Introduction to AdaBoost

Yunfei WANG

yunfeiwang@hust.edu.cn

[1]School of Computer Science & Technology
Huazhong University of Science & Technology

October 22, 2013

What's AdaBoost

AdaBoost(adaptive boosting) is an algorithm for constructing a "strong" classifier as linear combination of "weak" classifiers $h_m(x) \in \{-1, 1\}$

$$f(x) = \sum_{m=1}^{L} \alpha_m h_m(x) \tag{1}$$

where α_m is the contribution factor of $h_m(x)$.

Final classifier: $H(x) = sign(f(x))$.

AdaBoost is adaptive in the sense that subsequent classifiers are chosen in favor of samples misclassified by previous classifiers.

As long as a classifier is slightly better than random, it can improve the final model. Even classifiers with higher error rate will be help, since they'll have negative coefficients and behave like their inverse.

Scouting

Scouting is done by testing each classifier with a training set of N samples $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N) | y_i \in \{-1, 1\}\}$.

	1	2	3	$\cdots$	L
x_1	0	0	1	$\cdots$	1
x_2	1	0	0	$\cdots$	0
x_3	0	1	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_N	1	0	0	$\cdots$	0

Table: Scouting Result of L Classifiers

0-Correct 1-Wrong

Modelling

At the m-th iteration, we have included $m-1$ classifiers, the linear combination of classifiers is

$$f_{m-1}(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_{m-1} h_{m-1}(x) \tag{2}$$

We want to recruit the next classifier

$$f_m(x) = f_{m-1}(x) + \alpha_m h_m(x) \tag{3}$$

The exponential loss of current classifier $sign(f_m(x))$

$$\mathcal{L}_m = \sum_{i=1}^{N} e^{-y_i(f_{m-1}(x_i)+\alpha_m h_m(x_i))} \tag{4}$$

where α_m and h_m are to be determined in an optimal way.

How to choose h_m?

According to the results of classifier h_m, we split $\mathcal{L}_m$ into two parts

$$\mathcal{L}_m = \sum_{i:y_i=h_m(x_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{i:y_i\neq h_m(x_i)} w_i^{(m)} e^{\alpha_m} = W_c e^{-\alpha_m} + W_e e^{\alpha_m} \tag{5}$$

where $w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$ is the weight assigned to each sample.

When selecting h_m, we change the form of $\mathcal{L}_m$

$$\mathcal{L}_m = (W_c + W_e)e^{-\alpha_m} + W_e(e^{\alpha_m} - e^{-\alpha_m}) \tag{6}$$

$W = W_c + W_e$ is the sum of weights of all samples, which is fixed in current iteration. We'll choose the classifier with lowest $W_e(\alpha_m > 0)$ or highest $W_e(\alpha_m < 0)$.

What is the optimal α_m for h_m?

Having picked the m-th classifier, we need to determine its weight α_m

$$\frac{\partial \mathcal{L}_m}{\partial \alpha_m} = -W_c e^{-\alpha_m} + W_e e^{\alpha_m} \tag{7}$$

Setting the partial derivative to zero and rescaling it by e^{α_m}

$$-W_c + W_e e^{2\alpha_m} = 0 \tag{8}$$

The optimal α_m is thus

$$\alpha_m = \frac{1}{2} log(\frac{W_c}{W_e}) = \frac{1}{2} log(\frac{W - W_e}{W_e}) = \frac{1}{2} log(\frac{1 - \epsilon_m}{\epsilon_m}) \tag{9}$$

where $\epsilon_m = W_c / W_e$ is the rate of error given the weights of all samples.

Reweighting

Reweighting formula

$$w_i^{(m+1)} = \frac{w_i^{(m)} e^{-y_i \alpha_m h_m(x_i)}}{\mathcal{Z}_m} = \frac{e^{-y_i \sum_{q=1}^{m-1} \alpha_q h_q(x_i)} \cdot e^{-y_i \alpha_m h_m(x_i)}}{N \prod_{q=1}^{m} \mathcal{Z}_q} \tag{10}$$

Effect on training set

Increase the weight of wrongly classified examples by e^{α_m};
Decrease the weight of correctly classified examples by $e^{-\alpha_m}$;

