The Boosted/Bagged Subclass Method

I. Takigawa, N. Abe, Y. Shidara, and M. Kudo
Division of Systems and Information Engineering,
Graduate School of Engineering, Hokkaido University,
Kita-13, Nishi-8, Kita-ku, Sapporo, 060-8628, JAPAN
(e-mail: {1gac,chokujin,shidara,mine}@main.eng.hokudai.ac.jp)

Abstract
The subclass method is one of pattern classification methods proposed by Kudo *et al.* (1989), which is based on the approximation of each class region by a set of axis-parallel hyper-rectangles. This study improved it using an adaptive resampling technique known as *boosting*. Boosting is a well known ensemble learning method as an effective tool for improving the classification performance. Regarding the subclass method as a method for controlling the performance of resultant classifier, we could investigate 1) how the performance of a base classifier effects the classification results by boosting, and 2) how much boosting can improve the results compared with the original subclass method. Moreover, we also investigated the result using *bagging* which is an another popular ensemble technique.
Keywords : Subclass Method, Boosting, Adaboost, Bagging, Resampling.

1 Introduction

As attention has been increasing recently in machine learning community, boosting is a general method for improving the accuracy of a given learning algorithm. For various problems which we confront in practical situation, it is generally difficult to obtain a highly accurate classifier only from given instances. However, if we just have to obtain a rough classification result, we might have a something simple algorithm heuristically in many cases.

For example, Schapire (2002) takes the spam filtering problem. It is generally a difficult task to detect spam mails with high accuracy. But we will easily have some intuitive but rough rules such as "if the phrase 'buy now' occurs in the email, then predict it is spam", which will be significantly better than random guessing. If we can obtain a more accurate rule by combining such rough rules, we can avoid the bothersome problem of constructing the highly accurate single rule for our various practical problems. Boosting and similar ensemble learning approaches have been developed for achieving this purpose. Kearns and Valiant (1988) showed that boosting and its variants can "boost" the weak rule up to a strong rule.

In order to boost a learning algorithm, we first use it for a randomly selected subset of original training data. Then we obtain a weak classifier. Next, we test it

for all of given training data. With the classification result, we select again a subset of training data for the next step so as to emphasize the data misclassified by the classifier. Then, misclassified samples at each step will be selected more often in the subsequent step than correctly classified ones. After this procedure is repeated many times, we collect the generated weak classifiers and combine them as a final single classifier which will be more accurate than any of each weak classifiers. Thus, boosting can improve a given learning algorithm.

In this paper, we study on improvement of *the subclass method* proposed by Kudo *et al.* (1989, 1996) from the viewpoint of boosting. The subclass method is based on approximation of each class region by a set of axis-parallel hyper-rectangles. Since the parameter of the subclass method can adjust the accuracy of approximation, the subclass method can be regarded as a method for controlling the classification performance. By using it as a base learning algorithm for boosting, we can also investigate the relation between the performance of a base learning algorithm and the classification results by boosting it. In addition, we also investigated the result using *bagging*, which is an another popular ensemble technique.

2 Methodology

In this paper, we treat only binary pattern classification problems (2 class discrimination problems) on some feature space $\mathcal{X}$. For given n data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathcal{X}$ is a feature vector of data and $y_i \in \{-1, 1\}$ is its class label, our goal is to construct a good classifier $f : \mathcal{X} \to \{-1, 1\}$ which produces -1 or 1 for any (that is, unobserved) data point $x \in \mathcal{X}$ as precise as possible.

First, we describe briefly the boosting algorithm, the bagging algorithm and the subclass algorithm.

2.1 Boosting and Adaboost Algorithm

The theory guarantees that boosting can reduce the training error (the training error can approach to zero exponentially quickly as T increases), namely we can boost a weak learner as long as our base learner is always better than random guessing. Moreover, it also tends to reduce the test error. It is known that boosting can be regarded as a greedy optimization method for searching the Bayes-optimal classifier by logistic regression in statistics (Friedman *et al.*, 2000). It can maximize the minimum "margin" on training data in the viewpoint of large margin classifiers framework (Schapire *et al.*, 1998).

