

Boosting

JÉRÉMIE MARY

équipe TAO
LRI

30 janvier 2006



Plan

- 1 Introduction
- 2 Historique
- 3 Principes de base d'Adaboost
- 4 Autres façons de voir Adaboost
- 5 Autres expériences et extensions
- 6 Références



Comment puis-je vous aider ?

Objectif [Gorin & all]

Identifier et catégoriser des requêtes faites par téléphone.

- Bonjour, je voudrais accepter un appel en PCV (Réception)
- Je dois passer un appel, mais je souhaite que la facture soit faite au nom de ma société (FacturationAutrePersonne)
- Oui, je souhaite utiliser ma mastercard pour passer l'appel (AppelParCarte)
- J'ai fait un faux numéro, je souhaite que celui-ci ne figure pas sur la facture (CreditFacturation)

Observations

- 1 Il est **facile** de produire des règles qui sont **"souvent" justes**.
 - SI apparaît le mot "mastercard" ALORS prédire AppelParCarte
- 2 **Difficile** d'en avoir qui soient justes tout le temps.



Approche par Boosting

- 1 Écrire un **programme de prise de décision** permettant de conclure a peu près correctement ;
- 2 Appliquer ce programme à un **échantillon des exemples** d'apprentissage ;
- 3 Trouver une **première règle** de décision ;
- 4 Appliquer le programme à un **second échantillon des exemples** d'apprentissage ;
- 5 Trouver une **seconde règle** de décision ;
- 6 Répéter ces opérations T fois.



En pratique

Deux soucis :

- 1 Comment **choisir les échantillons** d'apprentissage ?
 - On se concentre sur les exemples les plus difficiles (ceux qui sont mal classés par les règles précédentes)
- 2 Comment **combiner les différentes règles** de décision apprises ?
 - On utilise un système de vote pondéré des différentes règles



En bref

Boosting

Le boosting est une technique générale qui permet de transformer des règles de décisions grossières en une règle de décision très précise.

Techniquement

- On doit disposer d'un algorithme de construction de règles légèrement meilleur que le hasard (disons 55% de réponses correctes)
- Si l'on dispose de suffisamment de données, il sera probablement possible de construire une règle de décision dont la précision sera très élevée (disons 99%).



Pose du problème

Formalisme PAC

- 1 "strong" PAC Algorithm
 - Pour toute distribution
 - $\forall \epsilon > 0, \delta > 0$
 - Étant donné des exemples tirés aléatoirement
 - On trouve un classifieur ayant une erreur $\leq \epsilon$ avec une probabilité $1 - \delta$
- 2 "weak" PAC Algorithm
 - Pareil mais seulement pour $\epsilon \geq \frac{1}{2} - \gamma$

weak \Rightarrow strong ? [Kearns & Valiant '88]



Naissance du Boosting

Historique

- 1 [Schapire '89] Premier algorithme de boosting
 - Appelle un weak learner trois fois en modifiant la distribution des exemples
 - Gains théoriques démontrés
 - Gains pratiques faibles
- 2 [Freund '90] Algorithme "optimal" par votes pondérés
- 3 [Drucker & al '92] Premières mises en œuvres réelles, gros inconvénients pratiques



Adaboost

Freund & Schapire '95

Enfin ça marche...

Applications d'Adaboost :

[Tieu & Viola '00] [Drucker & Cortes '96] [Abney, Schapire&Singer '99] [Walker, Rambow & Rogati '01] [Jackson & Craven '96] [Haruno, Shirai&Ooyama '99] [Rochery, Schapire, Rahim&Gupta'01] [Freund&Schapire '96] [Cohen&Singer' 99] [Merler, Furlanello, Larcher &Sboner'01] [Quinlan '96] [Dietterich '00] [Di Fabbriizio, Dutton, Gupta et al.'02] [Breiman '96] [Schapire & Singer '00] [Qu, Adam, Yasui et al. '02] [Maclin & Opitz '97] [Collins '00] ' [Tur, Schapire & Hakkani-Tur '03] [Bauer & Kohavi '97] [Escudero, Marquez & Rigau '00] [Viola&Jones '04] [Schwenk & Bengio '98] [Iyer, Lewis, Schapire et al. '00] [Middendorf, Kundaje, Wiggins et al. '04] [Onoda, Ratsch&Muller '00] [Schapire, Singer&Singhal '98]...

