COST-SENSITIVE BOOSTING ALGORITHMS AS GRADIENT DESCENT

Qu-Tang Cai, Yang-Qui Song, Chang-Shui Zhang

State Key Laboratory on Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology (TNList),
Department of Automation, Tsinghua University, Beijing, China.

ABSTRACT

AdaBoost is a well known boosting method for generating strong ensemble of weak base learners. The procedure of AdaBoost can be fitted in a gradient descent optimization framework, which is important for analyzing and devising its procedure. Cost sensitive boosting (CSB) is an emerging subject extending the boosting methods for cost sensitive classification applications. Most CSB methods are performed by directly modifying the original AdaBoost procedure. Unfortunately, the effectiveness of most cost sensitive boosting methods are checked only by experiments. It remains unclear whether these methods can be viewed as gradient descent procedures like AdaBoost. In this paper, we show that several typical CSB methods can also be view as gradient descent for minimizing a unified objective function. We then deduce a general greedy boosting procedure. Experimental results also validate the effectiveness of the proposed procedure.

Index Terms— Boosting, Cost-sensitive Classification, Gradient Descent, Optimization

1. INTRODUCTION

Boosting algorithms are currently among the most popular and most successful algorithms for pattern recognition tasks. AdaBoost [1] is a practically successful boosting algorithm, which can also be viewed as the gradient descent procedure of a certain surrogate function [2, 3]. The gradient descent view is essential for both devising new algorithms and studying the algorithm's properties such as convergency and consistency [4, 5]. Due to the practical success of AdaBoost, it is interesting to extend the procedure to various tasks, one of which is cost sensitive learning [6].

In general, classification algorithms are designed to minimize the misclassification error. However, there are many problems which are naturally cost sensitive, and methods for minimizing the misclassification error tend to be unsatisfactory. For example, the cost of misdiagnosis of classifying healthy people as sick and that of classifying sick people as healthy are apparently not equal, since the latter may lead to serious results. Cost-sensitive learning is a suitable way for solving such problems, where classifiers are designed to be optimal for weighted loss. The weights can emphasize the more important errors. To extent boosting for cost sensitive learning scenarios, several cost sensitive boosting (CSB) methods have been proposed, such as AdaCost [7], AdaC{1,2,3} [8], and asymmetric boosting [9].

Most of the CSB methods originated from heuristically modifying the weights and confidence parameters of AdaBoost, and their effectiveness is checked only by experiments. It remains unclear in theory whether or why these manipulations work as expected. One important problem is whether CSB algorithms can be viewed as gradient descent procedure like AdaBoost. If they can, the previous results of AdaBoost concerning gradient descent, such as convergency and consistency, may be applicable parallelly. Unfortunately, there are a vast increasing amount of CSB methods till now, and it is not possible to cover them all in this paper. However, since the motivation of CSB methods is to extend AdaBoost to cost sensitive learning, we only consider several typical CSB methods which can include AdaBoost as a special case. In later sections, we will show that the CSB algorithms can be fitted into a unified gradient descent framework for a common surrogate function.

1.1. Basic Settings and Notations

Like AdaBoost and the CSB algorithms, we will only consider binary classification problems. We use X as the feature space, and $Y = \{+1, -1\}$ as the set of labels. Each example is represented as a feature-label pair, (x, y), where $x \in X$ and $y \in Y$. The set of weak base classifiers in the boosting procedure is denoted by $\mathcal{H}$, whose linear span is denoted by $\mathcal{F}$. Each weak learner in $\mathcal{H}$ outputs the binary labels in Y. We denote $I(\cdot)$, $sign(\cdot)$, and $E(\cdot)$, the indicator function, the sign function and the expectation, respectively.

2. ADABOOST AND COST SENSITIVE BOOSTING

We revisit AdaBoost and the considered typical CSB methods, including AdaCost [7], AdaC{1,2,3} [8], and asymmetric boosting(AsymBoost) [9], which have AdaBoost as a special case.

Supported by National 863 project(No. 2006AA10Z210).