Adaboost algorithm (Freund and Schapire, 1996) is one of the most widely used approaches for boosting. Boosting requires a base learning algorithm (called weak learner or base learner). If "base learner" can accept weighted instances in the case of, for example, using C4.5, Adaboost can improve it not by resampling but by simply reweighting (Quinlan, 1996) which is a more direct implementation of the

theory. However, we use a more general (but weak) version for boosting the subclass method. Adaboost algorithm using resampling is shown in Figure 1. We also use a restart determination due to Breiman (1996b) where we reset all weights to be equal and restart if either weighted error $\epsilon^{(t)}$ is not less than 0.5 or $\epsilon^{(t)}$ becomes 0.

Algorithm [Adaboost by resampling]

1. **Input:** Training data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$
   Number of iterations T
2. **Initialize:** $w_i^{(1)} := 1/n$ for all $i = 1, \ldots, n$
3. **Do for:** $t = 1, \ldots, T$
   (a) $\mathcal{B}^{(t)} :=$ resampled n samples with replacement from $\mathcal{D}$ according to the weights $w_i^{(t)}$ as a probability mass for (x_i, y_i).
   (b) Train the "base learner" $h^{(t)}(x)$ with $\mathcal{B}^{(t)}$.
   (c) Test it for $\mathcal{D}$ and measure its weighted error.

$$\epsilon^{(t)} := \sum_{i=1}^{n} w_i^{(t)} \mathbb{1}_{(y_i \neq h^{(t)}(x_i))} \qquad \alpha^{(t)} := \sqrt{(1 - \epsilon^{(t)})/\epsilon^{(t)}}$$

   (d) If not $0 < \epsilon^{(t)} \leqslant 1/2$ then restart with $w_i^{(t)} := 1/n$ (i.e., go to (a))
   Therefore, we can obtain $0 < \epsilon^{(t)} \leqslant 1/2$ (i.e., $\alpha^{(t)} > 1$).
   (e) Update the weights so as to emphasize the misclassified samples

$$w_i^{(t+1)} := \begin{cases} w_i^{(t)} \times 1/\alpha^{(t)} & \text{if } y_i = h^{(t)}(x_i) \text{ (correctly classified)} \\ w_i^{(t)} \times \alpha^{(t)} & \text{if } y_i \neq h^{(t)}(x_i) \text{ (incorrectly classified)} \end{cases}$$

   (f) Normalize the weights so as to satisfy $\sum_{i=1}^{n} w_i^{(t+1)} = 1$.
4. **Output:** Combined learner as weighted majority voting

$$f(x) := \text{sign}\left(\sum_{t=1}^{T} (\log \alpha^{(t)}) h^{(t)}(x)\right)$$

Figure 1: Adaboost algorithm

2.2 Bagging

This paper also discusses an another popular approach called *bagging* (Breiman, 1996a) for combining weak classifiers. Bagging uses each subsampled training data

parallel, whereas boosting uses them sequentially. The algorithm is shown in Figure 2. It is well known that bagging tends to reduce the variance of the overall estimate and it can stabilize the base learning algorithm. It is quite often found to improve the performance of complex and unstable classifiers.

