Boosting in the Online Setting

CHARLES MARSH
crmarsh@princeton.edu

Abstract

Boosting in the batch setting is a well-known machine learning technique with strong theoretical foundations and extensive use in practice. In this paper, we study the role of *online* boosting, in which we must ensemble online weak learners into a single online strong learner. We begin by discussing existing online boosting algorithms (as well as their theoretical foundations) before conducting several experiments to assess the impact of (1) the number of weak learners ensembled, and (2) the type of weak learners ensembled. Our results show that the advanced weighting schemes of Chen et al. [5] are more robust than the original online boosting algorithm of Oza and Russell [15] to increases in the number of weak learners ensembled.

1 Introduction

Boosting in the online setting first appeared in Oza and Russell [15], who attempted to generalize the most well-known offline boosting algorithm, AdaBoost [8], to the online setting. Since then, online boosting has become known as one of the most successful online algorithms [12], achieving notable successful in applications of computer vision [10], namely for object recognition, detection, and tracking [11]. Online boosting is typically benchmarked by comparison to the performance of the weak learner ensembled.

In the traditional batch learning model, a learner is fed a number of labeled training examples before being assessed on the accuracy of their predictions on a separate, unlabeled test set. The online learning setting differs from this model in that the learner is fed a single example at a time and asked to predict its label; afterwards, the learner is told the example's true label before repeating the process; the hope is that the online learner is able to improve as data comes in. In the online setting, however, we have no concept of how much data we'll be receiving, or, in some cases, what the data will look like.

The distinctive factors of online boosting, then, are as follows:

1. The weak learners themselves must be online algorithms.
2. The number of weak learners ensembled must be specified prior to training.

Many popular online learning algorithms have focused on generalizing batch learning algorithms to this new setting, such as with Incremental Decision Trees [19]. Similarly, many popular online boosting algorithms generalize existing offline algorithms, such as Oza and Russell's generalization of AdaBoost and Leistner et al.'s generalization of GradientBoost. In this paper, we explore several such algorithms from a theoretical and experimental perspective.

The structure of this paper is as follows: In Section 2, we present an overview of several existing online boosting algorithms. This is followed by a discussion of several commonly used online weak learners in Section 3. Sections 4 and 5 present the results of a number of experiments investigating the role of the choice and number of weak learners used in online boosting. Note that in online boosting, the number of weak learners must be decided in-advice, unlike in offline boosting, where a new weak learner is typically added on every round; this is a key distinction between the two settings, and one that has not been extensively explored in literature.

Python code for all of the ensembling algorithms and weak learners can be found on GitHub.

2 Existing Online Boosting Algorithms

OzaBoost

The first online boosting algorithm, "OzaBoost", was outlined in Oza and Russell [15], and extended AdaBoost to the online setting. The key idea behind this algorithm was to model the arrival of examples as sampling (with replacement) from a Poisson process, where "harder" examples are given a higher mean.

In more detail: OzaBoost gives each new example an initial weight $\lambda = 1$. This example is then passed to the first weak learner, which is trained on this example $Poisson(\lambda)$ times. If the example is still misclassified by this weak learner, λ is increased before passing the example on to the next weak learner. In this way, an example that is misclassified by the first weak learner will be given more attention by the second weak learner, and so forth. Through a clever weighting scheme, OzaBoost gives the examples misclassified in one round half the total weight of all examples, just as in AdaBoost. Predictions are made by taking a weighted majority vote in which weak learner m has weight $w_m = \log(\frac{1-\epsilon_m}{\epsilon_m})$, where ϵ_m is its base error rate.

It is claimed in [15] that the OzaBoost algorithm converges to the performance of offline AdaBoost in the limit, and this is supported empirically by results from the same paper. Full code for OzaBoost is available here.

Online Gradient Boost (OGBoost)

From Leistner et al. [12], Online GradientBoost ("OGBoost") was designed in an attempt to make online boosting more robust to label noise. The high-level approach was to develop robust loss functions and derive an algorithm (OGBoost) parameterized on the choice of loss function (common choices include: 0-1 Loss, Exponential, Logit, and Hinge).

Before describing OGBoost in more detail, we note its use of *selectors*, which essentially add another layer of ensembling to the algorithm. OGBoost is parameterized on M *selectors* and uses K weak learners *per selector* (making for $M \times K$ total weak learners). On each round, for each selector, the best of the K weak learners is identified, and this learner is chosen as the representative for that selector. Each of the M representatives is then ensembled.

