Projet d'Advanced Machine Learning : AdaBoost

Charly Delfosse, Arnaud Palgen, Victor Dheur

28 décembre 2020

1 Principe du boosting

Le boosting est une méthode permettant d'augmenter les performances d'algorithmes apprenants faibles (voir def. 1). Cette méthode permet aussi de résoudre deux problèmes rencontrés lors de l'apprentissage :

1. Le tradeoff biais-complexité
2. La complexité des calculs

Le principe général du boosting consiste à commencer par une hypothèse de base, qui est ajustée à chaque itération de l'algorithme pour produire une hypothèse plus précise.

Définition 1. *Algorithme γ apprenant faible*

Un algorithme A est γ apprenant faible (weak learner) pour une classe $\mathcal{H}$ s'il existe une fonction $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ tel que pour tout $\delta \in (0,1)$, pour toute distribution $\mathcal{D}$ sur $\mathcal{X}$ et pour chaque fonction de labelisation $f : \mathcal{X} \to \{\pm 1\}$, si l'hypothèse est valable pour $\mathcal{H}$, $\mathcal{D}$, $\mathcal{F}$, alors lors de l'execution de l'algorithme d'apprentissage sur $m > m_{\mathcal{H}}(\delta)$ i.i.d exemples générés par $\mathcal{D}$ et labelisés par f, l'algorithme retourne, avec une probabilité $1-\delta$, une hypotèse h tel que $L_{(\mathcal{D},f)}(H) \leq \frac{1}{2} - \gamma$

2 Adaboost

Adaboost (adaptative boosting) est une technique de boosting très répandue. L'article [] (chapitre 10) présente son fonctionnement. Dans cette section, on explique le principe d'Adaboost et on présente son algorithme.

2.1 Principe

Le principe d'Adaboost est de modifier le processus d'apprentissage d'un weak learner pour que celui-ci se concentre sur les exemples les plus pertinents. Adaboost va en fait itérer un certain nombre de fois le même procédé. A chaque étape, le weak learner s'entraine sur le même jeu de données mais ne considère pas de la même façon les exemples par rapport à l'étape précédentes. Il se concentre en fait sur les exemples les plus problématiques, ceux-ci sont désignés par Adaboost. Adaboost répète ce procédé un certain nombre de fois et ensuite combine les différentes hypothèses obtenues à chaque étape pour fournir l'hypothèse finale. Le but d'Adaboost est de tirer le meilleur profit possible du tradeoff biais-variance en essayant d'avoir une

hypothèse finale avec une erreur d'entrainement la plus petite possible sans être trop complexe.

2.2 Algorithme

On présente maintenant l'algorithme d'Adaboost, on fournit d'abord une description étape par étape et on rentre ensuite dans les détails des points importants. Le pseudo-code du processus Adaboost est repris par l'algorithme 1.

Description

Soit f une fonction cible, soit $S = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ le jeu de données d'entrainement tel que $\forall i, 1 \geq i \geq m, f(x_i) = y_i$, soit $T \geq 1$ un naturel (non nul) représentant le nombre d'itérations de l'algorithme Adaboost. D'abord, Adaboost génère une distribution $D^{(0)} \in R_+^m$ telle que $\forall i, 1 \geq i \geq m, D_i^{(0)} = \frac{1}{m}$ (distribution équitable). Cette distribution représente l'importance que doit accorder le weak learner à chaque exemple. De fait, plus le poids $D_i^{(t)}$ $(1 \geq i \geq m$ et $1 \geq t \geq T)$ est grand, plus celui-ci doit accorder d'importance à l'exemple (x_i, y_i) et inversement. Une fois que cela est fait, Adaboost va itérer T fois le même procédé. A chaque itération, le weak learner s'entraine sur le jeu de données S. L'erreur d'entrainement $E_t(h_t)$ du weak learner est calculée en prenant compte de $D^{(t)}$:

$$E_t(h_t) = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[h_t(x_i) \neq y_i]} \tag{1}$$

Comme on le voit dans (1), plus le poids d'un exemple (dans $D^{(t)}$) est grand, plus il a d'importance dans le calcul de l'erreur. De plus, par définition du weak learner, il y a une grande probabilité $(1 - \delta)$ que $E_t(h_t) < \frac{1}{2} - \gamma$. Ainsi, à l'itération t, le weak learner fournit une hypothèse h_t et l'erreur $E_t(h_t)$ associée (1), Adaboost détermine ensuite la nouvelle distribution $D^{(t+1)}$ (2.2) et passe à l'itération $t + 1$. Une fois que les T itérations sont terminées, Adaboost combine les différentes hypothèses h_t pour obtenir l'hypothèse finale h_f :

$$h_f(x) = sign(\sum_{t=1}^{T} w_t h_t(x)) \tag{2}$$

Dans (2), on remarque que l'hypothèse h_t a un poids w_t, celui-ci est en fait calculé à l'itération t après le calcul de l'erreur : $w_t = \frac{1}{2} log(\frac{1}{E_t(h_t)} - 1)$. Plus l'erreur d'entrainement de l'hypothèse h_t est petite, plus elle aura d'importance dans l'hypothèse finale h_f et inversement. Adaboost essaie ainsi de donner plus d'importances aux hypothèses prometteuses qu'aux hypothèses moins performantes.

