Ensemble learning

Robi Polikar (2009), Scholarpedia, 4(1):2776. doi:10.4249/scholarpedia.2776 revision #91224 [link to/cite this article]

- **Dr. Robi Polikar**, Rowan University

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting. This article focuses on classification related applications of ensemble learning, however, all principle ideas described below can be easily generalized to function approximation or prediction type problems as well.

Contents

Introduction

An ensemble-based system is obtained by combining diverse models (henceforth classifiers). Therefore, such systems are also known as multiple classifier systems, or just ensemble systems. There are several scenarios where using an ensemble based system makes statistical sense, which are discussed below in detail. However, in order to fully and practically appreciate the importance of using multiple classifier systems, it is perhaps instructive to look at a psychological backdrop to this otherwise statistically sound argument: we use such an approach routinely in our daily lives by asking the opinions of several experts before making a decision. For example, we typically ask the opinions of several doctors before agreeing to a medical procedure, we read user reviews before purchasing an item (particularly big ticket items), we evaluate future employees by checking their references, etc. In fact, even this article is reviewed by several experts before being accepted for publication. In each case, a final decision is made by combining the individual decisions of several experts. In doing so, the primary goal is to minimize the unfortunate selection of an unnecessary medical procedure, a poor product, an unqualified employee or even a poorly written and misguiding article.

Model Selection

This is perhaps the primary reason why ensemble based systems are used in practice: what is the most appropriate classifier for a given classification problem? This question can be interpreted in two different ways: i) what type of classifier should be chosen among many competing models, such as multilayer perceptron (MLP), support vector machines (SVM), decision trees, naive Bayes classifier, etc; ii) given a particular classification algorithm, which realization of this algorithm should be chosen - for example, different initializations of MLPs can give rise to different decision boundaries, even if all other parameters are kept constant. The most commonly used procedure - choosing the classifiers with the smallest error on training data - is unfortunately a flawed one. Performance on a training dataset - even when computed using a cross-validation approach - can be misleading in terms of the classification performance on the previously unseen data. Then, of all (possibly infinite) classifiers that may all have the same training - or even the same (pseudo) generalization performance as computed on the validation data (part of the training data left unused for evaluating the classifier performance) - which one should be chosen? Everything else being equal, one may be tempted to choose at random, but with that decision comes the risk of choosing a particularly poor model. Using an ensemble of such models - instead of choosing just one - and combining their outputs by - for example, simply averaging them - can reduce the risk of an unfortunate selection of a particularly poorly performing classifier. It is important to emphasize that there is no guarantee that the combination of multiple classifiers will always perform better than the best individual classifier in the ensemble. Nor an improvement on the ensemble's average performance can be guaranteed except for certain special cases (Fumera 2005). Hence combining classifiers may not necessarily beat the performance of the best classifier in the ensemble, but it certainly reduces the overall risk of making a particularly poor selection.

In order for this process to be effective, the individual experts must exhibit some level of *diversity* among themselves, as described later in this article in more detail. Within the classification context, then, the diversity in the classifiers – typically achieved by using different training parameters for each classifier – allows individual classifiers to generate different decision boundaries. If proper diversity is achieved, a different error is made by each classifier, strategic combination of which can then reduce the total error. Figure 1 graphically illustrates this concept, where each classifier - trained on a different subset of the available training data - makes different errors (shown as instances with dark borders), but the combination of the (three) classifiers provides the best decision boundary.

Too much or too little data

Ensemble based systems can be - perhaps surprisingly - useful when dealing with large volumes of data or lack of adequate data. When the amount of training data is too large to make a single classifier training difficult, the data can be strategically partitioned into smaller subsets. Each partition can then be used to train a separate classifier which can then be combined using an appropriate combination rule (see below for different combination rules). If, on the other hand, there is too little data, then bootstrapping can be used to train different classifiers using

different **bootstrap samples** of the data, where each bootstrap sample is a random sample of the data drawn with replacement and treated as if it was independently drawn from the underlying distribution (Efron 1979).

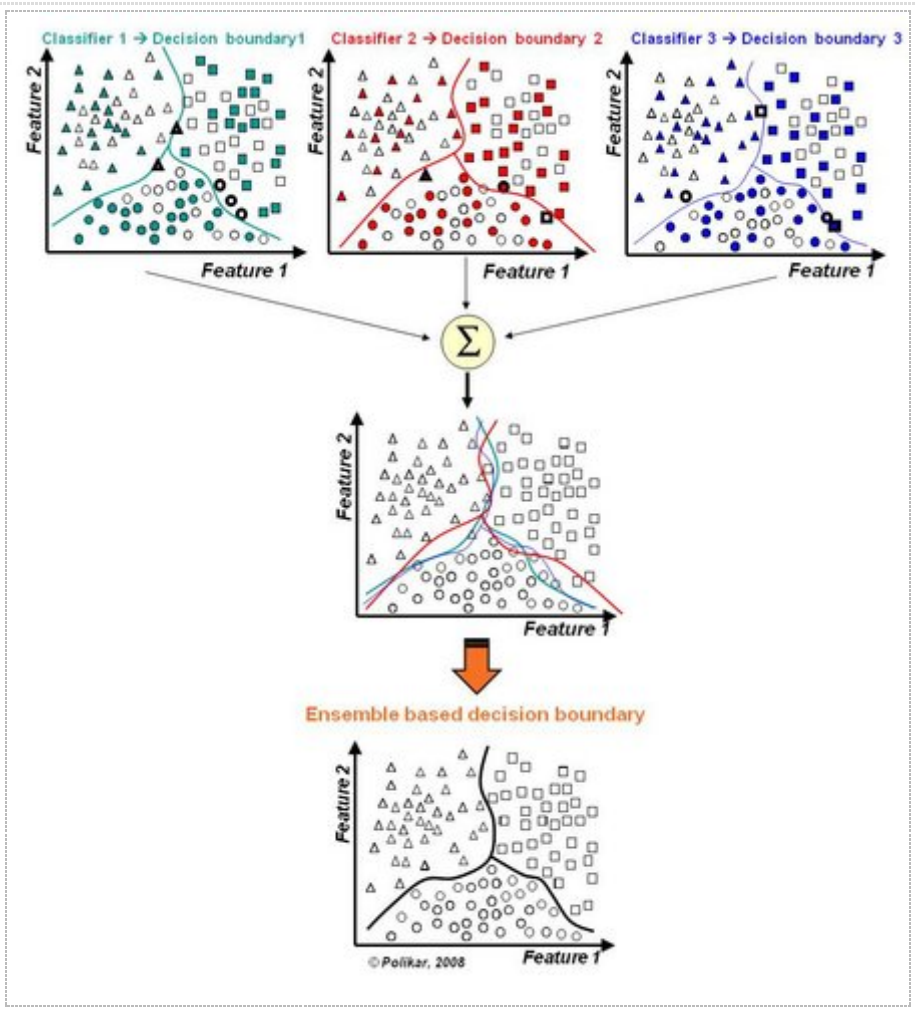

Figure 1: Combining an ensemble of classifiers for reducing classification error and/or model selection.

