ABC-Boost: Adaptive Base Class Boost for Multi-class Classification

Ping Li PINGLI@CORNELL.EDU
Department of Statistical Science, Cornell University, Ithaca, NY 14853 USA

Abstract

We propose ***abc-boost*** (**a**daptive **b**ase **c**lass boost) for multi-class classification and present ***abc-mart***, an implementation of *abc-boost*, based on the multinomial logit model. The key idea is that, at each boosting iteration, we ***adaptively*** and greedily choose a ***base*** class. Our experiments on public datasets demonstrate the improvement of *abc-mart* over the original *mart* algorithm.

1. Introduction

Classification is a basic task in machine learning. A training data set $\{y_i, \mathbf{X}_i\}_{i=1}^N$ consists of N feature vectors (samples) $\mathbf{X}_i$, and N class labels, y_i. Here $y_i \in \{0, 1, 2, ..., K-1\}$ and K is the number of classes. The task is to predict the class labels. Among many classification algorithms, **boosting** has become very popular (Schapire, 1990; Freund, 1995; Freund & Schapire, 1997; Bartlett et al., 1998; Schapire & Singer, 1999; Friedman et al., 2000; Friedman, 2001).

This study focuses on multi-class classification (i.e., $K \geq 3$). The multinomial logit model has been used for solving multi-class classification problems. Using this model, we first learn the class probabilities:

$$p_{i,k} = \mathbf{Pr}\left(y_i = k|\mathbf{x}_i\right) = \frac{e^{F_{i,k}(\mathbf{x_i})}}{\sum_{s=0}^{K-1} e^{F_{i,s}(\mathbf{x_i})}}, \tag{1}$$

and then predict each class label according to

$$\hat{y}_i = \underset{k}{\text{argmax}}\ p_{i,k}. \tag{2}$$

A classification error occurs if $\hat{y}_i \neq y_i$. In (1), $F_{i,k} = F_{i,k}(\mathbf{x}_i)$ is a function to be learned from the data. Boosting algorithms (Friedman et al., 2000; Friedman, 2001) have been developed to fit the multinomial logit model. Several search engine ranking algorithms used ***mart*** (**m**ultiple **a**dditive **r**egression **t**rees) (Friedman, 2001) as the underlying learning procedure (Cossock & Zhang, 2006; Zheng et al., 2008; Li et al., 2008).

Note that in (1), the values of $p_{i,k}$ are not affected by adding a constant C to each $F_{i,k}$, because

$$\frac{e^{F_{i,k}+C}}{\sum_{s=0}^{K-1} e^{F_{i,s}+C}} = \frac{e^C e^{F_{i,k}}}{e^C \sum_{s=0}^{K-1} e^{F_{i,s}}} = \frac{e^{F_{i,k}}}{\sum_{s=0}^{K-1} e^{F_{i,s}}} = p_{i,k}$$

Therefore, for identifiability, one should impose a constraint on $F_{i,k}$. One popular choice is to assume $\sum_{k=0}^{K-1} F_{i,k} = const$, which is equivalent to $\sum_{k=0}^{K-1} F_{i,k} = 0$, i.e., the **sum-to-zero** constraint.

This study proposes ***abc-boost*** (**a**daptive **b**ase **c**lass boost), based on the following two key ideas:

1. Popular loss functions for multi-class classification usually need to impose a constraint such that only the values for $K-1$ classes are necessary (Friedman et al., 2000; Friedman, 2001; Zhang, 2004; Lee et al., 2004; Tewari & Bartlett, 2007; Zou et al., 2008). Therefore, we can choose a **base class** and derive algorithms only for $K-1$ classes.
2. At each boosting step, we can **adaptively** choose the base class to achieve the best performance.

We present a concrete implementation named ***abc-mart***, which combines *abc-boost* with *mart*. Our extensive experiments will demonstrate the improvements of our new algorithm.

2. Review Mart & Gradient Boosting

Mart is the marriage of regress trees and function gradient boosting (Friedman, 2001; Mason et al., 2000). Given a training dataset $\{y_i, \mathbf{x}_i\}_{i=1}^N$ and a loss function L, (Friedman, 2001) adopted a "greedy stagewise" approach to build an additive function $F^{(M)}$,

$$F^{(M)}(\mathbf{x}) = \sum_{m=1}^{M} \rho_m h(\mathbf{x}; \mathbf{a}_m), \tag{3}$$

such that, at each stage m, $m = 1$ to M,

$$\{\rho_m, \mathbf{a}_m\} = \underset{\rho,\mathbf{a}}{\text{argmin}} \sum_{i=1}^{N} L\left(y_i, F^{(m-1)} + \rho h(\mathbf{x}_i; \mathbf{a})\right) \tag{4}$$

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

Here $h(\mathbf{x};\mathbf{a})$ is the "weak" learner. Instead of directly solving the difficult problem (4), (Friedman, 2001) approximately conducted steepest descent in function space, by solving a least square (Line 4 in Alg. 1)

$$\mathbf{a}_m = \underset{\mathbf{a},\rho}{\mathrm{argmin}} \sum_{i=1}^{N} \left[-g_m(\mathbf{x}_i) - \rho h(\mathbf{x}_i;\mathbf{a})\right]^2,$$

where

$$-g_m(\mathbf{x}_i) = -\left[\frac{\partial L\left(y_i, F(\mathbf{x}_i)\right)}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F^{(m-1)}(\mathbf{x})}$$

is the steepest descent direction in the N-dimensional data space at $F^{(m-1)}(\mathbf{x})$. For the other coefficient ρ_m, a line search is performed (Line 5 in Alg. 1):

$$\rho_m = \underset{\rho}{\mathrm{argmin}} \sum_{i=1}^{N} L\left(y_i, F^{(m-1)}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a}_m)\right).$$

Algorithm 1 Gradient boosting (Friedman, 2001).

