1 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.

1.1 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.

1.2 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.

1.2.1 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 $\forall i, 1 \geqslant i \geqslant m, f(x_i) = y_i$, soit $T \geqslant 1$ 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 $\forall i, 1 \geqslant i \geqslant m, D_i^{(0)} = \frac{1}{m}$ (distribution équitable). Cette distribution représente l'importance que doit accorder le weak learner à chaque exemple. De fait, plus le poids $D_i^{(t)}$ $(1 \geqslant i \geqslant m$ et $1 \geqslant t \geqslant T)$ 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 $D^{(t+1)}$ (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.

1.2.2 Modification de la distribution D

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

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

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 : $\forall 1 \geqslant t \geqslant T, \sum_{i=1}^{m} D_i^t = 1$.

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