15
Ensemble Learning

The expression *ensemble learning* refers to a broad class of classification and regression algorithms operating on many learners. Every learner in an ensemble is typically *weak*; by itself it would predict on new data quite poorly. By aggregating predictions from its weak learners, the ensemble often achieves an excellent predictive power. The number of weak learners in an ensemble usually varies from a few dozen to a few thousand. One of the most popular choices for the weak learner is decision tree. However, every classifier reviewed in this book, so far, can be used in the weak learner capacity.

If we repeatedly trained the same weak learner on the same data, all learners in the ensemble would be identical and the ensemble would be as poor as the weak learner itself. Data generation (induction) for the weak learner is crucial for ensemble construction. The weak learner can be applied to the induced data independently, without any knowledge of the previously learned weak models, or taking into account what has been learned by the ensemble participants, so far. The training data can be then modified for consecutive learners. The final step, undertaken after all weak learners have been constructed, is aggregation of predictions from these learners.

Approaches to ensemble learning are quite diverse and exercise a variety of choices for data induction and modification, weak learner construction and ensemble aggregation. The field of ensemble learning is comprised of algorithms derived from different principles and methodologies. In this chapter, we review a few well-known ensemble algorithms. By necessity we omit many others.

Ideas that laid foundation to ensemble learning can be traced to decades ago. The first practical ensemble algorithms came out in the mid 1990s. The first convincing application of an ensemble technique to particle physics is particle identification by boosted trees at MiniBOONE (Yang *et al.*, 2005). At the time of this writing, ensemble learning has become fairly popular in the particle and astrophysics communities, although its applications have been mostly confined to AdaBoost and random forest. We aim to broaden the arsenal of the modern physicist with other algorithms.

Statistical Analysis Techniques in Particle Physics, First Edition. Ilya Narsky and Frank C. Porter.

15.1 Boosting

A popular class of ensemble methods is boosting. Algorithms in this class are sequential: the sampling distribution is modified for every weak learner using information from the weak learners constructed earlier. This modification typically amounts to adjusting weights for observations in the training set, allowing the next weak learner to focus on the poorly studied region of the input space.

In this section, we review three theoretical approaches to boosting, namely minimization of convex loss by stagewise additive modeling, maximization of the minimal classification margin, and minimization of nonconvex loss. The AdaBoost algorithm, which has gained some popularity in the physics community, can be explained in several ways. In one interpretation, AdaBoost works by minimizing the (convex) exponential loss shown in Table 9.1. In another interpretation, AdaBoost achieves high accuracy by maximizing the minimal margin. These two interpretations gave rise to other successful boosting algorithms, to be discussed here as well. Boosting by minimizing nonconvex loss can succeed in the regime where these two approaches fail, that is, in the presence of significant label noise. We focus on binary classification and review multiclass extensions in the concluding section.

15.1.1 Early Boosting

An algorithm introduced in Schapire (1990) may be the first boosting algorithm described in a journal publication. Assume that you have an infinite amount of data available for learning. Assume that the classes are perfectly separable, that is, an oracle can correctly label any observation without prior knowledge of its true class. Define your objective as constructing an algorithm such that, when applied to independent chunks of data drawn from the same parent distribution, learns a model with generalization error at most ϵ at least $100(1-\delta)\%$ of the time. As usual, generalization error is measured on observations not seen at the learning (training) stage.

At the core of this algorithm, there is a simple three-step procedure which works as follows. Draw a training set. Learn the first classifier on this set. Form a new training set with observations correctly classified and misclassified by the first classifier mixed in equal proportion. Learn the second classifier on this set. Make a third training set by drawing new observations for which the first and second classifiers disagree. Learn the third classifier on this set. The three training sets must be sufficiently large to provide the desired values of ϵ and δ. To predict the label for a new observation using the learned three-step model, put this observation through the first two classifiers. If they agree, assign the predicted class label. If they do not agree, use the label predicted by the third classifier. This completes one three-step pass.

