Théorie du Boosting - partie II

Gilles Blanchard[1]

Fraunhofer FIRST.IDA
Berlin, Allemagne

28 Mai 2008 – EPIT 2008

Plan

Petit rappel de notations...

- $(\mathbf{x}_i)$: les observations
- y_i : les valeurs à prédire :
  - $y_i \in \{-1, 1\}$: classification binaire
  - $y_i \in \{1, \ldots, K\}$: classification multiclasses
  - $y_i \in \mathbb{R}$: régression
- m : le nombre d'exemples en apprentissage
- L : un algorithme apprenant faible

Rappel : AdaBoost

Algorithme (AdaBoost)

Initialisation

$D_1(i) = 1/m, i = 1, \ldots, m$ // Poids initiaux

Pour $t = 1, \ldots, T$ **:**

$h_t = L(LS, \mathbf{D}_t)$ // Apprenant faible avec poids $\mathbf{D_t}$

$\widehat{\varepsilon}_t = \sum_{i:h(x_i)\neq y_i} D_t(i)$ // Erreur pondérée de h_t

$\alpha_t = \frac{1}{2} \ln \frac{1-\widehat{\varepsilon}_t}{\widehat{\varepsilon}_t}$ // Coefficient de h_t dans l'ensemble

Pour $i = 1, \ldots, m$ **:** // Renforcement du poids des exemples

Si $h_t(x_i) \neq y_i$: // mal classés

$\widetilde{D}_{t+1}(i) = D_t(i) \frac{1-\widehat{\varepsilon}_t}{\widehat{\varepsilon}_t}$

Sinon, $\widetilde{D}_{t+1}(i) = D_t(i)$

Fin

$D_{t+1}(i) = \widetilde{D}_{t+1}(i) / (\sum_i \widetilde{D}_{t+1}(i))$ // Normalisation

Fin

$f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ // Vote pondéré

$H_T(\mathbf{x}) = \mathrm{sign}(f_T(\mathbf{x}))$ // Classifieur final

Rappel : AdaBoost

Algorithme (AdaBoost)

Initialisation

$D_1(i) = 1/m, i = 1, \ldots, m$ // Poids initiaux

Pour $t = 1, \ldots, T$ **:**

$h_t = L(LS, \mathbf{D}_t)$ // Apprenant faible avec poids $\mathbf{D_t}$

$\widehat{\varepsilon}_t = \sum_{i:h(x_i)\neq y_i} D_t(i)$ // Erreur pondérée de h_t

$\alpha_t = \frac{1}{2} \ln \frac{1-\widehat{\varepsilon}_t}{\widehat{\varepsilon}_t}$ // Coefficient de h_t dans l'ensemble

Pour $i = 1, \ldots, m$ **:** // Renforcement du poids des exemples

// mal classés

$$\widetilde{D}_{t+1}(i) = D_t(i) \exp(-\alpha_t h_t(\mathbf{x}_i) y_i)$$

Fin

$D_{t+1}(i) = \widetilde{D}_{t+1}(i) / (\sum_i \widetilde{D}_{t+1}(i))$ // Normalisation

Fin

$f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$ // Vote pondéré

$H_T(\mathbf{x}) = \mathrm{sign}(f_T(\mathbf{x}))$ // Classifieur final

Rappels (suite)

Un rôle crucial est joué par la fonction “exponentielle de la marge” :

$$R_m(f) = \frac{1}{m}\sum_{i=1}^{m} \exp(-f(\mathbf{x}_i)y_i) \ :$$

- $\mathcal{E}_m(f) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}\left\{f(\mathbf{x}_i) \neq y_i\right\} \leq R_m(f)\,;$
- $D_t(i) \propto \exp(-f_{t-1}(\mathbf{x}_i)y_i)$;
- Satisfait :

$$\begin{aligned}\frac{\partial R_m(f_t + \alpha h)}{\partial \alpha} &= \frac{-1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)y_i \exp(-y_i(f_t(\mathbf{x}_i) + \alpha h(\mathbf{x}_i))) \\ &\propto -\sum_{i:h(x_i)=y_i} D_t(i)e^{-\alpha} + \sum_{i:h(x_i)\neq y_i} D_t(i)e^{\alpha} \\ &= -e^{-\alpha}(1-\widehat{\varepsilon}_t) + e^{\alpha}\widehat{\varepsilon}_t\,.\end{aligned}$$

Rappels (suite)

$$R_m(f) = \frac{1}{m}\sum_{i=1}^{m} \exp(-f(\mathbf{x}_i)y_i) \ :$$

- On a :
$$\frac{\partial R_m(f_t + \alpha h)}{\partial \alpha}\Big|_{\alpha=0} \propto (2\widehat{\varepsilon}_t(h) - 1)\,;$$
- Pour tout h_t retourné par l'apprenant faible, le coefficient α_t est choisi pour minimiser $R_m(f_{t-1} + \alpha_t h_t)$;
- $R_m(f_T) = 2^m \left(\Pi_{i=1}^m \widehat{\varepsilon}_t(1-\widehat{\varepsilon}_t)\right)^{\frac{1}{2}}$;

Descente de gradient par coordonnée par coordonnée

Supposons que l'apprenant faible choisisse à chaque étape t parmi une famille fixée de classifieurs $\mathcal{H} = \{h^1, h^2, \ldots, h^K\}$ celui qui a la plus petite erreur pondérée $\widehat{\varepsilon}_t(h)$. Alors :

- le choix du classifieur faible h_t est équivalent à la minimisation

$$\min_{h\in\mathcal{H}} \frac{\partial R_m(f_t + \alpha h)}{\partial \alpha}\Big|_{\alpha=0} \quad [\propto \min_{h\in\mathcal{H}}(2\widehat{\varepsilon}_t(h) - 1)\,];$$

- le choix de (h_t, α_t) est aussi équivalent à la minimisation

$$\min_{h\in\mathcal{H},\alpha\in\mathbb{R}} R_m(f_t + \alpha h) \quad [= R_m(f_t) \min_{h\in\mathcal{H}} 2\sqrt{\widehat{\varepsilon}_t(1-\widehat{\varepsilon}_t)}\,];$$

  Le choix de h semble aussi être bon si $\widehat{\varepsilon}_t$ est proche de 1 ? ? ?

- si le classifieur faible ne minimise qu'approximativement l'erreur pondérée, on peut aussi interpréter cela comme une approximation de l'un des deux problèmes ci-dessus.

Ainsi, AdaBoost peut s'interpréter comme une descente de gradient par coordonnée pour un modèle addititif ($f = \sum_{h\in\mathcal{H}} \alpha_h h$) en utilisant la fonction de perte $\ell(f, \mathbf{x}, y) = \exp(-f(\mathbf{x})y)$.

Généralisations du boosting

Avantage de l'interprétation de AdaBoost comme une descente de gradient : on peut suivre ce point de vue pour généraliser l'algorithme à d'autres situations, ou en proposer des variantes :

- Classification avec plus de 2 classes
- Autres fonctions de perte
- Boosting pour la régression
- Boosting pour l'apprentissage d'ordonnancements (**ranking**)
- Alternatives non-itératives d'AdaBoost minimisant directement la fonction de perte

Désavantage : on s'éloigne alors de l'interprétation initiale du boosting, notamment l'interprétation via la théorie des jeux.

Généralisations du Boosting, suite

D'une façon abstraite, on peut donc interpréter le Boosting comme ayant pour but la minimisation de la fonctionnelle

$$R_m(f) = \frac{1}{m}\sum_{i=1}^{m} \ell(f(\mathbf{x}_i), y_i)\,, \qquad (*)$$

pour des fonctions f de la forme $f = \sum_{i=1}^{T} \alpha_i h_i$, avec $h_i \in \mathcal{H}$ pour tout i .
Les stratégies pour ce faire peuvent être très variées :

- dans l'esprit d'AdaBoost, de façon itérative, en choisissant à chaque étape t un nouveau (h_t, α_t) qui s'ajoute à l'estimée courante.
  $\longrightarrow$ procédures itératives
- on peut envisager d'attaquer directement l'optimisation de (*)
  $\longrightarrow$ procédures directes
- dans ce second cas on considère alors soit une contrainte du type $\|\alpha\|_1 \leq M$, soit une version régularisée du type

$$\min_{\alpha} R_m(f) + C\,\|\alpha\|_1 \ . \qquad (**)$$

- . . . le Boosting s'identifie alors avec le LASSO [Tibshirani, 1996].

Questions générales

- Quelles fonctions de perte $\ell(f(x), y)$ sont-elles “acceptables” – en quel sens ?
- Comment est contrôlée la complexité dans les procédures itératives ? (Nombre d’itérations)
- Comment est contrôlée la complexité dans les procédures directes ? (La norme L_1 des coefficients)
- Ces procédures sont-elles consistantes ?

Exemple simple : apprenant faible avec possibilité d'abstention

- Cadre : classification binaire, mais un apprenant faible est autorisé à ne pas faire de prédiction en certains points ($h(\mathbf{x}) = 0$) .
- Exemple d'application : données partiellement observées avec coordonnées manquantes. Un classifieur faible faisant appel à une coordonnée non observée s'abstient sur les exemples correspondants.
- Considérons à nouveau la fonction de perte

$$R_m(f) = \frac{1}{m} \sum_{i=1}^{m} \exp(-f(\mathbf{x}_i) y_i) \,,$$

et suivons le principe du Boosting pour la minimiser itérativement.

Apprenant faible avec possibilité d'abstention

- Supposons qu'on a $D_t(i) \propto \exp(-f_{t-1}(\mathbf{x}_i)y_i)$ où $f_t = \sum_{i=1}^{t} \alpha_t h_t$;
- pour un nouveau classifieur faible h_t notons W_+, W_- et W_0 les proportions pondérées de classifications correctes, incorrectes et avec abstention ; on a

$$R_m(f_{t-1} + \alpha h_t) = \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i(f(\mathbf{x}_i) + \alpha h(\mathbf{x}_i)))$$
$$\propto W_0 + e^{\alpha} W_- + e^{-\alpha} W_+ .$$

- Le choix optimal de coefficient est alors $\alpha_t = \frac{1}{2} \ln\left(\frac{W_+}{W_-}\right)$;
- Et la mise à jour des poids :

$$\widetilde{D}_{t+1}(i) = D_t(i).\left(\frac{W_-}{W_+}\right)^{\kappa_i/2},$$

où $\kappa_i = 1, -1, 0$ pour les exemples corrects, incorrects et avec abstention, respectivement.

Boosting multiclasses

Il existe plusieurs (!) généralisations de AdaBoost au cas multiclasse, par exemple dans [Schapire et Singer, 1999]. En voici une très simple :

- Il est déconseillé d'appliquer directement AdaBoost à un problème à $K > 2$ classes ! En effet, comme l'erreur ε_t peut être moins que $\frac{1}{2}$, cela mène à des incohérences pour le choix des poids et des coefficients.
- Utilisons le point de vue "descente de gradient itérative" pour proposer une extension à K classes :
- Y est un vecteur de taille K , avec coordonnée égale à 1 pour la vraie classe et 0 sinon.
- Un classifieur faible h est un vecteur de taille K, avec coordonnée égale à 1 pour la classe prèdite et $-1/(K-1)$ sinon.
- La fonction de perte à minimiser est

$$R_m(F) = \frac{1}{m}\sum_{i=1}^{m} \exp(-\langle Y_i, F(\mathbf{x}_i)\rangle)\,,$$

et le classsifieur associé est $H(F(\mathbf{x})) = \operatorname{Arg}\operatorname{Max}_k F^{(k)}(\mathbf{x})$.

- ($R_m(F)$ n'est plus une borne sur l'erreur de classification de $H(F)$? Est-ce grave ?)

AdaBoost multiclasses

Algorithme

Initialisation

$D_1(i) = 1/m, i = 1, \ldots, m$ // Poids initiaux

Pour $t = 1, \ldots, T$ **:**

$h_t = L(LS, \mathbf{D}_t)$ // Apprenant faible avec poids $\mathbf{D_t}$

$\widehat{\varepsilon}_t = \sum_{i:h(x_i)\neq y_i} D_t(i)$ // Erreur pondérée de h_t

$\alpha_t = \ln\left((K-1)\frac{1-\widehat{\varepsilon}_t}{\widehat{\varepsilon}_t}\right)$ // Coefficient de h_t dans l'ensemble

Pour $i = 1, \ldots, m$ **:** // Renforcement du poids des exemples

Si $h_t(x_i) \neq y_i$: // mal classés

$\widetilde{D}_{t+1}(i) = D_t(i)(K-1)\frac{1-\widehat{\varepsilon}_t}{\widehat{\varepsilon}_t}$

Sinon, $\widetilde{D}_{t+1}(i) = D_t(i)$

Fin

$D_{t+1}(i) = \widetilde{D}_{t+1}(i)/(\sum_i \widetilde{D}_{t+1}(i))$ // Normalisation

Fin

$\forall y \in \mathbf{Y} : f_T(\mathbf{x}, y) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}, y)$ // Vote pondéré pour chaque classe

$H_T(\mathbf{x}) = \underset{y\in\mathbf{Y}}{\mathrm{Arg\,Max}}\, f_T(\mathbf{x}, y)$ // Classifieur final

Exercices. . .

