Projet d'Advanced Machine Learning : AdaBoost

Charly Delfosse, Arnaud Palgen, Victor Dheur

14 janvier 2021

Introduction

Dans le domaine du machine learning, on recherche l'hypothèse qui s'approche le plus d'une fonction cible. Cette recherche peut être couteuse en temps de calcul et si celle-ci est trop poussée, l'hypothèse résultante pourrait être trop complexe et trop spécifique au jeu de données d'entrainement (tradeoff biais-complexité). Pour solutionner ces problèmes, on utilise le boosting : une approche permettant d'augmenter les performances d'algorithmes de recherche d'hypothèse simple (weak learners). Ce rapport explique le principe général du boosting, présente en détails l'algorithme de boosting AdaBoost et les performances de celui-ci par rapport aux bornes sur l'erreur de généralisation.

1 Principe du boosting

Le boosting est une méthode permettant d'augmenter les performances d'algorithmes weak learners (voir def. 2). 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. *Soit $f : \mathcal{X} \to \mathcal{Y}$ la fonction cible de labellisation et $\mathcal{D}$ une distribution des probabilités sur $\mathcal{X}$ qui assigne une probabilité à chaque élément de $\mathcal{X}$. On définit $L_{(\mathcal{D},f)}(h)$ comme étant l'erreur d'une hypothèse $h : \mathcal{X} \to \mathcal{Y}$ telle que $L_{(\mathcal{D},f)} = \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)]$.*

Définition 2. *Dans la classification binaire, un algorithme A est un γ-weak learner pour une classe d'hypothèses $\mathcal{H}$ s'il existe une fonction $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ telle que, pour tout $\delta \in (0,1)$, pour toute distribution des probabilités $\mathcal{D}$ sur $\mathcal{X}$ et pour chaque fonction de labellisation binaire $f : \mathcal{X} \to \{\pm 1\}$, si on émet l'hypothèse qu'il existe une hypothèse $h \in \mathcal{H}$ telle que $L_{D,f}(h) = 0$, alors, lors de l'exécution de l'algorithme d'apprentissage sur $m \geq m_{\mathcal{H}}(\delta)$ exemples (indépendants et identiquement distribués) générés par $\mathcal{D}$ et labelisés par f, l'algorithme retourne, avec une probabilité $1-\delta$, une hypothèse h tel que $L_{(\mathcal{D},f)}(h) \leq \frac{1}{2} - \gamma$ ($\gamma \in [0, \frac{1}{2}]$).*

Dans la définition 2, le terme γ est là pour exprimer le fait que l'erreur sur l'hypothèse d'un weak learner doit être légèrement meilleure (plus proche de zéro) que l'erreur faite par un classifieur aléatoire. La borne sur l'erreur est donc beaucoup moins contraignante que la borne imposée par les classifieurs PAC (probablement approximativement correct).

2 AdaBoost

AdaBoost (adaptative boosting) est une technique de boosting très répandue. L'article [BBLR03] (chapitre 10) présente son fonctionnement. Dans cette section, on explique le principe d'AdaBoost pour une classification binaire 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 précédente. 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 combine ensuite 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-complexité 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 et $S = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ le jeu de données d'entrainement tel que $\forall 1 \leq i \leq m, f(x_i) = y_i$. Soit $T \geq 1$ un naturel représentant le nombre d'itérations de l'algorithme AdaBoost. D'abord, AdaBoost génère une distribution $D^{(0)} \in \mathbb{R}_+^m$ telle que $\forall 1 \leq i \leq m, D_i^{(0)} = \frac{1}{m}$ (distribution équitable). Cette distribution représente l'importance que doit accorder le weak learner à chaque exemple. De fait, plus le poids $D_i^{(t)}$ ($1 \leq i \leq m$ et $1 \leq t \leq T$) 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 $R_m(h_t)$ du weak learner (qui produit l'hypothèse h_t) est calculée selon $D^{(t)}$:

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

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 $R_m(h_t) < \frac{1}{2} - \gamma$ ($\gamma \in [0, \frac{1}{2}]$). Ainsi, à l'itération t, le weak learner fournit une hypothèse h_t et l'erreur $R_m(h_t)$ 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 (classification binaire) :

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

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 : $w_t = \frac{1}{2}\log(\frac{1}{R_m(h_t)} - 1)$. Plus l'erreur d'entrainement de l'hypothèse h_t est petite, plus elle aura d'importance dans l'hypothèse finale h et inversement. AdaBoost essaie ainsi de donner plus d'importance aux hypothèses prometteuses qu'aux hypothèses moins performantes.

