Curtailed Online Boosting

Raphael Pelossof
Columbia University
500 W 120th St, New York, NY 10027
pelossof@cs.columbia.edu

Michael Jones
MERL
201 Broadway, Cambridge, MA 02139
mjones@merl.com

Abstract

The purpose of this work is to lower the average number of features that are evaluated by an online algorithm. This is achieved by merging Sequential Analysis and Online Learning. Many online algorithms use the example's margin to decide whether the model should be updated. Usually, the algorithm's model is updated when the margin is smaller than a certain threshold. The evaluation of the margin for each example requires the algorithm to evaluate all the model's features. Sequential Analysis allows us to early stop the computation of the margin when uninformative examples are encountered. It is desirable to save computation on uninformative examples since they will have very little impact on the final model. We show the successful speedup of Online Boosting while maintaining accuracy on a synthetic and the MNIST data sets.

1. Introduction

Many Online Algorithms base their model update on the margin of each example in the stream. Online algorithms such as Kivinen and Warmuth's Exponentiated Gradient [12] and Oza and Russell's Online Boosting [17] update their respective models by using a margin based potential function. Passive online algorithms, such as Rosenblatt's perceptron [20] and Crammer *et al.*'s online passive-aggressive algorithms [5], define a margin based filtering criterion for update, which only updates the algorithm's model if the value of the margin falls below a defined threshold. All these algorithms fully evaluate the margin for each example, which means that they evaluate all their features for every example.

The running time of these algorithms is linear in the number of features, or the dimensionality of the input space. Since models today may have thousands of features, this running time seems daunting, and depending on the task, one might wish to speed up these online algorithms, by pruning uninformative examples. We propose to early stop the computation of feature evaluations for uninformative examples by connecting Sequential Analysis [24, 14] to Online margin based algorithms.

We use Sequential Analysis to lower the number of features evaluated for each example in the stream. Many of the online algorithms described above will perform an insignificant model update on uninformative examples. These examples are usually easily classifiable and will have a large positive margin. By early stopping the calculation of the margin on these easy to classify examples we speed up the online algorithm. Our framework allows the online algorithm to early stop computation on uninformative examples, thereby concentrating most of the computational effort on highly informative important examples.

Our approach of using Sequential Analysis breaks up the calculation of the margin of each example in the stream. This allows the algorithm to make a decision after the evaluation of each feature whether the next feature should also be evaluated or the feature evaluation should be stopped. We use the terms margin and full margin to describe the summation of all the feature evaluations, and partial margin as the summation of a part of the feature evaluations. The margin is computed as a weighted sum of feature evaluations of an example. This decision making process allows us to early stop the evaluation of features on examples with a large partial margin after having evaluated only a few features. Examples with a large partial margin are unlikely to have a full margin below the required threshold. Therefore, by rejecting these examples early, large savings in computation are achieved.

Many discriminative machine learning algorithms use margin based functions as a basis for their optimization. AdaBoost for example, maximizes the margins of the examples in the training set at each iteration. An example's margin is the sum of its classification results weighted by the confidence rate of each weak hypothesis. The sum can be viewed as a random walk with varying step sizes, where each step size is set as the weight of the equivalent vote in the sum and where the classification result is the direction. When training, AdaBoost weights more heavily examples with a negative margin, and therefore concentrates on clas-

sifying those examples correctly. For large data sets it is time consuming to evaluate the margin for all the examples. It is more efficient to speed up the process by filtering examples which already have a large positive margin after been evaluated by few weak hypotheses. Similarly, examples with a very large margin will have little effect in later rounds of Online Boosting. The algorithm can be sped up by not performing any update when these examples are encountered. Filtering examples can present a large advantage when considered for online boosting algorithms.

In this work we propose a very simple sequential test to early stop the computation of the margin. Sequential Analysis has been an active research field for over 60 years. It is mostly used and developed by the Clinical testing and Economics communities. In clinical testing the researcher would like to design a test which requires the smallest number of patients to prove the efficacy of a drug. The cost of many tests is monetarily high and patients may die throughout the test, therefore the tests are designed sequentially, so that the minimum number of patients will be tested. The connection between the clinical tests and margin evaluation is done by looking at each feature as a patient, and checking whether a test statistic is significant enough after each feature evaluation. If at any interim point in the test there isn't enough significance for the test, we will early stop the test, reject the example, and save on computational costs. We demonstrate that this simple procedure can speed up Online Boosting by an order of magnitude.

