LPBoost with Strong Classifiers

Yu K. Fang†
School of Computer Science and Engineering,
University of Electronic Science and Technology of China,
Chengdu 610054, Sichuan, P. R. China
E-mail: liusha.fang@gmail.com or fangyuke@uestc.edu.cn

Yan Fu, Chong J. Sun, Jun. L Zhou
School of Computer Science and Engineering,
University of Electronic Science and Technology of China,
Chengdu 610054, Sichuan, P. R. China
E-mail: fuyan@uestc.edu.cn; sumchongjing2005@163.com; jilzhou@uestc.edu.cn

Abstract

The goal of boosting algorithm is to maximize the minimum margin on sample set. Based on minimax theory, the goal can be converted into minimize the maximum edge. This idea motivates LPBoost and its variants (including TotalBoost, SoftBoost, ERLPBoost) which solve the optimization problem by linear programming. These algorithms ignore the strong classifier and just minimize the maximum edge of weak classifiers so that all the edges of weak classifier are at most $\gamma$. This paper shows that the edge of strong classifier may be higher than the maximum edge of weak classifiers and proposes a novel boosting algorithm which introduced strong classifier into the optimization problem and constrained the edges of both weak and strong classifiers no more than $\gamma$. Furthermore, we justified the reasonability of introducing strong classifier using minimax theory.

We compared our algorithm with other approaches including AdaBoost, LPBoost, TotalBoost, SoftBoost, and ERLPBoost on the UCI benchmark dataset. In simulation studies we show that our algorithm converges faster than SoftBoost and ERLPBoost. In a benchmark comparison we illustrate the competitiveness of our approach from the aspect of time consuming, and generalization error.

Keywords: boosting, strong classifier, soft margin, minimax theory, linear programming

1. Introduction

Boosting algorithms have shown considerable success in many fields, such as OCR (optical character recognition), face recognition, ranking or recommendation, text classification, natural language process. Boosting algorithms originated from PAC\textsuperscript{\textcopyright},\textsuperscript{2} (Probably Approximately Correct) learning theory.

†Corresponding address: School of Computer Science and Engineering, University of Electronic Science and Technology of China, No.2006, XiYuan Road, Chengdu, 611731, Sichuan, China. Tel.:+86-028-61830375
Yu K. Fang et al.

Kearns and Valiant (1989) postulated the boosting conjecture in the framework of PAC learning. In this method, a weak classifier (with success probability just a bit over 50%) can be boosted into a strong one (strong classifier) in the sense that the training error of the new one would go to zero with a polynomial-time run time.

The AdaBoost algorithm, proposed by Freund and Schapire\(^1,4,5\), is an efficient stage-wise-optimum method, which boosts a series of simple and weak learners into one strong learner. The corrective update of sample distribution in AdaBoost can be viewed as a solution to minimize a relative entropy of current sample distribution versus uniform distribution, and this optimum problem is subject to some linear constraints that the edge of the last hypothesis is zero\(^6,7\). One of the important properties of AdaBoost is that it has a decent iteration bound and approximately maximizes the margin of the examples\(^8\). Similar algorithms including LogitBoost\(^9\), AdaBoost\(^8,10\), all of which can be viewed as “corrective” family of boosting algorithms that enforce only a single constraint at each iteration\(^6\) (the edge of the hypothesis must be at most \(\gamma\), where \(\gamma\) is adapted).

However, a natural idea is to constrain the edges of all past hypotheses to be at most \(\gamma\) and otherwise minimize the relative entropy to the initial distribution. Basing on this idea, such algorithms (i.e. LPBoost\(^11,12\), TotalBoost\(^13\)) were proposed and are called totally corrective in the sense that they optimize their weight based on all past hypotheses. Moreover LPBoost and TotalBoost are provable maximizing the margin with linear program. Nevertheless, unlike LPBoost, in which the upper bound \(\gamma\) on the edge is chosen to be as small as possible in each iteration\(^6\) (the edge of the hypothesis must be at most \(\gamma\), where \(\gamma\) is adapted).

