Predicting Multiple Responses with Boosting and Trees

Ping Li$^{1,2}$ and John Abowd$^2$

Department of Statistics & Biostatistics
Department of Computer Science

Rutgers University (1)

and

Department of Statistical Science

Cornell University (2)

November 4, 2013
Motivation: Multi-Label Learning

- Traditional classification methods only deal with a single response (label) for each example. For example, handwritten digit recognition (0, 1, 2, ..., 9).

- In many practical problems, however, one example may involve multiple responses (labels). In scene classification, an image might be both “Mountain” and “Beach”. In Census survey forms, one can choose to declare multiple races, for example, both “American Indian” and “White”.

- Multi-label learning is more challenging. Our working progress demonstrates that it is very promising to use boosting and trees for this type of problems.
History and Progress

- **LogitBoost** (Fridman et. al., 2000) is a well-known work on boosting in statistics. It is also known that the original version had numerical problem.

- **MART** (Fridman, 2001) avoided the numerical problem by using only the first-order information to build the trees (the base learner for boosting). The algorithm is extremely popular in industry.

- **ABC-MART, ABC-LogitBoost** (Ping Li, 2009, 2010) substantially improved MART and logitboost by writing the traditional derivatives of logistic regression in a different way, for the task of multi-class (not multi-label) classification.

- **Robust LogitBoost** (Ping Li, 2010) derived the new tree-split criterion for logitboost and fully solved the numerical issue. (Robust) Logitboost is often more accurate than MART due to the use of second-order information.

- Our idea is to extend multi-class boosting algorithms to multi-label settings, using essentially the same (logistic regression) framework.
Why Tree-Based Boosting Algorithms Are Popular in Industry?

- Scale up easily to large datasets.
- No need to clean / transform / normalized / kernelize the data.
- Few parameters and parameter tuning is simple.
What is Classification?

An Example: USPS Handwritten Zipcode Recognition

Person 1:

1 4 8 5 3

Person 2:

1 4 8 5 3

Person 3:

1 4 8 5 3

The task: Teach the machine to automatically recognize the 10 digits.
Multi-Class Classification

Given a training data set

\[ \{y_i, X_i\}_{i=1}^{N}, \quad X_i \in \mathbb{R}^p, \quad y_i \in \{0, 1, 2, \ldots, K - 1\} \]

the task is to learn a function to predict the class label \( y_i \) from \( X_i \).

- \( K = 2 \) : binary classification
- \( K > 2 \) : multi-class classification

Many important practical problems can be cast as (multi-class) classification. For example, Li, Burges, and Wu, NIPS 2007, *Learning to Ranking Using Multiple Classification and Gradient Boosting*. 
Logistic Regression for Classification

First learn the class probabilities

\[ \hat{p}_k = \Pr \{ y = k | X \}, \quad k = 0, 1, \ldots, K - 1, \]

\[
\sum_{k=0}^{K-1} \hat{p}_k = 1, \quad \text{(only } K - 1 \text{ degrees of freedom}).
\]

Then assign the class label according to

\[ \hat{y} | X = \arg \max_k \hat{p}_k \]
Multinomial Logit Probability Model

\[ p_k = \frac{e^{F_k}}{\sum_{s=0}^{K-1} e^{F_s}} \]

where \( F_k = F_k(x) \) is the function to be learned from the data.

Classical logistic regression:

\[ F(x) = \beta^T x \]

The task is to learn the coefficients \( \beta \).
Flexible additive modeling:

\[ F(x) = F^{(M)}(x) = \sum_{m=1}^{M} \rho_m h(x; a_m), \]

\( h(x; a) \) is a pre-specified function (e.g., trees).

The task is to learn the parameters \( \rho_m \) and \( a_m \).

Both LogitBoost (Friedman et. al, 2000) and MART (Multiple Additive Regression Trees, Friedman 2001) adopted this model.
Seek $F_{i,k}$ to maximize the multinomial likelihood: Suppose $y_i = k$,

$$Lik \propto p_{i,0}^0 \times \ldots \times p_{i,k}^1 \times \ldots \times p_{i,K-1}^0 = p_{i,k}$$

or equivalently, maximizing the log likelihood:

$$\log Lik \propto \log p_{i,k}$$

Or equivalently, minimizing the negative log likelihood loss

$$L_i = - \log p_{i,k}, \quad (y_i = k)$$
The Negative Log-Likelihood Loss

\[ L = \sum_{i=1}^{N} L_i = \sum_{i=1}^{N} \left\{ - \sum_{k=0}^{K-1} r_{i,k} \log p_{i,k} \right\} \]

\[ r_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases} \]
Two Basic Optimization Methods for Maximum Likelihood

1. **Newton’s Method**

   Uses the first and second derivatives of the loss function.
   