1: $F_{\mathbf{x}} = \mathrm{argmin}_\rho \sum_{i=1}^{N} L(y_i, \rho)$
2: For $m = 1$ to M Do
3: $\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F^{(m-1)}(\mathbf{x})}$.
4: $\quad \mathbf{a}_m = \mathrm{argmin}_{\mathbf{a},\rho} \sum_{i=1}^{N} \left[\tilde{y}_i - \rho h(\mathbf{x}_i;\mathbf{a})\right]^2$
5: $\quad \rho_m = \mathrm{argmin}_\rho \sum_{i=1}^{N} L\left(y_i, F^{(m-1)}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a}_m)\right)$
6: $\quad F_{\mathbf{x}} = F_{\mathbf{x}} + \rho_m h(\mathbf{x};\mathbf{a}_m)$
7: $\quad$ End
8: End

Mart adopted the multinomial logit model (1) and the corresponding *negative multinomial log-likelihood loss*

$$L = \sum_{i=1}^{N} L_i, \qquad L_i = -\sum_{k=0}^{K-1} r_{i,k} \log p_{i,k} \tag{5}$$

where $r_{i,k} = 1$ if $y_i = k$ and $r_{i,k} = 0$ otherwise.

(Friedman, 2001) used the following derivatives:

$$\frac{\partial L_i}{\partial F_{i,k}} = -\left(r_{i,k} - p_{i,k}\right), \tag{6}$$

$$\frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,k}\left(1 - p_{i,k}\right). \tag{7}$$

Alg. 2 describes *mart* for multi-class classification. At each stage, the algorithm solves the mean square problem (Line 4 in Alg. 1) by regression trees, and implements Line 5 in Alg. 1 by a one-step Newton update within each terminal node of trees. *Mart* builds K regression trees at each boosting step. It is clear that the constraint $\sum_{k=0}^{K-1} F_{i,k} = 0$ need not hold.

Algorithm 2 Mart (Friedman, 2001, Alg. 6)

0: $r_{i,k} = 1$, if $y_i = k$, and $r_{i,k} = 0$ otherwise.
1: $F_{i,k} = 0$, $k = 0$ to $K-1$, $i = 1$ to N
2: For $m = 1$ to M Do
3: $\quad$ For $k = 0$ to $K-1$ Do
4: $\quad\quad p_{i,k} = \exp(F_{i,k}) / \sum_{s=0}^{K-1} \exp(F_{i,s})$
5: $\quad\quad \{R_{j,k,m}\}_{j=1}^{J} = J$-terminal node regression tree
: $\quad\quad\quad$ from $\{r_{i,k} - p_{i,k},\ \ \mathbf{x}_i\}_{i=1}^{N}$
6: $\quad\quad \beta_{j,k,m} = \frac{K-1}{K} \frac{\sum_{\mathbf{x}_i \in R_{j,k,m}} r_{i,k} - p_{i,k}}{\sum_{\mathbf{x}_i \in R_{j,k,m}} (1-p_{i,k}) p_{i,k}}$
7: $\quad\quad F_{i,k} = F_{i,k} + \nu \sum_{j=1}^{J} \beta_{j,k,m} 1_{\mathbf{x}_i \in R_{j,k,m}}$
8: $\quad$ End
9: End

Alg. 2 has three main parameters. The number of terminal nodes, J, determines the capacity of the weak learner. (Friedman, 2001) suggested $J = 6$. (Friedman et al., 2000; Zou et al., 2008) commented that $J > 10$ is very unlikely. The shrinkage, ν, should be large enough to make sufficient progress at each step and small enough to avoid over-fitting. (Friedman, 2001) suggested $\nu \leq 0.1$. The number of iterations, M, is largely determined by the affordable computing time.

3. Abc-boost and Abc-mart

Abc-mart implements *abc-boost*. Corresponding to the two key ideas of *abc-boost* in Sec. 1, we need to: (A) re-derive the derivatives of (5) under the sum-to-zero constraint; (B) design a strategy to adaptively select the base class at each boosting iteration.

3.1. Derivatives of the Multinomial Logit Model with a Base Class

Without loss of generality, we assume class 0 is the base. Lemma 1 provides the derivatives of the class probabilities $p_{i,k}$ under the logit model (1).

Lemma 1

$$\frac{\partial p_{i,k}}{\partial F_{i,k}} = p_{i,k}\left(1 + p_{i,0} - p_{i,k}\right), \quad k \neq 0$$

$$\frac{\partial p_{i,k}}{\partial F_{i,s}} = p_{i,k}\left(p_{i,0} - p_{i,s}\right), \quad k \neq s \neq 0$$

$$\frac{\partial p_{i,0}}{\partial F_{i,k}} = p_{i,0}\left(-1 + p_{i,0} - p_{i,k}\right), \quad k \neq 0$$

Proof: *Note that* $F_{i,0} = -\sum_{k=1}^{K-1} F_{i,k}$. *Hence*

$$p_{i,k} = \frac{e^{F_{i,k}}}{\sum_{s=0}^{K-1} e^{F_{i,s}}} = \frac{e^{F_{i,k}}}{\sum_{s=1}^{K-1} e^{F_{i,s}} + e^{\sum_{s=1}^{K-1} -F_{i,s}}}$$

$$\frac{\partial p_{i,k}}{\partial F_{i,k}} = \frac{e^{F_{i,k}}}{\sum_{s=0}^{K-1} e^{F_{i,s}}} - \frac{e^{F_{i,k}}\left(e^{F_{i,k}} - e^{-F_{i,0}}\right)}{\left(\sum_{s=0}^{K-1} e^{F_{i,s}}\right)^2}$$

$$= p_{i,k}\left(1 + p_{i,0} - p_{i,k}\right).$$

The other derivatives can be obtained similarly. □

Lemma 2 provides the derivatives of the loss (5).

Lemma 2 *For $k \neq 0$,*

$$\frac{\partial L_i}{\partial F_{i,k}} = (r_{i,0} - p_{i,0}) - (r_{i,k} - p_{i,k}),$$
$$\frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,0}(1 - p_{i,0}) + p_{i,k}(1 - p_{i,k}) + 2p_{i,0}p_{i,k}.$$

Proof:

$$L_i = -\sum_{s=1,s\neq k}^{K-1} r_{i,s} \log p_{i,s} - r_{i,k} \log p_{i,k} - r_{i,0} \log p_{i,0}.$$

