AdaBoost typical Algorithm and its application research

TU Chengsheng , LIU Huacheng , XU Bing
Computer Science and Engineering college，Chongqing Three Gorges University ,ChongQing WanZhou 404000

Abstract. Boosting is popular algorithm in the field of machine learning. Adaboost is the most typical Algorithm in the Boosting family.This paper introduces Boosting and its research status briefly，and introduces the typical algorithms of each series respectively.

KeyWords: Machine learning；Adaboost Boosting algorithm; Text categorization; Combining prediction

1.Introduction

Boosting was suggested by Freund and Schapire in 1990 [1] .It is an efficient instrument for improving the predicting ability of learning system and a most typical method in coordinating learning. For Boosting, there are two main problems: how to adjust the training set to enable the weak classifier to conduct training on it, and how to combine the weak classifiers, gained through training, into a strong one. To solve the above problems, Freund and Schapire introduced Adaboost (adaptive boosting) algorithm in 1995, which works through adjusting weight without the need of any priori knowledge on learner learning. Algorithms of AdaBoost**.**M1 、 AdaBoost**.**M2 、 AdaBoost**.**R, etc. were suggested based on later innovation. These algorithms could change voting-weights and solved many practical problems of the early Boosting algorithm. [3] To solve the multi-class problem when there is a big category, Freund and Schapire introduced AdaBoost**.**OC algorithm [5] in 1997 based on the combination of AdaBoost.M2 与 ECOC[4]. A more popular AdaBoost algorithm was suggested in 1998 by them, and a confidence predication was introduced so as to improve the quality of Boosting. They also proposed the algorithms of AdaBoost**.**MH 、 AdaBoost**.**MR, which could solve multi-class and multi-label problems, and recommended the AdaBoost**.**MO [6] based on the combination of ECOC and AdaBoost.MH.

The study and application of AdaBoost algorithm mainly focus on classification problems. For example, the application of literature [7] ,mainly solves the two-class problem, multi-class single label problem, multi-classification and multi-label problem, class single label problem and regression problem.

Based on the application of AdaBoost in text categorization, this paper presents the basic idea and application of the two-class single label problem, and analyzes the algorithms of AdaBoost.M1and AdaBoost.M2 and their application in solving the multi-class single label problem, and the algorithms of AdaBoost.MR and AdaBoost.MH and their application in solving the multi-classification and multi-label problem.

2. AdaBoost algorithm and the two-class single label problem

Prepare training sets$(x_1, y_1), \ldots, (x_n, y_n)$. $x_i \in X$, X represents a certain domain or instance space, and each member is a training example with a label. $y_i \in Y=\{-1,+1\}$,proposes the two-class problem of learning. The weights of all the training examples are initially set to be 1/m equally. Adaboost conducts T times of iteration through repeatedly calling weak learning algorithm. When the tth$(t=1,2,\ldots,T)$iteration was effected, the weight distributing on the sample $\{x_i, y_i\}$ was recorded as $D_t(i)$, and a set of distribution of weights are preserved in the training set after each iteration. The weight of each instance would increase while the training fails each time, so as to enable the weak learner to forcedly focus on the instance of failure in training. As per D_t distribution, the weak learner finds

appropriate weak hypothesis h_t: $X \rightarrow R$; thus, predicting function sequence is gained. Each predicting function gives weight, and the weight of a learner with a better predicating effect is bigger. In the simplest case, if the scope of each h_t is two-valued {−1，+1}, the task of the learner is to minimize the error. Combining T- weak hypotheses, the final predicting function(hypothesis) H is gained after T times of circulation, with a weighted majority voting method. The learning accuracy rate of a single weak learner is not high enough; however, with the application of Boosting algorithm, the accuracy rate of the final result is to be improved. The AdaBoost algorithm is described in detail, as follows:

⑴With the given N-training sample set with labels: $(x_1,y_1),(x_2,y_2),.....(x_n,y_n)$,among which $x_i \in X$, $y_i \in Y=\{-1,+1\}$.

⑵For t=1,2,……,T，initially set up weight D(i) and make D(i)=1/N.

