combining [45].

Some ensemble combination techniques such as majority voting can generally be applied to any type of classifier, while others rely on specific outputs, or specific interpretations of the output. The concept of voting is both simple to implement and appealing [4,5,13,18,47,82]. It is in fact the simplest ensemble scheme: given a new pattern x, each classifier votes for a target class. The class that gets the highest number of votes is selected. The rationale for averaging is based on the result that the outputs of parametric classifiers that are trained to minimize a cross-entropy or mean square error (MSE) function, given "*one-of-L*" desired output patterns, approximate the *a posteriori* probability densities of the corresponding class [90,92]. In particular, the MSE is shown to be equivalent to:

$$MSE = K_1 + \sum_{i=1}^{L} \int_x D_i(x) \left(p(C_i|x) - h_i(x)\right)^2 dx$$

where K_1 and $D_i(x)$ depend on the class distributions only, $h_i(x)$ is the classifier output intended to approximate the probability of class C_i given an output x, $p(C_i|x)$ denotes the true posterior probability of class C_i and the summation is over all classes [99]. Thus minimizing the MSE is equivalent to a weighted least squares fit of the network outputs to the corresponding posterior probabilities [47,55,67].

3.1 Simple Averaging

One of the most popular ways of combining multiple classifiers is through simple averaging of the corresponding output values [46,71,83,108,111]. If M classifiers $(h_i^m(x), m \in \{1, 2, \ldots, M\})$are available, the class C_i output of the averaging combiner is:

$$h_i^{ave}(x) = \frac{1}{N} \sum_{m=1}^{M} h_i^m(x), \tag{1}$$

This simple averaging achieves most of the benefits of ensembles in that it reduces the variance of the estimate of the output class posteriors [74,108]. Because of its simplicity it has been widely applied to real world problems [63,97,110]. Though considered "too simple" to be effective, many detailed studies have shown simple averaging to be a very effective ensemble method, particularly in large complex data sets [110,119].

For classification problems, simple averaging reduces the model error (the error in addition to the Bayes error) by a factor of M for M independent, unbiased classifiers [55,109,110]. [1] If the classifiers are not independent, the reduction in model error for an average classifier can be expressed as [110]:

$$E_{model}^{ensemble} = \frac{1 + \rho(M-1)}{M} \, E_{model} \, . \tag{2}$$

where ρ is the average correlation among the *errors* of the different classifiers. The model error reduction depends on the correlation among the classifiers. If the classifiers are uncorrelated, then the reduction by a factor of M holds. If the classifiers are fully correlated, then as common sense would dictate, there is no improvement. This equation highlights the need for selecting classifiers that have uncorrelated errors.

[1] This result has an equivalent for regression estimators where a reduction in Mean Square Error by a factor of M can be obtained by pooling M independent estimators [83].

3.2 Weighted Averaging

Weighted averaging has also been proposed, along with different methods of computing the proper classifier weights [8,48,71,76,83]:

$$h_i^{ave}(x) = \frac{1}{M} \sum_{m=1}^{M} w_m h_i^m(x), \tag{3}$$

Weighted averaging is mathematically similar to simple averaging, but due to the added degrees of freedom, can in some instances provide better solutions. In practice though, it has often failed to provide improvement to justify its added complexity, especially when there is limited training data with which the weights can be properly estimated [110].

3.3 Stacking

Stacking introduced a new concept to the ensemble literature in that it actively seeks to improve the performance of the ensemble by "correcting" the errors [12,117]. Stacked generalization addresses the issue of classifier bias with respect to a training set, and aims at learning and using these biases to improve classification. The main concept is to use a new classifier to correct the errors of a previous classifier (hence the classifier are "stacked" on top of one another).

The simple idea behind stacking is that if an input-output pair (x, y) is left out of the training set of h_i, after training has been completed for h_i, the output y can still be used to assess the model's error. In fact, since (x, y) was not in the training set of h_i, $h_i(x)$ may differ from the desired output y. A new classifier then can be trained to estimate this discrepancy, given by $y - h(x)$. In essence, a second classifier is trained to learn the errors the first classifier has made. Adding the estimated errors to the outputs of the first classifier can provide an improved final classification decision.

3.4 Bagging

Earlier we discussed the concept of combining classifiers through voting. Although used sparingly, this concept really took hold when Breiman introduced Bootstrapped Aggregating (Bagging), which combined voting with a method for generating the classifiers that provide the votes. The simple idea was based

Bagging(T,M)
For each $m = 1, 2, \ldots, M$,
$T_m = Sample\_With\_Replacement(T, |T|)$
$h_m = L_b(T_m)$
Return $h_{fin}(x) = \text{argmax}_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y)$

Sample_With_Replacement(T,N)
$S = \emptyset$
For $i = 1, 2, \ldots, N$,
$r = random\_integer(1, N)$
Add $T[r]$ to S.
Return S.

Fig. 2. Batch Bagging Algorithm and Sampling with Replacement: T is the original training set of N examples, M is the number of base models to be learned, L_b is the base model learning algorithm, the h_i's are the classification functions that take a new example as input and return the predicted class from the set of possible classes Y, $random\_integer(a, b)$ is a function that returns each of the integers from a to b with equal probability, and $I(A)$ is the indicator function that returns 1 if event A is true and 0 otherwise.

on allowing each base classifier to be trained with a different random subset of the patterns with the goal of bringing about diversity in the base classifiers.

Devising different ways of generating base classifiers that perform well but are diverse (i.e., make different errors) has historically been one of the most active subtopics within ensemble methods research. Bagging is a simple example of one such method of generating diverse base classifiers; therefore, we discuss it in more detail here. Bagging generates multiple bootstrap training sets from the original training set and uses each of them to generate a classifier for inclusion in the ensemble. The algorithms for bagging and doing the bootstrap sampling (sampling with replacement) are shown in Figure 2.

To create a bootstrap training set from a training set of size N, we perform N Multinomial trials, where in each trial, we draw one of the N examples. Each example has probability $1/N$ of being drawn in each trial. The second algorithm shown in Figure 2 does exactly this—N times, the algorithm chooses a number r at random from 1 to N and adds the rth training example to the bootstrap training set S. Clearly, some of the original training examples will not be selected for inclusion in the bootstrapped training set and others will be chosen one time or more. On average, each generated bootstrapped training set will contain $0.63N$ unique training examples even though it will contain N actual training examples. In bagging, we create M such bootstrap training sets and then generate classifiers using each of them. Bagging returns a function $h(x)$ that classifies new examples by returning the class y that gets the

AdaBoost$(\{(x_1, y_1), \ldots, (x_N, y_N)\}, L_b, M)$
Initialize $D_1(n) = 1/N$ for all $n \in \{1, 2, \ldots, N\}$.
For $m = 1, 2, \ldots, M$:
$h_m = L_b(\{(x_1, y_1), \ldots, (x_N, y_N)\}, D_m)$.
Calculate the error of $h_m : \epsilon_m = \sum_{n:h_m(x_n) \neq y_n} D_m(n)$.
If $\epsilon_m \geq 1/2$ then,
set $M = m - 1$ and abort this loop.
Update distribution D_m:

$$D_{m+1}(n) = D_m(n) \times \begin{cases} \frac{1}{2(1-\epsilon_m)} & \text{if } h_m(x_n) = y_n \\ \frac{1}{2\epsilon_m} & \text{otherwise} \end{cases}$$

Output the final model:
$h_{fin}(x) = \text{argmax}_{y \in Y} \sum_{m:h_m(x)=y} log \frac{1-\epsilon_m}{\epsilon_m}$.

