# An Empirical Study of the Convergence of RegionBoost

Xinzhu Yang, Bo Yuan, Wenhuang Liu

Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, P. R. China
yangxz03@mails.tsinghua.edu.cn, {yuanb, liuwh}@sz.tsinghua.edu.cn

**Abstract.** RegionBoost is one of the classical examples of Boosting with dynamic weighting schemes. Apart from its demonstrated superior performance on a variety of classification problems, relatively little effort has been devoted to the detailed analysis of its convergence behavior. This paper presents some results from a preliminary attempt towards understanding the practical convergence behavior of RegionBoost. It is shown that, in some situations, the training error of RegionBoost may not be able to converge consistently as its counterpart AdaBoost and a deep understanding of this phenomenon may greatly contribute to the improvement of RegionBoost.

**Keywords:** Boosting, RegionBoost, Convergence, kNN, Decision Stumps.

## 1 Introduction

Since ensemble learning can help improve the accuracy of a single learning model, it has been an active topic in supervised learning for more than two decades. An ensemble of classifiers refers to a set of classifiers whose individual decisions are combined in some way to determine the class labels of unknown samples [1]. As a result, how to combine the individual classifiers is a key question in ensemble learning. In the past, many strategies have been developed, such as unweighted voting in Bagging [2] and Random Forests [3], weighted voting in Boosting [3], learning a combiner function [4] and stacking [5].

In addition to the static weighting methods, various dynamic weighting schemes have also been proposed in recent years, which take the features of input samples into consideration. Typical examples include dynamic voting(DV), dynamic selection(DS), and dynamic voting selection(DVS) [6]. Some of these schemes are independent of the underlying ensemble algorithms. For example, DV, DS, DVS have been applied to Bagging, Boosting and Random Forests respectively [6, 7]. Other strategies focus on specific ensemble methods, among which Boosting is the most popular one.

Boosting [8] encompasses a family of successful ensemble mechanisms by sequentially producing a series of classifiers and combining them using weighted voting. The training set used for each classifier is chosen based on the performance of the earlier classifier(s) in the series [9]. AdaBoost [10] is a one of the most commonly used Boosting algorithms. In the training process, AdaBoost decreases the weights of training samples classified correctly by the current classifier and increases the weights of those classified incorrectly to create a new training set for the next iteration.

When combining the decisions, AdaBoost assigns a single value weight $\alpha_t$ to each basic classifier $h_t(x)$ based on its error on the corresponding weighted training set, which means that the weights of classifiers are constant and will not change for different new input samples. Obviously, this approach ignores the inherent performance variability of classifiers on new samples with different feature values. In fact, the same basic classifier may not be as effective at classifying samples in some areas of the feature space as in other areas.

This issue has been addressed by several improved Boosting algorithms independently, which replace the fixed weighting methods with dynamic weighting schemes, including RegionBoost [9], DynaBoost [11], iBoost [12], WeightBoost [13] and local Boost [14]. A crucial factor of these schemes is how to dynamically adjust the weights with regard to the input samples.

Apart from some limited progresses, theoretical analysis on ensemble mechanisms has shown to be a very challenging task. For the dynamic weighting schemes, both theoretical analysis and comprehensive empirical studies have been rare in the literature. The purpose of this paper is to conduct a preliminary investigation of the practical behavior of Boosting with dynamic weighting schemes with focus on RegionBoost in order to gain deeper insights into these algorithms.

## 2 Dynamic Weighting Schemes

One of the major advantages of ensemble learning over a single model is that an ensemble approach allows each basic model to cover a different aspect of the dataset. When combined together, they are able to explain the whole dataset more thoroughly. Therefore, in order to take the full strength of ensemble learning, a good combination strategy should be able to examine the input pattern and only invoke the basic models that are appropriate for the input pattern [13].

### 2.1 Framework

A number of Boosting methods with dynamic weighting schemes have been proposed independently with proven superior performance over standard Boosting algorithms. However, the essential ideas are very similar: the basic classifiers should be weighted based on not only the training error but also the sample to be classified.

