# Ensemble Learning

Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, Oregon 97331-3202 USA
tgd@cs.orst.edu

September 4, 2002

To appear in *The Handbook of Brain Theory and Neural Networks, Second edition,* (M.A. Arbib, Ed.), Cambridge, MA: The MIT Press, 2002. http://mitpress.mit.edu

# 1 Introduction

“Learning” describes many different activities ranging from CONCEPT LEARNING (q.v.) to REINFORCEMENT LEARNING (q.v.). The best-understood form of statistical learning is known as *supervised learning* (see LEARNING AND STATISTICAL INFERENCE). In this setting, each data point consists of a vector of features (denoted $\mathbf{x}$) and a class label $y$, and it is assumed that there is some underlying function $f$ such that $y = f(\mathbf{x})$ for each training data point $(\mathbf{x}, y)$. The goal of the learning algorithm is to find a good approximation $h$ to $f$ that can be applied to assign labels to new $x$ values. The function $h$ is called a *classifier*, because it assigns class labels $y$ to input data points $x$. Supervised learning can be applied to many problems including handwriting recognition, medical diagnosis, and part-of-speech tagging in language processing.

Ordinary machine learning algorithms work by searching through a space of possible functions, called *hypotheses*, to find the one function, $h$, that is the best approximation to the unknown function $f$. To determine which hypothesis $h$ is best, a learning algorithm can measure how well $h$ matches $f$ on the training data points, and it can also assess how consistent $h$ is with any available prior knowledge about the problem.

As an example, consider the problem of learning to pronounce the letter “K” in English. Consider the words “desk”, “think”, and “hook” where the “K” is pronounced, and the words “back”, “quack”, and “knave” where the “K” is silent (in “back” and “quack”, we will suppose that the “C” is responsible for the “k” sound). Suppose we define a vector of features that consists of the two letters prior to the “K” and the two letters that follow the “K”. Then each of these words can be represented by the following data points:

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| e | s | _ | _ | $+1$ |
| i | n | _ | _ | $+1$ |
| o | o | _ | _ | $+1$ |
| a | c | _ | _ | $-1$ |
| a | c | _ | _ | $-1$ |
| _ | _ | n | a | $-1$ |

where $y = +1$ if the “K” is pronounced and $-1$ if the “K” is silent, and where “_” denotes positions beyond the ends of the word.

One of the most efficient and widely-applied learning algorithms searches the hypothesis space consisting of decision trees. Figure 1 shows a decision tree that explains the data points given above. This tree can be used to classify a new data point as follows. Starting at the so-called “root” (i.e., the top) of the tree, we first check whether $x_2 =$ ”c”. If so, then we follow the left (“yes”) branch to the $y = -1$ “leaf”, which predicts that the “K” will be silent. If not, we follow the right (“no”) branch to another test: Is $x_3 =$ “n”. If so, then we follow the left branch to another $y = -1$ leaf. If not, then we follow the right branch to the $y = +1$ leaf, where the tree indicates that the “K” should be pronounced.

A decision tree learning algorithm searches the space of such trees by first considering trees that test only one feature (in this case $x_2$ was chosen) and making an immediate classification. Then they consider expanding the tree by replacing one of the leaves by a test of a second feature (in this case, the right leaf was replaced with a test of $x_3$). Various heuristics are applied to choose which test to include in each iteration and when to stop growing the tree. For a good discussion of decision trees, see the books by Quinlan (1993) and by Breiman, et al. (1984).

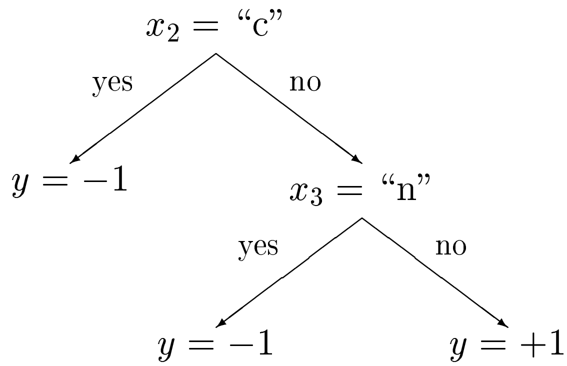


Figure 1: A decision tree for pronouncing the letter "K". First, feature $x_2$ is tested to see if it is the letter "c". If not, the feature $x_3$ is tested to see if it is the letter "n". "K" is pronounced only if $x_2$ is not "c" and $x_3$ is not "n".