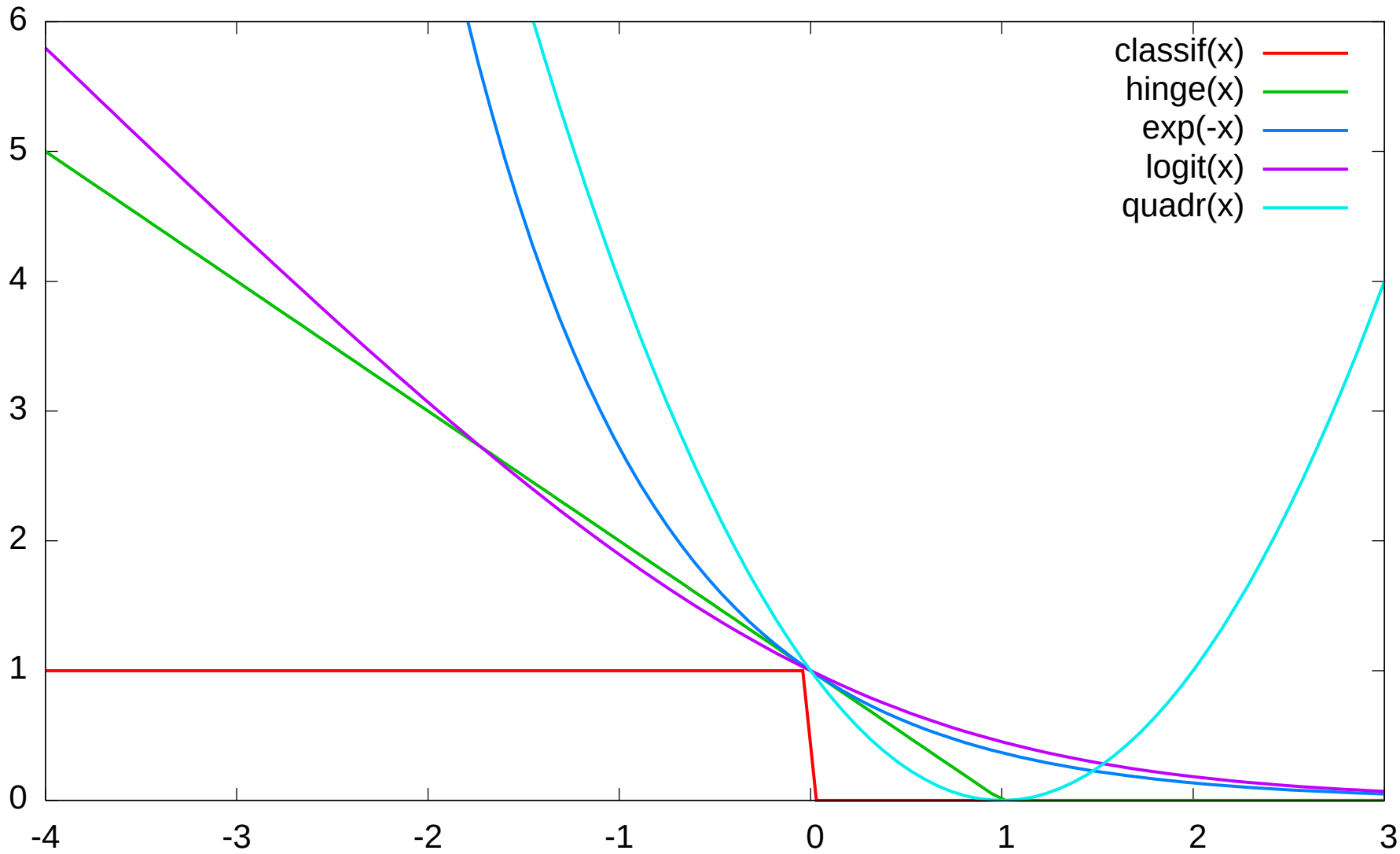
1. Dériver l'algorithme multiclasse à partir de la fonction de perte multiclasse proposée (et du "codage" formel de la réponse des classifieurs faibles).

2. (Exercice ouvert) Supposons que l'apprenant faible soit un algorithme qui n'accepte en entrée que les problèmes de classification binaire. Comment utiliser le Boosting pour obtenir un classifieur final multiclasse ?

Alternatives à la perte exponentielle en classification

Fonctions de la forme $\ell(x, y) = \varphi(-xy)$:

Nom	$\varphi(x)$
Exponentiel	$\exp(-x)$
Logit	$\log_2(1 + \exp(-x))$
Quadratique	$(1 - x)^2$
Quadratique tronqué	$(1 - x)_+^2$
Hinge (SVM)	$(1 - x)_+$


6
5
4
3
2
1
0
-4
-3
-2
-1
0
1
2
3
classif(x)
hinge(x)
exp(-x)
logit(x)
quadr(x)

Boosting (itératif) avec fonction de perte arbitraire

- Appliquons le point de vue "descente de gradient itératif" pour approcher la minimisation de

$$R_m(f) = \frac{1}{m}\sum_{i=1}^{m} \ell(f(\mathbf{x}_i), y_i)\,, \qquad (*)$$

où l'on suppose que $\ell(\cdot,\cdot)$ est une fonction de perte dérivable en la première variable .

- Pour cela calculons la dérivée de $R_m(f)$ dans la "direction" d'un nouveau h :

$$\begin{aligned}\frac{\partial R_m(f_t+\alpha h)}{\partial \alpha}\Big|_{\alpha=0} &= \frac{1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)\ell'(f_t(\mathbf{x}_i), y_i) \\ &= \frac{1}{m}\left\langle h(\mathbf{X}), \ell'(f_t(\mathbf{X}), \mathbf{Y})\right\rangle\,.\end{aligned}$$

Boosting (itératif) avec fonction de perte arbitraire

2 façons d'interpréter la quantité $\langle h(\mathbf{X}), \ell'(f_t(\mathbf{X}), \mathbf{Y})\rangle$ (à maximiser en h) :

En classification :
supposons $\ell(f(\mathbf{x}, y)) = \varphi(-yf(\mathbf{x}))$, et $h \in \{-1, 1\}$ est un classifieur, alors

$$\begin{aligned}\langle h(\mathbf{X}), \ell'(f_t(\mathbf{X}), \mathbf{Y})\rangle &= -\sum_{i=1}^{m} y_i h(\mathbf{x}_i)\varphi'(-y_i f_t(\mathbf{x}_i)) \\ &= -2\sum_{i=1}^{m} \mathbf{1}\{h(\mathbf{x}_i) \neq y_i\}\varphi'(-y_i f_t(\mathbf{x}_i)) + \sum_{i=1}^{m} \varphi'(-y_i f_t(\mathbf{x}_i)) .\end{aligned}$$

Équivalent à demander à l'apprenant faible de minimiser l'erreur pondérée de h avec les poids $\varphi'(-y_i f_t(\mathbf{x}_i))$. (Qui sont positifs si φ est, de plus, croissante)

Boosting (itératif) avec fonction de perte arbitraire

Cas général :
Si on considère la philosophie de suivre le plus fort gradient de la fonction de perte, il faut fixer une normalisation des fonctions pouvant être proposées par l'apprenant faible. Supposons que h est normalisé de façon que $\frac{1}{m}\sum_{i=1}^{m} h(\mathbf{x}_i)^2 = \frac{1}{m}\langle h, h\rangle = 1$.

Lemme

F étant un vecteur fixé et $\mathcal{H}$ une famille de "candidats", la maximisation de $\left|\left\langle \frac{h}{\|h\|}, F\right\rangle\right|$ pour $h \in \mathcal{H}$ est obtenue pour le même h que la minimisation de $\|F - \lambda h\|^2$ en $(h \in \mathcal{H}, \lambda \in \mathbb{R})$.

(Preuve : il suffit de l'écrire)
Conclusion : avec la normalisation ci-dessus, viser la minimisation du gradient est équivalent à demander à l'apprenant faible de minimiser l'erreur quadratique en régression pour le vecteur $(\ell'(-yf_t(\mathbf{x}_i), y_i))$.

Boosting (itératif) avec fonction de perte arbitraire

Algorithme (AdaBoost généralisé)

Initialisation

$f_0 = 0$

Pour $t = 1, \ldots, T$ **:**

$W_t(i) = \ell'(f_{t-1}(\mathbf{x}_i), y_i),\ i = 1, \ldots, n$ // "Cibles" ou "Poids"

$h_t = L(LS, \mathbf{W}_t)$ // Apprenant faible avec cibles $\mathbf{W_t}$

$\alpha_t = \underset{\alpha \in \mathbb{R}}{\text{Arg Min}}\, R_m(f_{t-1} + \alpha h_t)$ // Recherche linéaire

$f_t = f_{t-1} + \alpha_t h_t$

Fin

Retourner f_T

- En classification, l'apprenant faible doit minimiser l'erreur pondérée.
- Dans le cas général, l'apprenant faible doit trouver un h_t réalisant une régression aux moindres carrés sur les cibles.
- On peut aussi envisager de demander à l'apprenant faible de minimiser directement en (h, α) la perte $R_m(f_{t-1} + \alpha h)$, mais cela empêche d'utiliser comme apprenant faible un algorithme "standard".

Boosting pour apprentissage d'ordonnancements

- une fonction d'ordonnancement est représentée par une fonction $f : \mathcal{X} \to \mathbb{R}$.
- on a un ensemble d'apprentissage $(\mathbf{x}_i)$; l'information dont on dispose est la préférence entre $(\mathbf{x}_i)$ et $(\mathbf{x}_j)$ pour tout couple $(\mathbf{x}_i, \mathbf{x}_j)$.
- on considère la fonction de perte

$$R_m(f) = \sum_{i,j:\mathbf{x}_i \succ \mathbf{x}_j} \exp(-(f(\mathbf{x}_i) - f(\mathbf{x}_j)))\,.$$

$$\frac{\partial R_m(f+\alpha h)}{\partial \alpha} = \sum_{\mathbf{x}_i \succ \mathbf{x}_j} (h(\mathbf{x}_j) - h(\mathbf{x}_i)) D(\mathbf{x}_i, \mathbf{x}_j)$$

$$= \sum_{\mathbf{x}} h(\mathbf{x}) \left(\sum_{\mathbf{x}' \succ \mathbf{x}} D(\mathbf{x}', \mathbf{x}) - \sum_{\mathbf{x} \succ \mathbf{x}'} D(\mathbf{x}, \mathbf{x}') \right) ,$$

où $D(\mathbf{x}, \mathbf{x}') = \exp(-(f(\mathbf{x}) - f(\mathbf{x}')))$. Cas similaire aux précédents !

Boosting (itératif) avec shrinkage

Par analogie avec des méthodes additives utilisées en statistiques, il peut être utile d'utiliser en plus un "coefficient de shrinkage" $\nu \in (0,1)$:

Algorithme (Boosting généralisé avec "shrinkage" $\nu \in (0,1)$)

Pour $t = 1, \ldots, T$ **:**

$W_t(i) = \ell'(f_{t-1}(\mathbf{x}_i), y_i),\ i = 1, \ldots, n$ // "Cibles" ou "Poids"

$h_t = L(LS, \mathbf{W}_t)$ // Apprenant faible avec cibles $\mathbf{W_t}$

$\alpha_t = \underset{\alpha \in \mathbb{R}}{\text{Arg Min}}\, R_m(f_{t-1} + \alpha h_t)$ // Recherche linéaire

$f_t = f_{t-1} + \nu \alpha_t h_t$

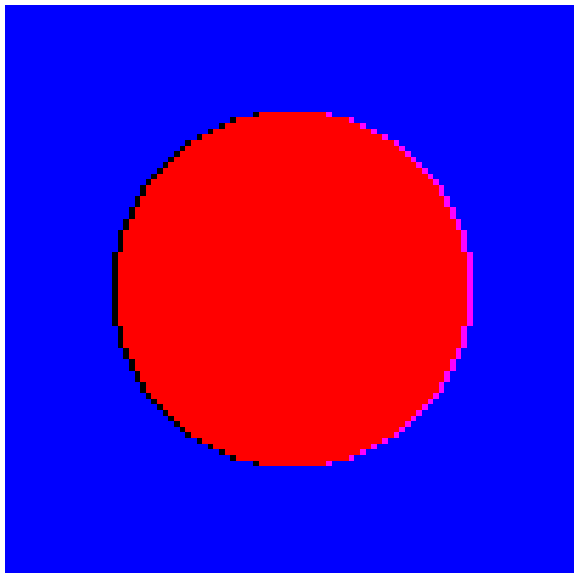
Fin

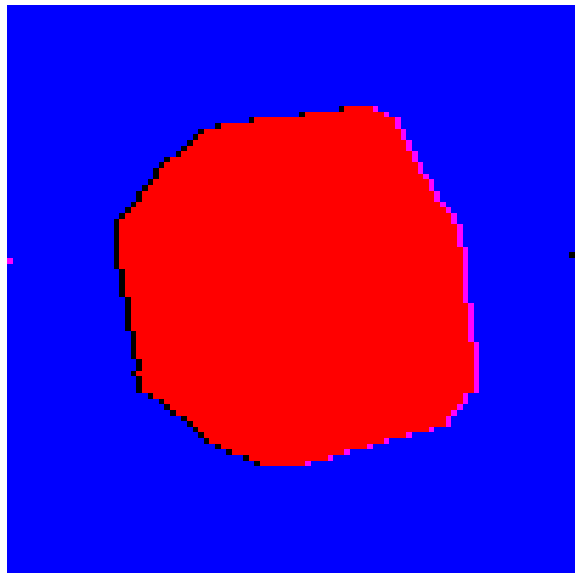
Retourner f_T

- Recommandé si le cas standard ($\nu = 1$) sur-apprend trop rapidement.
- Le "shrinkage" est donc aussi une forme de régularisation.
- Pour Adaboost, cela revient juste à changer l'étape de repondération des exemples : on multiplie les poids des exemples mal classés par $\left(\frac{1-\widehat{\varepsilon}_t}{\widehat{\varepsilon}_t}\right)^{\nu}$ (avant renormalisation).