Développements et travaux théoriques :

[Koltchinskii,Panchenko&Lozano '01] [Breiman '98, '99] [Duffy&Helmbold '99, '02] [Collins, Schapire&Singer '02] [Schapire, Freund, Bartlett&Lee '98] [Freund&Mason '99] [Demiriz, Bennett&Shawe-Taylor '02] [Grove&Schuurmans '98] [Ridgeway, Madigan&Richardson '99] [Lebanon&Lafferty '02] [Mason, Bartlett&Baxter '98] [Kivinen&Warmuth '99] [Wyner '02] [Schapire&Singer '99] [Friedman, Hastie&Tibshirani '00] [Rudin, Daubechies&Schapire '03] [Cohen&Singer '99] [Ratsch, Onoda&Muller '00] [Jiang '04] [Freund&Mason '99] [Ratsch, Warmuth, Mika et al. '00] [Lugosi&Vayatis '04] [Domingo&Watanabe '99] [Allwein, Schapire&Singer '00] [Friedman '01] [Zhang '04] [Mason, Baxter, Bartlett&Frean '99, '00]...



Description formelle du boosting

On dispose

- 1 d'une base d'exemples $(x_1, y_1), \dots, (x_n, y_n)$
- 2 On suppose $y_i \in \{-1, 1\}$
- 3 Pour $t = 1 \dots T$
 - Construire une distribution D_t sur $1 \dots m$
 - Trouver un weak classifier :

$$h_t : X \rightarrow \{-1, 1\}$$

tel que pour ε_t petit,

$$\varepsilon_t = Pr_{D_t}(h_t(x_i) \neq y_i)$$

- 4 Combiner les h_t pour produire le classifieur final H .



Adaboost

Construction de D_t

- 1 $D_1(i) = 1/m$
- 2 Pour D_t et h_t donnés :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(i))$$

avec

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} > 0$$

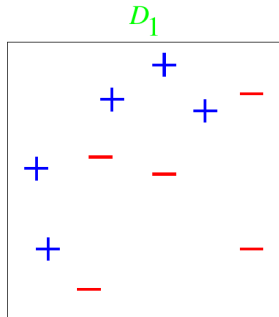
et Z_t constante de normalisation

Construction de H

$$H(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$$



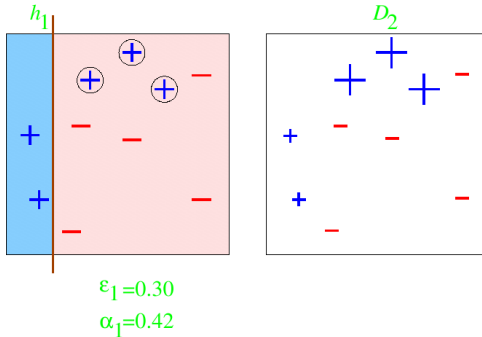
Exemple



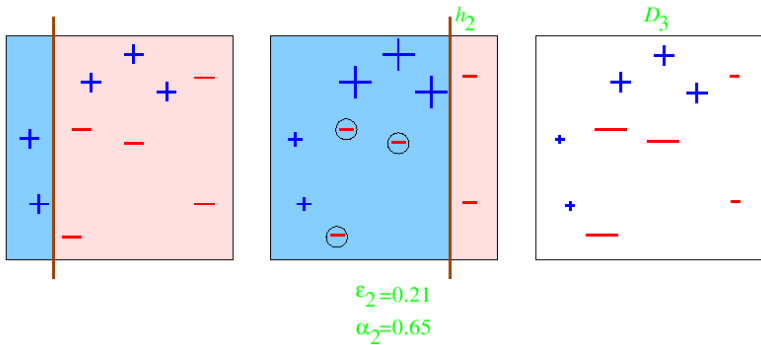
Les weak learners sont les hyperplans verticaux ou horizontaux



