Principe de la recherche d'information Application à l'indexation et la recherche d'images par le contenu

Apprentissage par combinaison de classifieurs : le boosting

Alexis LECHERVY

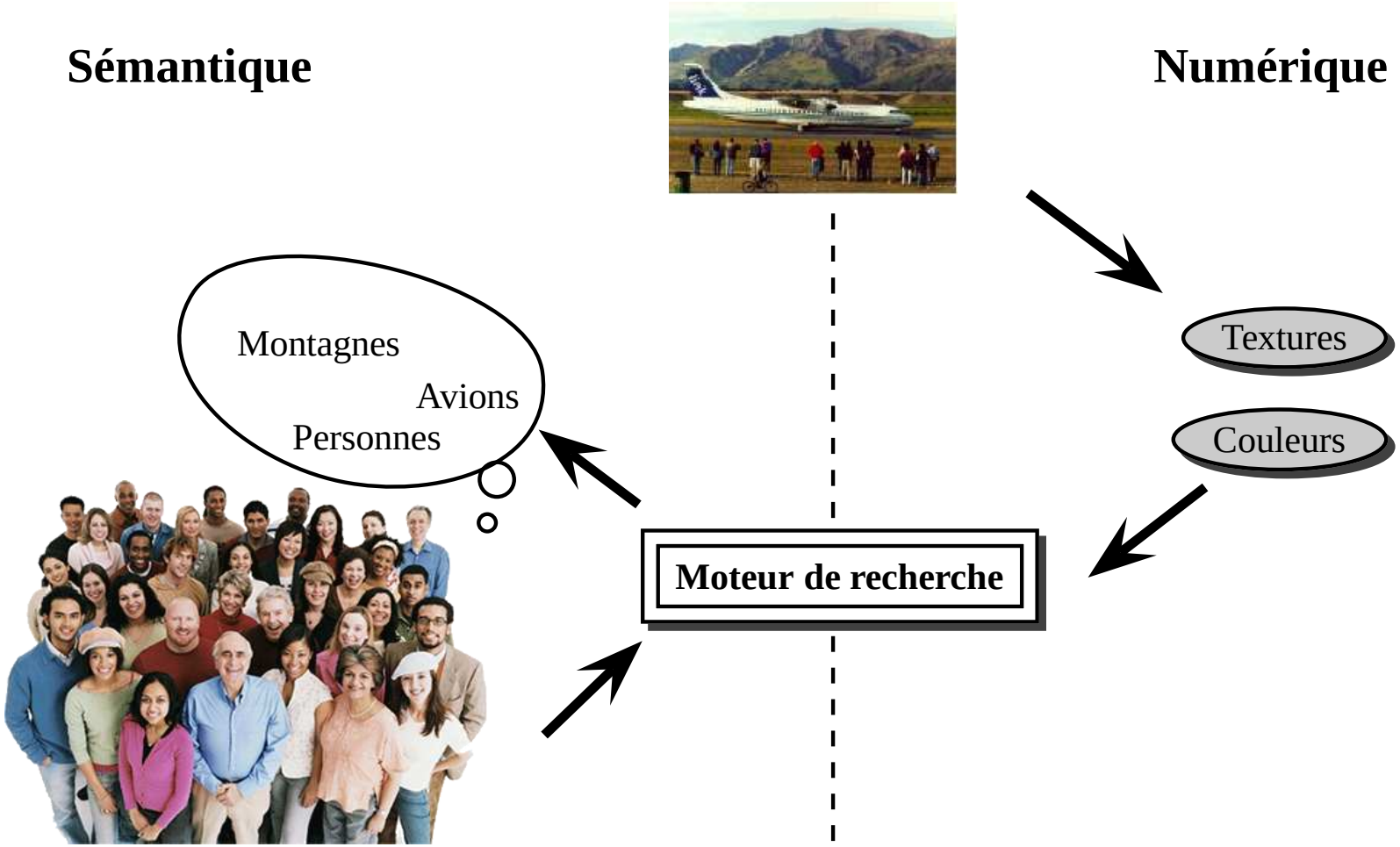
4 février 2011

Contenu largement inspiré du livre et des cours d'Antoine Cornuéjols et de Laurent Miclet et du cours de Frédéric Precioso

Sommaire

Sommaire

Apprentissage par le contenu



Principe général

Observation :

- Il est "facile" d'avoir des règles qui sont "généralement" justes
  - SI un objet est entouré de ciel ALORS c'est un avion
- Il est difficile d'avoir des règles toujours justes

Sommaire

Exemple : La prédiction de courses hippiques

Comment gagner aux courses ?

1re solution : Trouver un expert des paries

Problèmes :

- Coûte très cher
- Demande beaucoup d'expérience et de connaissances des courses
- Pas de règles simples qui marchent tout le temps

=> Cher et difficile à trouver

2ème solution : Interroger les parieurs

Problèmes :

- Beaucoup de règles (généralement simple), de cas par cas
- Des règles peu performantes mais meilleures que le hasard

=> Simple, facile à trouver mais peu performant

Une idée

1. Demander aux parieurs des heuristiques
2. Recueillir un ensemble de cas pour lesquels ces heuristiques échouent (cas difficiles)
3. Ré-interroger les joueurs pour qu'ils fournissent des heuristiques pour les cas difficiles
4. etc ...

Il ne reste plus qu'à combiner toutes ces règles.

Points importants

Comment choisir les courses (échantillons d'apprentissage) à chaque étapes ?

Se concentrer sur les courses (exemples) les plus "difficiles" (celles sur lesquelles les heuristiques précédentes sont les moins performantes)

Comment combiner les heuristiques en une seule règle de prédiction ?

Prendre une vote majoritaire pondéré de ces règles

La solution : le Boosting

Définition

Méthode générale pour convertir des règles de prédiction peu performantes (weak classifier) en une règle de prédiction (très) performante (strong classifier)

Définition plus formel :

1. PAC apprenable au sens fort si :
   - Pour toute distribution
   - $\forall \epsilon > 0, \delta > 0$
   - On trouve un classifieur ayant une erreur $\leq \epsilon$ avec une probabilité $1 - \delta$
2. PAC apprenable au sens faible si :
   - Pareil mais il suffit juste que l'erreur vérifie : $\frac{1}{2} - \gamma \leq \epsilon$
3. PAC apprenable au sens faible $\Rightarrow$ PAC apprenable au sens fort

La petite histoire

- Une question de **Kearns** : " Est-il possible de rendre aussi bon que l'on veut un algorithme d'apprentissage faible " (c'est-à-dire un peu meilleur que le hasard) ?
- La réponse de **Schapire**, en 90, est : " Oui ! " , et il propose le premier algorithme élémentaire de boosting, qui montre qu'un algorithme de classification binaire faible peut toujours améliorer sa performance en étant entraîné sur **trois échantillons d'apprentissage** bien choisis. L'algorithme d'apprentissage n'a pas d'importance (un arbre de décision, une règle bayésienne de classification, une décision dépendant d'un hyperplan, etc.), mais il faut choisir les trois sous-ensembles d'apprentissage en fonction de ses performances.

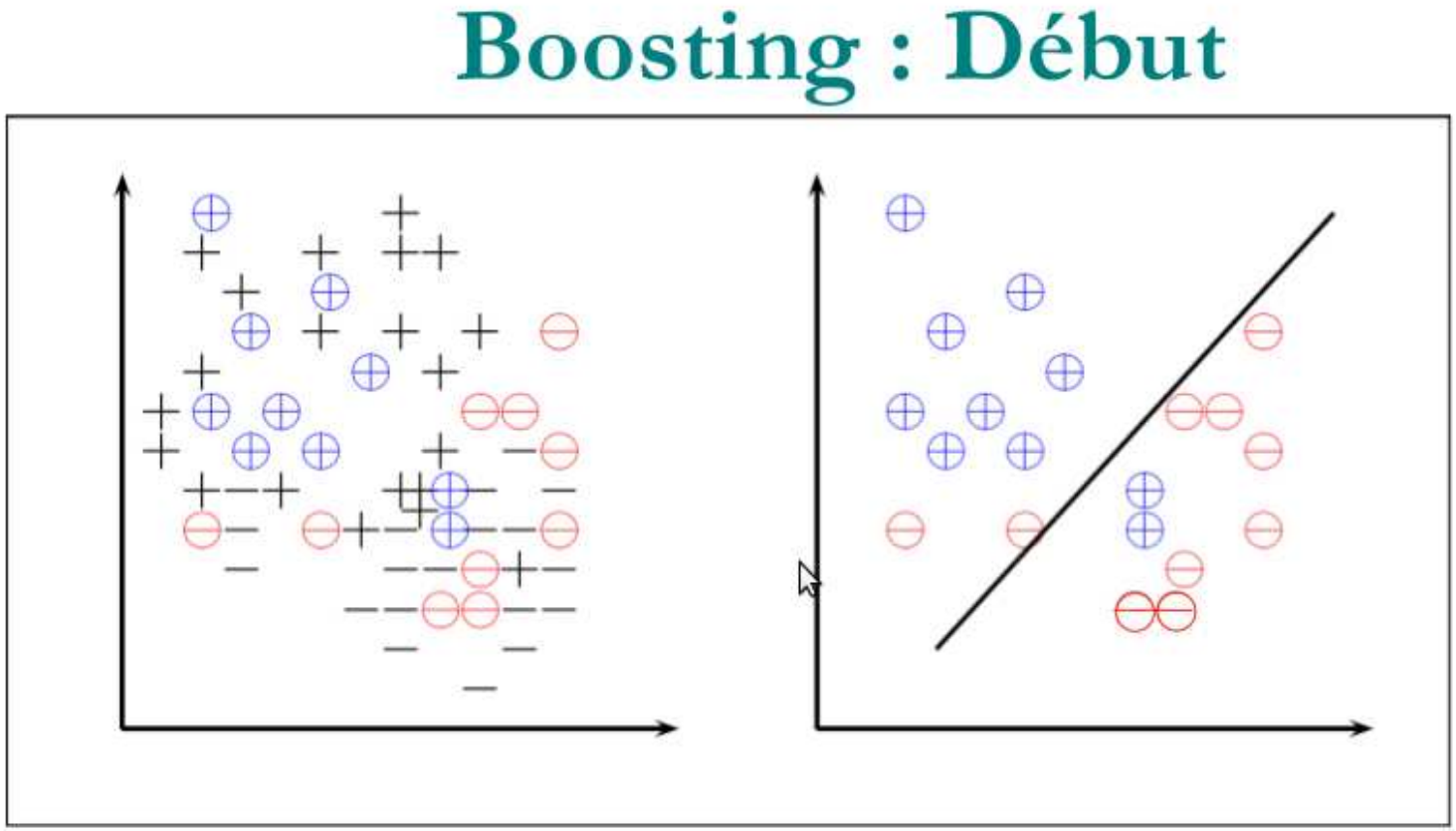
Un premier algorithme de boosting élémentaire

1. On obtient d'abord une première hypothèse h_1 sur un sous-échantillon S_1 d'apprentissage de taille $m_1 < m$ (m étant la taille de S l'échantillon d'apprentissage disponible).
2. On apprend alors une deuxième hypothèse h_2 sur un échantillon S_2, de taille m_2, choisi dans $S - S_1$ dont la moitié des exemples sont mal classés par h_1
3. On apprend finalement une troisième hypothèse h_3 sur m_3 exemples tirés dans $S - S_1 - S_2$ pour lesquels h_1 et h_2 sont en désaccord.
4. L'hypothèse finale est obtenue par un vote majoritaire des trois hypothèses apprises : $H =$ vote majoritaire(h_1, h_2, h_3)