In addition to decision trees, there are many other representations for hypotheses that have been studied including PERCEPTRONS, ADALINES, and BACKPROPAGATION (q.v.), RADIAL BASIS FUNCTION NETWORKS (q.v.), GAUSSIAN PROCESSES (q.v.), GRAPHICAL MODELS (q.v.), HELMHOLTZ MACHINES (q.v.), and SUPPORT VECTOR MACHINES (q.v.). In all cases, these algorithms find one best hypothesis $h$ and output it as the "solution" to the learning problem.

Ensemble learning algorithms take a different approach. Rather than finding one best hypothesis to explain the data, they construct a *set* of hypotheses (sometimes called a "committee" or "ensemble") and then have those hypotheses "vote" in some fashion to predict the label of new data points. More precisely, an ensemble method constructs a set of hypotheses $\{h_1, \ldots, h_K\}$, chooses a set of weights $\{w_1, \ldots, w_K\}$ and constructs the "voted" classifier $H(\mathbf{x}) = w_1 h_1(\mathbf{x}) + \ldots + w_K h_K(\mathbf{x})$. The classification decision of the combined classifier $H$ is $+1$ if $H(\mathbf{x}) \geq 0$ and $-1$ otherwise.

Experimental evidence has shown that ensemble methods are often much more accurate than any single hypothesis. Freund and Schapire (1996) showed improved performance in 22 benchmark problems, equal performance in one problem, and worse performance in four problems. These and other studies are summarized in Dietterich (1997).

## 2 Why Ensemble Methods Work

Learning algorithms that output only a single hypothesis suffer from three problems that can be partly overcome by ensemble methods: the statistical problem, the computational problem, and the representation problem.

The statistical problem arises when the learning algorithm is searching a space of hypotheses that is too large for the amount of available training data. In such cases, there may be several different hypotheses that all give the same accuracy on the training data, and the learning algorithm must choose one of these to output. There is a risk that the chosen hypothesis will not predict future data points well. A simple vote of all of these equally-good classifiers can reduce this risk.

The computational problem arises when the learning algorithm cannot guarantee to find the best hypothesis within the hypothesis space. In neural network and decision tree algorithms, for example, the task of finding the hypothesis that best fits the training data is computationally intractable, so heuristic methods must be employed. These heuristics (such as gradient descent) can get stuck in local minima and hence fail to find the best hypothesis. As with the statistical problem, a weighted combination of several different local minima can reduce the risk of choosing the wrong local minimum to output.

Finally, the representational problem arises when the hypothesis space does not contain any hypotheses that are good approximations to the true function $f$. In some cases, a weighted sum of hypotheses expands the space of functions that can be represented. Hence, by taking a weighted vote of hypotheses, the learning algorithm may be able to form a more accurate approximation to $f$.

A learning algorithm that suffers from the statistical problem is said to have high "variance". An algorithm that exhibits the computational problem is sometimes described has having "computational variance". And a learning algorithm that suffers from the representational problem is said to have high "bias". Hence, ensemble methods can reduce both the bias and the variance of learning algorithms. Experimental measurements of bias and variance have confirmed this.

# 3 Review of Ensemble Algorithms

Ensemble learning algorithms work by running a "base learning algorithm" multiple times, and forming a vote out of the resulting hypotheses. There are two main approaches to designing ensemble learning algorithms.

The first approach is to construct each hypothesis independently in such a way that the resulting set of hypotheses is accurate and diverse—that is, each individual hypothesis has a reasonably low error rate for making new predictions and yet the hypotheses disagree with each other in many of their predictions. If such an ensemble of hypotheses can be constructed, it is easy to see that it will be more accurate than any of its component classifiers, because the disagreements will "cancel out." Such ensembles can overcome both the statistical and computational problems discussed above.

The second approach to designing ensembles is to construct the hypotheses in a coupled fashion so that the weighted vote of the hypotheses gives a good fit to the data. This approach directly addresses the representational problem discussed above.

We will discuss each of these two approaches in turn.