Table 1. Algorithmic Procedure of AdaBoost

Input: $(x_1, y_1), \ldots, (x_n, y_n)$; where $x_i \in X$, $y_i \in \{-1, +1\}$, and $i = 1, \ldots, n$.

Initialization: Set weights $w_i^{(0)} = \frac{1}{n}$ on training data.

Repeat for $t = 1, 2, \ldots, T$:

(a) Train weak learner f_t using weights $w_i^{(t-1)}$ on the training data,

$$f_t = \arg\max_{f \in \mathcal{H}} \sum_{i=1}^{n} y_i w_i^{(t-1)} h(x_i). \quad (1)$$

(b) Compute the (nonnegative) weight α_t of f_t:

$$\alpha_t = \frac{1}{2} \ln \frac{1 - err_t}{err_t}, \quad (2)$$

where $err_t = \sum_{i=1}^{n} D_i^{(t-1)} I(f_t(x_i) \neq y_i)$ and $D_i^{(t-1)} \doteq \frac{w_i^{(t-1)}}{\sum_{i=1}^{n} w_i^{(t-1)}}$.

(c) Reweight: update weights of training data

$$w_i^{(t)} = w_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i)}. \quad (3)$$

Output: output the final classifier $sign(\sum_{t=1}^{T} \alpha_t f_t(x))$.

2.1. AdaBoost

The AdaBoost procedure is described in Table 1. It employs an iterative procedure for ensemble learning, which produces a linear combination of weak hypotheses. In each stage of the boosting procedure, AdaBoost produces a probability distribution on the examples, and then obtain a weak hypothesis whose misclassification error is better than random guess. The weak hypothesis is then used to update the distribution, and the hard examples receive high probability. At the end of each iteration, the weak hypothesis is added to the linear combination to form the current hypothesis of the algorithm.

2.2. Cost Sensitive Boosting Methods

For misclassifying each x_i, cost sensitive boosting methods introduce a prescribed cost c_i, and incorporate it into the boosting procedure.

AdaCost: AdaCost [7] incorporates a cost adjustment function β in the computation of err and in the reweight step. AdaCost assigns a cost $c_i \in [-1, +1]$ to the misclassification of x_i, and pre-weights x_i with weight $\frac{c_i}{\sum_{i=1}^{n} c_i}$. In the reweight stage, $w_i^{(t)}$ is updated to

$$w_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i) \{0.5 + 0.5 c_i [I(y_i \neq f_t(x_i)) - I(y_i = f_t(x_i))]\}}. \quad (4)$$

In (4), if x_i is misclassified by f_t, its weight is increased, otherwise it is decreased. Let $\beta_i^{(t)} \doteq 0.5 + 0.5 c_i [I(y_i \neq f_t(x_i)) - I(y_i = f_t(x_i))]$. The parameter α_t in (4) is determined for minimizing

$$\sum_{i=1}^{n} D_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i) \beta_i^{(t)}}, \quad (5)$$

which can be done by line search. When all c_i's are identical and approaches 0, AdaCost can then be reduced to AdaBoost.

AdaC{1, 2, 3}: AdaC1, AdaC2 and AdaC3 [8] assign cost c_i for misclassifying x_i. They alter the weight update rule, and computing method for α_t. Due to their similarity, we only consider AdaC1. Define c_i as the misclassification cost of the i-th example. The weight update rule for AdaC1 is

$$w_i^{(t)} = w_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i) c_i}, \quad (6)$$

and α_t is calculated for minimizing

$$\sum_{i=1}^{n} D_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i) c_i}, \quad (7)$$

which is approximate calculated as

$$\alpha_t = \frac{1}{2} \ln \frac{1 + \sum_{i: y_i = f_t(x_i)} c_i D_i^{(t-1)} - \sum_{i: y_i \neq f_t(x_i)} c_i D_i^{(t-1)}}{1 - \sum_{i: y_i = f_t(x_i)} c_i D_i^{(t-1)} + \sum_{i: y_i \neq f_t(x_i)} c_i D_i^{(t-1)}}.$$

AsymBoost: Asymmetric boosting (AsymBoost) is based on the statistical interpretation of boosting, which consider the asymmetric misclassification cost of different classes. It attempts to minimize

$$\sum_{i=1}^{n} I(y_i = 1) e^{-c_+ y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)} + I(y_i = -1) e^{-c_- y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)}, \quad (8)$$

where c_+, c_- are the misclassification costs for positive and negative examples, respectively. When $c_+ = c_- = 1$, asymmetric boosting is identical to LogitBoost [10], a generalization of AdaBoost.