Although the implementation details are more or less different, the framework of Boosting with dynamic weighting schemes can be illustrated as Fig. 1. An extra learner $\alpha_t(x)$ is introduced for every classifier $h_t(x)$ as a competency predictor to evaluate the dynamic input-dependent weights for each classifier. Generally speaking, the extra learner is trained to simulate the performance of each basic classifier on the sample space. In other words, this extra learner is used to indicate whether a certain basic classifier is likely to yield accurate results given an unknown sample.

Many of the commonly used classification methods have been employed as the competency predictors such as *k*-Nearest Neighbor (RegionBoost), Neural Networks (RegionBoost) and Decision Trees (iBoost). It is clear that one of the key issues in Boosting with dynamic weighting schemes is how to construct the competency predictors in order to appropriately determine the weights, which can significantly affect the performance of the combined classifiers.

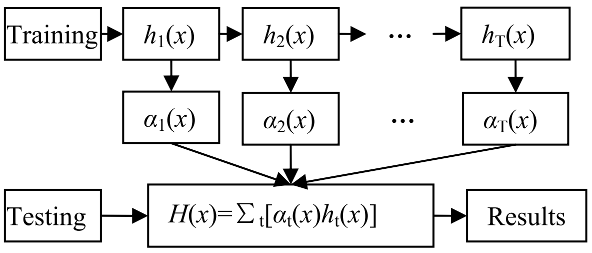


**Fig. 1.** The framework of Boosting with dynamic weighting schemes.

## 2.2 RegionBoost

In this paper, we used RegionBoost as the representative example of Boosting with dynamic weighting schemes in the empirical analysis. The main idea behind RegionBoost is to build an extra model upon each basic classifier based on its training results (whether a training sample is classified correctly or not). By doing so, this new model is able to estimate the accuracy or the competency of the classifier for each new sample.

One of the intuitive approaches to estimating model competency is to use the kNN (*k*-Nearest Neighbor) method to find the *k* points in the training set nearest to the new sample to be classified and use the performance of each classifier on these *k* points as the measurement [9]. More specifically, the weight of each classifier for this new sample to be classified is determined by the percentage of points (out of *k* points) correctly classified by this classifier [9].

RegionBoost has been reported having better performance compared to AdaBoost in terms of accuracy on a number of UCI benchmark datasets (though it makes little difference on certain datasets) [9]. It should be mentioned that, in AdaBoost, the value of $\alpha_t$ is selected so that the overall error on the training set is minimized. In the meantime, the overall training error enjoys a proven upper bound that consistently decreases during iterations [10]. By contrast, in RegionBoost, $\alpha_t(x)$ depends on the local error of the basic classifier $h_t(x)$ on the specific $x$. As a result, the convergence behavior of RegionBoost can no longer be explained according to the existing theoretical work on AdaBoost.

## 3 Convergence of RegionBoost

This section presents some interesting phenomena of the convergence of RegionBoost on a synthesized dataset as well as a few UCI benchmark datasets. A detailed analysis is also given to provide some deeper insights into the mechanism of RegionBoost.

## 3.1 The Triangle Dataset

First of all, a simple 2D dataset called Triangle as shown in Fig. 2(a) was used to illustrate the convergence behavior of RegionBoost. The dataset with 1000 samples ("○" vs. "x") was randomly divided into training set and testing set of equal sizes. The decision stumps model was used as the basic classifier, which is a weak classification method and can only split along a single attribute (a simplified decision tree model). An ensemble of 200 classifiers was created (the process of creating basic models in RegionBoost is the same as that in AdaBoost) and kNN (*k*=5) was used to determine the dynamic weights for each basic classifier.

When comparing the training error curves of RegionBoost and AdaBoost shown in Fig. 2(b), we found two distinctly different patterns. The training errors of Adaboost decreased consistently and reached zero in about 120 iterations. In contrast, the training errors of RegionBoost decreased faster at the beginning, but quickly flattened out after about 10 iterations.