## 3.1 Methods for Independently Constructing Ensembles

One way to force a learning algorithm to construct multiple hypotheses is to run the algorithm several times and provide it with somewhat different training data in each run. For example, Breiman (1996) introduced the Bagging ("Bootstrap Aggregating") method which works as follows. Given a set of $m$ training data points, Bagging chooses in each iteration a set of data points of size $m$ by sampling uniformly with replacement from the original data points. This creates a resampled data set in which some data points appear multiple times and other data points do not appear at all. If the learning algorithm is *unstable*—that is, if small changes in the training data lead to large changes in the resulting hypothesis—then Bagging will produce a diverse ensemble of hypotheses.

A second way to force diversity is to provide a different subset of the input features in each call to the learning algorithm. For example, in a project to identify volcanoes on Venus, Cherkauer (1996) trained an ensemble of 32 neural networks. The 32 networks were based on 8 different subsets of the 119 available input features and 4 different network sizes. The input feature subsets were selected (by hand) to group together features that were based on different image processing operations (such as principal component analysis and the fast fourier transform). The resulting ensemble classifier was significantly more accurate than any of the individual neural networks.

A third way to force diversity is to manipulate the output labels of the training data. Dietterich and Bakiri (1995) describe a technique called error-correcting output coding. Suppose that the number of classes, $C$, is large. Then new learning problems can be constructed by randomly partioning the $C$ classes into two subsets $A_k$ and $B_k$. The input data can then be re-labeled so that any of the original classes in set $A_k$ are given the derived label $-1$ and the original classes in set $B_k$ are given the derived label 1. This relabeled data is then given to the learning algorithm, which constructs a classifier $h_k$. By repeating this process $K$ times (generating different subsets $A_k$ and $B_k$), an ensemble of $K$ classifiers $h_1, \ldots, h_K$ is obtained.

Now given a new data point $\mathbf{x}$, how should it be classified? The answer is to have each $h_k$ classify $\mathbf{x}$. If $h_k(\mathbf{x}) = -1$, then each class in $A_k$ receives a vote. If $h_k(\mathbf{x}) = 1$, then each class in $B_k$ receives a vote. After each of the $K$ classifiers has voted, the class with the highest number of votes is selected as the prediction of the ensemble.

An equivalent way of thinking about this method is that each class $j$ is encoded as an $K$-bit codeword $C_j$, where bit $k$ is 1 if $j \in B_k$ and 0 otherwise. The $k$-th learned classifier attempts to predict bit $k$ of these codewords (a prediction of $-1$ is treated as a binary value of 0). When the $L$ classifiers are applied to classify a new point $\mathbf{x}$, their predictions are combined into a $K$-bit binary string. The ensemble's prediction is the class $j$ whose codeword $C_j$ is closest (measured by the number of bits that agree) to the $K$-bit output string. Methods for designing good error-correcting codes can be applied to choose the codewords $C_j$ (or equivalently, subsets $A_k$ and $B_k$). Dietterich and Bakiri report that this technique improves the performance of both decision-tree and backpropagation learning algorithms on a variety of difficult classification problems.

A fourth way of generating accurate and diverse ensembles is to inject randomness into the learning algorithm. For example, the backpropagation algorithm can be run many times, starting each time from a different random setting of the weights. Decision tree algorithms can be randomized by adding randomness to the process of choosing which feature and threshold to split on. Dietterich (2000) showed that randomized trees gave significantly improved performance in 14 out of 33 benchmark tasks (and no change in the remaining 19 tasks).

Ho (1998) introduced the "random subspace method" for growing collections of decision trees ("decision forests"). This method chooses a random subset of the features at each node of the tree, and constrains the tree-growing algorithm to choose its splitting rule from among this subset. She reports improved performance in 16 benchmark datasets. Breiman (2001) combines Bagging with the random subspace method to grow random decision forests that give excellent performance.

## 3.2 Methods for Coordinated Construction of Ensembles

In all of the methods described above, each hypothesis $h_k$ in the ensemble is constructed independently of the others by manipulating the inputs, the outputs, the features, or by injecting randomness. Then an unweighted vote of the hypotheses determines the final classification of a data point.

A contrasting view of an ensemble is that it is an *additive model*—that is, it predicts the class of a new data point by taking an weighted sum of a set of component models. This view suggests developing algorithms that choose the component models and the weights so that the weighted sum fits the data well. In this approach, the choice of one component hypothesis influences the choice of other hypotheses and of the weights assigned to them. In statistics, such ensembles are known as *generalized additive models* (Hastie & Tibshirani, 1990).

