The State of Online Boosting

Di Qi
dqi@princeton.edu

Ioannis Christos Karakozis
ick@princeton.edu

1. Motivation and Problem Description

Boosting in the batch setup is a well defined and well developed problem. There is a consensus on the definition of weak learners, based on weak PAC-learnability, the definition of strong learners, based on strong PAC-learnability, and the theoretical bounds for the performance of each type of learners [10]. However, in many cases in the real world, algorithms need to learn online, as examples are not readily available for batch training. Therefore, we need to be able to boost weak learners into strong learners in an online setup as well.

Online boosting is widely accepted to be defined as the task of converting any weak online learner into a strong online learner [2, 3, 8]. The difference from batch boosting is that there does not seem to be a consensus on the boosting framework and the definitions of weak and strong learnability in the online setup. As it will be analyzed in Section 2, there are very few attempts to formalize online boosting by developing online boosting algorithms with asymptotically optimal theoretical bounds on the number of weak learners and the number of iterations needed to attain a target error rate given a minimum "edge" of the weak online learners.

The first goal of this project is to provide a comprehensive overview of the state of the field on online boosting. The second goal of our work is to experimentally verify the bounds proposed in Beygelzimer *et al*. in [2] for the optimal and adaptive algorithms introduced by their work. We do so by performing binary classification experiments on the MNIST, Fashion MNIST, and CIFAR datasets, which are more challenging datasets than those used in the original paper [6, 7, 11]. Given this is the most recent work that proposes a formal definition for the online boosting framework, this second objective will allow us to verify whether the theoretical performance bounds for online boosting can be reproduced in practical applications.

We make our Python Notebooks with the experimental results publicly available at this link.

2. Differences between Batch and Online Boosting

An online boosting algorithm learns as it is being evaluated and the goal is to minimize the number of mistakes made while learning over T iterations, for large enough T. This implies that we have to predetermine the number of weak online learners our online boosting algorithm will use before training. This is unlike batch boosting, where we generate a weak learner for every iteration of our algorithm, until we achieve a satisfactory number of iterations, as seen in AdaBoost [10].

Another fundamental difference is that in batch boosting we see all of the examples from the very beginning, while in the online setup we observe one example at a time. In batch boosting, this means that we can easily measure the normalized weight-difficulty $p_t^{(i)}$ of classifying correctly each example (x_t, y_t) with respect to the weak learners generated so far and the other examples by updating $p_t^{(i)}$ at every iteration i, as seen in Adaboost [10]. However, in online boosting, the weight $w_t^{(i)}$ of each example x_t for each weak learner i must be determined before seeing the remaining examples [3]. Therefore, in online boosting, we need to predetermine the number of weak learners N AND carefully craft the weight update function to ensure online weak learnability.

Any online boosting algorithm has the following general structure: Given a sample (x_t, y_t), for every $i = 1, ..., N$ in order:

1. weak learner i predicts $h_t^{(i)}(x_t)$.
2. we update the weak learner $h_t^{(i)} \to h_{t+1}^{(i)}$, according to the weak learning algorithm.
3. we compute the weight $w_t^{(i+1)}$ of the example for the next weak learner $h_t^{(i+1)}$ from the current weak learner weight $w_t^{(i+1)}$, according to the weight update function.

3. The State of Online Boosting

The first online boosting algorithm was proposed by Oza and Russell [8]. They proved that, given the same training set of T examples for batch and online boosting algorithms, if the weak learners are Naive Bayes classifiers, then $\forall i \in \{1, ..., N\}$, the online weak learner $h_o^{(i)}$ converges to the batch weak learner $h_b^{(i)}$ in probability as T goes to infinity, where T is also the number of iterations of the online algorithm [8]. Simply put, they proved that for some class of weak learners, online boosting converges to batch boosting as the number of iterations becomes very large.

Oza and Russell showed that online boosting has the potential to achieve very low error rates, just like batch boosting. However, they did not specify how to pick the number of online weak learners N and they did not specify what weak learning means in the online setup. They overlooked this by using Naive Bayes, which is a batch weak learner that can be adjusted for the online setup [8]. Given that and the fact no theoretical justifications were given on the number of iterations T needed for convergence, there was a lot lacking in their online boosting framework.