Its first derivative is

$$\begin{aligned}\frac{\partial L_i}{\partial F_{i,k}} =& -\sum_{s=1,s\neq k}^{K-1} \frac{r_{i,s}}{p_{i,s}}\frac{\partial p_{i,s}}{F_{i,k}} - \frac{r_{i,k}}{p_{i,k}}\frac{\partial p_{i,k}}{F_{i,k}} - \frac{r_{i,0}}{p_{i,0}}\frac{\partial p_{i,0}}{F_{i,k}}\\
=& \sum_{s=1,s\neq k}^{K-1} -r_{i,s}\left(p_{i,0} - p_{i,k}\right) - r_{i,k}\left(1 + p_{i,0} - p_{i,k}\right)\\
&- r_{i,0}\left(-1 + p_{i,0} - p_{i,k}\right)\\
=& -\sum_{s=0}^{K-1} r_{i,s}\left(p_{i,0} - p_{i,k}\right) + r_{i,0} - r_{i,k}\\
=& \left(r_{i,0} - p_{i,0}\right) - \left(r_{i,k} - p_{i,k}\right).\end{aligned}$$

And the second derivative is

$$\begin{aligned}\frac{\partial^2 L_i}{\partial F_{i,k}^2} =& -\frac{\partial p_{i,0}}{\partial F_{i,k}} + \frac{\partial p_{i,k}}{\partial F_{i,k}}\\
=& -p_{i,0}\left(-1 + p_{i,0} - p_{i,k}\right) + p_{i,k}\left(1 + p_{i,0} - p_{i,k}\right)\\
=& p_{i,0}(1 - p_{i,0}) + p_{i,k}(1 - p_{i,k}) + 2p_{i,0}p_{i,k}. \ \square\end{aligned}$$

3.2. The Exhaustive Strategy for the Base

We adopt a greedy strategy. At each training iteration, we try each class as the base and choose the one that achieves the **smallest** training loss (5) as the final base class, **for the current iteration**. Alg. 3 describes *abc-mart* using this strategy.

3.3. More Insights in Mart and Abc-mart

First of all, one can verify that *abc-mart* recovers *mart* when $K = 2$. For example, consider $K = 2$, $r_{i,0} = 1$, $r_{i,1} = 0$, then $\frac{\partial L_i}{\partial F_{i,1}} = 2p_{i,1}$, $\frac{\partial^2 L_i}{\partial F_{i,1}^2} = 4p_{i,0}p_{i,1}$. Thus, the factor $\frac{K-1}{K} = \frac{1}{2}$ appeared in Alg. 2 is recovered.

When $K \geq 3$, it is interesting that *mart* used the averaged first derivatives. The following equality

$$\sum_{b\neq k}\left\{-(r_{i,b} - p_{i,b}) + (r_{i,k} - p_{i,k})\right\} = K(r_{i,k} - p_{i,k}),$$

Algorithm 3 *Abc-mart* using the exhaustive search strategy. The vector B stores the base class numbers.

$r_{i,k} = 1$, if $y_i = k$, $r_{i,k} = 0$ otherwise.
$F_{i,k} = 0$, $p_{i,k} = \frac{1}{K}$, $k = 0$ to $K-1$, $i = 1$ to N
For $m = 1$ to M Do
  For $b = 0$ to $K-1$, Do
    For $k = 0$ to $K-1$, $k \neq b$, Do
      $\{R_{j,k,m}\}_{j=1}^{J}$ = J-terminal node regression tree
       from $\{-(r_{i,b} - p_{i,b}) + (r_{i,k} - p_{i,k}),\ \ \mathbf{x}_i\}_{i=1}^{N}$
    $\beta_{j,k,m} = \frac{\sum_{\mathbf{x}_i\in R_{j,k,m}} -(r_{i,b}-p_{i,b})+(r_{i,k}-p_{i,k})}{\sum_{\mathbf{x}_i\in R_{j,k,m}} p_{i,b}(1-p_{i,b})+p_{i,k}(1-p_{i,k})+2p_{i,b}p_{i,k}}$
      $G_{i,k,b} = F_{i,k} + \nu\sum_{j=1}^{J}\beta_{j,k,m}1_{\mathbf{x}_i\in R_{j,k,m}}$
    End
    $G_{i,b,b} = -\sum_{k\neq b} G_{i,k,b}$
    $q_{i,k} = \exp(G_{i,k,b})/\sum_{s=0}^{K-1}\exp(G_{i,s,b})$
    $L^{(b)} = -\sum_{i=1}^{N}\sum_{k=0}^{K-1} r_{i,k}\log(q_{i,k})$
  End
  $B(m) = \underset{b}{\text{argmin}}\ L^{(b)}$
  $F_{i,k} = G_{i,k,B(m)}$
  $p_{i,k} = \exp(F_{i,k})/\sum_{s=0}^{K-1}\exp(F_{i,s})$
End

holds because

$$\begin{aligned}&\sum_{b\neq k}\left\{-(r_{i,b} - p_{i,b}) + (r_{i,k} - p_{i,k})\right\}\\
=& -\sum_{b\neq k} r_{i,b} + \sum_{b\neq k} p_{i,b} + (K-1)(r_{i,k} - p_{i,k})\\
=& -1 + r_{i,k} + 1 - p_{i,k} + (K-1)(r_{i,k} - p_{i,k})\\
=& K(r_{i,k} - p_{i,k}).\end{aligned}$$

We can also show that, for the second derivatives,

$$\begin{aligned}&\sum_{b\neq k}\left\{(1 - p_{i,b})p_{i,b} + (1 - p_{i,k})p_{i,k} + 2p_{i,b}p_{i,k}\right\}\\
\geq&(K+2)(1 - p_{i,k})p_{i,k},\end{aligned}$$

with equality holding when $K = 2$, because

$$\begin{aligned}&\sum_{b\neq k}\left\{(1 - p_{i,b})p_{i,b} + (1 - p_{i,k})p_{i,k} + 2p_{i,b}p_{i,k}\right\}\\
=&(K+1)(1 - p_{i,k})p_{i,k} + \sum_{b\neq k} p_{i,b} - \sum_{b\neq k} p_{i,b}^2\\
\geq&(K+1)(1 - p_{i,k})p_{i,k} + \sum_{b\neq k} p_{i,b} - \left(\sum_{b\neq k} p_{i,b}\right)^2\\
=&(K+1)(1 - p_{i,k})p_{i,k} + (1 - p_{i,k})p_{i,k}\\
=&(K+2)(1 - p_{i,k})p_{i,k}.\end{aligned}$$

The factor $\frac{K-1}{K}$ in Alg. 2 may be reasonably replaced by $\frac{K}{K+2}$ (both equal $\frac{1}{2}$ when $K = 2$), or smaller. In a sense, to make the comparisons more fair, we should have replaced the shrinkage factor ν in Alg. 3 by ν',

$$\nu' \geq \nu \frac{K-1}{K}\frac{K+2}{K} = \nu\frac{K^2+K-2}{K^2} \geq \nu.$$

In other words, the shrinkage used in *mart* is effectively larger than the same shrinkage used in *abc-mart*.