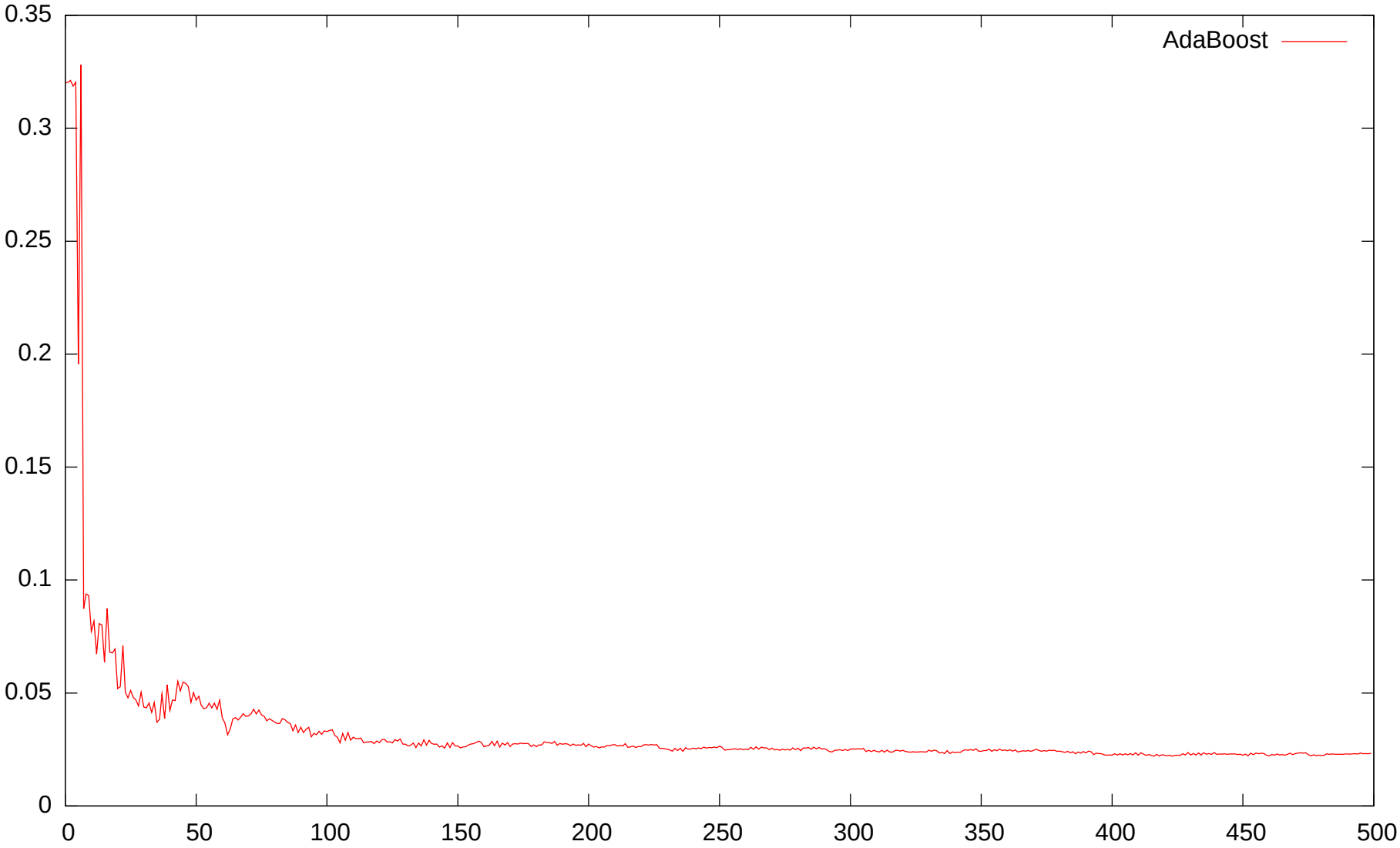
Exemples

- Exemple simple (cercle)
- Apprenant faible : meilleur d'entre 1000 classifieurs linéaires tirés au hasard

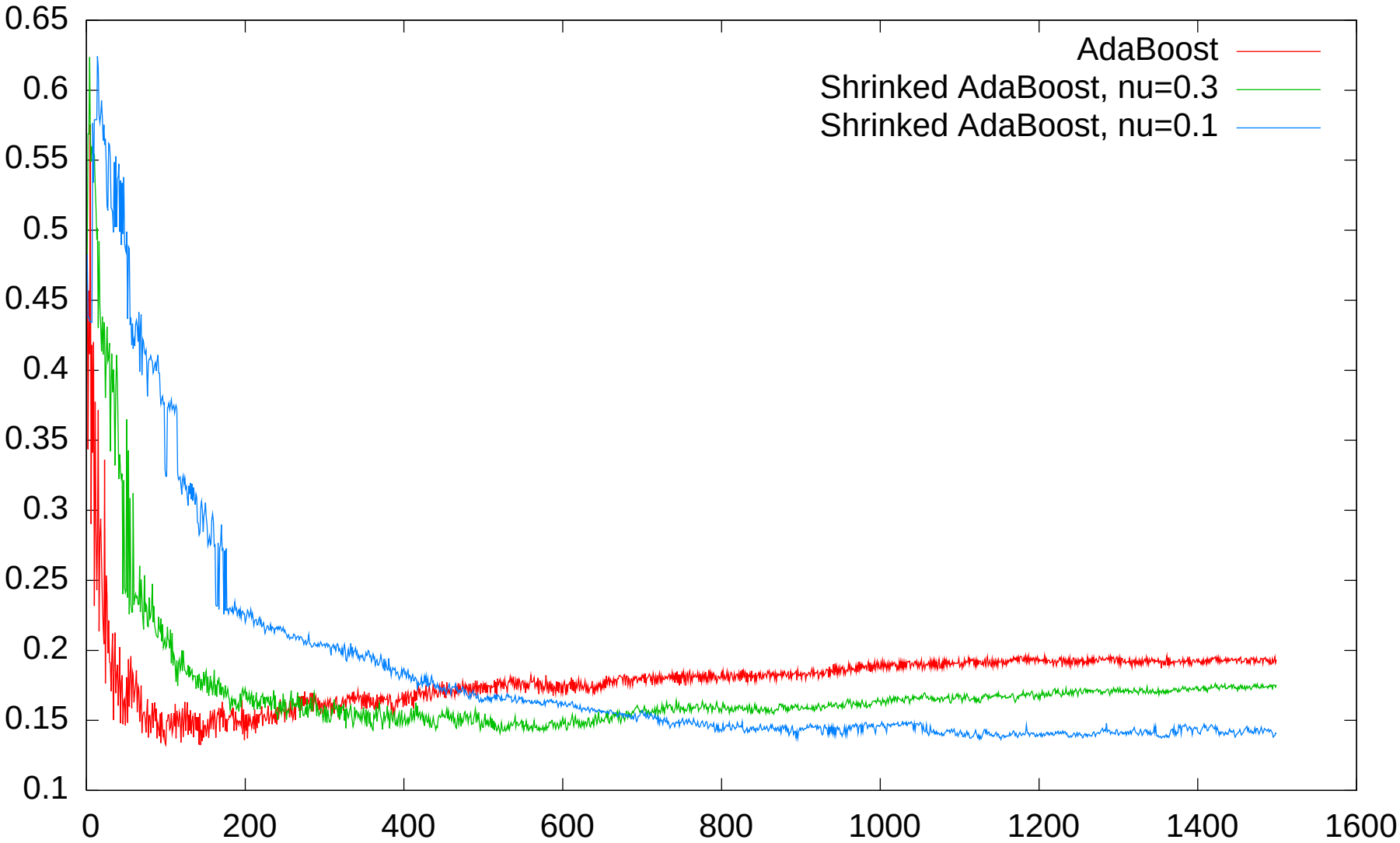


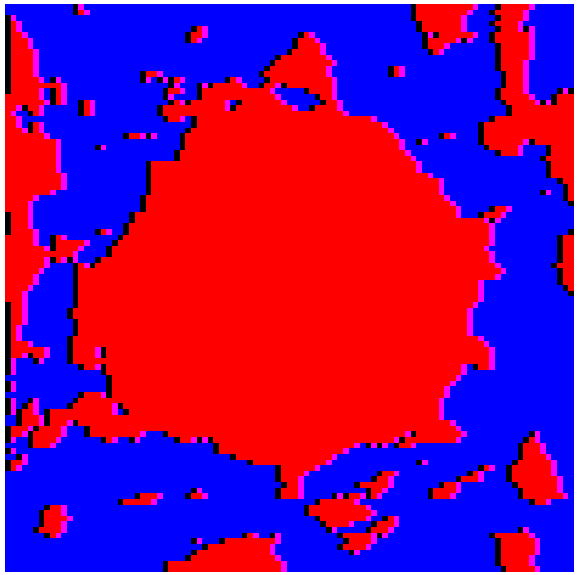
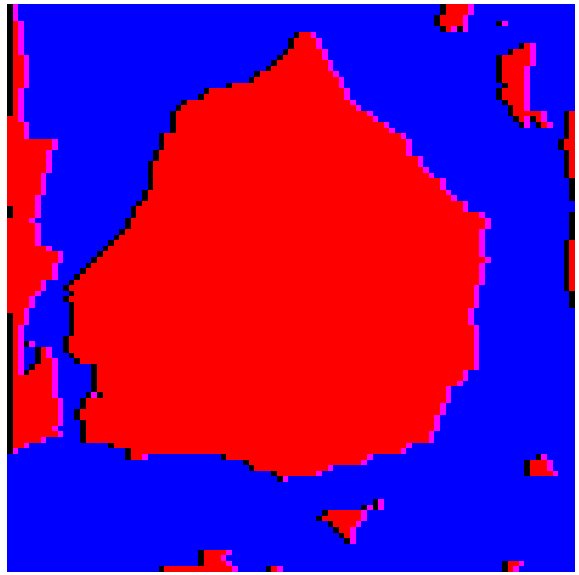


Exemples : Adaboost sur 500 exemples non bruités



Exemples : Adaboost sur 500 exemples, 30% de labels bruités



Les fonctions finales de classification après 500 itérations pour Adaboost (gauche) et AdaBoost+shrinkage ($\nu = 0.1$,gauche)

Formulation

Étant donnée une perte $\ell(f(\mathbf{x}), y) = \varphi(yf(\mathbf{x}))$, on note comme prédédemment

$$R_m(f) = \frac{1}{m}\sum_{i=1}^{m} \varphi(f(\mathbf{x}_i)y_i)\,;$$

On va étudier un algorithme de type minimisation directe sous contrainte

$$\widehat{f}_\lambda = \underset{f\in\lambda\mathrm{co}(\mathcal{H})}{\mathrm{Arg\,Min}}\, R_m(f)\,,$$

où

$$\mathrm{co}(\mathcal{H}) = \left\{ f = \sum_{i=1}^{k} \alpha_i h_i \,\middle|\, k\in\mathbb{N}\,, \alpha_i \leq 1, \sum_{i=1}^{k} |\alpha_i| = 1 \right\}\,,$$

et $\mathcal{H}$ est un ensemble de classifieurs de dimension de Vapnik-Chervonenkis finie (ou plus généralement, de fonctions bornées par 1 et ayant une complexité de Rademacher en $O(n^{-\xi})$) . On suppose que la minimisation ci-dessus est atteinte (sinon, on peut la supposer atteinte à ε_m près avec $\varepsilon_m \to 0$)

Équivalence avec la pénalisation

Une formulation très liée est le problème pénalisé de type LASSO

$$\widetilde{f}_C = \underset{\substack{f=\sum_{i=1}^k \alpha_i h_i \\ k\in\mathbb{N}\,,\alpha_i\leq 1}}{\mathrm{Arg\,Min}} \; R_m(f) + C\sum_i |\alpha_i| \;;$$

Notons qu'on a

$$\widetilde{f}_C = \underset{\lambda\geq 0}{\mathrm{Arg\,Min}} \left(R_m(\widehat{f}_\lambda) + C\lambda \right)$$

atteint pour un certain $\lambda(C)$; ainsi on a

$$\widetilde{f}_C = \widehat{f}_{\lambda(C)}\,.$$

Pourvu que φ soit strictement convexe on peut montrer que $\lambda(C)$ est bijective et ainsi, les solutions $\widetilde{f}_C$ et $\widehat{f}_\lambda$ décrivent le même ensemble lorsque C ou λ décrivent $\mathbb{R}$.