   The method in LogitBoost.

2. **Gradient Descent**

   Only uses the first order derivative of the loss function.

   MART used a creative combination of gradient descent and Newton’s method.
Derivatives Used in LogitBoost and MART

The loss function:

\[
L = \sum_{i=1}^{N} \left( \sum_{k=0}^{K-1} L_i = - r_{i,k} \log p_{i,k} \right)
\]

The first derivative:

\[
\frac{\partial L_i}{\partial F_{i,k}} = - (r_{i,k} - p_{i,k})
\]

The second derivative:

\[
\frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,k} (1 - p_{i,k})
\]
The Original LogitBoost Algorithm

1: $F_{i,k} = 0$, $p_{i,k} = \frac{1}{K}$, $k = 0$ to $K - 1$, $i = 1$ to $N$
2: For $m = 1$ to $M$ Do
3: For $k = 0$ to $K - 1$, Do
4: $w_{i,k} = p_{i,k} (1 - p_{i,k})$, $z_{i,k} = \frac{r_{i,k} - p_{i,k}}{p_{i,k} (1 - p_{i,k})}$.
5: Fit the function $f_{i,k}$ by a weighted least-square of $z_{i,k}$ to $x_i$ with weights $w_{i,k}$.
6: $F_{i,k} = F_{i,k} + \nu \frac{K - 1}{K} \left( f_{i,k} - \frac{1}{K} \sum_{k=0}^{K-1} f_{i,k} \right)$
7: End
8: $p_{i,k} = \exp(F_{i,k}) / \sum_{s=0}^{K-1} \exp(F_{i,s})$, $k = 0$ to $K - 1$, $i = 1$ to $N$
9: End
The Original MART Algorithm

1: \( F_{i,k} = 0, \ p_{i,k} = \frac{1}{K}, \ k = 0 \) to \( K - 1, \ i = 1 \) to \( N \)

2: For \( m = 1 \) to \( M \) Do

3: For \( k = 0 \) to \( K - 1 \) Do

4: \( \{R_{j,k,m}\}_{j=1}^{J} = J\)-terminal node regression tree from \( \{r_{i,k} - p_{i,k}, \ x_{i}\}_{i=1}^{N} \)

5: \( \beta_{j,k,m} = \frac{K-1}{K} \frac{\sum_{x_{i} \in R_{j,k,m}} r_{i,k} - p_{i,k}}{\sum_{x_{i} \in R_{j,k,m}} (1 - p_{i,k}) p_{i,k}} \)

6: \( F_{i,k} = F_{i,k} + \nu \sum_{j=1}^{J} \beta_{j,k,m} 1_{x_{i} \in R_{j,k,m}} \)

7: End

8: \( p_{i,k} = \frac{\exp(F_{i,k})}{\sum_{s=0}^{K-1} \exp(F_{i,s})}, \ k = 0 \) to \( K - 1, \ i = 1 \) to \( N \)

9: End
The Numerical Issue in LoigtBoost

4: \[ w_{i,k} = p_{i,k}(1 - p_{i,k}), \quad z_{i,k} = \frac{r_{i,k} - p_{i,k}}{p_{i,k}(1 - p_{i,k})}. \]

5: Fit the function \( f_{i,k} \) by a weighted least-square of \( z_{i,k} \) to \( x_i \) with weights \( w_{i,k} \).

6: \[ F_{i,k} = F_{i,k} + \nu \frac{K - 1}{K} \sum_{k=0}^{K-1} f_{i,k} - \frac{1}{K} \]

The “instability issue”:

When \( p_{i,k} \) is close to 0 or 1, \( z_{i,k} = \frac{r_{i,k} - p_{i,k}}{p_{i,k}(1 - p_{i,k})} \) may approach infinity.