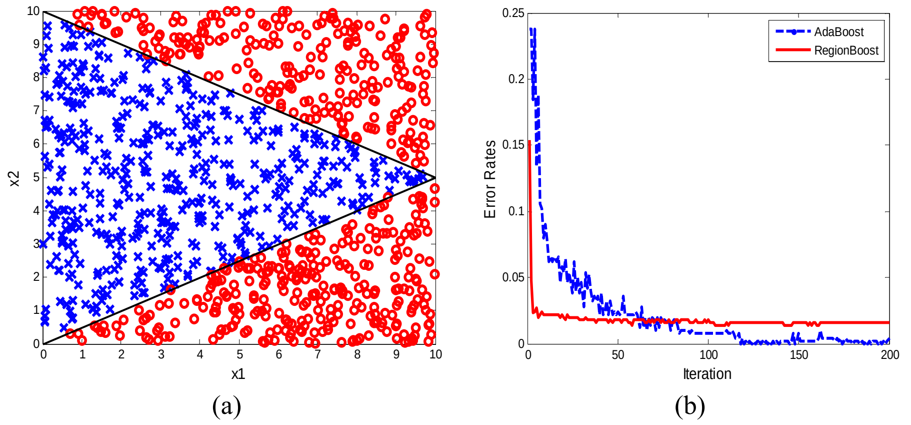


**Fig. 2. (a)** The Triangle Dataset. **(b)** The training errors of AdaBoost and RegionBoost.

Since RegionBoost employs an extra parameter *k* compared to AdaBoost, additional experiments were conducted to examine the influence of different *k* values. Table 1 shows the results of RegionBoost with stumps and kNN with *k* = 3, 5, 7, 9 respectively. It is interesting to see that the training errors of RegionBoost with stumps were very close to those of kNN with the same *k* values.

**Table 1.** The training errors of RegionBoost and kNN (The Triangle Dataset).

| **K** | **RegionBoost** | **kNN** |
|---|---|---|
| 3 | 0.0340 | 0.0340 |
| 5 | 0.0220 | 0.0220 |
| 7 | 0.0200 | 0.0220 |
| 9 | 0.0240 | 0.0260 |

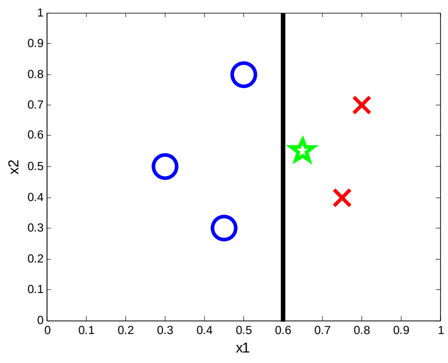
### 3.2 Analysis of the Triangle Dataset

Since it was observed that the training error of RegionBoost did not converge to zero and instead it was quite close to the error of kNN, it is reasonable to hypothesize that the convergence of RegionBoost has a strong relation with the accuracy of kNN, which is used to determine the weights of basic classifiers. In this section, we will give some explanations of this relationship from three aspects.

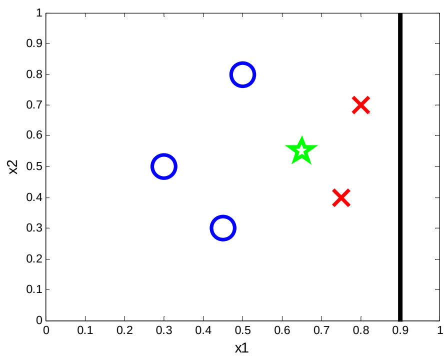
Firstly, let's take a look into the classification process of a single sample. Given a query point (denoted by a star), depending on the locations of its neighbors and the decision boundary, there are two possible situations. In the first one, the boundary divides the neighborhood of the sample into two parts, as shown in Fig. 3(a), which is called "Split" pattern for that sample. In the second one, the boundary does not go through the neighborhood, which is called "Non-Split" pattern, as shown in Fig. 3(b).

