Boosting Models for Edit, Imputation and Prediction of Multiple Response Outcomes

Ping Li[1] and John M. Abowd[2]

[1] Department of Statistics & Biostatistics and Department of Computer Science, Rutgers University
[2] Economics Department and Department of Statistical Sciences, Cornell University

February 5, 2014

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.

An Example: Predicting (Possibly Multiple) Race and Ethnicity Responses

Application Multi-label Race Outcomes: American Community Survey and 2010 Census)
Task: Build a Model for the 7 Basic Race Responses

1. White
2. Black or African American
3. American Indian
4. Alaska Native
5. Asian
6. Native Hawaiian and Other Pacific Islander
7. Other

Other Multi-label Outcomes of Interest

Application Multi-label Ethnicity Outcomes: American Community Survey and 2010 Census)
Task: Build a Model for the 100+ Detailed Ethnicity Responses

- Any ethnicity alone
- All combinations
- Interactions of Hispanic ethnicity and write-in Hispanic race
- Similar issues with other (smaller) ethnic groups
- This matters because there is extensive editing of these variables
- Current edit and imputation procedures don't use model based methods
- There is scientific interest in the determinants of multi-racial and multi-ethnic declarations

Other Multi-label Outcomes of Interest

Application Detailed Work-history Edits in the LEHD Infrastructure)
Task: Build a Model for Imputing the Missing Establishment in Multi-establishment Employers in the LEHD

- Single-unit employers have only one establishment
- Multi-unit employers have two or more establishments (in the same state)
- Training data show that the same person can be employed in several establishments
- Current imputation model (U2W) doesn't allow this
- Thousands of potential conditioning variables, most continuous
- Multi-label boosted logit is a candidate to replace the current logistic regression models

Preparing the ACS Data

Individual Level Data

- Used the 2005-2007 Public Use Micro Sample archived by CISER (http://ciser.cornell.edu/ASPs/search_athena.asp?IDTITLE=2532)
- Extracted 8.8 million individual records (all states and DC)
- Constructed the 7-category multi-label race variable from the three recoded variables on the PUMS (rac1p, rac2p, rac3p)
- Extracted all variables for use in the modeling
- Continuous variables used as coded (e.g., total person earnings)
- Discrete variables converted to indicators (e.g., sex)
- Unweighted proportions: about 3% declared more than 1 race, 0.7% declared more than 2 races, and 0.1% declared more than 3 races

Preparing the ACS Data

Household Level Data

- Appended all variables from the household record to the individual record, using same coding procedures
- Computed household averages, excluding individual, of other individuals in household
- Intend to do the same thing for detailed geographical variables in the confidential ACS
- About 160 variables available for boosted logit models

Preparing the Training and Testing Samples

- Training sample is a 3% simple random sample of the individual records
- Testing sample is a non-overlapping 3% simple random sample of the individual records
- Training and testing samples are 265,000 records each

A Statistical Model for Multiple Outcomes

Training data: N observations, $i = 1$ to N. X_i is p-dim vector of variables. S_i is a set of labels.

$$\{S_i,\ X_i\}_{i=1}^{N}, \qquad X_i \in \mathbb{R}^p, \qquad S_i \subseteq \{0, 1, 2, ..., K-1\}$$

In particular, if $|S_i| = 1, \forall i$, then it is the usual multi (K)-class classification.

Class Probabilities: $y_i \in S_i$.

$$\hat{p}_{i,k} = \mathbf{Pr}\{y_i = k | X_i\}, \qquad k = 0, 1, ..., K-1,$$

$$\sum_{k=0}^{K-1} \hat{p}_{i,k} = 1, \qquad \text{(only } K-1 \text{ degrees of freedom)}.$$

Extending Multi-class to Multi-label Learning

Multi-class Learning: Suppose $y_i = k$,

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

Multi-label Learning (Option 1): Suppose $y_i \in S_i = \{0, k\}$, $|S_i| = 2$

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

Multi-label Learning (Option 2): Suppose $y_i \in S_i = \{0, k\}$, $|S_i| = 2$

$$Lik \propto p_{i,0}^{1/2} \times ... \times {p_{i,k}}^{1/2} \times ... \times p_{i,K-1}^0 = p_{i,0}^{1/2} p_{i,k}^{1/2}$$

Which option to use? We choose option 2 in this presentation.