Robust LogitBoost avoids this pointwise thresholding and is essentially free of numerical problems.
Tree-Splitting Using the Second-Order Information

**Feature values:** \( x_i, \ i = 1 \) to \( N \). Assume \( x_1 \leq x_2 \leq \ldots \leq x_N \).

**Weight values:** \( w_i, \ i = 1 \) to \( N \). **Response values:** \( z_i, \ i = 1 \) to \( N \).

We seek the index \( s, 1 \leq s < N \), to maximize the gain of weighted SE:

\[
Gain(s) = SE_T - (SE_L + SE_R)
\]

\[
= \sum_{i=1}^{N} (z_i - \overline{z})^2 w_i - \left[ \sum_{i=1}^{s} (z_i - \overline{z}_L)^2 w_i + \sum_{i=s+1}^{N} (z_i - \overline{z}_R)^2 w_i \right]
\]

where \( \overline{z} = \frac{\sum_{i=1}^{N} z_i w_i}{\sum_{i=1}^{N} w_i} \), \( \overline{z}_L = \frac{\sum_{i=1}^{s} z_i w_i}{\sum_{i=1}^{s} w_i} \), \( \overline{z}_R = \frac{\sum_{i=s+1}^{N} z_i w_i}{\sum_{i=s+1}^{N} w_i} \).
After simplification, we obtain

\[ \text{Gain}(s) = \left[ \frac{s}{\sum_{i=1}^{s} z_i w_i} \right]^2 + \left[ \frac{N}{\sum_{i=s+1}^{N} z_i w_i} \right]^2 - \left[ \frac{N}{\sum_{i=1}^{N} z_i w_i} \right]^2 \]

\[ = \left[ \frac{s}{\sum_{i=1}^{s} r_{i,k} - p_{i,k}} \right]^2 + \left[ \frac{N}{\sum_{i=s+1}^{N} r_{i,k} - p_{i,k}} \right]^2 - \left[ \frac{N}{\sum_{i=1}^{N} r_{i,k} - p_{i,k}} \right]^2 \]

Recall \( w_i = p_{i,k}(1 - p_{i,k}) \), \( z_i = \frac{r_{i,k} - p_{i,k}}{p_{i,k}(1 - p_{i,k})} \).

This procedure is numerically stable.
MART only used the first order information to construct the trees:

\[
MARTGain(s) = \frac{1}{s} \sum_{i=1}^{s} \left( r_{i,k} - p_{i,k} \right)^2 + \frac{1}{N-s} \sum_{i=s+1}^{N} \left( r_{i,k} - p_{i,k} \right)^2 \\
- \frac{1}{N} \sum_{i=1}^{N} \left( r_{i,k} - p_{i,k} \right)^2 .
\]

Which can also be derived by letting weights \( w_{i,k} = 1 \) and response 
\( z_{i,k} = r_{i,k} - p_{i,k} \).

LogitBoost used more information and could be more accurate in many datasets.
Robust LogitBoost

1: \( F_{i,k} = 0, p_{i,k} = \frac{1}{K}, k = 0 \) to \( K - 1 \), \( i = 1 \) to \( N \)
2: For \( m = 1 \) to \( M \) Do
3: \( \text{For } k = 0 \) to \( K - 1 \) Do

4: \( \{ R_{j,k,m} \}_{j=1}^{J} = J \)-terminal node regression tree from \( \{ r_{i,k} - p_{i,k}, x_i \}_{i=1}^{N} \)
   with weights \( p_{i,k}(1 - p_{i,k}) \).

5: \( \beta_{j,k,m} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{j,k,m}} r_{i,k} - p_{i,k}}{\sum_{x_i \in R_{j,k,m}} (1 - p_{i,k}) p_{i,k}} \)

6: \( F_{i,k} = F_{i,k} + \nu \sum_{j=1}^{J} \beta_{j,k,m} 1_{x_i \in R_{j,k,m}} \)

7: End

8: \( p_{i,k} = \exp(F_{i,k}) / \sum_{s=0}^{K-1} \exp(F_{i,s}) \), \( k = 0 \) to \( K - 1 \), \( i = 1 \) to \( N \)

9: End
Experiments on Binary Classification

(Multi-class classification is even more interesting!)

**Data**

**IJCNN1**: 49990 training samples, 91701 test samples

This dataset was used in a competition. LibSVM was the winner.