Divide and Conquer

Certain problems are just too difficult for a given classifier to solve. In fact, the decision boundary that separates data from different classes may be too complex, or lie outside the space of functions that can be implemented by the chosen classifier model. Consider the two dimensional, two-class problem with a complex decision boundary depicted in Figure 2. A linear classifier, one that is capable of learning linear boundaries, cannot learn this complex non-linear boundary. However, appropriate combination of an ensemble of such linear classifiers can learn any non-linear boundary. As an example, assume that we have access to a classifier model that can generate circular boundaries. Such a classifier cannot learn the boundary shown in Figure 2. Now consider a collection of circular decision boundaries generated by an ensemble of such classifiers as shown in Figure 3, where each classifier labels the data as class O or class X, based on whether the instances fall within or outside of its boundary. A decision based on the majority voting of a sufficient number of such classifiers can easily learn this complex non-circular boundary (subject to i) classifier outputs be independent, and ii) at least half of the classifiers classify an instance correctly - see the discussion on voting based combination rules for proof and detailed analysis of this approach). In a sense, the classification system follows a divide-and-conquer approach by dividing the data space into smaller and easier-to-learn partitions, where each classifier learns only one of the simpler partitions. The underlying complex decision boundary can then be approximated by an appropriate combination of different classifiers.

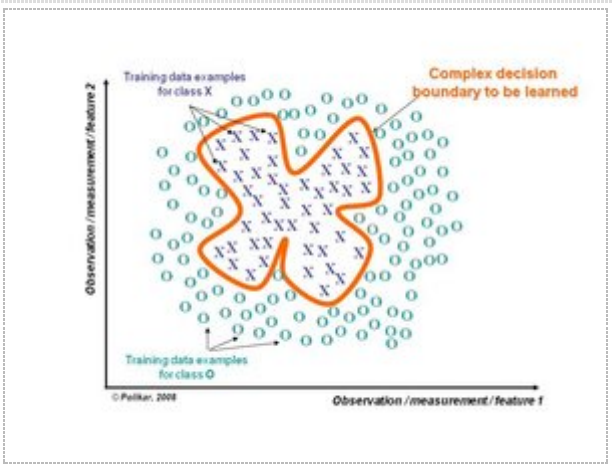

Figure 2: A complex decision boundary that cannot be realized by circular boundaries.

Data Fusion

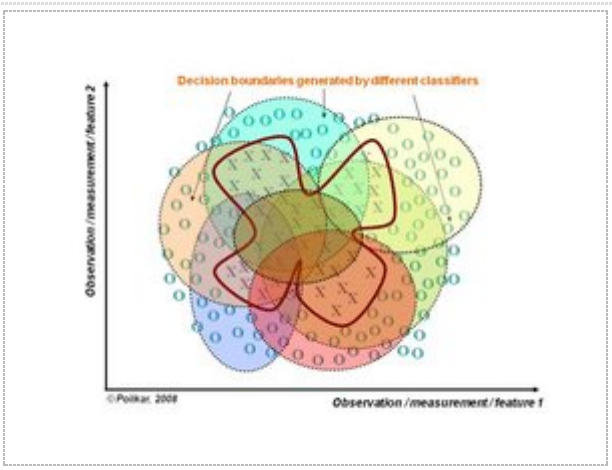

Figure 3: A combination of several circular boundaries can realize this complex boundary.

In many applications that call for automated decision making, it is not unusual to receive data obtained from different sources that may provide complementary information. A suitable combination of such information is known as **data or information fusion,** and can lead to improved accuracy of the classification decision compared to a decision based on any of the individual data sources alone. For example, for diagnosis of a neurological disorder, a neurologist may use the electroencephalogram (one-dimensional time series data), magnetic resonance imaging MRI, functional MRI, or positron emission tomography PET scan images (two-dimensional spatial data), the amount of certain chemicals in the cerebrospinal fluid along with the subjects demographics such as age, gender, education level of the subject, etc. (scalar and or categorical values). These heterogeneous features cannot be used all together to train a single classifier (and even if they could - by converting all features into a vector of scalar values - such a training is unlikely to be successful). In such cases, an ensemble of classifiers can be used (Parikh 2007), where a separate classifier is trained on each of the feature sets independently. The decisions made by each classifier can then be combined by any of the combination rules described below.

Confidence Estimation

The very structure of an ensemble based system naturally allows assigning a confidence to the decision made by such a system. Consider having an ensemble of classifiers trained on a classification problem. If a vast majority of the classifiers agree with their decisions, such an outcome can be interpreted as the ensemble having high confidence in its decision. If, however, half the classifiers make one decision and the other half make a different decision, this can be interpreted as the ensemble having low confidence in its decision. It should be noted that an ensemble having high confidence in its decision does not mean that decision is correct, and conversely, a decision made with low confidence need not be incorrect. However, it has been shown that a properly trained ensemble decision is usually correct if its confidence is high, and usually incorrect if its confidence is low. Using such an approach then, the ensemble decisions can be used to estimate the posterior probabilities of the classification decisions (Muhlbaier 2005).

Other Reasons for Using Ensemble System

In his 2000 review article, Dietterich lists three primary reasons for using an ensemble based system: i) statistical; ii) computational; and iii) representational (Dietterich 2000). Note that these reasons are similar to those listed above. The statistical reason is related to lack of adequate data to properly represent the data distribution; the computational reason is the model selection problem, where among many models that can solve a given problem, which one we should choose. Finally, the representational reason is to address to cases when the chosen model cannot properly represent the sought decision boundary, which is discussed under divide and conquer section above.

History

Perhaps one of the earliest work on ensemble systems is Dasarathy and Sheela's 1979 paper (Dasarathy 1979), which first proposed using an ensemble system in a divide-and-conquer fashion, partitioning the feature space using two or more classifiers. Over a decade later, Hansen and Salamon (Hansen 1990) showed the variance reduction property of an ensemble system, and that the generalization performance of a neural network can be

improved by using an ensemble of similarly configured neural networks. But it was Schapire's work that put the ensemble systems at the center of machine learning research, as he proved that a strong classifier in probably approximately correct (PAC) sense can be generated by combining weak classifiers through a procedure he called boosting, (Schapire 1990). Boosting was the predecessor of the AdaBoost family of algorithms - which arguably became one of the most popular machine learning algorithms in recent times. Since these seminal works, research in ensemble systems have expanded rapidly, appearing often in the literature under many creative names and ideas. The long list includes composite classifier systems (Dasarathy 1979), mixture of experts (Jacobs 1991, Jordan 1994), stacked generalization (Wolpert 1992), combination of multiple classifiers (Ho 1994, Rogova 1994, Lam 1995, Woods 1997), dynamic classifier selection (Woods 1997), classifier fusion (Cho 1995, Kuncheva 2001), classifier ensembles, among many others. These approaches usually differ from each other primarily in two ways: i) specific procedure used for generating individual classifiers; and/or ii) the strategy employed for combining the classifiers. Ensemble systems can also be categorized based on whether classifiers are selected or fused (Woods 1997, Kuncheva 2001, Kuncheva 2002, Ho 2000): In *classifier selection*, each classifier is trained to become an expert in some local area of the feature space (as in Figure 3). The combination of the classifiers is then based on the given instance: the classifier trained with data closest to the vicinity of the instance, according to some distance metric, is given the highest credit. One or more local experts can be nominated to make the decision (Jacobs 1991, Woods 1997, Alpaydin 1996, Giacinto 2001). In *classifier fusion*, all classifiers are trained over the entire feature space. In this case, the classifier combination involves merging the individual (usually weaker and/or diverse) classifiers to obtain a single (stronger) expert of superior performance. Examples of this approach include bagging predictors (Breiman 1996), boosting (Schapire 1990), AdaBoost (Freund 2001) and their many variations. The combination may apply to classification labels only, or to the class-specific continuous valued outputs of the individual experts (Kittler 1998). In the latter case, classifier outputs are often normalized to the [0, 1] interval, and these values are interpreted as the support given by the classifier to each class, or as class-conditional posterior probabilities. Such interpretation allows forming an ensemble through algebraic combination rules (majority voting, maximum / minimum / sum / product or other combinations of posterior probabilities) (Kittler 1998, Kuncheva 2002, Roli 2002), fuzzy integral (Cho 1995), the Dempster-Shafer based fusion (Rogova 1994), and more recently, the decision templates (Kuncheva 2001). A sample of the immense literature on classifier combination can be found in Kuncheva's recent book (Kuncheva 2005), the first text devoted to theory and implementation of ensemble based classifiers, and references therein. Two recent tutorials written by the current curator of this article also provide a comprehensive overview of ensemble systems (Polikar 2006, Polikar 2007).