An Interpretation of Option 2: Suppose each observation is an image of m pixels. We can view one original observation as m observations. When $S_i = \{0, k\}$, we can write the "joint likelihood" as

$$Lik \propto {p_{i,0}}^{m/2} \times ... \times {p_{i,k}}^{m/2} \times ... \times p_{i,K-1}^{0} = {p_{i,0}}^{m/2} {p_{i,k}}^{m/2}$$

Now if we assume all images have the same number (i.e., m) of pixels, we can basically remove m in the likelihood because it is the same for all i.

For future study: This interpretation provides an the intuition that we might be able to improve the multi-label learning system by assigning fractional labels. For example, instead of only assigning $S_i = \{0, k\}$, we can add the weights $S_i = \{0(1/3), k(2/3)\}$, to indicate that users are certain 1/3 of this observation should belong to class 0 and 2/3 to class k.

The Joint Likelihood or Loss Function

Based on the likelihood (option 2), we seek to minimize the following joint loss function:

$$L = \sum_{i=1}^{N} L_i = \sum_{i=1}^{N} \left\{ - \sum_{k=0}^{K-1} w_{i,k} \log p_{i,k} \right\}, \quad w_{i,k} = \begin{cases} 1/|S_i| & \text{if } y_i \in S_i \\ 0 & \text{otherwise} \end{cases}$$

This is a generalization of the classical multi-class logistic regression.
At this point, we resort to the usual logit probability model to model.

Multinomial Logit Probability Model

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

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

Classical logistic regression:

$$F(\mathbf{x}) = \beta^{\mathsf{T}}\mathbf{x}$$

The task is to learn the coefficients β.

Flexible additive modeling:

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

$h(\mathbf{x}; \mathbf{a})$ is a pre-specified function (e.g., trees).
The task is to learn the parameters ρ_m and $\mathbf{a}_m$.

Both **LogitBoost** (Friedman et. al, 2000) and **MART** (Multiple Additive Regression Trees, Friedman 2001) adopted this model.

Advantages of Tree Algorithms

- A natural (although in a sense crude) way to model nonlinearity of the data.
- A natural way to model high-order interactions.
- Simple efficient (i.e., easy to parallelize) algorithm.
- Interpretable model.
- No need to clean/scale/noramlize the data. This is crucial for industrial large-scale applications.
- The accuracy of a single tree is in general not too high, but trees integrated with boosting can lead to extremely accurate models.

Our Prior Experience with Boosting and Tree Algorithms

- **Ping Li et. al.**, *Mcrank: Learning to rank using multiple classification and gradient boosting*, **NIPS 2007**
- **Ping Li**, *Learning to Rank Using Robust LogitBoost*, **Yahoo! Learning to Rank Grand Challenge, 2010**
- **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**

The MART Boosting Algorithm for Multi-label Learning

1: $F_{i,k} = 0, \;\; p_{i,k} = \frac{1}{K}, \quad k = 0 \text{ to } K-1, \;\; i = 1 \text{ to } N$

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 $\{w_{i,k} - p_{i,k}, \;\; \mathbf{x}_i\}_{i=1}^{N}$

5: $\beta_{j,k,m} = \frac{K-1}{K} \frac{\sum_{\mathbf{x}_i \in R_{j,k,m}} w_{i,k} - p_{i,k}}{\sum_{\mathbf{x}_i \in R_{j,k,m}} (1-p_{i,k}) p_{i,k}}$

6: $F_{i,k} = F_{i,k} + \nu \sum_{j=1}^{J} \beta_{j,k,m} 1_{\mathbf{x}_i \in R_{j,k,m}}$

7: End

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

9: End

Important Parameters in the MART Algorithm

- J: number of leaves in one tree. $J = 20$ (for relatively large dataset with many variables) or $J = 10$ (for relatively small datasets) are common.
- M: number of boosting iterations. Total number of trees would be $J \times M$
- ν: shrinkage parameter. $\nu = 0.05$ is usually a good choice.

