Le boosting vu comme une descente de grandient dans l'espcace des fonctions

Pascal Vincent

Avril 2007

Notation

Les notations que j'utilise ici suivent celles utilisées dans le tutoriel sur l'AdaBoost classique de Jiri Matas et Jan Sochman.

Ensemble d'apprentissage:	$\{(x_1, y_1), ..., (x_m, y_m)\}, x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Classifieur faible choisi à l'étape t:	$h_t: \mathcal{X} \rightarrow \{-1, +1\}$
Poids du classifieur h_t:	α_t

La fonction discriminante du classifieur fort à l'étape t est obtenu par une combinaison linéaire de tous les classifieurs faibles obtenus à cette étape:

$$f_t(x) = \sum_{t'=1}^{t} \alpha_t \, h_t(x)$$

La décision du classifieur fort à l'étape t est $H_t(x) = \mathrm{sign}(f_t(x))$

Le classifieur fort final est obtenu à l'étape T et correspond à la fonction dscriminante $f = f_T$ et à la fonction de décision $H = H_T$

Fonction de coût (perte) et descente de gradient dans l'espace des fonctions

Définissons une fonction de coût $L(f(x), y) = e^{-y f(x)}$.

Notez que la quantité $y f(x)$ est positive si $f(x)$ est du mêm signe que y donc que la fonction f prédit la bonne classe pour x. La magnitude de $y f(x)$, qu'on appelle la *marge*, peut être interprêtée comme la confiance qu'a le classifieur f en sa prédiction: si c'est fortment positif alors le classifieur fait la bonne prédiction avec une grande confiance, si c'est fortement négatif il a très confiance en sa prédiction mais se trompe.

A partir d'une telle fonction de coût, on peut définit un coût moyen sur les exemples d'apprentissage, c.a.d. le risque empirique

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

qu'on veut minimiser.

On va à chaque étape t ajouter à f un classifieur faible h_t avec un poids α_t dans le but de réduire le plus possible ce risqe empirique $\hat{R}$.

Comment trouver h_{t+1}

Pour diminuer le risque empirique $\hat{R}$, il faudrait changer f de manière à changer le vecteur de prédicitons $(f(x_1), ..., f(x_m))$ dans la direction opposée du vecteur de gradient:

$$\begin{aligned}
\nabla\hat{R}(f) &= \left(\frac{\partial\hat{R}}{\partial f(x_1)}, ..., \frac{\partial\hat{R}}{\partial f(x_m)}\right)\\
&= \left(\frac{\partial L(f(x_1), y_1)}{\partial f(x_1)}, ..., \frac{\partial L(f(x_m), y_m)}{\partial f(x_m)}\right)\\
&= \left(\frac{\partial e^{-y_1 f(x_1)}}{\partial f(x_1)}, ..., \frac{\partial e^{-y_m f(x_m)}}{\partial f(x_m)}\right)\\
&= \left(-y_1 e^{-y_1 f(x_1)}, ..., -y_m e^{-y_m f(x_m)}\right)
\end{aligned}$$

Pour ce faire, on va chercher un $h_{t+1} = h$ (à ajouter à f_t) qui donnerait un vecteur $\vec{h} = (h(x_1),\ ...,\ h(x_m))$ allant le plus possible dans le sens opposé de ce gradient, c.a.d. qu'on va chercher une fonction h qui maximise le produit scalaire $\left\langle \vec{h}, -\nabla\hat{R}(f)\right\rangle$.

$$h_{t+1} = \mathrm{argmax}_h \left[\sum_{i=1}^{m} h(x_i)\, y_i\, e^{-y_i f_t(x_i)}\right]$$