Fig. 3. AdaBoost algorithm: $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ is the set of training examples, L_b is the base model learning algorithm, and M is the number of base models to be generated.

maximum number of votes from the base models $h_1, h_2, \ldots, h_M$. In bagging, the M bootstrap training sets that are created are likely to have some differences. If these differences are enough to induce noticeable differences among the M base models while leaving their performances reasonably good, then the ensemble will probably perform better than the base models individually.

In general, bagged ensembles tend to improve upon their base models more if the base model learning algorithms are *unstable*—differences in their training sets tend to induce significant differences in the models [13]. Another way of stating this is that bagging does more to reduce the variance in the base models than the bias, so bagging performs best relative to its base models when the base models have high variance and low bias. Decision trees are unstable, which explains why bagged decision trees often outperform individual decision trees; similarly, bagging, or in general bootstrapping works well for neural network classifiers [109]. However, Naïve Bayes classifiers or k-nearest neighbor (kNN) classifiers are stable, which explains why bagging is not particularly effective for Naïve Bayes or kNN classifiers [39].

3.5 Boosting

The AdaBoost algorithm, which is one of the most popular within the class of boosting algorithms, generates a sequence of base models with different weight distributions over the training set. The AdaBoost algorithm is shown

in Figure 3. Its inputs are a set of N training examples, a base model learning algorithm L_b, and the number M of base models that we wish to combine. AdaBoost was originally designed for two-class classification problems; therefore, for this explanation we will assume that there are two possible classes. However, AdaBoost is regularly used with a larger number of classes. [2]

The first step in AdaBoost is to construct an initial distribution of weights D_1 over the training set. This distribution assigns equal weight to all N training examples. We now enter the loop in the algorithm. To construct the first base model, we call L_b with distribution D_1 over the training set. [3] After getting back a model h_1, we calculate its error ϵ_1 on the training set itself, which is just the sum of the weights of the training examples that h_1 misclassifies. We require that $\epsilon_1 < 1/2$ (this is the *weak learning* assumption—the error should be less than what we would achieve through randomly guessing the class [4])—if this condition is not satisfied, then we stop and return the ensemble consisting of the previously-generated base models. If this condition is satisfied, then we calculate a new distribution D_2 over the training examples as follows. Examples that were correctly classified by h_1 have their weights multiplied by $\frac{1}{2(1-\epsilon_1)}$. Examples that were misclassified by h_1 have their weights multiplied by $\frac{1}{2\epsilon_1}$. Note that, because of our condition $\epsilon_1 < 1/2$, correctly classified examples have their weights reduced and misclassified examples have their weights increased. Specifically, examples that h_1 misclassified have their total weight increased to 1/2 under D_2 and examples that h_1 correctly classified have their total weight reduced to 1/2 under D_2. We then go into the next iteration of the loop to construct base model h_2 using the training set and the new distribution D_2. We construct M base models in this fashion. The ensemble returned by AdaBoost is a function that takes a new example as input and returns the class that gets the maximum weighted vote over the M base models, where each base model's weight is $log(\frac{1-\epsilon_m}{\epsilon_m})$, which is proportional to the base model's accuracy on the weighted training set presented to it.

Clearly, the heart of AdaBoost is the distribution updating step. The idea behind it is as follows. We can see from the algorithm that ϵ_m is the sum of the

[2] In fact, the AdaBoost algorithm we discuss here is called AdaBoost.M1. There are other versions of AdaBoost that are designed to work on multi-class classification problems. In this paper, we will continue to refer to AdaBoost.M1 as AdaBoost, since this is the only version that we discuss in detail.

[3] If L_b cannot take a weighted training set, then one can call it with a training set generated by sampling with replacement from the original training set according to the distribution D_m.

[4] This requirement is perhaps too strict when there are more than two classes. There is a multi-class version of AdaBoost [38] that does not have this requirement. However, the AdaBoost algorithm presented here is often used even when there are more than two classes if the base model learning algorithm is strong enough to satisfy the requirement.

weights of the misclassified examples. The misclassified examples' weights are multiplied by $\frac{1}{2\epsilon_m}$, so that the sum of their weights is increased to $\epsilon_m * \frac{1}{2\epsilon_m} = \frac{1}{2}$. The correctly classified examples start out having total weight $1-\epsilon_m$, but their weights are multiplied by $\frac{1}{2(1-\epsilon_m)}$, therefore, the sum of their weights decreases to $(1-\epsilon_m)*\frac{1}{2(1-\epsilon_m)} = \frac{1}{2}$. The point is that the next base model will be generated by a weak learner (i.e., the base model will have error less than 1/2); therefore, at least some of the examples misclassified by the previous base model will have to be learned.

Boosting does more to reduce bias than variance. For this reason, boosting tends to improve upon its base models most when they have high bias and low variance. Examples of such models are Naïve Bayes classifiers and decision stumps (decision trees of depth one). Boosting's bias reduction comes from the way it adjusts its distribution over the training set. The examples that a base model misclassifies have their weights increased, causing the base model learning algorithm to focus more on those examples. If the base model learning algorithm is biased against certain training examples, those examples get more weight, yielding the possibility of correcting this bias. However, this method of adjusting the training set distribution causes boosting to have difficulty when the training data are noisy [23]. Noisy examples are normally difficult to learn. Because of this, the weights assigned to noisy examples often become much higher than for the other examples, often causing boosting to focus too much on those noisy examples and overfit the data. Nevertheless, boosting is currently one of the best performing supervised classifier learning algorithms.

3.6 Order Statistics

Approaches to pooling classifiers can be separated into two main categories: simple combiners, e.g., averaging, and computationally expensive combiners, e.g., stacking. The simple combining methods are best suited for problems where the individual classifiers perform the same task, and have comparable success. However, such combiners are susceptible to outliers and to unevenly performing classifiers. In the second category, "meta-learners," i.e., either sets of combining rules, or full fledged classifiers acting on the outputs of the individual classifiers, are constructed. This type of combining is more general, but suffers from all the problems associated with the extra learning (e.g., overparameterizing, lengthy training time).

Both these categories of methods are in fact ill-suited for problems where *most* (but not all) classifiers perform within a well-specified range. In such cases the simplicity of averaging the classifier outputs is appealing, but the prospect of one poor classifier corrupting the combiner makes this a risky choice. Although weighted averaging of classifier outputs appears to provide some flexibility, ob-

taining the optimal weights can be computationally expensive. Furthermore, the weights are generally assigned on a per classifier, rather than per sample or per class basis. If a classifier is accurate only in certain areas of the inputs space, this scheme fails to take advantage of the variable accuracy of the classifier in question. Using a meta learner that would have weights for each classifier on each pattern would solve this problem, but at a considerable computational cost and danger of overfitting the training data. The robust combiners presented in this section aim at bridging the gap between simplicity and generality by allowing the flexible selection of classifiers without the associated cost of training meta classifiers.

Order statistics combiners that selectively pick a classifier on a per sample basis were introduced in [107,110]. In general the model error is given by:

$$E_{model}^{ensemble} = \alpha \, E_{model} \, . \tag{4}$$

where α is a factor that depends on the number of classifiers M and the order statistic chosen and the error model (combining M independent classifiers by linear averaging is equivalent to having $\alpha = \frac{1}{M}$). Values for α were tabulated in detail in [113], based on early work on order statistics [94]. Table 1 shows an example of α values for Gaussian model error. Note that the reduction factors are lower than they would be for independent classifier errors (e.g., $\alpha \geq \frac{1}{M}$). For example, selecting the median of three classifiers' outputs reduces the error by a factor of 0.449 rather than a theoretical best of 0.333 for averaging independent classifiers.

Table 1
Error Reduction Factors α, for the *min, max* and *med* Combiners (Gaussian Error Model).

