Ensemble Learning

Gavin Brown
School of Computer Science, The University of Manchester, United Kingdom

Synonyms

Multiple classifier systems, committee machines.

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 committee decision, 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 will have 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. Pierre-Simon de Laplace [6] 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 [15] 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 will 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 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 we only show notation for the latter case. 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, $t = \{1...T\}$, the ensemble probability estimate is,

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

If the weights $w_t = \frac{1}{T}, \forall t$, this is a simple uniform averaging of the probability estimates. The notation clearly allows for the possibility of a non-uniformly weighted average. If the classifiers have different accuracies on the data, a non-uniform 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 [15, p282]), 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 non-uniform, but are learnt and dependent on the input value $\mathbf{x}$. An alternative combiner is the *product rule*:

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

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 will 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* [15] 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

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

The weights $\mathbf{w}$ can be uniform for a simple majority vote, or non-uniform 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 references [15] or [18].

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 judgements *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 [9] or Rotation Forests [20]), or combinations of the two (the Random Forests [2] algorithm). Many other 'randomization' schemes are of course possible.

An alternative is to *explicitly* encourage diversity, constructing each ensemble member with some measurement ensuring 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 [17] 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 [3, 4]), 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, some based on sound learning-theoretic principles. Here we review three algorithms that have received the most attention in the literature. 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 [4, 15, 18].

Bagging

In the Bagging algorithm [1], 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 **B**ootstrap **AGG**regat**ING**. Since a bootstrap samples N items uniformly at random with replacement, the probability of any individual data item *not* being selected is $p = (1 - \frac{1}{N})^N$. 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 [4]). 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 $S = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$
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 [7] is the most well known of the *Boosting* family of algorithms [24]. 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 [22] addressed a question posed by Kearns and Valiant [11] 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 [7] was the first practical Boosting method. The authoritative historical account of the development can be found in reference [23], 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 non-uniform distribution.

Algorithm 2 Adaboost

Input: Required ensemble size T
Input: Training set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, 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'),
$H(x') = sign\Big(\sum_{t=1}^{T} \alpha_t h_t(x')\Big)$

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, $\sum_{h_t(\mathbf{x}_i) \neq y_i} D_{t+1}(i) = 0.5$. Thus, if h_t has an error rate of 10%, then examples from that small 10% will 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 $\frac{1}{2(1-\epsilon_t)}$ if $h_t(x_i)$ is correct, and by $\frac{1}{2\epsilon_t}$ otherwise.

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

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

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 eq (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 et al. [19] for details. The following bound holds with probability $1 - \delta$,

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

where the $\tilde{O}$ notation hides constants and logarithmic terms, and d is the VC-dimension of the model used. Roughly speaking, 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 [12] 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 & Wyner [16] 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 [10]. 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. In this way 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.

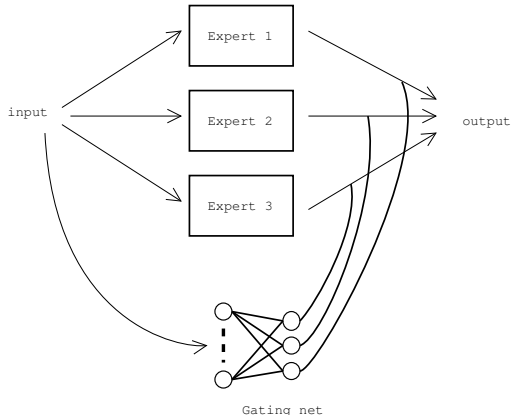

Figure 1: The Mixtures of Experts architecture

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 re-express 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 we will describe the existing work to understand diversity in three distinct cases: 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* [13]. The result states that the squared error of the linearly combined ensemble, $\bar{f}(\mathbf{x})$, can be broken into a sum of two components:

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

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 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) will result 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 will be extremely large.