Boosting : efficacité

- Schapire montre que le classifieur final est meilleur qu'un classifieur directement appris sur S
- Le classifieur est de type hyperplan

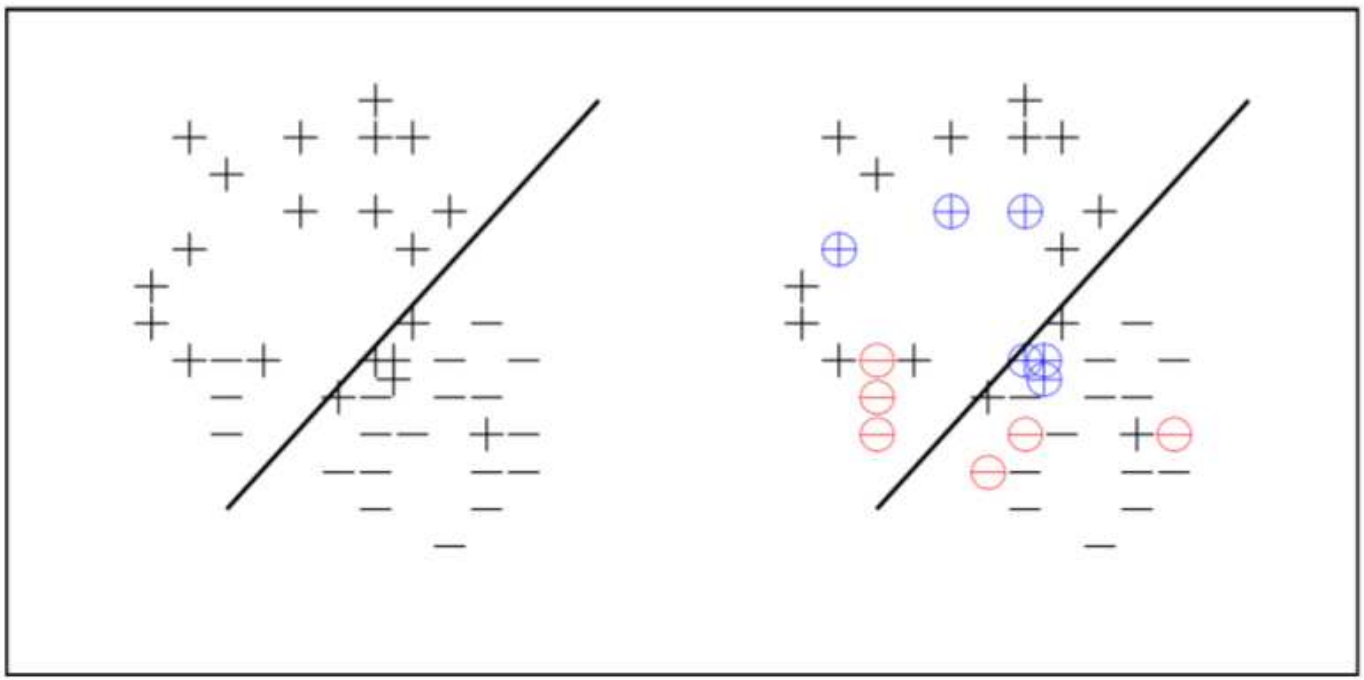
Boosting : Première étape



A gauche : l'ensemble d'apprentissage S et le sous-ensemble S_1 *(points rouges ou bleus entourés)*.
A droite : l'ensemble S_1 et la droite C_1 apprise sur cet ensemble.

Boosting : Deuxième étape

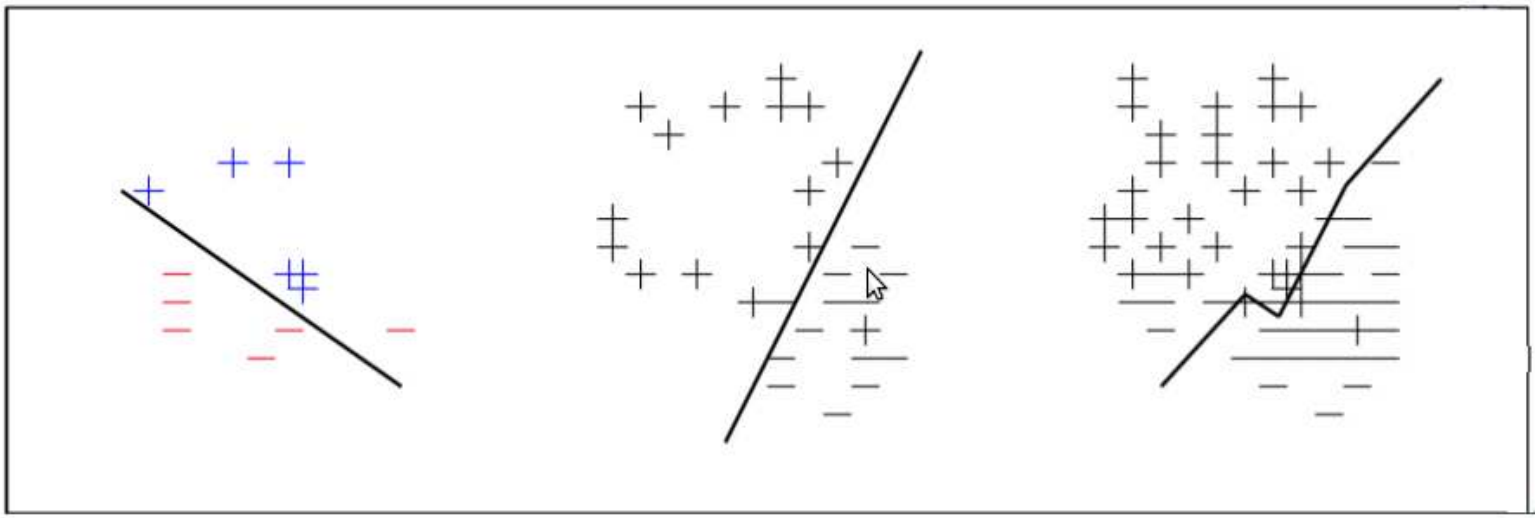
Boosting : Suite



A gauche : l'ensemble d'apprentissage $S - S_1$ et la droite C_1 apprise sur S_1.
A droite : un ensemble S_2 *inclus* dans $S - S1$ parmi les plus informatifs pour C_1 *(points rouges ou bleus entourés).*

Boosting : Dernière étape

Boosting : ...et fin



A gauche : S_2 *est la droite* C_2 *apprise sur* S_2.

Au centre : $S_3 = S - S_1 - S_2$ *et la droite* C_3 *apprise sur* S_3.

A droite : S *et la combinaison des 3 droites.*

En pratique

- Idéalement, les trois ensembles d'exemples extraits de S devrait parcourir l'ensemble des exemples d'apprentissage.
- Mais cela n'est pas nécessairement facile à faire : si l'algorithme A est performant sur S, m_2 pourra être pris bien inférieur à m_1, alors que la proportion pourrait être inverse si A est seulement un peu meilleur qu'un tirage aléatoire.
- En général, on règle empiriquement les proportions des trois ensembles en faisant plusieurs essais, jusqu'à ce que tous les éléments de S ou presque participent au processus.

Généralisation : le Boosting probabiliste

Le boosting probabiliste

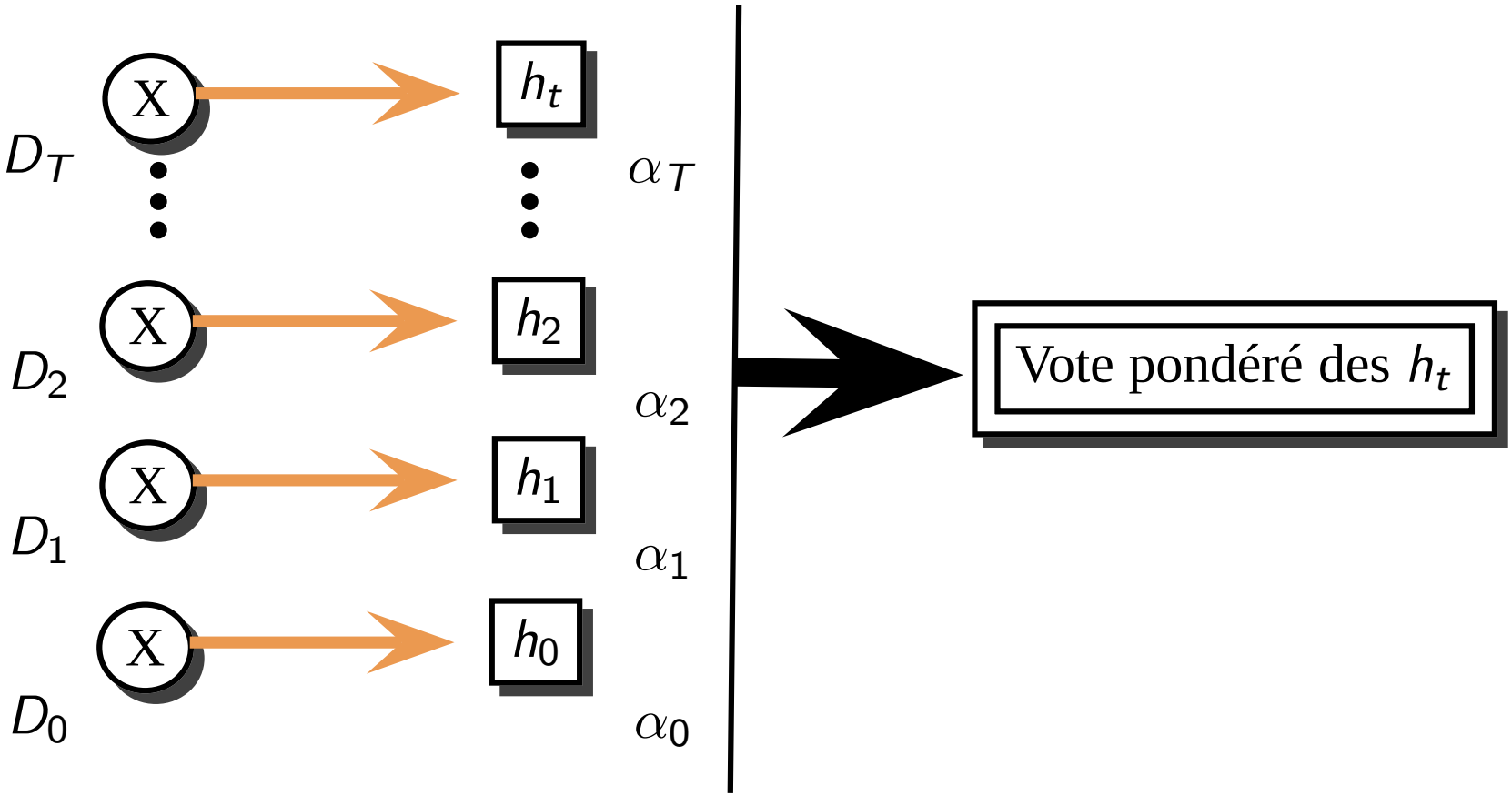
1. Utilise une comité d'experts que l'on fait voter pour atteindre une décision
2. Les votes sont pondérés en fonction de la qualité de l'expert
3. Les exemples ont un poids qui évolue par mise à jour multiplicative, en fonction de la qualité de classement des experts déjà choisie

Algorithme général :

Algorithme

- *Paramètre* : Un ensemble d'hypothèses h
- *Entrée* : Un ensemble d'apprentissage $E = (x_i, y_i)$
- Initialisation : Tous les exemples ont le même poids $D(i)$
- Pour t=0 à T faire :
  1. Sélectionner un classifieur h_t
  2. Renforce le poids des exemples mal classés par l'hypothèse h_t
- *Sortie* : H un vote majoritaire pondéré des différents h_t sélectionnés

Illustration de l'algorithme :



Points importants :

L'algorithme générique précédent repose sur 3 points importants :

- La **sélection**, à chaque itération du boosting, d'un classifieur faible
- La **mise à jour des poids** des exemples d'apprentissage
- La nature des **classifieurs faibles**

AdaBoost : un algorithm de boosting probabiliste

L'algorithme standard s'appelle **AdaBoost** (Adaptive Boosting). Il s'appuie sur :

- Choisie, selon un critère, le meilleur des classifieurs faibles par rapport aux pondérations des exemples
- Les poids $D_t(i)$ des exemples sont mis à jour à chaque étapes en fonction des poids précédent et des résultats du classifieurs sélectionnés.
- Au départ, tout les exemples ont la même importance.