Maximizing the hard margin is a provably approach for low generalization error\(^3\) when the data is linearly separable. However, in case of inseparable data, maximizing the soft margin is a more robust and efficient choice. The soft margin maximization can be implemented via linear program with capping constraints for some small hard examples. Based on this idea, there are lots of boosting algorithms including AdaBoost with soft-margin\(^8\), MadaBoost\(^14\), v-arc\(^15,16\), SmoothBoost\(^17,18\), SoftBoost\(^19\), corrective ERLPBoost\(^20\), ERLPBoost\(^21\). This line of research culminated in SoftBoost and ERLPBoost, which both require \(O\left(\frac{\ln N}{\delta^2}\right)\) iteration bound within \(\delta\) accuracy converging to the maximum minimum soft margin. More accurately, SoftBoost minimizes the relative entropy to the initial distribution subject to some linear constraints on the edges of all weak hypotheses in the past, and ERLPBoost added a factor \(\frac{1}{\eta}\) of the relative entropy to the initial distribution and made a trade-off between maximizing the soft margin and minimizing the relative entropy which solves the main problem in SoftBoost: the generalization error decreases slowly in early iterations.

However, in total corrective family of algorithms, all of them update the sample distribution with weak classifiers ignoring the strong classifier. At each iteration in total corrective boosting algorithms, they just constrain the edges of existing weak classifiers to be at most \(\gamma\) even though the edge of strong classifier is larger than \(\gamma\). It can be shown that the strong classifier edge is possibly larger than the maximum edge of all weak classifiers. Thus, a natural algorithm emerged: simply add edge constraint of strong classifier into edge-restraint conditions of ERLPBoost, which make the constraints stricter. Based on this, we proposed the StrongLPBoost which introduces the constraint of strong hypothesis to improve the convergence rate.

Our new algorithm is most similar to ERLPBoost because their goals are both to optimize the soft margin with all past hypotheses on condition of minimizing relative entropy. The most important difference is that we use tighter constraints. An important result of our work is to show that this strategy may help to increase the convergence speed.

The paper is organized as follows: in Section 2 we introduce the relevant notations, basic concepts and LPBoost. Section 3 deeply discusses the 4 problems existed in LPBoost and gives solution correspondingly before describing the detailed algorithms StrongLPBoost in Section 4. Finally, Section 5 contains our experimental evaluation of StrongLPBoost and its competitors. And the paper concludes with an outlook and discussion in Section 6.

2. Preliminaries

To conclude this section, we like to point the reader to Table 1 which summarizes our notations. And then some basic notations and relevant concepts about LPBoost will be presented in this section. Firstly, we will introduce two definitions: edge and margin, which are of a provable dual problem.

The edge \(\gamma_m\) of a weak classifier \(h(x_m)\) on dataset \(X = \{(x_m, y_m); 1 \leq m \leq M\}\) is denoted by:
\[ \gamma_h = \sum_{m=1}^{M} d_m y_m h(x_m), \]

where \( M \) is sample dataset size, \( d_m \) is the weight of a sample \((x_m, y_m), y_m \in \{-1, 1\} \) and \( h(x_m) \in \{-1, 1\} \). Similarly, error rate \( e_h \) of \( h \) can be defined as:

\[ e_h = \sum_{m=1}^{M} d_m (y_m \neq h(x_m)), \]

then \( \gamma_h \) is an affine transformation of \( e_h \) on:

\[ \rho = -\frac{1}{2} \gamma_h. \]