In Fig. 3, suppose that the star belongs to "x". In the "Non-Split" pattern, if the left side of the boundary is classified as "○", the query point will be classified as "○" with weight 0.6 (3 out of 5). On the flip side, the query point will be classified as "x" with weight 0.4 (2 out of 5). By combining the above two classifiers, the query point will be misclassified as "○", which is the same as kNN. As a result, if within a large number of iterations, the numbers of the two types of classifiers are close, the results of RegionBoost should be identical to kNN.

However, there is another situation that can make RegionBoost different from kNN. In Fig. 3(a), the classifier will either make the 5 neighbors all right or all wrong. When all neighbors are classified correctly, the query point will be correctly classified as "x" and the weight of the classifier is 1. On the flip side, the query point will be incorrectly classified as "○" but the weight is 0, making no negative effect on the overall decision at all. Consequently, in this situation, RegionBoost can always classify the query point correctly while kNN will make a wrong decision. This is an important reason that RegionBoost may reach lower training errors than kNN.



(a) Split Pattern



(b) Non-Split Pattern

**Fig. 3.** Relationships of the neighborhood of a query point and the decision boundary.

Note that in the "Non-Split" case, if all neighbors are of the same class label as the true class label of the query point, the weight of the basic classifier will be 1 if it makes the right decision (0 if it is wrong). As a result, RegionBoost can make the right decision on these query points immediately after a correct classifier is created. By contrast, in AdaBoost, multiple correct classifiers may be required to overturn the wrong classification results made by a previous classifier. This fact explains the faster convergence speed of RegionBoost at the beginning of training.

Secondly, we analyzed the influence of proportions of the two patterns on the convergence of RegionBoost. Let *NS*(*i*) and *S*(*i*) be the number of "Non-Split" pattern classifiers and the number of "Split" pattern classifiers for the *i*th sample respectively. Fig. 4(a) shows the histogram of $p(i)=NS(i)/T$ ($T$=200). It is clear that for most samples, "Non-Split" pattern classifiers accounted for more than 90% of the classifiers.

Let $NS_+(i)$ be the number of "Non-Split" classifiers that produce the same result as kNN on the *i*th sample and $NS_-(i)$ be the number of the rest "Non-Split" classifiers.

$$D(i) = (NS_+(i) - NS_-(i)) / NS(i) \tag{1}$$

The minimum *D* (*i*) that can guarantee the consistence between RegionBoost and kNN can be calculated as follows. For $k$=5, the minimum weight of a classifier is 0.6 if it produces the same result as kNN (e.g., the query point is classified as "○"and there are three neighbors belonging to "○") while the maximum weight of a classifier is 0.4 if it produces the opposite result as kNN. As a result, it is easy to see that, as long as *D* (*i*) is greater than -0.2 (i.e., 40% or more for $NS_+$ vs. 60% or less for $NS_-$), the set of "Non-Split" classifiers will be functionally identical to kNN.

Fig. 4(b) shows the distribution of *D* (*i*) indicating that all *D* (*i*) values were greater than -0.2. So the classification results from the ensemble of "Non-Split" classifiers were exactly the same as those of kNN.

Finally, the classification results from the ensemble of "Non-Split" classifiers, the ensemble of "Split" pattern and the overall results were compared. It turns out that the overall results of 498 out of 500 samples were the same as the results from the ensemble of "Non-Split" classifiers, which have been shown to be identical to kNN. In other words, the performance of RegionBoost was dominated by "Non-Split" classifiers and its training error should be close to the error of kNN (instead of converging towards 0).