______________

**Forest100k**: 100000 training samples, 50000 test samples

**Forest521k**: 521012 training samples, 50000 test samples

The two largest datasets from Bordes et al. JMLR 2005, *Fast Kernel Classifiers with Online and Active Learning*
IJCNN1 Test Errors

Test: J = 20  ν = 0.1

Test misclassification error

MART
LibSVM
Robust LogitBoost

Iterations

0 5000 10000
Test misclassification error

Test: J = 20  ν = 0.1

SVM

MART

Robust LogitBoost
Forest521k Test Errors

Test misclassification error vs. Iterations

Test: $J = 20$, $\nu = 0.1$

- SVM
- MART
- Robust LogitBoost
ABC-Boost for Multi-Class Classification

**ABC** = Adaptive Base Class

**ABC-MART** = ABC-Boost + MART

**ABC-LogitBoost** = ABC-Boost + (Robust) LogitBoost

The key to the success of ABC-Boost is the use of “better” derivatives.
Review Components of Logistic Regression

The multinomial logit probability model:

\[ p_k = \frac{e^{F_k}}{\sum_{s=0}^{K-1} e^{F_s}} , \quad p_k = 1 \]

where \( F_k = F_k(x) \) is the function to be learned from the data.

The sum-to-zero constraint:

\[ F_k(x) = 0 \]

is commonly used to obtain a unique solution (only \( K - 1 \) degrees of freedom).
**Why the sum-to-zero constraint?**

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

For identifiability, one should impose a constraint.

One popular choice is to assume \( \sum_{k=0}^{K-1} F_{i,k} = \text{const} \), equivalent to

\[
F_{i,k} = 0.
\]

This is the assumption used in many papers including LogitBoost and MART.
The negative log-Likelihood loss

\[
L = \sum_{i=1}^{N} L_i = \sum_{i=1}^{N} \left( - \sum_{k=0}^{K-1} r_{i,k} \log p_{i,k} \right)
\]

\[
r_{i,k} = \begin{cases} 
1 & \text{if } y_i = k \\
0 & \text{otherwise}
\end{cases}
\]

For \( k = 0 \),
\[
r_{i,k} = 1
\]
Derivatives used in LogitBoost and MART:

\[
\frac{\partial L_i}{\partial F_{i,k}} = - (r_{i,k} - p_{i,k})
\]

\[
\frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,k} (1 - p_{i,k})
\]

which could be derived without imposing any constraints on \( F_k \).
Derivatives Under Sum-to-zero Constraint

The loss function:

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

The probability model and sum-to-zero constraint:

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

Without loss of generality, we assume $k = 0$ is the base class

$$F_{i,0} = - \sum_{i=1}^{K-1} F_{i,k}$$
New derivatives:

\[
\frac{\partial L_i}{\partial F_{i,k}} = (r_{i,0} - p_{i,0}) - (r_{i,k} - p_{i,k}),
\]

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

MART and LogitBoost used:

\[
\frac{\partial L_i}{\partial F_{i,k}} = -(r_{i,k} - p_{i,k}), \quad \frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,k}(1 - p_{i,k}).
\]
### Datasets