In the offline setting, on each round of boosting, GradientBoost searches for the weak learner that most closely correlates with the negative gradient of the loss. To translate this behavior to the online setting, OGBoost solves this same optimization problem iteratively by assigning each example a weight and, after training the K weak learners associated with selector m, updating the example's weight with the negative derivative of the loss to that selector, such that the n-th example (x_n, y_n) has weight:

$$w_n = -\ell'(y_n F_m(x_n))$$

where ℓ is the loss function and F_m is the ensembled classifier for the m-th selector. The next K weak learners for selector $(m+1)$ use this weight. Full code for OGBoost is available here.

To make for a fair comparison between OzaBoost and OGBoost, we assert that $K = 1$ in Sections 4 and 5, as is standard practice in the literature [5].

Online SmoothBoost

Chen et al. [5] were some of the first to discuss the number of weak learners chosen in online boosting. The concern is that choosing too many weak learners could lead to predictions dominated by redundant weak learners that did not learn well. However, including too few weak learners could make the ensembling process itself redundant, as the goal of boosting is to improve upon the performance of the weak learners [5].

The general approach of Chen et al. [5] was to adjust the offline weighting scheme of SmoothBoost [18] to the online setting in the form of Online SmoothBoost ("OSBoost"). In OSBoost, each weak hypothesis is given a weight α_i. The default weighting scheme is uniform (in this paper, OSBoost refers to this uniform weighting), but more intelligent approaches were devised in [5] and are described later on.

The weak learning assumption presented in [5] differs from that of the batch setting (typically, that the weak learners do better than random guessing):

Assumption 1 (Weak Learning)**.** *There exists an online weak learner h which can achieve advantage $2\gamma > 0$ for any sequence of examples and weights (on the examples) such that $|w| \geq c/\gamma^2$ and $w_t \in [0,1]$ for some constant c, where advantage is defined as:*

$$\sum_{t=1}^{T} \frac{w_t y_t h(x_t)}{|w|}$$

assuming that the weak learner h is trained on (x_t, y_t) after prediction.

Essentially, instead of looking at the generalization error of the weak learner, we look at its "reward", $y_t h_t(x_t)$, weighted by the importance of the current example. This approach leads to weights that are "smooth" and noise-tolerant. Full code for the uniform version of OSBoost is available here.

Online Boosting with OCP

OSBoost provides a scheme for weighting examples, but it doesn't intelligently weight the weak learners themselves. That is, as in OzaBoost, we'd like our ensembler to somehow give greater importance to good hypotheses. Thus, we assign each hypothesis h_i a weight α_i initialized to $\frac{1}{M}$ (for M weak learners). Chen et al. present two approaches to updating these weights.

The first such approach is to use Online Convex Programming (OCP). Our strong learning hypothesis at round t is formulated as:

$$H_t(x_t) = sign(\sum_{i=1}^{N} \alpha_t^{(i)} h_t^{(i)}(x))$$

where $h_t^{(i)}$ is the i-th weak learner at round t, and $\alpha_t^{(i)}$ is its weight. We thus want to find the optimal values of $\alpha_t = (\alpha_t^{(1)}, ..., \alpha_t^{(N)})$. In "OSBoost.OCP", the OCP problem uses the N-dimensional probability simplex as the feasible set and defines a loss function:

$$\ell_t(\alpha) = \max\{0, \theta - \sum_{i=1}^{N} \alpha_t^{(i)} h_t^{(i)}(x)\}$$

for some constant θ. This loss function is convex in α and allows us to solve for the optimal weights. Note that projection to the simplex is done in $\mathcal{O}(N)$ using the method of Duchi et al. [7]. Full code for OSBoost.OCP, including the simplex projection method, is available here.

Online Boosting with EXP

The second variant of OSBoost, "OSBoost.EXP", is based on the framework of *Predicting With Expert Advice* [4]. However, Chen et al. note that if we merely treat each weak learner as an expert, performance will be bounded by that of the best weak learner. Given that boosting aims to improve upon the performance of the weak learners, this would be an unsatisfying conclusion.

Instead, OSBoost.EXP takes the original N weak learners and creates N experts, where expert i predicts on round t using:

$$H_t^{(i)}(x) = sign(\frac{1}{i}\sum_{j=1}^{i} h_t^{(j)}(x))$$

thus taking a vote over the first i experts. These N experts are then passed into the Weighted Majority Algorithm of Littlestone and Warmuth [13].