M	OS Combiners	
	min/max	*med*
1	1.000	1.000
2	0.682	0.532
3	0.560	0.449
4	0.492	0.305
5	0.448	0.287
10	0.344	0.139
15	0.301	0.102
20	0.276	0.074

4 Ensemble Applications

In this section we take a closer look at four types of problems that have dominated the field of pattern recognition over the last decade. They are remote sensing, person recognition (face, fingerprint), one-vs-all classification (which covers many types of fault and intrusion detection), and medicine. Each of these fields has statistical difficulties (ranging from "too much data" to "too little of a particular type of data") that make it hard for standard pattern recognition algorithms to be applied and as such are fertile ground for the application of ensembles.

4.1 Remote Sensing

4.1.1 Review

As satellite/sensor technology has improved, the amount of remote sensing data collected has expanded both in sheer volume (e.g., terabytes) and detail (e.g., hundreds of spectral bands). As a consequence, this domain poses distinct challenges to classification algorithms [9,42,44,46,88,100,121]. In particular, classification algorithms need to account for:

- a large number of inputs as the patterns are collected repeatedly for large spaces [44,121];
- a large number of features as the data is collected across hundreds of bands [96];
- a large number of outputs as the classes cover many types of terrain (forest, agricultural area, water) and man-made objects (houses, streets) [65]
- missing or corrupted data as different bands or satellites may fail to collect data at certain times [100]; and
- poorly labeled (or unlabeled) data as the data needs to be postprocessed and assigned to classes [88].

Because of these difficulties hyper-spectral data classification is particularly well suited to the divide and conquer approach of classifier ensembles, where different classifiers with different properties can extract the necessary information from the raw data before an ensemble puts it all together. This allows for the inputs and/or features to be sampled to reduce the difficulty each classifier faces; the data labels to be estimated; and the multi class problems to be reduced to multiple two-class problems.

4.1.2 Example Applications

To show the breadth of applicability of classifier ensembles to remote sensing, we will highlight four types of ensembles on four different types of remote sensing data. The ensembles will range from:

- Bagging/boosting and random forests of classification trees [44];
- Majority voting and belief combination for neural networks [42];
- Hierarchical classification of binary Bayesian classifiers [30]; and
- Information fusion for fuzzy set interpretation of classifier outputs [65].

The domains include:

- Mountainous terrain from Colorado [44];
- Agricultural land from the United Kingdom (UK) [42];
- Urban terrain from Iceland [30]; and
- Wetlands from Florida [65].

Random Forests and Mountainous Terrain: Using random forests to provide land classification is a good example of the applicability of classifier ensembles to remote sensing [44]. In this work, Gislason, Benediktsson and Sveinsson use Bagging, Boosting, and Random Forests (ensembles of classification trees, hence the name) composed of individual CART-like [14] trees. Random forests are ideally suited for this type multisource classification particularly when the importance of particular variables (e.g., channels) is not known a priori. Random forests are generated just like bagged decision trees except that, at every decision tree node, the best feature is chosen from a random subset of unused features rather than all the unused features. For the final classification stage each tree gets one vote to determine the winning class to which a given output belongs. One key feature of this algorithm is that it limits the number of inputs it uses to build the tree and hence is readily applicable to large data sets.

The results reported in this paper are based on Landsat, elevation, slope and Aspect data [9,44] collected from mountainous areas in Colorado. They focus on a ten-class problem where, in addition to water and meadows, eight types of trees (or tree combinations) are classified. The results show that six of the seven ensemble methods used (three types of bagging, three types of boosting and one type of random forest) provide improvement over simply using CART, showing the great appeal of using ensemble methods for classifying large multisource data sets.

Majority voting for agricultural land: While the work above focused on classification trees and bagging/boosting, neural networks are also ideally suited for classifying remote sensing images. Giacinto and Roli use ensembles of neural networks to classify images from an agricultural area from the town

of Feltwell, UK [42]. This is a challenging problem of classifying five classes (sugar cane, stubble, bare soil, potatoes and carrots), using both a multi-band optical sensor (eleven bands) and a multi-channel radar sensor (twelve channels).

The authors used three types of neural networks, a multi-layered perceptron (MLP), a radial basis function (RBF) network and a probabilistic neural network (PNN), as well as two statistical classifiers, a Gaussian classifier and a k-nearest neighbor classifier (KNN), as their base classifiers. In using a variety of parameters for each, in total they trained 182 base classifiers. Their ensemble results showed that even using as few as three classifiers provided significant improvements over the base classifiers for this difficult remote sensing problem.

Hierarchical classification of wetlands: The third example for this section focuses on using a hierarchical classification scheme to classify wetlands around the Kennedy Space Center in Florida [65,88]. In this work Kumar et. al use an automatically generated hierarchy of binary classifiers for a C-class problem. They generate 'meta classes' to partition the data and then recursively continue to partition the data until each meta class is reduced to one of the original classes. Because the resulting classification problems are simpler and two-dimensional, they select a Bayesian classifier as their base classifier. They also report results on a variety of single and ensemble classifiers for this domain.

The data used in this work consists of a 12-class landcover data that has 180 bands. Their results further show that using a hierarchical classification scheme provides improvements over both base classifiers and an ensemble of MLPs.

Information fusion for Urban areas: The final application uses the interesting concept of interpreting the outputs of base classifiers as fuzzy sets and using that information to perform information fusion [30]. Fauvel et. al apply this idea to urban satellite images from Iceland. They use both a neural network and a fuzzy classifier to obtain probability of membership to a particular class. They then interpret these classifications as memberships in fuzzy sets (classes) and fuse the fuzzy sets to obtain the final classification results.

The data is based high resolution pictures of Rejkjavik, Iceland, and consists of six classes (large buildings, houses, large roads, streets, open areas and shadows). Each image is single channel with one meter resolution. The information fusion based results show significant improvement over both base classifiers. What is particularly interesting in this work is that the two base classifiers have very different strengths and perform unevenly across the different classes. The fuzzy classifier for example performs well on shadows (83% correct) and

poorly on streets (9.8% correct), whereas the neural network performs well on the streets (55% correct) but poorly on houses (33%). The fusion of the two though generally performs as well as the best method, providing accurate final classification.

4.2 Person Recognition

4.2.1 Review

Person recognition is the problem of verifying the identity of a person using characteristics of that person, typically for security applications. Examples include iris recognition, fingerprint recognition, face recognition, and behavior recognition (such as speech and handwriting)—recognizing characteristics of a person, as opposed to depending upon specific knowledge that the person may have (such as usernames and passwords for computer account access). Person recognition has historically been one of the most frequent applications of ensemble learning methods.

There are many facets to the person recognition problem, some of which make it especially well-suited to be addressed by ensemble methods. Person recognition can involve multiple types of features. In case of iris, fingerprint, and face recognition, the features are those that are often used in computer vision. Often, such as in face recognition, Principal Components Analysis (PCA) is used to identify a set of features that is smaller than the entire image but is nevertheless more informative. In speech recognition, characteristics of the speech signal such as frequencies and amplitudes must be captured and appropriate new features extracted or selected. Speech and other behavioral characteristics, unlike more static features such as fingerprints and faces, also can be represented as time series. Combining such diverse feature types into one recognizer is difficult because they have different scales. Ensembles consisting of individual recognizers for each modality would work better because they combine at the decision level where the scales would be the same.

Another facet to the person recognition problem is the difficulty in collecting good data. The environment and data collection instruments introduce noise, which is common to many applications. However, some environments are particularly difficult. For example, Erdogan et. al perform person recognition in a vehicle [28]. If person recognition is to be performed during vehicle operation, then obviously the data collection method must not interfere with vehicle operation, must be non-intrusive, and must be robust to the noise typical of roads and highways.

In person recognition, just as in medical applications and certain anomaly detection applications, there are different misclassification costs. For example,

in person recognition, denying system access to a legitimate user who should be given such access may be less costly than allowing access to an illegitimate user. In some systems, it may be possible for the recognition system to defer to a human in cases where the recognition system cannot firmly decide whether the person seeking access is a legitimate or illegitimate user.