Diversity

The success of an ensemble system - that is, its ability to correct the errors of some of its members - rests squarely on the diversity of the classifiers that make up the ensemble. After all, if all classifiers provided the same output, correcting a possible mistake would not be possible. Therefore, individual classifiers in an ensemble system need to make different errors on different instances. The intuition, then, is that if each classifier makes different errors, then a strategic combination of these classifiers can reduce the total error, a concept not too dissimilar to low pass filtering of the noise. Specifically, an ensemble system needs classifiers whose decision boundaries are adequately different from those of others. Such a set of classifiers is said to be diverse. Classifier diversity can be achieved in several ways. Preferably, the classifier outputs should be class-conditionally independent, or better yet negatively correlated. The most popular method is to use different training datasets to train individual classifiers. Such datasets are often obtained through re-sampling techniques, such as bootstrapping or bagging (see below), where training data subsets are drawn randomly, usually with replacement, from the entire training data. To ensure that individual boundaries are adequately different, despite using substantially similar training data, weaker or more unstable classifiers are used as base models, since they can generate suffi-ciently different decision boundaries even for small perturbations in their training parameters.

Another approach to achieve diversity is to use different training parameters for different classifiers. For example, a series of multilayer perceptron (MLP) neural networks can be trained by using different weight initializations, number of layers / nodes, error goals, etc. Adjusting such parameters allows one to control the instability of the individual classifiers, and hence contribute to their diversity. Alternatively, entirely different type of classifiers, such MLPs, decision trees, nearest neighbor classifiers, and support vector machines can also be combined for added diversity. Finally, diversity can also be achieved by using different features, or different subsets of existing features. In fact, generating different classifiers using random feature subsets is known as the random subspace method (Ho 1998), as described later in this article.

For a comprehensive review of diversity issues, see (Brown 2005).

Commonly used ensemble learning algorithms

Bagging

Bagging, which stands for *bootstrap aggregating*, is one of the earliest, most intuitive and perhaps the simplest ensemble based algorithms, with a surprisingly good performance (Breiman 1996). Diversity of classifiers in bagging is obtained by using bootstrapped replicas of the training data. That is, different training data subsets are randomly drawn – with replacement – from the entire training dataset. Each training data subset is used to train a different classifier of the same type. Individual classifiers are then combined by taking a simple majority vote of their decisions. For any given instance, the class chosen by most number of classifiers is the ensemble decision. Since the training datasets may overlap substantially, additional measures can be used to increase diversity, such as using a subset of the training data for training each classifier, or using relatively weak classifiers (such as decision stumps). The pseudocode of Bagging is provided in Figure 4.

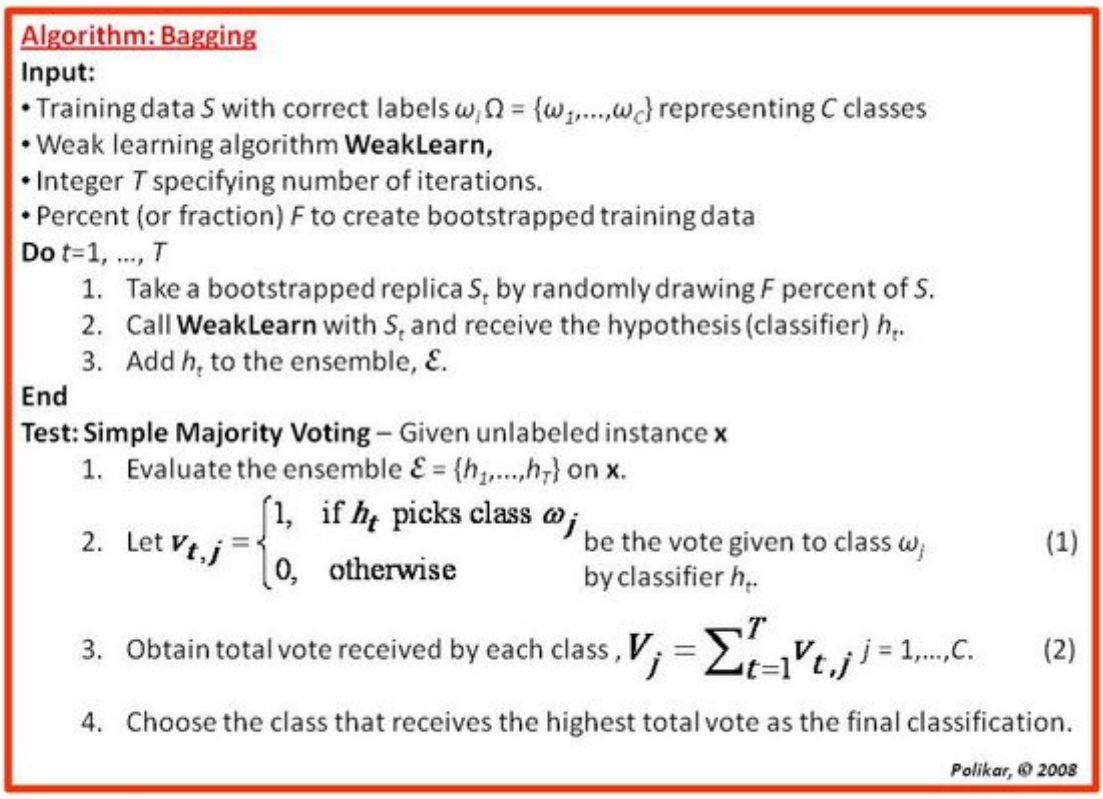

Figure 4: Algorihm Bagging.

Boosting

Similar to bagging, boosting also creates an ensemble of classifiers by resampling the data, which are then combined by majority voting. However, in boosting, resampling is strategically geared to provide the most informative training data for each consecutive classifier. In essence, each iteration of boosting creates three weak classifiers: the first classifier C_1 is trained with a random subset of the available training data. The training data subset for the second classifier C_2 is chosen as the most informative subset, given C_1. Specifically, C_2 is trained on a training data only half of which is correctly classified by C_1, and the other half is misclassified. The third classifier C_3 is trained with instances on which C_1 and C_2 disagree. The three classifiers are combined through a three-way majority vote. The pseudocode and implementation detail of boosting is shown in Figure 5.