Modification de la distribution D

Adaboost modifie donc la distribution $D^{(t)}$ après chaque itération :

$$\forall i, 1 \geq i \geq m, D_i^{(t+1)} = \frac{D_i^{(t)} e^{-w_t y_i h_t(x_i)}}{\sum_{j=1}^{m} D_j^{(t)} e^{-w_t y_j h_t(x_j)}} \tag{3}$$

Dans (3) on remarque que le nouveau poids $D_i^{(t+1)}$ de l'exemple (x_i, y_i) est proportionnel à son ancien poids D_i^t, cela permet à Adaboost de limiter la variance. De plus, si on suppose que $w_t > 0$ (ce qui est vrai dans la majorité des cas car $w_t > 0 \Leftrightarrow E_t(h_t) < \frac{1}{2}$ et il y a une probabilité supérieure à $1 - \delta$ que cela soit vrai (en effet, $E_t(h_t) < \frac{1}{2} - \gamma$ avec une probabilité $1 - \delta$)), on a que :

- si $sign(y_i) = sign(h_t(x_i))$ alors $e^{-w_t y_i h_t(x_i)} < 1$,
- si $sign(y_i) \neq sign(h_t(x_i))$ alors $e^{-w_t y_i h_t(x_i)} > 1$.

Ainsi, si l'hypothèse h_t avait fait la bonne prédiction pour l'exemple (x_i, y_i) alors le nouveau poids $D_i^{(t+1)}$ sera plus grand que si elle s'était trompée. Cela est en accord avec le fait qu'Adaboost incite le weak learner à se concentrer sur les exemples problématiques (et donc pertinents). Le dénominateur est uniquement la pour normaliser et assurer la définition de distribution : $\forall 1 \geq t \geq T, \sum_{i=1}^m D_i^t = 1$.

Algorithm 1 Adaboost

INPUT : $S = (x_1, y_1)(x_2, y_2), ..., (x_m, y_m)$, le jeu de données d'entrainement,
WL, un weak learner,
T, un naturel (non nul) représentant le nombre d'itérations d'Adaboost.
OUTPUT : l'hypothèse finale h_f

function $Adaboost(S, WL, T)$
  $D^{(1)} = (\frac{1}{m}, ..., \frac{1}{m})$
  for $t = 1, ..., T$ **do**
    $h_t = WL(D^{(t)}, S)$
    $E_t(h_t) = \sum_{i=1}^m D_i^{(t)} \mathbb{1}_{[h_t(x_i) \neq y_i]}$
    $w_t = \frac{1}{2} log(\frac{1}{E_t(h_t)} - 1)$
    for $i = 1, ..., m$ **do**
      $D_i^{(t+1)} = \frac{D_i^{(t)} e^{-w_t y_i h_t(x_i)}}{\sum_{j=1}^m D_j^{(t)} e^{-w_t y_j h_t(x_j)}}$
  $h_f(x) = sign(\sum_{t=1}^T w_t h_t(x))$
  return h_f

3 Bornes sur l'erreur de généralisation

On montre maintenant comment obtenir des bornes sur l'erreur de généralisation d'AdaBoost. Nous commençons par obtenir la dimension VC d'AdaBoost, puis nous appliquons l'inégalité VC. Cette approche est tirée du livre [SS14], et des détails supplémentaires ont été ajoutés.

Nous avons vu que la sortie de l'algorithme AdaBoost est une hypothèse composée d'une combinaison linéaire d'hypothèses faibles. AdaBoost se base sur un weak-learner dont l'espace d'hypothèses est dénoté B. T hypothèses $h_1, ..., h_T$ sont créées par ce weak-learner. La sortie d'AdaBoost fait partie de cet ensemble d'hypothèses :

$$L(B, T) = \left\{ x \mapsto \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x) \right) \mid \forall t, w_t \in \mathbb{R} \wedge h_t \in B \right\}.$$

Pour un espace d'hypothèses $\mathcal{H}$, nous dénotons $d_{VC}(\mathcal{H})$ sa dimension VC et $m_{\mathcal{H}}$ sa growth function.

