

E

Ensemble Learning

GAVIN BROWN

The University of Manchester,
School of Computer Science,
Kilburn Building, Oxford Road,
Manchester, M13 9PL, UK

Synonyms

Committee machines; Multiple classifier systems

Definition

Ensemble learning refers to the procedures employed to train multiple learning machines and combine their outputs, treating them as a “committee” of decision makers. The principle is that the decision of the committee, with individual predictions combined appropriately, should have better overall ►**accuracy**, on average, than any individual committee member. Numerous empirical and theoretical studies have demonstrated that ensemble ►**models** very often attain higher accuracy than single models.

The members of the ensemble might be predicting real-valued numbers, class labels, posterior probabilities, rankings, clusterings, or any other quantity. Therefore, their decisions can be combined by many methods, including averaging, voting, and probabilistic methods. The majority of ensemble learning methods are generic, applicable across broad classes of model types and learning tasks.

Motivation and Background

If we could build the “perfect” machine learning device, one which would give us the best possible answer every time, there would be no need for *ensemble learning* methods – indeed, there would be no need for this encyclopedia either. The underlying principle of ensemble learning is a recognition that in real-world situations, every model has limitations and will make errors.

Given that each model has these “limitations,” the aim of ensemble learning is to manage their strengths and weaknesses, leading to the best possible decision being taken overall. Several theoretical and empirical results have shown that the accuracy of an ensemble can significantly exceed that of a single model.

The principle of combining predictions has been of interest to several fields over many years. Over 200 years ago, a controversial question had arisen, on how best to estimate the mean of a probability distribution given a small number of sample observations. Laplace (1818) demonstrated that the sample mean was not always optimal: under a simple condition, the sample median was a better combined predictor of the population mean. The financial forecasting community has analyzed model combination for several decades, in the context of stock portfolios. The contribution of the machine learning (ML) community emerged in the 1990s – automatic construction (from data) of both the models and the method to combine them. While the majority of the ML literature on this topic is from 1990 onward, the principle has been explored briefly by several independent authors since the 1960s. See Kuncheva (2004b) for historical accounts.

The study of ensemble methods, with model outputs considered for their abstract properties rather than the specifics of the algorithm which produced them, allows for a wide impact across many fields of study. If we can understand precisely why, when, and how particular ensemble methods can be applied successfully, we would have made progress toward a powerful new tool for Machine Learning: *the ability to automatically exploit the strengths and weaknesses of different learning systems*.

Methods and Algorithms

An ensemble consists of a set of models and a method to combine them. We begin this section by assuming that we have a set of models, generated by any of the

learning algorithms in this encyclopedia; we explore popular methods of combining their outputs, for classification and regression problems. Following this, we review some of the most popular ensemble algorithms, for *learning* a set of models given the knowledge that they will be combined, including extensive pointers for further reading. Finally, we take a theoretical perspective, and review the concept of ensemble *diversity*, the fundamental property which governs how well an ensemble can perform.

Methods for Combining a Set of Models

There exist numerous methods for model combination, far too many to fully detail here. The *linear* combiner, the *product* combiner, and the *voting* combiner are by far the most commonly used in practice. Though a combiner could be specifically chosen to optimize performance in a particular application, these three rules have shown consistently good behavior across many problems, and are simple enough that they are amenable to theoretical analysis.

The linear combiner is used for models that output real-valued numbers, so is applicable for [▶regression](#) ensembles, or for [▶classification](#) ensembles producing class probability estimates. Here, notation for the latter case is only shown. We have a model $f_t(y|\mathbf{x})$, an estimate of the probability of class y given input \mathbf{x} . For a set of these, $t = \{1, \dots, T\}$, the ensemble probability estimate is,