Résultat de consistance

On va prouver le résultat de consistance de [Lugosi et Vayatis, 2004] : notons d'abord le φ-risque en généralisation

$$R(f) = \mathbb{E}\left[\varphi(\mathbf{x}y)\right] .$$

Théorème

Supposons que $\mathcal{H}$ soit un ensemble de classifieurs de dimension VC finie et

$$\lim_{\lambda\to\infty} \inf_{f\in\lambda\mathrm{co}(\mathcal{H})} R(f) \to R^* = \inf_f R(f) .$$

Supposons que la fonction φ est convexe, positive, différentiable et décroissante. Alors, pour toute suite λ_m telle que

$$\lambda_m \to \infty \qquad \textit{et} \qquad -\lambda_m \varphi'(-\lambda_m)\sqrt{\frac{\log m}{m}} \to 0 ,$$

on a presque sûrement $\mathcal{E}(\widehat{f}_{\lambda_m}) \to \mathcal{E}(f^*)$, où $\mathcal{E}$ est l'erreur de généralisation en classification et f^* le classifieur de Bayes.

Preuve

On va prouver que la procédure est consistante pour le φ-risque :

$$R(\widehat{f}_{\lambda_m}) \to R^* \,,$$

ce qui impliquera aussi la consistance pour l'erreur de classification sous les hypothèses faites sur φ (voir dernière partie du présent cours).
La preuve réside principalement sur les outils de contrôle uniforme de l'erreur développés dans le cours de J. Shawe-Taylor. Pour λ fixé on va en fait contrôler avec grande probabilité uniformément la différence

$$\sup_{f \in \lambda \mathrm{co}(\mathcal{H})} |R_m(f) - R(f)|$$

Comme $|\varphi(f)| \leq \varphi(-\lambda)$ pour $f \in \lambda\mathrm{co}(\mathcal{H})$, on a par l'inégalité de McDiarmid, avec proba au moins $1-\delta$

$$\sup_{f \in \lambda \mathrm{co}(\mathcal{H})} |R_m(f) - R(f)| \leq \mathbb{E}\left[\sup_{f \in \lambda \mathrm{co}(\mathcal{H})} |R_m(f) - R(f)|\right] + \varphi(-\lambda)\sqrt{\frac{\log \delta^{-1}}{m}} \,.$$

Preuve (suite)

De plus, en utilisant le principe de symétrisation, on a

$$\mathbb{E}\left[\sup_{f\in\lambda\mathrm{co}(\mathcal{H})}|R_m(f)-R(f)|\right]\leq\mathbb{E}\left[\sup_{f\in\lambda\mathrm{co}(\mathcal{H})}\frac{1}{m}\sum_{i=1}^{m}\varepsilon_i\varphi(-y_i f(\mathbf{x}_i))\right],$$

où les (ε_i) sont des variables de Rademacher indépendantes. Comme φ est Lipschitz de constante $|\varphi'(-\lambda)|$ sur $[-\lambda,\lambda]$ (par convexité, dérivabilité et décroissance) , on a en appliquant le principe dit de contraction :

$$\begin{aligned}\mathbb{E}\left[\sup_{f\in\lambda\mathrm{co}(\mathcal{H})}\frac{1}{m}\sum_{i=1}^{m}\varepsilon_i\varphi(-y_i f(\mathbf{x}_i))\right]&\leq|\varphi'(-\lambda)|\,\mathbb{E}\left[\sup_{f\in\lambda\mathrm{co}(\mathcal{H})}\frac{1}{m}\sum_{i=1}^{m}\varepsilon_i y_i f(\mathbf{x}_i)\right]\\&=|\varphi'(-\lambda)|\,\mathbb{E}\left[\lambda\sup_{h\in\mathcal{H}}\left|\frac{1}{m}\sum_{i=1}^{m}\varepsilon_i h(\mathbf{x}_i)\right|\right]\\&\leq\lambda\,|\varphi'(-\lambda)|\sqrt{\frac{d\log m}{m}},\end{aligned}$$

la dernière inégalité venant de $\dim_{VC}(\mathcal{H})=d$.

Preuve (suite)

Bilan : on a avec probabilité $1-\delta$, pour tout $f \in \lambda_n \mathrm{co}(\mathcal{H})$:

$$|R_m(f) - R(f)| \leq \lambda_m \, |\varphi'(-\lambda_m)| \sqrt{\frac{d \log m}{m}} + \varphi(-\lambda_m) \sqrt{\frac{\log \delta^{-1}}{m}} \, ;$$

en prenant $\delta_m = m^{-2}$, et en utilisant $\varphi(-\lambda_m) \leq \varphi(0) - \lambda_m \varphi'(-\lambda_m)$, on a donc sauf avec probabilité m^{-2} :

$$|R_m(f) - R(f)| \leq \varepsilon_m = O\left(\lambda_m \, |\varphi'(-\lambda_m)| \sqrt{\frac{\log m}{m}} \right) = o(1) \, .$$

Soit f_λ le minimiseur de $R(f)$ sur $f \in \lambda \mathrm{co}(\mathcal{H})$. Donc :

$$\begin{aligned} R(\widehat{f}_{\lambda_m}) &\leq R_m(\widehat{f}_{\lambda_m}) + \varepsilon_m \\ &\leq R_m(f_{\lambda_m}) + \varepsilon_m \\ &\leq R(f_{\lambda_m}) + 2\varepsilon_m \, . \end{aligned}$$

Cette expression tend vers R^* lorsque $m \to \infty$, et l'inégalité est satisfaite ps pour n assez grand (lemme de Borel-Cantelli).

Conclusions

- Le paramètre λ (ou C dans la version pénalisée) joue bien le rôle de contrôle de la complexité qui doit "grandir lentement" avec m.
- La consistance universelle est obtenue même pour des classes de classifieurs de base relativement modestes : ainsi dans $[0,1]^d$, les classes suivantes $\mathcal{H}$ satisfont le critère d'approximation universel pour $\mathrm{co}(\mathcal{H})$:
  - indicatrices de parallélépipèdes (dim VC : $2d$)
  - arbres de décision à $(d+1)$ divisions (dim VC : $d \log_2(2d)$)
  - indicatrices de demi-espaces (dim VC : $d+1$)
  - indicatrices de boules (dim VC : $d+2$)
- L'étape suivante après la preuve de la consistance est l'établissement de vitesses de convergence, qui dépendent du compromis optimal entre approximation et complexité (voir [Blanchard, Lugosi et Vayatis, 2003]).

Algorithme

Pour résoudre en pratique la minimisation sous contrainte, on peut considérer à nouveau un algorithme itératif. Rappelons qu'on cherche pour un λ fixé

$$\widehat{f}_\lambda = \operatorname*{Arg\,Min}_{f \in \lambda \mathrm{co}(\mathcal{H})} R_m(f)$$

Algorithme (variation de [Zhang, 2003])

Initialisation

$f_0 = 0$

Pour $t = 1, \ldots, T$ **:**

$W_t(i) = \ell'(f_{t-1}(\mathbf{x}_i), y_i), \ i = 1, \ldots, n$ // "Cibles"

$h_t = \operatorname*{Arg\,Min}_{h \in \mathcal{H}} \sum_{i=1}^n W_t(i) h(i)$ // Direction de gradient minimum

$\alpha_t = \operatorname*{Arg\,Min}_{\alpha \in \mathbb{R}} R_m((1-\alpha) f_{t-1} + \alpha \lambda h_t)$ // Recherche linéaire

$f_t = f_{t-1} + \alpha_t h_t$

Fin

Retourner f_T

Convergence de l'optimisation

Théorème

Supposons $\ell(x,\cdot)$ *convexe,* $\sup_{h\in\mathcal{H}} \|h\|_\infty \leq B$ *et* $\sup_{x\in[-M,M]} \ell''(x,\cdot) \leq M$ *.*
Alors l'algorithme précédent satisfait pour toute itération t *:*

$$R_m(f_t) - R_m(\widehat{f}_\lambda) \leq \frac{8\lambda^2 B^2 M}{t}\,.$$

Preuve

Notons $F_{f,g}(\alpha) = R_m((1-\alpha)f + \alpha\lambda g)$. Par développement de Taylor, il existe un $\theta \in [0,1]$ tel que :

$$R_m((1-\alpha)f + \alpha\lambda g) = F_{f,g}(\alpha) = R_m(f) + \eta F'_{f,g}(0) + \frac{\alpha^2}{2} F''_{f,g}(\theta)$$

On peut vérifier que les hypothèses entraînent $F''_{f,h}(\theta) \in [0, C]$ avec $C = 4\lambda^2 B^2 M$, pour tout $h \in \mathcal{H}$.

Preuve (Suite)

On pose $\Delta_t = R_m(f_t) - R_m(\widehat{f}_\lambda)$. En écrivant le développement précédent pour $(f_t, \lambda^{-1}\widehat{f}_\lambda)$, $\alpha = 1$, on a

$$\begin{aligned}\Delta_t \leq -F'_{f_t, \lambda^{-1}\widehat{f}_\lambda}(0) &= -\lambda^{-1} \sum_{h \in \mathcal{H}} \widehat{\gamma}_\lambda(h) F'_{f_t,h} \\ &\leq - \inf_{h \in \mathcal{H}} F'_{f_t,h} \,.\end{aligned}$$

On écrit maintenant le développement pour (f_t, h_t), $\alpha = \alpha_t$:

$$R_m(f_{t+1}) \leq R_m(f_t) + \alpha_t F'_{f_t,h_t}(0) + \frac{C\alpha_t^2}{2} \,.$$

or F'_{f_t,h_t} est précisément choisi égal à $\inf_{h \in \mathcal{H}} F'_{f_t,h}$, donc en utilisant la première inégalité on a

$$\Delta_{t+1} \leq \Delta_t - \alpha_t \Delta_t + \frac{C\alpha_t^2}{2} \,.$$

Preuve (Fin)

Par récurrence, supposons $\Delta_t \leq \frac{2C}{t}$. Comme on choisit le α_t optimal par recherche linéaire dans $[0,1]$, on a alors en optimisant la borne précédente en α_t :

$$\begin{aligned}\Delta_{t+1} &\leq \Delta_t - \frac{\Delta_t^2}{2C} \\ &\leq \frac{2C}{t}\left(1-\frac{1}{t}\right) \\ &\leq \frac{2C}{t}\left(1-\frac{1}{t+1}\right) = \frac{2C}{t+1}\,.\end{aligned}$$