Algorithm [Bagging]

1. **Input:** Training data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$
   Number of iterations T
2. **Do for:** $t = 1, \ldots, T$
   (a) $\mathcal{B}^{(t)}$:= resampled n bootstrap samples from $\mathcal{D}$
   i.e., randomly select n samples from $\mathcal{D}$ with replacement.
   (b) Train the "base learner" $h^{(t)}(x)$ with $\mathcal{B}^{(t)}$.
3. **Output:** Combined learner by voting $f(x) := \text{sign}\left(\sum_{t=1}^{T} h^{(t)}(x)\right)$

Figure 2: Bagging algorithm

2.3 The Subclass Method

The subclass method (Kudo and Shimbo, 1989; Kudo *et al.*, 1996) based on approximation of each class region by a set of hyper-rectangles so that each hyper-rectangle includes training samples in a positive class maximally and excludes those of other classes (a negative class). Thus, it basically requires a sort of combinatorial examination of samples. The revised version (Kudo *et al.*, 1996) resolved this problem of computational complexity by introducing randomized mechanism. The algorithm is summarized in Figure 3.

Thus, the subclass method includes a kind of random search. Because the number of iterations, L, in randomized subclass method are directly connected to the accuracy of approximation for the target class ($1 \leqslant \#$ of rectangles $\leqslant L$), one can control heuristically the "roughness" of the generated rule by *early stopping* of the random search. Of course, if we stop the search or iteration too early, this kind of compromise makes the result poor or unstable. However, we can often expect that the generated classifier is at least better than random guessing, and in this case, it is possible to boost the early stopped results. Even for the classifiers other than the subclass method, if the learning algorithm includes some random search or convergence-type iterated algorithm, then similar approach will be possible.

Thus, the use of subclass method with early stopping as base classifiers can give a insight on the problem of what type base learning algorithm should we use. This model selection problem essentially have a problem-depend (or data-depend) aspect. Therefore, we performed an empirical study for some real datasets and examined the performance of boosted/bagged subclass method compared to the previous results.

Algorithm [Subclass Method]

Make a positive sample set from samples of a target class
and a negative sample set from the other samples.

1. **Input:** Positive data $\mathcal{P} = \{x_1, \ldots, x_n\}$ and negative data $\mathcal{N} = \{y_1, \ldots, y_m\}$
   the number of iterations L.
2. **Initialize:** $\Phi := \emptyset$, $w_i^{(0)} := 1$ for all $i = 1, \ldots, n$
3. **Do for:** $l = 1, \ldots, L$
   (a) $\phi := \emptyset$
   (b) Reorder positive samples of $\mathcal{P}$ by choosing them with the probability
   proportional to $1/w_i^{(l)}$
   (c) **Do for:** $i = 1, \ldots, n$
   (c-1) $\phi := \phi + \{x_i\}$
   (c-2) Check the exclusiveness of ϕ. Here, ϕ is exclusive,
   if the hyper-rectangle spanned by ϕ does not include
   any $y \in \mathcal{N}$.
   (c-3) If ϕ is exclusive, then $w_i^{(l)} := w_i^{(l-1)} + 1$.
   Otherwise, $\phi := \phi - \{x_i\}$
   (d) $\Phi := \Phi + \{\phi\}$
4. Removing duplication of Φ.
5. **Output:** Output Φ.

Discrimination: For a given sample x, find the largest subclass $\phi \in \Phi$ including x, where the *size* is measure by the ratio of samples included in the subclass to the size of the positive sample set. Label x by the class of the largest subclass. If no subclass includes x, the nearest subclass is used for assignment of class label.

Figure 3: Subclass algorithm

2.4 Strength of Base Learner for Boosting

In order to use boosting and bagging, we have to determine only two things: the type of "base learner" and the number of rounds, T. Clearly, "base learner" is a key ingredient of boosting/bagging algorithm because we need a good base learner for successful results. However, it is unclear how to choose a base learner appropriately for each problem. When a base learner is too weak, then we cannot expect good performance. On the other hand, when we use a strong base learner, it will lead to overfitting (Meir and Rätsch, 2003) or too much computational cost.

If our base learner is better than random guessing in many cases, we can expect at least the smaller training error than that of original one. However, in practical situation, we often fixed the number of round T. Thus, the performance of boosted classifiers for unseen test samples is directly related to the choice of base learning algorithm.