Algorithm 1 AdaBoost for two classes. Class labels y_n are drawn from the set $\{-1, +1\}$. The indicator function I equals one if its argument is true and zero otherwise.

Input: Training data $\{\boldsymbol{x}_n, y_n, w_n\}_{n=1}^N$ and number of iterations T

1: **initialize** $w_n^{(1)} = \frac{w_n}{\sum_{n=1}^N w_n}$ for $n = 1, \ldots, N$
2: **for** $t = 1$ **to** T **do**
3: Train classifier on $\{\boldsymbol{x}_n, y_n, w_n^{(t)}\}_{n=1}^N$ to obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$
4: Calculate training error $\epsilon_t = \sum_{n=1}^N w_n^{(t)} I(y_n \neq h_t(\boldsymbol{x}_n))$
5: **if** $\epsilon_t == 0$ **or** $\epsilon_t \geq 1/2$ **then**
6: $T = t - 1$
7: *break loop*
8: **end if**
9: Calculate hypothesis weight $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
10: Update observation weights $w_n^{(t+1)} = \frac{w_n^{(t)} \exp[-\alpha_t y_n h_t(\boldsymbol{x}_n)]}{\sum_{n=1}^N w_n^{(t)} \exp[-\alpha_t y_n h_t(\boldsymbol{x}_n)]}$
11: **end for**

Output: $f(\boldsymbol{x}) = \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x})$

Learn the classifiers in this three-step procedure recursively, using the same three-step procedure. At every level of recursion, relax the desired value of ϵ forcing it to gradually approach 1/2. When ϵ gets sufficiently close to 1/2, stop and return a weak learner whose generalization error is below ϵ at this recursion level (and therefore barely below 1/2).

This algorithm is mostly of theoretical value. It relies on construction of the first and second classifier in the three-step scheme with known accuracies, but in practice these are not known. A remarkable accomplishment of Schapire's paper is the proof of concept. To construct a model with an arbitrarily small classification error, all you need is a classifier assigning correct labels to a little more than half of the data!

15.1.2
AdaBoost for Two Classes

It took a few years after Schapire's paper to develop successful practical algorithms. Proposed in Freund and Schapire (1997), *AdaBoost* (adaptive boosting) earned a reputation of a fast and robust algorithm with an excellent accuracy in high-dimensional problems. AdaBoost and similar boosting methods were named "the best available off-the-shelf classifiers" in Breiman (1998). We show this famous algorithm here with a simple modification to account for the initial assignment of observation weights (see Algorithm 1).

AdaBoost initializes observation weights to those in the input data and enters a loop with at most T iterations. At every iteration, AdaBoost learns a weak hypothesis h_t mapping the space of input variables $\mathcal{X}$ onto the class set $\{-1, +1\}$. The hypothesis is learned on weighted data, and the weights are updated at every pass; hence, every time AdaBoost learns a new weak hypothesis. After learning a new weak hypothesis, AdaBoost estimates its error by summing the weights of observations misclassified by this hypothesis. AdaBoost then computes the weight for this hypothesis using line 9 in Algorithm 1. The hypothesis weight α_t approaches $+\infty$ as the error approaches 0, and α_t approaches 0 as the error approaches 1/2. If the error is not in the $(0, 1/2)$ interval, AdaBoost cannot compute α_t and exits. If the error is in the allowed range, AdaBoost multiplies the weights of the misclassified observations by $(1 - \epsilon_t)/\epsilon_t$ and the weights of the correctly classified observations by the inverse of this expression. Then AdaBoost renormalizes the weights to keep their sum at one. The soft score $f(\boldsymbol{x})$ returned by the final strong hypothesis is a weighted sum of predictions from the weak hypotheses. The predicted class is $+1$ if the score is positive and -1 otherwise.