3. GRADIENT DESCENT COST SENSITIVE BOOSTING

3.1. AdaBoost as Gradient Descent

AdaBoost can be viewed as an optimization procedure [2] for minimizing

$$\min_{F \in \mathcal{F}} \hat{E}(e^{-yF(x)}) = \min_{f_t \in \mathcal{H}, \alpha_t \in \Re} \sum_{i=1}^{n} \frac{1}{n} e^{-y_i \sum_{t=1}^{T} \alpha_t f_t(x_i)}. \quad (9)$$

Let $J(F) \doteq \hat{E}(e^{-yF(x)})$. AdaBoost performs a forward gradient descent procedure [2] to seek F for minimizing $J(F)$: in the k-th stage of AdaBoost, AdaBoost attempts to minimize $J(F_{k-1} + \alpha_k f_k)$ w.r.t. α_k and f_k where $F_{k-1} = \sum_{t=1}^{k-1} \alpha_t f_t$. For seeking α_k and f_k, AdaBoost incorporates an alternative optimization technique:

Step 1: Obtain the maximal descent direction f_k at F_{k-1},

$$f_k = \arg\max_{f\in\mathcal{H}} -\frac{\partial J(F_{k-1}+\alpha f)}{\partial \alpha}|_{\alpha=0} = \arg\max_{f\in\mathcal{H}} \sum_{i=1}^{n} y_i w_i^{(k-1)} f(x_i). \quad (10)$$

Since $y_i f(x_i) \in \{+1, -1\}$, a rearrange of (10) leads to

$$f_k = \arg\min_{f\in\mathcal{H}} D_i^{(k-1)} I(f(x_i) \neq y_i), \quad (11)$$

which indicates that f_k can be obtained by minimizing the training error under weights $w_i^{(k-1)}$.

Step 2: Seek the optimal α_k along the descent direction f_k:

$$\alpha_k = \arg\max_{\alpha\in\Re} J(F_{k-1}+\alpha f_k). \quad (12)$$

Note that $J(F_{k-1}+\alpha f_k)$ is convex with respect to α, so $J(F_{k-1}+\alpha f_k)$ can be globally minimized when $\frac{dJ(F_{k-1}+\alpha f_k)}{d\alpha} = 0$, which is given by (2).

In each stage of AdaBoost, the main role of training f_t is to seek some descent direction in function space. The optimality of f_t is not a crucial requirement. Actually, in some scenarios, f_t is hard to globally minimize the training error under current weights. For example, it is difficult to train a pruned tree classifier which minimizes the training error. Once f_t is chosen, the descent of the surrogate function is determined by the step size α_t.

3.2. General Objective Function for CSB algorithms

To fit the CSB algorithms into a optimization procedure, it is essential to identify their surrogate functions.

AdaCost: In the t-th stage, after f_t is obtained, α_t is determined to minimize (5). Therefore, f_t can be viewed to serve as a descent direction, and by (4-5), up to a scale of the weight normalizer, AdaCost decreases the following objective function in the t-th stage,

$$\sum_{i=1}^{n} w_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i)\beta_i^{(t)}} = \sum_{i=1}^{n} w_i^{(0)} \prod_{k=1}^{t} e^{-\alpha_t y_i f_t(x_i)\beta_i^{(t)}}$$
$$= \sum_{i=1}^{n} w_i^{(0)} \prod_{k=1}^{t} e^{-\alpha_t y_i f_k(x_i)[0.5-0.5c_i y_i f_k(x_i)]}$$
$$= \sum_{i=1}^{n} w_i^{(0)} e^{-\sum_{k=1}^{t}\alpha_t\cdot[0.5 y_i f_k(x_i)-0.5c_i]}. \quad (13)$$