4.2.2 Example Applications

Unobtrusive Person Identification: One example of a person recognition application is covered by Suutala and Roning [103]. Here, the goal is to keep track of the locations and identities of people within an environment. Continuous monitoring of this sort requires that the recognition system identify people in as unobtrusive a manner as possible—these are known as calm applications [115]. That is, the system should not require people to take any specific authentication actions the way fingerprint or iris scanning systems do. Also, the system should not require people to wear sensors. The sensors should be hidden in the environment so as to not make users conscious that they are being constantly monitored—this makes computer vision-based gait recognition systems [73,53,57] less than ideal for this task. The system developed in [103] recognizes people by characterizing their walking patterns using a pressure-sensitive floor.

The authors tested a variety of ensembles for this task. They used three different sets of features as inputs: spatial and statistical features of individual footsteps calculated from the pressure signal, frequency domain of the pressure signals, and the frequency domain of the derivative of the pressure signal. They tested individual classifiers with either one feature set or all the feature sets together. For classifiers, they used K-nearest neighbors, Learning Vector Quantization (LVQ), Multi-Layer Perceptrons (MLPs), Radial Basis Function (RBF) networks, and Support Vector Machines (SVMs). For combining methods they used maximum, minimum, median, sum, and product combiners. They also compared classification based on single footsteps versus classification based on footstep sequences. In addition to these, they tested what happens when classifiers are allowed to reject options that they are not sure about, with the idea that rejected options can be evaluated by humans. In particular, they allowed rejection of examples that were not clearly a part of any class (the maximum posterior probability over the classes is too low) as well as examples that are too close to the boundary between two classes to allow one to clearly determine which class they are in (the difference between the highest and second-highest posterior probabilities is too low). In their work, SVMs and MLPs generally outperformed the other single classifiers. Having separate classifiers for each feature type worked best, ensembles outperformed individual classifiers, and the reject method greatly increased classifier reliability.

Face Recognition: Chawla and Bowyer [19] addressed the standard problem of face recognition [1,35,20,93] but under different lighting conditions and with different facial expressions. They tested recognition of subjects that they trained with but also subjects that they did not train with. Their data consisted of images of multiple subjects, each with multiple pictures taken under two different lighting conditions and with two different facial expressions (neutral and smiling). They converted the image matrices into vectors and performed PCA on these vectors, and they retained $m - 1$ components, where m is the number of training images. They used 1-Nearest Neighbor (1-NN) and Linear Discriminant Analysis (LDA) as their classifiers. They created both specialized classifiers (classifiers trained on only training images of one lighting condition and one facial expression) and classifiers trained across multiple conditions. They also experimented with ensembles of 1-NN classifiers. They refer to their ensembles as Ensemble-1, Ensemble-2, and Ensemble-3, where 1, 2, and 3 refer to the number of images per subject (out of 4) in the training set that they randomly select for inclusion in the training set of each 1-NN classifier. They found that 1-NN always outperformed LDA—the authors feel that this is because there were not enough training images for LDA to generalize. They found that their ensembles always outperformed LDA and the specialized classifiers.

Multi-modal Person Recognition: Erdogan, et. al. [28] prepared a multimodal person recognition system. Their application involved recognition of the driver of a vehicle. They discuss several benefits to such an application:

- Ensuring that only authorized drivers drive the vehicle.
- Personalizing the vehicle for the driver's physical and behavioral characteristics.
- Warning the driver and appropriate authorities if the driver is not in the proper condition to drive.
- Allowing secure transactions, such as banking, from within the car.

In a vehicle, face recognition, speech recognition, and driver behavior recognition (e.g., steering and pedal inputs) are possible. However, all have difficulties. All the usual problems in face recognition exist, including changes in illumination, pose, and facial expression. The usual problems in speech recognition are also present, including noise and variations in speech due to illness and emotions. However, these problems are bigger in a vehicle since the recognition needs to be performed in an unobtrusive manner. For example, the microphone cannot be near the driver's mouth. The authors of [28] hoped to use all three recognition modalities to yield better performance than any one modality would by itself.

For speech recognition, they used Mel-Frequency Cepstral Coefficients (MFCC) [75] as inputs, and Gaussian Mixture Models (GMMs) with eight mixture compo-

nents as the classifiers. For Face Recognition, they use PCA to extract features from face images and these serve as inputs to a single Gaussian model. For driver recognition, they use brake and acceleration pedal positions and derivatives as inputs and GMMs with eight mixture components as the classifiers. They compared the performance using just one modality with the performances of ensembles combining classifiers of any two modalities and with the performance using the ensemble containing all three modalities' classifiers. They performed three types of recognition tasks. The first type is verification, where the system confirms that the identified person is who he or she claims to be. The second type is closed set identification, where the system can only classify the inputs as representing one of the known people. The third type is open set identification, which is similar to closed set identification except that the system also has the option of rejecting the input if it is not similar enough to any past example that it has seen. They tested their work on a subset of the in-car corpus of data collected by the Center for Integrated Acoustic Information Research (CIAIR) [60]. Across all three sets of tests, the ensembles of the three classifiers outperformed the ensembles with two modalities' classifiers. The ensembles with two modalities almost always outperformed the individual modalities' classifiers. The only exception was that, for open-set identification, the audio-only classifier outperformed the ensemble that used face recognition and driver signals.

User-specific Speech Recognition: So far, the solutions that we have examined have devised one model (whether an ensemble or not) to classify everyone. Fierrez-Aguilar, et. al., [34] have used both a global model of this type but also user-specific models for speech recognition and combined them with the goal of improving recognition rates [56,33,31,106,32,64,101,84]. [5] In particular, they developed user-independent and user-dependent versions of several baseline systems. These baseline systems used either MFCC, phone sequences, energy contours of speech signals, or word n-grams as inputs and GMMs, SVMs, or n-gram recognizers as the classifiers. These were then combined in various ways. They found that a combination of user-independent and user-dependent recognizers performed better than just a combination of user-independent recognizers. Each user enrolled in the system had a combination of a user-independent recognizer and that user's user-dependent recognizer created. It was formed by calculating a convex combination of sufficient statistics of a user-independent model and that user's model. All the models were multivariate Gaussians, so the calculations were relatively simple.

As is the case with ensemble learning in general, ensemble learning applied to person recognition has largely focused on the inputs and the base classifiers to

[5] Obviously, this is best suited for the verification type of recognition problem, as discussed in the previous paragraph.

be combined rather than the combining scheme itself[6] or the metric to be optimized. The authors of [85] decided to theoretically examine which combiners do the best job of optimizing the Equal Error Rate (EER), which is the error rate of a classifier when the false acceptance rate is equal to the false rejection rate. In particular, they calculated the EER of mean, weighted sum, and order statistics combiners (maximum, minimum, and median) under the assumption that class-independent scores (scores for clients and impostors) are normally distributed. They conducted multiple experiments with synthetic and real datasets in which they combined classifier scores in various ways. They found that the weighted sum yielded the best improvement in EER over individual classifiers.

4.3 One vs. All Recognition

4.3.1 Review

One versus all recognition encompasses several different problems. One of them is anomaly detection, which is the problem of detecting unusual patterns, i.e., what does not fit into the set of identified patterns. The opposite problem is target recognition—finding what fits into an identified pattern. Intrusion detection is a problem that could be solved both ways—look for one of a set of known types of attacks (target recognition) or look for anomalies in the usage patterns (anomaly detection).

