

Apprentissage

Les méthodes d'apprentissage par ensembles

Alexis LECHERVY



- 1 Le Bagging
- 2 Le Boosting
- 3 Exemple d'application : Détection de visages dans des images

Sommaire

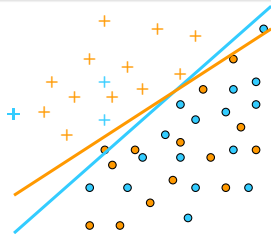
- 1 Le Bagging
- 2 Le Boosting
- 3 Exemple d'application : Détection de visages dans des images

Remarques introductives

Observation :

- Les résultats d'un classifieur dépendent de l'ensemble d'apprentissage utilisé pour l'apprentissage.
- L'utilisation d'un autre ensemble d'apprentissage peut conduire à une autre règle de classification.

⇒ Comment réduire l'impacte de l'ensemble d'apprentissage sur les performances du classifieur final.



Principe

Objectif

- Réduire l'impacte du choix de l'ensemble d'apprentissage sur les performances de classifications.

Idée

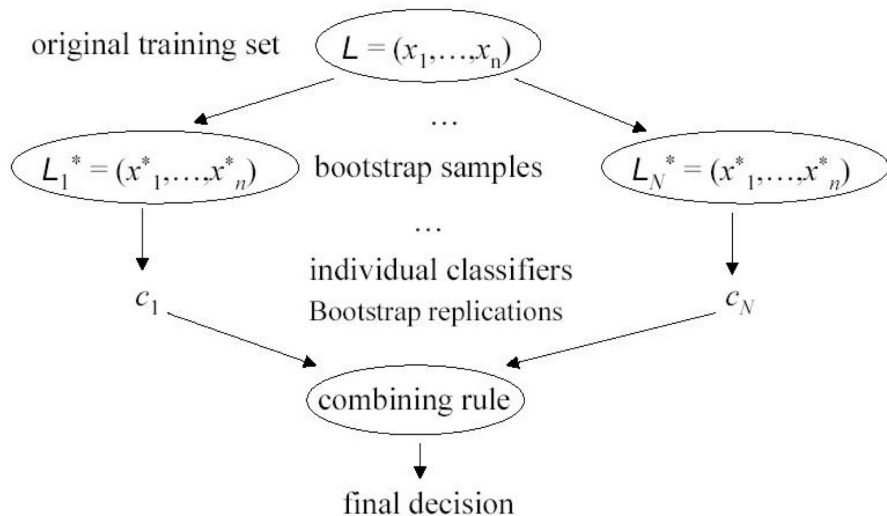
- Combiner les résultats de classification de plusieurs classifieurs appris sur des ensembles d'apprentissages différents.
- Réduire l'erreur de prévision et la variance en moyennent les prévisions de plusieurs modèles indépendants.

Le bagging (Bootstrap AGGregatING)

Le bagging

- Une méthode proposée par *L. Breiman (1996)*.
- Construire un ensemble de classifieurs entraîné sur une réplique différente de la base d'apprentissage.
- Les répliques de la base d'apprentissage sont très peu différentes de la base d'origine mais suffisamment diverses pour obtenir des classifieurs différents, qu'on va pouvoir combiner.
- On agrège ensuite les résultats de tout les classifieurs pour renvoyer un résultat unique.

Algorithme de bagging



Comment réaliser le Bootstrap ?

Bootstrap

Les différents ensembles d'apprentissage sont obtenus par Bootstrap :

- Un échantillon $L_i^* = (x_1^*, \dots, x_n^*)$ est le résultat d'un tirage aléatoire de taille n dans l'ensemble d'apprentissage initial. C'est un tirage avec remise.
- Chaque exemple de l'ensemble d'apprentissage initial peut apparaître aucune, une ou plusieurs fois.

Comment effectuer la combinaison des différentes règles de décisions ?

Agrégation des différents classifieurs

- Moyenne :

$$F(x) = \frac{1}{T} \sum_{t=0}^T f_t(x)$$

- Vote majoritaire :

$$F(x) = \arg \max_c \sum_{t=0}^T 1_{f_t(x)=c}$$

Avantages et Inconvénients

Avantages

- Simple à mettre en place.
- S'adapte facilement à n'importe quelles méthodes d'apprentissage.
- Permet de réduire l'impacte du choix de l'ensemble d'apprentissage sur les résultats de classification.

Inconvénients

- Calcul important pour évaluer un nombre important de classifieur.
- Nécessite de stocker tous les modèles de la combinaison pour évaluer un nouvel exemple.
- Le modèle final n'est pas facilement interprétable. Approche de type boîte noire.