- **UCI-Covertype**  
  Total 581012 samples.  
  Two datasets were generated: *Covertype290k, Covertype145k*
- **UCI-Poker**  
  Original 25010 training samples and 1 million test samples.  
  *Poker25kT1, Poker25kT2, Poker525k, Poker275k, Poker150k, Poker100k.*
- **MNIST**  
  Originally 60000 training samples and 10000 test samples.  
  *MNIST10k* swapped the training with test samples.
- **Many variations of MNIST**  
  Original MNIST is a well-known easy problem. ([www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007](http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007)) created a variety of much more difficult datasets by adding various background (correlated) noise, background images, rotations, etc.
- **UCI-Letter**  
  Total 20000 samples.
<table>
<thead>
<tr>
<th>dataset</th>
<th>$K$</th>
<th># training</th>
<th># test</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covertype290k</td>
<td>7</td>
<td>290506</td>
<td>290506</td>
<td>54</td>
</tr>
<tr>
<td>Covertype145k</td>
<td>7</td>
<td>145253</td>
<td>290506</td>
<td>54</td>
</tr>
<tr>
<td>Poker525k</td>
<td>10</td>
<td>525010</td>
<td>500000</td>
<td>25</td>
</tr>
<tr>
<td>Poker275k</td>
<td>10</td>
<td>275010</td>
<td>500000</td>
<td>25</td>
</tr>
<tr>
<td>Poker150k</td>
<td>10</td>
<td>150010</td>
<td>500000</td>
<td>25</td>
</tr>
<tr>
<td>Poker100k</td>
<td>10</td>
<td>100010</td>
<td>500000</td>
<td>25</td>
</tr>
<tr>
<td>Poker25kT1</td>
<td>10</td>
<td>25010</td>
<td>500000</td>
<td>25</td>
</tr>
<tr>
<td>Poker25kT2</td>
<td>10</td>
<td>25010</td>
<td>500000</td>
<td>25</td>
</tr>
<tr>
<td>Mnist10k</td>
<td>10</td>
<td>10000</td>
<td>60000</td>
<td>784</td>
</tr>
<tr>
<td>M-Basic</td>
<td>10</td>
<td>12000</td>
<td>50000</td>
<td>784</td>
</tr>
<tr>
<td>M-Rotate</td>
<td>10</td>
<td>12000</td>
<td>50000</td>
<td>784</td>
</tr>
<tr>
<td>M-Image</td>
<td>10</td>
<td>12000</td>
<td>50000</td>
<td>784</td>
</tr>
<tr>
<td>M-Rand</td>
<td>10</td>
<td>12000</td>
<td>50000</td>
<td>784</td>
</tr>
<tr>
<td>M-RotImg</td>
<td>10</td>
<td>12000</td>
<td>50000</td>
<td>784</td>
</tr>
<tr>
<td>M-Noise1</td>
<td>10</td>
<td>10000</td>
<td>2000</td>
<td>784</td>
</tr>
<tr>
<td>M-Noise2</td>
<td>10</td>
<td>10000</td>
<td>2000</td>
<td>784</td>
</tr>
<tr>
<td>M-Noise3</td>
<td>10</td>
<td>10000</td>
<td>2000</td>
<td>784</td>
</tr>
<tr>
<td>M-Noise4</td>
<td>10</td>
<td>10000</td>
<td>2000</td>
<td>784</td>
</tr>
<tr>
<td>M-Noise5</td>
<td>10</td>
<td>10000</td>
<td>2000</td>
<td>784</td>
</tr>
<tr>
<td>M-Noise6</td>
<td>10</td>
<td>10000</td>
<td>2000</td>
<td>784</td>
</tr>
<tr>
<td>Letter15k</td>
<td>26</td>
<td>15000</td>
<td>5000</td>
<td>16</td>
</tr>
<tr>
<td>Letter4k</td>
<td>26</td>
<td>4000</td>
<td>16000</td>
<td>16</td>
</tr>
<tr>
<td>Letter2k</td>
<td>26</td>
<td>2000</td>
<td>18000</td>
<td>16</td>
</tr>
</tbody>
</table>
## Summary of test mis-classification errors