As a base learner of boosting and bagging, most existing works has been typically used the learning algorithm such as decision tree classifiers like C4.5 and MC4 (or just a decision stump), neural networks (two-layer perceptron, RBF nets, SVM, etc.), Naive-Bayes classifiers. Then the investigation of when we use the subclass method as a base learner is one of our motivations. Boosting is a kind of greedy and empirical method for improving a given base learner. In practical situation, whether it works well or not depends on the property of training data. Hence, we need the empirical study (this is a essential restriction came from the famous *No Free Lunch Theorem* in algorithm-free learning theory which states that in the absence of prior information about the problem there are no reasons to prefer one learning algorithm over another (Duda *et al.*, 2001)).

Thus, one of the prominent keywords for boosting-like approaches is "rough". We can investigate not only how effective the boosted/bagged subclass method is compared with the original subclass method, but also how the "roughness" of a base learner can effect the boosted results in practical situation.

3 Experiments

3.1 Effects in 2-Dimensional Classification Problem

First of all, we confirmed intuitively the effects of boosting and bagging for the subclass method by visualizing the decision boundary in 2-dimensional classification problem. Here, we used 2-dimensional data for a binary classification problem used in Fukunaga (Fukunaga, 1990). The results is shown in Figure 4. We can see that boosting makes the minimum margin large, which is a illustration of the fact previously mentioned in the subsection 2.1. We can also see that bagging makes boundary smooth and stable. Moreover, from this figures, we can understand intuitively how the subclass method works.

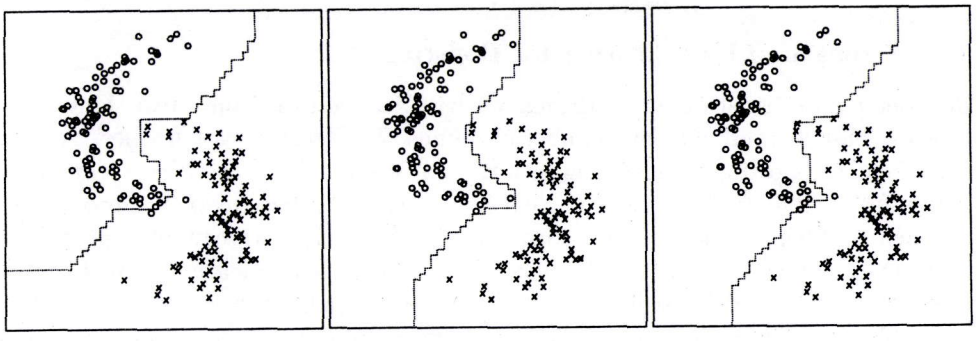

Figure 4: Decision boundaries of original subclass (left), Adaboosted subclass (center), and bagged subclass (right). The number of round is 100.

3.2 Subclass Method vs. Boosted Subclass Method

Using datasets from UCI Machine Learning Databases (Murphy and Aha, 1996), Friedman (Friedman, 1977) and UCI Irvine (Sklansky), we compared three types of algorithms, subclass method (SUB), adaboosted subclass method (Boosted), and well-known decision tree classifier (C4.5). We used 6 datasets which is for binary classification and have numerical features with no missing values: **sonar** (203 samples, 60 dimensional), **musk** (476 samples, 166 dimensional), **wdbc** (569 samples, 30 dimensional), **wpbc** (198 samples, 32 dimensional), **mammo** (86 samples, 65 dimensional), and **fried** (100 samples, 10 dimensional).

The original subclass method requires (# of samples$\times$# of features) times search. However, this randomized version often works well even when the number of iteration (L in Figure 3) is restricted to a fixed constant.

Results by early stopped version can be generally unstable because its successful result is based on some kinds of luckiness in terms of the randomness. However, when it is used for boosting, we can obtain better generalization performance than before. Table 1 shows that the parameter L can indeed control the classification performance for unseen test samples. Table 2 shows the results of recognition rate. For evaluation, SUB is based on 10 times average of 10-fold cross validation, and the boosted/bagged version is based on 5 times average of 10-fold cross validation. We used the fixed number of boosting iteration $T = 100$.

