Empirical Comparisons of Online Boosting Algorithms

Sun Xiaowei[1, a *]

[1]Software College,Shenyang Normal University,Shenyang 110034,China,

[a]junyaomail@163.com

Keywords: boosting, ensemble learning,online learning, accuracy, running time

Abstract. Boosting is an effective classifier combination method, which can improve classification performance of an unstable learning algorithm due to its theoretical performance guarantees and strong experimental results. However, the algorithm has been used mainly in batch mode, i.e., it requires the entire training set to be available at once and, in some cases, require random access to the data. Recently, Nikunj C.oza(2001) proved that some preliminary theoretical results and some empirical comparisons of the classification accuracies of online algorithms with their corresponding batch algorithms on many datasets. In this paper, we present online versions of some boosting methods that require only one pass through the training data. Specifically, we discuss how our online algorithms mirror the techniques that boosting use to generate multiple distinct base models. We also present theoretical and experimental evidence that our online algorithms succeed in this mirroring. Our online algorithms are demonstrated to be more practical with larger datasets. We also compare the online and batch algorithms experimentally in terms of accuracy .

Introduction

Traditional supervised learning algorithms generate a single model such as a Naïve Bayes classifier or TAN[1] classifier or BAN[2] classifier and use it to classify examples. Ensemble learning algorithms combine the predictions of multiple base models, each of which is learned using a traditional algorithm. Boosting [3] is a well-known ensemble learning algorithm that has been shown to improve generalization performance compared to the individual base models. Theoretical analysis of Boosting's performance supports these results [4-7].

Nikunj C.oza(2001)[8] developed online versions of bagging and boosting. Online learning algorithms process each training example once “on arrival” without the need for storage and reprocessing, and maintain a current model that reflects all the training examples seen so far. Such algorithms run faster than typical batch algorithms in situations where data arrive continuously. They are also faster with large training sets for which the multiple passes through the training set required by most batch algorithms are prohibitively expensive.

In this paper, we also discussed some preliminary theoretical results and some empirical comparisons of the classification accuracies of our online algorithms with their corresponding batch algorithms on many datasets of varying size. We chose Naïve Bayes classifiers because a lossless online learning algorithm is available for them. For a given training set, a loss-less online learning algorithm returns a model identical to that returned by the corresponding batch algorithm. For BAN, we are forced to use a lossy online learning algorithm. In particular, we do not allow the BAN’s back-propagation algorithm to cycle through the entire training set in multiple epochs the way back-propagation is normally allowed to do. Overall, our online boosting algorithms perform comparably to their batch counterparts in terms of classification accuracy when using Naïve Bayes base models. The loss experienced by online BAN relative to batch BAN leads to a significant loss for online boosting relative to the batch versions. Online boosting never performs significantly better than single online BAN in our tests.

Oline Boosting Algorithm

Our online boosting algorithm is designed to be an online version of AdaBoost.M1[9]. AdaBoost generates a sequence of base Models h1,h2,……,hM using weighted training sets (weighted by D1,D2,……,DM) such that the training examples misclassified by model hm-1 are given half the

total weight when generating model hm and the correctly classified examples are given the remaining half of the weight.