AdaC1: Like AdaCost, it decreases the following objective function in the t-th stage,

$$\sum_{i=1}^{n} w_i^{(t-1)} e^{-\alpha_t y_i f_t(x_i) c_i} = \sum_{i=1}^{n} w_i^{(0)} e^{-\sum_{k=1}^{t}\alpha_t y_i f_k(x_i) c_i}. \quad (14)$$

AsymBoost: Unlike other CSB methods which heuristically modifying AdaBoost, AsymBoost is directly designed to minimize (15). For the t-th stage, it decreases

$$\sum_{i=1}^{n} I(y_i=1) e^{-c_+ y_i \sum_{k=1}^{t}\alpha_k f_k(x_i)} + I(y_i=-1) e^{-c_- y_i \sum_{k=1}^{t}\alpha_k f_k(x_i)}$$
$$= \sum_{i=1}^{n} e^{-c_+ I(y_i=1) y_i \sum_{k=1}^{t}\alpha_k f_k(x_i) - c_- I(y_i=-1) y_i \sum_{k=1}^{t}\alpha_k f_k(x_i)}. \quad (15)$$

Since $y_i \in \{\pm 1\}$, $c_+ I(y_i = 1)$ and $c_- I(y_i = -1)$ can be unified with $\frac{c_+ + c_-}{2} + y_i \frac{c_+ - c_-}{2}$. Therefore, we can reformulate (15) into

$$\sum_{i=1}^{n} e^{-\sum_{k=1}^{t}\alpha_k y_i f_k(x_i)[\frac{c_+ + c_-}{2} + y_i \frac{c_+ - c_-}{2}]}. \quad (16)$$

General Objective Function: A closer look at (13),(14) and (16) share a common formulation

$$\sum_{i=1}^{n} w_i^{(0)} e^{-\sum_{k=1}^{t}\alpha_k [a_i f_k(x_i) + b_i]}, \quad (17)$$

where a_i, b_i are related to the cost parameters and label information. For example, for AdaCost, $a_i = 0.5 y_i, b_i = -0.5 c_i$.

3.3. Gradient Descent Cost Sensitive Boosting

The expression of objective function (17) is like AdaBoost. Actually, it can also include AdaBoost as a special case when $a_i = y_i, b_i = 0$. Therefore, it is natural to employ the gradient descent procedure of AdaBoost for minimizing (17). In the t-th stage, since f_k, α_k, $k = 1, \cdots, t-1$ have been found in previous stages, we can write the objective function (17) by $G^{(t-1)}(f_t, \alpha_t)$, a function of f_t and α_t. We develop the following procedure for obtaining f_t and α_t, like (10) and (12),

Step 1: Obtain the descent direction f_t,

$$f_t = \arg\max_{f\in\mathcal{H}} -\frac{\partial G^{(t-1)}(f,\alpha)}{\partial \alpha}|_{\alpha=0} = \arg\max_{f\in\mathcal{H}} \sum_{i=1}^{n} w_i^{(0)} e^{-\sum_{k=1}^{t-1}\alpha_k [a_i f_k(x_i) + b_i]} a_i f(x_i). \quad (18)$$

For solving (18), assign each x_i the pseudo-label $\tilde{y}_i = sign(a_i)$ and let $w_i^{(t-1)} = |w_i^{(0)} e^{-\sum_{k=1}^{t-1}\alpha_k [a_i f_k(x_i) + b_i]} a_i|$. Then f_t can be solving via minimizing the training error under weights $w^{(t-1)}$ and labels $\tilde{y}_i$'s, like (10).

