

- But:
 - combiner des classifieurs "faibles" h_1, h_2, \dots qui sont à peine mieux qu'une décision aléatoire
 - dans une façon itérative
- Exemples de classifieurs de base:
 - petits arbres de décisions
 - arbres d'une feuille: **decision stumps**
 - réseaux de neurones avec quelques unités cachées

- Idée 1: pondération w_1, \dots, w_n des points d'entraînement
 - attribuer les plus grands poids aux points "difficiles"
 - si $h_t(\mathbf{x}_j) = y_j$ alors $w_j \downarrow$
 - si $h_t(\mathbf{x}_j) \neq y_j$ alors $w_j \uparrow$
- Idée 2: pondération c_1, \dots, c_T des classifieurs de base
 - le poids de h_t est monotone décroissant en l'erreur de h_t

- Démarche
 - entraîner h_t sur l'ensemble d'entraînement pondéré par $\text{BASE}(D_n, \mathbf{w})$
 - calculer c_t , le poids de h_t
 - repondérer les points (calculer w_1, \dots, w_n)
- Si h_t ne peut pas gérer les points pondérés:
 - re-échantillonner selon la distribution w_1, \dots, w_n

```

ADABOOST( $D_n, \text{BASE}(D_n, \mathbf{w}), T$ )
1   $\mathbf{w} \leftarrow (1/n, \dots, 1/n)$       ▷ poids initiaux
2  pour  $t \leftarrow 1$  à  $T$ 
3     $h_t \leftarrow \text{BASE}(D_n, \mathbf{w})$     ▷ hypothèse de base
4     $\epsilon_t \leftarrow \sum_{h_t(\mathbf{x}_i) \neq y_i} w_i$   ▷ erreur pondérée
5    si  $\epsilon_t \geq 1/2$  alors
6      retourner  $f_{t-1}(\cdot) = \frac{\sum_{j=1}^{t-1} c_j h_j}{\sum_{j=1}^{t-1} c_j}$ 
7     $c_t \leftarrow \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$   ▷ poids de  $h_t$ 
8    pour  $i \leftarrow 1$  à  $n$               ▷ repondération des points
9      si  $h_t(\mathbf{x}_i) \neq y_i$  alors
10        $w_i \leftarrow \frac{w_i}{2\epsilon_t}$ 
11     sinon
12        $w_i \leftarrow \frac{w_i}{2(1-\epsilon_t)}$ 
13  retourner  $f_T(\cdot) = \frac{\sum_{t=1}^T c_t h_t(\cdot)}{\sum_{t=1}^T c_t}$ 
    
```

- Observations:
 - $\sum_{i=1}^n w_i = 1$
 - $w_i^{(t+1)} = \frac{w_i^{(t)} e^{-c_t h_t(\mathbf{x}_i) y_i}}{2 \sqrt{\epsilon_t (1 - \epsilon_t)}} = \dots = \frac{1}{n \prod_{j=1}^t 2 \sqrt{\epsilon_j (1 - \epsilon_j)}} e^{-\sum_{j=1}^t c_j h_j(\mathbf{x}_i)}$
- Marge: $\gamma_i = f(\mathbf{x}_i) y_i$
 - $w_i^{(t+1)} = \frac{1}{n} \left(\prod_{j=1}^t 2 \sqrt{\epsilon_j^{1-\gamma_i} (1 - \epsilon_j)^{1+\gamma_i}} \right)^{-1}$
 - $-1 \leq \gamma_i \leq 1$
 - $\gamma_i < 0$: \mathbf{x}_i est mal classifié

- Théorème de convergence
 - $\frac{1}{n} \sum_{i=1}^n I_{\{f(\mathbf{x}_i) y_i < \gamma\}} \leq 2^T \prod_{t=1}^T \sqrt{\epsilon_t^{1-\gamma} (1 - \epsilon_t)^{1+\gamma}}$
 - soit $\gamma = 0$: $\sum_{i=1}^n I_{\{f(\mathbf{x}_i) y_i < 0\}} = \widehat{R}(f, D_n)$ - erreur d'entraînement
 - soit $\epsilon_t \leq \frac{1}{2} - \delta$: h_t est un peu mieux qu'une décision aléatoire
 - $\widehat{R}(f, D_n) \leq e^{-2T\delta^2}$