Ensemble methods are well-suited to solving one vs. all recognition problems. In target detection problems, one can easily envision having one model per possible target and running all the models in parallel and observing if one model fires or gives a stronger indication of recognition than the others. If two or more models fire at nearly equal strength, then other steps may need to be taken to disambiguate among the possible target types. In anomaly detection, an ensemble may consist of models designed to detect anomalies under different situations.

As mentioned earlier, the intrusion detection problem can be cast as either a target detection or anomaly detection problem. When intrusion detection problems are cast as target detection problems, they are sometimes referred to as misuse detection. In these problems, models of known attacks are devised, and if current computer system activity matches that described by any attack model, then it is assumed that the corresponding type of attack is taking place. The disadvantage of this method is that it requires knowledge of and data representing all the types of attacks that one expects. If one does not have this, then one can expect a large false negative rate because the system will miss

[6] Examples of work in this area include [66,112,40].

attacks of the types that it does not know about. Anomaly detection methods work by modeling normal system behavior and flagging significant deviations from normal behavior as examples of attacks. This avoids the problem of having to model every type of attack; however, anomaly detection systems are prone to high false alarm rates, since any legitimate activity that is not modeled may be needlessly flagged as an attack.

We will now survey three examples of ensemble methods applied to network intrusion detection.

4.3.2 Example Applications

Anomaly detection applied to network intrusion detection, as mentioned above, has the problem that any legitimate computer system activity that is not modeled may get needlessly flagged as an attack. Part of the problem is that many of the anomaly detection systems devised in the literature are monolithic models that are designed to cover all protocols and services offered by the computer system and network [86,69,29].

Modular Intrusion Detection: In Giacinto et. al, the authors design one classifier, which they refer to as a module, for each group of protocols or services [43]. Each group's behavior is then easier to model than all the groups' behaviors simultaneously, and this arrangement also allows each group's behavior module to be tuned separately.

The authors use three different types of unsupervised methods: Parzen Density Estimators, K-means clustering, and the ν-Support Vector Classifier (ν-SVC) which is similar to the one-class Support Vector Machine (SVM) [105]. They use three sets of features as inputs to these methods: intrinsic features, which are extracted from the headers of packets related to individual network connections; content features, which are extracted from the data portion of the packets; and traffic features, which are related to statistical measures on multiple connections related to a particular type of service. They developed classifiers for four types of services: HTTP, FTP, Mail, and ICMP.

Overall, they found that, when the allowed false alarm rate is set to be low (e.g., 1%), then the detection rates for their modular approach with one model per service type outperformed a monolithic model that was trained to detect attacks in all types of services. At higher false alarm rates (e.g., 4%), the monolithic models tended to have higher detection rates.

Hierarchical Intrusion Detection: Another approach to intrusion detection is demonstrated in [21]. They developed a multi-stage classification system in which each input pattern is tested by a hierarchical sequence of classifiers to determine whether the pattern is an example of normal traffic or one of

four different types of attacks (Probe, Denial of Service, Remote to Local, and User to Root). In particular, their first stage determines whether an input pattern is part of normal traffic or an attack that is either a Denial of Service or Probe attack. If the first stage determines that the input is one of these attacks, then the input is passed to a second stage classifier that determines which of the two attack types it is. If the first stage does not determine that the input is of a Denial of Service or Probe attack, then the input pattern is passed to a different second stage classifier that determines if the attack is of type User to Root. If it is not, then a third stage classifier determines if the attack is of type Remote to Local. If the third stage determines that the input does not represent a Remote to Local attack, then the input is determined to be normal. All the stages also have tuned reliability thresholds, with the idea that, if the classification decision has a reliability lower than the threshold, then the input is rejected and logged for human examination.

They tested this architecture on a subset of the database created by DARPA as part of the 1998 Intrusion Detection Evaluation Program. They tested variants on their architecture with their stage 1-3 classifiers swapped in different ways. They also compared their architecture to Learning Vector Quantization (LVQ), a three-layered Multi-Layer Perceptron (MLP), and ensembles of these. They obtained significant improvements with their original architecture over the other methods described. They presented results on data from FTP and HTTP, but state that they tested using data from other network services and obtained significant improvements with their architecture on these other services.

Intrusion Detection in Mobile Ad-hoc Networks: The authors of [16,15] applied ensemble methods to intrusion detection in Mobile Ad-hoc NETworks (MANETs) [54,123,124]. Examples of MANETs include wireless networks. MANETs are more difficult to work with than regular computer networks because the wireless nature of the networks leads to the structure of the network changing regularly, as computers, Personal Digital Assistants (PDAs) and other devices join the network and leave the network more frequently than they would with traditional wired networks. Because of this, they devised a hierarchical architecture. The bottom layer consists of nodes, which represent individual devices connected to the network. The next layer consists of clusters, which vary as nodes move around within the network. The top level is the manager. Anomaly indices within the nodes are averaged to form the anomaly indices at the cluster level, which in turn are averaged to form a manager-level anomaly index. The authors found that the anomaly indices at higher levels tended to perform better than those at the lower levels, demonstrating that the combination of distributed anomaly detectors yields better results than individual detectors.

4.4 Medical Applications

4.4.1 Review

There are many examples of medical applications of machine learning. These examples include many different types of problems, such as analyzing x-ray images, human genome analysis, and examining sets of medical test data to look for anomalies. However, the root of all these problems is in assessing the health of human beings. This root brings about several characteristics of medical applications that make them particularly difficult problems. In general, such problems have:

- Limited training and test examples, i.e., few training examples due to the nature of problem and privacy concerns;
- Imbalanced datasets, ie., very few anomalies or examples of patients with a disease;
- Too many attributes, i.e., often many more than the number of training and test examples; and
- different misclassification costs, i.e., false negatives significantly worse than false positives.

Because the number of training and test examples corresponds to the number of patients, the number of such examples is typically lower than for most other application areas. The examples also tend to be relatively imbalanced—the number of examples of anomalies or patients with diseases is, fortunately, much lower than the number of examples of normal patients. However, this can pose difficulties for machine learning algorithms—especially classification algorithms. With so few positive (e.g., disease present) examples, classification algorithms will tend to be strongly biased toward predicting that new examples are negative. However, incorrectly predicting that an example is negative (false negative) is typically much worse than incorrectly predicting that an example is positive. The number of attributes also tend to be much higher than the number of examples in problems such as human genome analysis and image analysis. This mandates the use of feature selection and/or feature extraction methods, and the selection of the appropriate such method(s) add significantly to the challenge of using machine learning for these problems.

We will now discuss a few examples of ensemble methods applied to several different problems in medicine.

4.4.2 Example Applications

Pharmaceutical Molecule Classification: One particular medical application is predicting the biological activity of a pharmaceutical molecule given a

quantitative description of the molecule [104]. Examples of biological activity to be predicted are potency against disease targets, toxicity, as well as absorption, distribution, metabolism, and excretion (ADME). The predicted values could either be continuous values or categorical values. The pharmaceutical molecule structure is described by a set of topological descriptors. This problem is an example of Quantitative Structure-Activity Relationship (QSAR) modeling [27,49].

The number of samples is exceeded by the number of features, many of which are irrelevant for the prediction task; therefore, feature selection is necessary. The authors chose decision trees as the base models for this task over artificial neural networks, Support Vector Machines, multiple linear regression, and other methods because decision tree learning algorithms include feature selection in their operation. They decided to use Random Forests as their ensemble algorithm with decision trees as the base models. They compared random forests with two other previously-used popular QSAR methods:—standard decision trees and partial least squares—as well as SVMs using a linear kernel and SVMs using a radial basis function (RBF) kernel. They used six publicly available ADME datasets for their assessment that included both classification and regression problems. The authors found that random forests without tuning any parameters (such as the number of tests to try at each decision tree node) yielded performance at least as good as the other methods. They attempted to tune the number of variables to test at each decision tree node and to eliminate irrelevant variables before training, but obtained very little improvement from that, demonstrating that random forests perform quite well "off-the-shelf" for their problem.