### LPBoost with Strong Classifiers

Table 1. Notation for margins, edges, and LPBoost

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>Domain of examples</td>
</tr>
<tr>
<td>( M )</td>
<td>Number of examples ( e ) in ( X )</td>
</tr>
<tr>
<td>((x_m, y_m))</td>
<td>( m^{th} ) example</td>
</tr>
<tr>
<td>( y_m \in {-1, 1} )</td>
<td>( m^{th} ) label</td>
</tr>
<tr>
<td>( \bar{d} )</td>
<td>Distribution on examples</td>
</tr>
<tr>
<td>( \bar{w} )</td>
<td>Distribution on hypotheses</td>
</tr>
<tr>
<td>( t )</td>
<td>Iteration number</td>
</tr>
<tr>
<td>( T )</td>
<td>Final number of iterations</td>
</tr>
<tr>
<td>( u_{i,j} = y_i h^j(x_i) )</td>
<td>Convenient notation for combining labels and hypotheses</td>
</tr>
<tr>
<td>( U = { u_{i,j} }, \quad i = 1:M, \quad j = 1:T )</td>
<td>Error matrix</td>
</tr>
<tr>
<td>( u_{i,j}. )</td>
<td>( j^{th} ) row and ( i^{th} ) column of matrix ( U )</td>
</tr>
<tr>
<td>( h^j(x_m) \in {-1, 1} )</td>
<td>Prediction of hypothesis ( j ) on example ( x_m )</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Margin value</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Edge value</td>
</tr>
<tr>
<td>( \varepsilon )</td>
<td>Precision parameter</td>
</tr>
<tr>
<td>( \phi \in [0, \infty]^M )</td>
<td>Slack variable for soft margin problem</td>
</tr>
<tr>
<td>( \nu )</td>
<td>Capping parameter</td>
</tr>
<tr>
<td>( H^t(x_m) \in [-1, 1] )</td>
<td>Prediction of strong hypothesis ( t ) on example ( x_m )</td>
</tr>
<tr>
<td>( \eta )</td>
<td>Entropy regularized parameter</td>
</tr>
<tr>
<td>( \text{sign} )</td>
<td>Signal function</td>
</tr>
</tbody>
</table>

If \( \gamma_h = 1 \), the weak classifier \( h(x_m) \) has zero error, and

in the case of a random classifier, \( \gamma_h = 0 \). Generally speaking, the higher \( \gamma_h \), the more useful the classifier is.

The final strong classifier of boosting algorithm is formed as:

\[ f_w = \sum_{t=1}^{T} w_t h^t(x_m), \]

where \( T \) is the number of the weak classifier, and \( w_t \) corresponds to the weight of weak classifier \( h^t \). We let \( \rho_n \) denote the margin of a sample \((x_m, y_m)\) to \( f_w \), and \( \rho_n = y_m f_w (x_m) \).

As for the data set \( X \), its margin is the minimum margin of set. Generally speaking, the margin represents for the generalization ability of a classifier. Obviously, more training data gives better generalization, and maximizing the margin can improve the ability to generalize.

\[
\begin{array}{cccc}
 h_1 & \ldots & h_m \\
 x_1 & u_{11} & \ldots & u_{1s} & d_1 \\
 \ldots & \ldots & \ldots & \ldots & \ldots \\
 x_n & u_{n1} & \ldots & u_{ns} & d_n \\
 \overline{w} & w_1 & \ldots & w_s \\
\end{array}
\]

Fig. 1. Error matrix

It’s noteworthy that edges are linear in the distribution over samples and margins are linear in the distribution over the current set of hypotheses. This optimization problem of maximizing margin can be converted into linear programming 11, 12. According to Refs. 11 we give a brief introduction of LPBoost. Given a fixed ensemble \( H \) and training set \( X \), the error matrix \( U \) (as shown in Fig.1) contains entries \( u_{i,j} = h_j(x_i) y_i \) such that \( u_{i,j} = 1 \) if \( h_j(x_i) = y_i \) and \( u_{i,j} = -1 \) if \( h_j(x_i) \neq y_i \).

In terms of \( U \), the margin on sample \( i \) corresponds to the dot product:

\[ \rho_i = \sum_{j=1}^{M} w_j u_{i,j} = \overline{w} \cdot π_i. \]

And the margin of a set of samples is denoted as:

\[ \gamma_s = \min_i \overline{w} \cdot π_i, \] i.e. the minimum margin of all the samples. The goal is to find a weight vector \( \overline{w} \) that obtains the largest possible margin subject to the constraints \( w_j \geq 0 \) and \( \sum_j w_j = 1 \). This is a maxi-min problem where we choose \( \overline{w} \) to maximize \( \gamma_s \) subject to \( w_j \geq 0 \) and \( \sum_j w_j = 1 \). Fortunately, this problem can be turned into a linear programming problem as seen in (1).
The task and objective of LPBoost have been presented in detail. For the issue (1), its dual problem can be proven as listed in (2) via Von-Neumann’s Minimax theory. Additionally, the objective values of (1) and (2) satisfy equation (3).

