Published in final edited form as:

Proc IEEE Int Symp Biomed Imaging. 2011 March 30; 2011: 1831–1834. doi:10.1109/ISBI.2011.5872763.

RBOOST: RIEMANNIAN DISTANCE BASED REGULARIZED BOOSTING

Meizhu Liu and Baba C. Vemuri

Department of CISE, University of Florida, Gainesville, FL 32611

Abstract

Boosting is a versatile machine learning technique that has numerous applications including but not limited to image processing, computer vision, data mining etc. It is based on the premise that the classification performance of a set of weak learners can be boosted by some weighted combination of them. There have been a number of boosting methods proposed in the literature, such as the AdaBoost, LPBoost, SoftBoost and their variations. However, the learning update strategies used in these methods usually lead to overfitting and instabilities in the classification accuracy. Improved boosting methods via regularization can overcome such difficulties. In this paper, we propose a Riemannian distance regularized LPBoost, dubbed RBoost. RBoost uses Riemannian distance between two square-root densities (in closed form) – used to represent the distribution over the training data and the classification error respectively – to regularize the error distribution in an iterative update formula. Since this distance is in closed form, RBoost requires much less computational cost compared to other regularized Boosting algorithms. We present several experimental results depicting the performance of our algorithm in comparison to recently published methods, LP-Boost and CAVIAR, on a variety of datasets including the publicly available OASIS database, a home grown Epilepsy database and the well known UCI repository. Results depict that the RBoost algorithm performs better than the competing methods in terms of accuracy and efficiency.

Index Terms

Square-root density; Riemannian distance; LPBoost; soft margin; hard margin

1. INTRODUCTION

The goal of Boosting is to produce a strong learner through a combination of weak learners. The weak learners can classify quite well on a subset of the dataset, but usually can not classify well on the whole dataset. However, the strong learners boost the performance and can do much better on the whole dataset. The basic idea behind the boosting algorithms is that it will iteratively combine a set of weak learners to produce a strong learner in accordance with a prescribed optimality criterion. The process is summarized in the following paragraph.

Given the training dataset $\{(x_n, y_n)\}_{n=1}^N$, $x_n \in X$ where X is the feature vector set derived from the raw data (images etc.); $y_n \in Y$ and $Y = \{1, -1\}$ is the label set. Let $\mathbf{d}^0 = \{d_1^0, \dots, d_N^0\}$ denote the distribution over $\{x_n\}_{n=1}^N$ and let $\mathcal{H}$ be the set of all weak learners. The task of boosting is to learn a strong learner $H: X \mapsto Y$, and $H(x) = \text{sign}(\sum_{t=1}^T w_t h_t(x))$, where $h_t \in \mathcal{H}$

is the t^{th} weak learner, w_t is the associated weight, and T is the number of weak learners. Let H be the maximizer of the classification accuracy, i.e. $H = \arg \max_{\tilde{H} \in \mathcal{H}} \sum_{n=1}^N d_n^0 \tilde{H}(x_n) y_n$. At the t^{th} iteration, a weak learner h_t is obtained from the training dataset with respect to distribution $\mathbf{d}^{t-1}$. The weighted weak learner h_t is then added to the previously learned strong learner to produce a final strong learner H using the weight w_t that is related with h_t 's accuracy. The accuracy of h_t is measured using $\rho^t = \sum_{n=1}^N d_n^{t-1} h_t(x_n) y_n$. To avoid notation clutter, we introduce a new vector, called the *edge* vector, $\mathbf{u}^t = (\mu_1^t, \dots, \mu_N^t)$ and $\mu_n^t = h_t(x_n) y_n$. Then the accuracy of h_t can be rewritten as $\rho^t = \sum_{n=1}^N d_n^{t-1} \mu_n^t = \mathbf{d}^{t-1} \cdot \mathbf{u}^t$. After this update of the strong learner, the distribution will be updated from $\mathbf{d}^{t-1}$ to $\mathbf{d}^t$ by the rule: samples that were misclassified will gain weight and samples that were classified correctly will lose weight. Thus the new weighted weak learners will focus more on the samples previously misclassified.