Our online boosting algorithm is an online algorithm, its inputs are the current set of base models h={h1,h2,……,hM } and the associated parameters λsc={ λ1sc, λ2sc ,…… λMsc}and λsw={ λ1sw, λ2sw ,…… λMsw}(these are the sums of the weights of the correctly classified and misclassified examples, respectively, for each of the M base models), as well as an online base model learning algorithm L0 and a new labeled training example (x,y). The algorithm's output is a new classification function that is composed of updated base models h and associated parameters λsc and λsw . The algorithm starts by assigning the training example (x,y) the "weight" λ=1. Then the algorithm goes into a loop, in which one base model is updated in each iteration. For the first iteration, we choose k according to the Possion(λ) distribution, and call L0, the online base model learning algorithm, k times with base model h1 and example (x,y). We then see if the updated h1 has learned the example, i.e., whether h1 classifies it correctly. If it does, we update λ1sc, which is the sum of the weights of the examples that h1 classifies correctly. We then calculate ε1 which, just like in boosting, is the weighted fraction of the total examples that h1 has misclassified. We then update λ by multiplying it by the same factor 1/(2(1-εm)) that we do in AdaBoost. On the other hand, if h1 misclassifies example x, then we increment λ1sw, which is the sum of the weights of the examples that h1 misclassifies. Then we calculate ε1 and update λ by multiplying it by 1/(2εm), which is the same factor that is used by AdaBoost for misclassified examples. We then go into the second iteration of the loop to update the second base model h2 with example (x,y) and its new updated weight λ. We repeat this process for all M base models. The final ensemble returned has the same form as in AdaBoost, i.e., it is a function that takes a new example and returns the class that gets the maximum weighted vote over all the base models, where each base model's vote is log((1-εm)/εm), which is proportional to the base model's accuracy on the weighted training set presented to it.

Online Boosting Algorithm:

Initial conditions : For all m ∈ {1,2,……,M}, λmsc=0， λmsw=0.

Online Boosting(h, L0, (x, y))

Set the example's "weighted" λ=1.

For each base model hm, (m ∈ {1,2,……,M}) in h,

Set k according to Poisson(λ).

Do k times

hm = L0(hm, (x, y)).

If y = hm (x)

then

λmsc← λmsc+ λ

εm← λmsw/(λmsc+ λmsw)

λ ← λ(1/(2(1-εm)))

else

λmsw← λmsw+ λ

εm← λmsw/(λmsc+ λmsw)

λ ← λ(1/(2εm)

To classify new examples:

Return h(x)=arg maxc∈ Y∑m:hm(x)=clog((1-εm)/εm).

Comparisons and results

In this section, we discuss results on several datasets, whose names and numbers of training examples, test examples, classes, attributes and missing values are given in Table 1. The Census Income dataset comes with fixed training and test sets, which we use in our experiments. For the

remaining datasets, we used 5-fold cross-validation. We tested with some small datasets to show that the online algorithms can often achieve performance comparable to batch algorithms even when given a small number of data points. Of course, our results with larger datasets are more important. All but three of the datasets are from the UCI KDD repository [10]. The remaining three are synthetic datasets that were chosen because the performance of a single Naïve Bayes classifier varies significantly across these three datasets. These datasets allow us to compare the performances of the online and batch ensemble algorithms on datasets of varying difficulty.

Table 1. Datasets used in the experiments

No.	Dataset	Training Set	Test Set	Classes	Attributes	Missing values
1	Promoters	84	22	2	57	×
2	Breast-cancer-w	559	140	2	10	√
3	German	800	200	2	20	×
4	Car Evaluation	1382	346	4	6	×
5	Mushroom	6499	1625	2	22	×
6	Synthetic-1	80000	20000	2	20	×
7	Synthetic-2	80000	20000	2	20	×
8	Synthetic-3	80000	20000	2	20	×
9	Census Income	199523	99762	2	40	√
10	Forest Covertype	464809	116203	7	54	√

We present results using three different base model types: Naïve Bayes classifiers, BAN classifiers. Both boosting algorithms were allowed to generate up to 100 base models. All the results shown are based on10 runs of 5-fold cross validation (except on the Census Income dataset, where we used the supplied training and test sets). All the online algorithms were run five times for every one time the batch algorithm was run, with different random orders of the training set. This was done to account for the effect that the order of the training examples can have on the performance of an online learning algorithm. The online BAN was trained by using backpropagation to update the BAN with each training example ten times upon arrival; however, the algorithm only ran through the entire training set once in the order in which it was presented. The batch BAN was trained by using backpropagation to update the BAN in ten epochs (ten cycles through the entire training set). All comparisons between algorithms were made using a paired t-test (α =0.05).

