Notes on Boosting: AdaBoost [1]

"One for all, and all for one"
The Three Musketeers by Alexandre Dumas, père

The AdaBoost Algorithm: From Weak to Strong

Given training examples $(x_1, y_1), \ldots, (x_m, y_m)$ such that $x_i \in X, y_i \in Y = \{-1, +1\}$.

Initialize $D_1(i) = 1/m$. ($D_t(i)$ represents how much weight is given to example i on iteration t.)

For $t = 1, \ldots, T$:

1. Train weak learner using distribution D_t: Outputs a weak classifier $h_t : X \to Y$ (h_t can be an ID tree, a NN-based classifier, ...)

2. Compute the error ϵ_t of the classifier h_t: $\epsilon_t =$ sum of the weights of the data samples that h_t classifies incorrectly, or more formally,

$$\epsilon_t = \sum_{i\,:\,h_t(x_i) \neq y_i} D_t(i)$$

3. Use the error to compute $\alpha'_t \in [0, \infty]$:

$$\alpha'_t = \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

(α'_t is the weight given to weak classifier h_t on the voting done to obtain H. [2])

4. Update the weight on the samples $i = 1, \ldots, m$:

$$D_{t+1}(i) = D_t(i) \times \frac{1}{2} \times \begin{cases} \frac{1}{\epsilon_t} & \text{if } h_t(x_i) \neq y_i \text{ (mistake)}, \\ \frac{1}{1-\epsilon_t} & \text{if } h_t(x_i) = y_i \text{ (correct)} \end{cases}$$

Output the final classifier to be a weighted majority vote of the T base classifiers:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha'_t h_t(x)\right)$$

Slogan: *"T just-above-average heads are as good as 1 excelent"*

[1] These notes were prepared in conjunction with Sourabh Niyogi. (Orig. date: Nov. 18, 2004; Last updated: Nov. 30, 2006)

[2] Note that α'_t is like the $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ presented in lecture *but without the 1/2 factor*; that is, $\alpha'_t = 2\alpha_t$. Removing the 1/2 factor is OK because scaling in any way does not change the output of the sign function, and therefore scaling does not change the final classifier H.

AdaBoost has Excellent Properties

- Integrates disparate and weak classifiers together (*i.e.*, combine classifiers that concentrate on different aspects of the problem or, in other words, put more weight to different data points)
- Theoretical bounds guarantee that adding a new classifier cannot hurt: the *training* error can only decrease!
- Easy to program: can use *any* weak learner; *No* local minima, i.e., achives zero training error eventually
- Tends to resist overfitting *in practice*, yet sensitive to outliers
- Has guaranteed bounds on the *true error* of the final classifier based (roughly) on (1) general assumptions about the true underlying process generating the data; (2) the amount of data m; (3) the number of rounds, i.e., the number T of weak classifiers $\{h_t\}$ used; and (4) the "flexibility" of the weak learning algorithm, e.g., roughly how many parameters the classifier has