AdaBoost (Adaptive Boosting)

Quick note

Abdelwahid Benslimane

wahid.benslimane@gmail.com

The principle of AdaBoost, one of the most used boosting algorithms, is to combine the outputs of several weak classifiers to obtain a stronger result (strong classifier). The weak classifier (a decision tree typically) must have a slightly better basic behaviour than the hazard: error rate less than 0.5 for a binary classification (i.e. it does not make mistakes more than half the time on average, if the distribution of classes is balanced). Each weak classifier is weighted by the quality of its classification: the better it classifies, the more important it will be. The misclassified examples will have a higher weight (we say they are boosted) for the weak learner in the next round, so that it can overcome the lack.

To each learning example x_i is associated a weight w_i which encodes the degree of difficulty of this example to be correctly classified with respect to the weak classifiers already chosen until the current iteration. Initially, all the training examples have the same weight ($w_i = \frac{1}{n}$).

At each iteration (assume that m indicates the number of the current iteration) we choose from a set $\mathcal{G}$ the classifier G_m which minimises the classification error on the training data weighted by the coefficients w_i.

Then we compute the coefficient α_m which is the weight of G_m in the final mixture, we update the weights w_i to boost the misclassified items and proceed to the next iteration. The detailed algorithm is given below.

As input we have the following elements:

- The training data: $(x_1, y_1), \ldots, (x_n, y_n)$, $x_i \in X$, $y_i \in \{-1, 1\}$ (binary classification problem).
- A set $\mathcal{G}$ of weak classifiers.
- The number M of weak classifiers to put in the final mixture.

I. Initialise the weights $w_i = \frac{1}{n}, i = 1, 2, \ldots, n$.

II. For $m = 1$ to M:

1: Choose the classifier $G_m \in \mathcal{G}$ that minimizes the error weighted by the coefficients $w_1, \ldots, w_n$ on the training data:

$$G_m = \underset{G_k \in \mathcal{G}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i I(y_i \neq G_k(x_i)).$$

The unit function $I(\cdots)$ helps to count the number of errors, i.e. $I(y_i \neq G_k(x_i))$ returns 1 if $y_i \neq G_k(x_i)$ and 0 otherwise.

2: Calculate the error rate: $e_m = \dfrac{\displaystyle\sum_{i=1}^{n} w_i I(yi \neq G_m(x_i))}{\displaystyle\sum_{i=1}^{n} w_i}$.

3: Calculate the weight α_m of the classifier G_m:

$\alpha_m = log(\frac{1-e_m}{e_m})$.

We can see that the error e_m must be lower than 0.5, otherwise α_m becomes negative, hence the need for weak classifiers to be slightly better than a random choice. If the error rate e_m is small, it means that we have a good classifier, so its weight α_m in the final mixture will be important. On the contrary, a classifier that makes many errors will have less impact (α_m will be small).

4: Readjust the weights: $w_i = w_i exp(\alpha_m I(yi \neq G_m(x_i))), i = 1..., n$.

If x_i is well classified by G_m ($G_m(x_i) = y_i$), then w_i remains unchanged (since $exp(0) = 1$). Otherwise, x_i is a difficult case and its weight is increased, especially if α_m is large (indicating that a good classifiercannot classify this sample correctly).

III. The final classifier is : $G(x) = sign(\displaystyle\sum_{m=1}^{M} \alpha_m G_m(x))$.

The diagram below summarizes the AdaBoost algorithm.

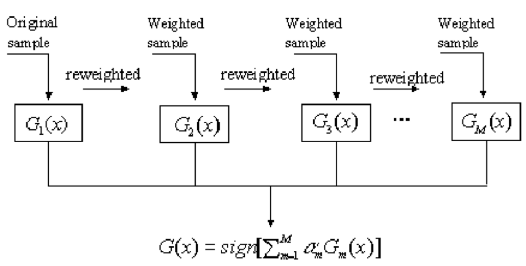

Image source: https://www.researchgate.net/publication/228715208_Boosting_applied_to_classification_of_mass_spectral_data