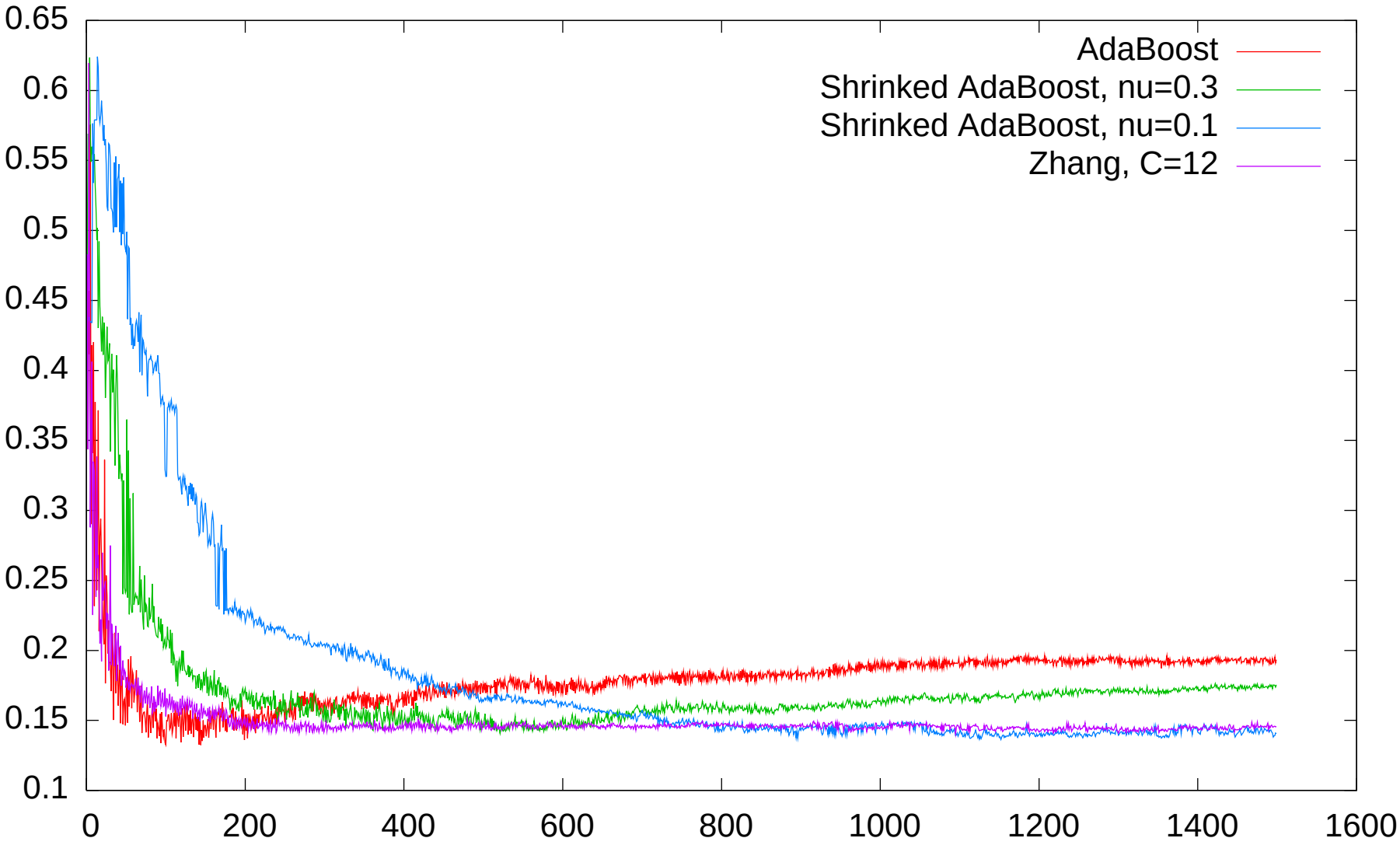
Conclusions

- on peut adapter le principe de la procédure gloutonne de type boosting pour résoudre le problème de minimisation avec contrainte sur la norme L_1 des coefficients.
- dans ce cas, il n'est pas nécessaire de limiter le nombre d'itérations (ou de faire du "shrinkage") puisque c'est la contrainte sur les coefficients qui doit régulariser contre le surapprentissage.
- il faut noter que la preuve de la consistance repose de façon fondamentale sur le fait qu'on minimise vraiment la perte $R_m(f)$ sous la contrainte sous les coefficients. Cette stratégie de preuve ne marche pas pour AdaBoost.

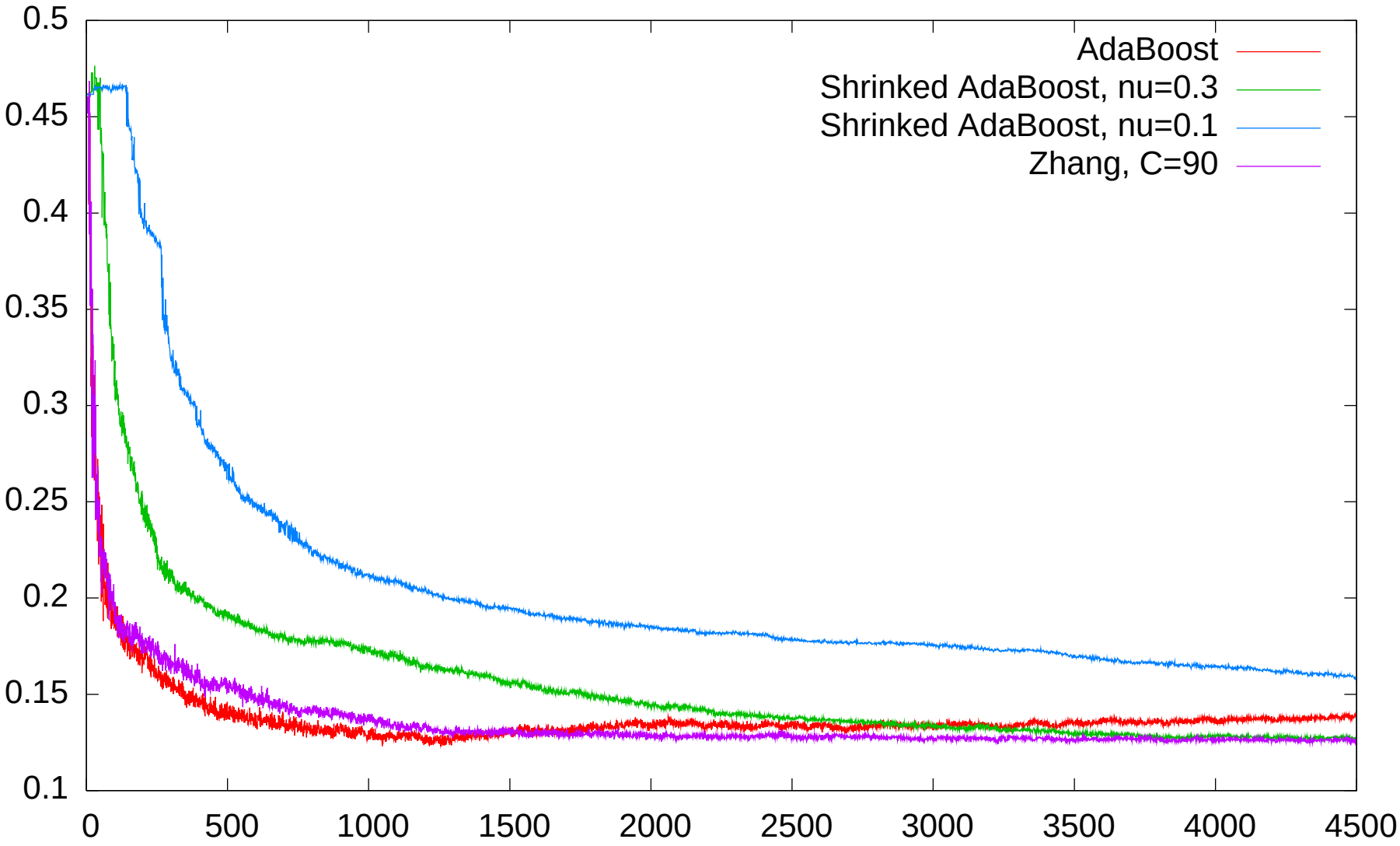
Considérations algorithmiques

- computationnellement plus coûteux puisqu'il faut choisir le paramètre λ, éventuellement par cross-validation.
- dans le cas particulier de la perte quadratique (en régression), ou de pertes linéaires par morceaux (type SVM), il existe un algorithme plus efficace pour calculer "le chemin entier de régularisation", c'est-à-dire savoir calculer $\widehat{f}_\lambda = \sum_{h\in\mathcal{H}} \widehat{\gamma}_\lambda(h)h$ pour tout $\lambda > 0$.
- dans ce cadre l'algorithme "least angle regression" [Efron et al. 2004] repose sur le fait que les coefficients $\widehat{\gamma}_\lambda(h)$ sont eux-mêmes linéaires par morceaux, avec un nombre $O(m)$ de morceaux, et au plus $m+1$ coefficients non nuls simultanément.

Exemple (cercle 2D)



Exemple (échiquier 2D)



Rappel

Algorithme (Boosting généralisé)

Initialisation

$f_0 = 0$

Pour $t = 1, \ldots, T$ **:**

$W_t(i) = \ell'(f_{t-1}(\mathbf{x}_i), y_i),\ i = 1, \ldots, n$ // "Cibles" ou "Poids"

$h_t = L(LS, \mathbf{W}_t)$ // Apprenant faible avec cibles $\mathbf{W_t}$

$\alpha_t = \underset{\alpha \in \mathbb{R}}{\text{Arg Min}}\, R_m(f_{t-1} + \alpha h_t)$ // Recherche linéaire

$f_t = f_{t-1} + \alpha_t h_t$

Fin

Retourner f_T

- En général, l'apprenant faible doit retourner un h_t minimisant l'erreur quadratique entre h_t et les cibles (càd réaliser une forme de régression aux moindres carrés sur les cibles).
- On va étudier le cas de la régression ($y \in \mathbb{R}$) avec $\ell(f(\mathbf{x}), y) = (f(x) - y)^2$.

L_2-Boosting [Buhlmann et Yu, 2003]

- Rappel : dans ce cas la fonction que l'on cherche à estimer est l'espérance conditionnelle $\mathbb{E}[Y|\mathbf{X}]$, puisque

$$\mathbb{E}[Y|\mathbf{X}] = \operatorname*{Arg\,Min}_f \mathbb{E}\left[(f(\mathbf{X}) - Y)^2\right]$$

- On a

$$\ell'(f(\mathbf{x}), y) = 2(f(x) - y) = 2(\text{Résidus de l'estimé de } y \text{ par } f),$$

- Une étape de boosting L^2 consiste donc à appliquer une méthode de régression pour estimer itérativement les *résidus*.
- Dans ce cadre particulier, cela est équivalent (comme pour AdaBoost !) à minimiser directement la fonction de perte elle-même...
- Procéder à deux itérations seulement était connu sous le nom de "twicing" en statistiques.

L_2-Boosting

On va se concentrer sur le cas "fixed design" ou le but est d'estimer la moyenne en chaque point $\mathbf{x}_i$ de l'échantillon. On note Y le vecteur des (y_i). Notons $L_X : \mathbb{R}^m \to \mathbb{R}^m$ l'algorithme de régression utilisé comme apprenant faible, prenant les valeurs (Y_i) en entrée et retournant les valeurs estimées de la moyenne en chaque point.

Proposition

Après m itérations de L_2-boosting, les valeurs estimées pour les moyennes en chaque point d'apprentissage sont données par

$$\widehat{F}_k = (I_m - (I_m - L)^{k+1})\, Y\,.$$

Preuve

Les résidus de l'étape $k-1$ sont donnés par le vecteur $\widehat{U}_{k-1} = Y - \widehat{F}_{k-1}$.
L'application de l'apprenant faible retourne les estimées pour ces résidus

$$\widehat{f}_k = L_X \widehat{U}_{k-1}$$

et donc

$$\widehat{U}_k = \widehat{U}_{k-1} - \widehat{f}_k = (I_m - L_X)\widehat{U}_{k-1} = (I_m - L_X)^k Y\,;$$

d'où

$$\widehat{F}_k = \sum_{i=1}^{k} \widehat{f}_i = \sum_{i=1}^{k} L_X (I_m - L_X)^i Y = (I_m - (I_m - L)^k) Y\,.$$

Cas d'un apprenant linéaire

- Pour simplifier l'étude on se concentre sur le cas où L_X est (le design $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ étant fixé) un opérateur linéaire symétrique sur les valeurs Y.
- C'est le cas de beaucoup de méthodes classiques ! Régression linéaire simple, ridge regression, estimation par splines, et plus généralement kernel ridge regression (== SVR).
- Soient $\lambda_1 \geq \ldots \geq \lambda_m$ les valeurs propres de L_X dans une base diagonalisante (e_i) :

$$L_X = UDU^t \,, D = \mathrm{diag}(\lambda_1, \ldots, \lambda_m) \,,$$

alors les estimées du L_2 Boosting après k itérations sont données par un opérateur diagonal dans la même base et dont les valeurs propres sont

$$\lambda_i^{(k)} = 1 - (1 - \lambda_i)^k \,.$$

Biais et variance

On considère le modèles classique de régression :

$$y_i = f(\mathbf{x}_i) + \sigma\varepsilon_i\,,$$

où les (ε_i) sont des variables aléatoires indépendantes de moyenne nulle et de variance 1.
Pour la perte quadratique, on a la décomposition classique de l'erreur d'une estimée $\widehat{f}$ de f :

$$\begin{aligned} EQM(\widehat{f}) &= \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}\left(\mathbb{E}\left[\widehat{f}(\mathbf{x}_i)\right] - f(\mathbf{x}_i)\right)^2 + \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{f}(\mathbf{x}_i) - \mathbb{E}\left[\widehat{f}(\mathbf{x}_i)\right]\right)^2 \\ &= \mathrm{Biais}(\widehat{f}) + \mathrm{Variance}(\widehat{f})\,, \end{aligned}$$

où l'espérance est par rapport aux (ε_i).