Step 2: Seek the optimal α_t:

$$\alpha_t = \arg\max_{\alpha\in\Re} G(f_t, \alpha). \quad (19)$$

Note that $G(f_t, \alpha)$ is convex with respect to α, so global optimal solution can at least be effectively calculated by line-search methods such as bisection.

The major steps of the gradient descent process are presented in Table 2.

Table 2. Main Procedure of Gradient Descent CSB

Repeat for $t = 1, 2, \ldots, T$:

(a) Train weak learner f_t using weights $w_i^{(t-1)}$ on the training data (with pseudo-labels), minimizing (18).

(b) Compute α_t by minimizing (19).

(c) Reweight: $w_i^{(t)} = w_i^{(t-1)} e^{-\alpha_t [a_i f_t(x_i) + b_i]}$.

4. EXPERIMENTS

To verify the effectiveness of our proposed gradient procedure, we use four two-class medical diagnosis data sets taken from the UCI Machine Learning Database [11] for experiments. These datasets are suitable for cost sensitive learning due to their class imbalance. These four data sets are: Breast cancer data (Cancer), Hepatitis data (Hepatitis), Pima Indians diabetes database (Pima), and Sick-euthyroid data (Sick). The disease category is treated as the positive class, and the normal category is treated as the negative class. Since the objective functions of different CSB methods are diverse, we only compare our algorithm with one of them, AdaCost, whose objective function is (13).

Each dataset is randomly divided into two disjointed parts: 90% for training and the remaining 10% for testing. This process is repeated 20 times to obtain a stable average result. C4.5 decision tree is used as base weak learner and the iteration numbers (T) are set to 20. We use F-measure [12], the weighted harmonic mean of precision and recall, for evaluating the performance. The misclassification costs for samples in the same category are set with the same value. We fix the cost of the positive class to 1 and change the cost item of the negative class from 0.1 to 0.9. The best (highest) F-measure of the cost settings are used for comparison. Experimental results are given in Table 3. The F-measures for AdaCost and the gradient procedure are close, which indicates that the proposed procedure is suitable for cost sensitive boosting, and more important, the procedure is able to achieve comparable results with other CSB methods with the same objective functions.

Dataset	C4.5	AdaBoost	AdaCost	Gradient
Cancer	38.59	41.39	50.86	53.68
Hepatitis	48.81	57.44	65.81	64.28
Pima	60.65	61.32	66.57	69.84
Sick	87.23	86.17	87.33	86.46

Table 3. F-measure evaluation on the experimental results.

5. CONCLUSIONS

We have studied the procedure of cost sensitive boosting methods, and found a general objective function for cost sensitive boosting. We then propose a unified gradient descent framework for optimizing this objective function. Experimental results show that the proposed method can also be used for cost sensitive learning tasks, and can serve as an alternative for other CSB methods with the common objective functions. Like the gradient descent view for AdaBoost, the proposed procedure is promising for developing new algorithms and analyzing their properties.

6. REFERENCES

[1] R. Schapire, "A brief introduction to boosting," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[2] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems*, 2000, vol. 12, pp. 512–518.

[3] J. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[4] P. Bickel, Y. Ritov, and A. Zakai, "Some theory for generalized boosting algorithms," *The Journal of Machine Learning Research*, vol. 7, pp. 705–732, 2006.

[5] P. L. Bartlett and M. Traskin, "Adaboost is consistent," in *Advances in Neural Information Processing Systems*, 2007, vol. 19, pp. 105–112.

[6] C. Elkan, "The foundations of cost-sensitive learning," *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[7] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "AdaCost: misclassification cost-sensitive boosting," in *Proceedings of the Sixteenth International Conference on Machine Learning*, 1999.

[8] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boostingnext term for previous termclassificationnext term of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, 2007.

[9] H. Masnadi-Shirazi and N. Vasconcelos, "Asymmetric boosting," in *Proceedings of the 24-th International Conference on Machine Learning*, 2007.

[10] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 38, no. 2, pp. 337–374, 2000.

[11] C. Blake and C. Merz, "UCI repository of machine learning databases," 1998.

[12] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley, MA, USA, 2005.