Chen *et al.* were the first to resolve some of those problems. They did so by building upon SmoothBoost and proving the following theorem using their online SmoothBoost algorithm [3]:

Theorem 1. *For any sequence of examples $\{(x_t, y_t)\}_{t=1}^T$ and weights $\{w_t\}_{t=1}^T$, obtained by an online boosting algorithm, such that the following condition are true:*

1. *the weights satisfy the condition: $|w| \geq c/\gamma^2$ for some small constant c and $\forall t = 1, ..., T$: $w_t \in [0, 1]$, and*
2. *there exists an offline linear hypothesis h with positive advantage $\sum_{t=1}^T (p_t h(x_t) y_t) = 3\gamma > 0$, where p_t is the weight assigned to example x_t when training h in the batch boosting setup.*

there exists an online weak learner which can achieve positive advantage $2\gamma > 0$ on the same sequence of examples and weights.

Note that positive advantage implies an error rate strictly lower than 1/2 on the underlying weight distribution, i.e. a positive "edge", as this is defined in the AdaBoost algorithm [10]. Thus, positive advantage implies weak learnability, which means this theorem defines weak learnability in the online setup based on whether weak learnability is possible in the batch setup.

Using Theorem 1, Chen *et al*. were able to show that after $O(c/(\epsilon\gamma^2)$ iterations, assuming no more than $O(1/(\epsilon\gamma^2)$ weak learners are used, the error rate of the strong online learner is upper bounded by ϵ [3]. This way they provided both a lower bound on the number of iterations needed for convergence and an upper bound on the number of weak learners to be used. The reasoning behind the latter upper bound is that if too many weak learners are used, the outcome can be eroded by the many redundant weak learners which do not learn as well [3].

Despite the fact Chen *et al*. addressed most of the weaknesses of the framework introduced in [8], condition 1 of Theorem 1 makes their analysis relevant for "smooth" distributions, as those are defined in [3]. Given that and the fact they do not explicitly define the conditions for online weak and strong learnability, Chen's framework could be further generalized. This is exactly what the work by Beygelzimer *et al*. did, by providing a general definition for online weak learnability [2]:

Definition. *For parameters $\gamma \in (0, 0.5)$, $\delta \in (0, 1)$, a learner is said to be a weak online learner with edge γ and excess loss S if, for any T and any input sequences of examples $\{(x_t, y_t)\}_{t=1}^{T}$ chosen adaptively, it generates $\hat{y}_t$ such that, with probability $\geq 1 - \delta$:*

$$\sum_{t=1}^{T} 1_{\hat{y}_t \neq y_t} \leq (0.5 - \gamma)T + S$$

This definition is inspired from the weak learning definition of the batch setup, with the main difference being the excess loss requirement. The excess loss requirement is necessary since an online learner cannot be expected to predict with any accuracy when given few examples [2]. Note the weak learner starts obtaining a distinct edge $\Omega(\gamma)$ over random guessing only when $T \gg S/\gamma$.

Similarly inspired from the batch setup definition, they define online strong learnability [2]:

Definition. *For parameter $\epsilon > 0$, $\epsilon \in (0, 1)$, a learner is said to be a strong online learner with error rate ϵ and excess loss S if:*

$$\sum_{t=1}^{T} 1_{\hat{y}_t \neq y_t} \leq \epsilon T + S$$

Again, the excess loss S yields a sample complexity bound: the fraction of mistakes made by the strong online learner is $\mathcal{O}(\epsilon)$ when $T \gg S/\epsilon$. Thus, minimizing the excess loss of an online boosting algorithm can yield significant benefits in the number of iterations needed to achieve high performance.