The idea of a trade-off between these two terms is reminiscent of the Bias-Variance decomposition [8]; in fact, there is a deep connection between these results. Taking the expected value of eq(6)

over all possible training sets gives us the ensemble analogy to the bias-variance decomposition, called the Bias-Variance-Covariance decomposition [27]. This shows that the expected squared error of an ensemble $\bar{f}(\mathbf{x})$ from a target d is:

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

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 reference [4] 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 [15, pg224]). 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 & Ghosh [26] showed that the reducible classification error (i.e. above the Bayes rate) of a simple averaging ensemble, e_{ave}, can be written as

$$e_{ave} = e_{add}\Big(\frac{1 + \delta(T-1)}{T}\Big) \tag{8}$$

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 $\frac{1}{T}$ of the individual error, $e_{ave} = \frac{1}{T}e_{add}$. When δ is negative, the models are negatively correlated, and the ensemble error is lower than the average individual error. However eq(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, can be found in reference [15].

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 $p = P(h_t(\mathbf{x}) \neq y)$, assuming they make statistically *independent* errors, the following error probability

of the majority voting committee holds,

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

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 [15] 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* [15, pg299], due to its simplicity and ease of computation.

Breiman [2] 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,

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

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,

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

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 [5]. 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 non-standard data. Cluster Ensembles [25] are ensemble techniques applied to unsupervised learning problems. Problems with *non-stationary* data, also known as *concept drift*, are receiving much recent attention [14]. The most up to date innovations are to be found in the biennial *International Workshop on Multiple Classifier Systems* [21].

Recommended Reading

Reference [15] is the standard reference in the field, which includes references to many further recommended readings. In addition, references [4, 18] provide extensive literature surveys. Reference [21] is an international workshop series dedicated to ensemble learning.

References

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

[2] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[3] G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, University of Birmingham, UK, 2004.

[4] G. Brown, J. L. Wyatt, R. Harris, and X. Yao. Diversity Creation Methods: A Survey and Categorisation. *Journal of Information Fusion*, 6(1):5–20, March 2005.

[5] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM New York, NY, USA, 2006.

[6] P. S. de Laplace. Deuxieme supplement a la theorie analytique des probabilites. *Paris, Gauthier-Villars*, 1818.

[7] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 1996.

[8] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[9] T. K. Ho. The Random Subspace Method for constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.

[11] M. Kearns and L. G. Valiant. Learning Boolean Formulae or Finite Automata is as hard as Factoring. Technical Report TR-14-88, Harvard University Aiken Computation Laboratory, August 1988.

[12] V. Koltchinskii and D. Panchenko. Complexities of convex combinations and bounding the generalization error in classification. *Annals of Statistics*, 33(4):1455, 2005.

[13] A. Krogh and J. Vedelsby. Neural Network Ensembles, Cross-Validation and Active Learning. In *Advances in Neural Information Processing Systems (NIPS 7)*, pages 231–238, 1995.

[14] L. I. Kuncheva. Classifier ensembles for changing environments. In *International Workshop on Multiple Classifier Systems, LNCS 3007*. Springer, 2004.

[15] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley Press, 2004.

[16] D. Mease and A. Wyner. Evidence Contrary to the Statistical View of Boosting. *Journal of Machine Learning Research*, 9:131–156, 2008.

[17] P. Melville and R. J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6(1):99–111, 2005.

[18] R. Polikar. Ensemble based systems in Decision Making. *Circuits and Systems Magazine, IEEE*, 6(3):21–45, 2006.

[19] G. Rätsch, S. Mika, B. Schölkopf, and K. R. Müller. Constructing Boosting Algorithms from SVMs: An Application to One-Class Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1184–1199, 2002.

[20] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. Rotation Forest: a new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, pages 1619–1630, 2006.

[21] F. Roli, J. Kittler, D. Windridge, N. Oza, R. Polikar, M. Haindl, and J.A. Benediktsson, editors. *Proceedings of the International Workshop on Multiple Classifier Systems 2000-2009*, Lecture Notes in Computer Science. Springer.

[22] R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5:197–227, 1990.

[23] R. E. Schapire. A Brief Introduction to Boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1999.

[24] R. E. Schapire. The Boosting Approach to Machine Learning: An Overview. *Lecture Notes in Statistics*, pages 149–172, 2003.