4. Evaluations

Our experiments were conducted on several public datasets (Table 1), including one large dataset and several small or very small datasets. Even smaller datasets will be too sensitive to the implementation (or tuning) of weak learners.

Table 1. Whenever possible, we used the standard (default) training and test sets. For *Covertype*, we randomly split the original dataset into halves. For *Letter*, the default test set consisted of the last 4000 samples. For *Letter2k* (*Letter4k*), we took the last 2000 (4000) samples of *Letter* for training and the remaining 18000 (16000) for test.

dataset	K	# training	# test	# features
Covertype	7	290506	290506	54
Letter	26	16000	4000	16
Letter2k	26	2000	18000	16
Letter4k	26	4000	16000	16
Pendigits	10	7494	3498	16
Zipcode	10	7291	2007	256
Optdigits	10	3823	1797	64
Isolet	26	6218	1559	617

Ideally, we hope that *abc-mart* will improve *mart* (or be as good as *mart*), for every reasonable combination of tree size J and shrinkage ν.

Except for *Covertype* and *Isolet*, we experimented with every combination of $\nu \in \{0.04,\ 0.06,\ 0.08,\ 0.1\}$ and $J \in \{4,\ 6,\ 8,\ 10,\ 12,\ 14,\ 16,\ 18,\ 20\}$. Except for *Covertype*, we let the number of boosting steps $M = 10000$. However, the experiments usually terminated earlier because the machine accuracy was reached.

For the *Covertype* dataset, since it is fairly large, we only experimented with $J = 6, 10, 20$ and $\nu = 0.1$. We limited $M = 5000$, which is probably already a too large learning model for real applications, especially applications that are sensitive to the test time.

4.1. Summary of Experiment Results

We define R_{err}, the "relative improvement of test mis-classification errors" as

$$R_{err} = \frac{\text{error of mart} - \text{error of abc-mart}}{\text{error of mart}}. \qquad (8)$$

Since we experimented with a series of parameters, J, ν, and M, we report, in Table 2, the overall "best" (i.e., smallest) mis-classification errors. Later, we will also report the more detailed mis-classification errors for every combination of J and ν, in Sec. 4.2 to 4.9. We believe this is a fair (side-by-side) comparison.

Table 2. Summary of test mis-classification errors.

Dataset	**mart**	**abc-mart**	R_{err} (%)	P-value
Covertype	11350	10420	8.2	0
Letter	129	99	23.3	0.02
Letter2k	2439	2180	10.6	0
Letter4k	1352	1126	16.7	0
Pendigits	124	100	19.4	0.05
Zipcode	111	100	9.9	0.22
Optdigits	55	43	21.8	0.11
Isolet	80	64	20.0	0.09

In Table 2, we report the numbers of mis-classification errors mainly for the convenience of future comparisons with this work. The reported P-values are based on the **error rate**, for testing whether *abc-mart* has statistically lower error rates than *mart*. We should mention that testing the statistical significance of the difference of two small probabilities (error rates) requires particularly strong evidence.

4.2. Experiments on the *Covertype* Dataset

Table 3 summarizes the smallest test mis-classification errors along with the relative improvements (R_{err}). For each J and ν, the smallest test errors, separately for *abc-mart* and *mart*, are the lowest points in the curves in Figure 2, which are almost always the last points on the curves, for this dataset.

Table 3. ***Covertype***. We report the test mis-classification errors of *mart* and *abc-mart*, together with the results from Friedman's MART program (in [ ]). The relative improvements (R_{err}, %) of *abc-mart* are included in ( ).

ν	M	J	**mart**	**abc-mart**
0.1	1000	6	40072 [39775]	34758 (13.3)
0.1	1000	10	29456 [29196]	23577 (20.0)
0.1	1000	20	19109 [19438]	15362 (19.6)
0.1	2000	6	31541 [31526]	26126 (17.2)
0.1	2000	10	21774 [21698]	17479 (19.7)
0.1	2000	20	14505 [14665]	12045 (17.0)
0.1	3000	6	26972 [26742]	22111 (18.0)
0.1	3000	10	18494 [18444]	14984 (19.0)
0.1	3000	20	12740 [12893]	11087 (13.0)
0.1	5000	6	22511 [22335]	18445 (18.1)
0.1	5000	10	15450 [15429]	13018 (15.7)
0.1	5000	20	11350 [11524]	10420 (8.2)

The results on *Covertype* are reported differently from other datasets. *Covertype* is fairly large. Building a very large model (e.g., $M = 5000$ boosting steps) would be expensive. Testing a very large model at run-time can be costly or infeasible for certain applications. Therefore, it is often important to examine the performance of the algorithm at much earlier boosting iterations. Table 3 shows that *abc-mart* may improve *mart* as much as $R_{err} \approx 20\%$, as opposed to the reported $R_{err} = 8.2\%$ in Table 2.

Figure 1 indicates that *abc-mart* reduces the training loss (5) considerably and consistently faster than *mart*. Figure 2 demonstrates that *abc-mart* exhibits considerably and consistently smaller test mis-classification errors than *mart*.

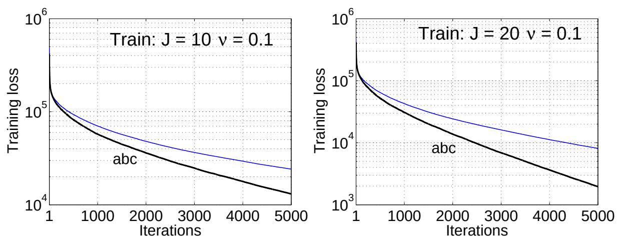

Figure 1. ***Covertype***. The training loss, i.e., (5).

