MAIAD

Vincent Guigue
Laboratoire d'Informatique de Paris 6 (LIP6)
d'après Freund & Schapire

Projet et données

- Le but du projet est de construire des machines de classification automatique sur les données issues de la compétition DEFT 2005 :
  Ces données regroupes deux catégories de texte : des discours de M. Mitterrand et des discours de M. Chirac.
  `http://www.lirmm.fr/~mroche/Recherche/Publications/DEFT2005/presentation_DEFT05.pdf`
  `http://fr.wikipedia.org/wiki/DEFT`
- Après avoir obtenu des premiers résultats à l'aide de l'algorithme du perceptron, vous chercherez à améliorer vos performances à l'aide du boosting.

Déroulement du projet

1. Séance 1 :
   - introduction au boosting et à l'algorithme AdaBoost
   - rappels sur le perceptron
   - base minimum théorique, exercices de TD
   - passage à la pratique
2. Séance 2 :
   - rappels sur le codage de texte en apprentissage
   - présentation des outils mis à disposition
   - mise en route du projet sur de la classification de texte
3. Séance 3 : séance pratique sur les données

Introduction au boosting

Vincent Guigue
Laboratoire d'Informatique de Paris 6 (LIP6)
d'après Freund & Schapire

Philosophie du boosting

- Face à un problème de classification complexe, il est difficile de trouver une règle :
  - générale
  - efficace
- Par contre, il est beaucoup plus facile de trouver une règle :
  - locale (pour une partie des données)
  - raisonnablement efficace (mieux que random...)

Boosting

Il suffit de construire plusieurs règles faibles et de les combiner !

Contrainte : faire une combinaison efficace et sans paramètre dont la convergence est prouvée

How may I help you

but :
classer les types de demande des utilisateurs
(Collect, CallingCard, PersonToPerson, etc.)

- yes I'd like to place a collect call long distance please (Collect)
- operator I need to make a call but I need to bill it to my office (ThirdNumber)
- yes I'd like to place a call on my master card please (CallingCard)
- I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill (BillingCredit)

observation :

- Il est facile de trouver des règles *faibles* valable *le plus souvent* e.g. : SI card apparait ALORS prédire CallingCard
- Il est difficile de trouver une règle *universelle* efficace.

Idée générale

Algorithme

1. Sélectionner un sous ensemble de données
2. Extraire des règles (*faibles*)
3. recommencer T fois

Questions

- Comment sélectionner les sous ensembles ?
- Comment combiner les règles ?

Boosting

Méthode générale pour convertir des règles *faibles* en un classifieur efficace

Idée générale

Algorithme

1. Sélectionner un sous ensemble de données
2. Extraire des règles (*faibles*)
3. recommencer T fois

Questions

- Comment sélectionner les sous ensembles ?
- Comment combiner les règles ?

Boosting

Méthode générale pour convertir des règles *faibles* en un classifieur efficace

- Se concentrer sur les exemples *durs* (mal classés)
- Système de votes pondérés des différentes règles

Historique (très rapide)

- [Kearns & Valiant '88] : *does weak learnability imply strong learnability ?*
- [Schapire '89] : premier algorithme *prouvé*
- [Freund '90]
- [Drucker, Schapire & Simard '92] : montrent l'intérêt expérimental
- [Freund & Schapire '95] : AdaBoost, algorithme de référence
- ... Beaucoup d'expériences, beaucoup de gens, des résultats intéressants

Formalisation

- A partir d'un ensemble d'exemples $\{x_i, y_i\}_{i=1\dots\ell}$, on veut construire une fonction $f : \mathcal{X} \to \{-1, 1\}$, ou $f : \mathcal{X} \to \mathbb{R}$ qui permette de prédire si un nouveau point $x \in \mathcal{X}$ appartienne à la classe -1 ou à la classe 1.
- Notons les hypothèses faibles h_t :

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

Algorithme

- pour t allant de 1 à T
  - Construire le sous ensemble D_t
  - Trouver une hypothèse h_t faisant peu d'erreur sur D_t
- Construire H_{final} à partir des h_t

