Devoir 3

à remettre *au plus tard* le vendredi 20 avril à 23:59

Vous avez la possibilité de remettre votre rapport sous un format électronique ou en version papier (auprès du professeur ou des démonstrateurs). Tous les fichiers de code source que vous aurez créés ou adaptés, et les fichiers de résultat produits (ex. courbes) devront en faire partie.

Pour remettre tous ces fichiers de manière électronique, connectez-vous sur la machine remise (`ssh remise`), puis utilisez la commande remise:

- Si vous êtes inscrits en IFT3390, faites par ex. `remise ift3390 tp3 rapport.pdf *.plearn`
- Si vous êtes inscrits en IFT6390, faites par ex. `remise ift6390 tp3 rapport.pdf *.plearn`

1 1. Critère d'arrêt d'AdaBoost vu comme descente de gradient

Soit un ensemble d'entraînement $\{(x_1,y_1),\ldots,(x_m,y_m)\}$ avec $y_i \in \{-1,+1\}$ et soit la perte empirique:

$$\hat{R}(f) = \frac{1}{m}\sum_{i=1}^{m} L(f(x_i),y_i)$$

De plus, étant donné qu'on se trouve à l'itération t de l'algorithme de boosting, on note

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

Vous avez vu que l'interprétation du boosting comme une descente de gradient dans l'espace des fonctions demande que la fonction $h_t(\cdot)$ maximise le produit scalaire $\langle \vec{h}_t, -\nabla\hat{R}(f_{t-1})\rangle$. Normalement, selon cette vue, l'apprentissage cesse lorsque $\langle \vec{h}_t, -\nabla\hat{R}(f_{t-1})\rangle < 0$, i.e. lorsque le vecteur $\vec{h}_t$ est dans le sens opposé à la direction désirée $-\nabla\hat{R}(f_{t-1})$.

Montrez que ce critère d'arrêt est équivalent à celui d'`AdaBoost` lorsque $L(f(x),y) = e^{-yf(x)}$, i.e. montrez que:

$$\langle \vec{h}_t, -\nabla \hat{R}(f_{t-1}) \rangle \leq 0 \Leftrightarrow \sum_{i=1}^{m} \delta_{y_i \neq h_t(x_i)} D_t(i) \geq \frac{1}{2}$$

2 Convergence d'AdaBoost

Au cours de cet exercice, vous allez démontrer que l'erreur d'entraînement d'AdaBoost décroît de manière exponentielle au cours des itérations de l'algorithme. L'algorithme analysé est une des premières versions d'AdaBoost, et est décrit par l'algorithme 1 (cet algorithme est équivalent à celui vu en cours, sous certaines conditions qui ne sont pas importantes ici). Notez en particulier que dans cette formulation de l'algorithme, les classes sont représentées par 0 et 1 plutôt que par -1 et $+1$. En particulier la cible $y_i \in \{0,1\}$, et les classifieurs faibles h_t donnent une répnse dans $0,1$, $h_t : \mathcal{X} \to \{0,1\}$.

Algorithm 1 AdaBoost

Les données d'entraînement sont $\{(x_1,y_1),\ldots,(x_m,y_m)\}$ avec $y_i \in \{0,1\}$
Initialiser $w_i^1 = \frac{1}{m}$ pour tout $i \in \{1,\ldots,m\}$
for $t = 1$ à T **do**
  Soit D^t le vecteur défini par $D_i^t = \frac{w_i^t}{\sum_{j=1}^m w_j^t}$ (i.e. D^t est un vecteur de poids normalisé)
  Apprendre $h_t = WeakLearn(D^t)$, i.e. le "weak learner" appris sur les données pondérées par les poids $\{D_1^t,\ldots D_m^t\}$
  Calculer l'erreur de h_t: $\varepsilon_t \leftarrow \sum_{i=1}^m D_i^t |h_t(x_i) - y_i|$
  $\beta_t \leftarrow \frac{\varepsilon_t}{1-\varepsilon_t}$
  Transformer les poids par $w_i^{t+1} \leftarrow w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$
end for
Renvoyer la solution h_f définie par $h_f(x) = 1$ ssi $\sum_{t=1}^T \left(\ln \frac{1}{\beta_t}\right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \ln \frac{1}{\beta_t}$, et 0 sinon.

1. Soit $a \geq 0$ et $b \in [0,1]$. Montrez que $a^b \leq 1-(1-a)b$. Indice: utilisez la concavité du logarithme. On rappelle que f est concave ssi $-f$ est convexe, et que g est convexe ssi pour tout x,y et tout $\theta \in [0,1]$ on a $g(\theta x + (1-\theta)y) \leq \theta g(x) + (1-\theta)g(y)$.