By introducing the Online Boost-by-Majority (BBM) algorithm described in Figure 3 and using the definitions above, they prove the following theorem for online boosting [2]:

Theorem 2. *Given a weak online learning algorithm with edge $\gamma > 0$ and excess loss S, and any target error $\epsilon > 0$, there is a strong online learning algorithm with error rate ϵ which uses $\mathcal{O}(\frac{1}{\gamma^2}\ln(\frac{1}{\epsilon}))$ copies of the weak online learner and has excess loss $\tilde{\mathcal{O}}(\frac{S}{\gamma} + \frac{1}{\gamma^2})$. Thus, its sample complexity is $\tilde{\mathcal{O}}(\frac{1}{\epsilon}(\frac{S}{\gamma} + \frac{1}{\gamma^2}))$. If the $S \geq \tilde{\Omega}(\frac{1}{\gamma})$, then the number of weak online learners is optimal up to constant factors, and the sample complexity is optimal up to polylogarithmic factors.*

Thus, the online boosting framework is complete. Oza and Russell showed that online boosting can indeed converge to batch boosting and achieve comparable performance given enough iterations [8]. Chen *et al.* identified the need for an upper bound on the number of weak learners N and a lower bound on the number of iterations T for a target error rate to be achieved [3]. And finally, Beygelzimer *et al.* established formal definitions for weak and strong online learning and provided optimal theoretical bounds for both N and T [2].

The online boosting framework of [2] seems to be complete, as it transforms all definitions and theorems found in batch boosting into ones fitting the online setup. Further proof of that is that the field is also agreeing on this, as this framework has been acknowledged as the most generalized online boosting framework by recent works on both online multiclass [5] and online gradient boosting [1].

4. Optimal and Adaptive Algorithms

In our experiments, we evaluate the two algorithms that are introduced by Beygelzimer *et al.* in [2]. The first algorithm is the Online BBM algorithm, an online version of the boost-by-majority algorithm. Its detailed implementation is delineated in Figure 4 found in the Appendix. This algorithm achieves the optimal number of weak learners and near optimal sample complexity [2]. This is also the algorithm used to prove the bounds stated in Theorem 2. Beygelzimer *et al.* proved the following for this algorithm:

Theorem 3. *For any T and N, with high probability, the number of mistakes m made by Online BBM is bounded as follows:*

$$m \leq \exp(-0.5N\gamma^2)T + \tilde{\mathcal{O}}(\sqrt{N}(S + 1/\gamma))$$

The second algorithm, AdaBoost.OL, is an online version of AdaBoost. Its detailed implementation is delineated in Figure 3 found in the Appendix.. This algorithm, albeit not optimal, is claimed that performs better in practice than Online BBM [2]. Beygelzimer *et al.* proved the following for this algorithm:

Theorem 4. *If the weak learners satisfy the online weak learning condition, then for any T and N, with high probability, the number of mistakes m made by AdaBoost.OL is bounded as follows:*

$$m \leq \frac{8T}{\gamma^2 N} + \tilde{\mathcal{O}}(N/\gamma^2 + S/\gamma)$$

5. Experimental Setup

Despite Beygelzimer *et al.* proposing that their adaptive algorithm performs better than the optimal algorithm in practice, their experiments showed the opposite [2]. Our hypothesis is that for the conjectured behavior to emerge, the datasets have to be rich and "hard" enough. Thus, we are going to evaluate the performance of these binary classification boosting algorithms on the following visual classification tasks:

1. 0-4 vs 5-9 digit classification on MNIST, which is significantly harder than one-vs-all and one-vs-one digit classification [7].
2. Pullover vs Shirt binary classification on Fashion-MNIST, which is a hard problem due to the visual similarity of the two object classes [11].
3. Car vs Horse classification task on CIFAR10, a dataset with highly complex image backgrounds and intra-class variation, which makes the binary classification task hard for a simple linear classifier, despite the dissimilarity between the two object classes [6].

The online weak learner we use is the perceptron algorithm introduced in [9]. We choose the perceptron for three reasons: 1) it is simple to implement and verify the implementation is not error-prone, 2) it is a linear classifier, so it is guaranteed to not achieve very high accuracy in the complex visual examples from CIFAR 10 and Fashion-MNIST and the 0-4 vs 5-9 complex classification task on MNIST , thus, having a moderate edge γ, and 3) Beygelzimer *et al.* used a variant of the online gradient descent algorithm on a linear classifier that matches the properties of linear bound and additive updates of the perceptron algorithm [2]. We tune the learning rate of the perceptron on each dataset using cross validation on the weak learner.