AdaBoost

Construire D_t

- Init. : $D_1(i) = \frac{1}{\ell}$
- soit D_t et h_t (ϵ_t : erreur de h_t sur D_t)

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

Avec : Z_t cst de normalisation

$\alpha_t = 0.5 \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$, $\quad \epsilon_t = Pr_{D_t}(h_t(x_i) \neq y_i)$

Construire H_{final}

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

Illustration du fonctionnement (1)

Les hypothèses sont de type :
frontières linéaires horizontales ou verticales

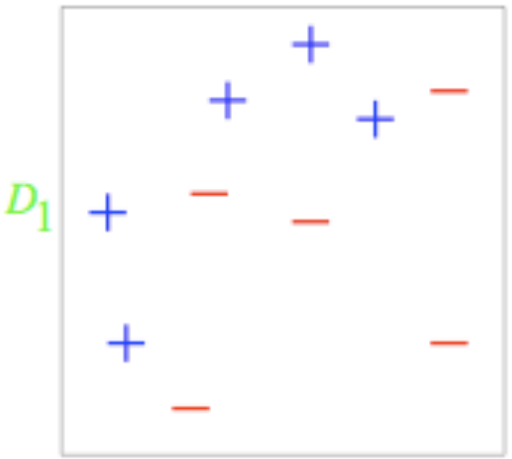
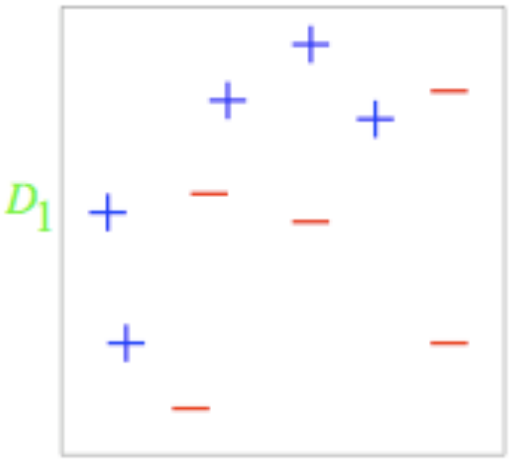
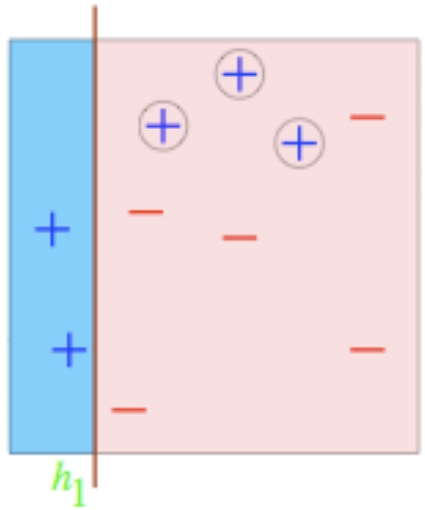


Illustration du fonctionnement (1)

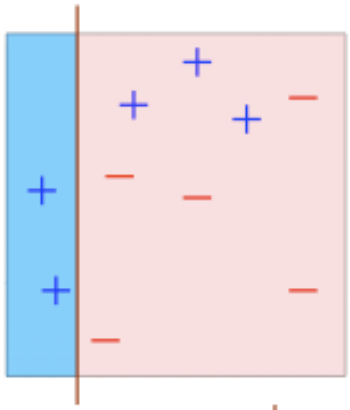
Les hypothèses sont de type :
frontières linéaires horizontales ou verticales

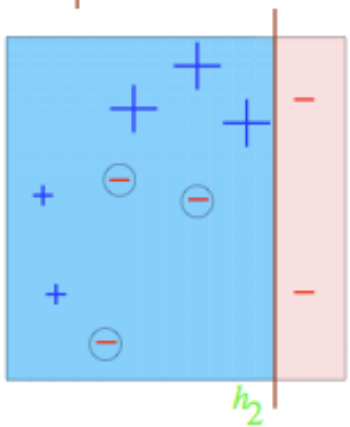




$\varepsilon_1 = 0.30$
$\alpha_1 = 0.42$

Illustration du fonctionnement (2)