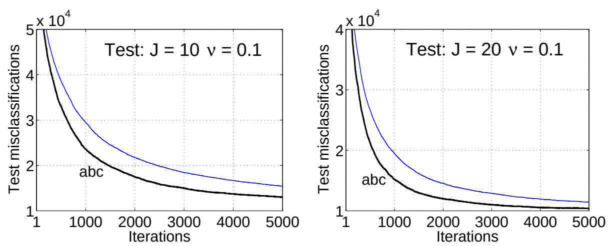

Figure 2. ***Covertype***. The test mis-classification errors.

4.3. Experiments on the *Letter* Dataset

We trained till the loss (5) reached machine accuracy, to exhaust the capacity of the learner so that we could provide a reliable comparison, up to $M = 10000$.

Table 4 summarizes the smallest test mis-classification errors along with the relative improvements. For *abc-mart*, the smallest error usually occurred at very close to the last iteration, as reflected in Figure 3, which again demonstrates that *abc-mart* exhibits considerably and consistently smaller test errors than *mart*.

One observation is that test errors are fairly stable across J and ν, unless J and ν are small (machine accuracy was not reached in these cases).

4.4. Experiments on the *Letter2k* Dataset

Table 5 and Figure 4 present the test errors, illustrating the considerable and consistent improvement of *abc-mart* over *mart* on this dataset.

Table 4. ***Letter***. We report the test mis-classification errors of *mart* and *abc-mart*, together with Friedman's MART program results (in [ ]). The relative improvements (R_{err}, %) of *abc-mart* are included in ( ).

	$\nu = 0.04$	$\nu = 0.06$	$\nu = 0.08$	$\nu = 0.1$
	mart			
$J = 4$	174 [178]	177 [176]	177 [177]	172 [177]
$J = 6$	163 [153]	157 [160]	159 [156]	159 [162]
$J = 8$	155 [151]	148 [152]	155 [148]	144 [151]
$J = 10$	145 [141]	145 [148]	136 [144]	142 [136]
$J = 12$	143 [142]	147 [143]	139 [145]	141 [145]
$J = 14$	141 [151]	144 [150]	145 [144]	152 [142]
$J = 16$	143 [148]	145 [146]	139 [145]	141 [137]
$J = 18$	132 [132]	133 [137]	129 [135]	138 [134]
$J = 20$	129 [143]	140 [135]	134 [139]	136 [143]
	abc-mart			
$J = 4$	152 (12.6)	147 (16.9)	142 (19.8)	137 (20.3)
$J = 6$	127 (22.1)	126 (19.7)	118 (25.8)	119 (25.2)
$J = 8$	122 (21.3)	112 (24.3)	108 (30.3)	103 (28.5)
$J = 10$	126 (13.1)	115 (20.7)	106 (22.1)	100 (29.6)
$J = 12$	117 (18.2)	114 (22.4)	107 (23.0)	104 (26.2)
$J = 14$	112 (20.6)	113 (21.5)	106 (26.9)	108 (28.9)
$J = 16$	111 (22.4)	112 (22.8)	106 (23.7)	99 (29.8)
$J = 18$	113 (14.4)	110 (17.3)	108 (16.3)	104 (24.6)
$J = 20$	100 (22.5)	104 (25.7)	100 (25.4)	102 (25.0)

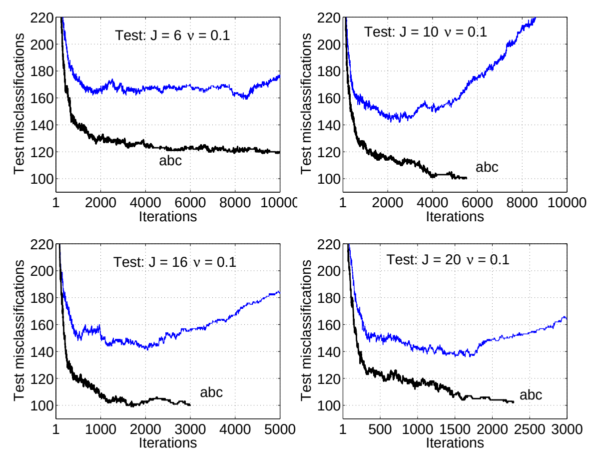

Figure 3. ***Letter***. The test mis-classification errors.

Table 5. ***Letter2k***. The test mis-classification errors.

	$\nu = 0.04$	$\nu = 0.06$	$\nu = 0.08$	$\nu = 0.1$
	mart			
J=4	2694 [2750]	2698 [2728]	2684 [2706]	2689 [2733]
J=6	2683 [2720]	2664 [2688]	2640 [2716]	2629 [2688]
J=8	2569 [2577]	2603 [2579]	2563 [2603]	2571 [2559]
J=10	2534 [2545]	2516 [2546]	2504 [2539]	2491 [2514]
J=12	2503 [2474]	2516 [2465]	2473 [2492]	2492 [2455]
J=14	2488 [2432]	2467 [2482]	2460 [2451]	2460 [2454]
J=16	2503 [2499]	2501 [2494]	2496 [2437]	2500 [2424]
J=18	2494 [2464]	2497 [2482]	2472 [2489]	2439 [2476]
J=20	2499 [2507]	2512 [2523]	2504 [2460]	2482 [2505]
	abc-mart			
J=4	2476 (8.1)	2458 (8.9)	2406 (10.4)	2407 (10.5)
J=6	2355 (12.2)	2319 (12.9)	2309 (12.5)	2314 (12.0)
J=8	2277 (11.4)	2281 (12.4)	2253 (12.1)	2241 (12.8)
J=10	2236 (11.8)	2204 (12.4)	2190 (12.5)	2184 (12.3)
J=12	2199 (12.1)	2210 (12.2)	2193 (11.3)	2200 (11.7)
J=14	2202 (11.5)	2218 (10.1)	2198 (10.7)	2180 (11.4)
J=16	2215 (11.5)	2216 (11.4)	2228 (10.7)	2202 (11.9)
J=18	2216 (11.1)	2208 (11.6)	2205 (10.8)	2213 (9.3)
J=20	2199 (12.0)	2195 (12.6)	2183 (12.8)	2213 (10.8)

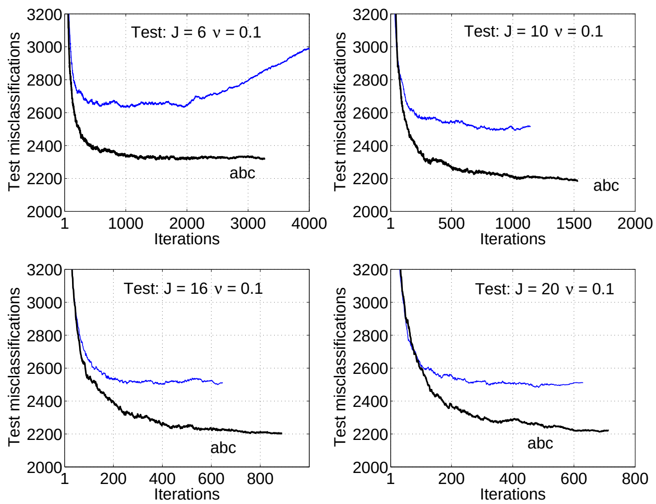

Figure 4. ***Letter2k***. The test mis-classification errors.