Following (2), another boosting approach can be deduced, alternatively, the distribution over samples can be computed by linear programming, which maximizes the margin over all the base hypotheses. In dual problem (2), the goal is that finding \(d_\gamma\) to minimize \(\gamma\) subject to the constraints \(\sum_{i=1}^{m} u_{i,j} d_i \leq \gamma\) for \(j = 1, \ldots, t\) and \(\sum_{i} d_i = 1, d_i \geq 0\). Note that these notations or parameters have good natural explanation: \(\sum_{i} d_i, u_{i,j}\) means a score of weak classifier \(h_j\) on set \(X\). Thus, LPBoost tried to find a distribution of samples to minimize the edge of best weak classifier, which increases the weights of misclassification samples and decreases the weights of accurate classification samples.

The SoftBoost and ERLBoost represent for latest research in the variants of LPBoost. In order to ignore the bad effect of noise or difficult samples, SoftBoost adds the slack variant \(\zeta_i\) for each \(x_i\) of these samples and maximizes the “soft margin”: \(\rho - \frac{v}{m} \sum_{i=1}^{m} \zeta_i\) to form the new primal problem as shown (4). Note that the relationship between capping and the hinge loss has long been exploited by the SVM community.

Moreover, the relative entropy is introduced to ERLBoost for updating the sample distribution smoothly and continually. The relative entropy is denoted as follow:

\[\Delta(d', d) := \sum_{i} d''_i \log \frac{d''_i}{d''_i},\]

where \(d', d''\) are distributions of samples in the different iterations. Then the dual problem is listed as (5).

3. Proposition of StrongLPBoost

3.1. Strong classifier and LPBoost

Following the dual problems (2) and (5), we can see that only one weak classifier with maximum edge was chose to minimize the edge when updating the distribution \(d\). Their convergence rates can be improved by tightening the constraints of the optimization problem. The edge of strong classifier may be larger than the maximum bound of the weak classifiers. Thus, we can convert the strong classifier into a new weak one to add to the edge constraints of ERLBoost and make the constraints stricter. In this way, the convergence speed can be accelerated.

We employ formal methods to describe it in detail. Referring to the above notations and definitions, the new notations defined as follows:

\[
\gamma_{\min} = \min_{i} \left( \sum_{j} h_j(x_i) y_{d_j} \right) \text{ for } j = 1 \text{ to } t;
\]
\[ \gamma_{\text{min}} \leq \sum_{i=1}^{m} h_j(x_i) y_i d_i \leq \gamma_{\text{max}} \quad \text{for } j = 1 \text{ to } t \] (6)

\[ \sum_{j=1}^{t} H(x_j) d_j y_j = \sum_{j=1}^{t} \sum_{i=1}^{m} h_j(x_i) w_j y_i \]

(7)

Incorporating with (6), we can obtain (8):

\[ \gamma_{\text{min}} \leq \sum_{i=1}^{m} H(x_i) d_i y_j \leq \gamma_{\text{max}} \quad \text{for } j = 1 \text{ to } t \] (8)

From inequality (6), we can find that the edge of strong classifier \( H(x_i) \) is lower than the maximum edge of the weak classifiers. Using this logic, the error of final strong classifier is higher than minimum error of the weak ones. However, this conclusion conflicts with LPBoost and runs counter to the boosting theory. Thus, what’s wrong with the above conclusion and deduction? Close inspection shows that the final strong classifier should not be \( H(x_i) \) but \( H'(x_i) \). Alternatively, \( H'(x_i) \) maybe not satisfy the inequation (9). Once the edge of strong classifier was found to may be higher than the weak, a natural algorithm emerged: simply adding the edge constraint of strong classifier to the LPBoost, which makes constrains tighter. And then, the convergence rate is improved further. The experiment section gives painstaking experimental verification.