[25] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

[26] K. Tumer and J. Ghosh. Error Correlation and Error Reduction in Ensemble Classifiers. *Connection Science*, 8(3-4):385–403, 1996.

[27] N. Ueda and R. Nakano. Generalization Error of Ensemble Estimators. In *Proceedings of International Conference on Neural Networks*, pages 90–95, 1996.

Title: Adaboost

Synonyms: None

Definition

Adaboost is an Ensemble Learning technique, and the most well-known of the Boosting family of algorithms. 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. See Ensemble Learning for full details.

References

[1] R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5:197–227, 1990.

[2] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 1996.

[3] R. E. Schapire. The Boosting Approach to Machine Learning: An Overview. *Lecture Notes in Statistics*, pages 149–172, 2003.

Title: Bagging

Synonyms: None

Definition

Bagging is an Ensemble Learning technique. The name 'Bagging' is an acronym derived from **B**ootstrap **AGG**regat**ING**. Each member of the ensemble is constructed from a different training dataset. Each dataset is a bootstrap sample from the original. The models are combined by a uniform average or vote. Bagging works best with Unstable learners, that is those that produce differing generalization patterns with small changes to the training data. Bagging therefore tends not to work well with linear models. See Ensemble Learning for more details.

References

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

Title: Bias-Variance decomposition

Definition

The bias-variance decomposition is a useful theoretical tool to understand the performance characteristics of a learning algorithm. The following discussion is restricted to the use of *squared loss* as the performance measure, although similar analyses have been undertaken for other loss functions. The case receiving most attention is the zero-one loss (i.e. classification problems), in which case the decomposition is non-unique and a topic of active research. See reference [1] for details.

The decomposition allows us to see that the mean squared error of a model (generated by a particular learning algorithm) is in fact made up of two components. The *bias* component tells us how accurate the model is, on average across different possible training sets. The *variance* component tells us how sensitive the learning algorithm is to small changes in the training set.

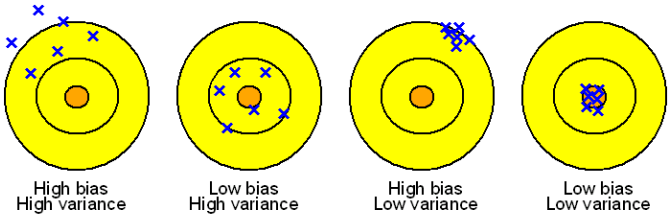

Figure 2: *The bias-variance decomposition is like trying to hit the bullseye on a dartboard. Each dart is thrown after training our 'dart-throwing' model in a slightly different manner. If the darts vary wildly, the learner is* high variance*. If they are far from the bullseye, the learner is* high bias*. The ideal is clearly to have both low bias and low variance; however this is often difficult, giving an alternative terminology as the bias-variance 'dilemma'.* (Dartboard analogy from reference [3].)

Mathematically, this can be quantified as a decomposition of the mean squared error function. For a testing example $\{\mathbf{x}, d\}$, the decomposition is:

$$\begin{aligned} \mathcal{E}_{\mathcal{D}}\{(f(\mathbf{x}) - d)^2\} &= (\mathcal{E}_{\mathcal{D}}\{f(\mathbf{x})\} - d)^2 + \mathcal{E}_{\mathcal{D}}\{(f(\mathbf{x}) - \mathcal{E}_{\mathcal{D}}\{f(\mathbf{x})\})^2\} \\ MSE &= \text{bias}^2 + \text{variance} \end{aligned}$$

where the expectations are with respect to all possible training sets. In practice this can be estimated by cross-validation over a single finite training set, enabling a deeper understanding of the algorithm characteristics. For example, efforts to reduce variance often cause increases in bias, and vice-versa. A large bias and low variance is an indicator that a learning algorithm is prone to overfitting the model.

References

[1] P. Domingos A Unified Bias-Variance Decomposition for Zero-One and Squared Loss, In *Proceedings of National Conference on Artificial Intelligence*, AAAI Press, 1992

[2] S. Geman. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1), 1992.