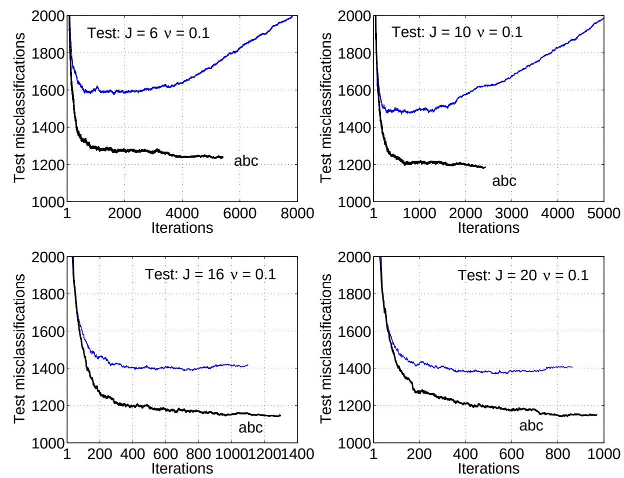

Figure 5. ***Letter4k***. The test mis-classification errors.

Figure 4 indicates that we train more iterations for *abc-mart* than *mart*, for some cases. In our experiments, we terminated training *mart* whenever the training loss (5) reached 10^{-14} because we found out that, for most datasets and parameter settings, *mart* had difficulty reaching smaller training loss. In comparisons, *abc-mart* usually had no problem of reducing the training loss (5) down to 10^{-16} or smaller; and hence we terminated training *abc-mart* at 10^{-16}. For the *Letter2k* dataset, our choice of the termination conditions may cause *mart* to terminate earlier than *abc-mart* in some cases. Also, as explained in Sec. 3.3, the fact that *mart* effectively uses larger shrinkage than *abc-mart* may be partially responsible for the phenomenon in Figure 4.

4.5. Experiments on the *Letter4k* Dataset

Table 6. ***Letter4k***. The test mis-classification errors.

		mart		
	$\nu = 0.04$	$\nu = 0.06$	$\nu = 0.08$	$\nu = 0.1$
$J = 4$	1681 [1664]	1660 [1684]	1671 [1664]	1655 [1672]
$J = 6$	1618 [1584]	1584 [1584]	1588 [1596]	1577 [1588]
$J = 8$	1531 [1508]	1522 [1492]	1516 [1492]	1521 [1548]
$J = 10$	1499 [1500]	1463 [1480]	1479 [1480]	1470 [1464]
$J = 12$	1420 [1456]	1434 [1416]	1409 [1428]	1437 [1424]
$J = 14$	1410 [1412]	1388 [1392]	1377 [1400]	1396 [1380]
$J = 16$	1395 [1428]	1402 [1392]	1396 [1404]	1387 [1376]
$J = 18$	1376 [1396]	1375 [1392]	1357 [1400]	1352 [1364]
$J = 20$	1386 [1384]	1397 [1416]	1371 [1388]	1370 [1388]
		abc-mart		
$J = 4$	1407 (16.3)	1372 (17.3)	1348 (19.3)	1318 (20.4)
$J = 6$	1292 (20.1)	1285 (18.9)	1261 (20.6)	1234 (21.8)
$J = 8$	1259 (17.8)	1246 (18.1)	1191 (21.4)	1183 (22.2)
$J = 10$	1228 (18.1)	1201 (17.9)	1181 (20.1)	1182 (19.6)
$J = 12$	1213 (14.6)	1178 (17.9)	1170 (17.0)	1162 (19.1)
$J = 14$	1181 (16.2)	1154 (16.9)	1148 (16.6)	1158 (17.0)
$J = 16$	1167 (16.3)	1153 (17.8)	1154 (17.3)	1142 (17.7)
$J = 18$	1164 (15.4)	1136 (17.4)	1126 (17.0)	1149 (15.0)
$J = 20$	1149 (17.1)	1127 (19.3)	1126 (17.9)	1142 (16.4)

4.6. Experiments on the *Pendigits* Dataset

Table 7. ***Pendigits***. The test mis-classification errors.

		mart		
	$\nu = 0.04$	$\nu = 0.06$	$\nu = 0.08$	$\nu = 0.1$
$J = 4$	144 [145]	145 [146]	145 [144]	143 [142]
$J = 6$	135 [139]	135 [140]	143 [137]	135 [138]
$J = 8$	133 [133]	130 [132]	129 [133]	128 [134]
$J = 10$	132 [132]	129 [128]	127 [128]	130 [132]
$J = 12$	136 [134]	134 [134]	135 [140]	133 [134]
$J = 14$	129 [126]	131 [131]	130 [133]	133 [131]
$J = 16$	129 [127]	130 [129]	133 [132]	134 [126]
$J = 18$	132 [129]	130 [129]	126 [128]	133 [131]
$J = 20$	130 [129]	125 [125]	126 [130]	130 [128]
		abc-mart		
$J = 4$	109 (24.3)	106 (26.9)	106 (26.9)	107 (25.2)
$J = 6$	109 (19.3)	105 (22.2)	104 (27.3)	102 (24.4)
$J = 8$	105 (20.5)	101 (22.3)	104 (19.4)	104 (18.8)
$J = 10$	102 (17.7)	102 (20.9)	102 (19.7)	100 (23.1)
$J = 12$	101 (25.7)	103 (23.1)	103 (23.7)	105 (21.1)
$J = 14$	105 (18.6)	102 (22.1)	102 (21.5)	102 (23.3)
$J = 16$	109 (15.5)	107 (17.7)	106 (20.3)	106 (20.9)
$J = 18$	110 (16.7)	106 (18.5)	105 (16.7)	105 (21.1)
$J = 20$	109 (16.2)	105 (16.0)	107 (15.1)	105 (19.2)

