International Journal of Computer Theory and Engineering, Vol.2, No.6, December, 2010
1793-8201

Comprehensive Evolution and Evaluation of Boosting

Amit P Ganatra*, Yogesh P Kosta**[1]

***Abstract*— At present, an active research topic is the use of ensembles of classifiers. They are obtained by generating and combining base classifiers, constructed using other machine learning methods. The target of these ensembles is to increase the predictive accuracy with respect to the base classifiers. One of the most popular methods for creating ensembles is boosting, a family of methods, of which AdaBoost is the most prominent member. Boosting is a general approach for improving classifier performances. Boosting is a well established method in the machine learning community for improving the performance of any learning algorithm. It is a method to combine weak classifiers produced by a weak learner to a strong classifier. Boosting refers to the general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb. Boosting Methods combine many weak classifiers to produce a committee. It resembles Bagging and other committee based methods. Many weak classifiers are combined to produce a powerful committee. Sequentially apply weak classifiers to modified versions of data. Predictions of these classifiers are combined to produce a powerful classifier. This paper contains comprehensive evolution of Boosting and evaluation of Boosting on various criteria (parameters) with Bagging.**

***Index Terms*— Ensemble, machine learning, predictive accuracy, classifiers, Boosting.**

I. Introduction

All ensemble systems consist of two key components. First, a strategy is needed to build an ensemble that is as diverse as possible. Some of the more popular ones, such are bagging, boosting, AdaBoost, stacked generalization, and mixture of experts. A second strategy is needed to combine the outputs of individual classifiers that make up the ensemble in such a way that the correct decisions are amplified, and incorrect ones are cancelled out [6][29].

Ensemble systems become naturally useful, such as incremental learning, data fusion, feature selection and error correcting output codes. Whereas there is no single ensemble generation algorithm or combination rule that is universally better than others. In the absence of any other prior information, the best ones are the simplest and least complicated ones that can learn the underlying data distribution [6][29].

[1]* Associate Professor & Head, CE-IT, CITC (amitganu@yahoo.com)
**Dean, Faculty of Technology & Engineering (ypkosta@yahoo.com) (IEEE Member) (SCPM, Stanford University)
Charotar University of Science Technology (CHARUSAT), Education Campus, Changa – 388421, Ta – Petlad, Dist – Anand, Gujarat (INDIA)

Boosting combines many weak classifiers to produce a powerful committee. It comes with a set of theoretical guarantee (e.g., training error, test error). It performs well on many tasks. In the early iterations, boosting is primary a bias-reducing method. In later iterations, it appears to be primarily a variance-reducing method. Why called boosting: it always attempts to boost the accuracy of any given learning algorithm, whatever weak it is. Motivation: combines the outputs of many weak classifiers to produce a powerful "committee". Strategy: fits model with a set of elementary "basis" functions in an additive way. Generates a hypothesis whose error on the training set is small by combining many hypotheses with large errors. Reality training data exhibit different degrees of hardness. Each learning algorithm has unstable behavior, i.e., appears sensitive to changes in training data. Reduces both variance and bias, while bagging can only significantly reduces variance. Records that are wrongly classified will have their weights increased. Records that are classified correctly will have their weights decreased.

TABLE 1

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

In above table, Example 4 is hard to classify. Its weight is increased; therefore it is more likely to be chosen again in subsequent rounds. Sequential production of classifiers. Each classifier is dependent on the previous one, and focuses on the previous one's errors. Examples that are incorrectly predicted in previous classifiers are chosen more often or weighted more heavily.

The actual performance of boosting depends on the data and the base learner. Boosting seems to be especially susceptible to noise. When the number of outliners is very large, the emphasis placed on the hard examples can hurt the performance. Make examples currently misclassified more important (or less, in some cases). Simple, boosted classifiers can reject many of the negative samples while detecting all positive instances. Series of such simple classifiers can achieve good detection performance while eliminating the need for further processing of negative samples. Subsequent classifiers are trained only on examples which pass through all the previous classifiers. "Boosting": convert a weak learning algorithm into a strong one. Main idea here is to

combine many weak classifiers to produce a powerful committee [33].

Intuition

At every stage, a weak learner is trained with the data. The output of the weak learner (a weak classifier) is added to the current strong classifier, with some strength (proportional to how accurate the weak classifier is). The data is reweighted: examples that the current strong classifier gets wrong are "boosted" in importance.

