Bagging et Boosting pour la Classification Binaire Supervisée

À partir d'un échantillon d'apprentissage $\mathcal{D}_n = \{(X_i, Y_i) : \ 1 \leq i \leq n\} \in (\mathbb{R}^d \times \{-1, 1\})^n$ (où $Y_i \in \{-1, 1\}$) on va proposer des méthodes pour améliorer les performances de certaines collections de classifieurs (binaire dans ce TP). Un classifieur peut s'écrire comme $\hat{h}(\mathcal{D}_n)$ où l'on fait transparaître la dépendance en l'échantillon observé, encore appelé échantillon d'apprentissage.

Bagging

Le Bagging [1] repose sur la méthode boostrap pour améliorer la prédiction d'une classifieur. On va donc tirer uniformément et avec remise les observations pour créer de nouveaux échantillons, sur chacun desquels on construit un classifieur. On obtient le classifieur final par un vote/consensus à la majorité parmi les classifieurs intermédiaires. L'idée majeure est donc d'obtenir divers classifieurs, basés sur de nombreux échantillons. L'algorithme peut alors s'écrire de la manière suivante :

Algorithme 1: Bagging

Data : les observations et leurs étiquettes $\mathcal{D}_n = \{(X_i, Y_i) : \ 1 \leq i \leq n\}$, le nombre de répliques d'échantillons B

Result : Un classifieur $\hat{h}^B_{\text{bag}}$

for $b = 1$ *to* B **do**

  Tirer un échantillon "boostrap" (avec remise dans $\mathcal{D}_n$) $\mathcal{D}^b_n = \{(X^b_i, Y^b_i) : \ 1 \leq i \leq n\}$;

  Calculer le classifieur sur cet échantillon : $\hat{h}^b(x) = \hat{h}(\mathcal{D}^b_n, x)$;

$$\hat{h}^B_{\text{bag}}(x) = \underset{k \in \{-1,1\}}{\arg\max} \, \text{Card}\{b \in [\![1, B]\!] \ : \ \hat{h}^b(x) = k\}$$

Reprendre la fonction qui tire N données selon le modèle bi-dimensionel des TP précédents ($d = 2$).

1. Coder une fonction qui prend en entrée un tel échantillon $\mathcal{D}_n$ et qui renvoie un échantillon boostrappé.
2. Proposer un fonction qui prend en entrée $\mathcal{D}_n$, un nombre de répliques B, et un tableau de points (sur lesquels on fera la validation et l'illustration graphique) et qui renvoie un classifieur obtenu par bagging avec B répliques pour la méthodes des k-NN.
3. Afficher les résultats pour les k-NN, avec $k = 3$ à la fois pour la version avec et sans bagging.

Boosting

D'un point de vue historique, un des premiers algorithmes de Boosting à rencontrer un réel succès s'appelle "l'AdaBoost.M1" et a été propsé par Freund et Schapire en 1997 [2]. On pourra pour cette partie se servir des articles suivants : [3, 4]. Quelques informations supplémentaires en lien avec [3], sont aussi disponibles sur la page de l'auteur http://www-stat.stanford.edu/~jhf/R-MART.html et dans [5, Chapitre 10].

- Théorie -

Rappelons que l'on note η la fonction de régression $\eta(x) \stackrel{\Delta}{=} \mathbb{E}(\mathbb{1}_{\{Y=1\}} | X = x) = f^*_{\varphi_0}(x)$, et que le classifieur de Bayes qui lui est associé est $h^*_{\varphi_0} = \text{sign}(\eta)$ minimise le risque de classification

$$R_{\varphi_0}(f) = \mathbb{P}(Y \neq f(X)) = \mathbb{P}(-Yf(X) \geq 0) = \mathbb{E}(\varphi_0 [-Yf(X)])$$

avec $\varphi_0 = \mathbb{1}_{\mathbb{R}^+}$ et donc $f^*_{\varphi_0} = \arg\min_f R_{\varphi_0}(f) = \text{sign}(\eta)$ où le min porte sur toute les fonction mesurables.

$$f^*_{\varphi_0} = \underset{f : \mathbb{R}^d \mapsto \{-1,1\}}{\arg\min} \ R_{\varphi_0}(f), \ \text{où} \ R_{\varphi_0}(f) = \mathbb{E}(\varphi_0 [-Yf(X)]).$$