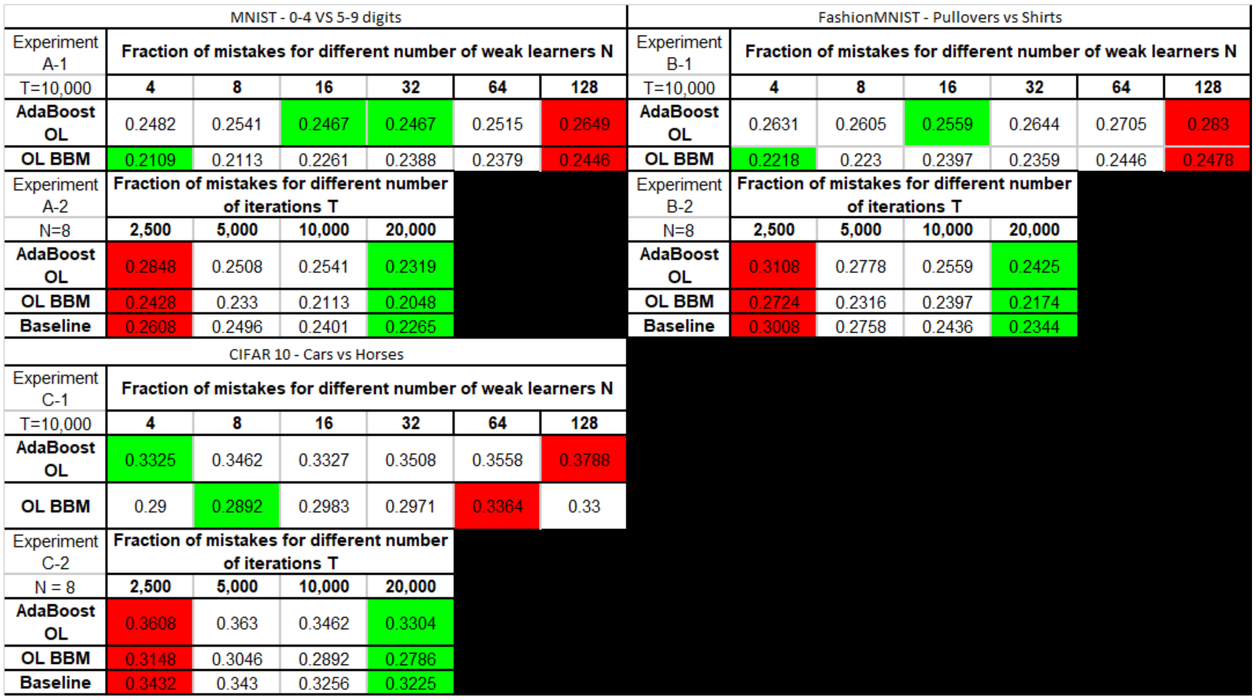
An important difference between the two algorithms is that Online BBM requires the knowledge of γ as a parameter, which is unfortunately not known ahead of time [2]. This is also the property that makes Online BBM non-adaptive. For a given dataset and experiment parametrization, we determine γ by initially guessing a value for it. We then run the experiment and measure the edge of each resulting weak learner. We repeat the experiment, setting as γ the minimum edge of the weak learners of the first experiment. We repeat this process until the minimum γ does not decrease for two successive iterations.

Each experiment is defined on a dataset and a binary classification task on that dataset and is parametrized by the number of iterations T and the number of weak learners N. We verify the implementations of our boosting and perceptron algorithms by performing multiple binary classification tasks between two digit classes on the MNIST datasets and reliably achieving around 94% accuracy with all algorithms, which is fairly high as expected.

6. Experimental Results

For each dataset, we perform two experiments. In experiment 1, we keep T fixed and vary N and in experiment 2 we keep N fixed and vary T. At each experiment, we record the fraction of mistakes made over all iterations and the minimum edge of the weak learners generated by the booster. For experiment 2, we also record the fraction of mistakes made by the baseline. In Figure

1, we present the results of our experiments, omitting the minimum edge values for the sake of clarity. Those values are incorporated on our bound verification calculations presented in Figure 2.