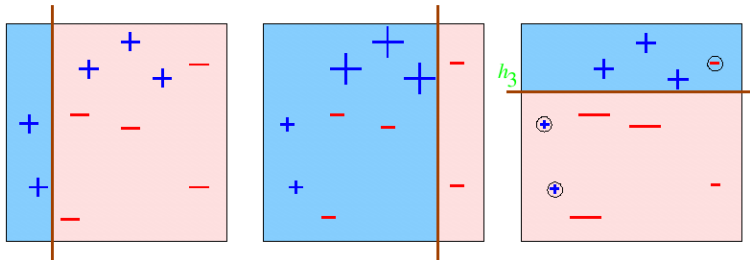
Étape 1



Étape 2



Étape 3

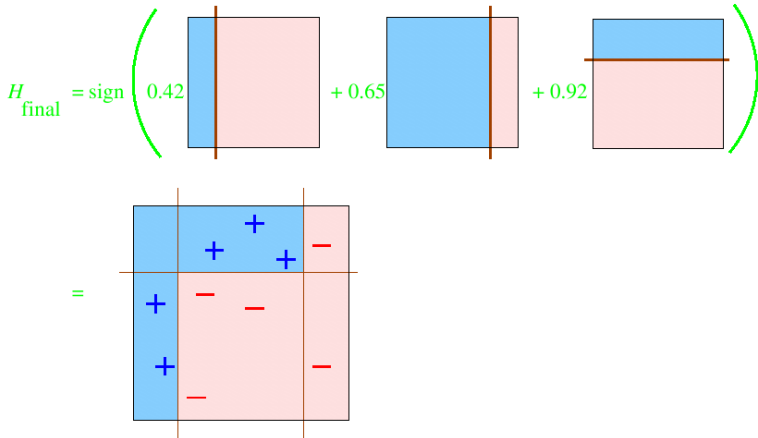


$\epsilon_3=0.14$

$\alpha_3=0.92$



Résultat



Résultats théoriques

Théorème

- ① toto
- ② On note $\varepsilon_t = 1 - \gamma_t$ alors :

$$\begin{aligned}
 \text{erreur finale de}(H) &\leq \prod_t 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \\
 &= \prod_t \sqrt{1 - 4\gamma_t^2} \\
 &\leq \exp\left(-2 \sum_t \gamma_t^2\right)
 \end{aligned}$$

Conséquences

- Si $\forall t \gamma_t \geq \gamma > 0$ alors l'erreur finale est majorée par $\exp -2\gamma^2 T$



Preuve

- 1 soit $f(x) = \sum_t \alpha_t(x) \Rightarrow H(x) = \text{sign}(f(x))$
- 2 Première étape :

$$\begin{aligned} D_{\text{final}}(i) &= \frac{1}{m} \frac{\exp(-y_i \sum_t \alpha_t h_t(x_i))}{\prod_t Z_t} \\ &= \frac{1}{m} \frac{\exp(-y_i f(x_i))}{\prod_t} \end{aligned}$$



Preuve (suite)

L'erreur sur la base d'apprentissage est inférieure à $\prod_t Z_t$

$$\begin{aligned} \text{erreur empirique de } H &= \frac{1}{m} \sum_i \mathbb{1}_{H(x_i) \neq y_i} \\ &= \frac{1}{m} \sum_i \mathbb{1}_{y_i f(x_i) < 0} \\ &\leq \frac{1}{m} \sum_i \exp(-y_i f(x_i)) \\ &= \sum_i D_{\text{final}}(i) \prod_t Z_t \\ &= \prod_t Z_t \end{aligned}$$



Preuve (fin)

Reste à montrer $Z_t = 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}$

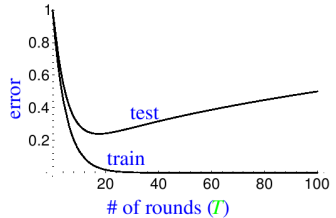
$$\begin{aligned} Z_t &= \sum_i D_t(i) \exp(-\alpha_t y_i f(x_i)) \\ &= \sum_{i: y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i: y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} \\ &= \varepsilon_t e^{\alpha_t} + (1 - \varepsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\varepsilon_t(1 - \varepsilon_t)} \end{aligned}$$



Comportement de l'erreur en généralisation

Comportement classique

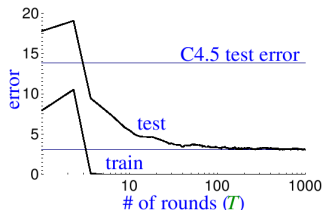
- 1 L'erreur empirique tend vers 0
- 2 L'erreur en test passe par un minimum puis remonte.
 - H devient trop complexe (overfitting)
 - Principe du rasoir d'Occam, mais difficile de savoir quand s'arrêter



Comportement de l'erreur en généralisation

Comportement réel pour le Boosting

- 1 L'erreur en généralisation ne remonte pas.
- 2 Elle continue à descendre alors que l'erreur empirique est déjà nulle...
- 3 Le rasoir d'Occam est sérieusement émoussé : H final contient plus de 200000 nœuds.