Biais et variance du L_2-boosting linéaire

Écrivons $U^t f = \mu$ l'écriture de la fonction-cible dans la base diagonalisante

Proposition

L'estimée $\widehat{f}_k$ après k itérations de L_2-boost satisfait :

$$\text{Biais}^2(\widehat{f}_k) = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbb{E}\left[\widehat{f}_k(\mathbf{x}_i)\right] - f(\mathbf{x}_i)\right)^2 = \frac{1}{m}\sum_{i=1}^{m}(1-\lambda_i)^{2k}\mu_i^2$$

$$\text{Variance}^2(\widehat{f}_k) = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{f}_k(\mathbf{x}_i) - \mathbb{E}\left[\widehat{f}_k(\mathbf{x}_i)\right]\right)^2 = \sigma^2\frac{1}{m}\sum_{i=1}^{m}(1-(1-\lambda_i)^k)^2$$

Preuve

Dans la base diagonalisante (e_i), en désignant $\widehat{\mathbf{f}}$ le vecteur des estimées, $\mathbf{f}$ celui des vraies moyennes, et B_k l'opérateur correspondant à k étapes de L_2-boosting, on a

$$\left\langle e_i, \mathbb{E}\widehat{\mathbf{f}}\right\rangle = \langle e_i, B_k\mathbf{f} + B_k\mathbb{E}\varepsilon\rangle = (1-(1-\lambda_i)^k)\mu_i\,.$$

Quelques conséquences élémentaires

Proposition

(i) Si pour tout k tel que $\lambda_k > 1$, on a $\mu_k^2/\sigma^2 > (1-\lambda_k)^{-1}$, alors le L_2-boosting améliore (au sens de l'EQM) l'apprenant faible utilisé seul.
(ii) Si l'apprenant faible n'est pas l'identité, et $\sigma > 0$, alors il existe un k tel que k itérations de L_2-Boosting font mieux (pour l'EQM) que le surapprentissage total (correspondant à l'opérateur identité).

- Ces deux propriétés sont faibles mais nous assurent qu'arrêter le L_2 boosting après un certain nombre d'itérations a un intéret par rapport à ne rien itérer du tout (point (i)) ou itérer jusqu'à l'infini (point (ii)).
- Noter que bien que le (ii) semble trivial, il n'est pas évident qu'un algorithme d'apprentissage linéaire fixé fasse mieux que le surapprentissage total en général (en particulier si le rapport signal/bruit est très grand – problème peu bruité).

Preuve

Rappelons la formule

$$EQM(k) = g(k) = \frac{1}{m}\sum_{i=1}^{m} \mu_i^2 (1-\lambda_i)^{2k} + \sigma^2 \frac{1}{m}\sum_{k=1}^{m} (1-(1-\lambda_i)^k)^2 \,;$$

On calcule

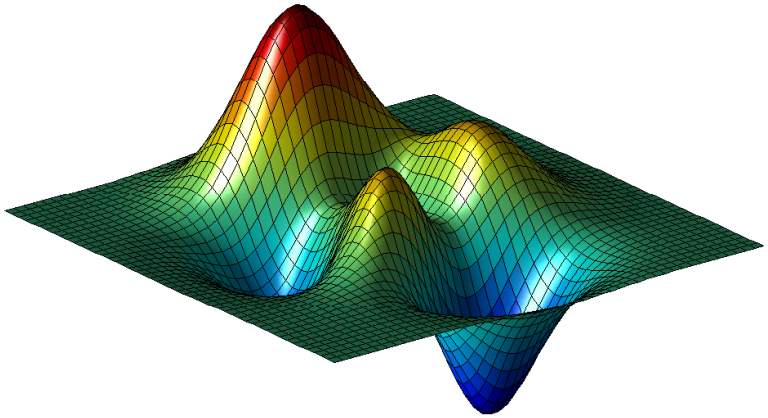
$$g'(x) = \frac{2}{m}\sum_{i=1}^{n} ((\mu_i^2(1-\lambda_i)^x - \sigma^2))(1-\lambda_i)^x \log(1-\lambda_i)\,,$$

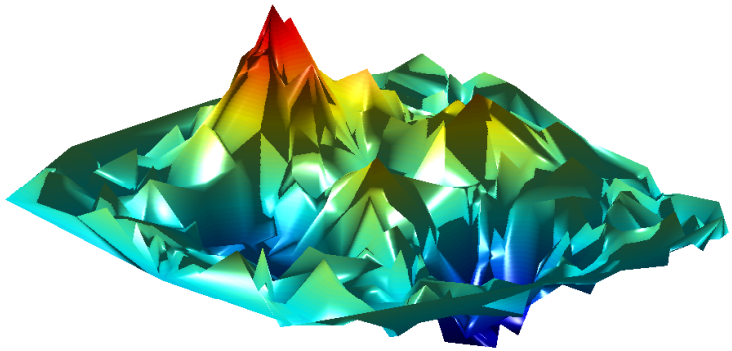
donc sous l'hypothèse du (i) $g'(x) \leq 0$ pour $x \in [1,2]$ et l'EQM décroît donc entre la première et la deuxième itération.
Par ailleurs en développant le carré du deuxième terme de $g(x)$ on a

$$g(x) = \sigma^2 + \frac{1}{m}\sum_{i=1}^{m} ((\mu_i + \sigma^2)(1-\lambda_i)^x - 2\sigma^2)(1-\lambda_i)^x\,,$$

et le deuxième terme est négatif pour x assez grand.

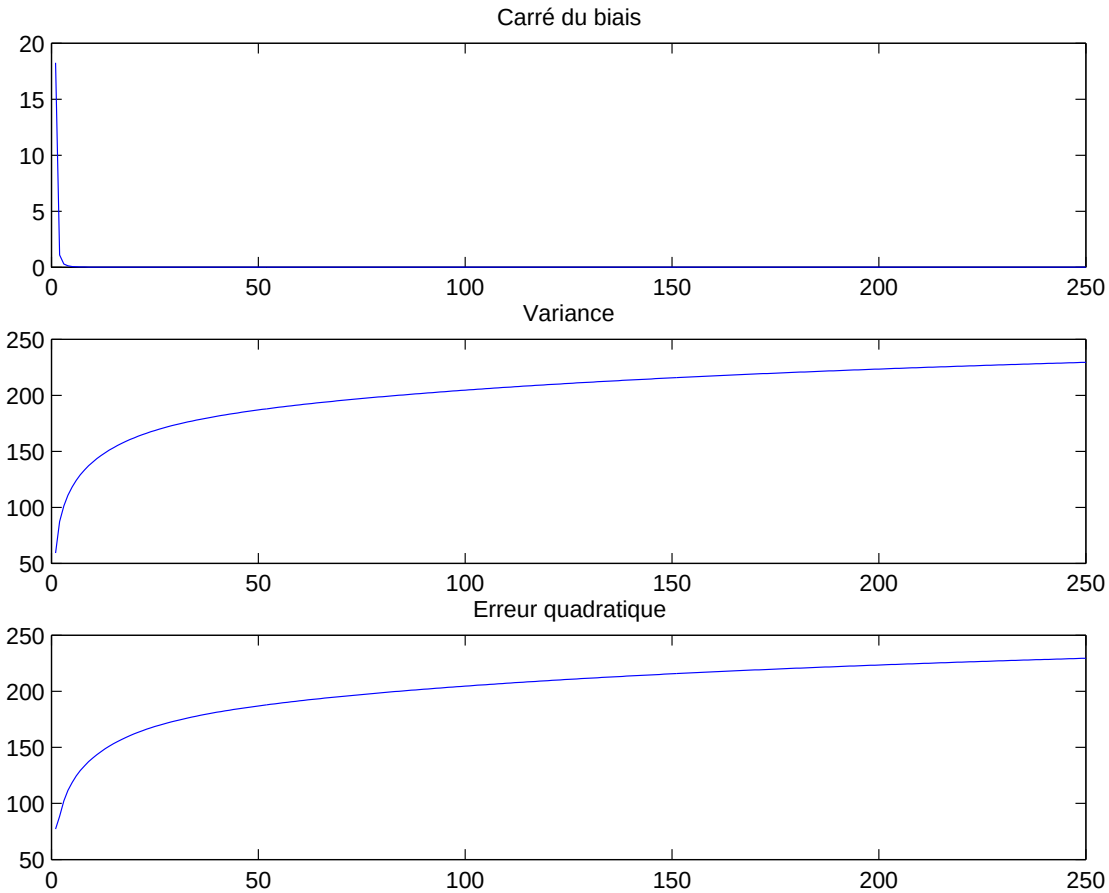
Exemple

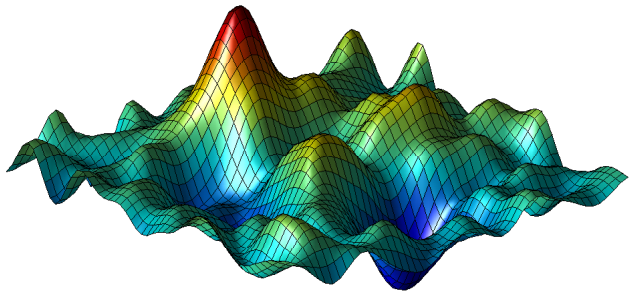




500 points d'apprentissage + bruit Gaussien
Apprenant faible : kernel ridge regression à noyau Gaussien