MNIST - 0-4 VS 5-9 digits

Experiment A-1	Fraction of mistakes for different number of weak learners N					
T=10,000	4	8	16	32	64	128
AdaBoost OL	0.2482	0.2541	0.2467	0.2467	0.2515	0.2649
OL BBM	0.2109	0.2113	0.2261	0.2388	0.2379	0.2446

Experiment A-2	Fraction of mistakes for different number of iterations T			
N=8	2,500	5,000	10,000	20,000
AdaBoost OL	0.2848	0.2508	0.2541	0.2319
OL BBM	0.2428	0.233	0.2113	0.2048
Baseline	0.2608	0.2496	0.2401	0.2265

FashionMNIST - Pullovers vs Shirts

Experiment B-1	Fraction of mistakes for different number of weak learners N					
T=10,000	4	8	16	32	64	128
AdaBoost OL	0.2631	0.2605	0.2559	0.2644	0.2705	0.283
OL BBM	0.2218	0.223	0.2397	0.2359	0.2446	0.2478

Experiment B-2	Fraction of mistakes for different number of iterations T			
N=8	2,500	5,000	10,000	20,000
AdaBoost OL	0.3108	0.2778	0.2559	0.2425
OL BBM	0.2724	0.2316	0.2397	0.2174
Baseline	0.3008	0.2758	0.2436	0.2344

CIFAR 10 - Cars vs Horses

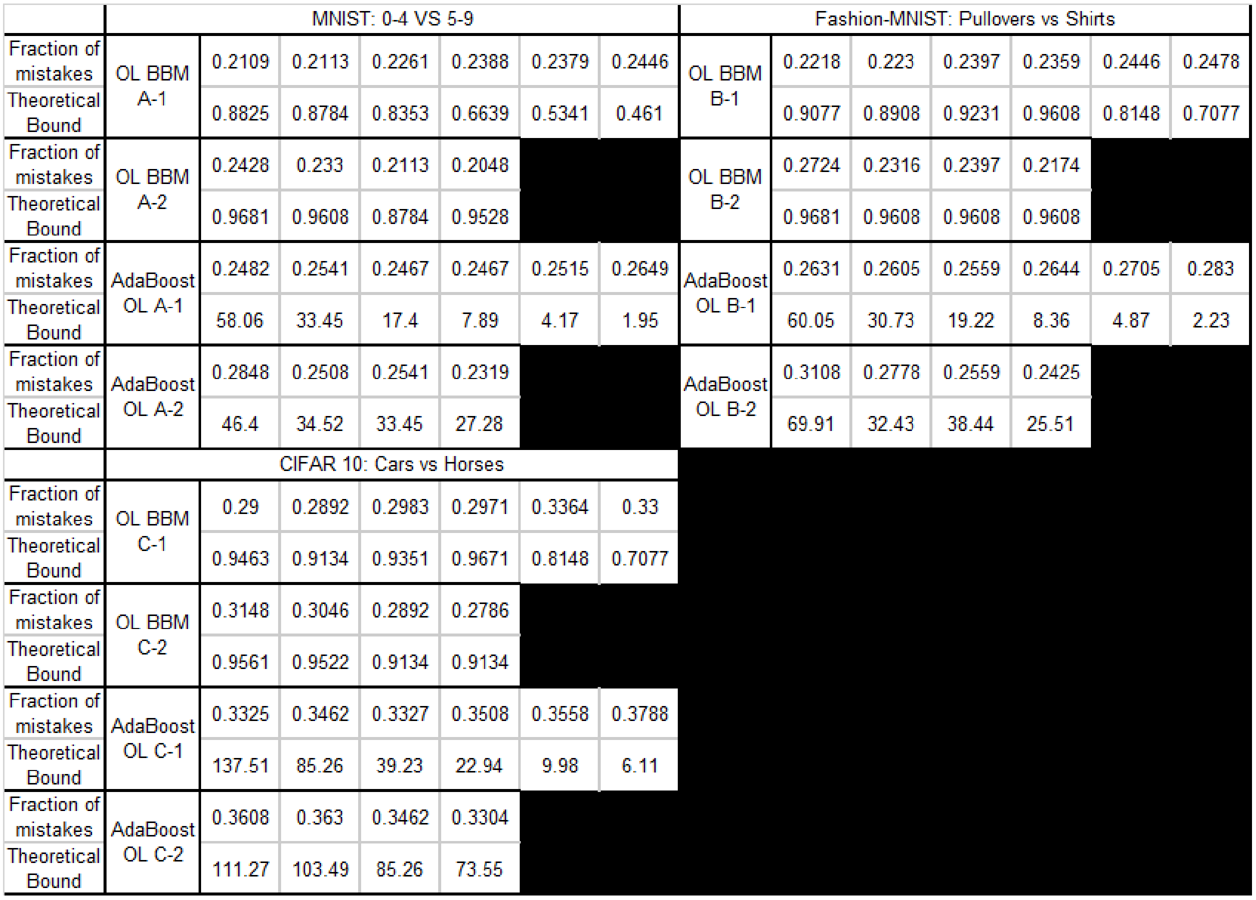
Experiment C-1	Fraction of mistakes for different number of weak learners N					
T=10,000	4	8	16	32	64	128
AdaBoost OL	0.3325	0.3462	0.3327	0.3508	0.3558	0.3788
OL BBM	0.29	0.2892	0.2983	0.2971	0.3364	0.33