The boosting method has been proven to be effective in various classification problems and this served as a motivation for volumes of research reported in literature. Schapire and Singer [1] proposed AdaBoost minimizing the exponential hinge loss, which was followed by the LPBoost [2] that maximizes the minimum margin between classes, this was followed by the TotalBoost, SoftBoost, LogitBoost and GentleBoost [3], and more recently, the Entropy Regularized LP-Boost (ELPBoost) [4]. ELPBoost outperforms others because it has a regularization term which makes the evolution of the distribution smooth and avoids overfitting [4]. However, ELP-Boost uses Kullback-Leibler (KL) divergence as the regularization. KL between two discrete probability distributions $\mathbf{d}$ and $\tilde{\mathbf{d}}$ is $KL(\mathbf{d}, \tilde{\mathbf{d}}) = \sum_i d_i \log(d_i/\tilde{d}_i)$. Unfortunately, the KL divergence suffers from a number of drawbacks. First it is not symmetric and it does not satisfy the triangle inequality and hence is not a metric. Therefore, many operations are computationally expensive and at times impossible. Moreover, in the case that $d_i \neq \tilde{d}_i = 0$ for some i , the KL will become ∞ . Therefore, an over-estimation of the difference may occur. To avoid these problems, we propose the Riemannian distance between square-root densities as the regularization. Let $\hat{\mathbf{d}}$ be the square root of $\mathbf{d}$, $\hat{\tilde{\mathbf{d}}}$ be the square root of $\tilde{\mathbf{d}}$ respectively, i.e., $\hat{\mathbf{d}} = \sqrt{\mathbf{d}}$ and $\hat{\tilde{\mathbf{d}}} = \sqrt{\tilde{\mathbf{d}}}$. Both $\hat{\mathbf{d}}$ and $\hat{\tilde{\mathbf{d}}}$ lie on unit sphere, thus the natural distance between them is their spherical distance, which is also the Riemannian distance. We denote the Riemannian distance between $\hat{\mathbf{d}}$ and $\hat{\tilde{\mathbf{d}}}$ by $\delta(\hat{\mathbf{d}}, \hat{\tilde{\mathbf{d}}})$, and it is given by

$$\delta(\hat{\mathbf{d}}, \hat{\tilde{\mathbf{d}}}) = \arccos(\hat{\mathbf{d}} \cdot \hat{\tilde{\mathbf{d}}}). \quad (1)$$

In this work, we will present the Riemannian distance regularized LPBoost for binary classification, but it can be quite easily generalized to multiclass classification. Furthermore, this boosting method is efficient and is robust, which makes it a promising candidate for online Boosting.

The rest of this paper is organized as follows. In Section 2, we present the proposed Riemannian distance regularized LPBoost. This is followed by an experimental validation of our algorithm on various medical imaging datasets in section 3. Finally, we conclude the paper in section 4.

¹Square root of a vector is the elementwise square root.

2. RIEMANNIAN DISTANCE REGULARIZED LPBOOST

In the Riemannian distance regularized LPBoost (RBoost) we use the Riemannian distance to regularize the conventional LPBoost. The reason for regularization is to make the boosting converge smoothly and quickly, increase its robustness to noise in the data, and avoid overfitting. If the samples are linearly separable, then we need to maximize the hard margin, which is the size of the margin between the positive and the negative samples. By maximizing the hard margin, the classification accuracy of the strong learner on the testing dataset will be optimized. Using LPBoost to maximize the hard margin is described by

$$\max_{\mathbf{w}, \rho} \rho, \text{ s.t. } \sum_{i=1}^t u_n^i \omega_i \geq \rho, n=1, \dots, N, \mathbf{w} \in \Delta_t. \quad (2)$$

Where, ρ is the minimum margin for all samples. Δ_t is the t -simplex and $\mathbf{w} \in \Delta_t$ implies

$$\sum_{i=1}^t \omega_i = 1, \omega_i \geq 0, i = 1, \dots, t.$$

When the dataset is not linearly separable, we need to maximize the soft margin where one allows some samples to fall below the margin but levies a penalty. Using LPBoost to maximize the soft margin is achieved by,

$$\begin{aligned} & \max_{\mathbf{w}, \rho, \zeta} \rho - \kappa \sum_{n=1}^N \zeta_n \\ & \text{s.t. } \sum_{i=1}^t u_n^i \omega_i \geq \rho - \zeta_n, n=1, \dots, N, \\ & \mathbf{w} \in \Delta_t, \zeta \geq 0. \end{aligned} \quad (3)$$

Where, $\zeta = \{\zeta_n\}_{n=1}^N$ is the vector of slack variables, κ is the constant factor which penalizes the slack variables, if κ is very large, say ∞ , then (3) becomes maximization of the minimum hard margin as in problem (2); if κ is small enough, then it will always lead to feasible solutions for (3).