2. Montrez qu'après chaque itération t d'AdaBoost, on a

$$\sum_{i=1}^m w_i^{t+1} \leq \left(\sum_{i=1}^m w_i^t\right)(1-(1-\varepsilon_t)(1-\beta_t))$$

3. En déduire une borne supérieure sur $\sum_{i=1}^m w_i^{T+1}$, où T est la dernière itération d'AdaBoost. Cette borne ne devra pas contenir de terme de la forme $\sum_{i=1}^m w_i^t$.

4. Montrez que la décision finale donnée par h_f sur l'exemple i est erronée ssi

$$\Pi_{t=1}^{T} \beta_t^{-|h_t(x_i)-y_i|} \geq \left(\Pi_{t=1}^{T} \beta_t\right)^{-1/2}$$

5. Montrez que le poids final de l'exemple i s'écrit

$$w_i^{T+1} = \frac{1}{m} \Pi_{t=1}^{T} \beta_t^{1-|h_t(x_i)-y_i|}$$

6. Utilisez les résultats des questions 4 et 5 pour montrer que

$$\sum_{i=1}^{m} w_i^{T+1} \geq \varepsilon \left(\Pi_{t=1}^{T} \beta_t\right)^{1/2}$$

où ε est l'erreur d'entraînement moyenne, i.e. $\varepsilon = \frac{1}{m} \sum_{i=1}^{m} |h_f(x_i) - y_i|$.

7. Déduisez des questions 3 et 6 une borne supérieure sur ε. Prouvez que cette borne est minimale lorsque $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$, et montrez qu'elle peut alors s'écrire

$$\varepsilon \leq \Pi_{t=1}^{T} (1 - 4\gamma_t^2)^{1/2}$$

avec $\gamma_t = \frac{1}{2} - \varepsilon_t$.

8. Bornez cette expression pour obtenir une nouvelle borne de la forme $\varepsilon \leq \exp\left(-\sum_{t=1}^{T} f(|\gamma_t|)\right)$ où f est une fonction positive et croissante. Indice : utilisez l'inégalité $\ln(1+x) \leq x$ valide pour tout $x \in]-1, +\infty[$.

9. Supposons que lorsqu'un modèle h_t "se trompe" (i.e. $\varepsilon_t > 1/2$), on continue tout de même l'optimisation jusqu'à la fin (étape T). Que nous dit cette borne sur l'erreur d'entraînement ?

3 Expérimentation avec le *Bagging*

Téléchargez le fichier `http://www.iro.umontreal.ca/~dift3390/devoir_3/files/devoir_3.tar.gz` et décompressez-le par `tar zxvf devoir_3.tar.gz`.

Utilisez la classe PLearn `BaggingLearner` pour calculer les erreurs d'entraînement et de test (sur les données fournies), avec un `BinaryStump` comme algorithme de base. Faites un graphique montrant comment ces erreurs évoluent lorsque le nombre de "bags" (c'est-à-dire le nombre de BinaryStumps à entraîner et moyenner) varie de 1 à 1000 (choisissez vous-mêmes l'incrément et l'échelle afin d'avoir une courbe lisible).

Rappels et indices PLearn:

- Vous pouvez vous inspirer du script `tester.plearn` de la démo 7 pour lancer une expérience qui calcule à la fois les erreurs d'entraînement et de test.
- La commande `plearn help ClassName` permet d'avoir l'aide sur une classe de PLearn.
- Vous pouvez utiliser la classe `AddCostToLearner` pour rajouter un coût à un PLearner (par exemple `class_error` ou `binary_class_error`). Remarque : ignorez les options de `AddCostToLearner` qui parlent de "bags", il s'agit de quelque chose complètement différent du bagging.
- Les "headers" de PLearn (fichiers .h) sont dans `$PLEARNDIR`, et le code complet (incluant les .cc) dans `/u/dift3390/PLearn.full`.
- Vous pouvez utiliser la méthode de votre choix pour afficher les graphiques (gnupot, la librairie Python matplotlib utilisée par certains scripts de démo et PLearn, Matlab, ...).
- Si vous avez besoin de convertir une matrice PLearn en ASCII, vous pouvez faire `plearn vmat convert plearn_matrix ascii_matrix.amat` (éditez le fichier résultat à la main si vous voulez enlever l'en-tête PLearn).