Table 1: Control the strength of classifiers using the subclass method

data	# of feat.	# of data.	SUB				C4.5
			$L = 1$	10	100	1000	
sonar	60	208	68.1	70.3	70.9	71.7	68.8
musk	166	476	80.0	88.8	89.1	89.3	86.3
wdbc	30	569	93.5	95.8	95.9	96.5	94.2
wpbc	32	198	67.7	75.7	77.0	78.0	65.2
mammo	65	86	65.5	67.1	66.9	66.2	73.3
fried	10	100	58.5	69.7	73.2	72.0	80.0

4 Discussion

The result in Table 2 shows that boosting and bagging for the subclass method performs well. The prominent and important result is that of boosted subclass method with $L = 1$. In Table 2, the result of $L = 1$ is often the best and boosted subclass method with $L = 1$ can give the subclass method a significant improvement in performance. When $L = 1$, the subclass method approximates a target class by just a single rectangle. Though just one time search seems to be too unstable, nevertheless the average result became often better than the original. This unstability for early

Table 2: Original subclass method vs. boosted/bagged subclass method

data	boosted subclass			bagged subclass			SUB	C4.5
	$L = 1$	10	100	1	10	100	1000	
sonar	**75.7**	74.3	74.5	72.0	74.3	**74.5**	71.7	68.8
musk	**90.3**	88.0	87.8	86.5	88.4	**89.1**	89.3	86.3
wdbc	**96.2**	96.1	96.1	94.9	95.6	**96.0**	**96.5**	94.2
wpbc	77.2	78.6	**79.3**	77.7	78.4	**79.8**	78.0	65.2
mammo	**74.9**	71.4	69.8	**74.4**	65.1	61.6	66.2	73.3
fried	65.0	80.2	**80.2**	57.0	77.2	**78.0**	72.0	80.0

stopping is resolved by combining many classifiers in boosting procedure. It would appear that the result of adaboosting accept our early stopping strategy. The reason for why $L = 1$ works well seems to be came from the following fact: The stronger classifier (in subclass case, larger L means strong approximation of a class region) might have a small change of decision boundary for resampled training data. Thus when we use a fixed number of round, T, it tend to have a small contribution at each round to the conclusive result for combined classifier. In contrast, the decision boundary of rough classifiers for each resampled data is very different usually. On the other hand, for bagging, we obtained better results in almost cases with the increase L (i.e., the performance of a base learner).

This result can also leads us to the new algorithm for the subclass method. In randomized procedure of the original subclass method, we can use adaptive resampling instead of random shuffling of samples (see Figure 5). Schapire (2002) gave the experimental result for comparing the boosting C4.5 and the boosting decision stumps. C4.5 is one of decision tree classifiers and decision stump is a single-test decision tree. Thus, this situation is similar to our relation between subclass method with $L \neq 1$ and $L = 1$. His experiment is based on 27 datasets of UCI benchmark reported by Freund and Schapire (1996). As a result, boosting stumps can usually give as good results as C4.5 but it does not have dominant performance compared with boosting C4.5. Moreover, it often cannot give the better results than random guessing, especially in multi class problems. Thus, instead of stumps, the use of a rectangle approximating a target class is one of possibilities for better and low cost results, i.e., we can use the boosted subclass method with $L = 1$ in Figure 5.

However, the obtained results here are restricted with these data previously listed and of course we must be careful about the fact that the performance is depend on the problem. Therefore we need further study on various type of real data for each problems which we would like to treat actually. The same test for 27 UCI datasets as Freund and Schapire (1996) will be a promising direction of research, but, unlike C4.5, the original subclass method could not have so far the methodology on treating the missing data and the nominal (not numerical) data often found in

27 UCI datasets. But the recent research on the subclass method will be able to give this type of discussion for boosting.