Evaluation Metrics for Multi-label Learning

After we have learned the model, we can estimate the class probabilities. Given a new test data point i, we can sort the potential class labels according to the magnitudes of the predicted class probabilities. Ideally, we hope the top $|S_i|$ predicted labels are exactly the same as the true class labels if the data point i has label set S_i. We choose 3 measures to evaluate the performance:

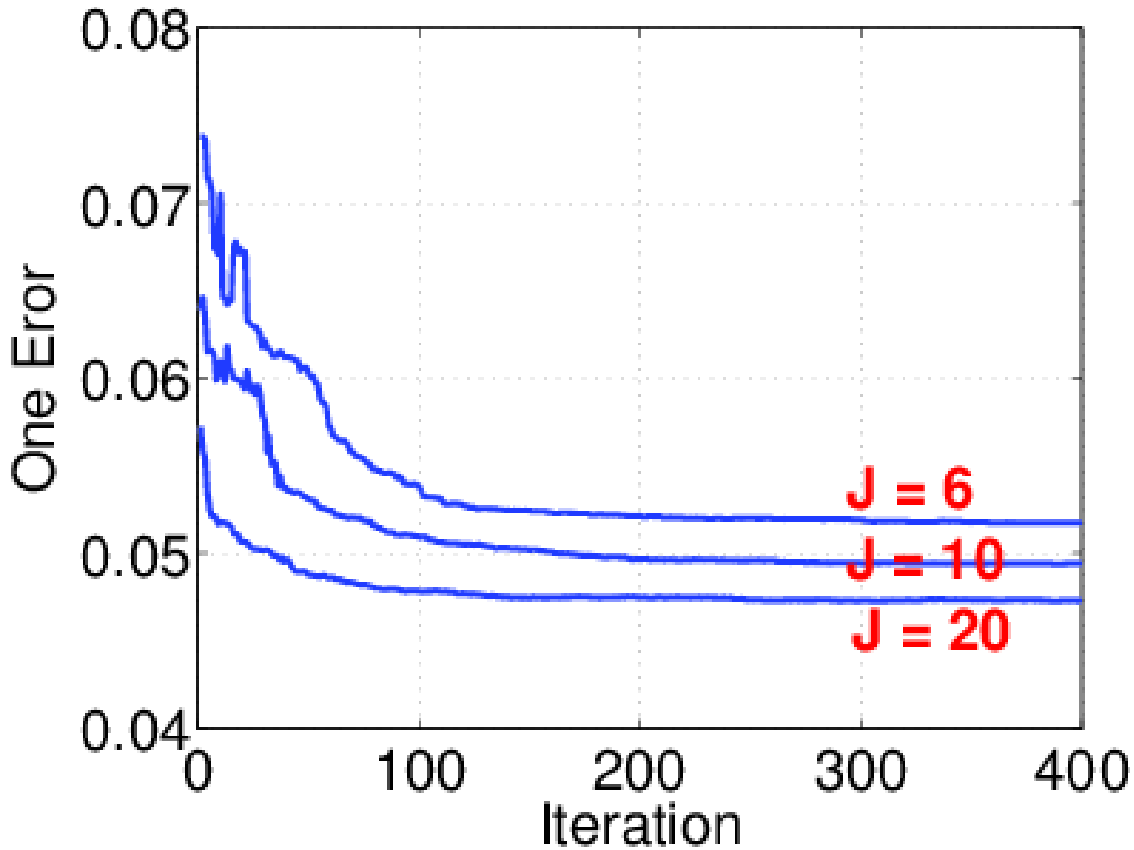
- **One-error**: An error if the top ranked label is not one of the true labels in S_i.
- **Coverage**: How many additional labels we have to go down the sorted list of predicted class labels. For example, if $|S_i| = 3$ and we have to look at the first 4 labels in order include the three true labels, then the coverage is 1.
- **Average Precision**: A ranking-based measure, higher is better.