ε_2=0.21
α_2=0.65

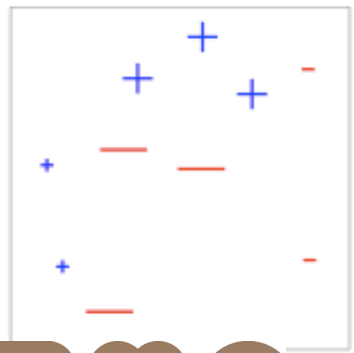


Illustration du fonctionnement (2)

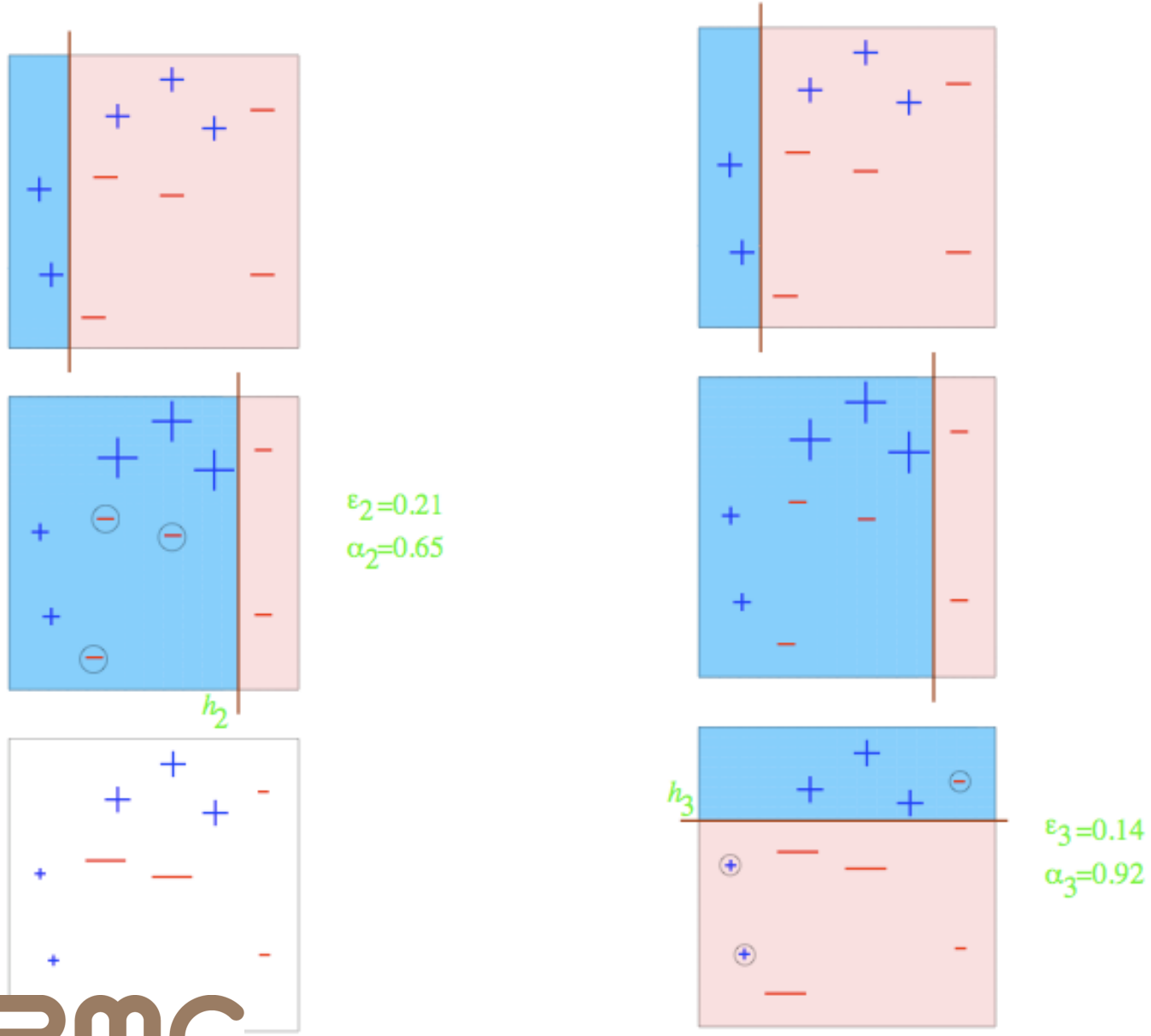
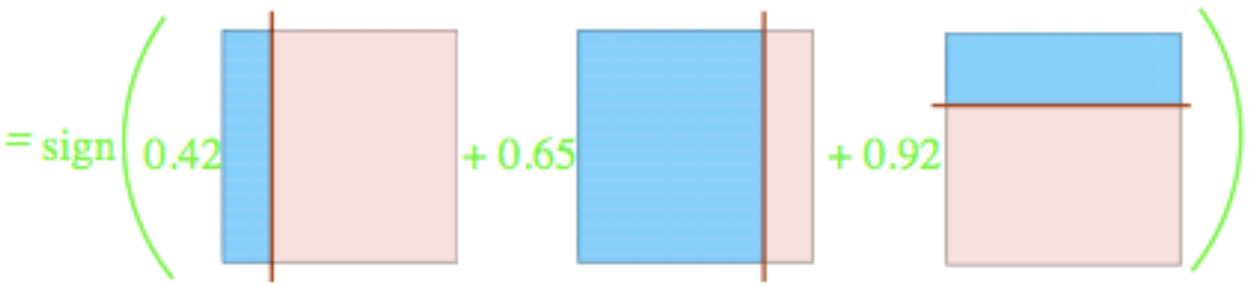
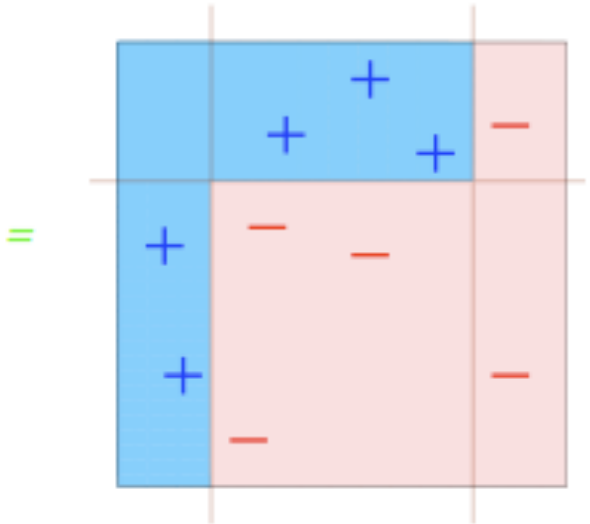