\[ \gamma_{\text{min}} \leq \sum_{i=1}^{m} H'(x_i) d_i y_i \leq \gamma_{\text{max}} \] (9)

3.2. Strong classifier and minimax theory

This section shows the necessity of introducing the strong classifier from the point of minimax theory. The goal of boosting algorithm is to maximize the margin over sample set, and this maximizing problem(left side of the equation (10)) can be converted into the minimax problem(right side of the equation (10)) according to the equation (3). In more specific terms, the minimax problem can be solved by two steps: the first is to find the weak classifier with maximum edge, and then, adjusting the weight \( d_{\text{av}} \) over samples to minimize the edge of classifier \( j \) (where \( j = \arg \max u_j d_{\text{av}} \)). All of the existing variants of LPBoost (including SoftBoost, ERLPBoost) basing on this idea find a weak classifier with maximum edge to minimize its edge.

\[ \max_{\omega} \min_{d_{\text{av}}} \omega^* U^* d_{\text{av}} = \min_{\omega} \max_{d_{\text{av}}} \omega^* U^* d_{\text{av}} \] (10)

We expand minimax equation (3) to equation (10) which can be proved to still hold via Von-Neumann’s Minimax theory. Different with (3), equation (10) employ weighted mixed strategy rather than pure strategy. Observe that when \( d_{\text{av}} \) in the left side and \( w \) in the right side are values of base vector, we arrive at the equation (3). Obviously, equation (10) is better than (3) because the mixed strategy is more practical than pure strategy. In terms of (10), its left side is weighted combination of margin over examples, and the right side is weighted combination of edge over hypotheses. In this respect, strong classifier can be representative for combination of weak hypotheses. Consequently, it is reasonable to introduce the strong classifier into the equation (3). Specifically, when strong classifier \( H \) is converted into a new weak classifier \( h' \), and add the \( h' \) into the cost matrix and solve the optimum problem (5).

4. StrongLPBoost

In the minimax problem that motivates the main algorithm of this paper, StrongLPBoost, a constraint of strong classifier is added to the constraint of linear programming (5). The modified linear programming problem is defined as minimizing problem (11) after appending the edge constraints of \( H'(x_i) \) to the dual problem. The pseudo-code is shown in Fig.2. StrongLPBoost minimizes the relative entropy to initial distribution of samples when maximizing the soft margin.

At each iteration, weak hypothesis \( h' \) is generated via calling oracle with parameter \( d^{-1} \), and then, new \( d' \) and \( w \) can be obtained by solving the (11) and (5) respectively. The problem (11) only devotes to the update of sample distribution and (5) is employed to get
Algorithm 1. StrongLPBoost:

1. Input: \( S = \{(x_1, y_1), ..., (x_n, y_n)\} \),
   desired error: \( \varepsilon > 0 \), confidence: \( \delta \),
   trade of faceor: \( \eta \), noise rate: \( 0 \leq \nu \leq 1 \),
   maximum iteration: \( \text{MaxIter} \),
   weak classifier generator: \( \text{oracle}(d) \),
   weak classifiers set: \( H \).

2. Initialize:
   \( d^0 \): uniform distribution, \( \delta^0 = \varepsilon, \text{err}(0) = 1 \).

3. for \( t = 1 \) to \( \text{MaxIter} \) do
   (a) call \( \text{oracle} \) with parameter \( d \) generating
   a new weak classifier \( h \),
   \( H = H \cup \{h\} \);
   (b) Solve the linear programming problem:
   \[
   \min_{d^0} \gamma + \eta \Delta(d^0, d^t)
   \]
   \[
   \text{s.t. } \sum_{i=1}^{n} y_i d_i \leq \gamma, \text{ for } 1 \leq j \leq t;
   \]
   \[
   \sum_{i=1}^{n} f^t(x_i) y_i d_i \leq \gamma;
   \]
   \[
   0 \leq d_j \leq \frac{\nu}{m} \sum_{j} d_j = 1;
   \]
   update distribution \( d^t \),
   \( \delta = \gamma + \eta \Delta(d^0, d^t) \);
   (c) Compute the optimum weights of classifiers
   with margin maximizing algorithm in \( \text{LPBoost} \) w:
   \[
   f^t(x) = \text{sign} \sum_{s=1}^{t} w_s h^s(x),
   \]
   \[
   \text{err}(t) = 1 - \frac{\sum_{i=1}^{n} f^t(x_i) = y_i}{n};
   \]
   (f) if \( (\delta - \delta^t) < \frac{\varepsilon^2}{2} \)
   break;
   end