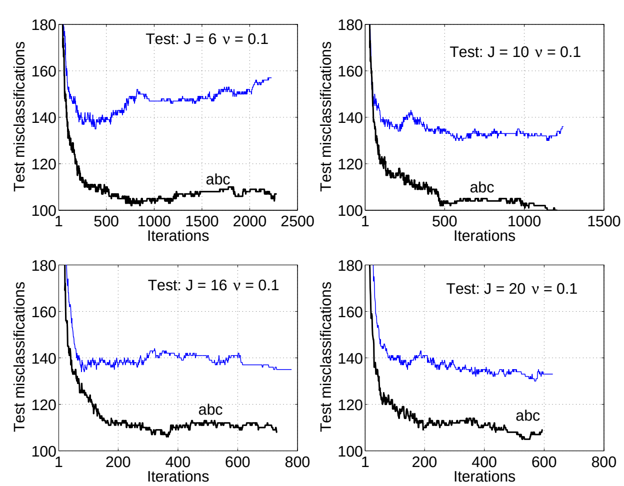

Figure 6. ***Pendigits***. The test mis-classification errors.

4.7. Experiments on the *Zipcode* Dataset

Table 8. ***Zipcode***. We report the test mis-classification errors of *mart* and *abc-mart*, together with the results from Friedman's MART program (in [ ]). The relative improvements (R_{err}, %) of *abc-mart* are included in ( ).

	$\nu = 0.04$	$\nu = 0.06$	$\nu = 0.08$	$\nu = 0.1$
mart				
$J = 4$	130 [132]	125 [128]	129 [132]	127 [128]
$J = 6$	123 [126]	124 [128]	123 [127]	126 [124]
$J = 8$	120 [119]	122 [123]	122 [120]	123 [115]
$J = 10$	118 [117]	118 [119]	120 [115]	118 [117]
$J = 12$	117 [118]	116 [118]	117 [116]	118 [113]
$J = 14$	118 [115]	120 [116]	119 [114]	118 [114]
$J = 16$	119 [121]	111 [113]	116 [114]	115 [114]
$J = 18$	113 [120]	114 [116]	114 [116]	114 [113]
$J = 20$	114 [111]	112 [110]	115 [110]	111 [115]
abc-mart				
$J = 4$	120 (7.7)	113 (9.5)	116 (10.1)	109 (14.2)
$J = 6$	110 (10.6)	112 (9.7)	109 (11.4)	104 (17.5)
$J = 8$	106 (11.7)	102 (16.4)	103 (15.5)	103 (16.3)
$J = 10$	103 (12.7)	104 (11.9)	106 (11.7)	105 (11.0)
$J = 12$	103 (12.0)	101 (12.9)	101 (13.7)	104 (11.9)
$J = 14$	103 (12.7)	106 (11.7)	103 (13.4)	104 (11.9)
$J = 16$	106 (10.9)	102 (8.1)	100 (13.8)	104 (9.6)
$J = 18$	102 (9.7)	100 (12.3)	101 (11.4)	101 (11.4)
$J = 20$	104 (8.8)	103 (8.0)	105 (8.7)	105 (5.4)

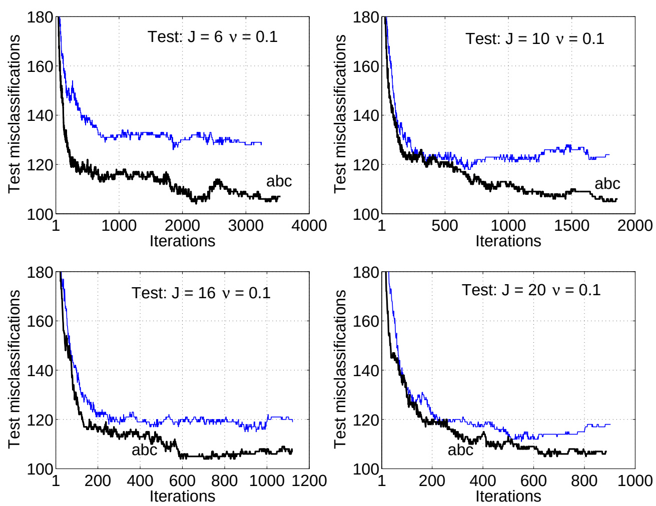

Figure 7. ***Zipcode***. The test mis-classification errors.

4.8. Experiments on the *Optdigits* Dataset

This dataset is one of two largest datasets (*Optdigits* and *Pendigits*) used in a recent paper on boosting (Zou et al., 2008), which proposed the multi-class **gentleboost** and **ADABOOST.ML**.

For *Pendigits*, (Zou et al., 2008) reported 3.69%, 4.09%, and 5.86% error rates, for gentleboost, ADABOOST.ML, and **ADABOOST.MH** (Schapire & Singer, 2000), respectively. For *Optdigits*, they reported 5.01%, 5.40%, and 5.18%, respectively.

Table 9. ***Optdigits***. The test mis-classification errors

	$\nu = 0.04$	$\nu = 0.06$	$\nu = 0.08$	$\nu = 0.1$
mart				
$J = 4$	58 [61]	57 [58]	57 [57]	59 [58]
$J = 6$	58 [57]	57 [54]	59 [59]	57 [56]
$J = 8$	61 [62]	60 [58]	57 [59]	60 [56]
$J = 10$	60 [62]	55 [59]	57 [57]	60 [60]
$J = 12$	57 [60]	58 [59]	56 [60]	60 [59]
$J = 14$	57 [57]	58 [61]	58 [59]	55 [57]
$J = 16$	60 [60]	58 [59]	59 [58]	57 [58]
$J = 18$	60 [59]	59 [60]	59 [58]	59 [57]
$J = 20$	58 [60]	61 [62]	58 [60]	59 [60]
abc-mart				
$J = 4$	48 (17.2)	45 (21.1)	46 (19.3)	43 (27.1)
$J = 6$	43 (25.9)	47 (17.5)	43 (27.1)	43 (24.6)
$J = 8$	46 (24.6)	45 (25.0)	46 (19.3)	47 (21.6)
$J = 10$	48 (20.0)	47 (14.5)	47 (17.5)	50 (16.7)
$J = 12$	47 (17.5)	48 (17.2)	47 (16.1)	47 (21.7)
$J = 14$	50 (12.3)	50 (13.8)	49 (15.5)	47 (14.6)
$J = 16$	51 (15.0)	48 (17.2)	46 (22.0)	49 (14.0)
$J = 18$	49 (18.3)	48 (18.6)	49 (20.0)	50 (15.3)
$J = 20$	51 (12.1)	48 (21.3)	50 (13.8)	47 (20.3)