Adaboost

Algorithme

1. Une base d'exemple $(x_1, y_1), ..., (x_m, y_m)$
2. Avec des labels $y_i \in \{-1, 1\}$
3. Pour tout $t = 1, ..., T$
   - On trouve le classifieur faible minimisant un critère d'erreur
   - On calcule son poids
   - On calcule une distribution $D_t(i)$ des exemples
4. On construit le classifieur fort en effectuant une somme pondérée des classifieurs sélectionnés

Adaboost

- Facteur de distribution des exemples :

$$\begin{cases} D_0(i) = \frac{1}{m} \\ D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{normalisation} \end{cases}$$

- Erreur à minimiser : $\epsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$
- Poids d'un classifieur $\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

Le classifieur fort

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

Adaboost : le classifieur final

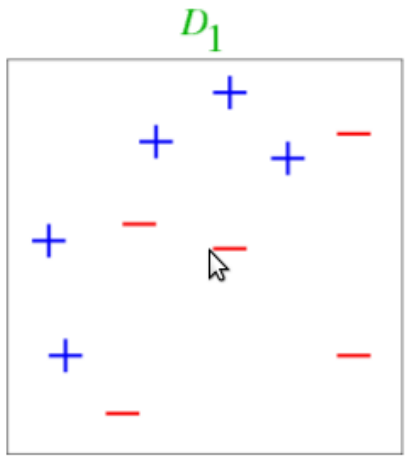
$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

En un sens, on voit que le boosting construit l'hypothèse finale comme une série additive dans une base de fonctions, dont les éléments sont les hypothèses h_t. On retrouve là un thème fréquent dans les techniques d'apprentissage (par exemple les SVM, les méthodes d'approximation bayésiennes, etc.).

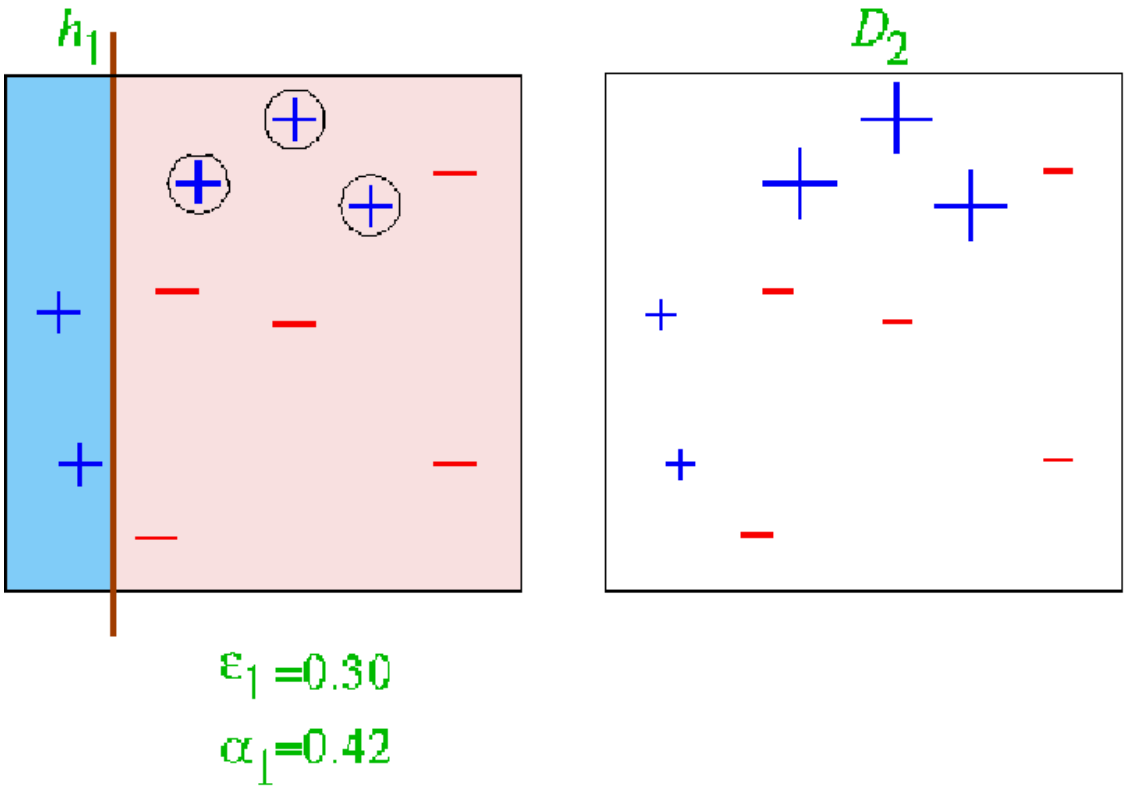
Exemple "Toy"

Principe

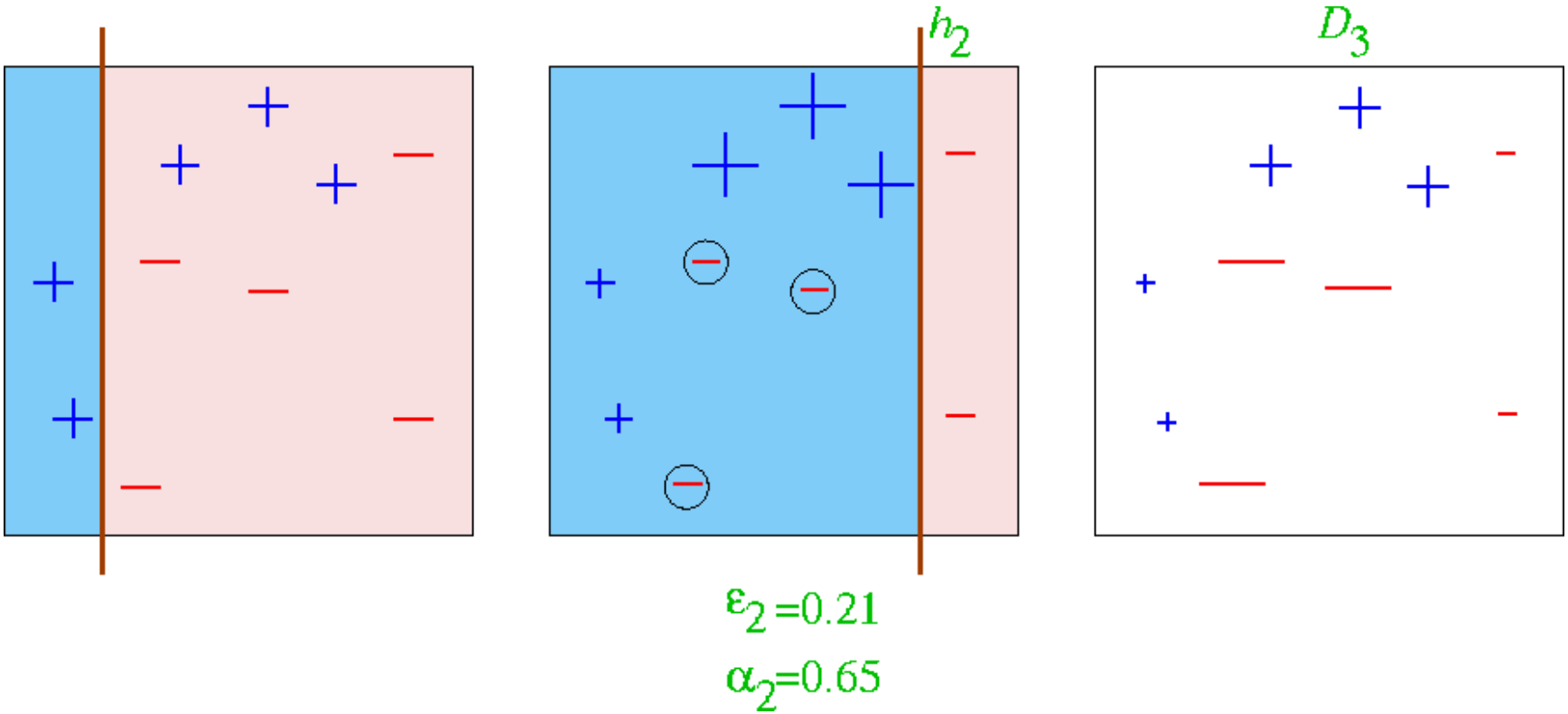
- On considère l'exemple «Toy», de points 2D dans un domaine D_1
- Les données sont soit positives, soit négatives (deux catégories)
- Les classifieurs faibles sont les droites horizontales et verticales