2. Related Work

Curtailed Online Boosting is closely related to the sampling and threshold selection techniques used by cumulative classifiers. Cumulative classifiers are trained with sampled data sets, and set early rejection score based thresholds. The sampling is done as a function of the margin. Curtailed Online Boosting sets margin based early stopping thresholds in an online manner.

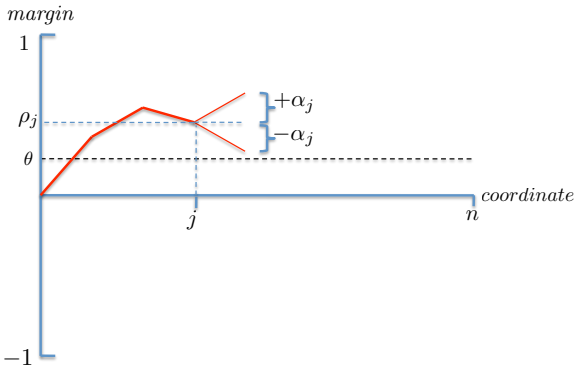
The most famous and widely accepted face detector is the very successful Viola and Jones cascaded face detector [23, 11]. The Viola and Jones cascaded face detector is a discriminative classifier that is trained using a batch algorithm similar to batch AdaBoost [9]. Extending the cascaded classifier is the cumulative classifier proposed by Bourdev and Brandt [1]. The two main differences between the training process of the cumulative classifier and batch AdaBoost are re-sampling and the setting of quick rejection thresholds. Re-sampling is used to narrow down the set of examples the algorithm is trained on at each iteration due to the extremely large size of the training set. The full training set may contain over 200 million examples. Quick rejection thresholds are thresholds which stop the processing of examples that can be easily classified as non faces, thereby making the detector extremely fast using selective processing [1, 25].

The training process incorporates multiple stages, first a boosted classifier is created by training AdaBoost on a frontal face data set. At each boosting round the algorithm re-samples a set of new examples for the next round. This subset usually consists of progressively harder to classify examples. At each round a feature is selected, and AdaBoost assigns a weight to it. This process repeats until a desired classification accuracy rate has been met. When the training process terminates, early stopping thresholds are selected. These thresholds are an integral part of the cumulative classifier, since they cut down the amount of processing used by the classifier by a few orders of magnitude. The thresholds are set to early stop processing of easy to classify negative examples. Since most of the examples are usually negative in face detection, the thresholds speed up the detector about 200 times on average.

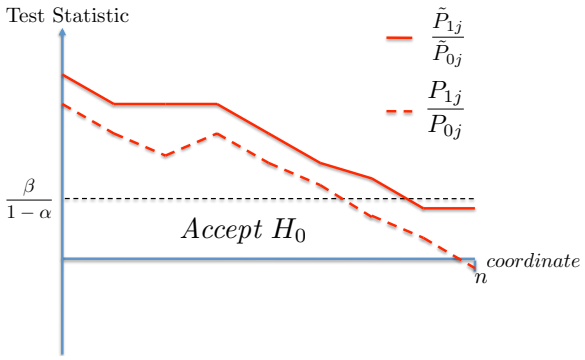
When training their AdaBoost based detector, both Jones and Viola [11] and Bourdev and Brandt [1] use re-sampling to reduce their training set from 200 million examples to 20 thousand. While maintaining high accuracy this process speeds up feature selection and the training time by a few orders of magnitude. In the online setting it's not possible to sample since the algorithm can only keep one example at a time in the memory. The equivalent to sampling in the online setting is filtering. In filtering an example is selected for processing if it matches a certain criteria. We would like the filtering process and the sampling process to select a similar set of examples. Similarly to sampling, we filter each example according to its margin.

Another closely related learning paradigm to filtered online boosting is selective sampling active learning. In selective sampling the active learning algorithm is presented with a set of unlabeled examples and it decides which examples labels to query at a cost. The algorithm's task is to pay for labels as little as possible while achieving specified accuracy and reliability rates [6, 3, 22]. Typically, for selective sampling active learning algorithms the algorithm would ignore examples that are easy to classify, and pay for labels for harder to classify examples that are close to the decision boundary [6]. We will use the motivation behind active learning to select which examples to update our classifier with.