MRI Classification: Another problem where feature selection is very important is the classification of Magnetic Resonance (MR) spectra [87]. The goal here is to find subregions of the MR spectra that are informative for classification. They used the Random Subspace Method [51]. This method constructs each base model in an ensemble using a random subset of the original features, thereby often obtaining diversity in the ensemble. However, rather than only using the original features, they also attempted to run RSM on the feature set constructed by PCA. They also experimented with a method that they devised [78] that uses genetic algorithms to guide a search for the best feature set to use. They used boxcar functions as the basis set and extracted features that represent averages of adjacent spectral regions. They referred to these discovered discriminatory patterns as the Domain Feature (DF) set. RSM then constructed an ensemble where a random subset of the domain feature set was chosen for each base classifier. In all their experiments, they used 31 base classifiers and majority voting to combine their results. For base models, they used either 31 1-Nearest Neighbor classifiers or 31 L_2-norm linear SVMs. They varied the number of features selected for each base classifier from 1 to 10, although they used the same number of features for all base classifiers in

a given ensemble.

They experimented on three MR datasets of increasing difficulty (decreased class separability). They found that the domain feature set was always at least as good as the other methods (PCA and original features), and has the additional advantage of maintaining interpretability, being that the features are merely averages of boxcar functions that can be easily visualized. PCA was only competitive for the first (easy) dataset. Although given that PCA creates features based on the inputs only and independently of the class labels, the poor performance of PCA is not surprising.

ECG Classification: Since medical applications often involve detecting issues with individual patients, patient-specific classifiers can be useful just as user-specific classifiers are with the person recognition systems discussed earlier (e.g., [34]). In [70] the authors used an approach similar to [34], in that they used both a generic classifier meant to distinguish normal and abnormal data across people and a user-specific classifier meant to distinguish between normal and abnormal data for a specific person [70].

Lincoln and Skrzypek address the problem of classifying electrocardiographic (ECG) signals [70]. An ECG signal represents cardiac activities, which cardiologists use to evaluate the performance of heart function. Automated analysis of ECG records is necessary because the recording may last for hours, which makes analysis by doctors excessively time-intensive. Many algorithms such as self-organizing maps (SOM), learning vector quantization (LVQ), multilayer perceptrons (MLPs), neural-fuzzy systems, and Support Vector Machines (SVMs) have been applied to ECG signals. However, these methods have been typically applied to distinguish normal signals from abnormal signals across patients. This is difficult because of the substantial variation in the morphologies of ECG signals across patients. For this reason, [70] implemented an ensemble consisting of a standard SVM designed to distinguish normal signals from abnormal signals across patients and a set of one-class SVMs [95] (one per patient) to distinguish normal signals for a given patient from all other signals [70]. The one-class SVMs were trained using only normal signals from a given patient. Being an anomaly detection algorithm, the one-class SVM builds a model that classifies the trained data as normal (although it can be trained to allow a user-determined fraction of the training points as abnormal) and then classifies the data outside the range of normal data as abnormal. They used decision templates to combine the two SVMs. They calculated one decision template DT_j for each class y_j ($j \in \{-1, +1\}$). DT_j is a two-element vector calculated by selecting the examples in the validation set that are in class y_j and averaging the outputs of the two classifiers on those examples. New examples are classified by having each of the two base classifiers classify the examples, putting those together into a vector, and finding the vector DT_j that it is closest to in Euclidian distance. On average, over 22 test sets, the

ensemble got a balanced classification rate (geometric mean of sensitivity and specificity) improvement of 13.8% over the global two-class SVM and 3.1% over the individual one-class SVMs.

5 Discussion and Future Directions

In this paper, we have summarized the most frequently used classifier ensemble methods, including averaging, bagging, boosting and order statistics combiners. Each ensemble method has different properties that make it better suited to particular types of classifiers and applications. We also presented four broad application domains that have received particular attention from the classifier ensemble community: remote sensing, person recognition, one vs. all recognition, and medical/biological applications. In each of these applications, harnessing a set of reasonable-performing but diverse set of base models yields a combined classifier that performs better than any individual classifier. The theory, algorithms, and applications of ensemble learning continue to be active areas of research. Based on the work surveyed in this article and the current state of ensemble learning, we anticipate three major research areas that will receive more attention in the future.

First, the sheer amount of available data is exploding as both data collection and processing algorithms handle amounts of data that would have been inconceivable a decade ago. Yet, most of the classification and ensemble algorithms do not address this issue directly. Though several of the applications have dealt with this problem indirectly (e.g. remote sensing, intrusion detection), new ensemble methods that use this richness of the data rather than aim to avoid it are needed. Indeed, were there methods available that could handle hundreds of classifiers trained on diverse data sets at different times and at different physical locations, many new application domains would open up to ensemble methods.

Second, there are many domains that are well-suited to unsupervised learning algorithms, such as clustering algorithms, that have not benefitted from the bulk of the classifier ensemble results. The central concept of clustering is that data points are partitioned into separate clusters so that data points within a cluster are "close" to each other, while data points from different clusters are "far" from each other. This concept is of great practical interest in terms of summarizing data, reaching quick decisions and flagging certain patterns for further study. In general however, there can be many different clusterings for the same data, depending on the clustering algorithm used [22,58,62]. The different clusterings analyze different aspects of the data and result in different biases in the clustering algorithms, but without a "target" it is difficult to select a specific clustering. Cluster combining, where multiple clusterings are

used to reach a consensus clustering would be invaluable in improving the performance of clustering algorithms. [2,58,59,102].

Third, all the ensemble algorithms discussed in this article are based on "passive" combining, in that the classification decisions are made based on static classifier outputs, assuming all the data is available at a centralized location at the same time. Distributed classifier ensembles using "active" or agent-based methods can overcome this difficulty by allowing agents to decide on a final ensemble classification based on multiple factors (e.g., classification, confidence) [2]. This approach can be viewed as "active" stacking in that a second layer of decision making is introduced. In addition to providing more flexibility and incorporating spatially and temporally separated data, such an approach will allow the infusion of prior information (as a "base" classifier with set decisions) and be applicable to both supervised and unsupervised learning.

All three extensions of classifier ensembles highlighted above address the different aspects of the problem arising from the complexity and richness of the data generated by new applications. In many ways, such applications will not only provide new domains for ensembles, but also provide the impetus for developing new ensemble methods. Finally, though we discussed these three extensions separately, in many domains, all three may be present simultaneously. For example, an agent-based, active ensemble method applied to a distributed clustering problem where terabytes of data are collected from different sources is precisely at the intersection of all these avenues. Research into such classifier ensembles for such domains offers the promise of new solutions to a multitude of problems including remote sensing, person recognition, intrusion detection and biological applications.

References

[1] Face recognition grand challenge and the biometrics experimentation environment. (URL: http://bee-biometrics.org).

[2] A. Agogino and K. Tumer. Efficient agent-based cluster ensembles. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan, May 2006.

[3] K. Al-Ghoneim and B. V. K. Vijaya Kumar. Learning ranks with neural networks (Invited paper). In *Applications and Science of Artificial Neural Networks, Proceedings of the SPIE*, volume 2492, pages 446–464, April 1995.

[4] R. Battiti and A. M. Colla. Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7(4):691–709, 1994.

[5] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, Sep. 1999.

[6] W. G. Baxt. Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation*, 4:772–780, 1992.

[7] S. Beck and J. Ghosh. Noise sensitivity of static neural classifiers. In *SPIE Conf. on Applications of Artificial Neural Networks SPIE Proc. Vol. 1709*, pages 770–779, Orlando, Fl. , April 1992.