Table 2 shows the results of running the boosting algorithms with Naïve Bayes classifiers. Entries in boldface/italics indicate that the ensemble algorithm performed significantly better/worse than a single Naïve Bayes classifier. In the "Online Boosting" column, any entry with a '+' or '-' after it indicates that online boosting performed significantly better/worse than batch boosting, respectively. With Naïve Bayes classifiers (Table 2), online boosting performed significantly worse than batch boosting on the Promoters, German and Car Evaluation datasets. For the remaining datasets, batch and online boosting performed comparably. On Mushroom, Census Income and Forest Covertype datasets, they performed comparably; while on Breast-Cancer-w they performed significantly better.

A scatter-plot comparing the test errors of batch and online boosting is shown in Figure 1—each point represents one dataset. Points above the diagonal line represent datasets for which the error of online boosting was higher than that of batch boosting and points below the line represent datasets for which online boosting had lower error. From Figure 1,we can see that batch boosting significantly outperforms online boosting in many cases—especially the smaller datasets. However, the performances of boosting and online boosting relative to a single Naive Bayes classifier agree to a

remarkable extent, i.e., when one of them is significantly better or worse than a single Naive Bayes classifier, the other one tends to be the same way.

Table 2. Experimental results with Boosting vs.online Boosting, Naïve Bayes

No.	Dataset	Naïve Bayes	Boosting	Online Boosting
1	Promoters	87.7	*84.6*	*71.4-*
2	Breast-cancer-w	96.5	*94.5*	*95.7+*
3	German	74.8	*73.5*	*68.8-*
4	Car Evaluation	85.7	**90.2**	**89.7-**
5	Mushroom	99.7	**100**	**99.9**
6	Synthetic-1	50	**50.7**	**50.1-**
7	Synthetic-2	78	**84.5**	**83.8-**
8	Synthetic-3	92.5	**96.8**	**96.9**
9	Census Income	76.3	**93.7**	**94**
10	Forest Covertype	67.6	67.5	67.5

Table 3 gives the results of running boosting with BANs. Entries in the online BAN and boosting column that are given in boldface/italics indicate that it significantly outperformed/underperformed relative to batch BANs. Entries in the online boosting column given in boldface/italics indicate times when it significantly outperformed/underperformed relative to the online BAN. With BAN classifiers (Table 3), online boosting performed significantly worse than batch boosting on most of the datasets. On Mushroom and Census Income datasets, they performed comparably.

Table 3. Experimental results with Boosting vs.online Boosting, BANs

No.	Dataset	BAN	Online BAN	Boosting	Online Boosting
1	Promoters	89.8	*80.4*	*86.4*	*62.5-*
2	Breast-cancer-w	96.2	*90.3*	**96.8**	*88.5-*
3	German	74.6	*70.6*	**75.1**	*68.2-*
4	Car Evaluation	94.2	*88.1*	**98.3**	*88.1-*
5	Mushroom	100	99.9	100	99.9
6	Synthetic-1	72.2	*65.4*	**72.3**	*63.4-*
7	Synthetic-2	85.6	*83.4*	**85.6**	*81.1-*
8	Synthetic-3	98.3	98.1	98.2	*95.8-*
9	Census Income	95.2	*94.8*	*94.9*	*94.3*
10	Forest Covertype	75.7	*69.7*	**77.8**	*65.3-*

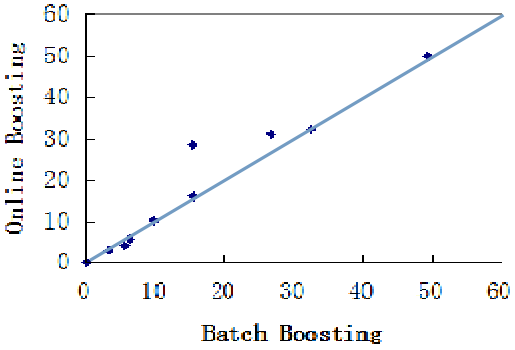

Figure 1. Test Error Rates:Batch Boosting vs. Online Boosting with Native Bayes base