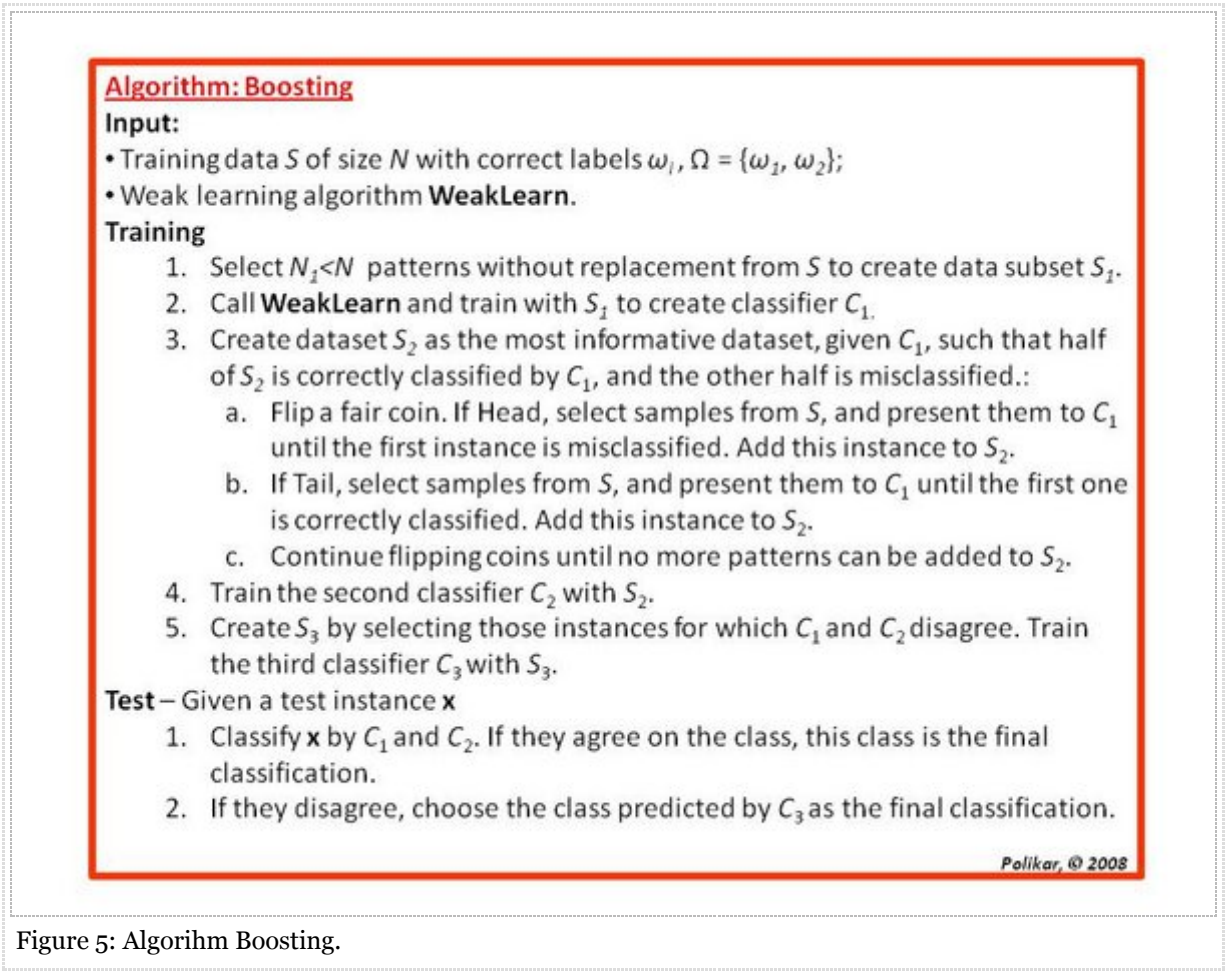

Figure 5: Algorihm Boosting.

Schapire showed that the error of this algorithm has an upper bound: if the algorithm A used to create the classifiers C_1, C_2, C_3 has an error of ϵ (as computed on S), then the error of the ensemble is bounded above by $f(\epsilon) = 3\epsilon^2 - 2\epsilon^3$. Note that $f(\epsilon) \leq \epsilon$ for $\epsilon < 1/2$. That is, as long as the original algorithm A can do at least better than random guessing, then the boosting ensemble that combines three classifiers generated by A on the above described three distributions of S, will always outperform A. Also, the ensemble error is a training error bound. Hence, a stronger classifier is generated from three weaker classifiers. A strong classifier in the strict PAC learning sense can then be created by recursive applications of boosting. A particular limitation of boosting is that it applies only to binary classification problems. This limitation is removed with the AdaBoost algorithm.

AdaBoost

Arguably the best known of all ensemble-based algorithms, **AdaBoost** (Adaptive Boosting) extends boosting to multi-class and regression problems (Freund 2001). AdaBoost has many variations, such as AdaBoost.M1 for classification problems where each classifier can attain a weighted error of no more than 1/2, AdaBoost.M2 for those weak classifiers that cannot achieve this error maximum (particularly for problems with large number of

classes, where achieving an error of less than 1/2 becomes increasingly difficult), AdaBoost.R (for regression problems), among many others. The most popular of AdaBoost's variations, AdaBoost.M1 for multi-class problems, is described here, whose pseudocode appears in Figure 6.

In AdaBoost.M1, bootstrap training data samples are drawn from a distribution D that is iteratively updated such that subsequent classifiers focus on increasingly difficult instances. This is done by adjusting D such that previously misclassified instances are more likely to appear in the next bootstrap sample. The classifiers are then combined through weighted majority voting. The distribution D starts out as uniform (Equation 3 in Figure 6), so that all instances have equal probability to be drawn into the first data subset S_1. At each iteration t, a new training set is drawn, and a weak classifier is trained to produce a hypothesis h_t. The error of this hypothesis with respect to the current distribution is calculated as the sum of distribution weights of the instances misclassified by h_t (Equation 4). AdaBoost.M1 requires that this error be less than 1/2. If a classifier cannot meet this requirement, the algorithm aborts. The normalized error β_t, is then computed (Equation 5) so that the actual error that is in the [0 0.5] interval is mapped to [0 1] interval. The normalized error is used in the distribution update rule of Equation 6. Note that $D_t(i)$ is reduced by a factor of $\beta_t, 0 < \beta_t < 1$, if $\mathbf{x}_i$ is correctly classified by h_t, and left unchanged otherwise. When the distribution is normalized so that $D_{t+1}(i)$ is a proper distribution, the weights of those instances that are misclassified are effectively increased. This update rule ensures that the weights of all correctly classified instances and the weights of all misclassified instances always add up to 1/2. More specifically, the requirement for the training error of the base classifier to be less than 1/2 forces teh algorithm to correct at least one mistake made by the previous base model. Once the training is complete, test data are classified by this ensemble of T classifiers using weighted majority voting, where each classifier receives a voting weight that is inversely proportional to its normalized error (Equation 7). The weighted majority voting then chooses the class ω receiving the highest total vote from all classifiers.