$$\bar{f}(y|\mathbf{x}) = \sum_{t=1}^T w_t f_t(y|\mathbf{x}). \quad (1)$$

If the weights $w_t = 1/T, \forall t$, this is a simple uniform averaging of the probability estimates. The notation clearly allows for the possibility of a nonuniformly weighted average. If the classifiers have different accuracies on the data, a nonuniform combination could *in theory* give a lower error than a uniform combination. However, in practice, the difficulty of estimating the \mathbf{w} parameters without overfitting, and the relatively small gain that is available (see Kuncheva, 2004b, p. 282), have meant that in practice the uniformly weighted average is by far the most commonly used. A notable exception, to be discussed later in this article, is the *mixture of experts* paradigm – in MoE, weights are nonuniform, but are learnt and dependent on the input value \mathbf{x} . An

alternative combiner is the *product rule*:

$$\bar{f}(y|\mathbf{x}) = \frac{1}{Z} \prod_{t=1}^T f_t(y|\mathbf{x})^{w_t}, \quad (2)$$

where Z is a normalization factor to ensure \bar{f} is a valid distribution. Note that Z is not *required* to make a valid decision, as the order of posterior estimates remain unchanged before/after normalization. Under the assumption that the class-conditional probability estimates are independent, this is the theoretically optimal combination strategy. However, this assumption is highly unlikely to hold in practice, and again the weights \mathbf{w} are difficult to reliably determine. Interestingly, the linear and product combiners are in fact special cases of the *generalized mean* (Kuncheva, 2004b) allowing for a continuum of possible combining strategies.

The linear and product combiners are applicable when our models output real-valued numbers. When the models instead output class labels, a majority (or plurality) vote can be used. Here, each classifier votes for a particular class, and the class with the most votes is chosen as the ensemble output. For a two-class problem the models produce labels, $h_t(\mathbf{x}) \in \{-1, +1\}$. In this case, the ensemble output for the *voting* combiner can be written as

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T w_t h_t(\mathbf{x}) \right). \quad (3)$$

The weights \mathbf{w} can be uniform for a simple majority vote, or nonuniform for a weighted vote.

We have discussed only a small fraction of the possible combiner rules. Numerous other rules exist, including methods for combining rankings of classes, and unsupervised methods to combine clustering results. For details of the wider literature, see Kuncheva (2004b) or Polikar (2006).

Algorithms for Learning a Set of Models

If we had a committee of *people* taking decisions, it is self-evident that we would not want them all to make the same bad judgments *at the same time*. With a committee of learning models, the same intuition applies: we will have no gain from combining a set of identical models. We wish the models to exhibit a certain element of “diversity” in their group behavior, though still retaining good performance individually.

We therefore make a distinction between two types of ensemble learning algorithms, those which encourage diversity *implicitly*, and those which encourage it *explicitly*. The vast majority of ensemble methods are *implicit*, in that they provide different *random subsets* of the training data to each learner. Diversity is encouraged “implicitly” by *random* sampling of the data space: at no point is a *measurement* taken to ensure diversity will emerge. The random differences between the datasets might be in the selection of examples (the ►[Bagging](#) algorithm), the selection of features (►[Random Subspaces](#), Ho, 1998 or ►[Rotation Forests](#), Rodriguez, Kuncheva, & Alonso, 2006), or combinations of the two (the Random Forests algorithm, Breiman, 2001). Many other “randomization” schemes are of course possible.

An alternative is to *explicitly* encourage diversity, constructing each ensemble member with some measurement ensuring that it is substantially different from the other members. ►[Boosting](#) algorithms achieve this by altering the distribution of training examples for each learner such that it is encouraged to make more accurate predictions where previous predictors have made errors. The DECORATE algorithm (Melville & Mooney, 2005) explicitly alters the distribution of class labels, such that successive models are forced to learn different answers to the same problem. ►[Negative correlation learning](#) (see Brown, 2004; Brown, Wyatt, Harris, & Yao, 2005), includes a penalty term when learning each ensemble member, explicitly *managing* the accuracy-diversity trade-off.