<table>
<thead>
<tr>
<th>Dataset</th>
<th>mart</th>
<th>abc-mart</th>
<th>logitboost</th>
<th>abc-logitboost</th>
<th>logistic regression</th>
<th># test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covertype290k</td>
<td>11350</td>
<td>10454</td>
<td>10765</td>
<td>9727</td>
<td>80233</td>
<td>290506</td>
</tr>
<tr>
<td>Covertype145k</td>
<td>15767</td>
<td>14665</td>
<td>14928</td>
<td>13986</td>
<td>80314</td>
<td>290506</td>
</tr>
<tr>
<td>Poker525k</td>
<td>7061</td>
<td>2424</td>
<td>2704</td>
<td>1736</td>
<td>248892</td>
<td>500000</td>
</tr>
<tr>
<td>Poker275k</td>
<td>15404</td>
<td>3679</td>
<td>6533</td>
<td>2727</td>
<td>248892</td>
<td>500000</td>
</tr>
<tr>
<td>Poker150k</td>
<td>22289</td>
<td>12340</td>
<td>16163</td>
<td>5104</td>
<td>248892</td>
<td>500000</td>
</tr>
<tr>
<td>Poker100k</td>
<td>27871</td>
<td>21293</td>
<td>25715</td>
<td>13707</td>
<td>248892</td>
<td>500000</td>
</tr>
<tr>
<td>Poker25kT1</td>
<td>43575</td>
<td>34879</td>
<td>46789</td>
<td>37345</td>
<td>250110</td>
<td>500000</td>
</tr>
<tr>
<td>Poker25kT2</td>
<td>42935</td>
<td>34326</td>
<td>46600</td>
<td>36731</td>
<td>249056</td>
<td>500000</td>
</tr>
<tr>
<td>Mnist10k</td>
<td>2815</td>
<td>2440</td>
<td>2381</td>
<td>2102</td>
<td>13950</td>
<td>60000</td>
</tr>
<tr>
<td>M-Basic</td>
<td>2058</td>
<td>1843</td>
<td>1723</td>
<td>1602</td>
<td>10993</td>
<td>50000</td>
</tr>
<tr>
<td>M-Rotate</td>
<td>7674</td>
<td>6634</td>
<td>6813</td>
<td>5959</td>
<td>26584</td>
<td>50000</td>
</tr>
<tr>
<td>M-Image</td>
<td>5821</td>
<td>4727</td>
<td>4703</td>
<td>4268</td>
<td>19353</td>
<td>50000</td>
</tr>
<tr>
<td>M-Rand</td>
<td>6577</td>
<td>5300</td>
<td>5020</td>
<td>4725</td>
<td>18189</td>
<td>50000</td>
</tr>
<tr>
<td>M-RotImg</td>
<td>24912</td>
<td>23072</td>
<td>22962</td>
<td>22343</td>
<td>33216</td>
<td>50000</td>
</tr>
<tr>
<td>M-Noise1</td>
<td>305</td>
<td>245</td>
<td>267</td>
<td>234</td>
<td>935</td>
<td>2000</td>
</tr>
<tr>
<td>M-Noise2</td>
<td>325</td>
<td>262</td>
<td>270</td>
<td>237</td>
<td>940</td>
<td>2000</td>
</tr>
<tr>
<td>M-Noise3</td>
<td>310</td>
<td>264</td>
<td>277</td>
<td>238</td>
<td>954</td>
<td>2000</td>
</tr>
<tr>
<td>M-Noise4</td>
<td>308</td>
<td>243</td>
<td>256</td>
<td>238</td>
<td>933</td>
<td>2000</td>
</tr>
<tr>
<td>M-Noise5</td>
<td>294</td>
<td>244</td>
<td>242</td>
<td>227</td>
<td>867</td>
<td>2000</td>
</tr>
<tr>
<td>M-Noise6</td>
<td>279</td>
<td>224</td>
<td>226</td>
<td>201</td>
<td>788</td>
<td>2000</td>
</tr>
<tr>
<td>Letter15k</td>
<td>155</td>
<td>125</td>
<td>139</td>
<td>109</td>
<td>1130</td>
<td>5000</td>
</tr>
<tr>
<td>Letter4k</td>
<td>1370</td>
<td>1149</td>
<td>1252</td>
<td>1055</td>
<td>3712</td>
<td>16000</td>
</tr>
<tr>
<td>Letter2k</td>
<td>2482</td>
<td>2220</td>
<td>2309</td>
<td>2034</td>
<td>4381</td>
<td>18000</td>
</tr>
</tbody>
</table>
Comparisons with SVM and Deep Learning

Datasets: M-Noise1 to M-Noise6

Results on SVM, Neural Nets, and Deep Learning are from

www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/
DeepVsShallowComparisonICML2007
Comparisons with SVM and Deep Learning