Effect on current classifier h_m

Weighted error of h_m in next iteration is $1/2$. Proof:

$$\frac{\sum_{i:y_i \neq h_m(x_i)} w_i^{(m+1)}}{\sum_{i:y_i = h_m(x_i)} w_i^{(m+1)}} = \frac{\sum_{i:y_i \neq h_m(x_i)} e^{\alpha_m}}{\sum_{i:y_i = h_m(x_i)} e^{-\alpha_m}} = \frac{e^{2\alpha_m} \epsilon_m}{1 - \epsilon_m} = 1 \tag{11}$$

Algorithm of AdaBoost

Input: Training set $\mathcal{S} = \{(x_1, y_1), \cdots, (x_N, y_N) | x_i \in \mathbf{X}, y_i \in \{-1, 1\}\}$

Input: Number of iterations T, threshold β;

1 Initialize weights $w_i^{(1)} = 1/N$ for $i = 1, 2, \cdots, N$;

2 **for** $m = 1, 2, \cdots, T$ **do** `// ` $\mathcal{H}$ `is the family of weak classifiers`

3 $h_m = \underset{h_m \in \mathcal{H}}{arg\max} |0.5 - \epsilon_m|$,where $\epsilon_m = \sum_{i=1}^{N} w_i^t I(y_i \neq h_t(x_i))$;

4 If $|0.5 - \epsilon_m| \leq \beta$ break;

5 Choose $\alpha_m \in \mathbb{R}$,typically $\alpha_m = \frac{1}{2} log(\frac{1-\epsilon_m}{\epsilon_m})$;

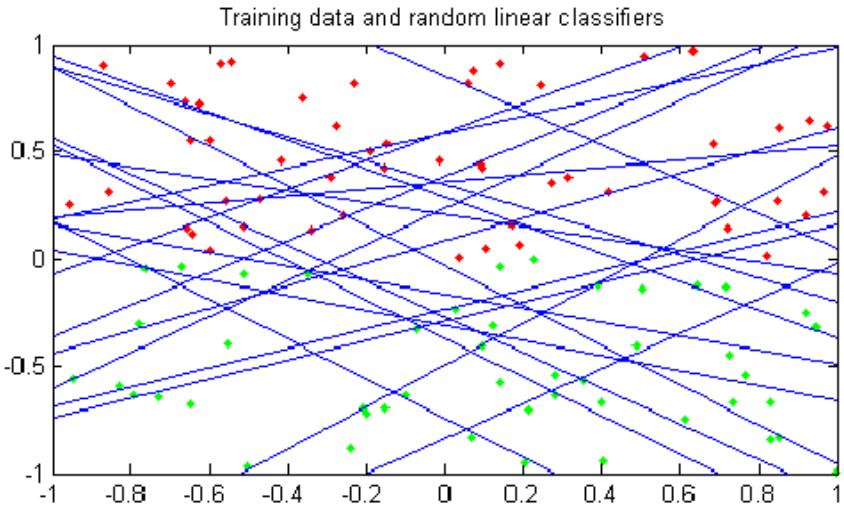
6 Update weights for each sample $w_i^{(m+1)} = \frac{w_i^{(m)} e^{-y_i \alpha_m h_m(x_i)}}{\mathcal{Z}_m}$,where $\mathcal{Z}_m$ is the normalization factor;