The dual problem of (3) at iteration t minimizes the maximum accuracy resulting from the weak learners learned thus far, and can be described as

$$\min_{\mathbf{d} \in \Delta_N, \mathbf{d} \leq \kappa \mathbf{1}} \max_{i=1, \dots, t} \mathbf{u}^i \cdot \mathbf{d}. \quad (4)$$

Note that adding the weight κ to the slack variables in the primal (3) results in $\mathbf{d} \leq \kappa \mathbf{1}$. To make such $\mathbf{d}$ exist, we require $\kappa \geq 1/N$. In [4] it was shown that $\kappa = 1/s$, and $s \in \{1, \dots, N\}$ are favorable, and we will use these values in our work as well. The dual (4) works well in many cases and has been widely used, however, the evolution of $\mathbf{d}$ might have instabilities for noisy datasets, which can reduce the efficiency and also lead to overfitting. To avoid these drawbacks, we add a regularization term given by the Riemannian distance

$\delta(\sqrt{\mathbf{d}}, \sqrt{\mathbf{d}^0})$, and (4) becomes

$$\min_{\mathbf{d} \in \Delta_N, \mathbf{d} \leq \kappa \mathbf{1}} \max_{i=1, \dots, t} \mathbf{u}^i \cdot \mathbf{d} + \lambda \delta(\sqrt{\mathbf{d}}, \sqrt{\mathbf{d}^0}) \quad (5)$$

where, $\lambda > 0$ is the regularization parameter that balances the importance of the Riemannian distance and the maximum edge. When $\lambda = 0$, the regularization term vanishes and (5) becomes the LPBoost problem which maximizes the minimum soft margin.

At each iteration, we will obtain a weak learner h_t , and based on the performance of the weaker learner, we will update the distribution and get $\mathbf{d}^t$. The algorithm for RBoost is summarized in Algorithm 1.

2.1. Calculate $\mathbf{d}^t$ and $\mathbf{w}$

To find the update of $\mathbf{d}^t$, we need to get the Lagrangian dual first. To find the Lagrangian, we rewrite (5) in the following form

$$\min_{\beta, \mathbf{d}} \beta + \lambda \delta(\sqrt{\mathbf{d}}, \sqrt{\mathbf{d}^0}) \\ \text{s.t. } \mathbf{d} \in \Delta_N, \mathbf{d} \leq \kappa \mathbf{1}, \mathbf{u}^i \cdot \mathbf{d} \leq \beta, i=1, \dots, t \quad (6)$$

Now it's clear that the Lagrangian of (5) is given by,

$$\Phi(\mathbf{d}, \beta, \mathbf{w}, \xi, \gamma) = \beta + \lambda \delta(\sqrt{\mathbf{d}}, \sqrt{\mathbf{d}^0}) + \sum_{i=1}^t w_i (\mathbf{u}^i \cdot \mathbf{d} - \beta) + \sum_{n=1}^N \xi_n (d_n - \kappa) + \gamma (\mathbf{d} \cdot \mathbf{1} - 1). \quad (7)$$

Differentiating Φ with respect to β and γ , we get

$$\frac{\partial \Phi}{\partial \beta} = 1 - \sum_{i=1}^t w_i = 0, \text{ and } \frac{\partial \Phi}{\partial \gamma} = \mathbf{d} \cdot \mathbf{1} - 1 = 0, \quad (8)$$

thus by enforcing $\sum_{i=1}^t w_i = 1$ and $\mathbf{d} \cdot \mathbf{1} = 1$ (can be done by normalization), so we can delete β and γ from the Lagrangian dual function (7). Moreover, according to the KKT condition [5], $\xi_n (d_n - \kappa) = 0$, therefore, we are left with the partial Lagrangian

$$\Phi(\mathbf{d}, \mathbf{w}) = \lambda \delta(\sqrt{\mathbf{d}}, \sqrt{\mathbf{d}^0}) + \sum_{i=1}^t w_i \mathbf{u}^i \cdot \mathbf{d}. \quad (9)$$

Differentiating Φ with respect to $\mathbf{d}$, setting it to 0 and normalizing $\mathbf{d}$, we get

$$d_n^t = \frac{d_n^0}{(\sum_{i=1}^t w_i u_n^i)^2 Z_t}, \quad (10)$$

where Z_t is the normalization parameter.