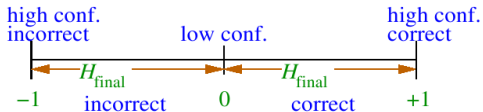
	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1



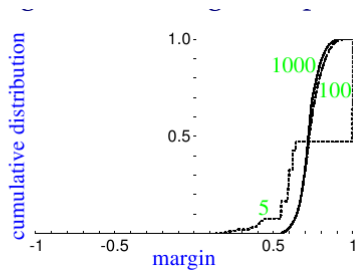
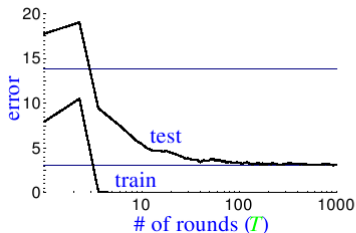
Essai d'explication : théorie des marges

Principe

- 1 L'erreur empirique ne mesure que si les **classifications sont correctes**
- 2 Il faut aussi prendre en compte la **confiance** dans le résultat
- 3 Le résultat étant un vote, on s'intéresse à la
fraction votant correctement - fraction ne votant pas correctement



Piste expérimentale



	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55



Piste théorique

Théorème

Grandes marges \Rightarrow Meilleure borne en généralisation (indép. de T)

- Idée : Si toutes les marges sont grandes, il est possible de prédire le résultat avec un classeur plus petit (comme pour les sondages).

Théorème

Le boosting a **tendance à maximiser les marges de la base d'apprentissage** (sous l'hypothèse du weak learning).

- Idée : similaire à la démo pour l'erreur empirique

On peut donc penser que **le classeur complexe obtenu par le boosting est en réalité proche d'un plus simple possédant de bonnes marges**, ce qui réduit l'erreur en généralisation.



Plus précisément

- 1 Avec grande probabilité, $\forall \theta > 0$

$$\text{erreur en généralisation} \leq \hat{P}(\text{marge} \leq \theta) + O\left(\frac{\sqrt{d/m}}{\theta}\right)$$

- m est le nombre d'exemples d'apprentissage
 - d est la "complexité" des weak learners
- 2 $\hat{P}(\text{marge} \leq \theta) \rightarrow 0$ exponentiellement vite en T si $\gamma_t > \theta$



Théorie des jeux

- On considère un jeu défini par une matrice M

	Rock	Paper	Scissors
Rock	1/2	1	0
Paper	0	1/2	1
Scissors	1	0	1/2

- Le joueur "ligne" choisit la ligne i
- Le joueur "colonne" choisit simultanément la colonne j
- L'objectif du joueur "ligne" est de minimiser la perte $M(i, j)$
- Si on joue aléatoirement selon une distribution P et Q sur les lignes et les colonnes, la perte est :

$$\sum_{i,j} = P(i)M(i, j)Q(j) = P^t M \equiv M(P, Q)$$



Minmax

théorème (Neumann's)

$$\begin{aligned} \min_P \max_Q M(P, Q) &= \max_Q \min_P M(P, Q) \\ &= v \\ &= \text{"valeur" du jeu } M \end{aligned}$$

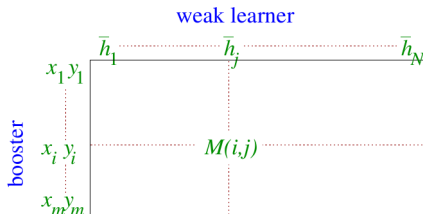
Autrement dit :

- $v = \min \max$ signifie que la stratégie du joueur "ligne" est telle que pour toute stratégie Q la perte est inférieure ou égale à v
- $v = \max \min$ signifie que la stratégie est optimale dans le sens où le joueur "colonne" à une stratégie Q^* telle que pour toute stratégie du joueur colonne, le perte est supérieure ou égale à v