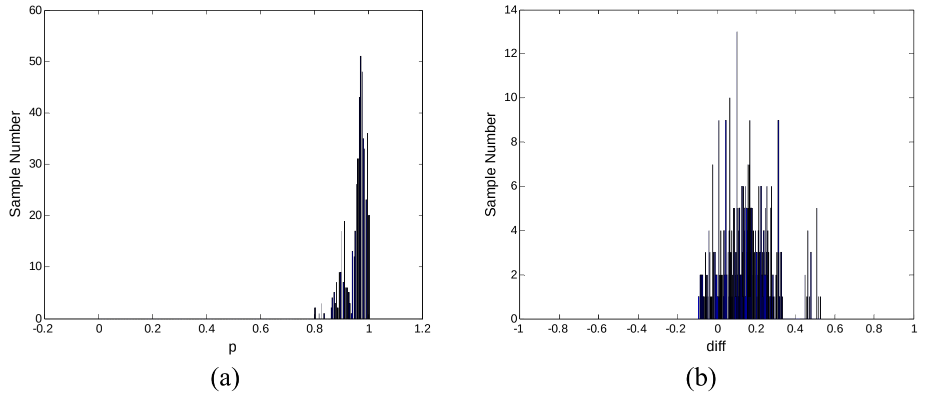


**Fig. 4.** The Triangle Dataset (a) The distribution of *p*(*i*). (b) The distribution of *D*(*i*).

### 3.3 Validation on UCI data

This section presents validation experiments on three commonly used UCI datasets: Pima, Credit Aus and Sonar [15]. The experimental settings were the same as in the Triangle dataset (50% samples were used as the training set). The training errors of AdaBoost, RegionBoost and kNN (*k*=5) are shown in Table 2. More specifically, on Pima and Credit Aus, the performance of RegionBoost was very close to kNN. In fact, the overall results of 383 out of 384 samples in Pima and 342 out of 345 samples in Credit Aus were the same as the results from the ensembles of "Non-Split" classifiers.

However the Sonar dataset showed something different. The training error of RegionBoost reached zero but the error of kNN was 0.1154. A closer look at the experimental results on Sonar showed that the final classification results of samples in Sonar were largely determined by the "Split" classifiers, which correctly classified 103 out of 104 samples. Note that the number of samples in Sonar is quite small (208) compared to its dimensionality (60). As a result, the neighbors of a query point are expected to be much sparser that in other cases and thus are more likely to be separated by the decision boundaries.

**Table 2.** The training errors on three UCI datasets.

| Dataset | AdaBoost | RegionBoost | kNN |
|---|---|---|---|
| Pima | 0.1328 | 0.1771 | 0.1797 |
| Credit Aus | 0.0667 | 0.0812 | 0.0899 |
| Sonar | 0 | 0 | 0.1154 |

## 4 The Effect of Basic Learners

The reason that the training errors of RegionBoost did not converge to zero in the experiments above may be partially due to the power of the basic learner in use. Since the decision stumps model can only create very simple decision boundaries, the neighbors of most query points may belong to the "Non-Split" category. In this section, we chose the decision trees model as the basic classifiers. The experimental results of AdaBoost, RegionBoost and kNN are shown in Table 3, showing that the training errors of RegionBoost with trees were much lower than with stumps. In the meantime, the results of RegionBoost with trees became apparently different from kNN, which may be due to the increasing percentage of the "Split" classifiers.

**Table 3.** A comparison of the training errors (Trees vs. Stumps).

| Dataset | AdaBoost (Trees) | RegionBoost (Trees) | AdaBoost (Stumps) | RegionBoost (Stumps) | kNN |
|---|---|---|---|---|---|
| Triangle | 0 | 0.0120 | 0 | 0.0220 | 0.0220 |
| Pima | 0 | 0.0547 | 0.1328 | 0.1771 | 0.1797 |
| Credit Aus | 0 | 0.0087 | 0.0667 | 0.0812 | 0.0899 |
| Sonar | 0 | 0 | 0 | 0 | 0.1154 |

## 5 Conclusions

In this paper, we empirically investigated the convergence of a typical Boosting algorithm with dynamic weighting schemes called RegionBoost, which employs kNN as the competency predictor of its basic classifiers. The major motivation was to provide one of the first attempts to better understand the behavior of RegionBoost.