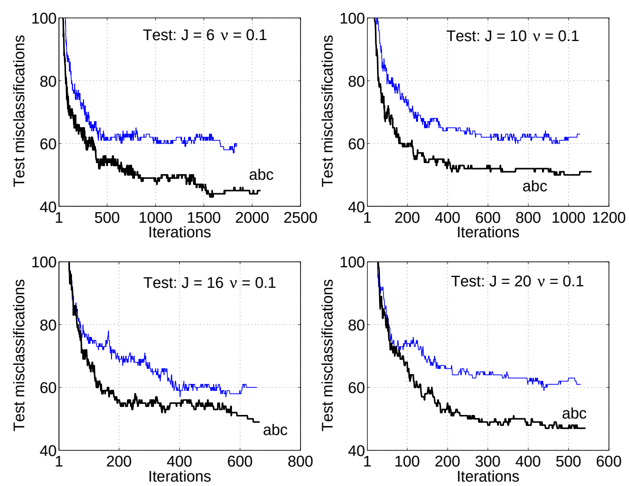

Figure 8. ***Optdigits***. The test mis-classification errors.

4.9. Experiments on the *Isolet* Dataset

This dataset is high-dimensional with 617 features, for which tree algorithms become less efficient. We have only conducted experiments for one shrinkage, i.e,. $\nu = 0.1$.

Table 10. ***Isolet***. The test mis-classification errors.

	mart	**abc-mart**
	$\nu = 0.1$	$\nu = 0.1$
$J = 4$	80 [86]	64 (20.0)
$J = 6$	84 [86]	67 (20.2)
$J = 8$	84 [88]	72 (14.3)
$J = 10$	82 [83]	74 (9.8)
$J = 12$	91 [90]	74 (18.7)
$J = 14$	95 [94]	74 (22.1)
$J = 16$	94 [92]	78 (17.0)
$J = 18$	86 [91]	78 (9.3)
$J = 20$	87 [94]	78 (10.3)

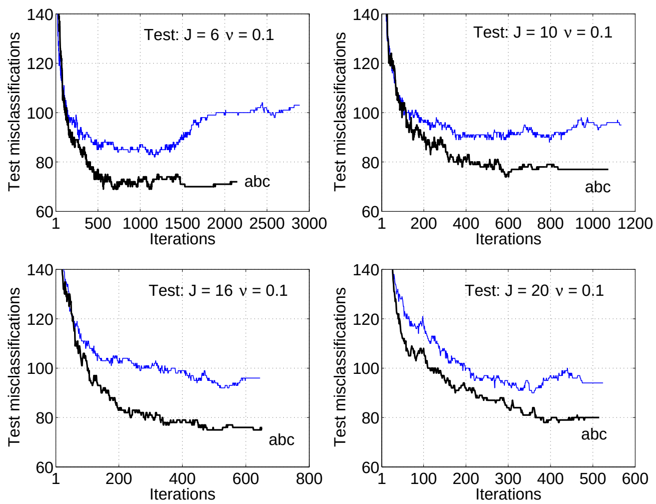

Figure 9. ***Isolet.*** The test mis-classification errors.

5. Conclusion

We present the concept of *abc-boost* and its concrete implementation named *abc-mart*, for multi-class classification (with $K \geq 3$ classes). Two key components of *abc-boost* include: (A) By enforcing the (commonly used) constraint on the loss function, we can derive boosting algorithms for only $K - 1$ classes using a **base class**; (B) We **adaptively** (and greedily) choose the base class at each boosting step. Our experiments demonstrated the improvements.

Comparisons with other boosting algorithms on some public datasets may be possible through prior publications, e.g., (Allwein et al., 2000; Zou et al., 2008). We also hope that our work could be useful as the baseline for future development in multi-class classification.

Acknowledgement

The author thanks the reviewers and area chair for their constructive comments. The author also thanks Trevor Hastie, Jerome Friedman, Tong Zhang, Phil Long, Cun-Hui Zhang, and Giles Hooker.

Ping Li is partially supported by NSF (DMS-0808864), ONR (N000140910911, young investigator award), and the "Beyond Search - Semantic Computing and Internet Economics" Microsoft 2007 Award.

References

Allwein, E. L., Schapire, R. E., & Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *J. of Machine Learning Research*, *1*, 113–141.

Bartlett, P., Freund, Y., Lee, W. S., & Schapire, R. E. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*, 1651–1686.

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. *Conf. on Learning Theory*, 605–619.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Inf. Comput.*, *121*, 256–285.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, *55*, 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H., Hastie, T. J., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, *28*, 337–407.

Lee, Y., Lin, Y., & Wahba, G. (2004). Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *J. of Amer. Stat. Asso.*, *99*, 67–81.

Li, P., Burges, C. J., & Wu, Q. (2008). Mcrank: Learning to rank using classification and gradient boosting. *Neur. Inf. Proc. Sys. Conf.* 897–904.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. *Neur. Inf. Proc. Sys. Conf.* 512-518

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*, 297–336.

Schapire, R. E., & Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, *39*, 135–168.

Tewari, A., & Bartlett, P. L. (2007). On the consistency of multiclass classification methods. *J. of Machine Learning Research*, *8*, 1007–1025.

Zhang, T. (2004). Statistical analysis of some multi-category large margin classification methods. *J. of Machine Learning Research*, *5*, 1225–1251.

Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., Sun, G. (2008). A General Boosting Method and its Application to Learning Ranking Functions for Web Search *Neur. Inf. Proc. Sys. Conf.* 1697–1704.

Zou, H., Zhu, J., & Hastie, T. (2008). New multi-category boosting algorithms based on multicategory fisher-consistent losses. *The Annals of Applied Statistics*, *2*, 1290–1306.