Si on suppose que h est un classifieur binaire donnant une réponse dans $\{-1, +1\}$, on peut réécrire $h(x_i) y_i = 1 - 2\delta_{y_i \neq h(x_i)}$, où δ représente la fonction indicatrice, donc

$$\begin{aligned}
h_{t+1} &= \mathrm{argmax}_h \left[\sum_{i=1}^{m} (1 - 2\delta_{y_i\neq h(x_i)})\, e^{-y_i f_t(x_i)}\right]\\
&= \mathrm{argmax}_h \left[\left(\sum_{i=1}^{m} e^{-y_i f_t(x_i)}\right) - 2\left(\sum_{i=1}^{m} \delta_{y_i\neq h(x_i)} e^{-y_i f_t(x_i)}\right)\right]\\
&= \mathrm{argmax}_h \left[-\sum_{i=1}^{m} \delta_{y_i\neq h(x_i)} e^{-y_i f_t(x_i)}\right]\\
&= \mathrm{argmin}_h \left[\sum_{i=1}^{m} \delta_{y_i\neq h(x_i)} D_{t+1}(i)\right]
\end{aligned}$$

avec $D_{t+1}(i) = \frac{e^{-y_i f_t(x_i)}}{Z'_t}$ où Z'_t est un simple facteur de normalisation afin que D_{t+1} puissent être interprété comme une distribution sur les x_i, c.a.d. des poids sur les exemples qui somment à 1. Notez que $\delta_{y_i\neq h(x_i)}$ vaut 0 si $h(x_i)$ classifie correctement x_i et 1 si h se trompe.

On peut alors écrire

$$h_{t+1} = \mathrm{argmin}_h\, \mathbf{E}_{x\sim D_{t+1}}[\delta_{y_i\neq h(x_i)}]$$

où $\mathbf{E}_{x\sim D_{t+1}}$ indique l'espérance pour des x tirés selon la distribution empirique pondérée donnée par les paires exemple poids $(x_i, D_{t+1}(i))$. On peut interpréter cette minimisation comme la minimisation du taux d'erreur de classification pour cet ensemble pondéré de points avec les poids D_{t+1}: lorsqu'on se trompe pour un point x_i qui a un poids $D_{t+1}(i)$ on paye un coût $D_{t+1}(i)$.

On peut donc trouver h_t en entraînant un classifieur faible sur l'ensemble d'entraînement repondéré $\{(x_1, y_1, D_{t+1}(1)), ..., (x_m, y_m, D_{t+1}(m))\}$.

On notera que

$$\begin{aligned} D_{t+1}(i) &= \frac{\exp(-y_i f_t(x_i))}{Z'_t} \\ &= \frac{\exp\Big(-y_i \sum_{k=1}^{t} \alpha_k h_k(x_i)\Big)}{Z'_t} \\ &= \frac{D_t(i)\exp(-\alpha_t\, y_i\, h_t(x_i))}{Z_t} \end{aligned}$$

ce qui correspond à la formule (récurrente) de repondération de l'algorithme AdaBoost classique, en posant $D_1(i) = \frac{1}{m}$ et avec les Z_t des facteurs de normalisation tels que D_{t+1} soit une distribution (c.a.d. que les $D_t(i)$ somment à 1) et on a alors $Z'_t = m \prod_{q=1}^{t} Z_q$.

Comment trouver α_{t+1}

Une fois qu'on a trouvé h_{t+1}, on va chercher le α_{t+1} qui minimise $\hat{R}(f_{t+1}) = \hat{R}(f_t + \alpha_{t+1}\, h_{t+1})$:

$$\alpha_{t+1} = \mathrm{argmin}_\alpha \frac{1}{m}\sum_{i=1}^{m} L(f_t(x_i) + \alpha h_{t+1}(x_i), y_i)$$

Le α optimal est celui qui va annuler le gradient

$$\frac{\partial \hat{R}(f_t + \alpha\, h_{t+1})}{\partial \alpha} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial L(f_t(x_i) + \alpha h_{t+1}(x_i), y_i)}{\partial \alpha}$$

Pour la fonction de coût L indiquée ci-haut, ceci peut se résoudre analytiquement et donne la formule pour α_{t+1} de l'algorithme AdaBoost classique.

Intérêt de cette façon de voir le boosting

Voir le boosting comme une descente de gradient dans l'espace des fonctions permet de généraliser la technique à d'autres fonctions de pertes, ou à des apprenants faibles qui ne sont pas des classifieurs binaires, ...