Algorithm: AdaBoost.M1 © 2008, Polikar

Inputs

- Training data $S = \{\mathbf{x}_1, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in X$ with correct labels $y_i \in \Omega$, $\Omega = \{\omega_1, \ldots, \omega_C\}$;
- Weak learning algorithm **WeakLearn**;
- integer T, number of classifiers

Initialize $D_1(i) = 1/n;\ i = 1, \cdots, n$ (3)

Do for $t = 1, 2, \ldots, T$:

1. Draw bootstrap training data subset S_t according to current distribution D_t.
2. Train **WeakLearn** with S_t, receive hypothesis $h_t : X \to \Omega$
3. Calculate the error of h_t: $\varepsilon_t = \sum_{i=1}^{n} I\left[\left[h_t(\mathbf{x}_i) \neq y_i\right]\right] \cdot D_t(i) = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$ (4)
   If $\varepsilon_t > 1/2$, **abort**.
4. Calculate normalize error $\beta_t = \varepsilon_t / (1 - \varepsilon_t) \Rightarrow 0 \leq \beta_t \leq 1$ (5)
5. Update distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t, & \text{if } h_t(\mathbf{x}_i) = y_i \\ 1, & \text{otherwise} \end{cases}$ (6)

   where Z_t is a normalization constant chosen so that D_{t+1} is a proper distribution.

Test – Weighted Majority Voting: Given an unlabeled instance $\mathbf{z}$,

1. Obtain total vote received by each class, $V_j = \sum_{t:h_t(\mathbf{z})=\omega_j} \log(1/\beta_t)$ $j = 1, \ldots, C$. (7)
2. Choose the class that receives the highest total vote as the final classification.

Figure 6: Algorihm AdaBoost.M1.

The theoretical analysis of this algorithm shows that the ensemble (training) error E is bounded above

$$E < 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)} .$$

Since $\epsilon_t < 1/2$, ensemble error E monotonically decreases as new classifiers are added. This result may appear to go against the conventional wisdom (see Occam's razor) indicating that adding too many classifiers – beyond a certain limit – would eventually lead to overfitting of the data. Empirical results have shown, however, that AdaBoost is surprising resistant to overfitting, a phenomenon whose explanation is based on the margin theory (Schapire 1998).

Stacked Generalization

In Wolpert's **stacked generalization** (or **stacking**), an ensemble of classifiers is first trained using bootstrapped samples of the training data, creating *Tier 1 classifiers*, whose outputs are then used to train a *Tier 2 classifier (meta-classifier)* (Wolpert 1992). The underlying idea is to learn whether training data have been properly learned. For example, if a particular classifier incorrectly learned a certain region of the feature space, and hence consistently misclassifies instances coming from that region, then the Tier 2 classifier may be able to learn this behavior, and along with the learned behaviors of other classifiers, it can correct such improper training. Cross validation type selection is typically used for training the Tier 1 classifiers: the entire training dataset is divided into T blocks, and each Tier-1 classifier is first trained on (a different set of) T-1 blocks of the training data. Each classifier is then evaluated on the Tth (pseudo-test) block, not seen during training. The outputs of these classifiers on their pseudo-training blocks, along with the actual correct labels for those blocks constitute the training dataset for the Tier 2 classifier (see Figure 7).

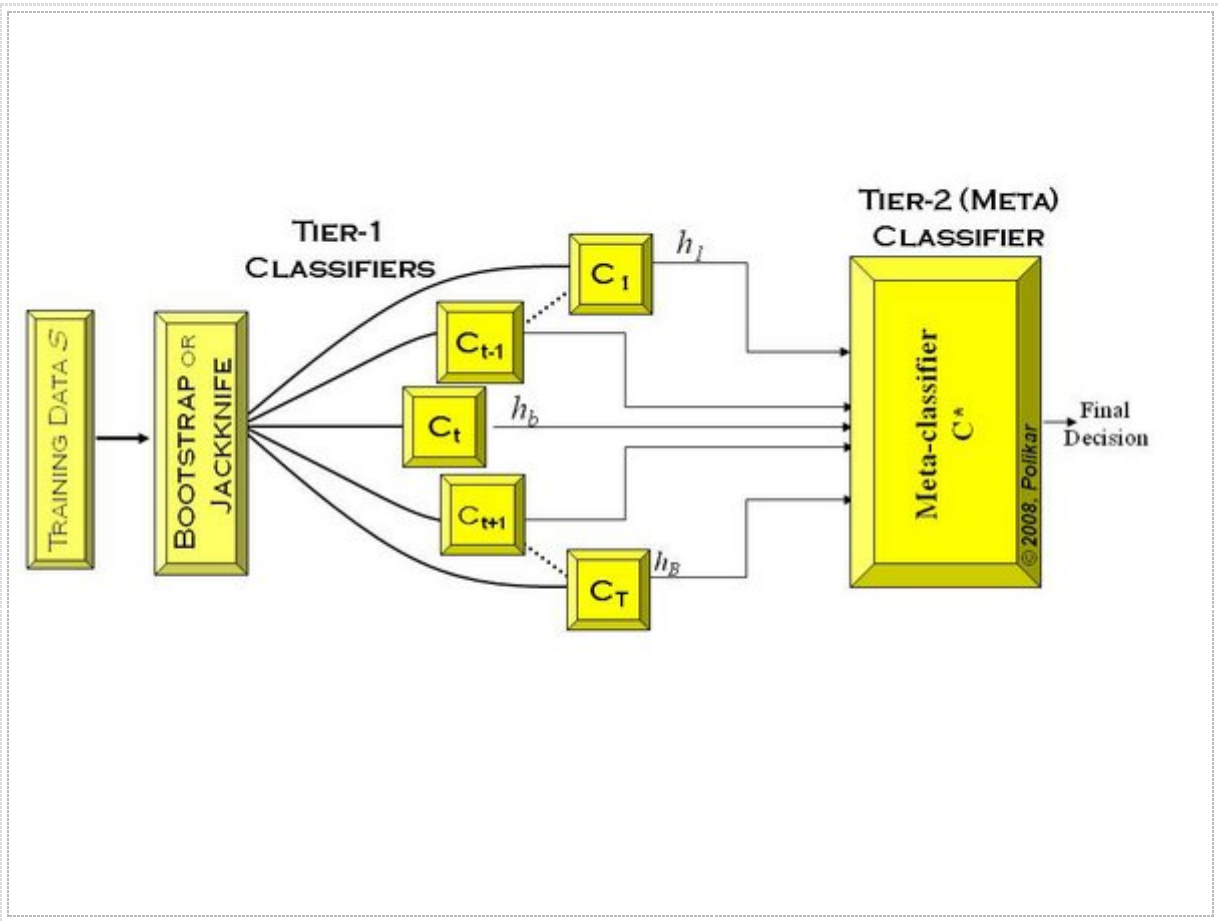

Figure 7: Stacked generalization.

Mixture of Experts

Jordan and Jacobs' **mixture of experts** (Jacobs 1991) generates several experts (classifiers) whose outputs are combined through a (generalized) linear rule. The weights of this combination are determined by a gating network, typically trained using the expectation maximization (EM) algorithm. Both the experts themselves and the gating network requires the input instances for training. Several mixture-of-experts models can also be further combined to obtain a hierarchical mixture of experts (Jordan 1994). Mixture of experts are particularly useful when different experts are trained on different parts of the feature space, or when heterogeneous sets of features are available to be used for a data fusion problem. Figure 8 illustrates the mixture of experts model.