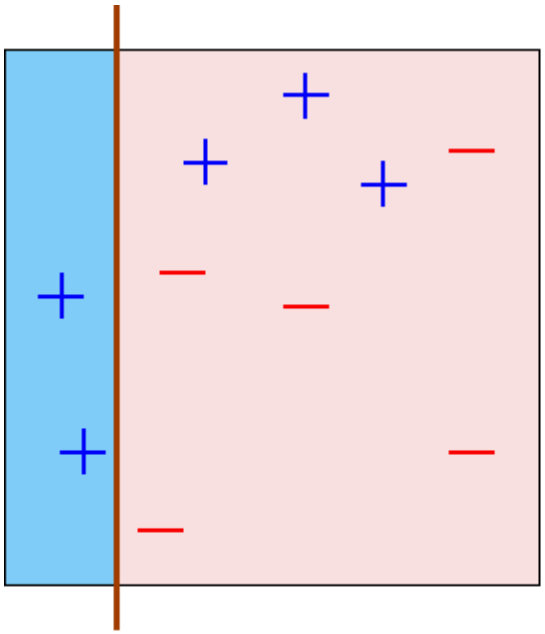
Exemple "Toy" : étape 1

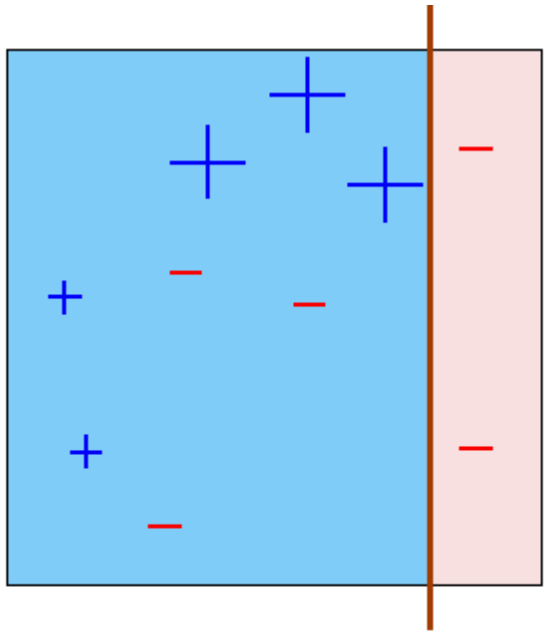


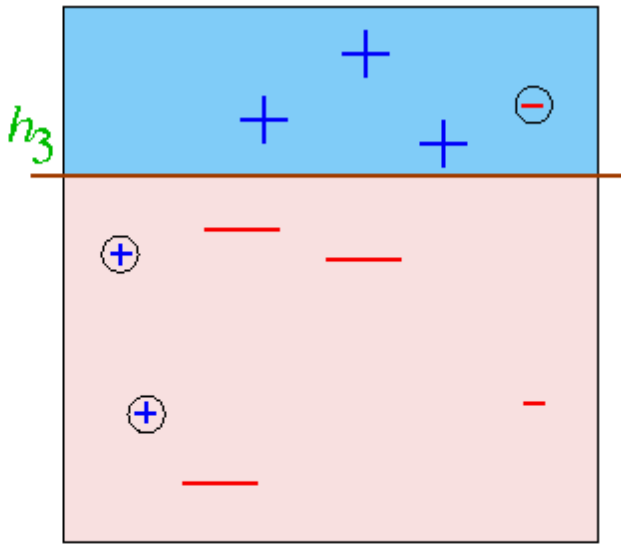
Exemple "Toy" : étape 2



Exemple "Toy" : étape 3



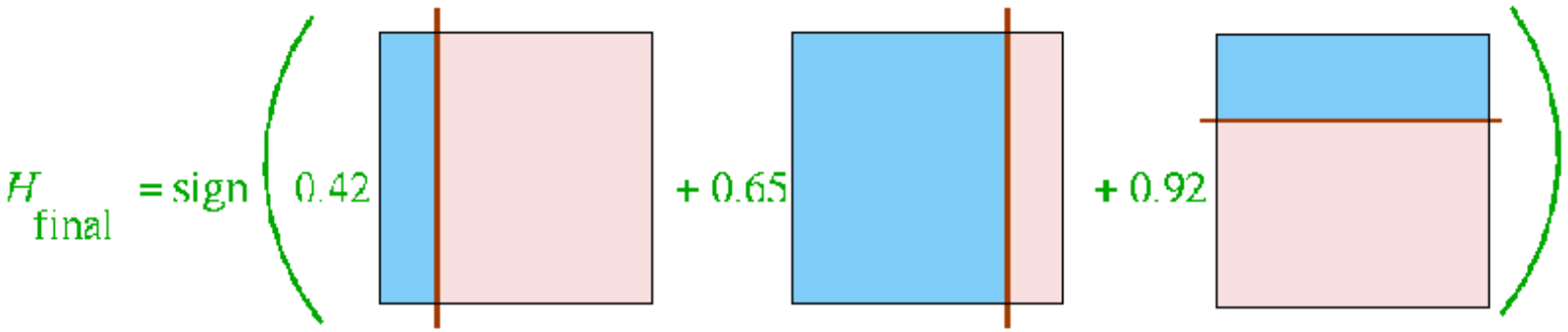


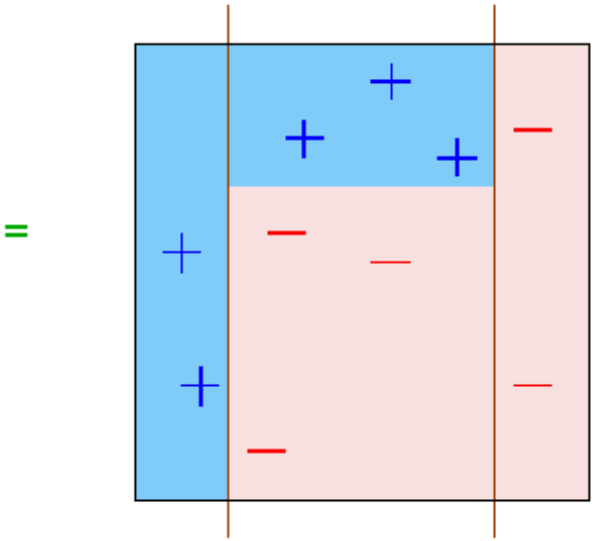


$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Exemple "Toy" : classifieur final





L'erreur empirique d'AdaBoost

Ecrivons l'erreur ϵ_t de h_t comme : $\frac{1}{2} - \gamma_t$, où γ_t mesure l'amélioration apportée par l'hypothèse h_t par rapport à l'erreur de base $\frac{1}{2}$. Freund et Schapire, ont montré que l'erreur empirique (la fraction d'erreur sur l'échantillon d'apprentissage S) de l'hypothèse finale H est bornée par :

$$\prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq exp\Big(-2\sum_t \gamma_t^2\Big)$$

Ainsi, si chaque hypothèse faible est légèrement meilleure que le hasard, $(\gamma_t > 0)$, alors l'erreur empirique diminue exponentiellement rapidement avec t.

Preuve :

Soit $H(x) = \sum_t \alpha_t h_t(x)$
On a :

$$\begin{aligned} D_{final}(x_i, y_i) &= \frac{D_t}{Z_t} exp(-\alpha_t \cdot y_i \cdot h_t(x_i)) \\ &= \frac{1}{m} \frac{exp(-y_i \sum_t \alpha_t h_t(x_i))}{\prod_t Z_t} \\ &= \frac{1}{m} \frac{exp(-y_i H(x_i))}{\prod_t Z_t} \end{aligned}$$

On a donc $\boxed{D_{final}(x_i, y_i) \cdot \prod_t Z_t = \frac{1}{m} exp(-y_i H(x_i))}$

Preuve : L'erreur empirique est majoré par $\prod_t Z_t$

$$
\begin{aligned}
err_H = Pr[H(x_i) \neq y_i] &= \frac{1}{m} \sum_i \mathbb{1}_{H(x_i) \neq y_i} \\
&= \frac{1}{m} \sum_i \mathbb{1}_{y_i H(x_i) \leq 0} \\
&\leq \frac{1}{m} \sum_i exp(y_i H(x_i)) \\
&= \frac{1}{m} \sum_i D_{final}(x_i, y_i) \cdot \prod_t Z_t \\
&= \frac{1}{m} \prod_t Z_t
\end{aligned}
$$

$$
\boxed{err_H \leq \frac{1}{m} \prod_t Z_t}
$$

Preuve : Fin de la preuve

Or

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

D'où l'erreur empirique est bornée par :

$$\boxed{\prod_t 2\sqrt{\epsilon_t(1-\epsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq exp\Big(-2\sum_t \gamma_t^2\Big)}$$

L'erreur en apprentissage diminue exponentiellement rapidement avec t

L'erreur en généralisation (théorie)

Erreur en généralisation

L'erreur en généralisation du classifieur H finale peut être bornée par :

$$Err_{réel}(H) \leq Err_{emp} + O\left(\sqrt{\frac{T \cdot d_H}{m}}\right)$$

- T est le nombre d'itération de Boosting
- d_H est la VC-dimension de l'espace des classifieurs faible h_t ($\approx$ complexité des classifeurs faibles)
- m le nombre d'exemples

L'erreur en généralisation (théorie)

Erreur en généralisation

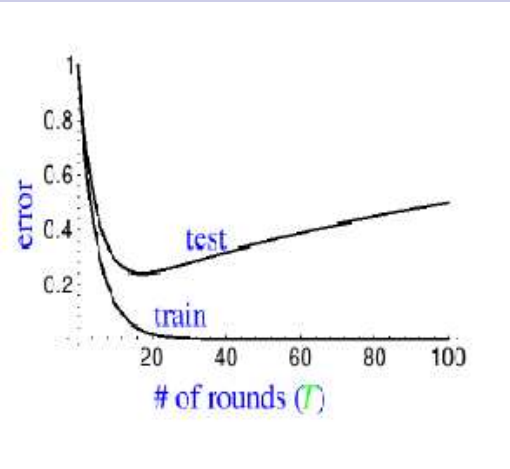
L'erreur en généralisation du classifieur H finale peut être bornée par :

$$Err_{réel}(H) \leq Err_{emp} + O\left(\sqrt{\frac{T \cdot d_H}{m}}\right)$$

Lorsque T grandit, le $2^{\text{ème}}$ terme grandit également, on peut donc s'attendre à un sur-apprentissage de la méthode (on se concentre trop sur les exemples ambigües).

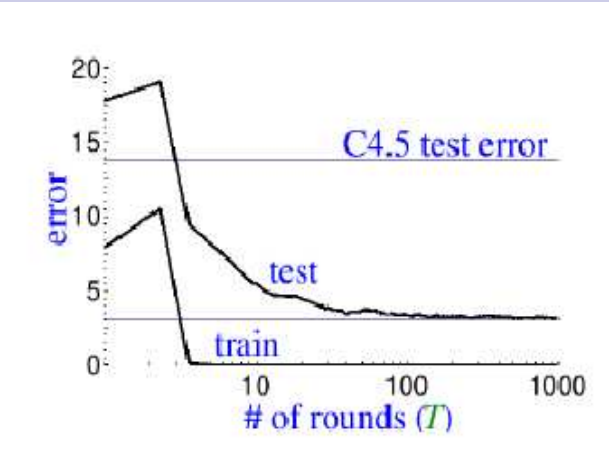
L'erreur en généralisation (pratique)

Comportement classique



- overfiting
- principe du rasoir d'Occam

Comportement réel



- L'erreur de généralisation ne remonte pas
- Le risque réel diminue même après que le risque empirique soit devenu nul

Essai d'explication : théorie des marges

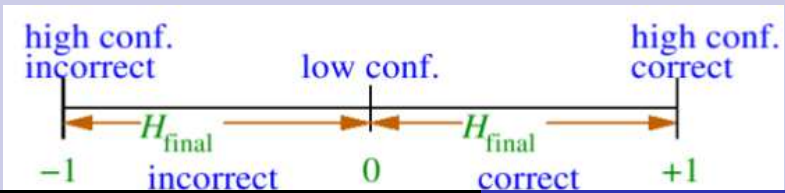
Principe

1. L'erreur empirique mesure uniquement 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 a :

   la fraction votant correctement - fraction ne votant pas correctement

4. Pour le boosting la marge est définis par :

$$\frac{\sum_t^T \alpha_t h_t(x)}{\sum_t^T \alpha_t}$$



Essai d'explication : théorie des marges

- On peut montrer que $\forall \theta > 0$

$$erreur \leq Pr(mage \leq \theta) + O(\frac{\sqrt{\frac{d}{m}}}{\theta})$$

  - m est le nombre d'exemple
  - d est la VC-dimension des classifieurs faibles
- Dans le cas du boosting $Pr(mage \leq \theta) \longrightarrow 0$ expontiellement vite en T.
- Le deuxième terme est maintenant indépendant de T

Théorie des jeux

- On considère un jeu défini par une matrice M

	Pierre	Feuille	Cissaux
Pierre	1/2	1	0
Feuille	1	1/2	0
Cissaux	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 :

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

MinMax

Théorème de Neumann

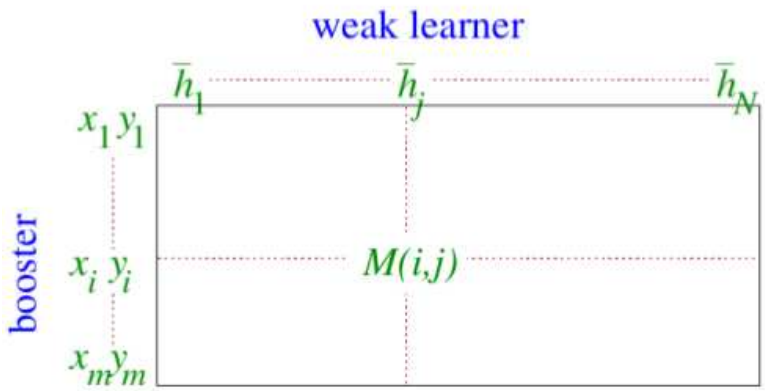
$$\min_P \max_Q M(P,Q) = \max_Q \min_P M(P,Q) = v(\text{la valeur du jeu M})$$

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

Le jeu du Boosting

- Soit $h_1, ..., h_n$, l'espace de tous les classifieurs faibles
- Le joueur "ligne" : les exemples
- Le joueur "colonne" : les classifieurs
- La Matrice M :
  - $M(i,j) = \mathbb{1}_{y_i = h_j(x_i)}$



MinMax et Boosting

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

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

- Donc

$$\min_Q \max_i M(i, Q) \geqslant \frac{1}{2} - \gamma$$

- 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.