4. Output: compute the optimum weights of classifiers
   with margin maximizing algorithm in \( \text{LPBoost} \) w:
   \[
   f_{\text{test}}(x) = \text{sign} \sum_{s=1}^{t} w_s h^s(x),
   \]
   where \( t \) : the number of weak classifier.

Fig. 2. StrongLPBoost algorithm pseudo-code.

Yu K. Fang et al.

the weights of hypotheses. Thus, we can get the current strong classifier:

\[
\hat{f}(x) = \sum_{s=1}^{t} w_s h^s(x).
\]
Table 2. Dataset description in the experiments

<table>
<thead>
<tr>
<th>dataset</th>
<th>Attribute number</th>
<th>Original attribute number</th>
<th>Sample size for each class</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>banana</td>
<td>2</td>
<td>2</td>
<td>1000/1000</td>
<td>2000</td>
</tr>
<tr>
<td>breast cancer</td>
<td>10</td>
<td>2</td>
<td>357/212</td>
<td>569</td>
</tr>
<tr>
<td>heart</td>
<td>14</td>
<td>2</td>
<td>150/120</td>
<td>270</td>
</tr>
<tr>
<td>image segment</td>
<td>19</td>
<td>7</td>
<td>990/1320</td>
<td>2310</td>
</tr>
<tr>
<td>ringnorm</td>
<td>21</td>
<td>2</td>
<td>3700/3700</td>
<td>7400</td>
</tr>
<tr>
<td>flare sonar</td>
<td>11</td>
<td>3</td>
<td>323/646</td>
<td>969</td>
</tr>
<tr>
<td>splice</td>
<td>60</td>
<td>3</td>
<td>1316/885</td>
<td>2201</td>
</tr>
<tr>
<td>new-thyroid</td>
<td>5</td>
<td>3</td>
<td>150/75</td>
<td>215</td>
</tr>
<tr>
<td>titanic</td>
<td>4</td>
<td>4</td>
<td>1316/885</td>
<td>2201</td>
</tr>
<tr>
<td>twonorm</td>
<td>21</td>
<td>2</td>
<td>3700/3700</td>
<td>7400</td>
</tr>
<tr>
<td>waveform</td>
<td>21</td>
<td>3</td>
<td>2000/2000</td>
<td>6000</td>
</tr>
<tr>
<td>spiral</td>
<td>3</td>
<td>2</td>
<td>900/900</td>
<td>1800</td>
</tr>
<tr>
<td>german</td>
<td>20</td>
<td>2</td>
<td>700/300</td>
<td>1000</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>2</td>
<td>500/268</td>
<td>768</td>
</tr>
</tbody>
</table>

dataset (Training: Test=60%:40%). This method will make this comparison more robust and the results more reliable. For the algorithm basing on soft margin, we set: $\varepsilon = 0.01$, $\nu = 0.05$ and $\eta = \frac{2}{\varepsilon} \ln \frac{N}{\nu}$.

5.2. Accuracy

Firstly, we evaluate the accuracy of StrongLPBoost comparing with other 4 boosting algorithms over 11 datasets. In Table III the average generalization performance (with standard deviation) over the 11 datasets with 5 models for every boosting algorithm are shown. For the purpose of more extensive comparison, we introduce other evaluation criterions (including recall, f-score, fp_rate, specificity, matthews $^{24,29,30}$). It’s difficult to list all the results of 5 algorithms, so we just show the results of AdaBoost, LPBoost, StrongLPBoost...
Yu K. Fang et al.