The Adaboost algorithm introduced by Freund and Schapire (1996, 1997) is an extremely effective method for constructing an additive model. It works by incrementally adding one hypothesis at a time to an ensemble. Each new hypothesis is constructed by a learning algorithm that seeks to minimize the classification error on a *weighted* training data set. The goal is to construct a weighted sum of hypotheses such that $H(\mathbf{x}_i) = \sum_k w_k h_k(\mathbf{x}_i)$ has the same sign as $y_i$, the correct label of $\mathbf{x}_i$.

The algorithm operates as follows. Let $d_k(\mathbf{x}_i)$ be the weight on data point $\mathbf{x}_i$ during iteration $k$ of the algorithm. Initially, all training data points $i$ are given a weight $d_1(\mathbf{x}_i) = 1/m$, where $m$ is the number of data points. In iteration $k$, the underlying learning algorithm constructs hypothesis $h_k$ to minimize the weighted training error. The resulting weighted error is $r = \sum_i d(\mathbf{x}_i) y_i h_k(\mathbf{x}_i)$, where $h_k(\mathbf{x}_i)$ is the label predicted by hypothesis $h_k$. The weight assigned to this hypothesis is computed by

$$w_k = \frac{1}{2} \ln \left( \frac{1+r}{1-r} \right).$$

To compute the weights for the next iteration, the weight of training data point $i$ is set to

$$d_{k+1}(\mathbf{x}_i) = d_k(\mathbf{x}_i) \frac{\exp(-w_k y_i h_k(\mathbf{x}_i))}{Z_k},$$

where $Z_k$ is chosen to make $d_{k+1}$ sum to 1.

Breiman (1997) showed that this algorithm is a form of gradient optimization in function space with the goal of minimizing the objective function

$$J(H) = \sum_i \exp(-y_i H(\mathbf{x}_i)).$$

The quantity $y_i H(\mathbf{x}_i)$ is called the *margin*, because it is the amount by which $\mathbf{x}_i$ is correctly classified. If the margin is positive, then the sign of $H(\mathbf{x}_i)$ agrees with the sign of $y_i$. Minimizing $J$ causes the margin to be maximized. Friedman, Hastie, and Tibshirani (2000) expand on Breiman's analysis from a satistical perspective.

In most experimental studies (Freund & Schapire, 1996; Bauer & Kohavi, 1999; Dietterich, 2000), Adaboost (and algorithms based on it) gives the best performance on the vast majority of data sets. The primary exception are data sets in which there is a high level of mislabeled training data points. In such cases, Adaboost will put very high weights on the noisy data points and learn very poor classifiers. Current research is focusing on methods for extending Adaboost to work in high noise settings.

The exact reasons for Adaboost's success are not fully understood. One line of explanation is based on the margin analysis developed by Vapnik (1995) and extended by Schapire, Freund, Barlett, and Lee (1998). This work shows that the error of an ensemble on new data points is bounded by the fraction of training data points for which the margin is less than some quantity $\Theta > 0$ plus a term that grows as

$$\sqrt{\frac{d}{m}} \frac{\log(m/d)}{\Theta},$$

ignoring constant factors and some log terms. In this formula, $m$ is the number of training data points, and $d$ is a measure of the expressive power of the hypothesis space from which the individual classifiers are drawn, known as the VC-dimension. The value of $\Theta$ can be chosen to minimize the value of this expression.

Intuitively, this formula says that if the ensemble learning algorithm can achieve a large "margin of safety" on each training data point while using only a weighted sum of simple classifiers, then the resulting voted classifier is likely to be very accurate. Experimentally, Adaboost has been shown to be very effective at increasing the margins on the training data points, and hence, this result suggests that Adaboost will make few errors on new data points.

There are three ways in which this analysis has been criticized. First, the bound is not tight, so it may be hiding the real explanation for Adaboost's success. Second, even when Adaboost is applied to large decision trees and neural networks, it is observed to work very well even though these representations have high VC-dimension. Third, it is possible to design algorithms that are more effective than Adaboost at increasing the margin on the training data, but these algorithms exhibit worse performance than Adaboost when applied to classify new data points.

### 3.3 Related Non-Ensemble Learning Methods

In addition to the ensemble methods described here, there are other non-ensemble learning algorithms that similar. For example, any method for constructing a classifier as a weighted sum of basis functions (e.g., see RADIAL BASIS FUNCTION NETWORKS) can be viewed as an additive ensemble where each individual basis function forms one of the hypotheses.