En pratique on n'a pas accès à la distribution inconnue de la loi jointe des observations. On cherche donc à optimiser sa contrepartie empirique (méthodes des moments). Nous allons voir comment il s'exprime, et légitimer son introduction par quelques rappels de cours.

$$\hat{f}_{n,\varphi_0} = \arg\min_f R_{n,\varphi_0}(f), \text{ où } R_{n,\varphi_0}(f) = \mathbb{E}_n(\varphi_0\left[-Yf(X)\right]) \triangleq \frac{1}{n}\sum_{i=1}^{n}\varphi_0(-Y_i f(X_i))$$

et le classieur associé est donc $\hat{h}_{n,\varphi_0} = \text{sign}(\hat{f}_{n,\varphi_0})$.

Supposons par la suite que l'on a M classifieurs experts disponibles, ou de manière équivalente que l'on dipose de $f_1, \ldots, f_M$ (*i.e.,* un dictionnaire fini de taille M). On peut alors chercher à résoudre le programme suivant

$$\hat{f}^M_{n,\varphi_0} = \underset{h\in \text{Conv}(h_1,\ldots,h_M)}{\arg\min} R_{n,\varphi}(h), \text{ où } \varphi_0 \leq \varphi \text{ et } \varphi \text{ est convexe.}$$

4. Démontrez le lemme suivant

Lemme 1. *Le minimiseur de la fonction* $h \to R_{\varphi_{\exp}}(h) = \mathbb{E}(\exp(-Yh(x)))$ *est atteint en* $h^*_{\varphi_{\exp}} = \frac{1}{2}\log\frac{\eta(x)}{1-\eta(x)}$. *[4, p. 215]*

Comme l'optimisation d'une fonction non-convexe (qui plus est sur un espace non convexe : la moyenne d'un classifieur n'est pas forcément un classifieur) n'est pas toujours une mince affaire, on ne cherche des solutions du programme où l'on restreint les choix possibles de candidats. L'algorithme AdaBoost vise à minimiser une version convexifiée R_{n,φ_0} du risque empirique correspondant à la fonction de perte exponentielle.

Algorithme 2: AdaBoost

Data : les observations et leurs étiquettes $\mathcal{D}_n = \{(X_i, Y_i) : 1 \leq i \leq n\}$, le nombre d' étapes M, un vecteur poids $w^0 \in \mathbb{R}^N$ (généralement on choisit $w^0_i = \frac{1}{N}$ pour tout $i \in [\![1, N]\!]$)

Result : Un classifieur $\hat{h}^M_{\text{boost}}$

for $m = 1$ *to* M **do**

Ajuster un classifieur (faible) $\hat{f}_m$ avec une distribution des observations suivant le poids w^{m-1} ;

Calculer l'erreur associée $c_m = \log[(1 - \text{err}_\text{m})/\text{err}_\text{m})]$ où

$\text{err}_\text{m} = \mathbb{P}_\text{w}(\text{Y} \neq \hat{\text{f}}_\text{m}(\text{X})) = \sum_{\text{i}=1}^{\text{n}} \text{w}_\text{i}^{\text{m}-1}\mathbb{1}_{\{\text{Y}_\text{i}\neq\hat{\text{f}}_\text{m}(\text{X}_\text{i})\}}$;

Mise à jour des poids : $w^m_i = w^{m-1}_i \exp(c_m \cdot \mathbb{1}_{\{Y_i\neq\hat{f}_m(X_i)\}})$ et normalisation : $\sum_{i=1}^N w^m_i = 1$

$\hat{h}^M_{\text{boost}} = \text{sign}(\sum_{m=1}^{M} c_m\hat{f}_m)$

REM : on voit que l'algorithme augmente le poids des données qui sont mal classées ($Y_i \neq f_m(X_i)$) par le m^e expert pour qu'on prête plus attention à eux à l'étape suivante.