• Anyboost

- descente de gradient dans l'espace des fonctions de base

- fonctions de base: $\mathcal{H} = \{h : \mathbb{R}^d \rightarrow \{-1, 1\}\}$

- fonctions de décision = combinaison convexe sur \mathcal{H}

$$\mathcal{F} = \left\{ f : f(\cdot) = \frac{\sum_{i=1}^T c_i h_i(\cdot)}{\sum_{i=1}^T c_i}, h_i \in \mathcal{H} \right\}$$

- fonctions de coût de marge:

$$C(f) = \frac{1}{n} \sum_{i=1}^n C(f(\mathbf{x}_i); y_i) = \frac{1}{n} \sum_{i=1}^n C(\gamma_i)$$

• Anyboost

- descente de gradient dans l'espace des fonctions de base:

$$C(f+h) \simeq C(f) - \frac{1}{n} \sum_{i=1}^n -C'(\gamma_i) h(\mathbf{x}_i) y_i$$

- soit $w_i = \frac{-C'(\gamma_i)}{\sum_{i=1}^n -C'(\gamma_i)}$

- minimiser $C(f+h) \equiv$ maximiser $\sum_{i=1}^n w_i h(\mathbf{x}_i) y_i \equiv$ minimiser $\sum_{i: h(\mathbf{x}_i) \neq y_i} w_i$

- arrêter si $\sum_{i: h(\mathbf{x}_i) \neq y_i} w_i \geq 1/2$

• Autres méthodes

Algorithme	fonction de coût	taille du pas
AdaBoost	$e^{-\gamma}$	optimisation en ligne
ARC-X4	$(1-\gamma)^5$	$1/t$
LogitBoost	$\ln(1+e^{-\gamma})$	Newton-Raphson

• Fonctions de coût "sigmoïde"

- théoriquement motivés
- convergence lente en pratique

• Exemples:

- fonction en escalier: $C(f) = \frac{1}{n} \sum_{i=1}^n I_{\{\gamma_i < 0\}} = \widehat{R}(f, D_n)$

- fonction quadratique: $C(f) = \frac{1}{n} \sum_{i=1}^n (\gamma_i - 1)^2 = \frac{1}{n} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2$

- fonction exponentielle: $C(f) = \frac{1}{n} \sum_{i=1}^n e^{-\gamma_i}$

• Anyboost

- trouver c_i : optimisation en ligne:

$$c_i = \arg \min_c \sum_{i=1}^n C(\gamma_i + y_i c h_i(\mathbf{x}_i))$$

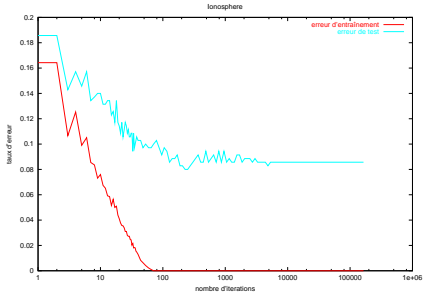
• Adaboost:

- $C(\gamma) = e^{-\gamma}$
- optimisation en ligne

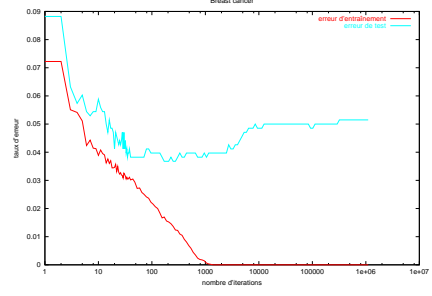
• Expériences

- decision stumps, 4 données de UCI, test croisé en 10 blocs
- l'erreur de test diminue même après que l'erreur d'entraînement devienne 0