Table 3. Accuracy of 5 algorithms over 11 datasets: the mean and variance of 5-fold-cross validation

<table>
<thead>
<tr>
<th>Data set</th>
<th>AdaBoost</th>
<th>LPBoost</th>
<th>SoftBoost</th>
<th>ERLPBoost</th>
<th>StrongLPBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>banana</td>
<td>0.819</td>
<td>0.813</td>
<td>0.826</td>
<td>0.825</td>
<td>0.826</td>
</tr>
<tr>
<td>breast cancer</td>
<td>0.957</td>
<td>0.961</td>
<td>0.964</td>
<td>0.967</td>
<td>0.967</td>
</tr>
<tr>
<td>heart</td>
<td>0.741</td>
<td>0.741</td>
<td>0.741</td>
<td>0.741</td>
<td>0.741</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.751</td>
<td>0.751</td>
<td>0.751</td>
<td>0.751</td>
<td>0.751</td>
</tr>
<tr>
<td>new-thyroid</td>
<td>0.745</td>
<td>0.745</td>
<td>0.745</td>
<td>0.745</td>
<td>0.745</td>
</tr>
<tr>
<td>twonorm</td>
<td>0.945</td>
<td>0.945</td>
<td>0.945</td>
<td>0.945</td>
<td>0.945</td>
</tr>
<tr>
<td>waveform</td>
<td>0.925</td>
<td>0.925</td>
<td>0.925</td>
<td>0.925</td>
<td>0.925</td>
</tr>
<tr>
<td>german</td>
<td>0.742</td>
<td>0.742</td>
<td>0.742</td>
<td>0.742</td>
<td>0.742</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.751</td>
<td>0.751</td>
<td>0.751</td>
<td>0.751</td>
<td>0.751</td>
</tr>
<tr>
<td>spiral</td>
<td>0.809</td>
<td>0.809</td>
<td>0.809</td>
<td>0.809</td>
<td>0.809</td>
</tr>
</tbody>
</table>

(Note that: Bold marking the statistics of the StrongLPBoost and underscores marking the weaker results comparing with the other algorithms)

Table 4. Six evaluation criterions of 3 algorithms over 11 datasets: the average value of 5-fold-cross validation

<table>
<thead>
<tr>
<th>Data set</th>
<th>accuracy</th>
<th>recall</th>
<th>fscore</th>
<th>fp rate</th>
<th>specificity</th>
<th>matthews</th>
</tr>
</thead>
<tbody>
<tr>
<td>banana</td>
<td>0.819</td>
<td>0.813</td>
<td>0.826</td>
<td>0.076</td>
<td>0.919</td>
<td>0.868</td>
</tr>
<tr>
<td>breast cancer</td>
<td>0.957</td>
<td>0.961</td>
<td>0.964</td>
<td>0.04</td>
<td>0.961</td>
<td>0.958</td>
</tr>
<tr>
<td>heart</td>
<td>0.826</td>
<td>0.791</td>
<td>0.826</td>
<td>0.016</td>
<td>0.833</td>
<td>0.83</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.751</td>
<td>0.883</td>
<td>0.705</td>
<td>0.163</td>
<td>0.837</td>
<td>0.771</td>
</tr>
<tr>
<td>new-thyroid</td>
<td>0.884</td>
<td>0.907</td>
<td>0.881</td>
<td>0.089</td>
<td>0.911</td>
<td>0.896</td>
</tr>
<tr>
<td>twonorm</td>
<td>0.946</td>
<td>0.948</td>
<td>0.948</td>
<td>0.005</td>
<td>0.942</td>
<td>0.946</td>
</tr>
<tr>
<td>waveform</td>
<td>0.926</td>
<td>0.928</td>
<td>0.927</td>
<td>0.012</td>
<td>0.929</td>
<td>0.928</td>
</tr>
<tr>
<td>german</td>
<td>0.742</td>
<td>0.73</td>
<td>0.74</td>
<td>0.035</td>
<td>0.641</td>
<td>0.705</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.752</td>
<td>0.75</td>
<td>0.75</td>
<td>0.038</td>
<td>0.679</td>
<td>0.705</td>
</tr>
<tr>
<td>spiral</td>
<td>0.809</td>
<td>0.809</td>
<td>0.809</td>
<td>0.082</td>
<td>0.808</td>
<td>0.808</td>
</tr>
</tbody>
</table>

(As seen in Table 4). Note that except for the heart and diabetes datasets, the performance of StrongLPBoost is better than other boosting algorithms in almost all cases. For the two datasets, even though StrongLPBoost perform not as good as to AdaBoost, experimental results still show the competiveness compared with other variants of LPBoost.