Chen et al. go on to prove that if Assumption 1 holds and $T \geq c/(\delta\gamma^2)$, the error of the best expert of this form will be at most δ. Combining this with the results of [13], the expected error rate of OSBoost.EXP is at most $\delta + 2\sqrt{(\ln N)/T}$. Full code for OSBoost.EXP is available here

3 Weak Learners

The choice of weak learner is an interesting component of any boosting algorithm. Unlike in offline boosting, the weak learners for online boosting must themselves be online algorithms. Recall that the goal of boosting is to improve upon the performance of the weak learners, and thus it is based on the performance of said weak learners that we evaluate our algorithms.

Traditionally, algorithms for which "small perturbations" in the training sets result in large changes in prediction have been considered good candidates for ensembling techniques like boosting, which are effective at reducing variance and improving accuracy. In other words: "Instability is an essential ingredient for [ensembling methods]" [3].

While the choice of online weak learner varied across literature, the Perceptron [16] and Naive Bayes methods were popular, appearing in [5], [6], [15], and others. The former is a typically unstable algorithm, while the latter is typically stable. As is predicted by theory, existing literature did not show significance gains from boosting on Naive Bayes weak learners, while boosting on Perceptrons was found to be quite useful [15].

Other popular weak learners included: Incremental Decision Trees [19], which appeared in [15]; online histograms [9], which appeared in [12]; and multi-layer perceptrons, which appeared in [14]. Each of these algorithms was implemented, along with random decision stumps and k-NN (nearly identical in the offline and online setting) due to their respective popularities.

Code for each of the aforementioned weak learners can be found here. The performance of various weak learners is explored in Section 5.

4 Number of Weak Learners

In online boosting, the number of weak learners must be chosen beforehand. This choice, however, has not been explored extensively in literature. For example, in both [1] and [15], the number of weak learners is held static at M=100. To explore how this choice affects performance, we ran a number of experiments using the UCI data sets [2]. All experiments use binary labels, as multi-class classification was not explored.

To start, we chose three popular online weak learners: Naive Bayes, Random Decision Stumps, and Perceptrons. We then evaluated the performance of five online boosting algorithms on the UCI *breast-cancer* data set for various choices of M, the number of weak learners. The dataset was shuffled with the same random seed for every experiment. For Decision Stumps, the results of three trials were averaged; due to computational restrictions, only a single trial was conducted for the other weak learners. A subset of the results can be seen in Tables 1, 2, and 3. Values in **bold** indicate best-performing online algorithms for a given value of M. The full results are presented graphically on Page 7.

In general, the OSBoost family of algorithms were best at adapting to large increases in M. For example, with Perceptrons (Table 3), OzaBoost's accuracy drops nearly 10% from its best-performing value of M=10 to its worst-performing value of M=1000. Similarly, with Stumps (Table 2), OzaBoost drops over 20% from its best-performing value of M=10 to its worst-performing value of M=1000. On the other hand, OSBoost was nearly monotonically *increasing* in M when using Stumps (Table 2). Similarly, OSBoost.EXP dropped no more than 1.17% from peak to trough (again at M=1000) when using Perceptrons (Table 3).

Performance generally decreased beyond M=100, although there are a number of exceptions: OSBoost with Stumps (Table 2) was most accurate for M=500 and M=1000, and OSBoost.EXP with Naive Bayes (Table 1) was relatively stable from M=10 to M=1000. The ability of the OSBoost family to adapt so well to increases in M is likely due to the intelligent weighting scheme assigned to weak learners in [5]. In fact, Chen et al. suggest setting M to a specific upper-bound value and allowing the weighting scheme to whittle out bad learners. One could view the strong performance of the OSBoost algorithms for large M as validating this approach.

OzaBoost was never the best-performing algorithm for this dataset beyond M=10, with the most such victories going to OSBoost.EXP.

5 Choice of Weak Learner

We evaluated the performance of five online boosting algorithms with six different weak learners. Experiments were conducted on three UCI data sets: *heart*, *australian*, and *breast-cancer*. For every trial, M=100 weak learners were used (as in the literature). For each choice of ensembler and weak learner, results from three shufflings of the dataset were averaged (with the same random seed used for every experiment). The results can be seen in Tables 4, 5, and 6. A baseline error rate is reported for the given weak learner operating on its own. Values in **bold** indicate strict improvement over this baseline, which we define to be a 'win' for the ensembler.