Experiment

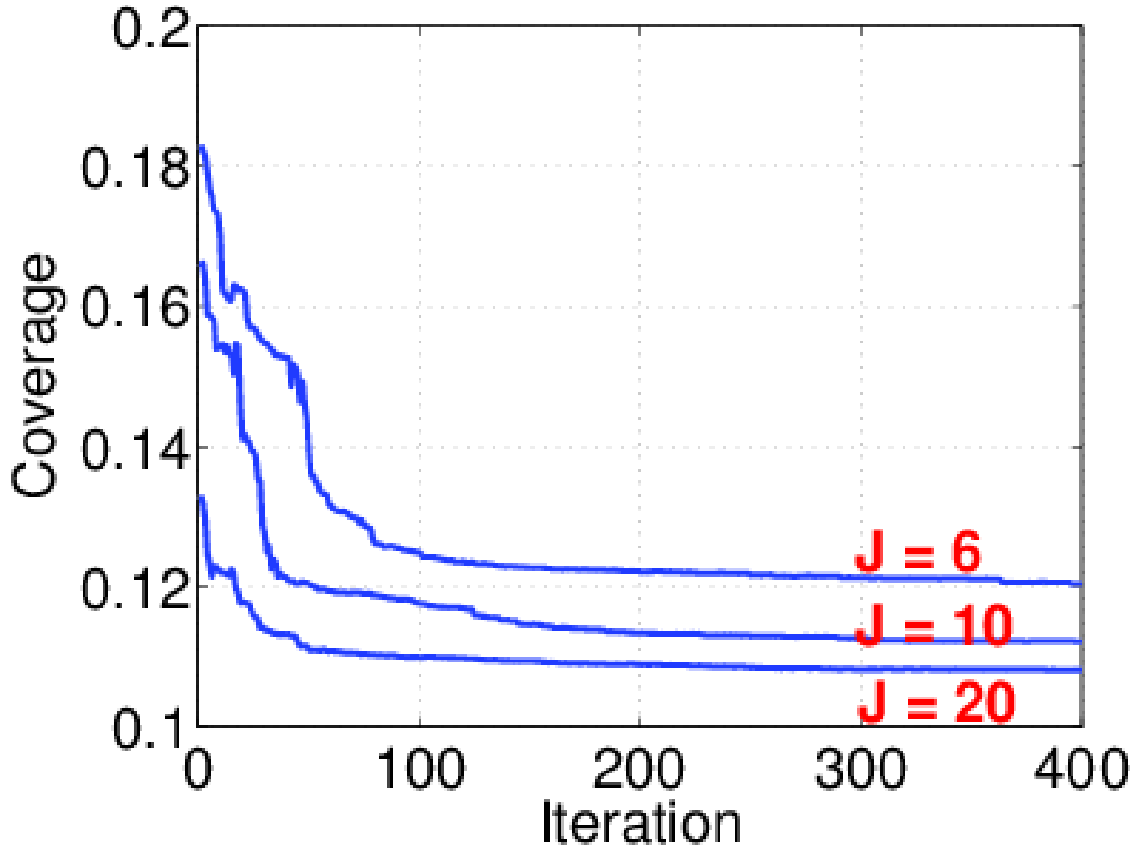
Data for Testing and Training

- The experiments were trained using the 265,000 observation simple random sample of the 2005-2007 American Community Survey Public Use Micro Sample
- The experiments were tested using the non-overlapping 265,000 simple random sample of the same data
- 160 variables from the individual, household, and "other-individuals in household" as explained above