In general, ensemble methods constitute a large class of algorithms – some based on heuristics, and some on sound learning-theoretic principles. The three algorithms that have received the most attention in the literature are reviewed here. It should be noted that we present only the most basic form of each; numerous modifications have been proposed for a variety of learning scenarios. As further study the reader is referred to the many comprehensive surveys of the field (Brown et al., 2005; Kuncheva, 2004b; Polikar, 2006).

Bagging

In the Bagging algorithm (Breiman, 1996), each member of the ensemble is constructed from a different training dataset, and the predictions combined either by uniform averaging or voting over class labels. Each

dataset is generated by sampling from the total N data examples, choosing N items uniformly at random *with replacement*. Each sample is known as a *bootstrap*; the name Bagging is an acronym derived from Bootstrap AGGREGatING. Since a bootstrap samples N items uniformly at random with replacement, the probability of any individual data item *not* being selected is $p = (1 - 1/N)^N$. Therefore with large N , a single bootstrap is expected to contain approximately 63.2% of the original set, while 36.8% of the originals are not selected.

Like many ensemble methods, Bagging works best with *unstable* models, that is those that produce differing generalization behavior with small changes to the training data. These are also known as *high variance* models, examples of which are ►[decision trees](#) and ►[neural networks](#). Bagging therefore tends not to work well with very simple models. In effect, Bagging samples randomly from the space of possible models to make up the ensemble – with very simple models the sampling produces almost identical (low diversity) predictions.

Despite its apparent capability for variance reduction, situations have been demonstrated where Bagging can converge *without* affecting variance (see Brown et al., 2005). Several other explanations have been proposed for Bagging’s success, including links to ►[Bayesian model averaging](#). In summary, it seems that several years from its introduction, despite its apparent simplicity, Bagging is still not fully understood.

Algorithm 1 Bagging

Input: Required ensemble size T

Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

for $t = 1$ to T **do**

Build a dataset S_t , by sampling N items, randomly *with replacement* from S .

Train a model h_t using S_t , and add it to the ensemble.

end for

For a new testing point (x', y') ,

If model outputs are continuous, combine them by averaging.

If model outputs are class labels, combine them by voting.

Adaboost

Adaboost (Freund & Schapire, 1996) is the most well known of the *Boosting* family of algorithms (Schapire, 2003). The algorithm trains models sequentially, with a new model trained at each round. At the end of each round, mis-classified examples are identified and have their emphasis increased in a new training set which is then fed back into the start of the next round, and a new model is trained. The idea is that subsequent models should be able to compensate for errors made by earlier models.

Adaboost occupies somewhat of a special place in the history of ensemble methods. Though the procedure seems heuristic, the algorithm is in fact grounded in a rich learning-theoretic body of literature. Schapire (1990) addressed a question posed by Kearns and Valiant (1988) on the nature of two complexity classes of learning problems. The two classes are *strongly learnable* and *weakly learnable* problems. Schapire showed that these classes were equivalent; this had the corollary that a weak model, performing only slightly better than random guessing, could be “boosted” into an arbitrarily accurate *strong* model. The original Boosting algorithm was a proof by construction of this equivalence, though had a number of impractical assumptions built-in. The Adaboost algorithm (Freund & Schapire, 1996) was the first practical Boosting method. The authoritative historical account of the development can be found in Schapire (1999), including discussion of numerous variants and interpretations of the algorithm. The procedure is shown in Algorithm 2. Some similarities with Bagging are evident; a key differences is that at each round t , Bagging has a uniform distribution D_t , while Adaboost adapts a nonuniform distribution.

The ensemble is constructed by iteratively adding models. Each time a model is learnt, it is checked to ensure it has at least $\epsilon_t < 0.5$, that is, it has performance *better than random guessing* on the data it was supplied with. If it does not, either an alternative model is constructed, or the loop is terminated.