⑶For t=1,2,……,T， T represents the maximum circulation times of training. For T-training, firstly, the weight distributing on the sample $\{X_i, Y_i\}$ is recorded as $D_t(i)$ while the Tth iteration happens, and as per the distribution D_t the weak learner finds its weak hypothesis $h_t: X \rightarrow \{+1, -1\}$ and adjusts distribution. Secondly, the error rate in computing h_t: $\varepsilon_t = \sum_{i=1}^{N} D_t(x_i)[h_t(x_i) \neq y_i]$. Thirdly, compute the weight of weak classifier based on the error rate: $\alpha_t=(1/2)ln((1-\varepsilon_t)/\varepsilon_t)$. Fourthly, update the sample weight:

$$D_{t+1}(x_i) = \frac{D_t(x_i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i)),$$

among which,

$$Z_t = \sum_{i=1}^{n} D_t(x_i) \exp(-\alpha_t y_i h_t(x_i))$$

Z_t is a standardized factor that meets the probability distribution.

⑷T-weak classifiers are gained after T times of circulation, and a strong classifier $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$ is gained after adding to the updated weight.

3. AdaBoost.M1 and AdaBoost.M2 algorithms solve multi-class single label problems

Each sample to be classified is a single class within multiple category. To define $[[\pi]]$, if π is true, $[[\pi]]$=1, otherwise $[[\pi]]$= 0.

3.1 AdaBoost.M1 algorithm

Input sequence of *N*-examples $(x_1, y_1), \cdots, (x_n, y_n)$, label $y_i \in Y=\{1,2, \cdots, k\}$, and set the distribution on instance is D , the weak learning algorithm WeakLearn, and the iteration T. In initialization, for $i=1,2,...,T$,set the weight vector $W_i^1 = D(i)$. For $t=1,2,\ldots\ldots,T$, execute the following:

⑴the whole distribution $P^t = W^t / \sum_{i=1}^{N} W_i^t$. ⑵call the WeakLearn and transmit the distribution P^t to it. ⑶ calculate the error rate of h_t $\varepsilon_t = \sum_{i=1}^{N} P_i^t [[h_t(x_i) \neq y_i]]$. If $\varepsilon_t > 1/2$, set T=t−1 and exit a loop. ⑷ establish $\beta_t = \varepsilon_t / (1-\varepsilon_t)$. ⑸ Calculate the new weight:

$$W_i^{t+1} = W_i^t \beta_t^{1-[[h_t(x_i) \neq y_i]]}$$

and gain the final hypothesis:

$$h_f(x) = \arg\max_{y \in Y} \sum (\log(1/\beta_t))[[h_t(x) = y]].$$

When AdaBoost.M1calls the WeakLearn, the error rate produced is the hypothesis $\varepsilon_1, \cdots, \varepsilon_T$. Propose each $\varepsilon_t \leq 1/2$, the upper boundary of the error $\varepsilon = \Pr_{i \sim D}[h_t(x_i) \neq y_i]$ of the final hypothesis is $\varepsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)}$.

This is the most direct multiclass extension of AdaBoost. When the WeakLearn is strong enough to gain proper precision in the difficult distribution generated by

AdaBoost, it is enough to solve multiclass problems. Provided that the WeakLearn can not gain a precision of no less than 50%, the method fails. Therefore, multiclass problems are generally solved by simplifying them into multiple "two-value problems". This idea is the preferred choice for algorithms of AdaBoost.M2， AdaBoost.MH， AdaBoost.MR, etc. to solve multiclass problems.

3.2 AdaBoost.M2 algorithm

Input N-example sequence $(x_1, y_1), \ldots, (x_n, y_n)$,label $yi \in Y=\{1,2, \ldots, k\}$, and set D as the distribution on the instance, WeakLearn as the weak learning method, and T as the iterative times. Initially, for $i=1,2,\ldots,T,$ set the weight vector $W_{i,y}^1 = D(i)/(k-1), y \in Y-\{y_i\}$. For $t=1,2,\ldots,T,$ execute: (1) establishing $W_i^t = \sum_{y \neq y_i} W_{i,y}^t$, $q_t(i, y) = W_{i,y}^t / W_i^t$, and for $y \neq y_i$, setting up $D_t(i) = W_i^t / \sum_{i=1}^N W_i^t$; (2) calling WeakLearn, and transmitting distribution D_t and label weight function q_t to it; returning to hypothesis $h_t: X \times Y \rightarrow [0,1]$; (3) computing pseudo loss of h_t :

$$\varepsilon_t = \frac{1}{2}\sum_{i=1}^{N} D_t(i)\left(1 - h_t(x_i, y_i) + \sum_{y \neq y_i} q_t(i,y) h_t(x_i, y)\right);$$

(4)set $\beta_t=\varepsilon_t/(1-\varepsilon_t)$; (5)computing the new weight: $W_{i,y}^{t+1} = W_{i,y}^t \beta_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y))}$, and gain the final hypothesis: $h_f(x) = \arg\max_{y \in Y} \sum (\log(1/\beta_t) h_t(x, y)$.