The intuitive idea

Altering the distribution over the domain in a way that increases the probability of the "harder" parts of the space, thus forcing the weak learner to generate new hypotheses that make fewer mistakes on these parts.

Two approaches

Select examples according to error in previous classifier (more representatives of misclassified cases are selected) – more common. Weigh errors of the misclassified cases higher (all cases are incorporated, but weights are different) – does not work for some algorithms.Freund and Schapire (1997), Breiman (1998). Data adaptively resampled. Predictor aggregation done by weighted voting. Easy to implement without additional information. For Boosting: no overfitting. Boosting (Schapire 1989)[10].

1. Randomly select n1 < n samples from D without replacement to obtain D1
    a. Train weak learner C1
2. Select n2 < n samples from D with half of the samples misclassified by C1 to obtain D2
    a. Train weak learner C2
3. Select all samples from D that C1 and C2 disagree on
    a. Train weak learner C3
4. Final classifier is vote of weak learners.

II. Boosting Process

1. Given a set of labeled training examples
2. On each round
3. The booster devises a distribution (importance) over the example set
4. The booster requests a weak hypothesis (rule-of-thumb) with low error
5. After T rounds, the booster combine the weak hypothesis into a single prediction rule.
6. This is the basic setting for boosting algorithm.

Why named it AdaBoost?

1. The example weight adapts
    a. Easy examples get lower weights
    b. Hard examples get higher weights
2. A learner put more emphasis on hard examples
3. Final classifier is weighted vote of T weak learner
    a. Importance of a learner:
1. A method to combine weak classifiers produced by a weak learner to a strong classifier
    a. Weak classifier: A classifier better than random guessing
    b. Weak learner: A learning algorithm produces weak classifiers

What's so good about AdaBoost?

1. Improves classification accuracy
2. Can be used with many different classifiers
3. Commonly used in many areas
4. Simple to implement
5. Not prone to overfitting
6. Start with RealAB (easy to implement)
7. Possibly try a variant, such as FloatBoost, WeightBoost, or LogitBoost
8. Try varying the complexity of the weak learner
9. Try forming the weak learner to minimize Z (or some similar goal)

Practical Advantages of AdaBoost

1. It is fast, simple and easy to program
2. It requires no parameters to tune (except T)
3. It is flexible — can combine with any learning algorithm
4. No prior knowledge needed about weak learner
5. Provably effective, provided can consistently find rough rules of thumb
6. Shift in mind set — goal now is merely to find classifiers barely better than random guessing
7. versatile
8. Can use with data that is textual, numeric, discrete, etc.
9. Has been extended to learning problems well beyond binary classification
10. Very simple to implement
11. General learning scheme - can be used for various learning tasks
12. Feature selection on very large sets of features
13. Good generalization
14. Seems not to overfit in practice (probably due to margin maximization) [6][21][22].

Disadvantages

1. Performance of AdaBoost depends on data and weak learner
2. Consistent with theory, AdaBoost can fail if
    a. Weak classifiers too complex! overfitting
    b. Weak classifiers too weak (t! 0 too quickly)! under fitting! Low margins! overfitting
5. Empirically, AdaBoost seems especially susceptible to uniform noise
6. Suboptimal solution (greedy learning)

Interesting properties

1. Ada Boost is capable reducing both bias (e.g. stumps) and variance (e.g. trees) of the weak classifiers
2. Ada Boost has good generalization properties (maximizes margin)
3. Ada Boost output converges to the logarithm of likelihood ratio
4. Ada Boost can be seen as a feature selector with a principled strategy (minimization of upper bound on empirical error)
5. Ada Boost is close to sequential decision making (it produces a sequence of gradually more complex classifiers)

III. Types of Boosting Algorithms

i) Adaboost - Adaptive Boosting

1. Instead of sampling, re-weight

a. Previous weak learner has only 50% accuracy over new distribution
2. Can be used to learn weak classifiers
3. Final classification based on weighted vote of weak classifiers
4. Learner = Hypothesis = Classifier
5. Weak Learner: < 50% error over any distribution
6. Strong Classifier: threshold linear combination of weak learner outputs.

ii) Brown Boost

BrownBoost is a boosting algorithm that may be robust to noisy datasets. BrownBoost is an adaptive version of the boost by majority algorithm. As is true for all boosting algorithms, BrownBoost is used in conjunction with other machine learning methods.