Randomized Subclass Method

Repeat L times:

1) Permute training samples randomly. (**random shuffling**)
2) Check the exclusiveness (add or discard) for each sample in a sequential order.
3) Obtain a subclass. (a rectangle approximating the target class region)

Remove duplication of generated subclasses.

Subclass Method using Adaptive Sampling

Initialize uniformly the weights on training samples.

Repeat L times:

1) Select training samples according to the weights. (**adaptive sampling**)
2) Check the exclusiveness (add or discard) for each sample in a sequential order.
3) Obtain a subclass. (a rectangle approximating the target class region)
4) Classify all training samples using only obtained subclass, and update weights so as to emphasize the misclassified samples.

Remove duplication of generated subclasses.

Figure 5: Procedure of original randomized subclass (top) and proposed modified subclass (bottom). The proposed method is the same as the adaboosted version of randomized subclass method using early stopping with $L = 1$.

5 Conclusion

We examined empirically the effectiveness of boosting and bagging when a hyper-rectangle base classifier called the subclass method is used. By using the early stopping strategy with it, we conclude that 1) boosting for the subclass method performed well. 2) it also worked well using early stopping of random search, particularly setting $L = 1$ can often give the significant improvement in practical performance. In this case, we proposed the new modified subclass method and gave the concise randomized procedure for the boosted subclass method using adaptive sampling instead of random shuffling of training samples. 3) boosting seemed not to require a strong base classifier in practical situations when we use a fixed number of iteration for boosting.

References

Breiman L. (1996a) Bagging Predictors. Machine Learning, 24, pp. 123–140.

Breiman L. (1996b) Bias, Variance, and Arcing Classifiers. Technical Report 460, UC-Berkeley.

Duda R. O., Hart P. E., and Stork D. G. (2001) Pattern Classification. 2nd Edition. John Wiley & Sons.

Freund, Y. and Schapire, R. E. (1996) Experiments with a New Boosting Algorithm. Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 148–156.

Friedman J. H. (1977) A Recursive Partitioning Decision Rule for Nonparametric Classification. IEEE Transactions on Computing. 26. pp.404–408.

Friedman J. H., Hastie H., and Tibshirani, R. (2000) Additive Logistic Regression: A Statistical View of Boosting. Annals of Statistics. 28, pp. 337–407.

Fukunaga K. (1990) Introduction to Statistical Pattern Recognition. 2nd Edition. Morgan Kaufmann.

Kearns M. and Valiant L. G. (1988) Learning Boolean formulae or Finite Automata is as Hard as Factoring. Technical Report TR–14–88, Harvard University Aiken Computation Laboratory.

Kudo M. and Shimbo M. (1989) Optimal Subclasses with Dichotomous Variables for feature selection and Discrimination. IEEE Transactions on Systems, Man, and Cybernetics, 19, pp. 1194–1199.

Kudo M., Yanagi S., and Shimbo M. (1996) Construction of Class Regions by a Randomized Algorithm: A Randomized Subclass Method. Pattern Recognition, 29(4), pp. 581–588.

Meir R. and Rätsch G. (2003) An Introduction to Boosting and Leveraging. Advanced Lectures on Machine Learning, LNCS, pp. 119-184. Springer.

Murphy P. M., and Aha D. W. (1996) UCI Repository of Machine Learning Databases [Machine-Readable Data Repository] Dept. of Information and Computation Science, University of California, Irvine.

Quinlan J. R. (1996) Bagging, Boosting, and C4.5. Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 725–730.

Schapire R. E. (2002) The Boosting Approach to Machine Learning An Overview. MSRI Workshop on Nonlinear Estimation and Classification.

Schapire R. E., Freund Y., Bartlett P., and Lee Wee S. (1998) Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics, 26(5), pp. 1651–1686.

Sklansky J., Mammogram Dataset. Pattern Recognition and Image Modeling Laboratory at University of California, Irvine.