AdaBoost.M2 is a special case of AdaBoost.MR. For each sample x_i *with correct label* y_i and each incorrect label y (y represents the other k-1 type except y_i), raise such a two-value question as : for x_i, the correct label is y or y_i? Here, the given h is used to answer the k-1 two-value questions, and separate out the correct label y_i from the incorrect label y. Propose h fetches its value from {0,1}, if $h(x_i, y)=0$ and $h(x_i, y_i)=1$, then the answer to the above question is y_i ; if $h(x_i, y)=1$ and $h(x_i, y_i)=0$, the answer to the said question is y. If $h(x_i, y)=h(x_i, y_i)$, then select any one from the two at random. When h fetch its value from [0,1], $h(x, y)$ can be interpreted as a random decision: select a random bit $b(x, y) \in \{0, 1\}$, the probability that the value is equal to 1 is $h(x, y)$, and the probability that the value is equal to 0 is $1-h(x, y)$. Hence, the probability to select incorrect answer y is:

$$\Pr[b(x_i, y_i) = 0 \wedge b(x_i, y) = 1] + \frac{1}{2}\Pr[b(x_i, y_i) = b(x_i, y)]$$

$$= \frac{1}{2}(1 - h(x_i, y_i) + h(x_i, y))$$

If answers of all the k-1 questions are equally important, the assumed loss is defined as average value:

$$(1 - h(x_i, y_i) + (\sum_{y \neq y_i} h(x_i, y)) / (k-1)) / 2$$

. However, various distinctions is very likely to have different importance in diverse situations. The method to attach different importance to diverse questions is to attach a weight to each question. Therefore, to specify weight $q(i,y)$ for every instance x_i and incorrect label $y \neq y_i$, it is linked to the differentiated label y and the correct label y_i . Put it into the above formula, and the result is referred to as the loss of q when h is on the training example i : $ploss_q(h,i)$

$$\approx \frac{1}{2}(1 - h(x_i, y_i) + \sum_{y \neq y_i} q(i, y) h(x_i, y)) .$$

The function $q:\{1,\ldots N\} \times Y \rightarrow [0,1]$ is termed as label weight function. For all the i, $\sum_{y \neq y_i} q(i, y) = 1$. The object of the WeakLearn is minimize the expected pseudo-loss of the distribution D and the weight function q: $ploss_{D,q}(h) := E_{i \sim D}[ploss_q(h,i)]$.

By controlling D and q, this algorithm effectively let the WeakLearn focus not only on difficult instances, but also on incorrect class labels which are most difficult to be eliminated. On the contrary, this kind of pseudo loss measurement may make the WeakLearn easily get weak dominance. AdaBoost.M2 is to produce a hypothesis with a pseudo loss of $\varepsilon_1, \ldots, \varepsilon_T$ when it calls the WeakLearn. Thus, the upper boundary of

$\varepsilon = \Pr_{i\sim D}[h_t(i) \neq y_i]$, the error of the final hypothesis h_f, is $\varepsilon \leq (k-1)2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)}$.

4. AdaBoost.MR and AdaBoost.MH algorithms solve multi-class and multi-label problems

Each object can belong to one or more of multi categories. A single label classification problem is a particular example of multi-label ones. Suppose γ is the finite set of a label or a class, and make $K=|\gamma|$. In the multi-label case, each instance $x_i \in X$ may belong to multi labels in γ; therefore, the example with a label is *(x, Y)* pairs, among which $Y \subseteq \gamma$ is a label set given to x. The purpose of learning is the need to look for a label hypothesis of a label set given to the instance by prediction, i.e., the purpose is to look for the probability $H:X \to \gamma$ of the minimized $H(x) \notin Y$ on a new type (x,Y). This type of measurement is referred to as one-error of hypothesis H since it measures a probability even with an incorrect label. The $one\text{-}error_D(H)$ is used to express the one-error about distribution D made by hypothesis H when observing on (x,Y), namely,

$$one-error_D(H) = \Pr_{(x,Y)\sim D}[H(x) \notin Y].$$

4.1 AdaBoost.MH algorithm based on the hamming loss

Input training sets $(x_1, y_1), \ldots, (x_m, y_m)$, among which $x_i \in X$, $Y_i \subseteq \gamma$. initially, $D_1(i,l)=1/(mk)$. *For t=1,2..., T* loops execution: (1)Distribution D_t trains WeakLearn; (2)obtain weak hypothesis h_t: $X \times \gamma \to R$; (3)choose $\alpha_t \in R$; (4)revise:

$$D_{t+1}(i,l) = (D_t(i,l)\exp(-\alpha_t Y_i[l] h_t(x_i,l))) / Z_t,$$

among which Z_t is a normalization factor, and then gain the final hypothesis:

$$H(x,l) = sign(\sum_{t=1}^{T} \alpha_t h_t(x,l))).$$

AdaBoost.MH operates by producing a set of two value problems to each sample x and each sample *y,* as follows : for sample *x*, its correct label is *y* or something else? Suppose the present target is only to predict all the correct labels, then learning algorithms generate the hypothesis of predictive label set, and the assumed loss depends on how large the difference b/w predictive set and observed result is. Consequently, the loss is $E_{(x,Y)\sim D}[h(x) \Delta Y] / k$, among which Δ represents symmetric difference, and is referred to as the hamming loss of H and expressed as $hloss_D(H)$. In order to minimize it, the problem is resolved into k-Orthogonal binary classification ones, namely, *Y* can be considered as the special value of k-binary labels (relying on whether a certain label y is included in the *Y*). Similarly, $h(x)$ can be considered as k-binary prediction. Thus, the loss can be regarded as an average error rate of h in k-binary problems. For $Y_i \subseteq \gamma, l \in \gamma, Y[l]$ is defined as $Y[l] = \begin{cases} +1 & if \quad l \in Y \\ -1 & if \quad l \notin Y \end{cases}$. To simplify the expression, the two- variables function of $H: X \times \gamma \to \{-1, +1\}$, defined with $H(x,l)=H(x)[l]$, is used to express the corresponding function of $H:X \to 2^{\gamma}$. This shows Boosting minimizes the hamming loss, and the main idea of the minimization is to replace each training example of (x_i, Y_i) with k-examples of $((x_i,l), Y_i[l])$ (for $l \in \gamma$). This minimization also lead to the select of the final hypothesis. It proves that the upper bound of the hamming loss is $hloss(H) \leq \prod_{t=1}^{T} Z_t$.

4.2 AdaBoost.MR based on ranking loss

Input training sets $(x_1, y_1), \ldots, (x_m, y_m)$, among which $x_i \in X$, $Y_i \subseteq \gamma$. Initially:

$$D_1(i, l_0, l_1) = \begin{cases} 1/(m \cdot |Y_i| \cdot |\gamma - Y_i|) & if \quad l_0 \notin Y_i \quad and \quad l_1 \in Y_i \\ 0 & others \end{cases}.$$

T loops execution: (1) use distribution D_t to train the WeakLearn; (2)gain weak hypothesis h_t: $X \times \gamma \to R$; (3) choose $\alpha_t \in R$; (4)revise:

$$D_{t+1}(i, l_0, l_1) = (D_t(i, l_0, l_1)\exp(\frac{1}{2}\alpha_t(ht(x_i, l_0) - ht(x_i, l_1)))) / Z_t,$$

among which Z_t is a normalization factor (make D_{t+1}as the distribution), and gain the final hypothesis: $f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$. The purpose of AdaBoost.MH is to

find the hypothesis of the label set connecting with each instance, while the purpose of AdaBoost.MR is to find a hypothesis which sorts the labels, in the hope that the right labels can reach the highest level. To formally express, find a hypothesis shaped like $f: X\times\gamma\to R$, and interpret it as: to sort the labels in a given instance x, γ as per $f(x)$, that is to say, if $f(x, l_1)>f(x, l_2)$, the level of a label l_1 is considered to be more higher than that of l_2. For an observation (x, Y), care only about the relative sort relation of key pair of l_0, l_1 ($l_0\notin Y, l_1\in Y$). If $f(x, l_1)\leq f(x, l_0)$, then the key pair of l_0, l_1 can be considered as being wrongly sequenced by f. The target here is to look for a function f with only a small amount of error sorting, and let the labels in Y be sorted before those outside of Y. Therefore, the anticipated key pair with error sorting need to be minimized. Such a measure is called a ranking loss. A distribution D on a observation is defined as

$$E_{(x,Y)\sim D}\left[\frac{\left|\left\{\left(l_0, l_1\right)\in\left(\gamma-Y\right)\times Y : f\left(x, l_1\right)\leq f\left(x, l_0\right)\right\}\right|}{\left|Y\right|\left|\gamma-Y\right|}\right],$$

and expressed by $rloss_D(f)$. It would not fail for any observation Y, and would not equal to γ; or otherwise, it needs no sorting. This algorithm maintains the distribution of D_t *on* $\{1,2, \ldots,m\}\times Y\times Y$. Only when l0，l1 are key pairs in relation to (x_i, Y_i), this distribution is non-zero in the triples（i, l_0, l_1）.