Le jeu du Boosting

- Soit $\{\bar{h}_1, \dots, \bar{h}_n\}$, l'espace de tous les weak learners
- joueur "ligne" : booster
- joueur colonne : weak learner
- Matrice M :
 - ligne : exemple (x_i, y_i)
 - colonne : weak learners
 - $M(i, j) = \mathbb{1}_{y_i = \bar{h}_j(x_i)}$



Minmax et Boosting

- Si pour toute distribution sur les exemples, $\exists h$ avec précision $\geq \frac{1}{2} - \gamma$
- Alors ;

$$\min_P \max_h M(P, h) \geq \frac{1}{2} - \gamma$$

- Et donc

$$\max_Q \min_i M(i, Q) \geq \frac{1}{2} - \gamma > \frac{1}{2}$$

- Ce qui signifie qu'il **existe un vote pondéré** des classifieurs qui **classe correctement tous les exemples** avec une marge 2γ .
- La marge optimale est l'équivalent de la valeur du jeu.



Adaboost et la théorie des jeux

- Adaboost est un cas particulier d'un algorithme général pour résoudre les jeux par répétitions
- On peut montrer que
 - La distribution sur les exemples converge vers une approximation de la stratégie **min max** pour le jeu du boosting.
 - Les poids sur les weak learners convergent vers une approximation de la stratégie **max min**.
- En utilisant plusieurs algorithmes de jeu, il est possible de réaliser un algorithme d'apprentissage en-ligne (par vote pondéré)

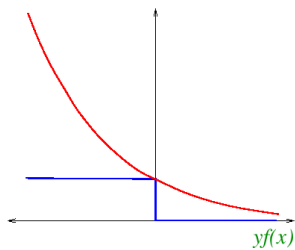


Adaboost et optimisation

- Les algorithmes d'apprentissage minimisent souvent une fonction de perte (ex : régression).
- Adaboost minimise :

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i \sum_t \alpha_t h_t(x))$$

- Adaboost est une approche gloutonne sur α_t et h_t
- L'exponentielle est une borne sup de l'erreur en classification ; c'est elle qui est minimisée



Descente par coordonnées

- $\{\bar{h}_1, \dots, \bar{h}_N\}$ espace de tous les weak learners
- On veut trouver $\lambda_1, \dots, \lambda_N$ minimisant

$$L(\lambda_1, \dots, \lambda_N) = \sum_i \exp(-y_i \sum_j \lambda_j \bar{h}_j(x_i))$$

- Adaboost réalise une descente par coordonnées sur ce problème d'optimisation :
 - Au départ $\lambda_j = 0$
 - à chaque étape on choisit une coordonnée λ_j (correspondant à h_t), et on la met à jour en l'augmentant de α_t
 - La mise à jour est choisie pour obtenir la plus grande décroissance possible de la fonction de perte.
- Technique réputée efficace pour réaliser une optimisation sur un grand espace de fonctions.



Descente fonctionnelle de gradient

- On veut minimiser

$$L(f) = L(f(x_1), \dots, f(x_m)) = \sum_i \exp(-y_i f(x_i))$$

- Supposons que l'on ait un \bar{f} approché de la solution et que l'on souhaite l'améliorer.
- Par descente de gradient, on a envie de choisir :

$$\bar{f} \leftarrow \bar{f} - \eta \nabla_f L(\bar{f})$$

- Problème : la mise à jour est restreinte par les weak learners disponibles.
- On choisit donc h_t "le plus près" du gradient
- équivalent à Adaboost



Bilan de cette nouvelle vue

- Ca donne des idées pour généraliser Adaboost à d'autres fonctions de pertes
 - fonction carrée
 - Régression logistique
- Problème : Adaboost n'est pas simplement un algorithme de minimisation d'une fonction de perte exponentielle :
 - Les autres algorithmes minimisant la même quantité donnent des résultats minables
 - L'utilisation de cette fonction de perte n'est donc pas la cause profonde qui fait qu'Adaboost fonctionne



Boosting de probabilités conditionnelles [Friedman & all]

- On veut estimer la probabilité que $y = 1$ sachant x
- Adaboost minimise l'estimateur empirique de :

$$E_{x,y} \left(e^{-yf(x)} \right) = E_x \left(P(y = 1/x)e^{-f(x)} + P(y = -1/x)e^{-f(x)} \right)$$

ou x, y sont tirés selon la vraie distribution.