Illustration du fonctionnement (3)

Fusion des hypothèses

H_{final}





Analyse de la méthode

- On peut montrer qu'il existe une combinaison linéaire d'hypothèses classant correctement tous les points d'apprentissage...
- $\Rightarrow$ risque de sur apprentissage
  - Dans la pratique, prendre des hypothèses *simples* régularise l'apprentissage
  - Il faut régler le nombre d'itération d'apprentissage (T n'a pas besoin d'être connu dès le départ)
    $\Rightarrow$ des critères existent pour régler T
  - Dans tous les cas, les *outliers* destabilisent rapidement la solution, via des poids très importants
    $\Rightarrow$ des heuristiques proposent de borner l'influence des points ou de détecter/éliminer les *outliers*

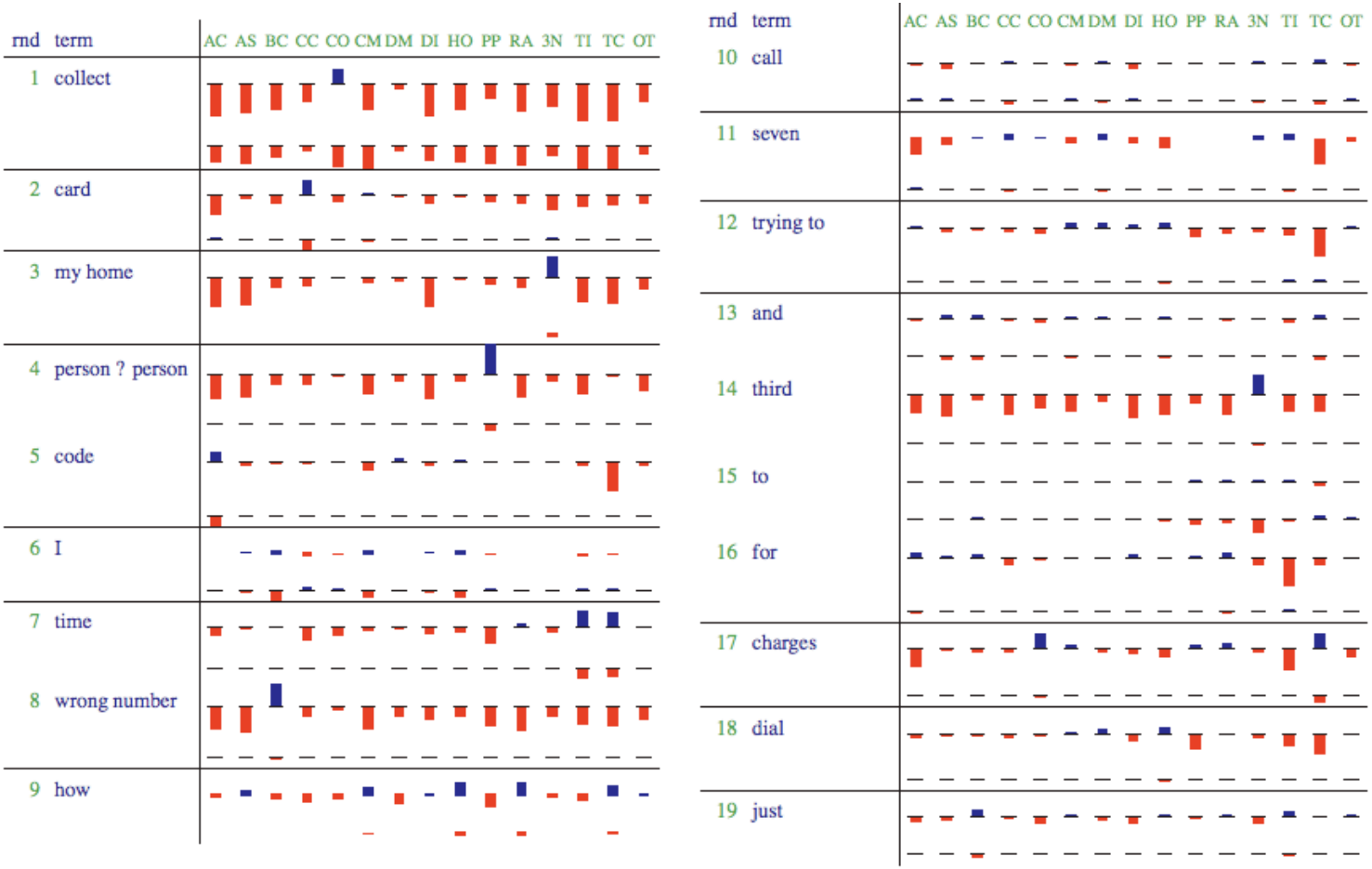
Atouts de la méthode

- La mise en œuvre est simple
- Le temps d'apprentissage est faible
- Le classifieur est (très) rapide
- Implentation possible sur des puces dédiées
- Résultats (empiriques & théoriques) concainquants
  Possibilité de traiter des grandes masses de données
- Extension directe au cas multi-classe : il suffit que h_t prédise un vecteur
- Interprétation des résultats possible (en fonction des hypothèses choisies)