Avantages d'AdaBoost

Avantages d'AdaBoost

- Évaluation/détection très rapide
- Facile à programmer
- Facile à régler, un seul paramètre : le nombre d'étape T du boosting
- Peux "booster" n'importe quel algorithme d'apprentissage
- Pas de sur-apprentissage
- Peux être adapté au cas multi-classe
- Peux détecter les exemples aberrants

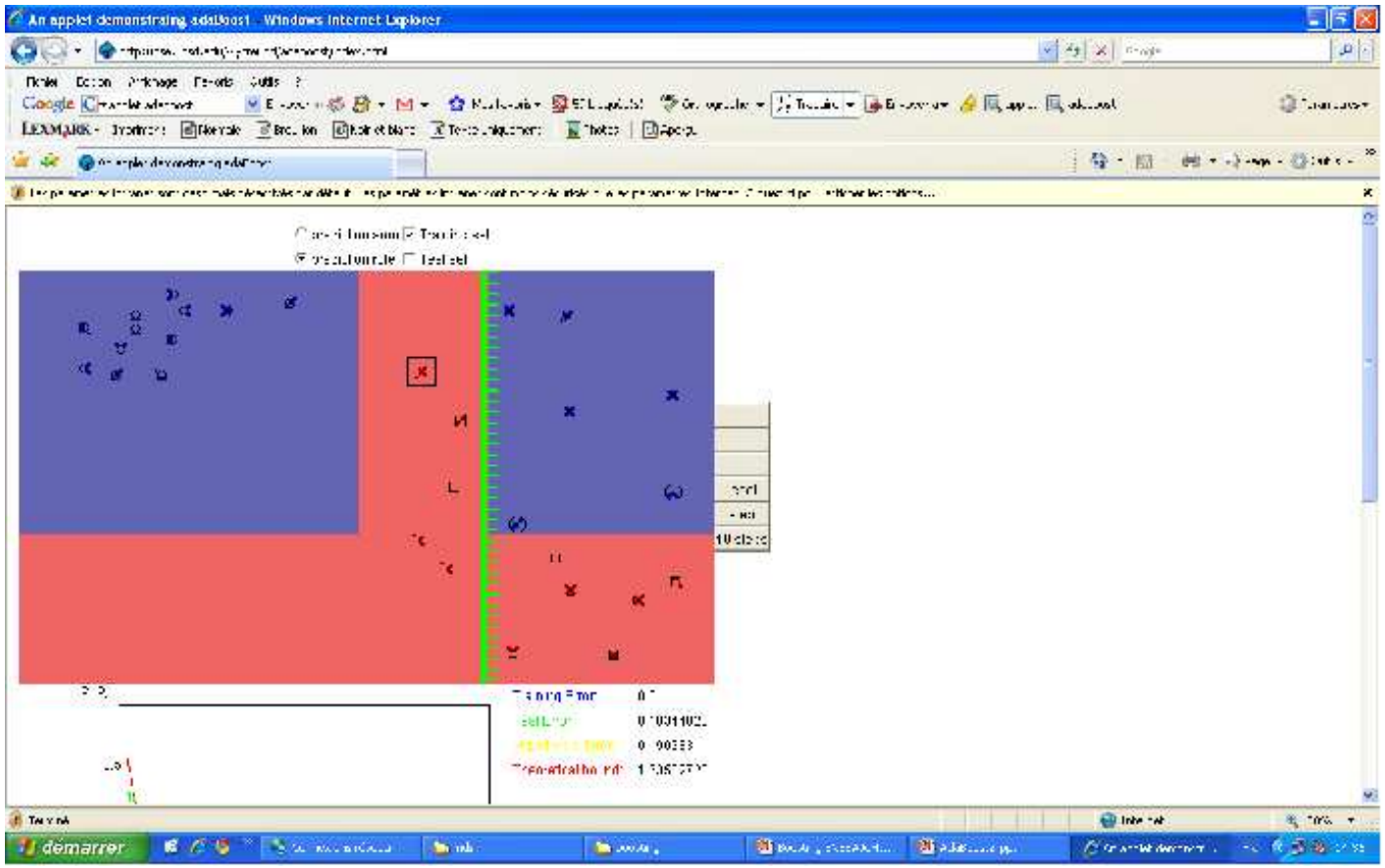
Défauts d'AdaBoost

Défauts d'AdaBoost

- Les performances dépendes du types de classifieurs faibles choisie
- Les performances dépendes des données
- Avec des classifieurs faibles "trop fort", AdaBoost sur-apprend
- Avec des classifieurs faibles "trop faible", AdaBoost converge trop lentement
- AdaBoost est sensible au bruit sur les données d'apprentissage
- Le classifieur n'est pas interprétable par un humain
- le temps d'apprentissage est très long et requière de nombreux exemples

Applet de démo

http ://cse.ucsd.edu/~yfreund/adaboost/index.html



Bibliographie sur le Boosting et AdaBoost

- *Valiant'84* introduction à la théorie de la Pac-apprenabilité et étude des méthodes d'apprentissage
- *Kearns et Valiant'88* énoncé du problème ayant conduit au boosting
- *Schapire'89*, *Freund'90* premier algorithme polynomial de boosting
- *Drucker, Schapire et Simard'92* première expérience utilisant du boosting
- *Freund et Schapire'95* AdaBoost : premier algorithme réellement fonctionnel

Bibliographie sur le Boosting et AdaBoost

- expériences utilisant AdaBoost : *Drucker et Cortes'95*, *Jackson et Cravon'96*, *Freund et Schapire'96*, *Quinlan'96*, *Breiman'96*, *Schapire et Singer'98*, *Maclin et Opitz'97*, *Bauer et Kohavi'97*, *Schwenk et Bengio'98*, *Dietterich'98*,
- prolongement de la théorie : *Schapire,Freund,Bartlett et Lee'97*, *Schapire et Singer'98*, *Breiman'97*, *Mason, Bartlett et Baxter'98*, *Grive et Schuurmans'98*, *Friedman, Hastie et Tibshirani'98*

Sommaire

Détection de visages

Démo

Rappel historique : le boosting et la détection de visage

- 1990 Schapire le Boosting : classifieurs faibles -> classifieur fort
- 1996 Freund et Schapire : AdaBoost (algorithme général)
- 2001 Viola et Jones : Détection des visages
- 2002 Lienhart et al. : Descripteur étendu

Détection des visages : Viola et Jones

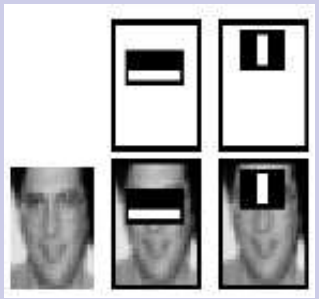
- Utilisation d'un AdaBoost classique
- Propose de nouveau classifieurs faibles basés sur les descripteurs de Haar
- Introduit le concept de cascade

Descripteur de Haar

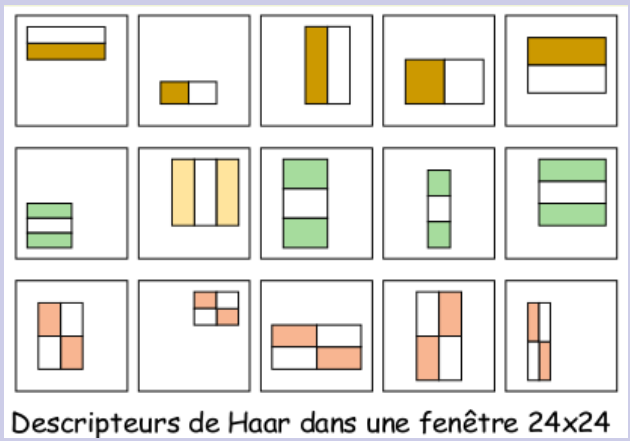
Principe

$$h_i(x) = \begin{cases} 1 \ si \ f_i(x) > \theta_i \\ 0 \ sinon \end{cases}$$

Avec $f_i(x) = \sum_i r_{i,blanche} + \sum_i r_{i,noire}$



Exemples



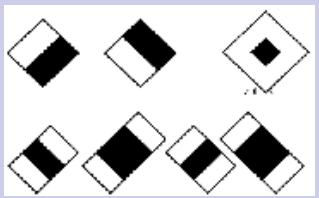
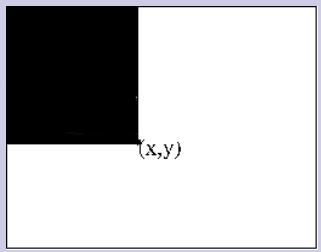


Image intégrale

Principe

- Calcul de l'image intégrale :

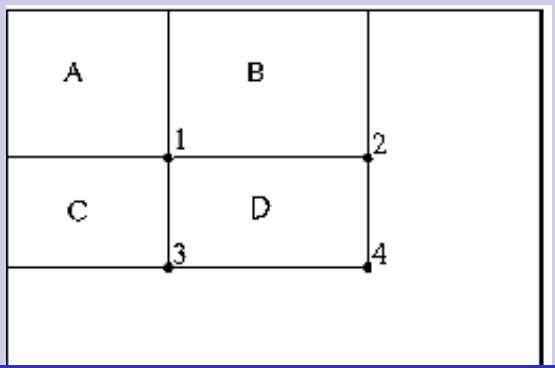
$$ii(x,y) = \sum_{0 \leq x' \leq x, 0 \leq y' \leq y} i(x',y')$$



Calcul plus rapide du descripteur de Haar :

- A partir de l'image ii on peut calculer la somme des pixels d'une région par :

$$\begin{aligned} 2 &= A + B \\ 3 &= A + C \\ 4 &= A + B + C + D \\ D &= 4 + 1 - (2 + 3) \end{aligned}$$



Cascade : Origine

Observation

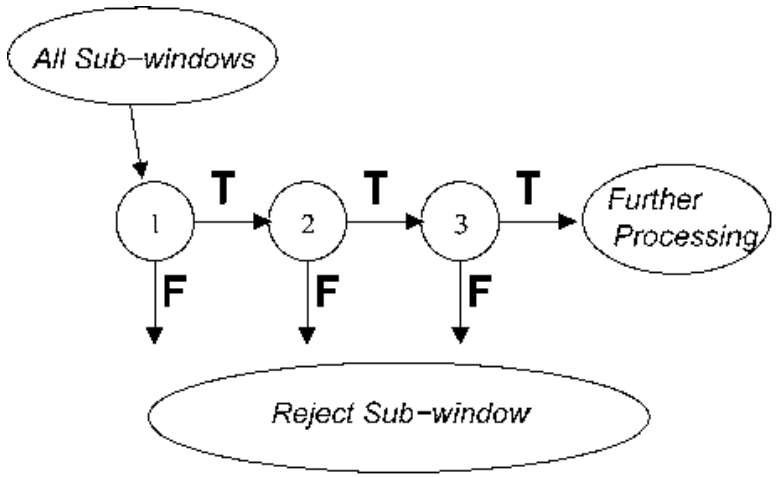
- Une image est constituée **majoritairement** de sous-fenêtres **négatives**
- La détection d'un objet **positif** est un événement **rare**.