- Le minimum est atteint quand :

$$f(x) = \frac{1}{2} \ln \left(\frac{P(y = 1/x)}{P(y = -1/x)} \right)$$

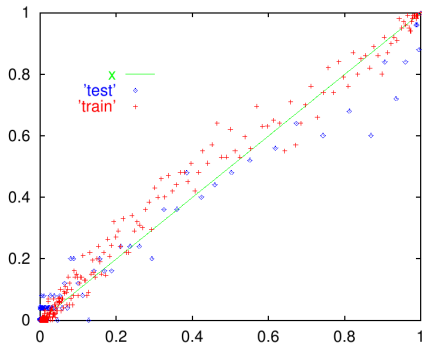
ou

$$P(y = 1/x) = \frac{1}{1 + e^{-2f(x)}}$$

- Ce qui donne un lien pour estimer des probabilités conditionnelles à partir du résultat fourni par Adaboost



Courbe de calibration



- On trie les exemples selon la valeur de f donnée par Adaboost
- On sépare en bases de r exemples
- Pour chaque base on affiche :
 - En x l'estimation moyenne de $P(y = 1)$;
 - En y la fraction d'exemples positifs présente.



Avantages Pratiques d'Adaboost

- 1 rapide
- 2 facile à coder
- 3 pas de paramètres (à par T)
- 4 Peut-être utilisé avec n'importe quel algorithme d'apprentissage
- 5 Pas de connaissance à utiliser sur le weak learner
- 6 Peut-être utilisé avec tout types de données, et a été étendu bien au dela de la classification binaire.



Défauts

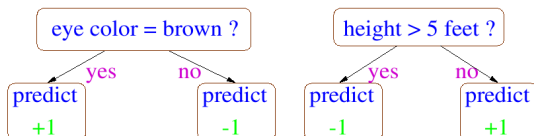
- 1 Les performances dépendent **des données** et des weak learners (**besoin d'instabilité**).
- 2 Comme on s'y attend, Adaboost échoue quand :
 - Les weak learners sont trop complexes : **overfitting**
 - Les weak learners sont trop faibles ($\gamma_t \rightarrow 0$ trop rapidement)
- 3 Epiriquement, Adaboost **supporte très mal un bruit uniforme**.
- 4 Problème avec les **outliers** (croissance exponentielle de leur poids).



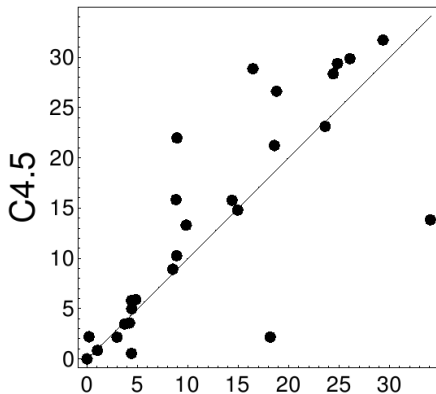
Expériences sur les jeux UCI

On utilise

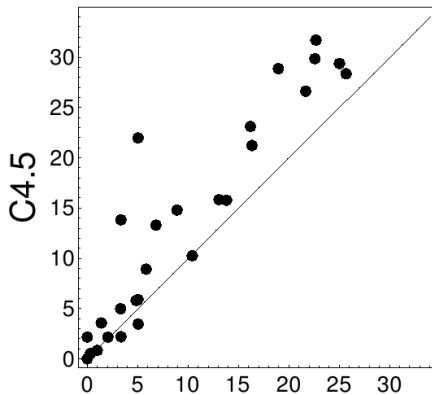
- C4.5
- Des "decision stumps" (règles de décision très simples) :



Expériences sur les jeux UCI



boosting Stumps



boosting C4.5

Problèmes multiclassés - M1

- On suppose que $y = \{1, \dots, k\}$
- Adaptation directe :

$$h_t : X \rightarrow Y$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(\alpha_t \mathbb{1}_{y_i \neq h_t(x_i)} - \alpha_t \mathbb{1}_{y_i = h_t(x_i)})$$

$$H_{final} = \arg \max_{y \in Y} \sum_{t: h_t(x)=y} \alpha_t$$

- On obtient la même borne sur l'erreur si $\forall t : \varepsilon_t < 1/2$
- En pratique **ne fonctionne bien que pour les "strong" weak learners** (ex C4.5)



Réduction d'un problème multiclassé à un problème binaire - MH

- Supposons que les labels possibles soient $\{a, b, c, d, e\}$
- Chaque exemple est remplacé par 5 exemples binaires :

$$x, c \rightarrow \begin{cases} (x, a), -1 \\ (x, b), -1 \\ (x, c), +1 \\ (x, d), -1 \\ (x, e), -1 \end{cases}$$

- On prédit comme label celui ayant reçu le plus de votes pondérés.