[8] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain. Parallel consensual neural networks with optimally weighted outputs. In *Proceedings of the World Congress on Neural Networks*, pages III:129–137. INNS Press, 1994.

[9] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy. Neural network approaches versus statistical methods in classification of multisource remote sensing. *IEEE Transactions of Geoscience and Remote Sensing*, 28:540–552, 1990.

[10] V. Biou, J. F. Gibrat, J. M. Levin, B. Robson, and J. Garnier. Secondary structure prediction: combination of three different methods. *Protein Engineering*, 2:185–91, 1988.

[11] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[12] L. Breiman. Stacked regression. Technical Report 367, Department of Statistics, University of California, Berkeley, 1993.

[13] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[14] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Pacific Grove, California, 1984.

[15] J. B. D. Cabrera, C. Gutierrez, and R. K. Mehra. Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks. *(to appear) Information Fusion, Special Issue on Applications of Ensemble Methods*, 2008.

[16] J. B. D. Cabrera, C. Gutierrez, and R. K. Mehra. Infrastructures and algorithms for distributed anomaly-based intrusion detection in mobile ad-hoc networks. In *Proceedings of the IEEE Conference on Military Communications*, pages 1831–1837. IEEE, Atlantic City, New Jersey, USA, 2005.

[17] P. Chan and S. Stolfo. On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Integration of Information*, 1998.

[18] P. K. Chan and S. J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the Twelfth International Machine Learning Conference*, pages 90–98, Morgan Kaufmann, Tahoe City, CA, 1995.

[19] N. Chawla and K. Bowyer. Designing multiple classifier systems for face recognition. In N. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 407–416. Springer, Berlin, 2005.

[20] R. Chellappa, C. Wilson, and S. Sirohey. Human and machine recognition of faces: A survey. In *Proceedings of the IEEE*, volume 83(5), page 705=740. Institute for Electrical and Electronics Engineers, 1995.

[21] L. Cordella, A. Limongiello, and C. Sansone. Network intrusion detection by a multi-stage classification system. In J. Kittler, F. Roli, and T. Windeatt, editors, *Proceedings of the Fifth International Workshop on Multiple Classifier Systems*, pages 324–333. Springer, Berlin, 2004.

[22] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[23] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–158, Aug. 2000.

[24] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[25] H. Drucker, R. Schapire, and P. Simard. Improving performance in neural networks using a boosting algorithm. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems-5*, pages 42–49. Morgan Kaufmann, 1993.

[26] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, NY, second edition, 2001.

[27] S. Ekins. Progress in predicting human adme parameters in silico. *Journal of Pharmacology Toxicology Methods*, 44:251–272, 2000.

[28] H. Erdoğan, A. Eril, H. K. Ekenel, S. Y. Bilgin, I. Eden, M. Kirisi, and H. Abut. Multi-modal person recognition for vehicular applications. In N. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 366–375. Springer, Berlin, 2005.

[29] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*. Kluwer, 2002.

[30] M. Fauvel, J Chanuscot, and J. A. Benediktsson. Decision fusion for the classification of urban remote sensing images. *IEEE Transactions of Geoscience and Remote Sensing*, 44:2828–2838, 2006.

[31] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Exploiting general knowledge in user-dependent fusion strategies

for multimodal biometric verification. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, volume 5, pages 617–620. Institute for Electrical and Electronics Engineers, 2004.

[32] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Bayesian adaptation for user-dependent multimodal biometric authentication. *Pattern Recognition*, 38(8):1317–1319, 2005.

[33] J. Fierrez-Aguilar, J. Ortega-Garcia, D. Garcia-Romero, and J. Gonzalez-Rodriguez. A comparative evaluation of fusion strategies for multimodal biometric verification. In J. Kittler and M. S. Nixon, editors, *Fourth International Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA)*, pages 830–837. Springer, Berlin, 2003.

[34] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Speaker verification using adapted user-dependent multilevel fusion. In N. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 356–365. Springer, Berlin, 2005.

[35] P. J. Flynn, K. W. Bowyer, and P. J. Phillips. Assessment of time dependency in face recognition. In *International Conference on Audio and Video Based Biometric Person Authentication*, pages 44–51. 2003.

[36] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37. Springer Verlag, March 1995.

[37] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Morgan Kaufmann, Bari, Italy, 1996.

[38] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[39] J. H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. 1997. (from: http://www-stat. stanford. edu/ jhf/#reports).

[40] G. Fumera and F. Roli. Analysis of linear and order statistics combiners for fusion of imbalanced classifiers. In F. Roli and J. Kittler, editors, *Proceedings of the Third International Workshop on Multiple Classifier Systems*, pages 252–261. LNCS Vol. 2364, Springer, Berlin, 2002.

[41] J. Ghosh. Multiclassifier systems: Back to the future. In F. Roli and J. Kittler, editors, *Proceedings of the Third International Workshop on Multiple Classifier Systems*, pages invited paper, 1–15. LNCS Vol. 2364, Springer, Berlin, 2002.

[42] G. Giacinto and F. Roli. Ensembles of neural networks for soft classification of remote sensing images. In *Proceedings of the European Symposium on Intelligent Techniques*, pages 166–170, Bari, Italy, March 1997.

[43] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *(to appear) Information Fusion, Special Issue on Applications of Ensemble Methods*, 2008.

[44] P. A. Gislason, J. A. Benediktsson, and J. R. Sveinsson. Decision fusion for the classification of urban remote sensing images. *Pattern Recognition Letters*, 27:294–300, 2006.

[45] C. W. J. Granger. Combining forecasts—twenty years later. *Journal of Forecasting*, 8(3):167–173, 1989.

[46] J. B. Hampshire and A. H. Waibel. The Meta-Pi network: Building distributed representations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):751–769, 1992.

[47] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1000, 1990.

[48] S. Hashem and B. Schmeiser. Approximating a function and its derivatives using MSE-optimal linear combinations of trained feedforward neural networks. In *Proceedings of the Joint Conference on Neural Networks*, volume 87, pages I:617–620, New Jersey, 1993.

[49] D. M. Hawkins, S. C. Basak, and X. Shi. Qsar with few compounds and many features. *Journal of Chemical Information and Computer Sciences*, 41:663–670, 2001.

[50] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[51] T. K. Ho. The random space method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[52] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–76, 1994.

[53] P. Huang, C. Harris, and M. Nixon. Comparing different template features for recognizing people by their gait. In J. Carter and M. Nixon, editors, *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, Southampton, UK, 1998.

[54] Y. -A. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *ACM Workshop on Security of Ad Hoc and Sensor Network (SASN03)*. October 2003.

[55] R. Jacobs. Method for combining experts' probability assessments. *Neural Computation*, 7(5):867–888, 1995.

[56] A. K. Jain and A. Ross. Learning user-specific parameters in a multibiometric system. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 57–60. Institute for Electrical and Electronics Engineers, 2002.

[57] A. Kale, A. Rajagopalan, N. Cuntoor, and V. Kruger. Gait-based recognition of humans using continuous hmms. In *Proceedings of the International Conference on Face and Gesture Recognition*, pages 321–326. Washington, D. C. , USA, 2002.

[58] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[59] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. In *Proceedings of the Design and Automation Conference*, 1997.

[60] N. Kawaguchi, S. Matsubara, I. Kishida, Y. Irie, H. Murao, Y. Yamaguchi, K. Takeda, and F. Itakura. Construction and analysis of the multi-layered in-car spoken dialogue corpus. In *DSP in Vehicular and Mobile Systems*, chapter 1. Springer, New York, 2005.

[61] J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. In *Intelligent Systems for Molecular Biology*, Heidelberg, Germany, 1999.

[62] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[63] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 231–238. M. I. T. Press, 1995.

[64] A. Kumar and D. Zhang. Integrating palmprint with face for user authentication. In *Proceedings of the Workshop on Multimodal User Authentication (MMUA 2003)*, pages 107–112. 2003.