15.1.2.1 Example: Boosting a Hypothetical Dataset

Let us examine step by step how AdaBoost separates two classes in the hypothetical dataset we used in Chapter 13. We show these steps in Figure 15.1. The weak learner for AdaBoost is a decision stump (a tree with two leaves) optimizing the Gini index. Before learning begins, each observation is assigned a weight of 1/7. The first stump separates the two lower stars from the other five observations. Since the upper leaf has more crosses than stars, the two upper stars are misclassified as crosses and their weights increase. The second split separates the left star from the rest of the observations. Because the top star has a large weight, the right leaf is dominated by the stars. Both leaves predict into the star class, and the weights of the crosses increase. The third stump is identical to the second stump. Now the crosses in the right leaf have more weight than the three stars. The three stars in the right leaf are misclassified and their weights increase. The fourth stump separates the top star from the rest.

In about ten iterations, AdaBoost attains perfect class separation. The light rectangle in Figure 15.2a shows the region with a positive classification score. The region with a negative score is shown in semi-dark gray, and the region with a large negative score is shown in dark gray. The class boundary does not look as symmetric as it did in Chapter 13, but keep in mind that we operate under two severe constraints: the limited amount of statistics and the requirement to use decision stumps in the space of the original variables. From the computer point of view, this boundary is as good as the one in Figure 13.1!

Evolution of the observation weights through the first ten iterations is shown in Figure 15.2b. The three crosses always fall on the same leaf, and their weights are therefore always equal. The same is true for the two lower stars.

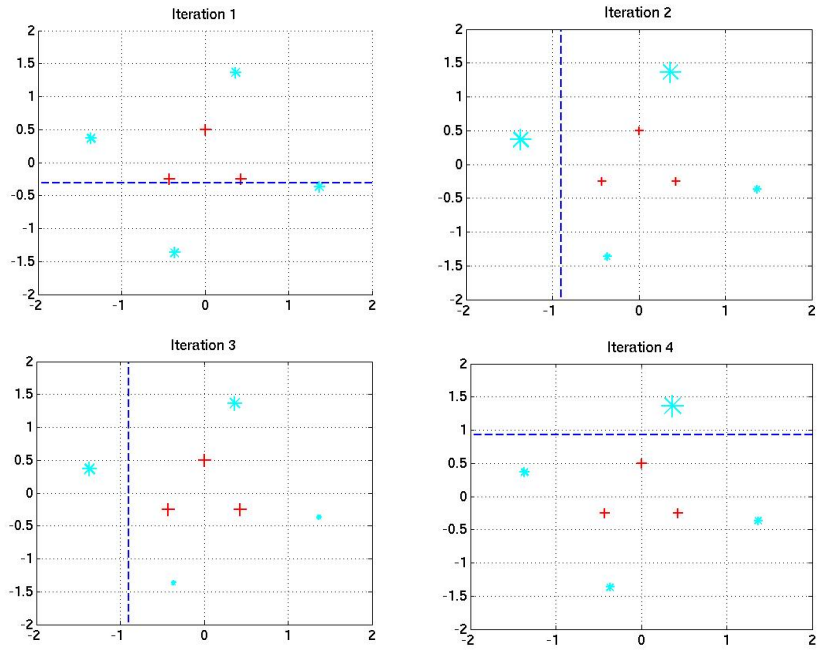

Figure 15.1 First four decision stumps imposed by AdaBoost. The size of each symbol is proportional to the observation weight at the respective iteration before the stump is applied.

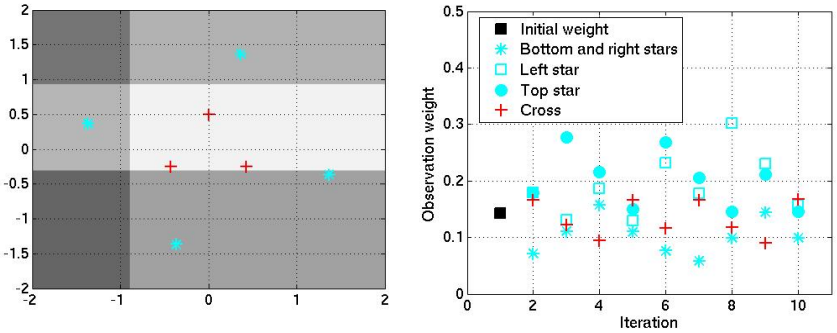

Figure 15.2 (a) Predicted classification scores, and; (b) evolution of observation weights in the first ten iterations.