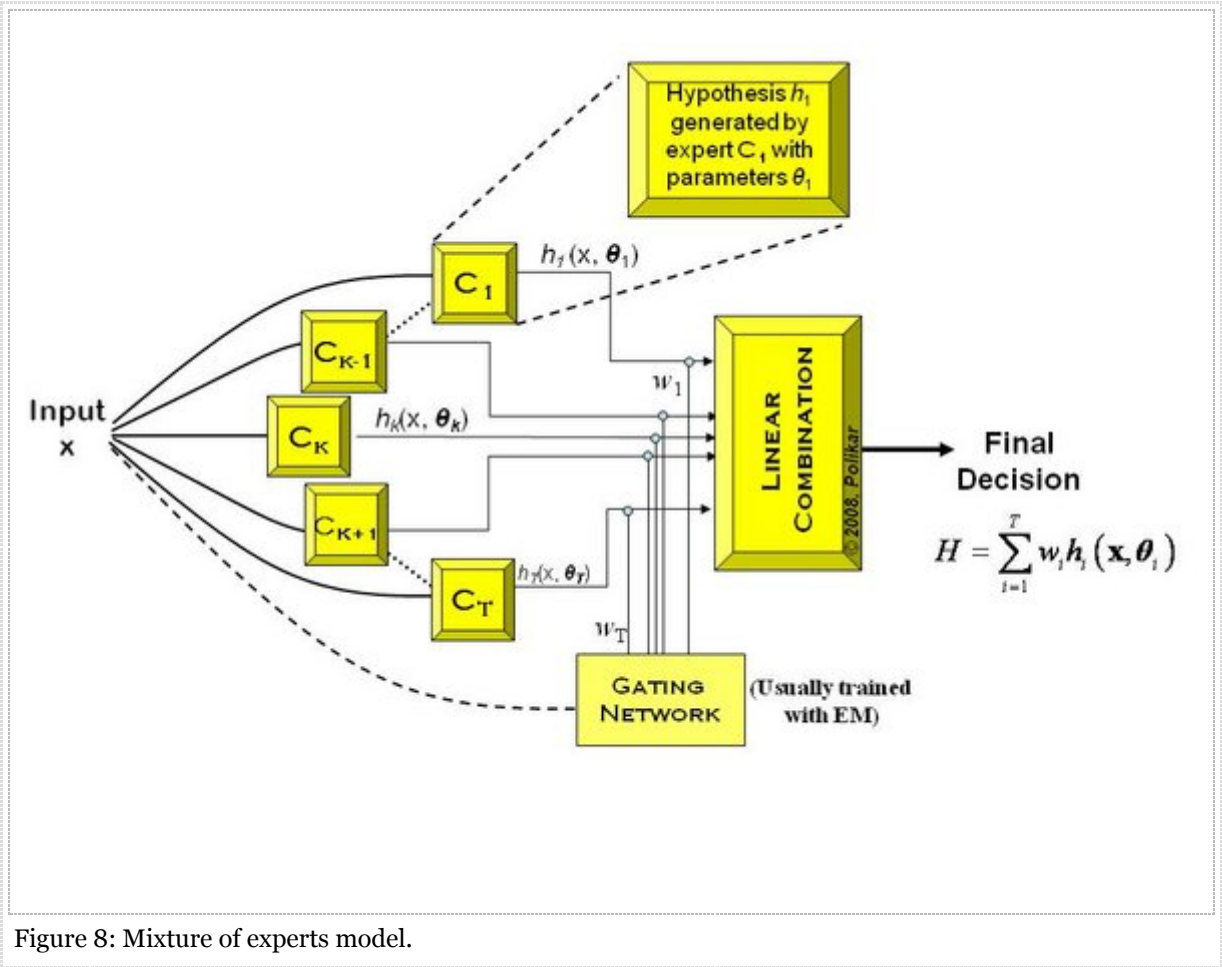

Figure 8: Mixture of experts model.

Ensemble combination rules

The algorithms described above have their built in combination rules, such as simple majority voting for bagging, weighted majority voting for AdaBoost, a separate classifier for stacking, etc. However, an ensemble of classifiers can be trained simply on different subsets of the training data, different parameters of the classifiers, or even with different subsets of features as in random subspace models. The classifiers can then be combined using one of several different combination rules. Some of these combination rules operate on class labels only, whereas others need continuous outputs that can be interpreted as support given by the classifier to each of the classes. Xu et al (Xu 1992). defines three types of base model outputs to be used for classifier combination. i)Abstract-level output, where each classifier outputs a unique class label for each input pattern; ii)Rank-level output, where each classifier outputs a list of ranked class labels for each input pattern; and iii)measurement-level output, where each classifier outputs a vector of continuous-valued measures that can represent estimates of class posterior probabilities or class-related confidence values that represent the support for the possible classification hypotheses.

For the former case of abstract level outputs, the decision of the tth classifier is defined as $d_{t,j} \in \{0, 1\}, t = 1, \cdots, T; j = 1, \cdots, C$ where T is the number of classifiers and C is the number of classes. If tth classifier chooses class ω_j, then $d_{t,j} = 1$, and 0, otherwise. For those combination rules that need continuous

outputs, the classifier outputs are defined as $d_{t,j} \in [0, 1]$ .Such outputs are usually normalized so that they add up to 1, which can the be interpreted as the normalized support given to clas j by classifier t, or even as the estimate of the posterior probability $P_t(\omega_j|\mathbf{x})$.

Algebraic combiners

Algebraic combiners are *non-trainable combiners*, where continuous valued outputs of classifiers are combined through an algebraic expression, such as minimum, maximum, sum, mean, product, median, etc. In each case, the final ensemble decision is the class j that receives the largest support $\mu_j(\mathbf{x})$ after the algebraic expression is applied to individual supports obtained by each class. Specifically

$h_{final}(\mathbf{x}) = \arg\max_j \mu_j(\mathbf{x})$,where the final class supports are computed as follows:

- **Mean rule:** $\mu_j(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} d_{t,j}(\mathbf{x})$
- **Sum rule:** $\mu_j(\mathbf{x}) = \sum_{t=1}^{T} d_{t,j}(\mathbf{x})$ (provides identical final decision as the mean rule)
- **Weighted sum rule:** $\mu_j(\mathbf{x}) = \sum_{t=1}^{T} w_t d_{t,j}(\mathbf{x})$ (where w_t is the weight assigned to the tth classifier h_t according to some measure of performance)
- **Product rule:** $\mu_j(\mathbf{x}) = \prod_{t=1}^{T} d_{t,j}(\mathbf{x})$
- **Maximum rule** $\mu_j(\mathbf{x}) = \max_{t=1,\cdots,T}\{d_{t,j}(\mathbf{x})\}$
- **Minimum rule** $\mu_j(\mathbf{x}) = \min_{t=1,\cdots,T}\{d_{t,j}(\mathbf{x})\}$
- **Median rule** $\mu_j(\mathbf{x}) = \underset{t=1,\cdots,T}{\text{med}}\{d_{t,j}(\mathbf{x})\}$
- **Generalized mean rule** $\mu_{j,\alpha}(\mathbf{x}) = \left(\frac{1}{T}\sum_{t=1}^{T} d_{t,j}(\mathbf{x})^{\alpha}\right)^{1/\alpha}$
  - $\alpha \to -\infty \Rightarrow$ Minimum rule
  - $\alpha \to \infty \Rightarrow$ maximum rule
  - $\alpha = 0 \Rightarrow$ Geometric mean rule
  - $\alpha = 1 \Rightarrow$ Mean rule