The weight vector $\mathbf{w}$ for the weak learners should satisfy the linear programming problem (3), and can be solved using the column generation method [2]. We use the delayed column generation and the details are found in [2].

2.2. Weak learners

The weak learners that we use are thresholds picked in the following way. We randomly select one feature from the feature vector, and extract the values of this feature for all training samples, sort them, and get the mean of every two consequent values. These means and the smallest, as well as the largest values will become the candidate threshold set. If a value is above the threshold, we will assign the corresponding sample to one class, otherwise to the other class. For example, the values of all samples for the j th feature are $\{a_{1j}, a_{2j}, \dots, a_{Nj}\}$, and the sorted values are $\{\hat{a}_{1j}, \hat{a}_{2j}, \dots, \hat{a}_{Nj}\}$, then the candidate thresholds are $\{\hat{a}_{1j}, (\hat{a}_{1j} + \hat{a}_{2j})/2, \dots, (\hat{a}_{N-1j} + \hat{a}_{Nj})/2, \hat{a}_{Nj}\}$. We will choose the threshold which gives us the highest classification accuracy and take it as our weak learner.

3. EXPERIMENTS

We evaluated our method on the OASIS MRI database [6], the Epilepsy dataset [7] and the UCI repository [8].

The OASIS database contains cross-sectional collection of 416 subjects aged 18 to 96. Out of the 416 subjects, 175 subjects are younger than 40 yrs. old that we designated as young (Y), and 195 subjects are above 60 yrs. that we designated as old (O), and the other 66 subjects are designated as middle (M) aged. Each subject is represented using a 3D histogram describing the deformation field required to co-register an emerging atlas to a subject's MR brain scan. This process was accomplished by groupwise registration [9] on the OASIS dataset. The number of bins in each direction was set to $(6 \times 6 \times 6)$ for constructing the histograms of the vectors. Further, among the old aged people, we took 70 subjects, and 35 of them were known to have been diagnosed with very mild to moderate Alzheimer disease (AD) while the rest were controls. We did four groups of experiments, first classify the age groups (Y vs. M, M vs. O, and O vs. Y), and then classify the healthy and the very mild to moderate AD patients (AD vs. Control). For each experiment, we use 5-fold cross validation and report the average classification accuracy. The parameters (κ in (3) and λ in (5)) of the algorithm are set to be those maximizing the accuracy of the training dataset. We compared our results with those from CAVIAR [10] and ELPBoost. The results are shown in Table 1, which shows that our method outperforms the competing methods. In addition, our algorithm converges very fast, it uses no more than 10 iterations for each experiment, which is much less than the number of iterations required by other methods. The average CPU time taken to converge for our algorithm on a desktop computer with Intel 8 Core 2.8GHz, 24GB of memory, GNU Linux and MATLAB (Version 2010a) is 0.17s, whereas, CAVIAR requires 0.2s, and ELPBoost requires 1.34s.

The epilepsy dataset consists of 3D histograms $(6 \times 6 \times 6)$ [7] of displacement vector fields representing the registration deformation field between the left and right hippocampi in 3D. The goal is to distinguish between left and right anterior temporal lobe (L/RATL) epileptics. We compared our method with KFD+RBF [7] and ELPBoost. The result is shown in the last row of Table 1, which depicts that our method has about 2% higher accuracy than other methods in classification.

We also used the UCI repository datasets to validate our boosting method. The description of the datasets that we use in this paper is summarized in Table 2. We compare our results with those obtained from other techniques in the literature, including AdaBoost, LPBoost and ELPBoost. For the implementation of AdaBoost and LPBoost, we used the existing

codes on the website 2, and for ELPBoost, we implemented it ourselves. We used the same experimental settings as [11], where the datasets are separated into 100 predefined splits for training and testing. We perform 5-fold cross-validation on the splits (in this example, 80 splits for training and 20 splits for testing) and show the average classification in Table 3, from this we can see that our accuracy is up to 20% higher than all others.

We also record the change in the training and testing classification accuracy with respect to the number of iterations on the arrhythmia and dermatology datasets, and compare our results with those from ELPBoost. We show the mean accuracy vs. iteration number in Fig. 1. From these plots, it is evident that RBoost starts increasing very quickly and gets to satisfactory and stable accuracy in 40 iterations, which is much less than the number of iterations required by ELPBoost.