Motivation

AdaBoost performs well on a variety of datasets; however, it can be shown that AdaBoost does not perform well on noisy data sets.[2] This is a result of AdaBoost's focus on examples that are repeatedly misclassified. In contrast, BrownBoost effectively "gives up" on examples that are repeatedly misclassified. The core assumption of BrownBoost is that noisy examples will be repeatedly mislabeled by the weak hypotheses and correctly non-noisy examples will be correctly labeled frequently enough to not be "given up on." Thus only noisy examples will be "given up on," whereas non-noisy examples will form contribute to the final classifier. In turn, if the final classifier is learned from the non-noisy examples, the generalization error of the final classifier may be much better than if learned from noisy and non-noisy examples.

The user of the algorithm can set the amount of error to be tolerated in the training set. Thus, if the training set is noisy (say 10% of all examples are assumed to be mislabeled), the booster can be told to accept a 10% error rate. Since the noisy examples may be ignored, only the true examples will contribute to the learning process. BrownBoost uses a non-convex potential loss function, thus it does not fit into the AnyBoost framework. The only parameter of BrownBoost (*c* in the algorithm) is the "time" the algorithm runs.

iii) Logit Boost

Two popular methods for classification are linear logistic regression and tree induction, which have somewhat complementary advantages and disadvantages. The former fits a simple (linear) model to the data, and the process of model fitting is quite stable, resulting in low variance but potentially high bias. The latter, on the other hand, exhibits low bias but often high variance: it searches a less restricted space of models, allowing it to capture nonlinear patterns in the data, but making it less stable and prone to overfitting.

It is a natural idea to try and combine these two methods into learners that rely on simple regression models if only little and/or noisy data is available and add a more complex tree structure if there is enough data to warrant such structure. For the case of predicting a numeric variable, this has lead to model trees, which are decision trees with linear regression models at the leaves. Although it is possible to use model trees for classification tasks by transforming the classification problem into a regression task by binarizing the class, this approach produces several trees (one per class) and thus makes the final model harder to interpret [3].

iv) LPBoost

Linear Programming Boosting (LPBoost) is a supervised classifier from the Boosting family of classifiers. LPBoost maximizes a margin between training samples of different classes and hence also belongs to the class of margin-maximizing supervised classification algorithms. Consider a classification function $f : x \to \{-1,1\}$ which classifies samples from a space into one of two classes, labeled 1 and -1, respectively. LPBoost is an algorithm to learn such a classification function given a set of training examples with known class labels. LPBoost is a machine learning technique and especially suited for applications of joint classification and feature selection in structured domains. [7]

LPBoost overview

As in all Boosting classifiers, the final classification function is of the form

$$f(x) = \sum_{j=1}^{J} \alpha_j h_j(x), \tag{1}$$

Where α_j are non-negative weightings for weak classifiers $h_j : x \to \{-1,1\}$. Each individual weak classifier h_j may be just a little bit better than random, but the resulting linear combination of many weak classifiers can perform very well.

LPBoost constructs f by starting with an empty set of weak classifiers. Iteratively, a single weak classifier to add to the set of considered weak classifiers are selected, added and all the weights α for the current set of weak classifiers are adjusted. This is repeated until no weak classifiers to add remain. The property that all classifier weights are adjusted in each iteration is known as totally-corrective property. Early Boosting methods, such as AdaBoost do not have this property and converge slower.

Taking a 1-norm of the slack variables in the margin maximization framework and optimizing the 1- norm of coefficients leads to a linear programme. This can be solved using a simplex-based column generation approach, resulting the in the algorithm LPBoost. LPBoost can be proved to converge in a finite number of iterations to a globally optimal solution within the hypothesis space. In the dual form the constraints are the weak learners. The algorithm proceeds by adding a weak learner, and checking if the linear programme is solved.

If not then the weak learner is found that violates the constraints the most. This process is repeated until the linear programme constraints are not violated, which leads to the global optimum solution. Although LPBoost iterations are typically slower than AdaBoost, the algorithm converges much more quickly.

v) MADABoost : A Modification of AdaBoost

Due to the problems in AdaBoost algorithm, the new variant of AdaBoost is used. These problems are: (1) AdaBoost cannot be used in the boosting by filtering framework, and (2) AdaBoost does not seem to be noise

resistant. In order to solve them, a new boosting algorithm MadaBoost is used by modifying the weighting system of AdaBoost. New boosting algorithm can be casted in the statistical query learning model and thus, it is robust to random classification noise.