Another close-related learning algorithm is the Hierarchical Mixture of Experts method (see MODULAR AND HIERARCHICAL LEARNING SYSTEMS). In a hierarchical mixture, individual hypotheses are combined by a gating network which decides—based on the features of the data point—what weights should be employed. This differs from Adaboost and other additive ensembles where the weights are determined once during training and then held constant thereafter.

## 4 Discussion

The majority of research into ensemble methods has focused on constructing ensembles of decision trees. Decision tree learning algorithms are known to suffer from high variance, because they make a cascade of choices (of which variable and value to test at each internal node in the decision tree) such that one incorrect choice has an impact on all subsequent decisions. In addition, because the internal nodes of the tree test only a single variable, this creates axis-parallel rectangular decision regions which can have high bias. Consequently, ensembles of decision tree classifiers perform much better than individual decision trees. Recent experiments suggest that Breiman's combination of bagging and the random subspace method is the method of choice for decision trees — it gives excellent accuracy and works well even when there is substantial noise in the training data.

If the base learning algorithm produces less expressive hypotheses than decision trees, then the Adaboost method is recommended. Many experiments have employed so-called "decision stumps", which are decision trees with only one internal node. In order to learn complex functions with decision stumps, it is important to exploit Adaboost's ability to directly construct an additive model. This usually gives better results than Bagging and other accuracy/diversity methods. Similar recommendations apply to ensembles constructed using the Naive Bayes and Fisher's linear discriminant algorithms. Both of these learn a single linear discrimination rule. The algorithms are very stable, which means that even substantial (random) changes to the training data do not cause the learned discrimination rule to change very much. Hence, methods like Bagging that rely on instability do not produce diverse ensembles.

Because the generalization ability of a single feed-forward neural network is usually very good, neural networks benefit less from ensemble methods. Adaboost is probably the best method to apply, but favorable results have been obtained just by training several networks from different random starting weight values, and Bagging is also quite effective.

For multiclass problems, the error-correcting output coding algorithm can produce good ensembles. However, because the output coding can create difficult two-class learning problems, it is important that the base learner be very expressive. The best experimental results have been obtained with very large decision trees and neural networks. In addition, the base learning algorithm must be sensitive to the encoding of the output values. The nearest neighbor algorithm does not satisfy this constraint, because it merely identifies the training data point $\mathbf{x}_i$ nearest to the new point $\mathbf{x}$, and outputs the corresponding value $y_i$ as the prediction for $h(\mathbf{x})$ regardless of how $y_i$ is encoded. Current research is exploring ways of integrating error-correcting output codes directly into the Adaboost algorithm.

## 5 References

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

*Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., 1984, *Classification and Regression Trees*, Wadsworth International Group.

*Breiman, L., 1996, Bagging predictors, *Machine Learning*, 24:123–140.

Breiman, L., 1997, Arcing the edge, `http://citeseer.nj.nec.com/breiman97arcing.html`, Technical Report 486, Department of Statistics, University of California, Berkeley, CA.

Breiman, L., 2001, Random forests, *Machine Learning*, 45:5–32.

Cherkauer, K. J., 1996, Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks, in *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models*, (P. Chan, Ed.), Menlo Park: AAAI Press, pp. 15–21.

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

Dietterich, T. G., and Bakiri, G., 1995, Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research*, 2:263–286.

*Dietterich, T. G., 1997, Machine learning research: Four current directions, *AI Magazine*, 18:97–136.

Freund, Y., and Schapire, R. E., 1996, Experiments with a new boosting algorithm, In *Proc. 13th International Conference on Machine Learning*, pp. 148–146. San Francisco: Morgan Kaufmann.

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

*Friedman, J. H., Hastie, T., and Tibshirani, R., 2000, Additive logistic regression: A statistical view of boosting, *Annals of Statistics*, 28:337–407.

*Hastie, T. J., and Tibshirani, R. J., 1990, *Generalized additive models*, London: Chapman and Hall.

Ho, T. K., 1998, The random subspace method for constructing decision forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844.

*Quinlan, J. R., 1993, *C4.5: Programs for Empirical Learning.* San Francisco: Morgan Kaufmann.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S., 1998, Boosting the margin: A new explanation for the effectiveness of voting methods, *Annals of Statistics*, 26:1651–1686.

Vapnik, V., 1995, *The Nature of Statistical Learning Theory.* New York: Springer.