Voting based methods

Voting based methods operate on labels only, where $d_{t,j}$ is 1 or 0 depending on whether classifier t chooses j, or not, respectively. The ensemble then chooses class J that receives the largest total vote:

- **Majority (plurality) voting**

$\sum_{t=1}^{T} d_{t,J}(\mathbf{x}) = \max_{j=1,\cdots,C}\sum_{t=1}^{T} d_{t,j}$

Under the condition that the classifier outputs are independent, it can be shown the majority voting combination will always lead to a performance improvement. If there are a total of T classifiers for a two-class problem, the ensemble decision will be correct if at least $\lfloor T/2 + 1 \rfloor$ classifiers choose the correct class. Now, assume that each classifier has a probability p of making a correct decision. Then, the ensemble's probability of making a correct decision has a binomial distribution, specifically, the probability of choosing $k > \lfloor T/2 + 1 \rfloor$ correct classifiers out of T is $P_{ens} = \sum_{k=(T/2)+1}^{T} \binom{T}{k} p^k (1-p)^{T-k}$

Then,

$$P_{ens} \to 1 ,$$

as $T \to \infty$ if $p > 0.5$

$$P_{ens} \to 0 ,$$

as $T \to \infty$ if $p < 0.5$

Note that the requirement of $p > 0.5$ is necessary and sufficient for a two class problem, whereas it is sufficient, but not necessary for multi class problems.

- **Weighted majority voting**

$\sum_{t=1}^{T} w_t d_{t,J}(\mathbf{x}) = \max_{j=1,\cdots,C} \sum_{t=1}^{T} w_t d_{t,j}$

The optimal weights for the weighted majority voting rule can be shown to be $w_t \propto \frac{p_t}{1-p_t}$ if the T classifiers are class-conditionally independent with accuracies $p_1, \cdots, p_T$

Other combination rules

There are several other combination rules, which are arguably more sophisticated than the ones listed above. These include *Borda count* which takes the rankings of the class supports into consideration; *behavior knowledge space* (Huang 1993) which uses a lookup table that lists the most common correct classes for every possible class combinations given by the classifiers; "decision templates" (Kuncheva 2001) which compute a similarity measure between the current decision profile of the unknown instance and the average decision profiles of instances from each class; and *Dempster-Schafer rule* which computes the plausibility based belief measures for each class. However, many empirical studies have shown that simpler rules such as the sum rule or the (weighted) majority voting often work remarkably well (Kittler 1998). For a detailed overview of these and other combination rules, see (Kuncheva 2005).

Other applications of ensemble systems

Ensemble based systems can be used in problem domains other than improving the generalization performance of a classifier.

Incremental learning

Incremental learning refers to the ability of an algorithm to learn from new data that may become available after a classifier (or a model) has already been generated from a previously available dataset. An algorithm is said to be an **incremental learning algorithm** if, for a sequence of training datasets (or instances), it produces a sequence of hypotheses, where the current hypothesis describes all data that have been seen thus far, but depends only on previous hypotheses and the current training data (Polikar 2001). Hence, an incremental learning algorithm must learn the new information, and retain previously acquired knowledge, without having access to previously seen data. A commonly used approach to learn from additional data - discarding the existing classifier, and retraining a new one with the old and new data combined together does not meet the definition of incremental learning, since it causes catastrophic forgetting of all previously learned information and uses previous data. Ensemble based systems can be used for such problems by training an additional classifier (or an additional ensemble of classifiers) on each dataset that becomes available. Learn^{++} primarily for incremental learning problems that do not introduce new classes (Polikar2001), and Learn^{++}.NC for those that introduce new classes with additional datasets (Muhlbaier 2008) are two examples of ensemble based incremental learning algorithms.

Error correcting output codes

Error correcting output codes (ECOC) are commonly used in information theory for correcting bit reversals caused by noisy communication channels, or in machine learning for converting binary classifiers, such as support vector machines, to multi-class classifiers by decomposing a multi-class problem into several two-class problems (Allwein 2000). Dietterich and Bakiri introduced ECOC to be used within the ensemble setting (Dietterich 1995). The idea is to use a different class encoding for each member of the ensemble. The encodings constitute a binary C by T code matrix, where C and T are the number of classes and ensemble size, respectively, combined by the minimum Hamming distance rule.

Class	CLASSIFIERS														
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
ω_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ω_2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
ω_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
ω_4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
ω_5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Figure 9: Error correcting output codes exhaustive code matrix for five classes.

Figure 8 shows a particular code matrix for a 5-class problem that uses 15 encodings. This encoding, suggested in (Dietterich 1995), is a (pseudo) exhaustive coding because it includes all possible non-trivial and non-repeating codes. In this formulation, the individual classifiers are trained on several meta two-class problems: for example, C_3 recognizes two meta-classes: original classes ω_1 and ω_4 constitute one class, and the others constitute the second class. During testing, each classifier outputs a "0" or "1" creating a $2^{C-1} - 1$ long output code vector. This vector is compared to each code word in the code matrix, and the class whose code word has the shortest Hamming distance to the output vector is chosen as the ensemble decision. More specifically, the support for class ω_j is given as $\mu_j(\mathbf{x}) = -\sum_{t=1}^{T} |o_t - M_{j,t}|$ where $o_t \in \{0, 1\}$ is the output of the tth binary classifier, and M is the code matrix. The negative sign converts the distance metric into a support value, whose largest value can be zero in case of a perfect match. For example, the output [0 1 1 1 0 1 0 1 0 1 0 1 0 1 0] is closest to ω_5 code word with a Hamming distance of 1 (support of -1), and hence ω_5 would be chosen for this output. Note that this output does not match any of the code words exactly, and this is where the error correcting ability of the ECOC lies. In fact, the larger the minimum Hamming distance between code words, the more resistant the ECOC ensemble becomes to misclassifications of individual classifiers.

Feature selection

As mentioned earlier in this article, one way to improve diversity in the ensemble is to train individual classifiers different subsets of the available features. Selecting the feature subsets at random is known as the **random subspace method,** a term coined by (Ho 1998), who used it on constructing decision tree ensembles. Subset selection need not be random, however: Oza and Tumer propose input decimation, where the features are selected based on their correlation with the class labels (Oza 2001).

References

- B. Efron (1979), "Bootstrap methods: another look at the jackknife," The Annals of Statistics, vol. 7, no. 1, pp. 1-26.
- B. V. Dasarathy and B. V. Sheela (1979), "Composite classifier system design: concepts and me-thodology," Proceedings of the IEEE, vol. 67, no. 5, pp. 708-713.
- T.G. Dietterich, "Ensemble Methods in Machine Learning," Int. Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, Vol. 1857, pp. 1-15, 2000, Springer-Verlag.
- L. K. Hansen and P. Salamon, "Neural network ensembles," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, 1990.
- R. E. Schapire, "The Strength of Weak Learnability," Machine Learning, vol. 5, no. 2, pp. 197-227, 1990.
- R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local ex-perts," Neural Computation, vol. 3, pp. 79-87, 1991.