[65] S. Kumar, J. Ghosh, and M. Crawford. Hierachical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, 5:210–220, 2002.

[66] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, February 2002.

[67] M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. Technical report, University of Toronto, 1993.

[68] J. Lee, J. -N. Hwang, D. T. Davis, and A. C. Nelson. Integration of neural networks and decision tree classifiers for automated cytology screening. In *Proceedings of the International Joint Conference on Neural Networks, Seattle*, pages I:257–262, July 1991.

[69] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the 28th Australasian Computer Science Conference (ACSC2005)*. 2005.

[70] P. Li, K. Chan, S. Fu, and S. M. Krishnan. An abnormal ecg beat detection approach for long-term monitoring of heart patients based on hybrid kernel machine ensemble. In Nikunj C. Oza, Robi Polikar, Josef Kittler, and Fabio Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 346–355. Springer, Berlin, 2005.

[71] W. P. Lincoln and J. Skrzypek. Synergy of clustering multiple back propagation networks. In D. Touretzky, editor, *Advances in Neural Information Processing Systems-2*, pages 650–657. Morgan Kaufmann, 1990.

[72] R. P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, pages 47–64, Nov 1989.

[73] J. Little and J. Boyd. Recognizing people by their gait: the shape of motion. *Videre*, 1:1–32, 1998.

[74] G. Mani. Lowering the variance of decisions by using artificial neural network portfolios. *Neural Computation*, 3:484–486, 1991.

[75] P. Mermelstein and S. B. Davis. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28:357–366, August 1990.

[76] C. J. Merz and M. J. Pazzani. Combining neural network regression estimates with regularized linear weights. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems-9*, pages 564–570. M. I. T. Press, 1997.

[77] T. M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, Boston, MA, 1997.

[78] A. E. Nikulin, B. Dolenko, T. Bezabeh, and R. Somorjai. Near-optimal region selection for feature space reduction: novel preprocessing methods for classifying mr spectra. *NMR in Biomedicine*, 11:209–216, 1998.

[79] N. J. Nilsson. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. McGraw Hill, NY, 1965.

[80] D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems-8*, pages 535–541. M. I. T. Press, 1996.

[81] N. C. Oza and K. Tumer. Input decimated ensembles: Decorrelation through dimensionality reduction. In J. Kittler and F. Roli, editors, *Proceedings of the Second International Workshop on Multiple Classifier Systems*, pages 238–249. Springer, Berlin, 2001.

[82] B. Parmanto, P. W. Munro, and H. R. Doyle. Reducing variance of committee prediction with resampling techniques. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4):405–426, 1996.

[83] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, chapter 10. Chapmann-Hall, 1993.

[84] N. Poh and S. Bengio. An investigation of f-ratio client-dependent normalisation on biometric authentication tasks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, volume 1, pages 721–724. Institute for Electrical and Electronics Engineers, 2005.

[85] N. Poh and S. Bengio. Eer of fixed and trainable fusion classifiers: A theoretical study with application to biometric authentication tasks. In N. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 74–85. Springer, Berlin, 2005.

[86] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*. 2001.

[87] E. Prackeviciene, R. Baumgartner, and R. Somorjai. Using domain knowledge in the random subspace method: Application to the classification of biomedical spectra. In N. C. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 336–345. Springer, Berlin, 2005.

[88] S. Rajan and J. Ghosh. Exploiting class hierarchies for knowledge transfer in hyperspectral data. In N. C. Oza, R. Polikar, J. Kittler, and F. Roli, editors, *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pages 417–428. Springer, Berlin, 2005.

[89] E. Rich and K. Knight. *Artificial Intelligence*. McGraw-Hill, Inc. , 2 edition, 1991.

[90] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[91] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[92] D. W. Ruck, S. K. Rogers, M. E. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer Perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[93] A. Samal and P. Iyengar. Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 25(1):65–77, 1992.

[94] A. E. Sarhan and B. G. Greenberg. Estimation of location and scale parameters by order statistics from singly and doubly censored samples. *Annals of Mathematical Statistics Science*, 27:427–451, 1956.

[95] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[96] S. B. Serpico, P. Pellegretti, and L. Bruzzone. Feature-selection for remote-sensing data classification. In *Image and Signal Processing for Remote Sensing*, volume 2315, pages 569–577, SPIE, Rome, Italy, 1994.

[97] A. J. J. Sharkey. (editor). *Connection Science: Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4), 1996.

[98] S. Shlien. Multiple binary decision tree classifiers. *Pattern Recognition*, 23(7):757–63, 1990.

[99] P. A. Shoemaker, M. J. Carlin, R. L. Shimabukuro, and C. E. Priebe. Least squares learning and approximation of posterior probabilities on classification problems by neural network models. In *Proc. 2nd Workshop on Neural Networks, WNN-AIND91,Auburn*, pages 187–196, February 1991.

[100] N. Short. Remote sensing tutorial, URL: http://rst. gsfc. nasa. gov/starthere. html, 2000.

[101] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. Jain. Large scale evaluation of multimodal biometric authentication using state-of-the-art systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:450–455, 2005.

[102] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002.

[103] J. Suutala and J. Roning. Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option. *(to appear) Information Fusion, Special Issue on Applications of Ensemble Methods*, 2008.

[104] V. Svetnik, A. Liaw, C. Tong, and T. Wang. Application of breiman's random forest to modeling structure-activity relationships of pharmaceutical molecules. In J. Kittler, F. Roli, and T. Windeatt, editors, *Proceedings of the Fifth International Workshop on Multiple Classifier Systems*, pages 334–343. Springer, Berlin, 2004.

[105] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[106] K. A. Toh, X. Jiang, and W. Y. Yau. Exploiting local and global decisions for multi-modal biometrics verification. *IEEE Transactions on Signal Processing*, 52:3059–3072, 2004.

[107] K. Tumer and J. Ghosh. Order statistics combiners for neural classifiers. In *Proceedings of the World Congress on Neural Networks*, pages I:31–34, INNS Press, Washington D. C., 1995.

[108] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, February 1996.

[109] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4):385–404, 1996.

[110] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999.

[111] K. Tumer and J. Ghosh. Robust order statistics based ensembles for distributed data mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pages 185–210. AAAI/MIT Press, 2000.

[112] K. Tumer and J. Ghosh. Robust combining of disparate classifiers through order statistics. *Pattern Analysis and Applications*, 5:189–200, 2002.

[113] K. Tumer. *Linear and Order Statistics Combiners for Reliable Pattern Classification*. PhD thesis, The University of Texas, Austin, TX, May 1996.

[114] K. Tumer and N. C. Oza. Input decimated ensembles. *Pattern Analysis and Applications*, 6:65–77, 2003.

[115] M. Weiser and J. Brown. The coming age of calm technology. In *Beyond Calculation: The Next Fifty Years of Computing*. Springer-Verlag, New York, 1997.

[116] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2005.

[117] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[118] D. H. Wolpert. On bias plus variance. *Neural Computation*, 9(6):1211–1243, 1997.

[119] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, May 1992.

[120] J. -B. Yang and M. G. Singh. An evidential reasoning approach for multiple-attribute decision making with uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):1–19, 1994.

[121] S. Yu. *Feature Selection and Classifier Ensembles: A Study of Hyperspectral Remote Sensing*. PhD thesis, Universiteit Antwerpen, 2003.

[122] X. Zhang, J. P. Mesirov, and D. L. Waltz. Hybrid system for protein secondary structure prediction. *J. Molecular Biology*, 225:1049–63, 1992.

[123] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th International Conference on Mobile Computing and Networking*. August 2000.

[124] Y. Zhang, W. Lee, and Y. -A. Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, September 2003.