4.3 Improved algorithm of AdaBoost.MR

Input training sets (x_1, y_1), ···, (x_m, y_m), among which $x_i\in X$, $Y_i\subseteq\gamma$。Initially：

$$v_1\left(i, l\right)=\left(m\cdot\left|Y_i\right|\cdot\left|\gamma-Y_i\right|\right)^{-\frac{1}{2}}$$

For $t=1,2,\ldots, T$ loops，and execute：(1)training WeakLearn with the distribution D_t；(2)gaining a weak hypothesis $h_t: X\times\gamma\to R$；(3)selecting $\alpha_t\in R$；(4) revising:

$$v_{t+1}\left(i, l\right)=\left(v_t(i, l)\exp\left(-\frac{1}{2}\alpha_t Y_i[l]h_t(x_i, l)\right)\right)/\sqrt{Z_t},$$

among which：

$$Z_t=\sum_i\left[\left(\sum_{l\in Y_i}v_t(i,l)\exp\left(\frac{1}{2}\alpha_t h_t(x_i,l)\right)\right)\left(\sum_{l\in Y_i}v_t(i,l)\exp\left(-\frac{1}{2}\alpha_t h_t(x_i,l)\right)\right)\right],$$

and gaining the final hypothesis $f(x,l)=\sum_{t=1}^{T}\alpha_t h_t(x,l)$.

When the original AdaBoost.MR method has a lot of labels, the efficiency both in time and space is not high enough. $|Y_i|\cdot|\gamma- Y_i|$-weights need to be maintained for each training example(x_i, Y_i) and each weight needs to be revised each time. In reality, for the same algorithm, $O(mk^2)$ may be the difference between space complexity and time complexity that each iteration needs , *and only weight* v_t needs to be maintained in $\{1,2,\ldots,m\}\times\gamma$. Provided that l_0, l_1 is the key pair in relation to (x_i, Y_i), it is true that $D_t(i,l_0,l_1)=v_t(i,l_0).v_t(i,l_1)$. Thus, it can be seen that this algorithm is equal to the original AdaBoost.AR algorithm and the overhead is only $O(mk)$.

5. Conclusion

AdaBoost has the advantages of being quick in speed, simple in operation and easy to be programmed. There is no need to adjust parameters except for the number of iterations. It can be flexibly combined with any method to look for weak hypothesis without the need of any priori knowledge of WeakLearn. Given sufficient data and a WeakLearn with only reliable moderate accuracy, it can provide a set of theoretical guarantee of learning. It focuses on look for a weak learning method only better than random predication, instead of trying to design an accurate algorithm in the whole space. However, AdaBoost still has its own disadvantages. For example, its real performance in a particular problem clearly relies on data and WeakLearn; in the event of insufficient data set, it has a poor performance in too complex and too weak hypotheses; and it seems to be very sensitive to noise.

Acknowledgements

It is a project supported by the scientific fund assistance project from Chongqing Education Committee (NO: KJ131115).

References

[1] Robert E. Schapire. ***The strength of weak learnability***. Machine Learning, 5(2):197-227, 1990

[2] Freund Y.，Schapire R.E.A.，***Decision-Theoretic***

Generalization of OnLine Learning and an Application to Boosting.Journal of Computer and System Sciences,1997,55(1):119-139.

[3] Yoav Freund and Robert E. Schapire. ***A decision-theoretic generalization of online learning and an application to boosting***. Journal of Computer and System Science, 55(1):119-139, August 1997

[4] Thomas G.Dietterich and Ghulum Bakiri. ***Solving multiclass learning problems via error-correcting output codes***. Journal of Artificial Intelligence Research, 2:263-286, January 1995

[5] Robert E. Schapire and Yoram Singer. ***Using output codes to boost multiclass learning problems***. In Machine Learning: Proceedings of the Fourteenth International Conference, pp.313-321. 1997

[6] Robert E. Schapire and Yoram Singer. ***Improved boosting algorithms using confidence-related predictions***. In Proceedings of the eleventh Annual Conference on Computational Learning Theory, pp.80-91, 1998

[7] Diao Li-li,Hu ke–yun,Lu Yu-chang, ***Improved Stumps Combined by boosting for Text Categorization***.Journal of Software.2202,13(8).

[8] Yoav Freund. ***An adaptive version of the boost by majority algorithm***. In Proceedings of the Twelfth Annual Conference on Computational Learning Theory, 1999