[3] Moore, D. S. and McCabe, G. P. Introduction to the Practice of Statistics. Published by *Michelle Julet*, 2002.

Title: Bias-Variance-Covariance decomposition

Definition

The Bias-Variance-Covariance decomposition is a theoretical result underlying Ensemble Learning algorithms. It is an extension of the Bias-Variance decomposition, for linear combinations of models. The expected squared error of the ensemble $\bar{f}(\mathbf{x})$ from a target d is:

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

The error is composed of the average bias of the models, plus a term involving their average variance, and a final term involving their average *pairwise co-variance*. This shows that while a single model has a two-way bias-variance tradeoff, an ensemble is controlled by a three-way tradeoff. This ensemble tradeoff is often referred to as the *accuracy-diversity* dilemma for an ensemble. See Ensemble Learning for more details.

References

[1] N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of International Conference on Neural Networks*, pages 90–95, 1996.

Title: Boosting

Synonyms: None

Definition

Boosting is a family of Ensemble Learning methods. The Boosting framework is an answer to a question posed on whether two complexity classes of learning problems are equivalent: *strongly learnable*, and *weakly learnable*. The Boosting framework is a proof by construction that the answer is positive, they are equivalent. The framework allows a "weak" model, only slightly better than random guessing, to be *boosted* into an arbitrarily accurate *strong* model. Adaboost is the most well known and successful of the Boosting family, though there exist many variants specialized for particular tasks, such as cost-sensitive and noise-tolerant versions. See Ensemble Learning for full details.

Title: Cluster Ensembles

Synonyms: None

Definition

Cluster Ensembles are an unsupervised Ensemble Learning method. The principle is to create multiple different clusterings of a dataset, possibly using different algorithms, then aggregate the opinions of the different clusterings into an ensemble result. The final ensemble clustering should be in theory more reliable than the individual clusterings.

References

[1] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

Title: Error Correcting Output Codes

Synonyms: ECOC

Definition

Error Correcting Output Codes are an Ensemble Learning technique. It is applied to a problem with multiple classes, decomposing it into several binary problems. Each class is first encoded as a binary string of length T, assuming we have T models in the ensemble. Each model then tries to separate a subset of the original classes from all the others. For example, one model might learn to distinguish 'class A' from 'not class A'. After the predictions, with T models we have a binary string of length T. The class encoding that is closest to this binary string (using Hamming distance) is the final decision of the ensemble.

References

[1] E.B. Kong, and T.G. Dietterich, Error-correcting output coding corrects bias and variance. *International Conference on Machine Learning, 1995.*

Title: Negative Correlation Learning

Synonyms: NC-learning, NCL

Definition

Negative Correlation Learning [1] is an Ensemble Learning technique. It can be used for regression or classification problems, though with classification problems the models must be capable of producing posterior probabilities. The model outputs are combined with a uniformly weighted average. The squared error is augmented with a penalty term which takes into account the diversity of the ensemble. The error for the ith model is,

$$E(f_i(x)) = \frac{1}{2}\Big(f_i(x) - d\Big)^2 - \lambda\Big(f_i(x) - \bar{f}(x)\Big)^2. \tag{12}$$

The coefficient λ determines the balance between optimizing individual accuracy, and optimizing ensemble diversity. With $\lambda = 0$, the models are trained independently, with no emphasis on diversity. With $\lambda = 1$, the models are tightly coupled, and the ensemble is trained as a single unit. Theoretical studies [2] have shown that NC works by directly optimizing the Bias-Variance-Covariance trade-off, thus it explicitly *manages* the ensemble diversity. When the complexity of the individuals is sufficient to have high individual accuracy, NC provides little benefit. When the complexity is low, NC with a well-chosen λ can provide significant performance improvements. Thus the best situation to make use of the NC framework is with a large number of low accuracy models.

References

[1] Y. Liu and X. Yao, Ensemble learning via negative correlation Neural Networks, vol 12(10), pp 1399-1404 (1999)

[2] G. Brown, J.L. Wyatt, and P. Tino Managing Diversity in Regression Ensembles. *Journal of Machine Learning Research*, Volume 6, pp 1621-1650 (2006).