Experimental results showed that the training errors of RegionBoost decreased faster than those of AdaBoost at the beginning of iterations but could not converge to 0 in some cases. Through detailed analysis, we showed that the reason lies in the fact that the training errors of RegionBoost are largely influenced by the accuracy of kNN, especially when the basic learners are very weak. We also demonstrated that the performance of RegionBoost can be enhanced with basic learners capable of creating more sophisticated decision boundaries.

In addition to the preliminary results presented here, there is still a huge open area for future work. For example, competency predictors other than kNN may also be used with RegionBoost and its convergence behavior in the new situation needs to be formally investigated and we hope that the techniques used in this paper may still be helpful to some extend. In the meantime, it is also important to think about the strategy for purposefully creating a set of basic classifiers that are better suited to the weighting schemes of RegionBoost.

## References

1. Dietterich, T. G.: Machine Learning Research: Four Current Directions. AI Magazine. vol. 18 (4), pp. 97-136 (1997)
2. Breiman, L.: Bagging Predictors. Machine Learning. vol. 24 (2), pp. 123-140 (1996)
3. Breiman, L.: Random Forests. Machine Learning. vol. 45 (1), pp. 5-32 (2001)
4. Jordan, M. I. and Jacobs, R. A.: Hierarchical Mixtures of Experts and the EM Algorithm. Neural Computation. vol. 6 (2), pp. 181-214 (1994)
5. Ting, K. M. and Witten, I. H.: Issues in Stacked Generalization. Journal of Artificial Intelligence Research. vol. 10, pp. 271-289 (1999)
6. Tsymbal, A. and Puuronen, S.: Bagging and Boosting with Dynamic Integration of Classifiers. In: Zighed, D. A. and Zytkow, H. J. K. J. M., (Eds.), 4th European Conference of Principles of Data Mining and Knowledge Discovery, LNCS, vol. 1910, pp. 116-125. Springer, Lyon, France (2000)
7. Tsymbal, A., Pechenizkiy, M., and Cunningham, P.: Dynamic Integration with Random Forests. In: Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., (Eds.), 17th European Conference on Machine Learning, LNCS, vol. 4212, pp. 801-808. Springer, Berlin, Germany (2006)
8. Schapire, R. E.: The Strength of Weak Learnability. Machine Learning. vol. 5 (2), pp. 197-227 (1990)
9. Maclin, R.: Boosting Classifiers Regionally. In: the 15th National Conference on Artificial Intelligence, pp. 700-705, Madison, WI (1998)
10. Freund, Y. and Schapire, R. E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences. vol. 55 (1), pp. 119-139 (1997)

11. Moerland, P. and Mayoraz, E.: DynaBoost: Combining Boosted Hypotheses in a Dynamic Way. In: IDIAP-RR, Switzerland (1999)
12. Kwek, S. and Nguyen, C.: iBoost: Boosting Using an Instance-Based Exponential Weighting Scheme. In: Elomaa, T., Mannila, H., and Toivonen, H., (Eds.), 13th European Conference on Machine Learning, LNCS, vol. 2430, pp. 245-257. Springer, Helsinki, Finland (2002)
13. Jin, R., Liu, Y., Si, L., Carbonell, J., and Hauptmann, A. G.: A New Boosting Algorithm Using Input-Dependent Regularizer. In: the 20th International Conference on Machine Learning, Washington, DC (2003)
14. Zhang, C.-X. and Zhang, J.-S.: A Local Boosting Algorithm for Solving Classification Problems. Computational Statistics & Data Analysis. vol. 52, pp. 1928-1941 (2008)
15. Asuncion, A. and Newman, D. J.: UCI Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html. University of California, School of Information and Computer Science (2007)