Sommaire

- 1 Le Bagging
- 2 Le Boosting
 - Principes généraux sur le Boosting
 - Le Boosting probabiliste
 - AdaBoost
- 3 Exemple d'application : Détection de visages dans des images

Remarques générales

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



Exemple : La prédiction de courses hippiques



Comment gagner aux courses ?

1^{re} solution : Trouver un expert des paris

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

2^{ème} 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 étape ?

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 :

- ❶ 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 $1 - \delta$
- ❷ PAC apprenable au sens faible si :
 - Pareil mais il suffit juste qu'il existe un $\gamma \leq \frac{1}{2}$ tel que l'erreur du classifieur soit inférieur à $\frac{1}{2} - \gamma$ avec une probabilité $1 - \delta$.
- ❸ 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

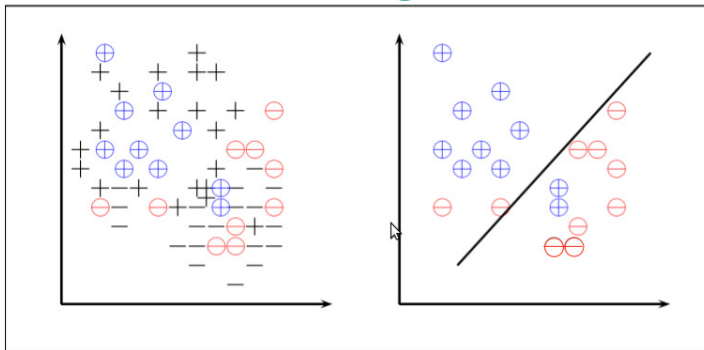
- 1 On obtient d'abord une première hypothèse h_1 sur un sous-échantillon S_1 d'apprentissage de taille $m_1 < m$ (m étant la taille de S l'échantillon d'apprentissage disponible).
- 2 On apprend alors une deuxième hypothèse h_2 sur un échantillon S_2 , de taille m_2 , choisi dans $S - S_1$ dont la moitié 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 : $H = \text{vote majoritaire}(h_1, h_2, h_3)$

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

Boosting : Début

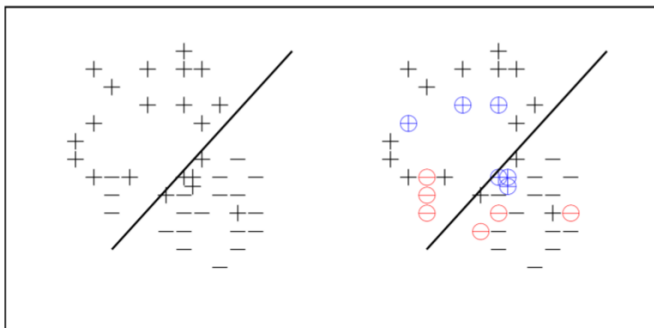


A gauche : l'ensemble d'apprentissage S et le sous-ensemble S_1 (points *rouges* ou *bleus* entourés).

A droite : l'ensemble S_1 et la droite C_1 apprise sur cet ensemble.

Boosting : Deuxième étape

Boosting : Suite

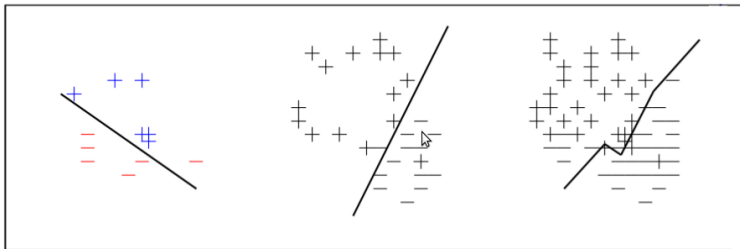


A gauche : l'ensemble d'apprentissage $S - S_1$ et la droite C_1 apprise sur S_1 .

A droite : un ensemble S_2 inclus dans $S - S_1$ 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 un 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

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 étape en fonction des poids précédent et des résultats du classifieurs sélectionnés.
- Au départ, tous les exemples ont la même importance.

Adaboost

Algorithme

- ① Une base d'exemple $(x_1, y_1), \dots, (x_m, y_m)$
- ② Avec des labels $y_i \in \{-1, 1\}$
- ③ Pour tout $t = 1, \dots, T$
 - 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 :

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

- 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

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

Adaboost : le classifieur final

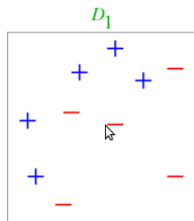
$$H(x) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(x) \right)$$

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.).