$\Longrightarrow$ Il faut éliminer rapidement les sous-images négatives

Cascade : Principe

Principe

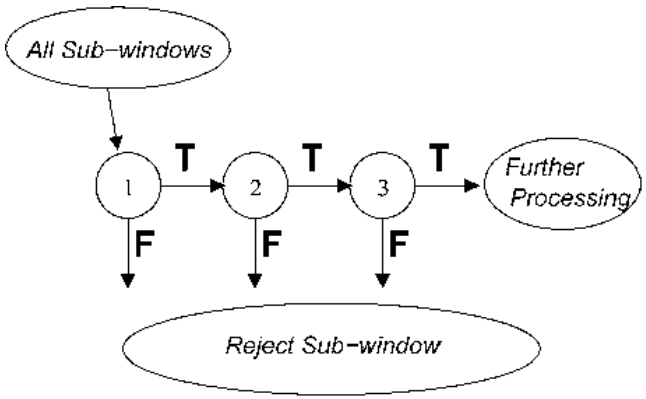
- Une **cascade** de classifieurs est **un arbre de décision** dégénéré dans laquelle, chaque étape est entraînée pour détecter un maximum d'objets intéressants tout **en rejetant une certaine fraction** des objets non-intéressants.



Cascade : Principe

Principe

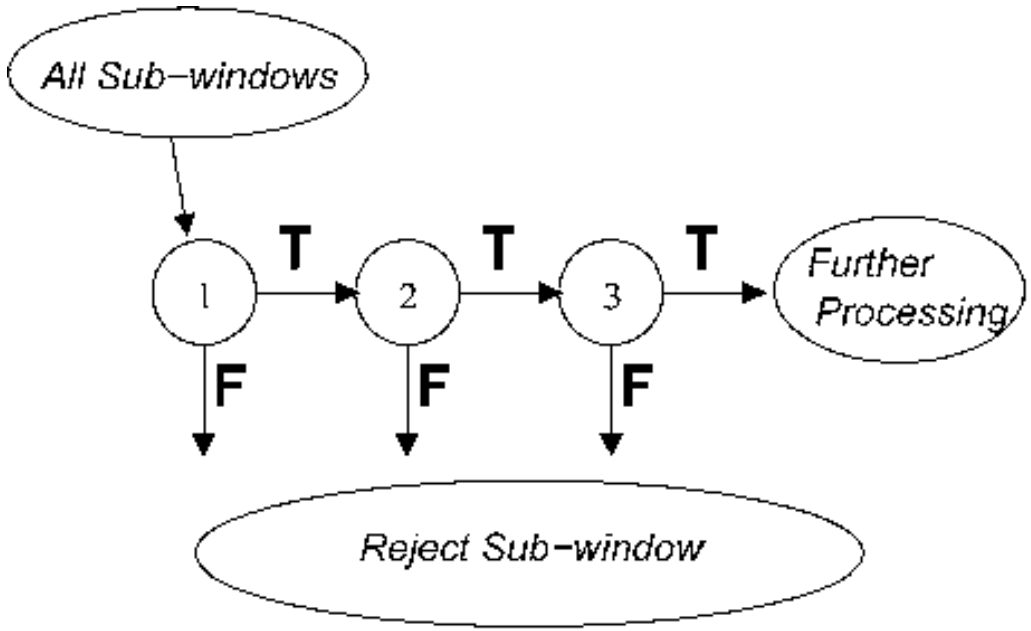
- Une sous-image doit passer tous les classifieurs fort afin d'être acceptée comme visage.
- Le déclenchement de tous les classifieurs par un résultat positif devient ainsi un événement rare.
- Il faut que le nombre de sous-images éliminées dès les premières étapes de la cascade soit très élevé.



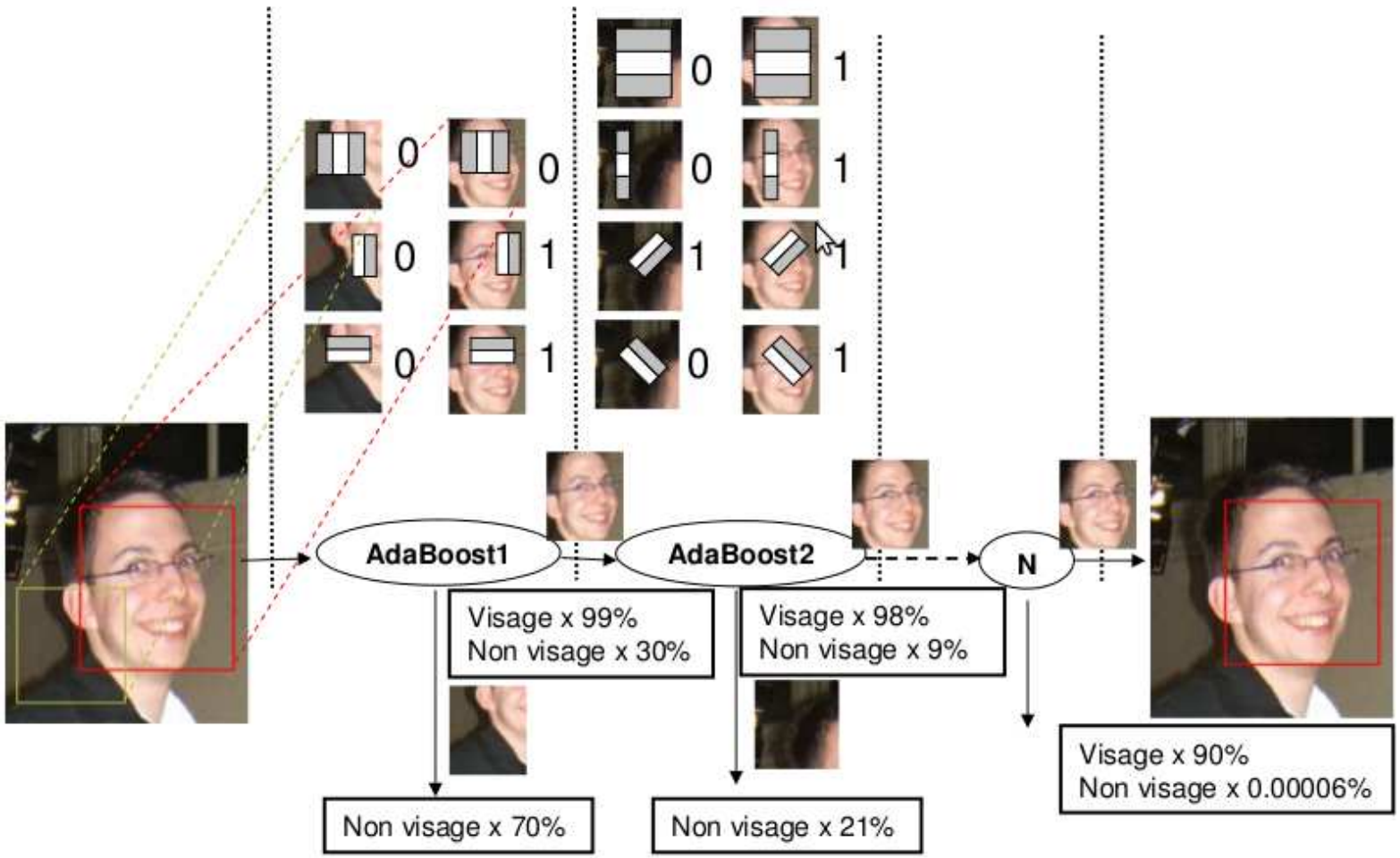
Cascade : Principe

Principe

- taux faux positif global : $\prod_{i=1}^{N} f_i$
- taux détection global : $\prod_{i=1}^{N} d_i$



Cascade : Exemple



Résultats


Visages de face
Visages de profil gauche
Visages de profil droit
LBC
مجلس الأمن في جلسة التصويت على القرار ١٦٣١
LBC

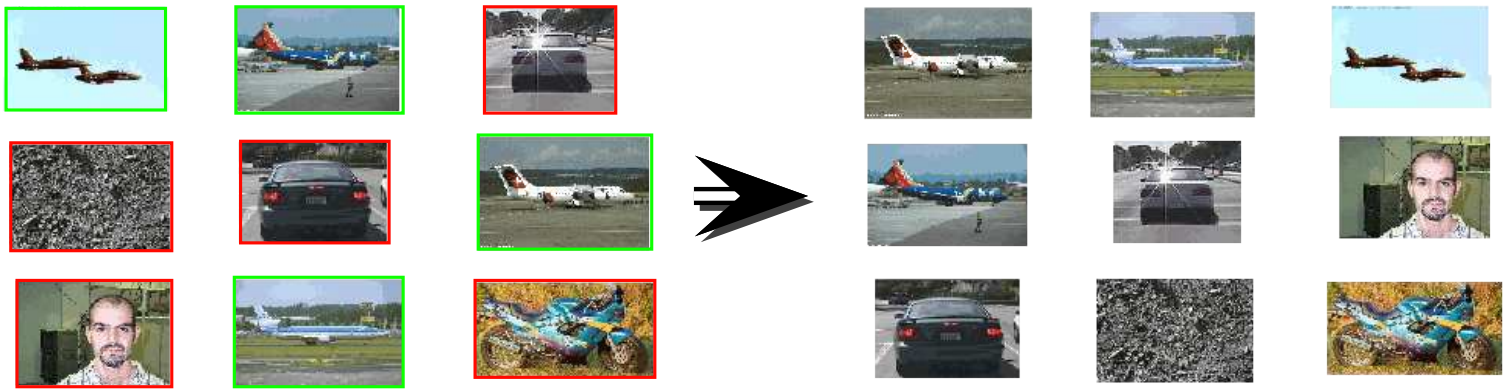
Remarques

- Méthode rapide et robuste
- Variante possible :
  - autres descripteurs
  - d'autres cascades
  - détection des yeux

Apprendre des concepts sémantiques

Objectifs

- Apprendre à reconnaitre des objets par l'exemple
- Apprendre à reconnaitre des événements