Théorème 1. *(Dimension VC d'AdaBoost)*
Nous allons montrer que, lorsque $T \geq 3$ et $d_{VC}(B) \geq 3$:

$$d_{VC}(L(B,T)) \leq T(d_{VC}(B)+1)(3\ln(T(d_{VC}(B)+1))+2).$$

Démonstration. Dénotons $d = d_{VC}(B)$ et supposons que $T \geq 3$ et $d_{VC}(B) \geq 3$. Soit $C = (x_1, ..., x_m)$ une séquence de points qui est shattered par $L(B,T)$. La création d'un labeling de C par une hypothèse $h \in L(B,T)$ se fait en 2 étapes. D'abord, T hypothèses $h_1, ..., h_T \in B$ sont sélectionnées par le weak-learner. Ensuite, un vecteur $w \in \mathbb{R}^T$ permet de créer la combinaison linéaire $\sum_{t=1}^{T} w_t h_t(x)$ pour un point x. On obtient ainsi un labeling $(h(x_1), ..., h(x_m))$ de C.

Nous allons utiliser le lemme de Sauer, qui permet de borner supérieurement la growth function $m_{\mathcal{H}}$ d'un espace d'hypothèses $\mathcal{H}$ en utilisant la VC dimension $d_{VC}(\mathcal{H})$:

$$m_{\mathcal{H}}(m) \leq \left(\frac{em}{d_{VC}(\mathcal{H})}\right)^{d_{VC}(\mathcal{H})}.$$

Par le lemme de Sauer, au plus $\left(\frac{em}{d}\right)^d$ labelings différents de C peuvent être créés à partir de l'espace d'hypothèses B. De plus, T hypothèses qui créent ces labelings doivent être choisies, ce qui donne au plus $\left(\frac{em}{d}\right)^{dT}$ labelings différents. En utilisant encore le lemme de Sauer, puisque la dimension VC d'un perceptron (sans biais) dans $\mathbb{R}^T$ est de T, la combinaison linéaire entraîne $\left(\frac{em}{T}\right)^T$ labelings différents. Nous avons donc :

$$m_{L(B,T)}(m) \leq \left(\frac{em}{d}\right)^{dT}\left(\frac{em}{T}\right)^{T}.$$

En utilisant les hypothèses que $T \geq 3$ et $d_{VC}(B) \geq 3$, nous avons :

$$\left(\frac{em}{d}\right)^{dT}\left(\frac{em}{T}\right)^{T} \leq m^{(d+1)T}.$$

Puisque C est shattered par $L(B,T)$, $m_{L(B,T)}(m) = 2^m$.
Nous avons donc :

$$2^m = m_{L(B,T)}(m) \leq m^{(d+1)T}$$

En passant au log :

$$m \leq \ln(m)\frac{(d+1)T}{\ln(2)}$$

Il est possible de montrer (voir [SS14] p.419 lemme A.1) que

$$\forall a > 0, x \leq a\ln(x) \implies x \leq 2a\ln(a).$$

On déduit une borne sur m qu'on borne encore par une expression plus simple :

$$m \leq \frac{2(d+1)T}{\ln(2)}\ln\frac{(d+1)T}{\ln(2)} \leq (d+1)T(3\ln((d+1)T)+2).$$

En d'autres termes, le nombre de points m qui peuvent être shattered par $L(B,T)$ est borné supérieurement par une expression qui dépend de d et T. Puisque la

dimension VC correspond au nombre maximum de points qui peuvent être shattered, l'expression reste vraie lorsque $m = d_{VC}(L(B,T))$:

$$d_{VC}(L(B,T)) \leq m \leq (d+1)T(3\ln((d+1)T)+2).$$

□

Il ne reste plus qu'à utiliser l'inégalité VC en bornant la growth function par le lemme de Sauer pour avoir une borne sur l'erreur de généralisation. Nous utilisons l'inégalité VC présentée dans [BBLR03] à la page 192.

En supposant que la loss produit des valeurs bornées dans $[0,1]$, pour toute précision $\epsilon > 0$, on obtient :

$$\begin{aligned}\mathbb{P}\left[\sup_{h\in L(B,T)}(R(h)-R_m(h)) \geq \epsilon\right] &\leq 4m_{L(B,T)}(2m)e^{-m\epsilon^2/8} \\ &\leq 4\left(\frac{2me}{d_{VC}(L(B,T))}\right)^{d_{VC}(L(B,T))} e^{-m\epsilon^2/8}.\end{aligned}$$

Un point important de ce développement est que la dimension VC de l'ensemble des hypothèse produites par AdaBoost augmente linéairement avec la dimension VC de B et avec T, en ignorant les facteurs constants et logarithmiques.

Références

[BBLR03] Olivier Bousquet, Stéphane Boucheron, Gábor Lugosi, and Gunnar Rätsch. Introduction to statistical learning theory. January 2003.

[SS14] Shai Shalev-Shwartz. *Understanding Machine Learning (From Theory to Algorithms)*. Cambridge University Press, 1 edition, May 2014.