Ensemble Data Mining Methods

Nikunj C. Oza
NASA Ames Research Center, USA

INTRODUCTION

Ensemble data mining methods, also known as *committee methods* or *model combiners,* are machine learning methods that leverage the power of multiple models to achieve better prediction accuracy than any of the individual models could on their own. The basic goal when designing an ensemble is the same as when establishing a committee of people: Each member of the committee should be as competent as possible, but the members should complement one another. If the members are not complementary, that is, if they always agree, then the committee is unnecessary — any one member is sufficient. If the members are complementary, then when one or a few members make an error, the probability is high that the remaining members can correct this error. Research in ensemble methods has largely revolved around designing ensembles consisting of competent yet complementary models.

BACKGROUND

A supervised machine learning task involves constructing a mapping from input data (normally described by several features) to the appropriate outputs. In a classification learning task, each output is one or more classes to which the input belongs. The goal of classification learning is to develop a model that separates the data into the different classes, with the aim of classifying new examples in the future. For example, a credit card company may develop a model that separates people who defaulted on their credit cards from those who did not, based on other known information such as annual income. The goal would be to predict whether a new credit card applicant is likely to default on his or her credit card and thereby decide whether to approve or deny this applicant a new card. In a regression learning task, each output is a continuous value to be predicted (e.g., the average balance that a credit card holder carries over to the next month).

Many traditional machine learning algorithms generate a single model (e.g., a decision tree or neural network). Ensemble learning methods instead generate multiple models. Given a new example, the ensemble passes it to each of its multiple *base* models, obtains their predictions, and then combines them in some appropriate manner (e.g., averaging or voting). As mentioned earlier, it is important to have base models that are competent but also complementary. To further motivate this point, consider Figure 1. This figure depicts a classification problem in which the goal is to separate the points marked with plus signs from points marked with minus signs. None of the three individual linear classifiers (marked *A, B,* and *C*) is able to separate the two classes of points. However, a majority vote over all three linear classifiers yields the piecewise-linear classifier shown as a thick line. This classifier is able to separate the two classes perfectly. For example, the plusses at the top of the figure are correctly classified by *A* and *B,* but are misclassified by *C.* The majority vote over these classifiers correctly identifies them as plusses. This happens because *A* and *B* are very different from *C.* If our ensemble instead consisted of three copies of *C,* then all three classifiers would misclassify the plusses at the top of the figure, and so would a majority vote over these classifiers.

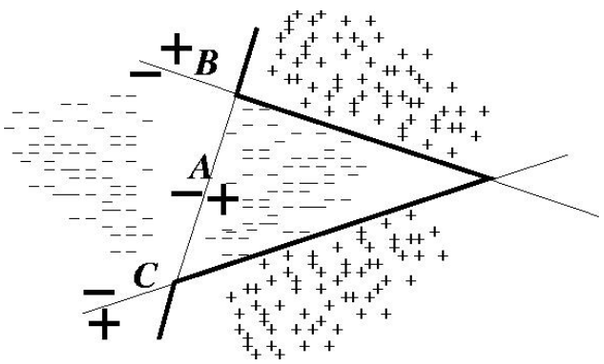
MAIN THRUST

We now discuss the key elements of an ensemble-learning method and ensemble model and, in the process, discuss several ensemble methods that have been developed.

Ensemble Methods

The example shown in Figure 1 is an artificial example. We cannot normally expect to obtain base models that misclassify examples in completely separate parts of the input space and ensembles that classify all the examples correctly. However, many algorithms attempt to generate a set of base models that make errors that are as uncorrelated as possible. Methods such as *bagging* (Breiman, 1994) and *boosting* (Freund & Schapire, 1996) promote diversity by presenting each base model with a different subset of training examples or different weight distributions over the examples. For example, in Figure 1, if the plusses in the top part of the figure were temporarily removed from the training set, then a linear classifier learning algorithm trained on the remaining examples would probably yield a classifier similar to *C.* On the other hand, removing the

Figure 1. An ensemble of linear classifiers



plusses in the bottom part of the figure would probably yield classifier *B,* or something similar. In this way, running the same learning algorithm on different subsets of training examples can yield very different classifiers, which can be combined to yield an effective ensemble. *Input decimation ensembles (IDE)* (Oza & Tumer, 2001; Tumer & Oza, 2003) and *stochastic attribute selection committees (SASC)* (Zheng & Webb, 1998) instead promote diversity by training each base model with the same training examples but different subsets of the input features. SASC trains each base model with a random subset of input features. IDE selects, for each class, a subset of features that has the highest correlation with the presence of that class. Each feature subset is used to train one base model. However, in both SASC and IDE, all the training patterns are used with equal weight to train all the base models.

So far we have distinguished ensemble methods by the way they train their base models. We can also distinguish methods by the way they combine their base models' predictions. Majority or plurality voting is frequently used for classification problems and is used in bagging. If the classifiers provide probability values, simple averaging is commonly used and is very effective (Tumer & Ghosh, 1996). Weighted averaging has also been used, and different methods for weighting the base models have been examined. Two particularly interesting methods for weighted averaging include *mixtures of experts* (Jordan & Jacobs, 1994) and Merz's use of *principal components analysis (PCA)* to combine models (Merz, 1999). In the mixtures of experts method, the weights in the weighted average combination are determined by a gating network, which is a model that takes the same inputs that the base models take and returns a weight on each of the base models. The higher the weight for a base model, the more that base model is trusted to provide the correct answer. These weights are determined during training by how well the base models perform on the training examples. The gating network essentially keeps track of how well each base model performs in each part of the input space. The hope is that each model learns to specialize in different input regimes and is weighted highly when the input falls into its specialty. Merz's method uses PCA to lower the weights of base models that perform well overall but are redundant and, therefore, effectively give too much weight to one model. For example, in Figure 1, if an ensemble of three models instead had two copies of *A* and one copy of *B,* we may prefer to lower the weights of the two copies of *A* because, essentially, *A* is being given too much weight. Here, the two copies of *A* would always outvote *B,* thereby rendering *B* useless. Merz's method also increases the weight on base models that do not perform as well overall but perform well in parts of the input space, where the other models perform poorly. In this way, a base model's unique contributions are rewarded.

When designing an ensemble learning method, in addition to choosing the method by which to bring about diversity in the base models and choosing the combining method, one has to choose the type of base model and base model learning algorithm to use. The combining method may restrict the types of base models that can be used. For example, to use average combining in a classification problem, one must have base models that can yield probability estimates. This precludes the use of linear discriminant analysis or support vector machines, which cannot return probabilities. The vast majority of ensemble methods use only one base model learning algorithm but use the methods described earlier to bring about diversity in the base models. Surprisingly, little work has been done (e.g., Merz, 1999) on creating ensembles with many different types of base models.

Two of the most popular ensemble learning algorithms are bagging and boosting, which we briefly explain next.

Bagging

Bootstrap aggregating (Bagging) generates multiple bootstrap training sets from the original training set (by using sampling with replacement) and uses each of them to generate a classifier for inclusion in the ensemble. The algorithms for bagging and sampling with replacement are given in Figure 2. In these algorithms, T is the original training set of N examples, M is the number of base models to be learned, L_b is the base model learning algorithm, the h_is are the base models, *random_integer (a,b)* is a function that returns each of the integers from a to b with equal probability, and *I(A)* is the indicator function that returns 1 if A is true and 0 otherwise.

To create a bootstrap training set from an original training set of size N, we perform N multinomial trials, where in each trial, we draw one of the N examples. Each example has the probability $1/N$ of being drawn in each