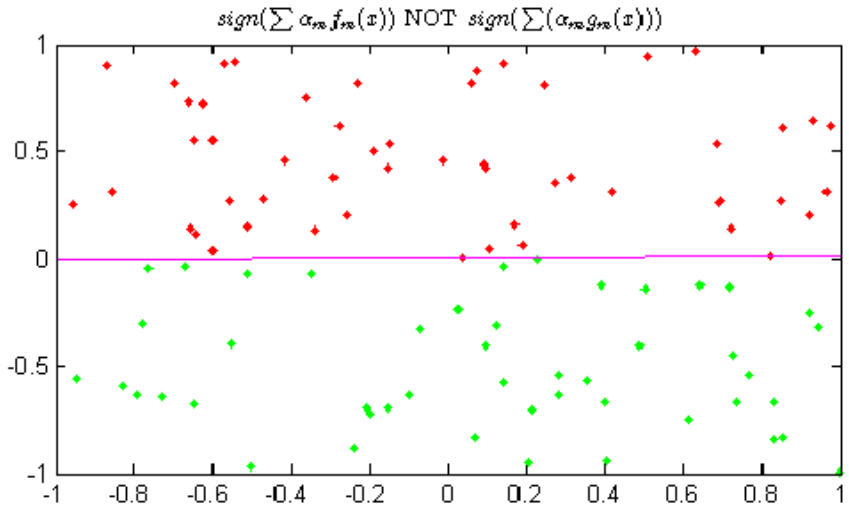
7 **end**

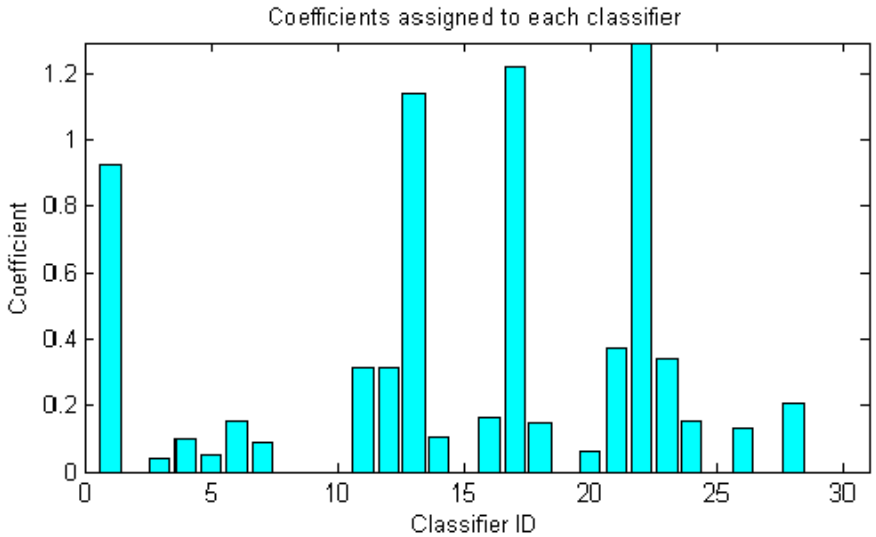
8 **return** the final strong classifier:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{12}$$

AdaBoost Experiments







Analysing Training Error I

Normalization constant at t-th iteration

$$\begin{aligned} \mathcal{Z}_q &= \sum_{i=1}^{N} w_i^{(q)} \exp(-y_i \alpha_i h_m(x_i)) \\ &= \sum_{i:y_i \neq h_m(x_i)} w_i^{(q)} \exp(-y_i \alpha_i h_m(x_i)) \\ &\quad + \sum_{i:y_i = h_m(x_i)} w_i^{(q)} \exp(-y_i \alpha_i h_m(x_i)) \\ &= \epsilon_q \exp(-\alpha_q) + (1-\epsilon_q) \exp(\alpha_q) \\ &= 2\sqrt{\epsilon_q(1-\epsilon_q)} \\ &= \sqrt{1-4\gamma_q^2} \leq \exp(-2\gamma_q^2) \end{aligned} \tag{13}$$

where $\gamma_q = 1/2 - \epsilon_q \in [-1/2, 1/2]$ measures how much better than random is h_t's performance.

Weight for each sample after including m-th classifier into consideration

$$w_i^{(m+1)} = \frac{\exp(-y_i \sum_{q=1}^{m} \alpha_q h_q(x_i))}{N \prod_{q=1}^{m} \mathcal{Z}_q} = \frac{\exp(-y_i f_m(x_i))}{N \prod_{q=1}^{m} \mathcal{Z}_q} \tag{14}$$

Analysing Training Error II

$$h_m(x_i) \neq y_i \Rightarrow y_i f_m(x_i) < 0 \Rightarrow \exp(-y_i f_m(x_i)) > 1 \tag{15}$$

$$h_m(x_i) = y_i \Rightarrow y_i f_m(x_i) > 0 \Rightarrow 0 < \exp(-y_i f_m(x_i)) < 1 \tag{16}$$

Upper bound of training error

$$\begin{aligned} TrainingError &= \frac{1}{N}\sum_{i=1}^{N} I\{y_i \neq h_m(x_i)\} \\ &\leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i f_m(x_i)) \\ &= \sum_{i=1}^{N} w_i^{(m+1)} \prod_{q=1}^{m} \mathcal{Z}_q \\ &= \prod_{q=1}^{m} \mathcal{Z}_q \\ &= \prod_{q=1}^{m} \sqrt{1-4\gamma_q^2} \\ &\leq \exp(-2\sum_{q=1}^{m} \gamma_q^2) \end{aligned} \tag{17}$$

Therefore,if each as long as each weak classifier is slightly better or worse than random $\gamma_q > 0$,then training error drops fast.