Title: Random Forests

Synonyms: Random Decision Forests

Definition

Random Forests is an Ensemble Learning technique. It is a hybrid of the Bagging algorithm and the Random Subspace Method, and uses Decision Trees as the base classifier. Each tree is constructed from a bootstrap sample from the original dataset. An important point is that the trees are not subjected to pruning after construction, enabling them to be partially overfitted to their own sample of the data. To further diversify the classifiers, at each branch in the tree, the decision of which feature to split on is restricted to a *random subset* of size n, from the full feature set. The random subset is chosen anew for each branching point. n is suggested to be $\log_2(N+1)$, where N is the size of the whole feature set.

References

[1] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

Title: Random Subspace Method

Synonyms: RSM, Random Subspaces

Definition

The Random Subspace Method is an Ensemble Learning technique. The principle is to increase diversity between members of the ensemble by restricting classifiers to work on different random subsets of the full feature space. Each classifier learns with a subset of size n, chosen uniformly at random from the full set of size N. Empirical studies have suggested good results can be obtained with the rule-of-thumb to choose $n = \frac{N}{2}$ features. The method is generally found to perform best when there are a large number of features (large N), and the discriminative information is spread across them. The method can underperform in the converse situation, when there are few informative features, and a large number of noisy/irrelevant features. Random Forests is an algorithm combining RSM with the Bagging algorithm, which can provide significant gains over each used separately.

References

[1] T.K. Ho. The Random Subspace Method for constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Title: Rotation Forests

Synonyms: None

Definition

Rotation Forests is an Ensemble Learning technique. It is similar to the Random Forests approach to building decision tree ensembles. In the first step, the original feature set is split randomly into K disjoint subsets. Next, Principal Components Analysis is used to extract n principal component dimensions from each of the K subsets. These are then pooled, and the original data projected linearly into this new feature space. A tree is then built from this data in the usual manner. This process is repeated to create an ensemble of trees, each time with a different random split of the original feature set.

As the tree learning algorithm builds the classification regions using hyperplanes parallel to the feature axes, a small rotation of the axes may lead to a very different tree. The effect of rotating the axes is that classification regions of high accuracy can be constructed with far fewer trees than in Bagging and Adaboost.

References

[1] J.J. Rodriguez, L.I. Kuncheva, and C.J. Alonso. Rotation Forest: a new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, pages 1619–1630, 2006.

Title: Stacked Generalization

Synonyms: Stacking

Definition

Stacking is an Ensemble Learning technique. A set of models are constructed from bootstrap samples of a dataset, then their outputs on a hold-out dataset are used as *input* to a 'meta'-model. The set of base models are called *level-0*, and the meta-model *level-1*. The task of the level-1 model is to combine the set of outputs so as to correctly classify the target, thereby correcting any mistakes made by the level-0 models.

References

[1] D. H. Wolpert Stacked Generalization. *Neural Networks 5(2).*, pages 241–259, 1992.

Title: Unstable Learner

Definition

An unstable learner produces large differences in generalization patterns when small changes are made to its initial conditions. The obvious initial condition is the set of training data used—for an unstable learner, sampling a slightly different training set produces a large difference in testing behavior. Some models can be unstable in additional ways, for example Neural Networks are unstable with respect to the initial weights. In general this is an undesirable property—high sensitivity to training conditions is also known as high variance, which results in higher overall mean squared error. The flexibility enabled by being sensitive to data can thus be a blessing or a curse. Unstable learners can however be used to an advantage in Ensemble Learning methods, where large variance is 'averaged out' across multiple learners.

Examples of unstable learners are: Neural Networks (assuming gradient descent learning), and Decision Trees. Examples of stable learners are Support Vector Machines, K-nearest neighbor classifiers, and Decision Stumps. It should of course be recognized that there is a continuum between 'stable' and 'unstable', and the opinion of whether something is 'sensitive' to initial conditions is somewhat of a subjective one. See also Bias-Variance decomposition for a more formal interpretation of this concept.