After each round, the distribution D_t is updated to emphasize incorrectly classified examples. The update causes half the distribution mass of D_{t+1} to be over the examples incorrectly classified by the previous model. More precisely, $\sum_{h_t(x_i) \neq y_i} D_{t+1}(i) = 0.5$. Thus, if h_t has an error rate of 10%, then examples from that small 10% will

Algorithm 2 Adaboost

Input: Required ensemble size T

Input: Training set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, where $y_i \in \{-1, +1\}$

Define a uniform distribution $D_1(i)$ over elements of S .

for $t = 1$ to T **do**

Train a model h_t using distribution D_t .

Calculate $\epsilon_t = P_{D_t}(h_t(x) \neq y)$

If $\epsilon_t \geq 0.5$ **break**

Set $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

Update $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

where Z_t is a normalization factor so that D_{t+1} is a valid distribution.

end for

For a new testing point (x', y') ,

$H(x') = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x'))$

be allocated 50% of the next model’s training “effort,” while the remaining examples (those correctly classified) are underemphasized. An equivalent (and simpler) writing of the distribution update scheme is to multiply $D_t(i)$ by $1/2(1 - \epsilon_t)$ if $h_t(x_i)$ is correct, and by $1/2\epsilon_t$ otherwise.

The updates cause the models to sequentially minimize an exponential bound on the error rate. The training error rate on a data sample S drawn from the true distribution D obeys the bound,

$$P_{\mathbf{x}, y \sim S}(yH(\mathbf{x}) < 0) \leq \prod_{t=1}^T 2\sqrt{\epsilon_t(1 - \epsilon_t)}. \quad (4)$$

This *upper bound* on the training error (though not the *actual* training error) is guaranteed to decrease monotonically with T , given $\epsilon_t < 0.5$.

In an attempt to further explain the performance of Boosting algorithms, Schapire also developed bounds on the *generalization* error of voting systems, in terms of the voting margin, the definition of which was given in (10). Note that, this is not the same as the *geometric margin*, optimized by [support vector machines](#). The difference is that the voting margin is defined using the one-norm $\|\mathbf{w}\|_1$ in the denominator, while the geometric margin uses the *two-norm* $\|\mathbf{w}\|_2$. While this is a subtle difference, it is an important one, forming links between SVMs and Boosting algorithms – see Rätsch,

Mika, Schölkopf, and Müller (2002) for details. The following bound holds with probability $1 - \delta$,

$$P_{\mathbf{x}, y \sim \mathcal{D}}(H(\mathbf{x}) \neq y) \leq P_{\mathbf{x}, y \sim \mathcal{S}}(yH(\mathbf{x}) < \theta) + \tilde{O}\left(\sqrt{\frac{d}{N\theta^2}} - \ln \delta\right), \quad (5)$$

where the \tilde{O} notation hides constants and logarithmic terms, and d is the [VC-dimension](#) of the model used. Roughly, this states that the generalization error is less than or equal to the training error plus a term dependent on the voting margin. The larger the minimum margin in the training data, the lower the testing error. The original bounds have since been significantly improved, see Koltchinskii and Panchenko (2005) as a comprehensive recent work. We note that this bound holds generally for *any* voting system, and is not specific to the Boosting framework.

The margin-based theory is only one explanation of the success of Boosting algorithms. Mease and Wyner (2008) present a discussion of several questions on why and how Adaboost succeeds. The subsequent 70 pages of discussion demonstrate that the story is by no means simple. The conclusion is, while no single theory can fully explain Boosting, each provides a different part of the still unfolding story.

Mixtures of Experts

The mixtures of experts architecture is a widely investigated paradigm for creating a combination of models (Jacobs, Jordan, Nowlan, & Hinton, 1991). The principle underlying the architecture is that certain models will be able to “specialize” to particular parts of the input space. It is commonly implemented with a neural network as the base model, or some other model capable of estimating probabilities. A *Gating network* receives the same inputs as the component models, but its outputs are used as the weights for a linear combiner. The Gating network is responsible for learning the appropriate weighted combination of the specialized models (“experts”) for any given input. Thus, the input space is “carved-up” between the experts, increasing and decreasing their weights for particular examples. In effect, a mixture of experts explicitly learns how to create expert ensemble members in different portions of the input space, and select the most appropriate subset for a new testing example (Fig. 1).