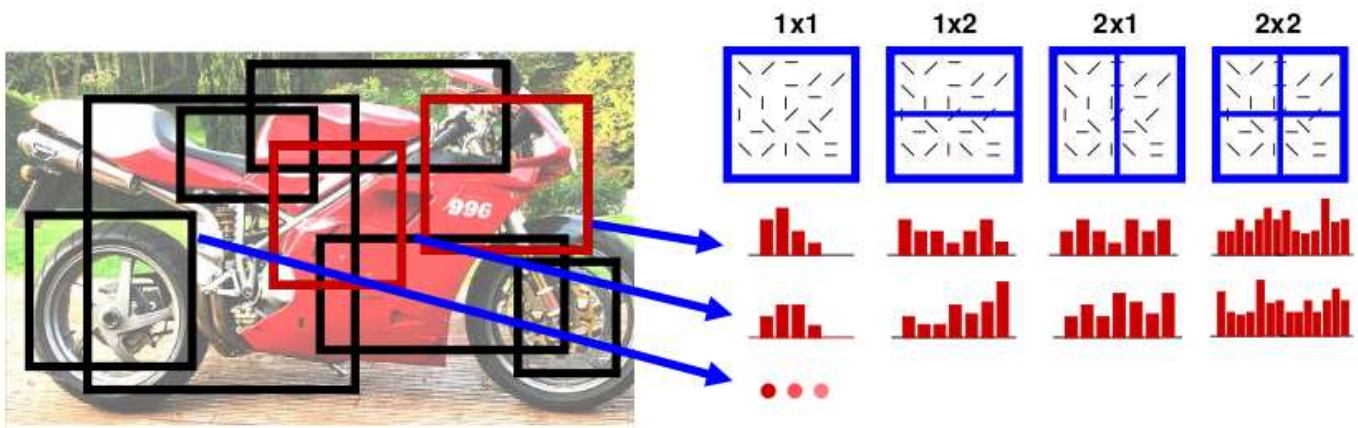


Descripteurs : Histogramme de gradiants orientés (HOG)

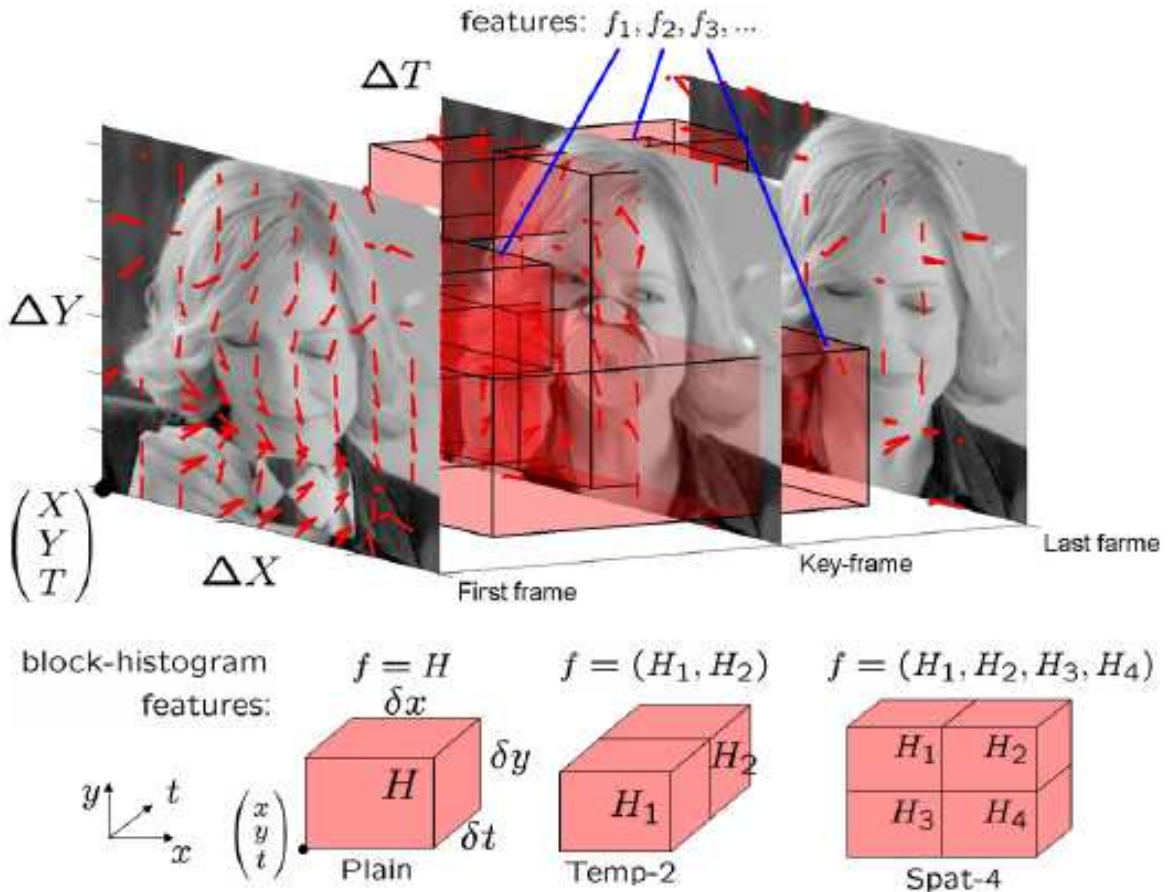
- Histogramme de gradiants (quatres directions)

$$\gamma(x,y) = arctan\frac{L_x(x,y)}{L_y(x,y)} \text{ avec } L_k = I * \frac{\partial}{\partial k}\left(\frac{1}{2\pi\sigma^2}e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}\right)$$

- Différents vecteurs pour prendre en compte des informations positionelles
- Normalisation des coordonnées par la norme du vecteur



Descripteurs pour la détection d'événement dans des vidéos (Ivan Laptev)



Classifieurs faibles : première méthode

Projection sur des vecteurs prédéfinis

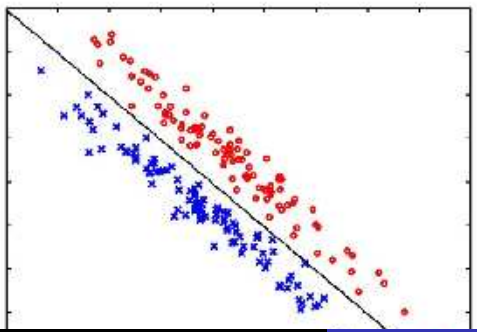
- $f(x) \in \mathbb{R}^N$ un descripteur de l'image x
- $w \in \mathbb{R}^N$ un vecteur de référence
- Le classifieur faible : $h(x) = w^T \cdot f(x) \in \mathbb{R}$

Classifieurs faibles : deuxième méthode

Projection par Fisher Linear Discriminant (FLD)

- Projection selon $w = \frac{\mu^+ - \mu^-}{\Sigma^+ + \Sigma^-}$
- μ : Moyenne des descripteurs
- Σ : Covariance des descripteurs
- Peu être adapté pour prendre le poids des exemples :

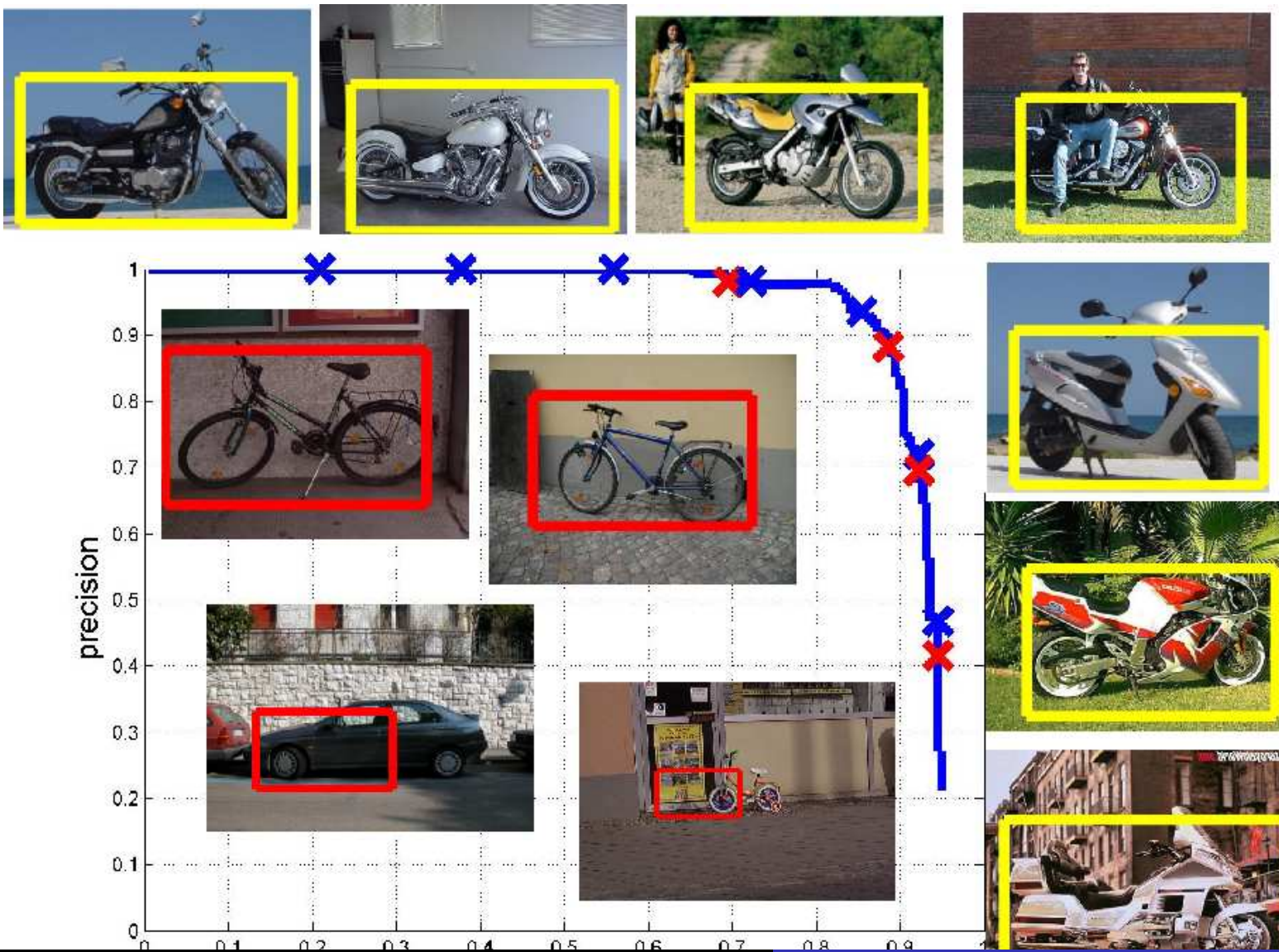
$$\mu_d = \frac{1}{n \sum d_i} \sum_i^n d_i f(x_i) \text{ et } S_d = \frac{1}{(n-1) \sum d_i^2} \sum_i^n d_i^2 (f(x_i) - \mu_d)(f(z_i) - \mu_d)^T$$