15.1.2.2 Why is AdaBoost so Successful?

One way to explain AdaBoost is to point out that every weak hypothesis is learned mostly on observations misclassified by the previous weak hypothesis. The strength of AdaBoost can be then ascribed to its skillful use of hard-to-classify observations. This statement, although true in some sense, is insufficient for understanding why AdaBoost works so well on real-world data. Indeed, a few questions immediately arise for Algorithm 1:

- Why use that specific formula on line 9 to compute the hypothesis weight α_t?
- Why use that specific formula on line 10 to update the observation weights?
- What is the best way to construct h_t on line 3? If we chose a very strong learner, the error would quickly drop to zero and AdaBoost would exit. On the other hand, if we chose a very weak learner, AdaBoost would run for many iterations converging slowly. Would either regime be preferable over the other? Is there a golden compromise?
- What is the best way to choose the number of learning iterations T?
- Could there be a more efficient stopping rule than the one on line 5? In particular, does it makes sense to continue learning new weak hypotheses after the training error for the final strong hypothesis drops to zero?

These questions do not have simple answers. A decade and a half after its invention, AdaBoost remains, to some extent, a mystery. Several interpretations of this algorithm have been offered. None of them provides a unified, coherent picture of the boosting phenomenon. Fortunately, each interpretation led to discoveries of new boosting algorithms. At this time, AdaBoost may not be the best choice for the practitioner, and is most certainly not the only boosting method to try.

In the following sections, we review several perspectives on boosting and describe modern algorithms in each class. For a good summary of boosting and its open questions, we recommend Meir and Ratsch (2003); Buhlmann and Hothorn (2007), as well as two papers followed by discussion, Friedman *et al.* (2000) and Mease and Wyner (2008).

15.1.3
Minimizing Convex Loss by Stagewise Additive Modeling

AdaBoost can be interpreted as a search for the minimum of a convex function. Put forward by three Stanford statisticians, Friedman, Hastie and Tibshirani, in the late 1990s, this interpretation has been very popular, if not dominant, and used as a theoretical basis for new boosting algorithms. We describe this view here.

Suppose we measure the quality of a binary classifier using exponential loss, $\ell(y, f) = e^{-yf}$, for labels $y \in \{-1, +1\}$ and scalar soft score $f(\boldsymbol{x})$. Learning a good model $f(\boldsymbol{x})$ over domain $\mathcal{X}$ then amounts to minimizing the expected loss

$$E_{X,Y}[\ell(Y, f(\boldsymbol{X}))] = \int_{\mathcal{X}} [e^{f(\boldsymbol{x})} P(Y = -1|\boldsymbol{x}) + e^{-f(\boldsymbol{x})} P(Y = +1|\boldsymbol{x})] P(\boldsymbol{x}) d\boldsymbol{x} \tag{15.1}$$

for pdf $P(\boldsymbol{x})$ and conditional probability $P(y|\boldsymbol{x})$ of observing class y at $\boldsymbol{x}$. The AdaBoost algorithm can be derived by mathematical induction. Suppose we learn $t-1$ weak hypotheses by AdaBoost to obtain an ensemble

$$f_t(\boldsymbol{x}) = \begin{cases} 0 & \text{if} \quad t = 1 \\ \sum_{i=1}^{t-1} \alpha_i h_i(\boldsymbol{x}) & \text{if} \quad t > 1 \end{cases} . \tag{15.2}$$