The architecture has received wide attention, and has a strong following in the probabilistic modeling community, where it may go under the pseudonym of a “mixture model.” A common training method is the [expectation-maximization algorithm](#).

Theoretical Perspectives: Ensemble Diversity

We have seen that all ensemble algorithms in some way attempt to encourage “diversity.” In this section, we take a more formalized perspective, to understand what is meant by this term.

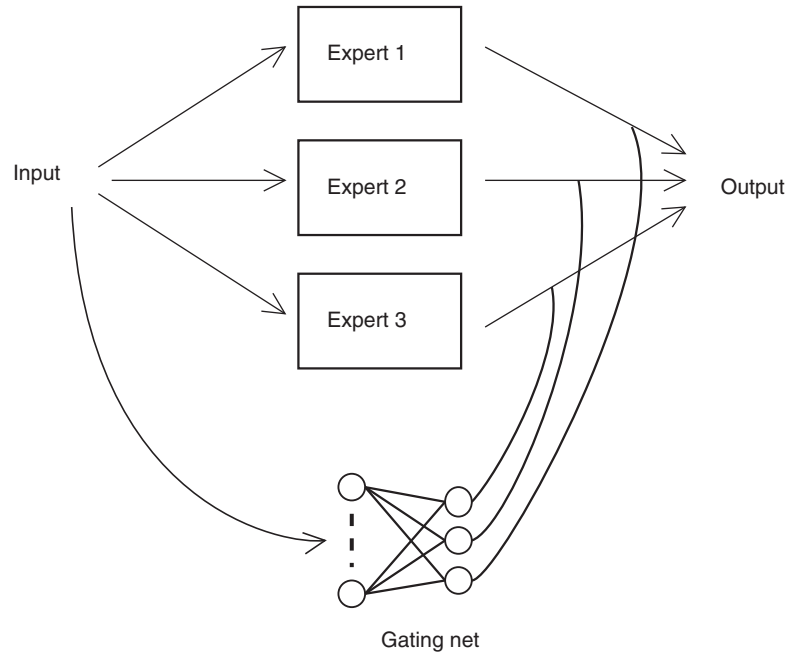
What is Diversity?

The optimal “diversity” is fundamentally a *credit assignment* problem. If the committee as a whole makes an erroneous prediction, how much of this error should be attributed to each member? More precisely, how much of the committee prediction is due to the accuracies of the individual models, and how much is due to their interactions when they were combined? We would ideally like to reexpress the ensemble error as two distinct components: a term for the accuracies of the individual models, plus a term for their interactions, i.e., their *diversity*.

It turns out that this so-called *accuracy-diversity* breakdown of the ensemble error is not always possible, depending on the type of error function, and choice of combiner rule. It should be noted that when “diversity” is referred to in the literature, it is most often meant to indicate classification with a majority vote combiner, but for completeness we address the general case here. In the following sections, the existing work to understand diversity in three distinct cases is described: for regression tasks (a linear combiner), and classification tasks, with either a linear combiner or a voting combiner.

Regression Error with a Linear Combination Rule

In a regression problem, it is common to use the squared error criterion. The accuracy-diversity breakdown for this case (using a linear combiner) is called the *ambiguity decomposition* (Krogh & Vedelsby, 1995). The result states that the squared error of the linearly combined



Ensemble Learning. Figure 1. The mixtures of experts architecture

ensemble, $\tilde{f}(\mathbf{x})$, can be broken into a sum of two components:

$$(\tilde{f}(\mathbf{x}) - d)^2 = \frac{1}{T} \sum_{t=1}^T (f_t(\mathbf{x}) - d)^2 - \frac{1}{T} \sum_{t=1}^T (f_t(\mathbf{x}) - \tilde{f}(\mathbf{x}))^2. \quad (6)$$

The first term on the right hand side is the average squared error of the individual models, while the second term quantifies the interactions *between* the predictions. Note that this second term, the “ambiguity,” is always positive. This guarantees that, for an arbitrary data point, the ensemble squared error is always less than or equal to the average of the individual squared errors.