Introduction

In the last decade, boosting techniques have been received a great deal of attention from the machine learning and computational learning communities. New boosting algorithm MadaBoost that mends some of the problems that have been detected in the so far most successful boosting algorithm. These problems are: (1) AdaBoost cannot be used in the boosting by filtering framework, and (2) AdaBoost does not seem to be noise resistant.

AdaBoost is defined for the "subsampling framework", where a sample of sufficient size, which is randomly selected before the boosting, is fixed throughout all the boosting process and distributions are defined only with respect to the sample. From the theoretical side, one consequence of having a boosting algorithm for the filtering framework is that we can directly get a bound on the generalization error of the boosting algorithm. From a more practical side, the advantage becomes more clear. First, we do not need to determine "sufficient sample size" for the boosting process. (Although some formulas exist to calculate sample size, they may not be easy to use and they usually give overestimated size.) Secondly, since a sample is not a priori fixed, we can run the weak learner on random samples of appropriate sizes at each iteration of the boosting; in this way, we can reduce the computation time particularly when the dataset is very large and we use appropriate sampling for scaling up the weak learner.

vi) Adaboost as Logistic Regression

1. Additive Logistic Regression: Fitting class conditional probability log ratio with additive terms
2. Discrete builds an additive logistic regression model via Newton-like updates for minimizing.
3. RealAB fits an additive logistic regression model by stage-wise and approximate optimization of
4. Even after training error reaches zero, AB produces a "purer" solution probabilistically.[26][3]

vii) WeightBoost

1. Uses input-dependent weighting factors for weak learners.

viii) BrownBoost

1. Non-monotonic weighting function
2. Examples far from boundary decrease in weight
3. Set a target error rate – algorithm attempts to achieve that error rate
4. No results posted (ever)

ix) Adaboost Variants Proposed By Friedman

1. LogitBoost

a. Solves

$$\min_{f(x)} E_{w(x)}\left(F(x)+\frac{1}{2}\frac{y^*-p(x)}{p(x)(1-p(x))}-(F(x)+f(x))\right)^2 \quad (2)$$

b. Requires care to avoid numerical problems

2. GentleBoost

a. Update is fm(x) = P(y=1 | x) – P(y=0 | x) instead of

$$f_m(x)=\frac{1}{2}\log\frac{P_w(y=1\mid x)}{P_w(y=-1\mid x)} \quad (3)$$

3. Gental AdaBoost

1. Start with weight

$w_i = 1/N, i = 1,2,....,N, F(x) = 0$

2. Repeat for $m = 1,2,...,M;$

(a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weight w_i

(b) Update $F(x) \leftarrow F(x) + f_m(x)$

(c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.

3. Output the classifier sign

$$[F(x)] = sign\left[\sum_{m=1}^{M} f_m(x)\right]$$

4. FloatBoost

Less greedy approach (FloatBoost) yields better results Stubs are not sufficient for vision [5].

5. AdaBoost.MH

- Training time:
  - Train one classifier: f(x'), where x=(x,c)
  - Replace (x,y) with k derived examples
    - ((x,1), 0)
    - …
    - ((x, y), 1)
    - …
    - ((x, k), 0)
- Decoding time: given a new example x
  - Run the classifier f(x, c) on k derived examples: (x, 1), (x, 2), …, (x, k)
  - Choose the class c with the highest confidence score f(x, c) [9].

6. AdaBoost.MI

- Training time:
  - Train k independent classifiers: f1(x), f2(x), …, fk(x)
  - When training the classifier fc for class c, replace (x,y) with
    - (x, 1) if y = c
    - (x, 0) if y != c
- Decoding time: given a new example x
  - Run each of the k classifiers on x
  - Choose the class with the highest confidence score fc(x).