Several interesting conclusions can be drawn from these results. Firstly, wins occurred most frequently with the Perceptron, which is consistent with the notion that boosting algorithms are most effective when paired with unstable weak learners (see Section 3). Further, wins occurred least frequently (only once) with k-NN, which is unsurprising given k-NN's extreme stability, and helps to explain its infrequent usage as a weak learner in literature.

Interestingly, while OSBoost.EXP had the most wins (six), OzaBoost was the second most effective (with five wins). On the other hand, if you instead measure a win as greater-than-or-equal-to the baseline, the OSBoost algorithms become more impressive, particularly with the k-NN and Decision Tree weak learners, performing exactly as well as the baseline for these weak learners in each table. This suggests that the OSBoost algorithms are 'safer' choices, as they rarely underperform compared to the weak learners, and often match their performance.

6 Conclusion

In this paper, we reviewed existing online boosting algorithms and assessed the impact of the number and type of weak learner. Modern boosting algorithms, such as OGBoost and the variants of OSBoost, were better equipped than OzaBoost to handle large increases in the number of weak learners. Further, these modern boosting algorithms were more adept at performing at least as well as their weak learners, whereas OzaBoost underperformed with some very stable weak learners, such as k-NN.

A suitable next-step would be to explore online boosting with multi-class datasets, such as with the Online Multi-Class Gradient Boost algorithm of Saffari et al. [17], and assess how the number and type of weak learner impacts performance in this more complicated setting.

M	OzaBoost	OGBoost	OSBoost	OSBoost.OCP	OSBoost.EXP
10	0.9399	**0.9516**	0.9267	0.9502	0.9502
50	0.9253	0.9472	0.9165	**0.9487**	**0.9487**
100	0.9326	0.9458	0.9165	0.9180	**0.9487**
200	0.9238	0.9472	0.9180	0.9282	**0.9516**
300	0.9238	**0.9487**	0.9150	0.9238	0.9472
500	0.9267	**0.9355**	0.9180	0.8623	0.9121
1000	0.9209	0.9370	0.9150	0.9209	**0.9443**

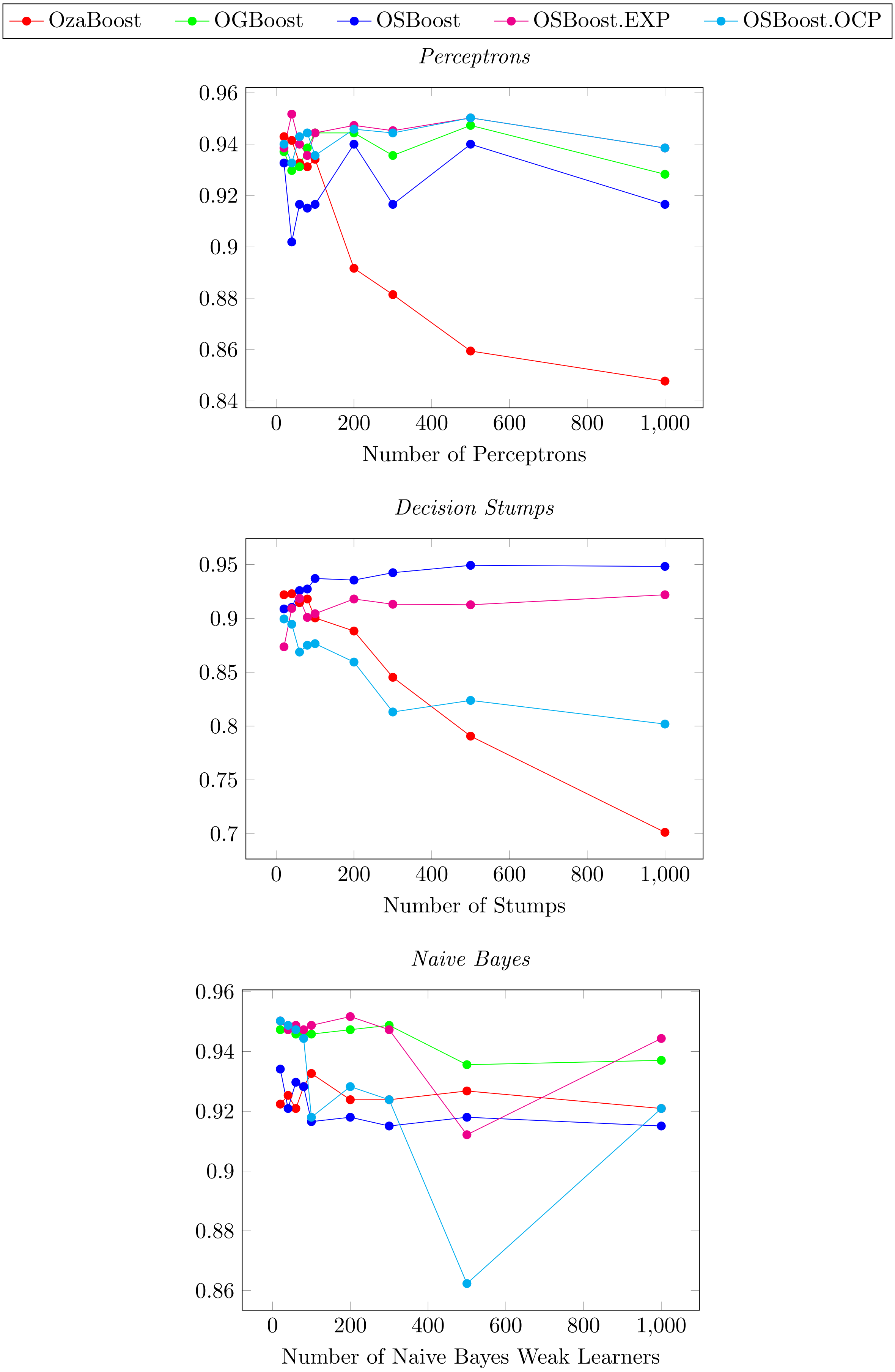
Table 1: Accuracy as a function of the number of weak learners using *Naive Bayes*.