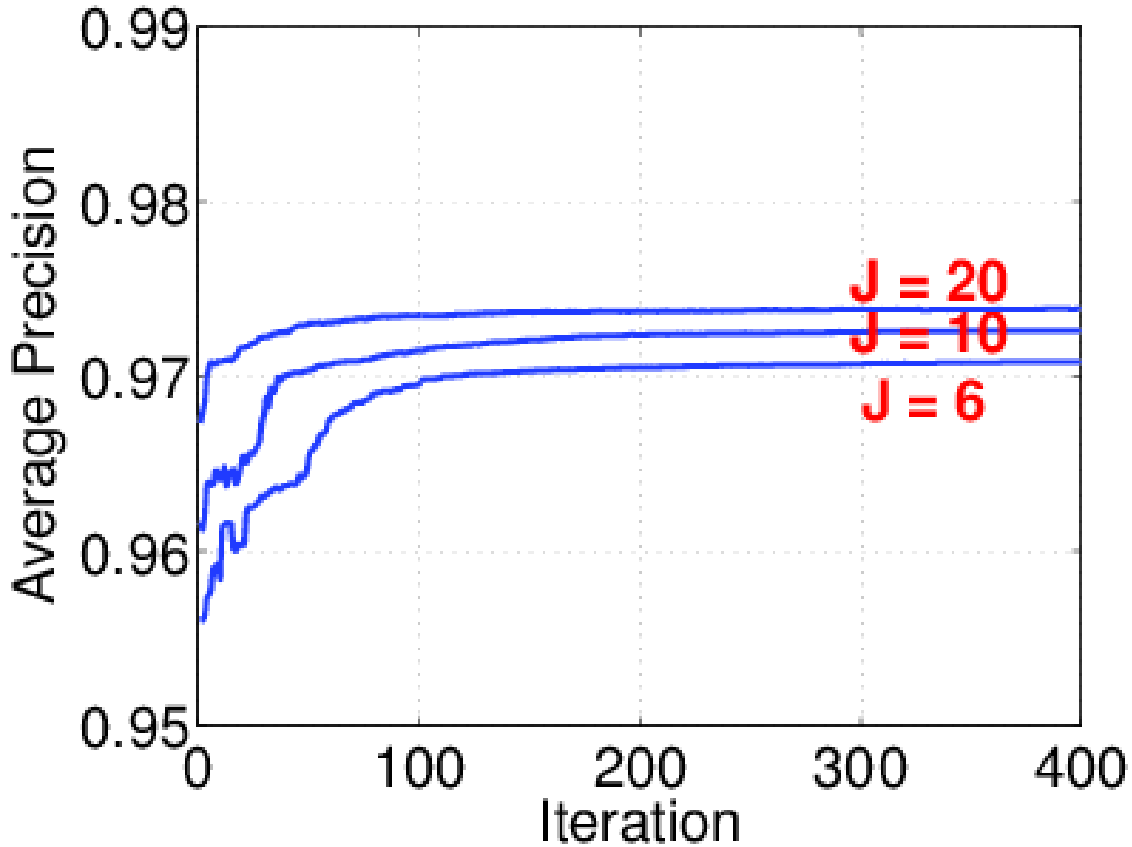
One-error: Lower is better



Coverage: Lower is better



Average Precision: Higher is better



Discussion about the Experiment

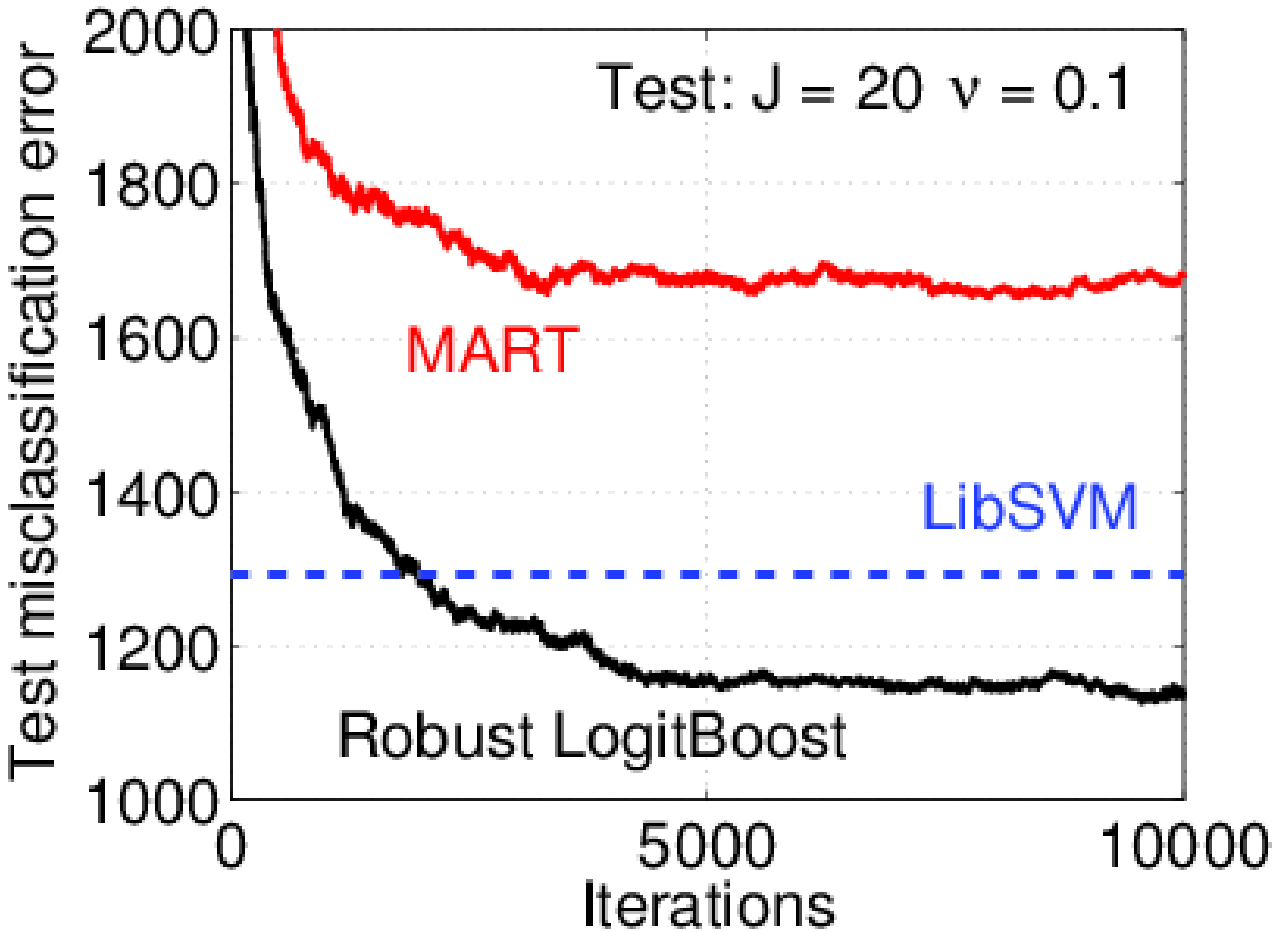
- The initial results are very encouraging. Using just the raw data with no cleaning or normalization, the errors are already very low.
- The work is still preliminary. There are numerous interesting and practically important issues to be studied. For example
  - Ideally, we should provide a "cut-off" (e.g., based on a probability threshold of 0.1) instead of the ranked list of predicted labels.
  - Without using parallelization, it is difficult to train a dataset with (e.g.,) > 5 million observations especially if we need to use a richer (higher dimensional) set of variables/features.
  - We can take advantage of the most recent development of boosting algorithms to further improve the results.