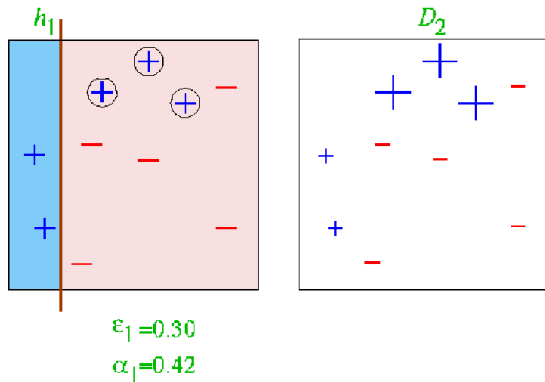
Exemple "Toy"

Principe

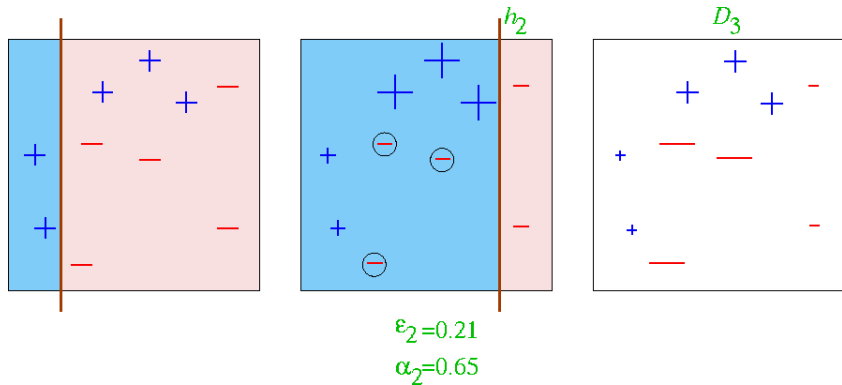
- On considère l'exemple «Toy», de points 2D dans un domaine D_1
- Les données sont soit positives, soit négatives (deux catégories)
- Les classifieurs faibles sont les droites horizontales et verticales



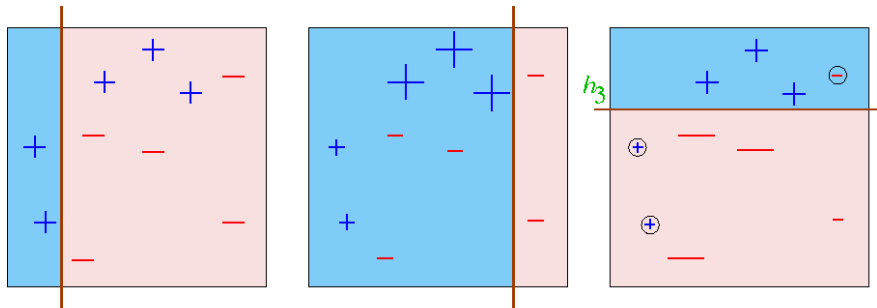
Exemple "Toy" : étape 1



Exemple "Toy" : étape 2



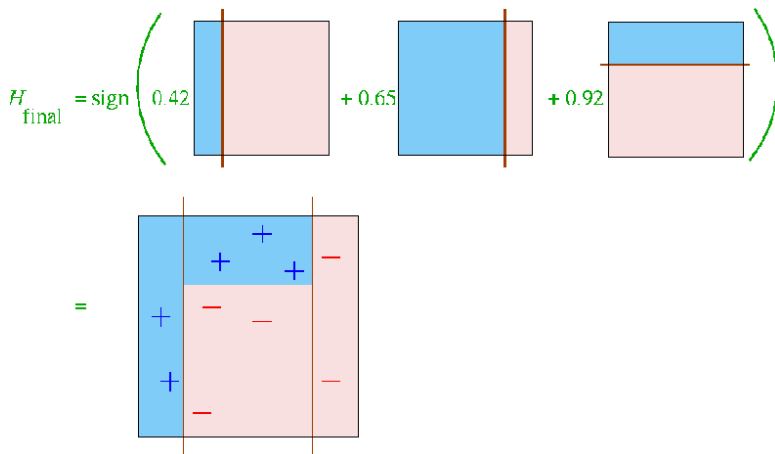
Exemple "Toy" : étape 3



$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

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 :

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

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

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

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

$$err_H \leq \prod_t Z_t$$

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 :

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

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 classifieurs faibles)
- m le nombre d'exemples

L'erreur en généralisation (théorie)

Erreur en généralisation

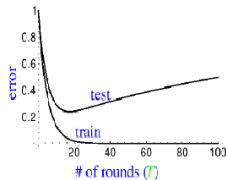
L'erreur en généralisation du classifieur H finale peut être bornée par :

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

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

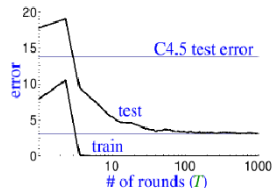
L'erreur en généralisation (pratique)