De plus on peut expliquer le choix des coefficients de pondération de la façon suivante. Ayant déjà à disposition un classifieur $\hat{F}_{m-1} = \sum_{k=1}^{m-1} c_k\hat{f}_k$ à l'étape m, on va essayer d'ajouter une contribution de notre nouvelle proposition $\hat{f}_m$pour minimiser le $\varphi_{\exp}$-risque. On veut donc obtenir la solution du programme suivant

$$\hat{h}^m_{n,\varphi_{\exp}} = \text{sign}(\sum_{k=1}^{m-1} c_k\hat{f}_k + c_m\hat{f}_m), \text{ où } c_m = \underset{c\in\mathbb{R}}{\arg\min}\,\mathbb{E}\left[\exp(-Y\left(\sum_{k=1}^{m-1} c_k\hat{f}_k(X) + c\cdot\hat{f}_m(X)\right)\right]$$

Mais comme $\mathbb{E}$ est inconnue, on cherche plutôt à résoudre la contrepartie empirique :

$$\hat{h}^m_{n,\varphi_{\exp}} = \text{sign}(\sum_{k=1}^{m-1} c_k\hat{f}_k + c_m\hat{f}_m), \text{ où } c_m = \underset{c\in\mathbb{R}}{\arg\min}\,\mathbb{E}_n\left[\exp(-Y\left(\sum_{k=1}^{m-1} c_k\hat{f}_k(X) + c\cdot\hat{f}_m(X)\right)\right]$$

En notant $\hat{F}_{m-1} = \sum_{k=1}^{m-1} c_k\hat{f}_k$

$$\arg\min_{c\in\mathbb{R}} \mathbb{E}_n \left[\exp(-Y\left(\sum_{k=1}^{m-1} c_k \hat{f}_k(X) + c\cdot \hat{f}_m(X)\right)\right] = \arg\min_{c\in\mathbb{R}} \mathbb{E}_{\omega^{m-1}} \left[\exp(-c\cdot Y\cdot \hat{f}_m(X))\right]$$

où $\omega_i^{m-1} \propto \exp(-Y_i \hat{F}_{m-1}(X_i))$.

5. Montrer que le c solution du dernier programme d'optimisation est : $\frac{1}{2}\log\left(\frac{\mathbb{P}_\omega(Y=\hat{f}_m)}{\mathbb{P}_\omega(Y\neq\hat{f}_m)}\right)$
6. Montrer que les poids $\omega^m \propto \omega^{m-1}\cdot\exp(-Y_i\hat{f}_m)$ et les poids $w^m \propto w^{m-1}\cdot\exp(c_m\cdot \mathbb{1}_{\{Y_i\neq\hat{f}_m(X_i)\}})$ ont les même équations de mise à jour, et la même initialisation (avec la convention $\hat{F}_0 = 0$).

- Application -

7. Ecrire une routine mettant en oeuvre AdaBoost avec des arbres de profondeur 1 ("stumps"), puis 2 (sans élagage).
8. Appliquer AdaBoost sur les données relatif au cancer du sein de l'UCI data repository http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic), découpées en deux échantillons : apprentissage et test. Tracer les erreurs d'apprentissage et de test en fonction du nombre d'itérations du boosting (pour le risque non convexifié).
9. Que remarquez vous ? Que se passe-t-il si la profondeur des arbres de classification est beaucoup plus grande ?
10. Comparer les performances sur les données "Breast Cancer" de l'UCI.

- Généralisation -

11. Dans le cas où l'on considère la fonction "logit"

$$\varphi_{logit}(y,f) = \log\left(\frac{1}{1+e^{2t}}\right),$$

calculer le minimiseur du risque théorique ainsi convexifié $R_{\varphi_{\text{logit}}}$. Que devient la procédure itérative décrite précédemment dans ce cas ?

Références

[1] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2) :123–140, 1996. 1

[2] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1) :119–139, 1997. 1

[3] J. Friedman. Greedy function approximation : a gradient boosting machine. *Ann. Statist*, 29(5) :1189–1232, 2001. http://www-stat.stanford.edu/~jhf/ftp/trebst.pdf. 1

[4] J. Friedman, T. Hastie, and Robert R. Tibshirani. Additive logistic regression : a statistical view of boosting. *Ann. Statist.*, 28(2) :337–407, 2000. http://www-stat.stanford.edu/~jhf/ftp/boost.pdf. 1, 2

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer Series in Statistics. Springer, New York, second edition, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 1