Exemple : apprenant trop fort

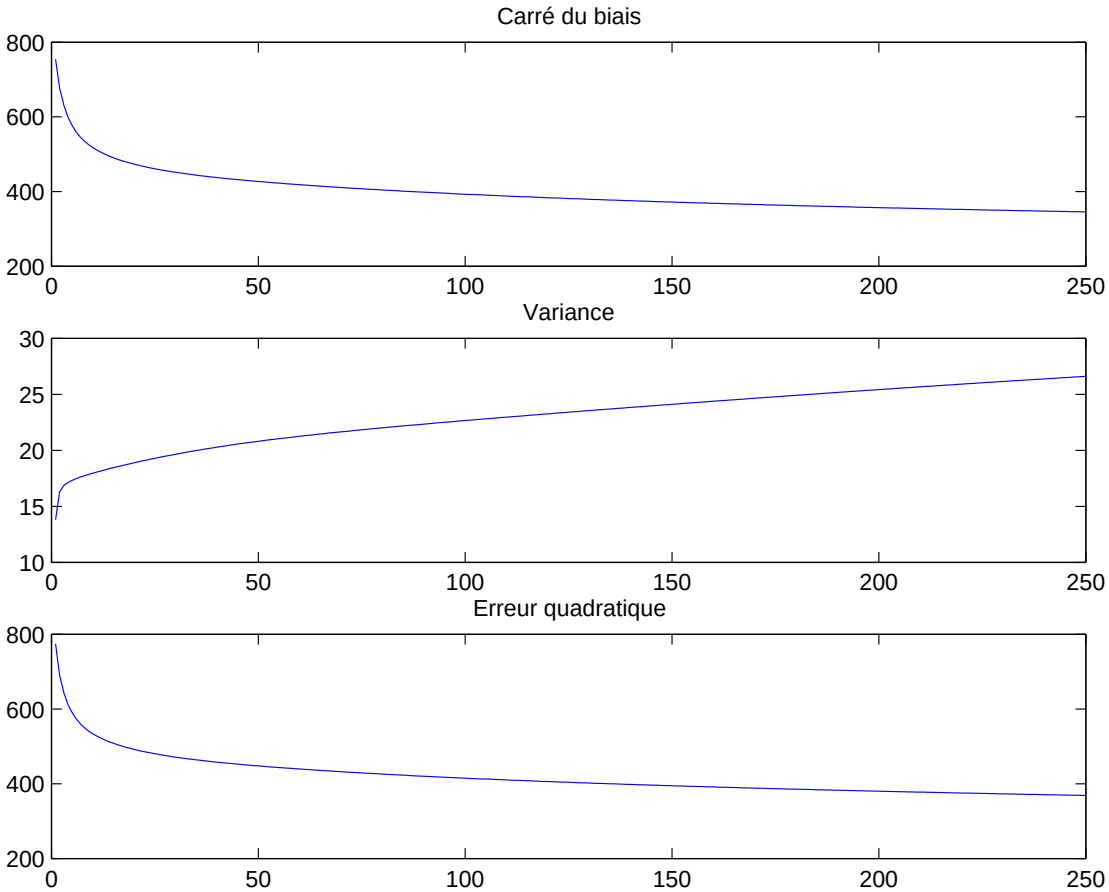


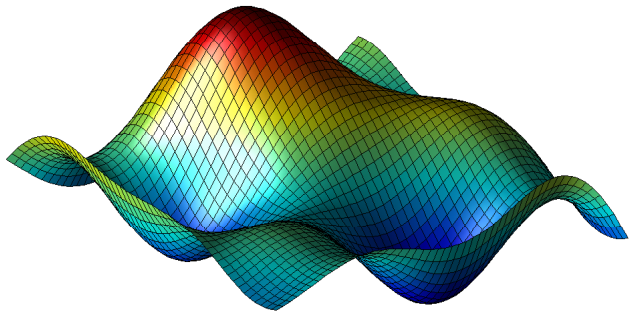


$(\sigma = 0.1, C = 1, \nu = 1)$

Fit à 50 itérations

Exemple : apprenant trop faible

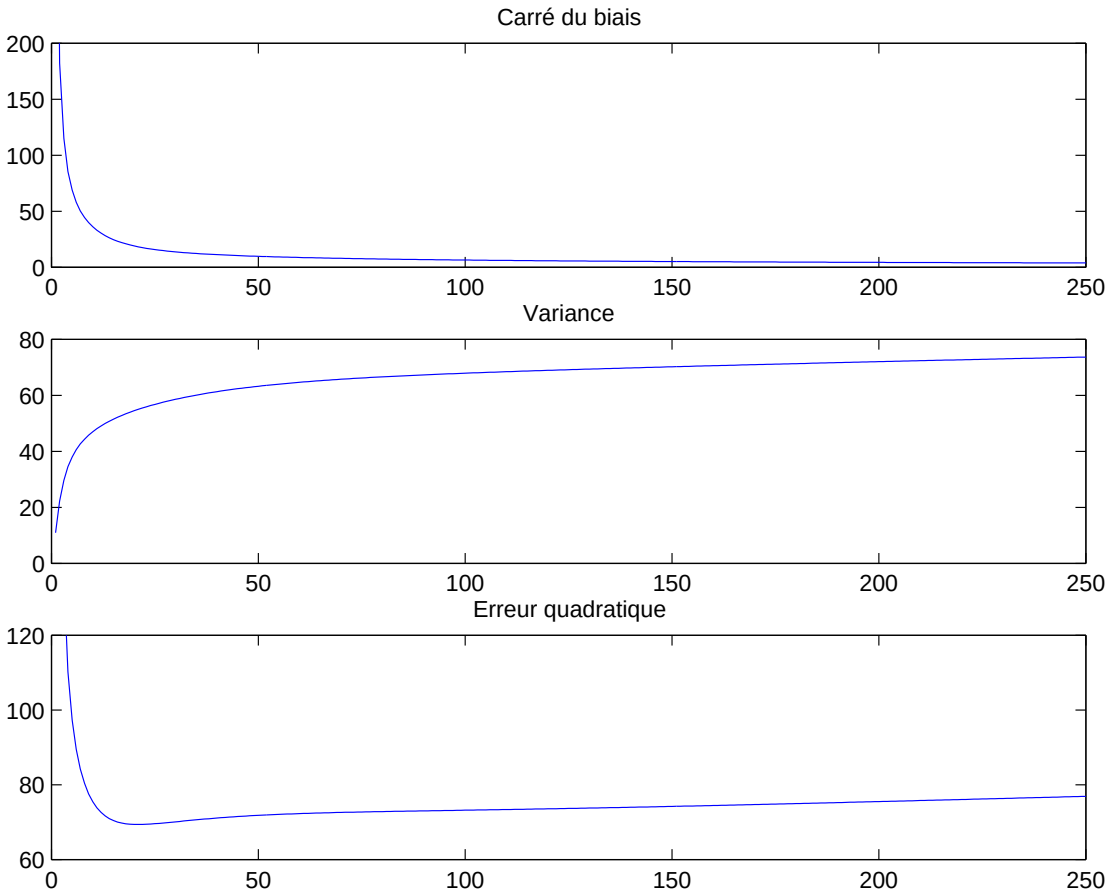


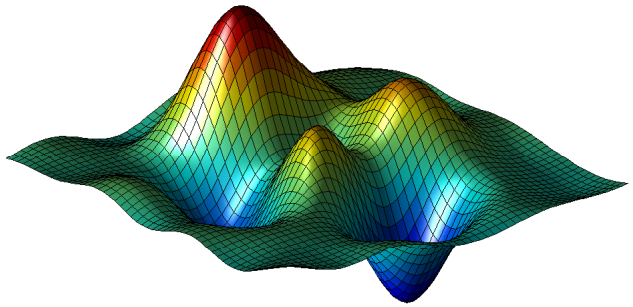


$(\sigma = 0.5, C = 1, \nu = 1)$

Fit à 50 itérations

Exemple : cas idéal





$(\sigma = 0.2, C = 5, \nu = 0.8)$

Fit à 50 itérations

Conclusions sur le L_2 boosting linéaire

- Lorsque le nombre d'itérations k croît, le biais de la procédure décroît vers 0 et sa variance croît vers σ^2 .
- La limite $k \to \infty$ correspond à la situation de surapprentissage total $\widehat{\mathbf{f}}_\infty = \mathbf{Y}$.
- Le nombre d'itérations fait donc bien office de régularisation.
- On peut choisir un nombre d'itérations soit par cross-validation, soit avec un critére de pénalisation de type AIC ou BIC en utilisant la notion de "degrés de liberté" de l'estimateur $(= \mathrm{tr}(B_k))$.
- Il est intéressant d'avoir un apprenant vraiment faible (càd avec des valeurs propres petites) : cela a de plus fortes chances de garantir une amélioration dans les premières itérations et permet de choisir le point de compromis optimal de maniére plus fine.
- L'utilisation de "shrinkage" à coefficient ν revient effectivement à "affaiblir" l'apprenant (en multipliant toutes les valeurs propres par $\nu < 1$) et peut être recommandable justement pour cette raison.

Le L_2 boosting avec apprenant non linéaire

- Buhlmann [Buhlmann, 2006] étudie aussi le L_2 boosting avec un apprenant non-linéaire :
- Soit $\mathcal{H}$ une famille de fonctions fixée , l'apprenant linéaire choisit étant données les entrées $((\mathbf{x}_i), y_i)$:

$$(\widehat{h}, \widehat{\alpha}) = \underset{\substack{h\in\mathcal{H}\\ \alpha\in\mathbb{R}}}{\text{Arg Min}} \frac{1}{m}\sum_{i=1}^{m}(y_i - \alpha h(\mathbf{x}_i))\,.$$

Noter que pour h fixé, le α optimal se calcule très facilement comme

$$\widehat{\alpha}(h, \mathbf{X}, bY) = \frac{\sum_{i=1}^m y_i h(\mathbf{x}_i)}{\sum_{i=1}^m h(\mathbf{x}_i)^2}\,.$$

- Cet algorithme se rapproche alors fortement du "matching pursuit".
- Buhlmann montre la consistance de cet algorithme sous les hypothèse suivantes :
  - $\mathbf{x}$ et y sont bornés p.s.
  - $\text{Card}(\mathcal{H}) = O(\exp(Cm^{1-\xi}))$ pour un certain $C > 0$ et $\xi > 0$;
  - $\mathbb{E}[y|\mathbf{x}] = \sum_{h\in\mathcal{H}} \beta_h \widehat{h}(\mathbf{x})$ où les $\widehat{h}$ sont les h normalisés pour la norme $L^2(P_X)$, et $\sum_h |\beta_h| < M$.

Remarques sur la consistance des algorithmes de boosting itératif

- La preuve de Buhlmann contient grosso modo trois ingrédients :
  - (i) étude d'un algorithme "idéalisé" (si on pouvait calculer les vraies espérances à chaque itération),
  - (ii) bornes uniformes sur les espérances concernées (ici, des covariances entre $(h(\mathbf{x}), h'(\mathbf{x}))$ ou $(h(\mathbf{x}), y)$ – tout est borné donc on peut appliquer par exemple l'inégalité de Hoeffding)
  - (iii) l'algorithme basé sur les données reste alors proche de l'algorithme idéalisé si on s'arrête assez tôt.
- Les même principes généraux peuvent être suivis pour montrer la consistance d'algorithmes de type Boosting itératif généralisés [Bickel, Ritov, Zakai 2005] ;
- La consistance d'Adaboost a été montrée récemment [Bartlett et Traskin, 2007] .

Buts

- On se place dans le cadre de la classification binaire. On a vu que AdaBoost revient à optimiser la fonction $R_m(f) = \frac{1}{m}\sum_{i=1}^m \exp(-f(\mathbf{x}_i)y_i)$.
- On peut utiliser à la place de $\ell(f(\mathbf{x}), y) = \exp(-f(\mathbf{x})y)$ d'autres fonctions de pertes de la forme $\ell(f(\mathbf{x}), y) = \varphi(f(\mathbf{x}), y)$.
- Quelles sont les propriétés importantes pour φ pour garantir que minimiser $R_{\varphi,m}(f) = \frac{1}{m}\sum_{i=1}^m \varphi(f(\mathbf{x}_i), y_i)$ va retourner un classifieur adéquat ?
- Par example, on a vu que $R_m(f)$ est un majorant de l'erreur de classification empirique. Est-ce une condition suffisante ? Nécessaire ?

Buts

On peut décomposer la question en deux-sous problèmes :

- Est-ce que $R_{m,\varphi}(f)$ est assez proche de $R_\varphi(f) = \mathbb{E}\left[\varphi(f(\mathbf{x})y)\right]$: question de théorie de l'apprentissage statistique (cf. cours de John Shawe-Taylor !)
- Si le point ci-dessus est garanti, on peut en déduire que que $R_\varphi(\widehat{f}_m) \to \inf_{f\in\mathcal{H}} R_\varphi(f)$ quand $m \to \infty$, cela implique-t-il alors que $\mathcal{E}(\widehat{f}_m) \to \mathcal{E}(f^*)$, ($\mathcal{E}$ est l'erreur de généralisation, f^* le classifieur de Bayes)
- On va se consacrer à cette question, en essayant en particulier d'obtenir des inégalités du type

$$\mathcal{E}(f) - \mathcal{E}(f^*) \leq F(R_\varphi(f) - R^*)\,,$$

où $R^* = \inf_f R_\varphi(f)$ est le φ-risque "optimal". (Travaux de [Bartlett, Jordan, McAuliffe, 2006])

- Ainsi, si la théorie de l'apprentissage peut garantir que $R_\varphi(\widehat{f}_m) \to R^*$, on pourra en déduire que $\widehat{f}_m$ converge vers Bayes – en ayant éventuellement une vitesse de convergence.

Préliminaires

Toutes les pertes étant des moyennes par rapport à la variable $\mathbf{x}$, il est intéressant de regarder ce qui se passe en un point $\mathbf{x}$ fixé, en regardant la φ-perte conditionnelle

$$\mathbb{E}\left[\varphi(yf(\mathbf{x}))|\mathbf{x}\right] = \eta(\mathbf{x})\varphi(f(\mathbf{x})) + (1-\eta)\varphi(-f(\mathbf{x}))\,,$$

où $\eta(\mathbf{x}) = P(Y=1|\mathbf{x})$. On définit pour simplifier l'écriture la fonction

$$C_\eta(\alpha) = \eta\varphi(\alpha) + (1-\eta)\varphi(-\alpha)\,.$$

Pour un $\mathbf{x}$ fixé et avec $\eta = \eta(\mathbf{x})$, la perte conditionnelle est donc minimisée pour le choix de $f(\mathbf{x}) = \alpha^*$ réalisant

$$H(\eta) = \inf_{\alpha\in\mathbb{R}} C_\eta(\alpha) = \inf_{\alpha\in\mathbb{R}} \left(\eta\varphi(\alpha) + (1-\eta)\varphi(-\alpha)\right).$$

Ainsi, la φ-perte optimale (en moyenne sur $\mathbf{x}$) est

$$R^* = \inf_f \mathbb{E}\left[R(f)\right] = \mathbb{E}\left[H(\eta(\mathbf{x}))\right]\,.$$

Calibration pour la classification

- Le choix optimal $f(\mathbf{x}) = \alpha^*$ minimisant la φ-perte ($\mathbf{x}$ fixé) correspondra à une fonction de classification optimale à condition que α^* soit du même signe que $2\eta - 1$.
- Posons

$$H^-(\eta) = \inf_{\alpha : \mathrm{sign}(\alpha(2\eta-1)) \leq 0} C_\eta(\alpha)\,,$$

  le meilleur φ-risque qu'on peut atteindre tout en faisant une erreur de classification.
- On est naturellement amené à la définition suivante :

Definition

La fonction de perte φ est dite calibrée pour la classification à condition que pour tout $\eta \neq \frac{1}{2}$:

$$H^-(\eta) > H(\eta)\,.$$

La fonction ψ

- Définissons pour $\eta \in [0,1]$

$$D(\eta) = H^-(\eta) - H(\eta) \ :$$

D mesure l'excès minimum de φ-perte lorsqu'on fait une erreur de classification.

- Puis pour $\theta \in [-1,1]$,

$$\widetilde{\psi}(\theta) = D\left(\frac{1+\theta}{2}\right) \ ,$$

- Et enfin définissons pour $\theta \in [-1,1]$ la fonction ψ comme le plus grand minorant convexe de ψ.