In a supervised setting, we do not actually have to pay for labels, however similarly to selective sampling active learning we would like to save computation by updating our model only on hard to classify exmaples. Previous work on selective sampling [4, 10, 3] put emphasis on querying examples that are hard to classify, and therefore have either a small absolute margin, or a negative margin. Examples with a large absolute margin will be discarded and their label is not requested, equivalently, our model is not updated on such uninformative examples.



(a) Margin evaluation as a random walk. At coordinate j feature j can vote either $+\alpha_j$ or $-\alpha_j$. We are interested in calculating the probability that the random walk after n steps will end below the threshold θ given the current position ρ_j. Equivalently, we want to calculate what is the probability that the full margin ρ_n is smaller than a given threshold θ given the current margin ρ_j. This probability naturally changes after every step of the random walk.



(b) Sequential testing of the test statistic. At each coordinate the sequential test requires a sufficient amount of confidence in order to continue testing. Feature evaluation stops when this requirement no longer holds. We upper bound the actual statistic defined by the sequential test, which results in a slightly less efficient test, however the required accuracy is maintained.

Figure 1. Curtailed Online Boosting as a Random Walk

To speed up the computation we look at the computation of the margin in more detail. The margin is computed as a dot product between the hypothesis vector and an example. In the case of Online Boosting the margin is computed as a sum of weak hypotheses votes given an example. With high dimensional data, or large sets of weak hypotheses evaluating this sum exactly may be very expensive. By looking at the evaluation of the margin as a random walk, we are able to compute traversal probabilities which are required by Sequential Analysis. The view of the margin evaluation as a random walk was presented and discussed previously by Freund and later by Long and Servedio [8, 15, 16]. Breaking up the margin as it's computed by boosting algorithms to its summands, we can view the margin computation as a random walk, where each hypothesis' vote is a random step, and the hypothesis' index is time. This view allows us to use probabilistic bounds to predict the value of the margin given a partial computation of the margin.

3. Online prediction of the margin

Our task is to find a filtering framework that would speedup Online Boosting by quickly rejecting examples of little importance. We measure the importance of an example by the size of its margin, the distance from the decision boundary geometrically. We define θ as the importance threshold, where examples that are important to us have a margin smaller than θ. This approach in a different setting was proposed by Bradley et al. [2].

We would like to bound the probability of the margin being smaller than θ given the margin computed so far. Let the margin of an example (x_i, y_i) at coordinate n be defined by

$$\rho_n = y_i \sum_{j=1}^{n} \alpha_j h_j(x_i) = \sum_{j=1}^{n} \alpha_j u_{ij},$$

where α_j is the weight assigned to the jth weak hypothesis h_j by some online algorithm, and $u_{ij} = y_i h_j(x_i)$. Let the label of the i'th example be defined by $y_i \in \{-1, 1\}$, the classification of the j'th weak hypothesis be defined by $h_j \in \{-1, +1\}$, and the weight assigned to the j'th weak hypothesis $\alpha \in R$. We define ρ_n as the full margin, ρ_j as the partial margin, and $\rho_{n-j} = \rho_n - \rho_j$ as the remaining margin. Once we computed the partial margin at coordinate j we know its value ρ_j. Our task is to bound the following probability (see Figure 1(a):

$$\begin{aligned} & P(\textit{reject example}|\textit{current margin}) && (1)\\ = \;& P(\rho_n \geq \theta|\rho_j) && (2)\\ = \;& P(\rho_n - \rho_j \geq \theta - \rho_j). && (3) \end{aligned}$$

Hoeffding's inequality upper bounds the probability of a sum S of random variables s_j deviating from its expectation by more than t where the summands are bounded

$s_j \in [a_j, b_j]$

$$P(S - E[S] \geq t) \leq exp\left(-\frac{2t^2}{\sum_{j=1}^{n}(b_j - a_j)^2}\right). \quad (4)$$

To apply this bound to margins we need to convert equation 3 to the form used by the Hoeffding bound:

$$P(\rho_{n-j} \geq \theta - \rho_j) = P(\rho_{n-j} - E[\rho_{n-j}] \geq \theta - \rho_j - E[\rho_{n-j}]). \quad (5)$$

We need to compute a new adaptive threshold t to match the bound. Let $t = \theta - \rho_j - E[\rho_{n-j}], a_j = -\alpha_j, b_j = \alpha_j$. Combining equations 3 - 5 we get the following inequality

$$P(\rho_n \geq \theta|\rho_j) \quad \leq \quad \exp\left(-\frac{(\theta - \rho_j - E[\rho_{n-j}])^2}{2\sum_{j=1}^{n}\alpha_j^2}\right) \quad (6)$$