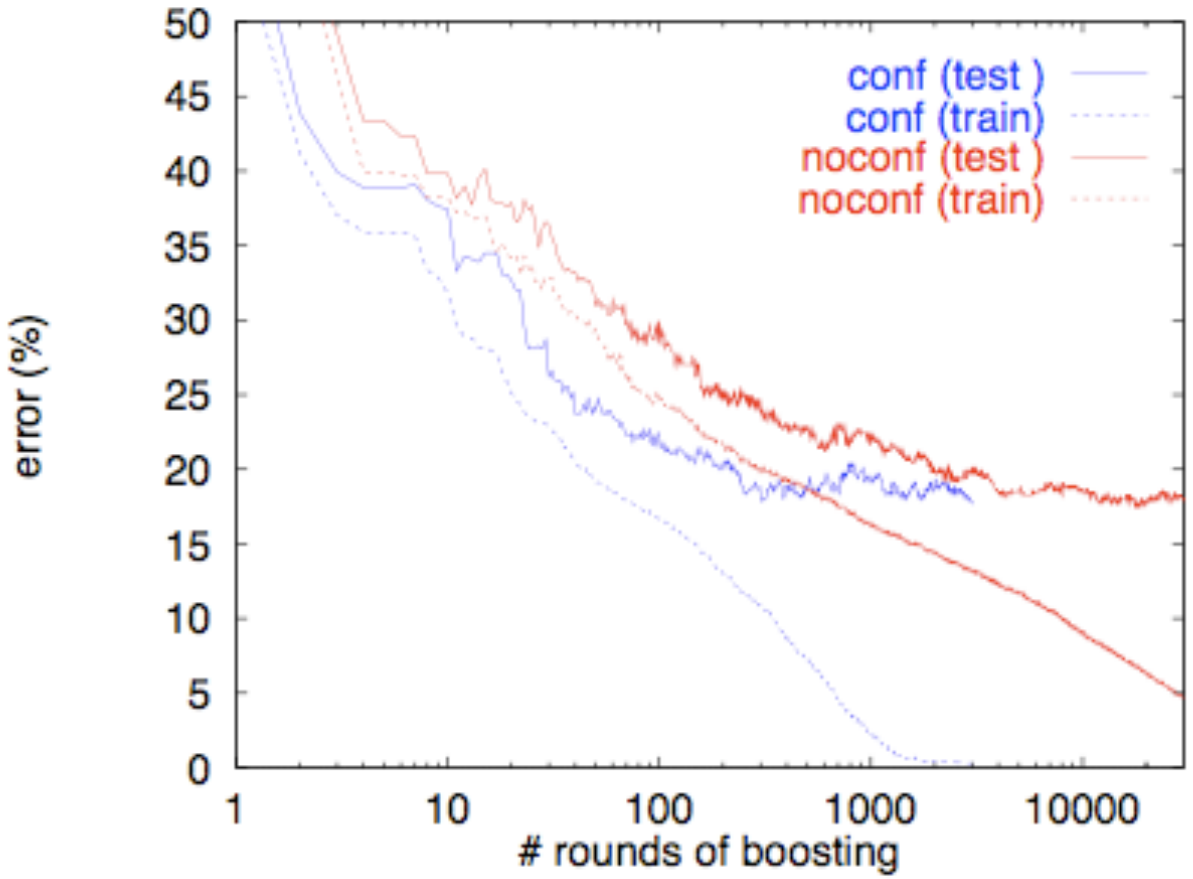
Ex : Boosting pour la classification de textes (1)

- weak hypotheses: very simple weak hypotheses that test on simple patterns, namely, (sparse) n-grams
  - find parameter α_t and rule h_t of given form which minimize Z_t
  - use efficiently implemented exhaustive search
- "How may I help you" data:
  - 7844 training examples (hand-transcribed)
  - 1000 test examples (both hand-transcribed and from speech recognizer)
  - categories: AreaCode, AttService, BillingCredit, CallingCard, Collect, Competitor, DialForMe, Directory, HowToDial, PersonToPerson, Rate, ThirdNumber, Time, TimeCharge, Other.

Ex : Boosting pour la classification de textes (2)



Ex : Boosting pour la classification de textes (3)



http://www.cs.ucsd.edu/~yfreund/adaboost/index.html