AdaBoost.M1 and AdaBoost.M2 were developed by Schapire and Freund from their AdaBoost algorithm. For binary classification problems the two versions are equivalent, differing only in the way they handle problems with more than two classes. AdaBoost.M1 has access to a learning algorithm (which the developers generically title WeakLearn) which it calls repeatedly with distributions over the training set. WeakLearn calculates a hypothesis, or

classifier, that attempts to correctly classify all instances of the test data, greater weighting for the next pass. Finally, the boost algorithm combines all the hypotheses into one final hypothesis.

7. AdaBoost.M1

Input: sequence of N examples $\langle (x_1, y_1), ..., (x_N, y_N) \rangle$ with labels $y_i \in Y = \{1, ..., k\}$ distribution *D* over the examples weak learning algorithm **WeakLearn;** integer *T* specifying number of iterations

Initialize the weight vector: $w_i^1 = D(i)$ for i=1,…,N.

Do for $t = 1, 2, ..., T$

1. Set
$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$
2. Call **Weaklearn**, providing it with the distribution p^t ; get back a hypothesis $h_t : X \to Y$
3. calculate the error of $h_t \in_t = \sum_{i=1}^{N} P_i^t [h_t(x_i) \neq y_i]$
   If $\in_t > 1/2$, then set $T = t-1$ and abort the loop.
4. Set $\beta_t = \in_t / (1 - \in_t)$
5. Set the new weight vector to be
$$w_i^{t+1} = w_i^t \beta_t^{1-[h_t(x_i) \neq y_i]}$$

Output the hypothesis

$$h_f(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) [h_t(x) = y]$$

8. AdaBoost.M2

Input: sequence of N examples $\langle (x_1, y_1), ..., (x_N, y_N) \rangle$ with labels $y_i \in Y = \{1, ..., k\}$ distribution *D* over the examples weak learning algorithm **WeakLearn;** integer *T* specifying number of iterations

Initialize the weight vector: $w_{i,y}^1 = D(i)/(k-1)$ for i=1,…,N, $y \in Y - \{y_i\}$.

Do for $t = 1, 2, ..., T$

1. Set $W_i^t = \sum_{y \neq y_i} w_{i,y}^t$;
$$q_t(i, y) = \frac{w_{i,y}^t}{W_i^t}$$
For $y \neq y_i$; and set
$$D_t(i) = \frac{W_i^t}{\sum_{i=1}^{N} W_i^t}$$
2. Call Weaklearn, providing it with the distribution D_t and label weighting function q_t ; get back a hypothesis $h_t : X \times Y \to [0,1]$
3. Calculate the psedo-loss of h_t :
$$\in_t = \frac{1}{2} \sum_{i=1}^{N} D_t(i) \left(1 - h_t(x_i, y_i) + \sum_{y \neq y_i} q_t(i, y) h_t(x_i, y) \right)$$
4. Set $\beta_t = \in_t / (1 - \in_t)$
5. Set the new weight vector to be
$$w_{i,y}^{t+1} = w_{i,y}^t \beta_t^{(1/2)(1 + h_t(x_i, y_i) - h_t(x_i, y))}$$
For $i = 1, ..., N, y \in Y - \{y_i\}$

Output the hypothesis

$$h_f(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) [h_t(x, y)]$$

IV. Conclusion

1. It is possible to make an accurate classifier even if we have only weak learning algorithms better than random guessing
2. It has no parameters to tune (except for the number of rounds)

It works well

1. Better than using the learning algorithms only
2. Better than bagging which does not alter distribution

Sometimes it works badly

1. Overfitting happens
2. The performance is dependant on the data

V. Future Work

1. How to learn weak hypotheses?
2. How to deal with multi-class problems?
3. Local decision vs. globally best sequence

Comparison of Bagging and Boosting

Sr.No.	Criteria	Bagging	Boosting
1	Models	Independent	Dependent
2	Model Construction	Parallel	Iterative Stage Wise (Sequence)
3	Computational Requirements in Model Building	Relatively (some what) High	Higher
4	Parallelism in model building	Yes	No
5	Predictive Capabilities	Moderate	Superior
6	Training Examples	Equally likely in subsequent phase	Each is used in subsequent phase
7	Weighted	No	Yes
8	Weight based on	N.A.	Classification Error
9	Input Taken	Equally likely	From a probability distribution
10	Invokes a given classification procedure	Parallel	Series
11	Aggregated Model	Vote based	Weighted average of the models constructed