Where threshold θ, the margin computed so far ρ_j, and the weights α_j are given known. This upper bound will decrease if the expected full margin $\rho_j + E[\rho_{n-j}]$ is too far from the specified threshold θ in either the positive or negative direction. However, we would like examples with a large negative margin to have a high probability of ending below the threshold theta. We can further upper bound this quantity:

$$\exp\left(-\frac{(\theta - \rho_j - E[\rho_{n-j}])^2}{2\sum_{j=1}^{n}\alpha_j^2}\right) \leq \exp\left(-\frac{\max(\theta - \rho_j - E[\rho_{n-j}], 0)^2}{2\sum_{j=1}^{n}\alpha_j^2}\right) \quad (7)$$

Extending this upper bound will end up giving us higher probabilities for large negative margins.

What is left to calculate the upper bound is to compute the expectation of the remaining margin $E[\rho_{n-j}]$. For simplicity we will assume that the probability of each weak hypothesis to vote either -1 or $+1$ is 0.5, and therefore the expectation is $E[\rho_{n-j}] = 0$. Further modeling about the statistics of margins can be incorporated to give tighter bounds, however, experimentation shows that this simple assignment works very well. Finally we get:

$$P(\rho_n \geq \theta|\rho_j) \quad \leq \quad \exp\left(-\frac{\max(\theta - \rho_j - E[\rho_{n-j}], 0)^2}{2\sum_{j=1}^{n}\alpha_j^2}\right) \approx \exp\left(-\frac{\max(\theta - \rho_j, 0)^2}{2\sum_{j=1}^{n}\alpha_j^2}\right). \quad (8)$$

If we L2 normalize the votes then the denominator in 8 is equal to 1. We can compute $||\alpha||$ in an online fashion by updating every time a vote is updated.

4. Curtailed Online Boosting

In his ground breaking paper [24] Wald describes a nearly optimal stopping algorithm for hypothesis testing. The optimal stopping tests are designed to early reject the null hypothesis. Stochastic Curtailment was later introduced by Lan at al. [14] as a method to incorporate information sequentially as the test progressed. We can use these tests for early stopping margin calculations.

Deterministic curtailment of the margin calculations can be computed in an online manner. The curtailment will stop the sequential computation of the margin if it's not possible at coordinate j for the margin to end up below the specified threshold θ (see figure 1(a).) At each coordinate we can compute the largest possible remaining vote $\max(\rho_{n-j}) = \sum_{k=n-j+1}^{n}|\alpha_j|$, and test whether $\rho_j - \max(\rho_{n-j}) \leq \theta$. If this doesn't hold, then it's not possible for the full margin to be smaller than the required threshold given the current threshold and the remaining features to be evaluated. Stochastic curtailment extends this idea of early stopping by introducing a probabilistic framework to the testing procedure.

With a slight abuse of mathematical notation we can define two hypotheses as:

$$\begin{aligned} H_0 &: \rho_n \geq \theta|\rho_j \\ H_1 &: \rho_n < \theta|\rho_j, \end{aligned} \quad (9)$$

where we would like to test the probability of the entire margin ρ_n being larger than a certain given threshold after having evaluated the partial margin ρ_j. The intuition is that if at a certain point the probability is extremely low for the full margin to be below the required threshold, we stop calculations on that specific example, reject the alternative hypothesis H_1 and accept H_0.

Wald defines the stopping rule as:
Accept H_1 if

$$\frac{p_{1j}}{p_{0j}} \geq A = \frac{1}{\alpha} \quad (10)$$

Accept H_0 if

$$\frac{p_{1j}}{p_{0j}} \leq B = \frac{\beta}{1 - \alpha}, \quad (11)$$

where p_{1j} is the likelihood that the margin satisfies the alternative hypothesis, and p_{0j} is the likelihood the the margins satisfies the null hypothesis. The upper bound on the Type I errors, rejecting H_0 when H_0 is true, is given by α, and Type II errors, accepting H_0 when H_1 is true, by $1 - \beta$. We are more interested in type II errors (β) since we would rather pay the penalty of more computation than erroneously not update our model on informative examples.