Experiment C-2	Fraction of mistakes for different number of iterations T			
N = 8	2,500	5,000	10,000	20,000
AdaBoost OL	0.3608	0.363	0.3462	0.3304
OL BBM	0.3148	0.3046	0.2892	0.2786
Baseline	0.3432	0.343	0.3256	0.3225

Figure 1. Metrics for both experiments performed on each dataset for different values of T and N. For each row, green indicates best performance and red indicates worst performance.

By looking at experiment 2 of each dataset, we observe that across all datasets, Online BBM reliably outperforms both the weak learner and AdaBoost.OL, achieving a 2%-4% boost in performance in all cases. This is consistent with the experimental results found in [2], in which Online BBM also outperformed the rest of the algorithms. Therefore, it seems that setting γ appropriately is not as hard as described in [2], to the extent that it would make an adaptive algorithm, like AdaBoost.OL, outperform Online BBM. The algorithm with the better theoretical bounds does perform better in practice, assuming a proper technique for estimating γ. An open experimental question is whether the adaptive algorithm developed in [4], which is optimal in terms of sample complexity but suboptimal in terms of the number of weak learners, can outperform the non-adaptive Online BBM.

From the results of experiment 1, we also observe that as the number of weak learners becomes very large, performance deteriorates. This is consistent with the hypothesis introduced in [3] by Chen *et al*. that if one boosts too many weak learners at the same time, the outcome may be dominated by the poor predictions made by the many redundant weak learners which do not learn well. This experimental evidence shows the importance of proving both a lower bound and an upper bound on the number of weak learners, as Chen *et al*. had first proposed. In other words, asymptotic complexity for the number of weak learners should be proven using $\Theta()$ bounds, to provide tight upper and lower bounds for the number of weak learners. This is indeed proven for Online BBM in [2], but not for AdaBoost.OL, which is a reason why AdaBoost.OL is not an optimal algorithm.



		MNIST: 0-4 VS 5-9							Fashion-MNIST: Pullovers vs Shirts					
Fraction of mistakes	OL BBM A-1	0.2109	0.2113	0.2261	0.2388	0.2379	0.2446	OL BBM B-1	0.2218	0.223	0.2397	0.2359	0.2446	0.2478
Theoretical Bound		0.8825	0.8784	0.8353	0.6639	0.5341	0.461		0.9077	0.8908	0.9231	0.9608	0.8148	0.7077
Fraction of mistakes	OL BBM A-2	0.2428	0.233	0.2113	0.2048			OL BBM B-2	0.2724	0.2316	0.2397	0.2174		
Theoretical Bound		0.9681	0.9608	0.8784	0.9528				0.9681	0.9608	0.9608	0.9608		
Fraction of mistakes	AdaBoost OL A-1	0.2482	0.2541	0.2467	0.2467	0.2515	0.2649	AdaBoost OL B-1	0.2631	0.2605	0.2559	0.2644	0.2705	0.283
Theoretical Bound		58.06	33.45	17.4	7.89	4.17	1.95		60.05	30.73	19.22	8.36	4.87	2.23
Fraction of mistakes	AdaBoost OL A-2	0.2848	0.2508	0.2541	0.2319			AdaBoost OL B-2	0.3108	0.2778	0.2559	0.2425		
Theoretical Bound		46.4	34.52	33.45	27.28				69.91	32.43	38.44	25.51		