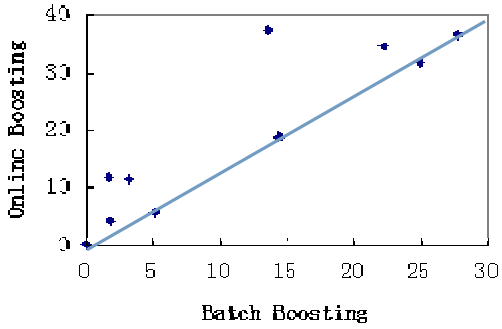

Figure 2. Test Error Rates:Batch Boosting vs. Online Boosting with BAN base models

Entries with a '-' after them indicate times when online boosting performed significantly worse than batch boosting. Clearly, the significant loss in using an online BAN instead of a batch BAN has rendered the online boosting algorithm significantly worse than batch boosting.

We can see from the tables and from the scatter-plots of batch and online boosting (Figure 2) that online boosting performs worse than batch boosting. Both batch boosting and online boosting do not improve upon BAN as much as they do upon Naive Bayes—especially on the larger datasets.

Summary

In this paper, we discussed online versions of boosting and gave both theoretical and experimental evidence that they can perform comparably to their batch counterparts in terms of accuracy while running much faster. We proved the convergence of the ensemble generated by the online boosting algorithm to that of batch boosting for BAN classifiers. The difference between the accuracies of the batch and online ensemble algorithms is largely a function of the differences between the accuracies of the batch and online base model learning algorithms. When lossless online base model learning algorithms are available (such as for Naïve Bayes classifiers), the performances of the ensemble algorithms tend to be comparable. In this paper, we experimented only with batch datasets, i.e., one is not concerned with concept drift. Online algorithms are useful for batch datasets that cannot be loaded into memory in their entirety. This paper provides a stepping stone to using ensemble learning algorithms on large datasets. We hope we can come up with new ideas to make ensemble learning algorithms more practical for modern data mining problems in the future.

References

[1] Cheng Jie, Greiner Russell, "Comparing Bayesian network classifiers", In: Kathryn Blackmond Laskey, Henri Prade eds. Proc of the 15th Conf on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann, pp.101-108, 1999.

[2] Friedman Nir, Geiger Dan, Goldszmidt Moises, "Bayesian network classifiers", Machine Learning, 29 (2/3), pp. 131-163, 1999.

[3] Bauer Eric, Kohavi Ron, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants", Machine Learning, 36 (1/2), pp. 105-139, 1999.

[4] Dietterich, Thomas, "Ensemble methods in machine learning", In Kittler, J., Roli, F., eds.: Multiple Classifier Systems. Lecture Notes Computer Sciences, Vol. 1857, pp. 1-15,2001.

[5] Freund Yoav, Robert Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting.Unpublished manuscript available electronically (on our web pages, or by email request)", An extended abstract appeared in Computational Learning Theory: Second European Conf, EuroCOLT, pp.23-37,1995.

[6] Xiaowei Sun, Hongbo Zhou, "An Empirical Comparison of Two Boosting Algorithms on Real Data Sets based on Analysis of Scientific Materials", Springer, Advances in Intelligent and Soft Computing, vol.105, pp.324-327, 2011.

[7] Hongbo Shi, Houkuan Huang, Zhihai Wang, "Boosting-Based TAN Combination Classifier", Journal of Computer research and development, 41(2), pp. 340-345,2004.
[8] Nikunj C. Oza, "Online Ensemble Learning," Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 2001.
[9] Yoav Freund and Robert Schapire, "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting," Journal of Computer System Sciences, Vol. 55, No. 1, pp. 119-139, 1997.
[10]Stephen D. Bay, "The UCI KDD Archive UCI Machine Learning Repository." http://archive.ics.uci.edu/ml/.