The intuition here can be understood as follows. Imagine five friends, playing “guess the weight of the cake” (an old English fairground game): if a player’s guess is close enough to the true weight, they win the cake. Just as they are about to play, the fairground manager states that they can only submit *one* guess. The dilemma seems to be in whose guess they should submit – however, the ambiguity decomposition shows us that taking the average of their guesses, and submitting that, will *always* be closer (on average) than choosing a person at random and submitting their guess. Note that this is qualified with “on average” – it may well be that one

of the predictions will in fact be closer than the average prediction, but we presume that we have no way of identifying *which* prediction to choose, other than random. It can be seen that greater diversity in the predictions (i.e., a larger ambiguity term) results in a larger gain over the average individual performance. However, it is also clear that there is a trade-off to be had: too much diversity and the average error is extremely large.

The idea of a trade-off between these two terms is reminiscent of the [►bias-variance](#) decomposition (Geman, Bienenstock, & Doursat, 1992); in fact, there is a deep connection between these results. Taking the expected value of (6) over all possible training sets gives us the ensemble analogy to the bias-variance decomposition, called the [►bias-variance-covariance](#) decomposition (Ueda & Nakano, 1996). This shows that the expected squared error of an ensemble $\tilde{f}(\mathbf{x})$ from a target d is:

$$\mathcal{E}_{\mathcal{D}}\{(\tilde{f}(\mathbf{x}) - d)^2\} = \overline{\text{bias}}^2 + \frac{1}{T} \overline{\text{var}} + \left(1 - \frac{1}{T}\right) \overline{\text{covar}}, \quad (7)$$

where the expectation is with respect to all possible training datasets \mathcal{D} . While the bias and variance terms are constrained to be positive, the covariance between models can become negative – thus the definition of

diversity emerges as an extra degree of freedom in the bias-variance dilemma. This extra degree of freedom allows an ensemble to approximate functions that are difficult (if not impossible) to find with a single model. See Brown et al. (2005) for extensive further discussion of this concept.

Classification Error with a Linear Combination Rule

In a classification problem, our error criterion is the misclassification rate, also known as the *zero-one* loss function. For this type of loss, it is well known there is no unique definition of bias-variance; instead there exist multiple decompositions each with advantages and disadvantages (see Kuncheva, 2004b, p. 224). This gives us a clue as to the situation with an ensemble – there is also no simple accuracy-diversity separation of the ensemble classification error. Classification problems can of course be addressed either by a model producing class probabilities (where we linearly combine), or directly producing class labels (where we use majority vote). Partial theory has been developed for each case.

For linear combiners, there exist theoretical results that relate the correlation of the probability estimates to the ensemble classification error. Tumer and Ghosh (1996) showed that the reducible classification error (i.e., above the Bayes rate) of a simple averaging ensemble, e_{ave} , can be written as

$$e_{ave} = e_{add} \left(\frac{1 + \delta(T-1)}{T} \right), \quad (8)$$

where e_{add} is the classification error of an individual model. The δ is a correlation coefficient between the model outputs. When the individual models are identical, the correlation is $\delta = 1$. In this case, the ensemble error is equal to the individual error, $e_{ave} = e_{add}$. When the models are statistically independent, $\delta = 0$, and the ensemble error is a fraction $1/T$ of the individual error, $e_{ave} = 1/T \times e_{add}$. When δ is negative, the models are negatively correlated, and the ensemble error is lower than the average individual error. However, (8) is derived under quite strict assumptions, holding only for a local area around the decision boundary, and ultimately resting on the bias-variance-covariance theory from regression problems. Further details, including recent work to lift some of the assumptions (Kuncheva, 2004b).