		CIFAR 10: Cars vs Horses					
Fraction of mistakes	OL BBM C-1	0.29	0.2892	0.2983	0.2971	0.3364	0.33
Theoretical Bound		0.9463	0.9134	0.9351	0.9671	0.8148	0.7077
Fraction of mistakes	OL BBM C-2	0.3148	0.3046	0.2892	0.2786		
Theoretical Bound		0.9561	0.9522	0.9134	0.9134		
Fraction of mistakes	AdaBoost OL C-1	0.3325	0.3462	0.3327	0.3508	0.3558	0.3788
Theoretical Bound		137.51	85.26	39.23	22.94	9.98	6.11
Fraction of mistakes	AdaBoost OL C-2	0.3608	0.363	0.3462	0.3304		
Theoretical Bound		111.27	103.49	85.26	73.55		

Figure 2. Fraction of mistakes made and theoretical upper bound computed for each experiment (the excess loss term is ignored). The gamma values used to compute the theoretical upper bounds are not presented but can be requested from the authors.

The theoretical bound we used for Online BBM is the one presented in Theorem 3. If we divide both sides by T, then, for relatively small N and large T, Theorem 3 implies that the fraction of mistakes made by the algorithm is upper bounded by $\exp\left(-0.5N\gamma^2\right)$. Similarly, for AdaBoost.OL, by using Theorem 4 and dividing both sides by T, we get that the fraction of mistakes made by the algorithm is upper bounded by $\frac{8}{\gamma^2 N}$ for relatively small N and large T. These are the theoretical upper bounds used to compute the results presented in Figure 2.

We observe that in every experiment, the bounds provided in Theorems 3 and 4, even if we ignore the second term, are correct upper bounds. However, both bounds are very loose and, for the particular choices of N and T, they have little practical value, as the guarantees they provide are trivially possible to achieve using random guesses. This is especially the case for AdaBoost.OL, for which the theoretical upper bounds computed for the fraction of mistakes are greater than 1.

If we focus on experiments A-1, B-1, and C-1 for both algorithms, we notice that the theoretical bound computed decreases as N increases. In fact, when we increased N by a large enough amount in an experiment not listed in Figure 1, the theoretical upper bound dipped below the fraction of mistakes any of our algorithms made for $T = 10,000$. However, this is still consistent with the theoretical upper bounds, since, as N becomes large, the second term of the upper bounds becomes very significant and we can no longer ignore it, like we did when computing the theoretical upper bound values presented in Figure 2. This shows that the excess loss term-second term of the theoretical upper bounds is indeed factors in for the fact that an online algorithm cannot perform that well if it has a large number of weak learners, as conjectured by Chen *et al.* and verified by our first experiment.

7. Conclusions

Our experiments show that AdaBoost.OL performs worse than Online BBM, which shows that the optimal algorithm does perform better in practice, disputing our first experimental hypothesis. They also verify that the bounds in Theorems 3 and 4 introduced by Beygelzimer *et al.* hold for any values of T and N and they satisfy the property that as N becomes very large, the upper bound initially decreases, but eventually increases back up, confirming the Chen hypothesis that using too many weak learners can hinder performance. However, the theoretical bounds do not provide much practical value, as it is really hard to estimate the correct constants to get bounds that provide guarantees on the actual performance metrics, as shown by our experiments.

Overall, the online boosting framework introduced by Beygelzimer *et al.* in [2] seems to be complete. Online BBM appears to be an online boosting algorithm that works well in practice. The same though cannot be said for AdaBoost.OL, which is even outperformed by the weak learner in certain cases. Therefore, it seems optimality on the bounds for sample complexity and number of weak learners is important for practical performance. Thus, a next step in the field of online boosting would be the development of an adaptive and optimal algorithm that does not require determining γ before training, which is the main weakness of the Online BBM algorithm.