Two Recent Developments of Boosting

- **Robust Logitboost**. People used to believe that the well-known logitboost algorithm had serious numerical problems. The MART algorithm was developed partly to overcome those issue by using less information to build trees. It turns out that numerical issue of logitboost did not really exist.
- **Adaptive Base Class (ABC) Boost**. A new invention which considerably improved the accuracy of multi-class classification. Because multi-class and multi-label problems are closely related, it is expected that ABC-Boost will be useful for multi-label classification.

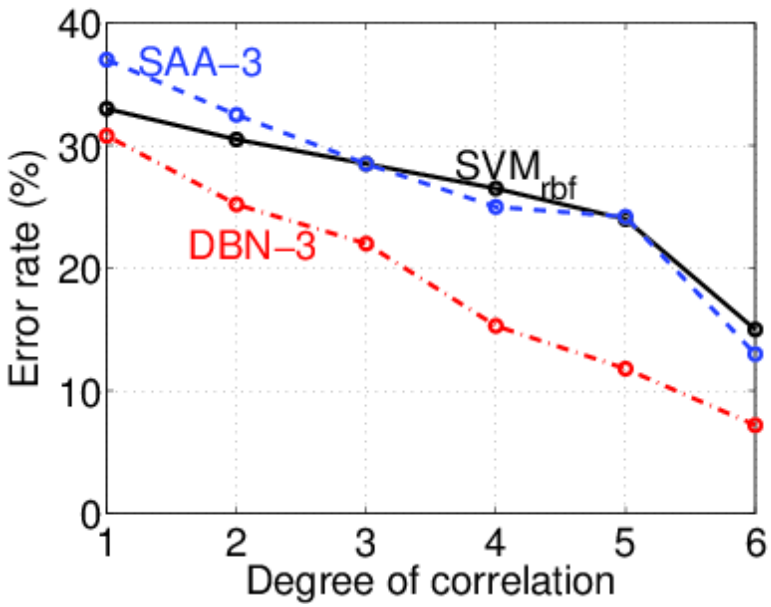
Example of Performance of Robust Logitboost

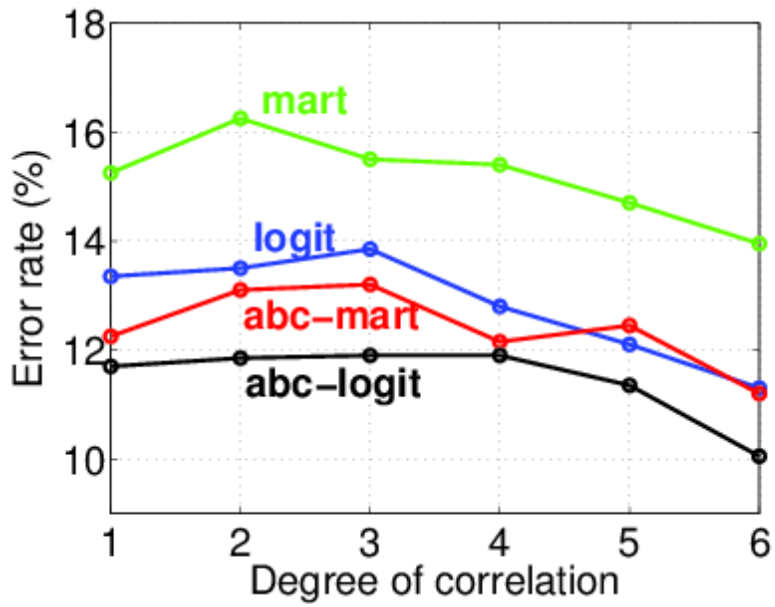
IJCNN1 data was used in a competition. LIBSVM was the winner.



ABC-Boost Compared with SVM and Deep Learning

The own datasets used by deep learning community. Compared with the results of deep learning and SVM (Left panel), ABC-Boost (Left panel: abc-mart and abc-logit) can be substantially more accurate





Conclusion

- Still a work in progress
- Application to the confidential ACS data will determine the feasibility and quality of this model for large-scale edit and imputation