12	Capability	Less	More correctly classifying the samples
13	Preciseness	Less	More
14	Distribution of weights over training set	No	Yes
15	Goodness of the Model	N.A.	Measured by error
16	Final Classifier's Property	Vote	Is based on weighted Majority vote
17	Reduce Sample Error	No	Yes
18	Dependent on	No dependency	Characteristics of data set being examined
19	Performance	consistent	For noisy dataset, may decrease
Sr.No.	**Criteria**	**Bagging**	**Boosting**
20	Gain in performance	N.A.	Because of first few classifier combined
21	Diversity	N.A.	More, errors on different parts of input space
22	Rely on resampling technique	Yes	Yes
23	Influenced by learning algorithms	Some what	Yes
24	Widely varying results	No	Yes
25	Sensitivity to noise	Less	Because of increase in error (occasionally)
26	Resilient to Noise	More	Less
27	Average Generalization Error	Moderate	Low
28	Disagreement among the Classifiers	Less	Somewhat
29	Effective on	Unstable learning algorithm	
30	Resampling depends on earlier classifiers?	No	Yes
31	Reducing errors	By reducing the variance, Bias	By reducing the variance
32	Non linear Predictions	No	Yes

REFERENCES

[1] Freund – "Experiments with a new boosting algorithm"
[2] Freund, Schapire – "A decision-theoretic generalization of on-line learning and an application to boosting"
[3] Friedman, Hastie, etc – "Additive Logistic Regression: A Statistical View of Boosting"
[4] Jin, Liu, etc (CMU) – "A New Boosting Algorithm Using Input-Dependent Regularizer"
[5] Li, Zhang, etc – "Floatboost Learning for Classification"
[6] Opitz, Maclin – "Popular Ensemble Methods: An Empirical Study"
[7] Ratsch, Warmuth – "Efficient Margin Maximization with Boosting"
[8] Schapire, Freund, etc – "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods"
[9] Schapire, Singer – "Improved Boosting Algorithms Using Confidence-Weighted Predictions"
[10] Schapire – "The Boosting Approach to Machine Learning: An overview"
[11] Zhang, Li, etc – "Multi-view Face Detection with Floatboost"
[12] G. Rätch, T. Onoda, and K.-R. Müller. An improvement of AdaBoost to avoid overfitting.
[13] G. Rätch, T. Onoda, and K.-R. Müller. Regularizing AdaBoost. 1999
[14] D. Mease and A. Wyner. Evidence Contrary to the Statistical View of Boosting. 2005
[15] Jerome H. Friedman(IMS 1999 Reitz Lecture):
[16] Greedy Function Approximation: A Gradient Boosting Machine
[17] Jerome H. Friedman(1999): Stochastic Gradient Boosting
[18] www.boosting.org
[19] T.Hastie, R.Tibshirani, J.Friedman. "The Elements of Statistical Learning- Data Mining,Inference, Prediction." Springer Verlag.
[20] R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNCS, pages 119-184. Springer, 2003. In press. Copyright by Springer Verlag. (PDF)
[21] R.E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.
[22] Y.Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September 1999. Appearing in Japanese, translation by Naoki Abe.
[23] Duda, Hart, ect – *Pattern Classification*Freund – "An adaptive version of the boost by majority algorithm"
[24] Freund – "Experiments with a new boosting algorithm"
[25] Freund, Schapire – "A decision-theoretic generalization of on-line learning and an application to boosting"
[26] Friedman, Hastie, etc – "Additive Logistic Regression: A Statistical View of Boosting"
[27] Jin, Liu, etc (CMU) – "A New Boosting Algorithm Using Input-Dependent Regularizer"
[28] Li, Zhang, etc – "Floatboost Learning for Classification"
[29] Opitz, Maclin – "Popular Ensemble Methods: An Empirical Study"
[30] Ratsch, Warmuth – "Efficient Margin Maximization with Boosting"
[31] Schapire, Freund, etc – "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods"
[32] Schapire, Singer – "Improved Boosting Algorithms Using Confidence-Weighted Predictions"
[33] Schapire – "The Boosting Approach to Machine Learning: An overview"
[34] Zhang, Li, etc – "Multi-view Face Detection with Floatboost"
[35] BAGGING PREDICTORS L.BREIMAN, MACHINE LEARNING, 26(2), P.123-140, 1996
[36] C. J. Alonso Gonz´alez and J. J. Rodr´ıguez Diez