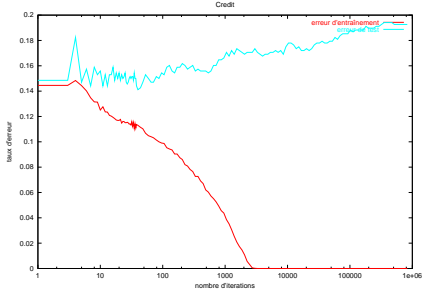
• Pas beaucoup d'overfitting



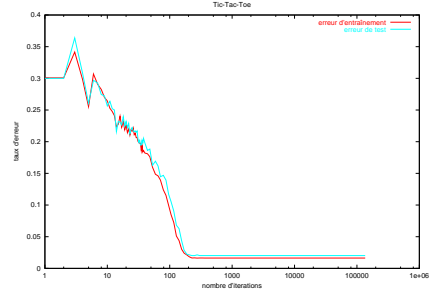
• Plus d'overfitting



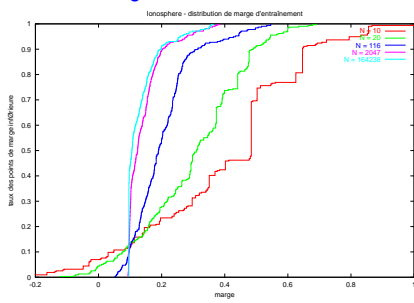
• Beaucoup d'overfitting



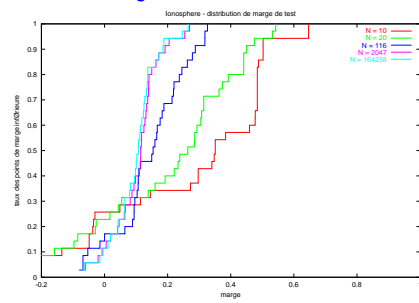
• L'erreur d'entraînement asymptotique n'est pas 0



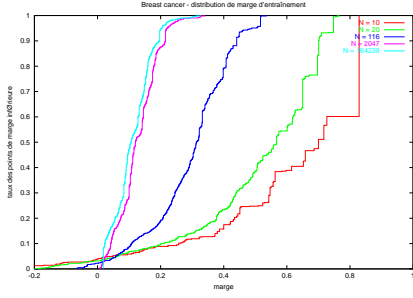
• Distribution de marge d'entraînement



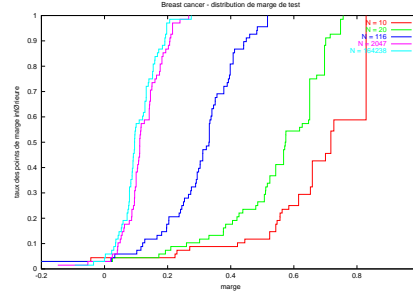
• Distribution de marge de test



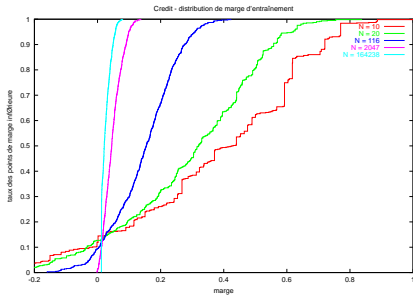
• Distribution de marge d'entraînement



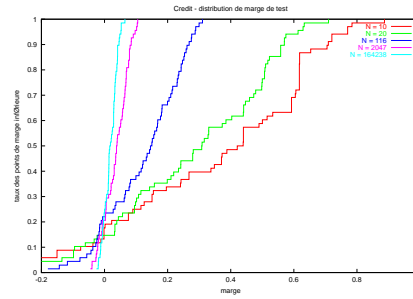
• Distribution de marge de test



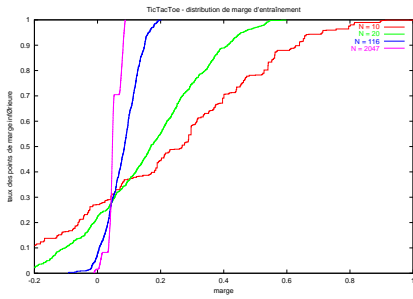
• Distribution de marge d'entraînement



• Distribution de marge de test



• Distribution de marge d'entraînement



• Distribution de marge de test