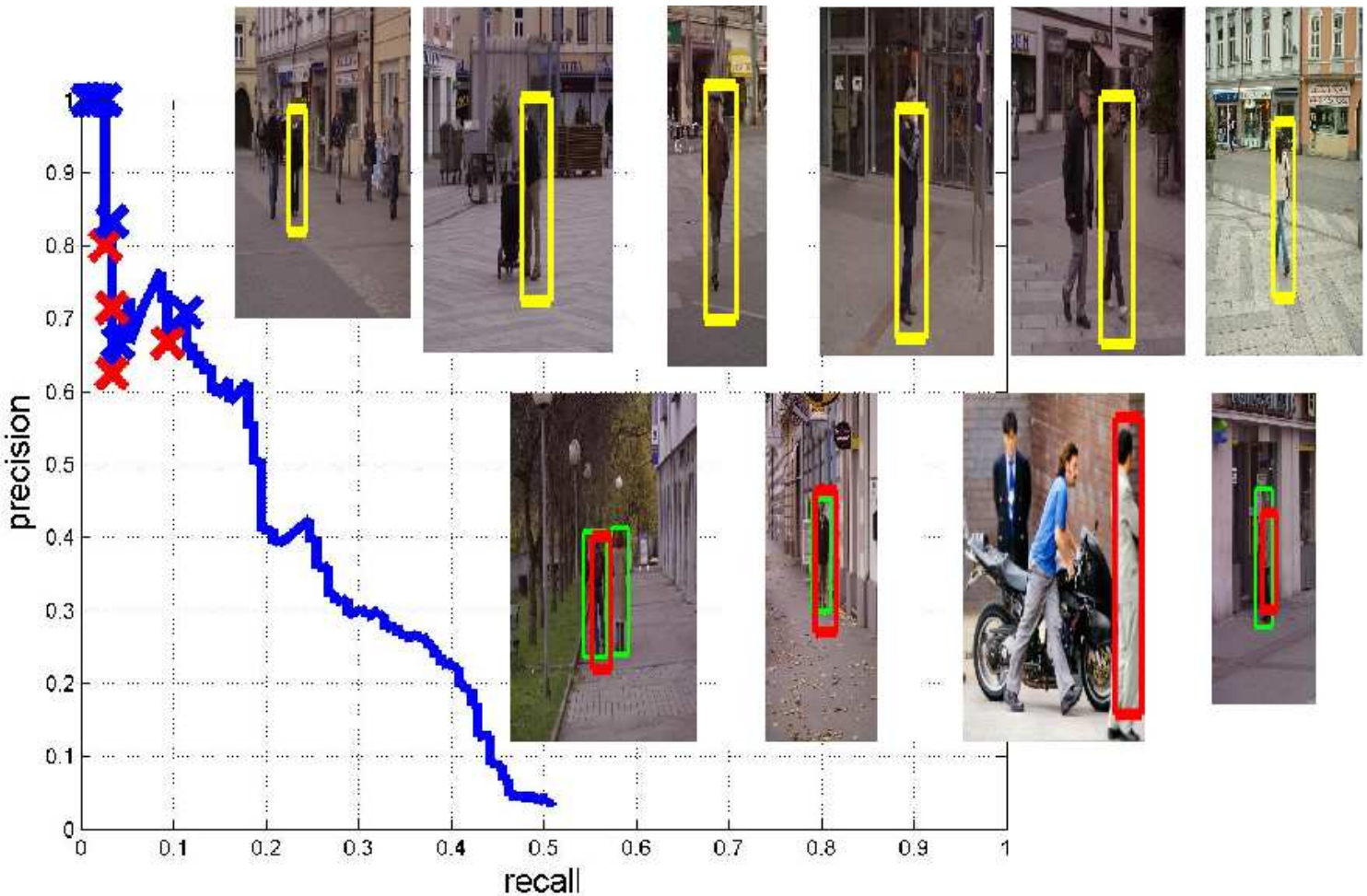
Détection d'objet

- Parcours de toute l'image avec une fenêtre à plusieurs résolution.
- Pour chaque fenêtre on lance AdaBoost
- Clustering des détections obtenus

Détection



Détection



Exemples 3 : Le boosting pour le tracking

FIGURE: Tracking robuste de deux objets

Exemples d'Application

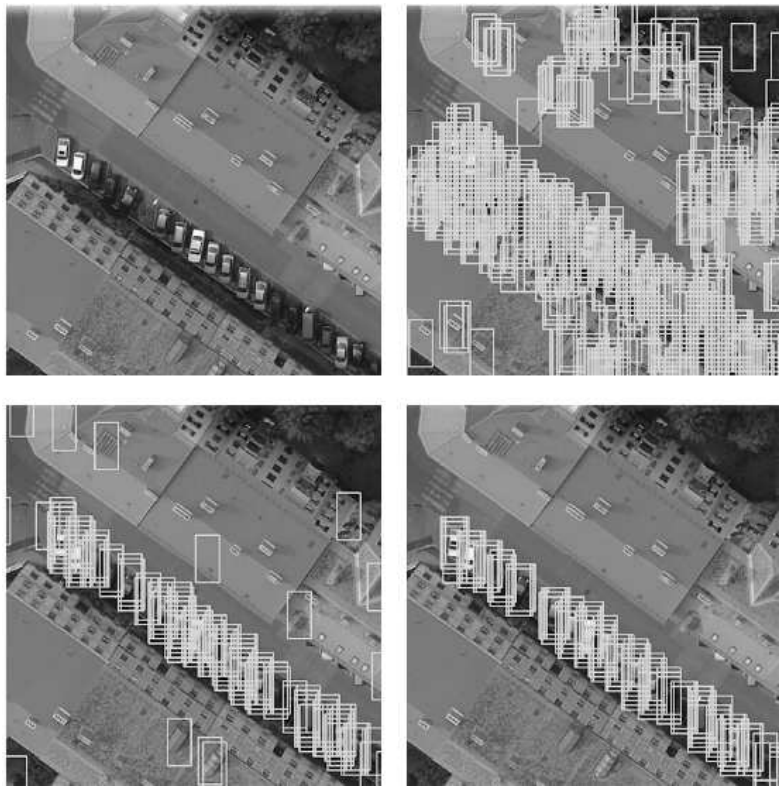
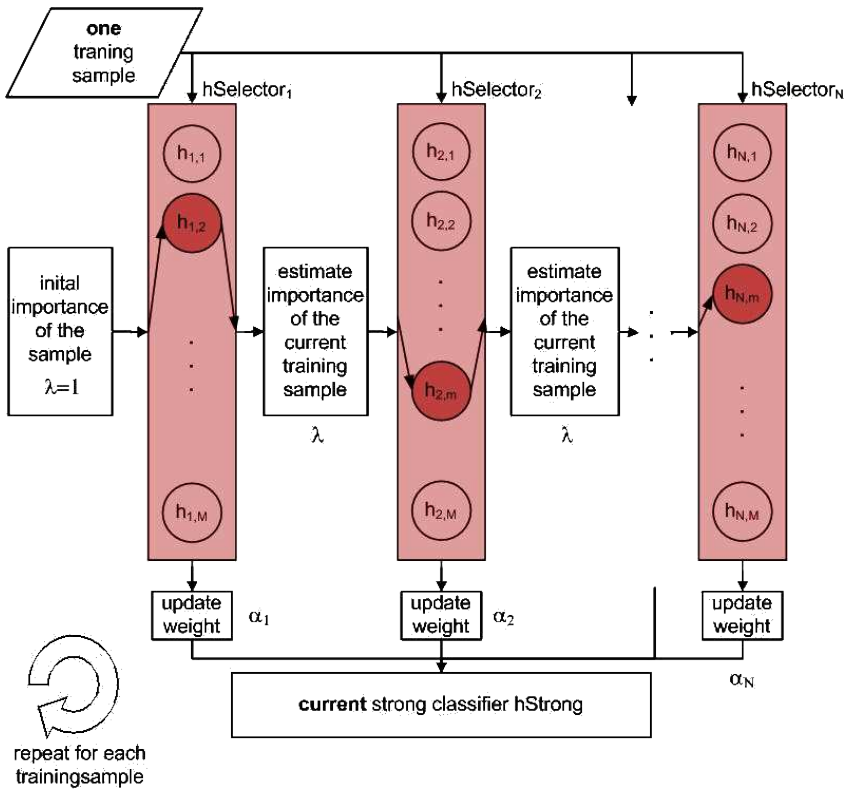


FIGURE: Détection avec 1, 10 et 50 exemples d'apprentisage

Exemples 3 : Le boosting pour le tracking

- Optimisation d'ensembles de classifieurs faibles
- Application au tracking d'objet



Approche de Grabner

Comparaison Adaboost | Online Boost

- AdaBoost : l'ensemble des exemples est utilisé afin de sélectionner un feature
- OnlineBoost : on introduit le Selecteur : sélectionne parmi M classifieurs (ou features) celui qui présente l'erreur estimée minimale.

Algorithme Online Boost

Algorithme

- Pour t=1,...,T faire :
  1. Parcourir la pool de rang t et pour chaque classifieur k faire :
     - Si le classifieur classe correctement l'exemple on a : $\lambda^{correct}_{t,k} += \lambda$ sinon $\lambda^{wrong}_{t,k} += \lambda$
     - Trouver le classifieur minimisant l'erreur de classification :

$$e = \frac{\lambda^{wrong}}{\lambda^{correct} + \lambda^{wrong}}$$

  2. Choisir $\alpha = \frac{1}{2} \ln\left(\frac{1-e_{min}}{e_{min}}\right)$
  3. Mettre à jour λ
     Si le classifieur minimisant l'erreur classe correctement l'exemple d'apprentissage :
     $\lambda = \lambda \cdot \frac{1}{2(1-e_{min})}$ sinon $\lambda = \lambda \cdot \frac{1}{2e_{min}}$

Sommaire

Autres algorithmes de Boosting

AdaBoost Multiclasse : $Y = (x_1, y_1), ..., (x_n, y_n) | x \in X, y \in 1, ..., k$

- AdaBoost.M1 :
  - Mise à jour des poids par : $w_{t+1}(i) = w_t(i)\beta_t^{1-1_{h_t(x_i)\neq y_i}}$ où $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
  - Classifieur final : $H(x) = argmax(\sum_{t=1}^{M} \alpha_t 1_{h_t(x)=y})$
- AdaBoost.M2 :
  - On remplace l'erreur de prédiction par la « pseudo-perte », qui est la moyenne pondérée des probabilités de préférer chaque mauvais label au bon.

Autres algorithmes de Boosting

Variantes antérieur à AdaBoost

Arc x4 (1994)

- Le poid w des exemples est égale à $1 + w^4$, où w est le nombre de classifieurs faibles précédemment sélectionné qui classe mal l'exemple

Autres algorithmes de Boosting

Solution contre l'overfitting en présence de bruit

- Weight Decay (1998) : introduit un facteur de régularisation dans l'erreur
- GentleBoost (1998) : utilisation de l'erreur $\sum_i w_i(h(x_i) - y_i)^2$
- LogitBoost (2000) : utilisation d'une erreur de régression logistic $\sum_t log(1 + exp(-H(x_i)y_i)$
- Regularized AdaBoost (2000) : utilisation de marge souple
- BrownBoost (2001) : éliminer les exemples plusieurs fois mal classifiés, jugé comme du bruit
- WeightBoost (2003) : poids des classifieurs sont dépendant de l'entrée.

Autres algorithmes de Boosting

Réduire le nombre de classifieurs faibles

- FloatBoost (2003) : éliminer les classifieurs faibles les plus mauvais après chaque itération
- JointBoost (2004) : entrainer N classifieurs binaires conjointement en partageant les mêmes features.

Et plus récemment

- Waldboost
- Rankboost
- AdaRank
- AnyBoost
- ...