5.3. Margin and iteration bound

Next, we compare the four maximizing margin algorithms from the aspects of the weak classifier relevance, margin, accuracy and iteration number. In order to make it easier to compare StrongLPBoost with SoftBoost and ERLPBoost, we use the Banana dataset...
in this experiment similar with the work by\textsuperscript{18, 19}. Note that the reason we leave out AdaBoost is that it is not based on margin maximizing theory. Fig. 4 and Fig. 5 are the experimental result of 4 algorithms over Banana dataset at a time. Fig. 4 shows the margin value along with iteration number. The result shows that StrongLPBoost has the best convergence rate; ERLPBoost and LPBoost have similar convergence speed and the convergence of SoftBoost is worse when compared with other 3 boosting algorithms. Fig. 5 describes the accuracy boosting with the iteration about four algorithms. From the two figures, it can be seen that StrongLPBoost has fast convergence speed to close to the optimum soft margin with a small quantity of weak classifiers.

5.4. Strong classifier edge constraint and convergence

In this subsection, we show that the edge constraint of strong classifier has an impact on convergence. Here,
Fig. 6. Edges of $H'(x), H(x)$ and $\gamma$ before adding the strong classifier $H'(x)$

Fig. 7. Edges of $H'(x), H(x)$ and $\gamma$ after adding the strong classifier $H'(x)$

Fig. 8. Accuracy convergence rate before and after adding the strong classifier $H'(x)$
the experimental data is based on spiral dataset (The reason why not using banana dataset is that four boosting algorithms are so easy to converge to the optimum of soft margin that it’s difficult to observe the convergence change after adding strong classifier edge constraint.). Fig. 6 shows the edges of $H(x)$, $H(x)$ and $\gamma$ along with iteration before adding the edge constraint of $H(x)$. Red curve represents for the edge of $H(x)$, however, the edge of $H(x)$ and $\gamma$ approach zero($10^{-6}$) after solving the dual problem(5) at each iteration. Similar with our conclusion in section 3, the edge of $H(x)$ (blue curve) is always lower than $\gamma$, and moreover, the edge of $H(x)$ may be lower and higher than $\gamma$. When the edge constraints of $H(x)$ are added, the three edges are shown as Fig. 7, they all approach zero. Then, we can see that $H(x)$ takes effect. The convergence changes after adding $H(x)$’s edge constraint are shown in Fig. 8.

5.5. Strong classifier and weak classifier

Finally, this experiment shows how the constraint of strong classifier influences the weak classifier. In order to simplify the experiment, 800 samples in banana dataset are used. Fig. 9 shows the 15 weak classifiers in the 20 iterations generated by SoftBoost. There are amounts of relevance and redundancy during the 20 classifiers. On the contrary, there are just seven weak

---

Fig. 9. Edges of $H(x)$, $H(x)$ and $\gamma$ before adding the strong classifier $H(x)$

Fig. 10. Weak classifiers (thin) and strong classifier (thick) generated by SoftBoost over Banana dataset
techniques established in this work can be applied to convergence. Moreover, it is interesting to see how the relation between strong classifier edge and margin under Grant No. 2008GZ0009.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No. 60903073, 60973120, 61003231; Programs for Science and Technology Development of Sichuan Province under Grant No. 2008GZ0009.

References


30. Junlin Zhou, Aleksandar Lazarevic, Kuo-Wei Hsu, etc unsupervised learning based distributed detection of global anomalies, IJITDM, 2010, Vol.9,No.6, 935-957