Classification Error with a Voting Combination Rule

The case of a classification problem with a majority vote combiner is the most challenging of all. In general, there is no known breakdown of the ensemble classification error into neat accuracy and diversity components. The simplest intuition to show that correlation between models does affect performance is given by the Binomial theorem. If we have T models each with identical error probability $p = P(h_i(\mathbf{x}) \neq y)$, assuming they make statistically *independent* errors, the following error probability of the majority voting committee holds,

$$P(H(\mathbf{x}) \neq y) = \sum_{k > (T/2)}^T \binom{T}{k} p^k (1-p)^{(T-k)}. \quad (9)$$

For example, in the case of $T = 21$ ensemble members, each with error $p = 0.3$, the majority voting error will be 0.026, an order of magnitude improvement over the individual error. However, this *only* holds for statistically independent errors. The correlated case is an open problem. Instead, various authors have proposed their own heuristic definitions of diversity in majority voting ensembles. Kuncheva (2004b) conducted extensive studies of several suggested diversity measures; the conclusion was that “*no measure consistently correlates well with the majority vote accuracy.*” In spite of this, some were found useful as an approximate guide to characterize performance of ensemble methods, though should not be relied upon as the “final word” on diversity. Kuncheva’s recommendation in this case is the *Q-statistic* (Kuncheva, 2004b, p. 299), due to its simplicity and ease of computation.

Breiman (2001) took an alternative approach, deriving not a *separation* of error components, but a *bound* on the generalization error of a voting ensemble, expressed in terms of the correlations of the models. To understand this, we must introduce concept of *voting margin*. The voting margin for a two-class problem, with $y \in \{-1, +1\}$, is defined,

$$m = \frac{y_t \sum_{t=1}^T w_t h_t(\mathbf{x})}{\sum_{t=1}^T |w_t|} = yH(\mathbf{x}). \quad (10)$$

If the margin is positive, the example is correctly classified, if it is negative, the example is incorrectly classified. The expected margin $s = \mathcal{E}_{\mathcal{D}}\{m\}$ measures the extent to

which the average number of votes for the correct class exceeds the average vote for any other class, with respect to the data distribution \mathcal{D} . The larger the voting margin, the more confidence in the classification. Breiman's bound shows,

$$P_{\mathcal{D}}(H(\mathbf{x}) \neq y) = P_{\mathcal{D}}(yH(\mathbf{x}) < 0) \leq \frac{\bar{\rho}(1-s^2)}{s^2}. \quad (11)$$

Here $\bar{\rho}$ is the average pairwise correlation between the errors of the individual models. Thus, the generalization error is minimized by a small $\bar{\rho}$, and an s as close to 1 as possible. The balance between a high accuracy (large s) and a high diversity (low $\bar{\rho}$) constitutes the tradeoff in this case, although the bound is quite loose.

Summary

In summary, the definition of diversity depends on the problem. In a regression problem, the optimal diversity is the trade-off between the bias, variance and covariance components of the squared error. In a classification problem, with a linear combiner, there exists partial theory to relate the classifier correlations to the ensemble error rate. In a classification problem with a voting combiner, there is no single theoretical framework or definition of diversity. However, the lack of an agreed definition of diversity has not discouraged researchers from trying to achieve it, nor has it stalled the progress of effective algorithms in the field.

Conclusions & Current Directions in the Field

Ensemble methods constitute some of the most robust and accurate learning algorithms of the past decade (Caruana & Niculescu-Mizil, 2006). A multitude of heuristics have been developed for randomizing the ensemble parameters, to generate diverse models. It is arguable that this line of investigation is nowadays rather oversubscribed, and the more interesting research is now in methods for nonstandard data. ► **Cluster ensembles** (Strehl & Ghosh, 2003) are ensemble techniques applied to unsupervised learning problems. Problems with *nonstationary* data, also known as *concept drift*, are receiving much recent attention (Kuncheva, 2004a). The most up to date innovations are to be found in the biennial *International Workshop on Multiple Classifier Systems* (Roli et al., 2000).