8. Acknowledgements

We thank Professor Schapire for guiding us in our topic selection process. We also thank Professor Haipeng Luo for assisting us in our literature review and for enabling this project with his work on Online Boosting.

References

[1] A. Beygelzimer, E. Hazan, S. Kale, and H. Luo. Online gradient boosting. *CoRR*, abs/1506.04820, 2015.

[2] A. Beygelzimer, S. Kale, and H. Luo. Optimal and adaptive algorithms for online boosting. *CoRR*, abs/1502.02651, 2015.

[3] S.-T. Chen, H.-T. Lin, and C.-J. Lu. An Online Boosting Algorithm with Theoretical Justifications. *ArXiv e-prints*, June 2012.

[4] D. J. Foster, S. Kale, H. Luo, M. Mohri, and K. Sridharan. Logistic regression: The importance of being improper. *CoRR*, abs/1803.09349, 2018.

[5] Y. H. Jung, J. Goetz, and A. Tewari. Online Multiclass Boosting. *ArXiv e-prints*, Feb. 2017.

[6] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).

[7] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[8] N. Oza and S. Russell. Online bagging and boosting. *In Eighth International Workshop on Artificial Intelligence and Statistics*, pages 105–112.

[9] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[10] R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[11] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

Appendix - Algorithms

```
Algorithm 1 Online BBM
for $t = 1$ to $T$ do
    Receive example $\mathbf{x}_t$.
    Predict $\hat{y}_t = \text{sign}(\sum_{i=1}^{N} \text{WL}^i(\mathbf{x}_t))$, receive label $y_t$.
    Set $s_t^0 = 0$.
    for $i = 1$ to $N$ do
        Set $s_t^i = s_t^{i-1} + y_t \text{WL}^i(\mathbf{x}_t)$.
        Set $k_t^i = \lfloor \frac{N-i-s_t^{i-1}+1}{2} \rfloor$.
        Set $w_t^i = \binom{N-i}{k_t^i} \left(\frac{1}{2} + \frac{\gamma}{2}\right)^{k_t^i} \left(\frac{1}{2} - \frac{\gamma}{2}\right)^{N-i-k_t^i}$.
        Pass training example $(\mathbf{x}_t, y_t)$ to $\text{WL}^i$
        with probability $p_t^i = \frac{w_t^i}{\|\mathbf{w}^i\|_\infty}$.
    end for
end for
```

Figure 3. Algorithmic steps of the Online BBM algorithm. WL stands for the weak learner used in this algorithm. Figure borrowed from [2].

```
Algorithm 2 AdaBoost.OL
Initialize: $\forall i : v_1^i = 1, \alpha_1^i = 0$.
for $t = 1$ to $T$ do
    Receive example $\mathbf{x}_t$.
    for $i = 1$ to $N$ do
        Set $\hat{y}_t^i = \text{sign}(\sum_{j=1}^{i} \alpha_t^j \text{WL}^j(\mathbf{x}_t))$.
    end for
    Randomly pick $i_t$ with $\Pr[i_t = i] \propto v_t^i$.
    Predict $\hat{y}_t = \hat{y}_t^{i_t}$, receive label $y_t$.
    Set $s_t^0 = 0$.
    for $i = 1$ to $N$ do
        Set $z_t^i = y_t \text{WL}^i(\mathbf{x}_t)$.
        Set $s_t^i = s_t^{i-1} + \alpha_t^i z_t^i$.
        Set $\alpha_{t+1}^i = \Pi\left(\alpha_t^i + \frac{\eta_t z_t^i}{1+\exp(s_t^i)}\right)$ with $\eta_t = 4/\sqrt{t}$.
        Pass example $(\mathbf{x}_t, y_t)$ to $\text{WL}^i$ with probability[5]
        $p_t^i = w_t^i = 1/(1 + \exp(s_t^{i-1}))$.
        Set $v_{t+1}^i = v_t^i \cdot \exp(-\mathbf{1}\{y_t \neq \hat{y}_t^i\})$.
    end for
end for
```

Figure 4. Algorithmic steps of the AdaBoost.OL algorithm. WL stands for the weak learner used in this algorithm. Figure borrowed from [2].