- M. J. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," Neural Computation, vol. 6, no. 2, pp. 181-214, 1994.
- D. H. Wolpert, "Stacked generalization," Neural Networks, vol. 5, no. 2, pp. 241-259, 1992.
- T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," IEEE Trans. on Pattern Analy. Machine Intel., vol. 16, no. 1, pp. 66-75, 1994.
- G. Rogova, "Combining the results of several neural network classifiers," Neural Networks, vol. 7, no. 5, pp. 777-781, 1994.
- L. Lam and C. Y. Suen, "Optimal combinations of pattern classifiers," Pattern Recognition Letters, vol. 16, no. 9, pp. 945-954, 1995.
- K. Woods, W. P. J. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 4, pp. 405-410, 1997.
- S. B. Cho and J. H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," IEEE Transactions on Systems, Man and Cybernetics, vol. 25, no. 2, pp. 380-384, 1995.
- L. I. Kuncheva, J. C. Bezdek, and R. Duin, "Decision templates for multiple classifier fu-sion: an experimental comparison," Pattern Recognition, vol. 34, no. 2, pp. 299-314, 2001.
- L. I. Kuncheva, "Classifier Ensembles for Changing Environments," 5th Int. Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, F. Roli, J. Kittler, and T. Windeatt, Eds., vol. 3077, pp. 1-15, 2004.
- L. I. Kuncheva, Combining Pattern Classifiers, Methods and Algorithms. New York, NY: Wiley Interscience, 2005.
- E. Alpaydin and M. I. Jordan, "Local linear perceptrons for classification," IEEE Transac-tions on Neural Networks, vol. 7, no. 3, pp. 788-792, 1996.
- G. Giacinto and F. Roli, "Approach to the automatic design of multiple classifier systems," Pattern Recognition Letters, vol. 22, no. 1, pp. 25-33, 2001.
- G. Fumera and F. Roli, "A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, no. 6 pp. 942-956, 2005.
- L. Breiman, "Bagging predictors," Machine Learning, vol. 24, no. 2, pp. 123-140, 1996.
- Y. Freund and R. E. Schapire, "Decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119-139, 1997.
- J. Kittler, M. Hatef, R. P. W. Duin, and J. Mates, "On combining classifiers," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no. 3, pp. 226-239, 1998.
- L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 2, pp. 281-286, 2002.
- S. B. Cho and J. H. Kim, "Multiple network fusion using fuzzy logic," IEEE Transactions on Neural Networks, vol. 6, no. 2, pp. 497-501, 1995.
- L. I. Kuncheva, "Using measures of similarity and inclusion for multiple classifier fusion by decision templates," Fuzzy Sets and Systems, L. I. Kuncheva, "Using measures of similarity and inclusion for multiple classifier fusion by decision templates," vol. 122, no. 3, pp. 401-407, 2001.
- L. I. Kuncheva, "Switching between selection and fusion in combining classifiers: An ex-periment," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 32, no. 2, pp. 146-156, 2002.
- R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," Annals of Statistics, vol. 26, no. 5, pp. 1651-1686, 1998.
- Y. S. Huang and C. Y. Suen, "Behavior-knowledge space method for combination of mul-tiple classifiers," Proc. of IEEE Computer Vision and Pattern Recog., pp. 347-352, 1993.
- L. Xu, A. Krzyzak, and C.Y. Suen, Methods for combining multiple classifiers and their applications to handwriting recognition, IEEE Trans. on Systems, Man, and Cyb., Vol. 22, No. 3, pp. 418-435, 1992.

- E. Allwein, R. E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," Journal of Machine Learning Research, vol. 1, pp. 113-141, 2000.
- T.K. Ho, "Complexity of classification problems and comparative advantages of combined classifiers," Int.Workshop on Multiple Classifier Systems, lecture Notes on Computer Science, Vol. 1857, pp. 97-106, 2000, Springer- Verlag.
- F. Roli, G. Giacinto, "Design of Multiple Classifier Systems," in H. Bunke and A. Kandel (Eds.) Hybrid Methods in Pattern Recognition, World Scientific Publishing, 2002.
- R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algo-rithm for supervised neural networks," IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, vol. 31, no. 4, pp. 497-508, 2001.
- R. Polikar, “Bootstrap inspired techniques in computational intelligence: ensemble of classifiers, incremental learning, data fusion and missing features, IEEE Signal Processing Magazine, v. 24, no. 4, pp. 59-72, 2007.
- R. Polikar, “Ensemble based systems in decision making,” IEEE Circuits and Systems Magazine, vol. 6, no.3, pp. 21-45, 2006.
- Muhlbaier M., Topalis A., Polikar R., “Ensemble confidence estimates posterior probability,” 6th Int. Workshop on Multiple Classifier Systems (MCS 2005), Springer Lecture Notes in Computer Science (LNCS), vol. 3541, pp. 326-335, Seaside. Monterey, CA, June 2005.
- M. Muhlbaier, A. Topalis, R. Polikar, "Learn++.NC: Combining Ensemble of Classifiers with Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes," IEEE Transactions on Neural Networks, In press, 2008.
- T. G. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes," Journal of Artificial Intel. Research, vol. 2, pp. 263-286, 1995.
- T. K. Ho, "Random subspace method for constructing decision forests," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 832-844, 1998.
- N. C. Oza and K. Tumer, "Input Decimation Ensembles: Decorrelation through Dimensio-nality Reduction," 2nd Int. Workshop on Multiple Classifier Systems, in Lecture Notes in Computer Science, J. Kittler and F. Roli, Eds., vol. 2096, pp. 238-247, 2001.
- G. Brown, J. Wyatt, R. Harris, X. Yao, "Diversity Creation Methods: A Survey and Categorisation," Journal of Information Fusion (Special issue on Diversity in Multiple Classifier Systems). Volume 6, issue 1, pp 5-20, March 2005

Internal references

- Jan A. Sanders (2006) Averaging. Scholarpedia, 1(11):1760.
- Paul L. Nunez and Ramesh Srinivasan (2007) Electroencephalogram. Scholarpedia, 2(2):1348.
- Joan Dawson and Paul C. Lauterbur (2008) Magnetic resonance imaging. Scholarpedia, 3(7):3381.
- Philip Holmes and Eric T. Shea-Brown (2006) Stability. Scholarpedia, 1(10):1838.

Recommended reading : Comprehensive tutorials on ensemble systems

- R. Polikar, “Bootstrap inspired techniques in computational intelligence: ensemble of classifiers, incremental learning, data fusion and missing features, IEEE Signal Processing Magazine, v. 24, no. 4, pp. 59-72, 2007.
- R. Polikar, “Ensemble based systems in decision making,” IEEE Circuits and Systems Magazine, vol. 6, no.3, pp. 21-45, 2006.

External links - People working on ensemble based systems (not a comprehensive list)

- Robi Polikar (http://users.rowan.edu/~polikar/) (current author / curator of this article)

See also

Boosting, Machine learning, Metalearning, Mixtures of experts, Multiple classifier systems, PAC learning, Pattern recognition

Sponsored by: Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia

Reviewed by (http://www.scholarpedia.org/w/index.php?title=Ensemble_learning&oldid=51655) : Anonymous

Accepted on: 2008-12-22 21:49:47 GMT (http://www.scholarpedia.org/w/index.php?title=Ensemble_learning&oldid=55250)

Category: Computational Intelligence