Recommended Reading

Kuncheva (2004b) is the standard reference in the field, which includes references to many further recommended readings. In addition, Brown et al. (2005) and Polikar (2006) provide extensive literature surveys. Roli et al. (2000) is an international workshop series dedicated to ensemble learning.

References

- Breiman, L. (1996). Bagging predictors. *Machine Learning* 24(2), 123–140.
- Breiman, L. (2001). Random forests. *Machine Learning* 45(1), 5–32.
- Brown, G. (2004). *Diversity in neural network ensembles*. PhD thesis, University of Birmingham.
- Brown, G., Wyatt, J.L., Harris, R., & Yao, X. (2005). Diversity creation methods: A survey and categorisation. *Journal of Information Fusion* 6(1), 5–20.
- Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168). New York: ACM.
- Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the thirteenth international conference on machine learning (ICML'96)* (pp. 148–156). San Francisco: Morgan Kaufmann Publishers.
- Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation* 4(1), 1–58.
- Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8), 832–844.
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation* 3(1), 79–87.
- Kearns, M., & Valiant, L. G. (1988). *Learning Boolean formulae or finite automata is as hard as factoring*. Technical report TR-14-88, Harvard University Aiken Computation Laboratory.
- Koltchinskii, V., & Panchenko, D. (2005). Complexities of convex combinations and bounding the generalization error in classification. *Annals of Statistics* 33(4), 1455.
- Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross-validation and active learning. In *Advances in neural information processing systems* (pp. 231–238). Cambridge, MA: MIT Press.
- Kuncheva, L. I. (2004a). Classifier ensembles for changing environments. In *International workshop on multiple classifier systems*. Lecture Notes in Computer Science 3007. Berlin: Springer.
- Kuncheva, L. I. (2004b). *Combining pattern classifiers: Methods and algorithms*. New York: Wiley.
- Laplace, P. S. (1818). *Deuxieme supplement a la theorie analytique des probabilites*. Paris: Gauthier-Villars.
- Mease, D., & Wyner, A. (2008). Evidence contrary to the statistical view of Boosting. *Journal of Machine Learning Research* 9, 131–156.
- Melville, P., & Mooney, R. J. (2005). Creating diversity in ensembles using artificial data. *Information Fusion* 6(1), 99–111.
- Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3), 21–45.

- Rätsch, G., Mika, S., Schölkopf, B., & Müller, K. R. (2002). Constructing Boosting algorithms from SVMs: An application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(9), 1184–1199.
- Rodriguez, J., Kuncheva, L., & Alonso, C. (2006). Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(10), 1619–1630.
- Roli, F., Kittler, J., Windridge, D., Oza, N., Polikar, R., Haindl, M., et al. (Eds.). *Proceedings of the international workshop on multiple classifier systems 2000–2009. Lecture notes in computer science*. Berlin: Springer. Available at: <http://www.informatik.uni-trier.de/ley/db/conf/mcs/index.html>
- Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning* 5, 197–227.
- Schapire, R. E. (1999). A brief introduction to Boosting. In *Proceedings of the 16th international joint conference on artificial intelligence* (pp. 1401–1406). San Francisco, CA: Morgan Kaufmann.
- Schapire, R. E. (2003). *The boosting approach to machine learning: An overview*. In D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick, & B. Yu (Eds.), *Nonlinear estimation & classification Lecture notes in statistics* (pp. 149–172). Berlin: Springer.
- Strehl, A., & Ghosh, J. (2003). Cluster ensembles – A knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research* 3, 583–617.
- Tumer, K., & Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science* 8(3–4), 385–403.
- Ueda, N., & Nakano, R. (1996). Generalization error of ensemble estimators. In *Proceedings of IEEE international conference on neural networks* (Vol. 1, pp. 90–95). ISBN: 0-7803-3210-5