M	OzaBoost	OSBoost	OSBoost.OCP	OSBoost.EXP
10	**0.9282**	0.8843	0.9048	0.8867
50	0.9253	**0.9316**	0.8931	0.9238
100	0.9004	**0.9370**	0.8765	0.9043
200	0.8882	**0.9355**	0.8594	0.9180
300	0.8452	**0.9424**	0.8130	0.9131
500	0.7906	**0.9492**	0.8238	0.9126
1000	0.7013	**0.9482**	0.8018	0.9219

Table 2: Accuracy as a function of number of weak learners using *Decision Stumps. OGBoost was excluded due to the lack of a prediction 'confidence' function for the weak learner.*

M	OzaBoost	OGBoost	OSBoost	OSBoost.OCP	OSBoost.EXP
10	0.9414	**0.9428**	0.9326	0.9385	0.9385
50	0.9399	0.9341	0.9121	0.9355	**0.9443**
100	0.9341	**0.9443**	0.9165	0.9355	**0.9443**
200	0.8916	0.9443	0.9399	0.9458	**0.9472**
300	0.8814	0.9355	0.9165	0.9443	**0.9452**
500	0.8594	0.9472	0.9399	**0.9502**	**0.9502**
1000	0.8477	0.9282	0.9165	**0.9385**	**0.9385**

Table 3: Accuracy as a function of the number of weak learners using *Perceptrons*.


OzaBoost
OGBoost
OSBoost
OSBoost.EXP
OSBoost.OCP
Perceptrons
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0
200
400
600
800
1,000
Number of Perceptrons
Decision Stumps
0.95
0.9
0.85
0.8
0.75
0.7
0
200
400
600
800
1,000
Number of Stumps
Naive Bayes
0.96
0.94
0.92
0.9
0.88
0.86
0
200
400
600
800
1,000
Number of Naive Bayes Weak Learners

Weak Learner	Baseline	OzaBoost	OGBoost	OSBoost	OSBoost.OCP	OSBoost.EXP
BinaryNB	0.7839	0.5543	0.7802	0.6851	0.7592	0.7740
GaussianNB	0.8037	0.7246	0.7271	0.7555	0.7925	0.7987
Perceptron	0.7506	**0.8098**	**0.7518**	0.7432	**0.7518**	**0.7530**
Stump*	0.7555	0.6975	-	0.7432	0.7074	0.6925
DecisionTree	0.6567	**0.6765**	-	0.6567	0.6567	0.6419
5-NN	0.7839	0.6641	-	0.7839	0.7839	0.7827

Table 4: Accuracy vs. choice of weak learner for the *heart* dataset.