Adaboost.MH

- On peut montrer que l'erreur empirique de H_{final} est $\leq \frac{k}{2} \prod Z_t$
- En effet un petit nombre d'erreurs dans les prédictions binaires peut faire changer la classification finale.
- Peut s'étendre immédiatement au **cas multi-labels**.



Réduction par codes de sorties

- Même genre d'idée, mais on utilise un code au lieu des labels :

	π_1	π_2	π_3	π_4
a	-	+	-	+
b	-	+	+	-
c	+	-	-	+
d	+	-	+	+
e	-	+	-	-

$$x, c \rightarrow \begin{cases} (x, \pi_1), +1 \\ (x, \pi_2), -1 \\ (x, \pi_3), -1 \\ (x, \pi_4), +1 \end{cases}$$

- Bornes sur l'erreur empirique **indépendante du nombre de classe**
- Meilleure résistance aux erreurs des classeurs binaires
- MAIS les problèmes binaires obtenus **peuvent se révéler plus difficiles.**

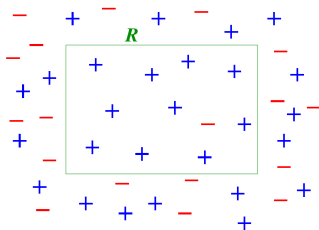


Classement

- Les problème de classement peuvent aussi être réduits à de la classification binaire
- Par exemple, si on veut apprendre à classer des films à partir d'exemples
- On se ramène à des questions de la forme : "A est-il à préférer à B"
- Et on booste...



Problèmes avec des prédictions difficiles



- On voudrait que en dehors de R le classeur ne réponde pas -1 mais plutôt "je ne sais pas"
- Si le weak learner est forcé de répondre -1 la vitesse de convergence décroît dramatiquement (il faudra "supprimer" les erreur qui vient d'être introduite).



Confiance dans les prédictions

- On prend $h_t : X \rightarrow \mathbb{R}$
 - Le signe de $h_t(x)$ est la prédiction ;
 - $|h_t(x)|$ la confiance.
- On conserve le même mécanisme de mise à jour et de combinaison
- **Problème** : Comment choisir α_t et h_t à chaque étape ?



Confiance dans les prédictions

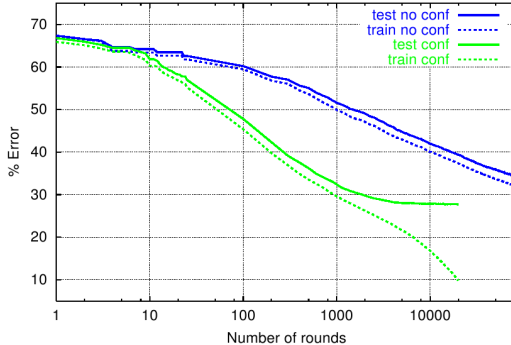
- $\alpha_t h_t$ devraient minimiser

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

- Dans les cas pas trop compliqués, il est assez facile de trouver le weak learner ayant la confiance la plus élevée.
- Sinon...



Gains obtenus avec l'ajout de la confiance



% error	round first reached		speedup
	conf.	no conf.	
40	268	16,938	63.2
35	598	65,292	109.2
30	1,888	>80,000	—



Application : Boosting pour la catégorisation de textes

- weak learners : test sur des n -grams
 - On recherche les paramètres α_t et h_t d'une forme donnée minimisant Z_t
 - Recherche exhaustive...
- données "How may I help you" :
 - 7844 exemples d'entraînement
 - 1000 exemples de test
 - catégories : *AreaCode*, *AttService*, *BillingCredit*, *CallingCard*, *Collect*, *Competitor*, *DialForMe*, *Directory*, *HowToDial*, *PersonToPerson*, *Rate*, *ThirdNumber*, *Time*, *TimeCharge*, *Other*.