Comportement classique



- overfitting
- 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

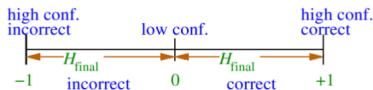
Principe

- 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 à :

la fraction votant correctement - fraction ne votant pas correctement

- 4 Pour le boosting la marge est définie 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 $\forall \theta > 0$

$$\text{erreur} \leq \Pr(\text{marge} \leq \theta) + O\left(\frac{\sqrt{\frac{d}{m}}}{\theta}\right)$$

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

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 dépendent du type de classifieurs faibles choisie
- Les performances dépendent 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

Sommaire

- 1 Le Bagging
- 2 Le Boosting
- 3 Exemple d'application : Détection de visages dans des images

Rappel historique : le boosting et la détection de visage

- 1990 Schapire le Boosting : classifieurs faibles \rightarrow 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

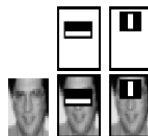
- Utilisation d'un AdaBoost classique
- Propose de nouveaux classifieurs faibles basés sur les descripteurs de Haar
- Introduit le concept de cascade

Descripteur de Haar

Principe

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

Avec $f_i(x) = \sum_j r_{i,blanche} - \sum_j r_{i,noire}$



Exemples



Descripteurs de Haar dans une fenêtre 24x24

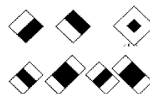
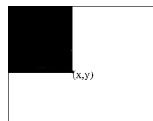


Image intégrale

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 :

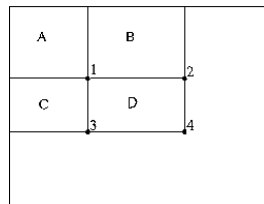
$$P_1 = A$$

$$P_2 = A + B$$

$$P_3 = A + C$$

$$P_4 = A + B + C + D$$

$$D = P_4 + P_1 - (P_2 + P_3)$$



Cascade : Origine

Observation

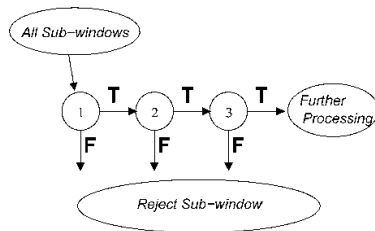
- Une image est constituée **majoritairement** de sous-fenêtres **négatives**
- La détection d'un objet **positif** est un événement **rare**.

⇒ Il faut éliminer rapidement les sous-images négatives

Cascade : Principe

Principe

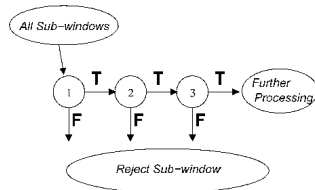
- Une **cascade** de classifieurs est un **arbre de décision** dégénéré dans laquelle, chaque étape est entraînée pour détecter un maximum d'objets intéressants tout **en rejetant une certaine fraction** des objets non-intéressants.



Cascade : Principe

Principe

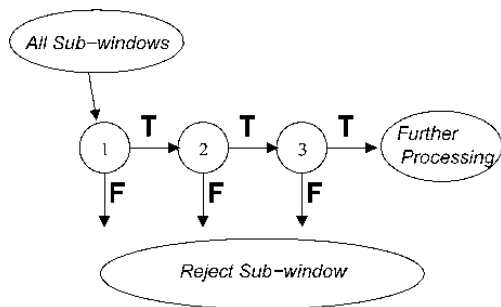
- Une sous-image doit passer tous les classifieurs fort afin d'être acceptée comme visage.
- Le déclenchement de tous les classifieurs par un résultat positif devient ainsi un événement rare.
- Il faut que le nombre de sous-images éliminées dès les premières étapes de la cascade soit très élevé.



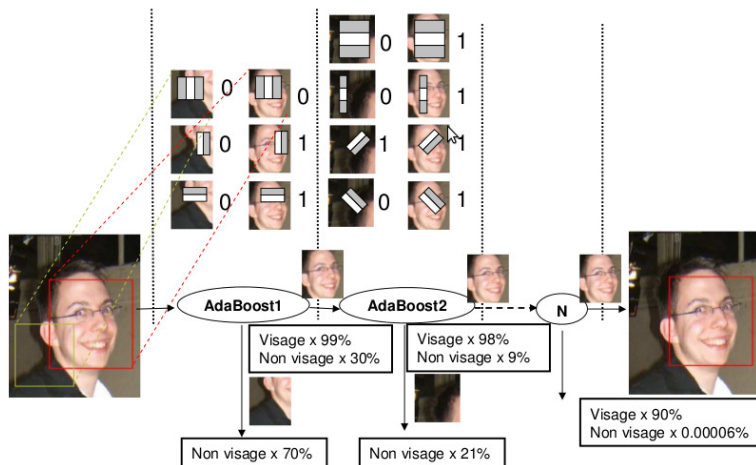
Cascade : Principe

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