4. CONCLUSIONS

In this paper, we proposed a novel boosting method, that we called the Riemannian distance regularized LPBoost (RBoost). RBoost uses Riemannian distance between the square-root densities to constrain the evolution of the distribution, which gives a smooth regularized distribution, and as a consequence, avoids overfitting. RBoost is able to maximize the soft margin for linearly inseparable dataset, which makes it robust to noisy data. Furthermore, RBoost can be easily generalized to multiclass datasets as well as confidence-related input, where the label is not binary but falls in interval $[-1, 1]$. Finally, RBoost is very efficient and can produce satisfactory results using just a few weak learners, which makes it a promising candidate in the applications of online boosting for large scale datasets, dynamic environments and multimedia datasets, which will be the focus of our future work.

Acknowledgments

This research was in part funded by the NIH grant NS066340

References

1. Schapire R, Singer Y. Improved boosting algorithms using confidence-rated predictions. *Mach Learn.* 1999; 37(3):297–336.
2. Demiriz A, Bennett KP, Shawe-Taylor J. Linear programming boosting via column generation. *Mach Learn.* 2002; 46(1):225–254.
3. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: A statistical view of boosting. *Ann Statist.* 2000:337–374.
4. Warmuth MK, Glocer KA, Vishwanathan SV. Entropy regularized LPBoost. *Proc Int Conf Alg Learn Theory.* 2008:256–271.
5. Boyd, S.; Vandenberghe, L. *Convex optimization.* Cambridge Uni. Press; Cambridge, England: 2004.
6. Marcus DS, et al. Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. *J Cog Neurosci.* 2007; 19:1498–1507.
7. Kodipaka S, et al. Kernel fisher discriminant for shape-based classification in epilepsy. *Med Img Anal.* 2007:79–90.
8. Frank, A.; Asuncion, A. *UCI machine learning repository.* 2010.
9. Joshi S, Davis B, Jomier M, Gerig G. Unbiased diffeomorphic atlas construction for computational anatomy. *NeuroImage.* 2004:S151–S160. [PubMed: 15501084]

²<http://www.kyb.mpg.de/bs/people/nowozin/gboost/>

10. Chen T, Rangarajan A, Vemuri BC. CAVIAR: Classification via aggregated regression and its application in classifying the OASIS brain database. IEEE ISBI. 2010
11. Warmuth MK, Glocer KA, Raetsch G. Boosting algorithms for maximizing the soft margin. NIPS. 2007

Fig. 1.

Training and testing average classification accuracy vs. iteration numbers. Comparison for RBoost and ELPBoost on arrhythmia (left) and dermatology (right) datasets.

Table 1

Classification accuracy of different methods on the OASIS and the Epilepsy datasets.

dataset	proposed	CAVIAR	ELPBoost
Y vs. M	98.2	97.7	97.8
M vs. O	99.1	98.4	96.4
O vs. Y	100	99.1	98.9
AD vs. Control	97.6	90.3	96.8
dataset	proposed	KFD+RBF	ELPBoost
LATL vs. RATL	93.5	90.3	91.7

Table 2

Description of the UCI datasets.

dataset	# instances	# attributes
breast cancer	699	9
diabetes	768	8
heart disease	303	74
liver disorders	345	6
arrhythmia	452	279
dermatology	366	33

Table 3

Classification accuracy on the UCI datasets using different methods.

dataset	proposed	AdaBoost	LPBoost	ELPBoost
b-cancer	95.7	70.2	73.4	72.8
diabetes	77.8	72.2	76.5	75.6
h-disease	88.5	79.8	81.2	82.7
l-disorders	86.5	75.3	76.5	78.3

Algorithm 1**Riemannian distance regularized Boosting**

Input: $\{x_n, y_n\}_{n=1}^N, x_n \in X$, and $y_n \in \{1, -1\}$

Output: $H(x)$

Initialization: $d_n^0 = 1/N, n = 1, \dots, N$

for $t = 1$ to T **do**

 {Find the weak learner}

$$h_t \leftarrow \arg \max_{h \in H} \sum_{n=1}^N d_n^{t-1} \text{sign}(y_n h(x_n))$$

 Update $\mathbf{d}^t$ according to (5) and $\mathbf{w}$ according to (3)

end for

Return $H(x) = \text{sign}(\sum_{t=1}^T w_t h_t(x))$