Weak Learners

rnd term	AC	AS	BC	CC	CO	CM	DM	DI	HO	PP	RA	3N	TI	TC	OT
1 collect															
2 card															
3 my home															
4 person ? person															
5 code															
6 I															



Weak Learners

rnd term	AC	AS	BC	CC	CO	CM	DM	DI	HO	PP	RA	3N	TI	TC	OT
7 time															
8 wrong number															
9 how															
10 call															
11 seven															
12 trying to															
13 and															



Weak Learners

rnd term	AC	AS	BC	CC	CO	CM	DM	DI	HO	PP	RA	3N	TI	TC	OT	
14 third	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
15 to	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
16 for	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
17 charges	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
18 dial	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
19 just	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■



Outlier

Les exemples avec les poids les plus élevés sont souvent des **Outliers** ou des exemples ambigus

- I'm trying to make a credit card call (Collect)
- hello (Rate)
- yes I'd like to make a long distance collect call please (CallingCard)
- calling card please (Collect)
- yeah I'd like to use my calling card number (Collect)
- can I get a collect call (CallingCard)
- yes I would like to make a long distant telephone call and have the charges billed to another number (CallingCard DialForMe)
- yeah I can not stand it this morning I did oversea call is so bad (BillingCredit)
- yeah special offers going on for long distance (AttService Rate)
- mister allen please william allen (PersonToPerson)
- yes ma'am I I'm trying to make a long distance call to a non dialable point in san miguel philippines (AttService Other)
- yes I like to make a long distance call and charge it to my home phone that's where I'm calling at my home (DialForMe)

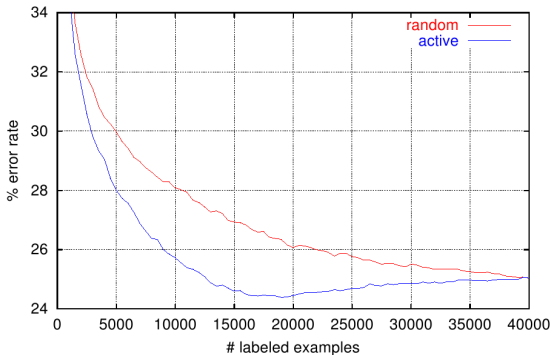


Apprentissage Actif

- Parfois les labels sont chers à obtenir (intervention d'un expert humain)
- L'idée est de se **focaliser sur les exemples pour lesquels on a peu de confiance** (en utilisant $|f(x)|$ comme mesure).
 - 1 On débute avec tous les exemples non labellés
 - 2 Choisir un nombre arbitraire d'exemples au hasard que l'on fait étiquetter (ex : 500)
 - 3 Utiliser le boosting et récupérer f
 - 4 Prendre de nouveaux exemples (ex : 250) ayant un $|f(x)|$ petit et les faire étiquetter
 - 5 Répéter



Résultat

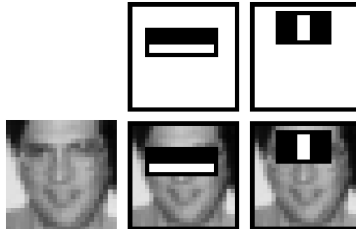


% error	first reached		% label savings
	random	active	
28	11,000	5,500	50
26	22,000	9,500	57
25	40,000	13,000	68



Détection de visages

- But : trouver des visages dans des films ou des photos
- weak learners : rectangles noirs/blancs dans un image



Conclusion

Le boosting est ...

- 1 Basé sur une théorie mathématique
- 2 **un outil efficace expérimentalement**
- 3 souvent **résistant à l'overfitting**
- 4 doté de nombreuses applications

Plusieurs façons de voir le boosting

- Aucune n'est pleinement satisfaisante (mais toutes ont des avantages) ;
- Il reste pas mal de boulot...



Références

- 1 "A Boosting Tutorial" www.cs.princeton.edu/~shapiro
- 2 Ron Meir and Gunnar Ratsch. An Introduction to Boosting and Leveraging. In Advanced Lectures on Machine Learning (LNAI2600), 2003. <http://www.boosting.org/papers/MeiRae03.pdf>