Modification de la distribution D

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

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

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 $w_t > 0$, on a que (pour une classification binaire) :

- 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 $e^{-w_t y_i h_t(x_i)} > 1$.

L'hypothèse précédente est vraie dans la majorité des cas car $w_t > 0 \Leftrightarrow R_m(h_t) < \frac{1}{2}$ et il y a une probabilité supérieure à $1-\delta$ que cela soit vrai, en effet, $R_m(h_t) < \frac{1}{2} - \gamma$ ($\gamma \in [0, \frac{1}{2}]$) avec une probabilité $1 - \delta$. 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 là pour normaliser et assurer la définition de distribution : $\forall 1 \leq t \leq T, \sum_{i=1}^{m} D_i^t = 1$.

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

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

3 Bornes sur l'erreur de généralisation

Nous montrons maintenant comment obtenir des bornes sur l'erreur de généralisation d'AdaBoost. Nous commençons par obtenir la dimension VC d'AdaBoost, puis nous appliquons une inégalité basée sur l'inégalité de Hoeffding. Le développement concernant la dimension VC est tiré 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 de T hypothèses $h_1, ..., h_T$ apprises par un weak learner. Nous dénotons l'espace d'hypothèses de ce weak learner par B. La sortie d'AdaBoost fait partie de cet ensemble d'hypothèses que nous nommons L(B, T) :

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

Pour un espace d'hypothèses $\mathcal{H}$, nous dénotons $d_{VC}(\mathcal{H})$ sa dimension VC et $G_{\mathcal{H}}$ 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 $L(B, T)$: chacun des 2^m labellings différents est générable par une hypothèse qui se trouve dans $L(B, T)$. 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 $G_{\mathcal{H}}$ d'un espace d'hypothèses $\mathcal{H}$ en utilisant la dimension VC $d_{VC}(\mathcal{H})$:

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

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.

On sait que la dimension VC d'un perceptron (sans biais) dans $\mathbb{R}^T$ est de T. Tout comme AdaBoost renvoie le signe d'une combinaison linéaire d'hypothèses apprises par un weak learner, un perceptron renvoie le signe d'une combinaison linéaire de ses entrées. En utilisant encore le lemme de Sauer, on multiplie la borne précédente par $\left(\frac{em}{T}\right)^T$. Nous avons donc :

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

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 $L(B,T)$, $G_{L(B,T)}(m) = 2^m$. Nous avons donc :

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

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 de ce dernier lemme 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 $m = d_{VC}(L(B,T))$:

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

□

Il ne reste plus qu'à borner l'erreur de généralisation en utilisant cette dimension VC. Il est possible de dériver une inégalité basée sur l'inégalité de Hoeffding qui tire profit de la growth function. Celle-ci est présentée dans [BBLR03] à la page 192 et est reprise à l'équation 4. Le lemme de Sauer permet de la borner par une expression dépendant de la dimension VC d'AdaBoost à l'équation 5.

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

$$\mathbb{P}\left[\sup_{h \in L(B,T)} (R(h) - R_m(h)) \geq \epsilon\right] \leq 4 G_{L(B,T)}(2m) e^{-m\epsilon^2/8} \tag{4}$$

$$\leq 4 \left(\frac{2me}{d_{VC}(L(B,T))}\right)^{d_{VC}(L(B,T))} e^{-m\epsilon^2/8}. \tag{5}$$

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

Conclusion

Le boosting est donc une approche très intéressante dans la recherche de la meilleure hypothèse tout en limitant la complexité et les temps de calcul. En particulier, l'algorithme Adaboost permet d'obtenir des résultats probants dans ce domaine.

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, 1st edition, May 2014.