Datasets: M-Noise1 to M-Noise6
## More Comparisons with SVM and Deep Learning

<table>
<thead>
<tr>
<th></th>
<th>M-Basic</th>
<th>M-Rotate</th>
<th>M-Image</th>
<th>M-Rand</th>
<th>M-RotImg</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-RBF</td>
<td>3.05%</td>
<td>11.11%</td>
<td>22.61%</td>
<td>14.58%</td>
<td>55.18%</td>
</tr>
<tr>
<td>SVM-POLY</td>
<td>3.69%</td>
<td>15.42%</td>
<td>24.01%</td>
<td>16.62%</td>
<td>56.41%</td>
</tr>
<tr>
<td>NNET</td>
<td>4.69%</td>
<td>18.11%</td>
<td>27.41%</td>
<td>20.04%</td>
<td>62.16%</td>
</tr>
<tr>
<td>DBN-3</td>
<td>3.11%</td>
<td>10.30%</td>
<td>16.31%</td>
<td>6.73%</td>
<td>47.39%</td>
</tr>
<tr>
<td>SAA-3</td>
<td>3.46%</td>
<td>10.30%</td>
<td>23.00%</td>
<td>11.28%</td>
<td>51.93%</td>
</tr>
<tr>
<td>DBN-1</td>
<td>3.94%</td>
<td>14.69%</td>
<td>16.15%</td>
<td>9.80%</td>
<td>52.21%</td>
</tr>
<tr>
<td><strong>mart</strong></td>
<td>4.12%</td>
<td>15.35%</td>
<td>11.64%</td>
<td>13.15%</td>
<td>49.82%</td>
</tr>
<tr>
<td>abc-mart</td>
<td>3.69%</td>
<td>13.27%</td>
<td>9.45%</td>
<td>10.60%</td>
<td>46.14%</td>
</tr>
<tr>
<td>logitboost</td>
<td>3.45%</td>
<td>13.63%</td>
<td>9.41%</td>
<td>10.04%</td>
<td>45.92%</td>
</tr>
<tr>
<td>abc-logitboost</td>
<td>3.20%</td>
<td>11.92%</td>
<td>8.54%</td>
<td>9.45%</td>
<td>44.69%</td>
</tr>
</tbody>
</table>
Extending Multi-Class to Multi-Label Learning

**Multi-Class Learning:** Suppose \( y_i = k \),

\[
Lik \propto p_{i,0}^0 \times \cdots \times p_{i,k}^1 \times \cdots \times p_{i,K-1}^0 = p_{i,k}
\]

**Multi-Label Learning:** Suppose \( y_i \in S_i = \{0, k\} \),

\[
Lik \propto p_{i,0}^1 \times \cdots \times p_{i,k}^1 \times \cdots \times p_{i,K-1}^0 = p_{i,0}p_{i,k}
\]

There are actually more than one ways to determine the weights. For example, we can choose the following loss function:

\[
L = \sum_{i=1}^{N} L_i = \sum_{i=1}^{N} \left( \sum_{k=0}^{K-1} w_{i,k} \log p_{i,k} \right), \quad w_{i,k} = \begin{cases} 
\frac{1}{|S_i|} & \text{if } y_i \in S_i \\
0 & \text{otherwise}
\end{cases}
\]
Combining Multi-Label Model with Boosting and Trees

- We need to modify the existing boosting algorithms (MART, LogitBoost, ABC-MART, ABC-LogitBoost) to incorporate the new models.

- For each example, the algorithm will again output a vector of class probabilities. We need to a criterion to truncate the list to assign class labels.

- We need a good evaluation criterion to assess the quality of multi-label learning.
Evaluation Criteria

Using our model and boosting, we learn the set of class probabilities for each example and sort them in descending order:

\[ \hat{p}_{i,(0)} \geq \hat{p}_{i,(1)} \geq \ldots \geq \hat{p}_{i,(K-1)} \]

We consider three criteria:

- **One-error**: How many times the top-ranked label is not in the true labels.
- **Coverage**: How far one needs, on average, to go down the list of labels in order to cover all the ground truth labels.
- **Precision**: A more comprehensive ranking measure borrowed from information retrieval (IR) literature.
Experiments and Comparisons

We implemented our method with MART (other implementations are forthcoming).

We compared our result with an existing publication on the same dataset.

Our method with boosting and trees (red curves) is substantially better than published results (dashed horizontal line).

Our precision is about 87% but the other paper did not report.
Ongoing Work

- Test our (and others’) multi-label algorithms on Census data.
- Experiment with various multi-label probability models.
- Implement (Robust) LogitBoost for multi-label learning
- Implement ABC-MART and ABC-LogitBoost for multi-label learning
References

- Ping Li et. al., Mcrank: Learning to rank using multiple classification and gradient boosting, NIPS 2007
- Ping Li, ABC-boost: adaptive base class boost for multi-class classification, ICML 2009
- Ping Li, Robust logitboost and adaptive base class (abc) logitboost, UAI 2010
- Ping Li, Fast abc-boost for multi-class classification, arXiv:1006.5051, 2010
- Ping Li, Learning to Rank Using Robust LogitBoost, Yahoo! Learning to Rank Grand Challenge, 2010