Weak Learner	Baseline	OzaBoost	OGBoost	OSBoost	OSBoost.OCP	OSBoost.EXP
BinaryNB	0.8661	0.6835	0.8628	0.7487	0.7671	0.8550
GaussianNB	0.8309	0.7714	0.7782	0.7743	0.8231	0.8202
Perceptron	0.7956	**0.8565**	**0.7956**	0.7352	**0.7971**	0.7932
Stump	0.8521	0.7637	-	0.7497	0.6710	0.7777
DecisionTree	0.7657	**0.8024**	-	0.7657	0.7657	0.7260
5-NN	0.8347	0.7560	-	0.8347	0.8347	0.8323

Table 5: Accuracy vs. choice of weak learner for the *australian* dataset.

Weak Learner	Baseline	OzaBoost	OGBoost	OSBoost	OSBoost.OCP	OSBoost.EXP
BinaryNB	0.9492	0.9282	0.9414	0.9165	0.8760	**0.9511**
GaussianNB	0.9511	0.9111	0.9492	**0.9536**	**0.9526**	**0.9531**
Perceptron	0.9428	0.9219	0.9375	0.9238	0.9428	0.9419
Stump	0.9194	0.9145	-	**0.9331**	0.8604	**0.9214**
DecisionTree	0.8745	**0.8916**	-	0.8745	0.8745	**0.9009**
5-NN	0.9614	0.8931	-	0.9614	0.9614	**0.9633**

Table 6: Accuracy vs. choice of weak learner for the *breast-cancer* dataset.

*For Random Decision Stumps, randomness is over the choice of attribute on which to split. The accuracy of best-performing attribute is reported (biasing the results slightly in favor of the weak learner).

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. A Family of Online Boosting Algorithms. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1346–1353, Sept 2009.

[2] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

[3] L. Breiman. Bias, Variance, and Arcing Classifiers. Technical Report 460, University of California, Berkely, 1996.

[4] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to Use Expert Advice. *J. ACM*, 44(3):427–485, May 1997.

[5] S.-T. Chen, H.-T. Lin, and C.-J. Lu. An Online Boosting Algorithm with Theoretical Justifications. *ICML 2012*, 2012.

[6] S.-T. Chen, H.-T. Lin, and C.-J. Lu. Boosting with Online Binary Learners for the Multiclass Bandit Problem. *ICML 2014*, 2014.

[7] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient Projections Onto the L1-ball for Learning in High Dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 272–279, New York, NY, USA, 2008. ACM.

[8] Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm, 1996.

[9] M. Godec, C. Leistner, A. Saffari, and H. Bischof. On-Line Random Naive Bayes for Tracking. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3545–3548, Aug 2010.

[10] H. Grabner, C. Leistner, and H. Bischof. Semi-Supervised On-Line Boosting for Robust Tracking. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision ECCV 2008*, volume 5302 of *Lecture Notes in Computer Science*, pages 234–247. Springer Berlin Heidelberg, 2008.

[11] O. Javed, S. Ali, and M. Shah. Online Detection and Classification of Moving Objects Using Progressively Improving Detectors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 696–701 vol. 1, June 2005.

[12] C. Leistner, A. Saffari, P. Roth, and H. Bischof. On Robustness of On-line Boosting - A Competitive Study. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1362–1369, Sept 2009.

[13] N. Littlestone and M. K. Warmuth. The Weighted Majority Algorithm, 1992.

[14] N. C. Oza. Online Bagging and Boosting. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 3, pages 2340–2345 Vol. 3, Oct 2005.

[15] N. C. Oza and S. Russell. Online Bagging and Boosting. In *In Artificial Intelligence and Statistics 2001*, pages 105–112. Morgan Kaufmann, 2001.

[16] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, 1962.

[17] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online Multi-Class LPBoost. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3570–3577, June 2010.

[18] R. A. Servedio. Smooth Boosting and Learning with Malicious Noise. *Journal of Machine Learning Research*, 4:473–489, 2003.

[19] P. E. Utgoff. An Improved Algorithm for Incremental Induction of Decision Trees. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 318–325. Morgan Kaufmann, 1994.