Premier résultat

Théorème

Pour toute fonction mesurable $f : \mathcal{X} \to \mathbb{R}$ et toute distributioin de probabilité P sur $\mathcal{X} \times \{-1, 1\}$, on a

$$\psi(\mathcal{E}(f) - \mathcal{E}(f^*)) \leq R_\varphi(f) - R^* \, .$$

Preuve

On constate que

$$\mathcal{E}(f) - \mathcal{E}(f^*) = \mathbb{E}\left[\mathbf{1}\left\{\mathrm{sign}(f(\mathbf{x})) \neq \mathrm{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right] \, .$$

On a alors par Jensen, φ étant convexe avec $\psi(0) = 0$:

$$\begin{aligned}\psi(\mathcal{E}(f) - \mathcal{E}(f^*)) &\leq \mathbb{E}\left[\psi\left(\mathbf{1}\left\{\mathrm{sign}(f(\mathbf{x})) \neq \mathrm{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right)\right] \\ &= \mathbb{E}\left[\mathbf{1}\left\{\mathrm{sign}(f(\mathbf{x})) \neq \mathrm{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\psi\left(\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right)\right] \, .\end{aligned}$$

Preuve (suite)

$$
\begin{aligned}
&\psi(\mathcal{E}(f) - \mathcal{E}(f^*)) \\
&\leq \mathbb{E}\left[\mathbf{1}\left\{\operatorname{sign}(f(\mathbf{x})) \neq \operatorname{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\psi\left(\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right)\right] \\
&= \mathbb{E}\left[\mathbf{1}\left\{\operatorname{sign}(f(\mathbf{x})) \neq \operatorname{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}D(\eta)\right] \\
&= \mathbb{E}\left[\mathbf{1}\left\{\operatorname{sign}(f(\mathbf{x})) \neq \operatorname{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\left(\inf_{\alpha:\alpha(2\eta(\mathbf{x})-1)\leq 0} C_{\eta(\mathbf{x})}(\alpha) - H(\eta(\mathbf{x}))\right)\right] \\
&\leq \mathbb{E}\left[C_{\eta(\mathbf{x})}(f(\mathbf{x})) - H(\eta(\mathbf{x}))\right] \\
&= R_\varphi(f) - R_\varphi^* \,.
\end{aligned}
$$

Optimalité

Le résultat précédent est en un certain sens optimal :

Théorème

Pour toute fonction de perte φ, pour tout $\theta \in [0,1]$ et $\varepsilon > 0$ il existe une distribution P sur $\mathcal{X} \times \{-1,1\}$ et une fonction f telle que $\mathcal{E}(f) - \mathcal{E}(f^*) = \theta$, et la borne du résultat précédent est précise à ε près .

Preuve (Résumée)

- Comme ψ est l'enveloppe convexe inférieure de $\widetilde{\psi}(\theta) = D((1+\theta)/2)$, il existe α_1, α_2 et γ tels que

$$\theta = \gamma\alpha_1 + (1-\gamma)\alpha_2\,,$$
$$\psi(\theta) \geq \gamma\widetilde{\psi}(\alpha_1) + (1-\gamma)\widetilde{\psi}(\alpha_2) - \varepsilon/2\,.$$

- Prendre deux points distincts x_1, x_2 et choisir :

$$P(x_1) = \gamma\,, \qquad P(x_2) = 1-\gamma\,;$$
$$\eta(\mathbf{x}_1) = \eta_1 = (1+\alpha_1)/2\,. \qquad \eta(\mathbf{x}_2) = \eta_2 = (1+\alpha_2)/2\,;$$

et $f(x_1), f(x_2)$ faisant une erreur de classification tout en minimisant le φ-risque à $\varepsilon/2$ près sous cette contrainte, càd

$$C_{\eta_1}(f(x_1)) \leq H^-(\eta_1) + \frac{\varepsilon}{2}\,; \qquad C_{\eta_2} f(x_2) \leq H^-(\eta_2) + \frac{\varepsilon}{2}\,.$$

Relation à la calibration en classification

Théorème

On a équivalence entre :
(a) φ *est calibrée pour la classification.*
(b) Pour toute suite (θ_i) , $\psi(\theta_i) \to 0 \Leftrightarrow \theta_i \to 0$
(c) Pour toute suite de fonctions f_i ,

$$R_\varphi(f_i) \to R_\varphi^* \Rightarrow \mathcal{E}(f_i) \to \mathcal{E}(f^*)\,.$$

Preuve

Conséquence du premier résultat et du fait que φ est calibrée pour la classification si et seulement si $D(\eta) > 0$ pour tout $\eta \neq \frac{1}{2}$ et donc ssi $\psi(\theta) > 0$ pour tout $\theta > 0$.

Cas d'une fonction de perte convexe

Théorème

Si la fonction φ est convexe, elle est calibrée pour la classification si et seulement si elle est différentiable en 0 avec $\varphi'(0) < 0$. Dans ce cas, on a

$$\psi(\theta) = \varphi(0) - H\left(\frac{1+\theta}{2}\right) .$$

Preuve

Laissée en exercice. . . Indices :

- Noter que la perte conditionnelle $C_\eta(\alpha)$ est alors convexe ;
- Par l'absurde, supposer que φ a une dérivée à droite et à gauche différentes en zéro ; pour un choix adapté de η , $C_\eta(\alpha)$ a une dérivée positive en zéro.
- Réciproquement si $\varphi'(0) \leq 0$, alors $C_\eta(\alpha)$ a une dérivée négative en zéro.

Rôle de la “condition de bruit en classification”

Une analyse plus fine montre que la relation entre φ-perte en excès et perte en classification est intimement liée à une certaine forme de conditionnement du problème :

Definition (Condition de bruit en classification [Tsybakov, 2004])

Un problème de classification satisfait la condition de bruit de Tsybakov avec exposant β s'il existe une constante $c > 0$, telle que pour tout $\varepsilon > 0$:

$$\mathbb{P}\left[0 < \left|\eta(\mathbf{x}) - \frac{1}{2}\right| \leq \varepsilon\right] \leq c\varepsilon^{\beta} .$$

- Toujours satisfait pour $\beta = 0$.
- Pour $\beta > 0$: meilleure séparabilité (la zone “ambiguë” avec η proche de $\frac{1}{2}$ est de proba faible)
- Cas “standard” si $\eta(\mathbf{x})$ est $\mathcal{C}^1$ avec $\nabla\eta(\mathbf{x}) \neq 0$ et la marginale de X est comparable à Lebesgue sur un compact de $\mathbb{R}^d$: satsisfait pour $\beta = 1$.

Résultat sous condition de bruit

Théorème

Si φ est calibrée en classification et la condition de bruit à exposant β est satisfaite, on a pour une certaine constante C , pour toute fonction mesurable f :

$$C(\mathcal{E}(f)-\mathcal{E}(f^*))^{\alpha}\psi\left(\frac{(\mathcal{E}(f)-\mathcal{E}(f^*))^{1-\alpha}}{2C}\right)\leq R_{\varphi}(f)-R_{\varphi}^*\,,$$

où $\alpha=\frac{\beta}{\beta+1}$.

Ceci est toujours mieux (à la constante près) que le résultat précédent, car on a

$$(\mathcal{E}(f)-\mathcal{E}(f^*))^{\alpha}\psi\left(\frac{(\mathcal{E}(f)-\mathcal{E}(f^*))^{1-\alpha}}{2C}\right)\geq\psi\left(\frac{(\mathcal{E}(f)-\mathcal{E}(f^*))}{2C}\right)\,.$$

Preuve

$$
\begin{aligned}
&\mathcal{E}(f) - \mathcal{E}(f^*) \\
&= \mathbb{E}\left[\mathbf{1}\left\{\operatorname{sign}(f(\mathbf{x})) \neq \operatorname{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right] \\
&= \mathbb{E}\left[\mathbf{1}\left\{\left|\eta(\mathbf{x}) - \frac{1}{2}\right| < \varepsilon\right\}\mathbf{1}\left\{\operatorname{sign}(f(\mathbf{x})) \neq \operatorname{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right] \\
&\quad + \mathbb{E}\left[\mathbf{1}\left\{\left|\eta(\mathbf{x}) - \frac{1}{2}\right| \geq \varepsilon\right\}\mathbf{1}\left\{\operatorname{sign}(f(\mathbf{x})) \neq \operatorname{sign}(\eta(\mathbf{x}) - \frac{1}{2})\right\}\left|2\eta(\mathbf{x}) - \frac{1}{2}\right|\right] \\
&= (I) + (II)\,.
\end{aligned}
$$

Le terme (I) est borné par $c\varepsilon^{\beta+1}$ par hypothèse. Pour le terme (II) : pourvu que $\varepsilon \leq \left|\eta(\mathbf{x}) - \frac{1}{2}\right|$, on a

$$
|2\eta(\mathbf{x}) - 1| \leq \frac{\varepsilon}{\psi(\varepsilon)}\psi\left(\left|\eta(\mathbf{x}) - \frac{1}{2}\right|\right)\,,
$$

car ψ convexe et $\psi(0) = 0$ implique $\frac{\psi(x)}{x}$ croissante (on utilise aussi $\psi(\varepsilon) > 0$ par la calibration)

Preuve

On suit ensuite le même raisonnement que pour le premier résultat, pour obtenir

$$(II) \leq \frac{\varepsilon}{\psi(\varepsilon)}(R_\varphi(f) - R_\varphi^*)\,.$$

En regroupant, on obtient

$$\left(\frac{\mathcal{E}(f) - \mathcal{E}(f^*)}{\varepsilon} - c\varepsilon^\beta\right)\psi(\varepsilon) \leq (R_\varphi(f) - R_\varphi^*)\,.$$

On prend pour conclure

$$\varepsilon = \frac{1}{2c}(\mathcal{E}(f) - \mathcal{E}(f^*))^{\frac{1}{\beta+1}}\,.$$

Exemples

Nom	$\varphi(x)$	$\alpha^*(\eta)$	$\psi(\theta)$
Exponentiel	$\exp(-x)$	$\frac{1}{2}\log\frac{\eta}{1-\eta}$	$1-\sqrt{1-\theta^2}$
Logit	$\log_2(1+\exp(-x))$	$\frac{1}{2}\log\frac{\eta}{1-\eta}$	$\ldots$
Quadratique tronqué	$(1-x)_+^2$	$2\eta-1$	θ^2


R. E. Schapire et Y. Singer.
Improved boosting algorithms using confidence-rated predictions.
Machine Learning, 37(3) :297-336, 1999.


G. Lugosi et N. Vayatis.
On the Bayes-risk consistency of regularized boosting methods.
Annals of Statistics, vol.32, pp.30–55.


G. Blanchard, G. Lugosi, N. Vayatis.
On the Rate of Convergence of Regularized Boosting Classifiers.
Journal of Machine Learning Research, 4 :861-894, 2003.


J.H. Friedman, T. Hastie, R. Tibshirani
Additive Logistic Regression : a Statistical View of Boosting.
Annals of Statistics, 28 : 337-374, 2000


P. Buhlmann et B. Yu.
Boosting with the L_2 loss : regression and classification.
Journal of the American Statistical Association 98, 324-339, 2003.


P. Buhlmann.
Boosting for high-dimensional linear models.
Annals of Statistics 34, 559-583, 2006.


P.L. Bartlett, M.I. Jordan, J.D. McAuliffe.
Convexity, classification, and risk bounds.
Journal of the American Statistical Association, 101(473) :138-156, 2006.


R. Meir, G Rätsch.
An introduction to boosting and leveraging.
In Advanced Lectures on Machine Learning (LNAI2600), Springer, 2003.


A. Tsybakov.
Optimal aggregation of classifiers in statistical learning.
Annals of Statistics, v.32, n.1, 135-166, 2004.


P. J. Bickel, Y. Ritov, A. Zakai
Some theory for generalized boosting algorithms
Journal of Machine Learning Research, 7, 705–732, 2005.


R. Tibshirani
Regression shrinkage and selection via the lasso.
J. Royal. Statist. Soc B., Vol. 58, No. 1, pages 267-288, 1996.


G. Blanchard.
Different paradigms for choosing sequential reweighting algorithms.
Neural Computation, 16 :811-836, 2004.


P. L. Bartlett et M. Traskin.
Adaboost is consistent.
Journal of Machine Learning Research, 8 :2347-2368, 2007.


T. Zhang.
Sequential greedy approximation for certain convex optimization problems.
IEEE Transaction on Information Theory, 49 :682-691, 2003.


B. Efron, T. Hastie, I. Johnstone, R. Tibshirani.
Least Angle Regression.
Annals of Statistics 32(2), 407-499, 2004.


Y. Freund, R. Iyer, R.E. Schapire, Y. Singer
An efficient boosting algorithm for combining preferences.
Journal of machine learning research 4 : 933-969, 2003.


J. Zhu, S. Rosset, H. Zou, T. Hastie
Multi-class AdaBoost.
Technical report, 2005.