Without making assumptions on the underlying distribution which generates the hypotheses and the data, we don't know these exact quantities, however, we can upper bound

them. We know from equation 7:

$$p_{1j} = P(\rho_n \geq \theta|\rho_j) \leq \exp\left(-\frac{\max(\theta-\rho_j-E[\rho_{n-j}],0)^2}{2\sum_{j=1}^{n}\alpha_j^2}\right). \quad (12)$$

Let us define $\tilde{p}_{1j} = \exp\left(-\frac{\max(\theta-\rho_j-E[\rho_{n-j}],0)^2}{2\sum_{j=1}^{n}\alpha_j^2}\right)$, and $\tilde{p}_{0j} \doteq 1-\tilde{p}_{1j}$. We know $p_{0j} = 1-p_{1j}$, and $p_{1j} \leq \tilde{p}_{1j}$, therefore $p_{0j} \geq 1-\tilde{p}_{1j}$, which is equivalent to $p_{0j} \geq \tilde{p}_{0j}$. Combining both inequalities we get the stopping rule for the margin calculation (see figure 1(b))

$$\frac{p_{1j}}{p_{0j}} \leq \frac{\tilde{p}_{1j}}{\tilde{p}_{0j}}.$$

If we only consider a one sided early stopping test, we have the following weaker early stopping rule:

$$\textit{Accept } H_0 \textit{ if } \quad \frac{\tilde{p}_{1j}}{\tilde{p}_{0j}} \leq \frac{\beta}{1-\alpha}.$$

By conducting the test at each evaluation of a hypothesis we are actually increasing the difficulty of the test [19, 13, 7]. To overcome this problem we can use a progressively harder test. The test will start with less stringent constraints on continuation, which will become more stringent as more features are evaluated. We can therefore consider the following interim test at each feature evaluation

$$\textit{Accept } H_0 \textit{ if } \quad \frac{\tilde{p}_{1j}}{\tilde{p}_{0j}} \leq \frac{\beta}{1-\alpha} + \frac{\log j}{\log n} - 1.$$

If the margin is fully evaluated then our new test obeys the original test's error requirements. If the modified test early terminates then the original test has already terminated since our test is a weaker one, which means that the error requirements are still maintained. In our experimentation we did not need to modify the test, however it is actually more stringent than the error requirement imposed by α.

The fact that we set $E(\rho_{n-j}) = 0$ violates the Hoeffding bound. However, in experimentation, we see that the type II error rates (β) are approximately accurate, and that we still get an order of magnitude speedup. Curtailed Online Boosting is detailed in algorithm 1, with L2 normalized votes. It is based on a slightly modification of Online Boosting to the more prevalent AdaBoost weighting rule [21, 18]. The algorithm does not detail the bayesian feature selection for clarity. However, it can easily be incorporated similarly to Online Boosting.

Algorithm 1 Curtailed Online Boosting

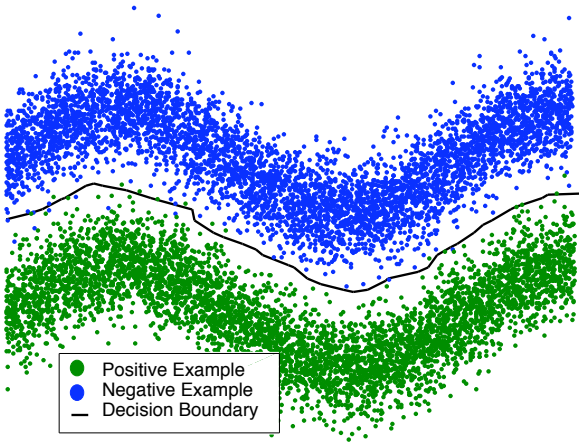
Input: $h_1, ..., h_n; (x_1, y_1), ..., (x_m, y_m), \epsilon, \theta, \alpha, \beta$
Initialize: $w_j^+ = \epsilon, w_j^- = \epsilon, \alpha_j = 0, j = 1, .., n, z_1 = 1$
Define $u_{ij} = y_i h_j(x_i)$
for $i = 1$ **to** m **do**
  $d_1 = 1$
  $\rho_0 = 0$
  for $j = 1$ **to** n **do**
    $\rho_j = \rho_{j-1} + \alpha_j u_{ij}/\sqrt{z_i}$
    $\tilde{p}_{1j} = \exp\left\{-0.5\max(\theta-\rho_j,0)^2\right\}$
    if $\frac{\tilde{p}_{1j}}{1-\tilde{p}_{1j}} \leq \frac{\beta}{1-\alpha}$ **then**
      Continue (exit for loop)
    else
      $w_j^- \leftarrow w_j^- + d_j \mathbf{1}_{[u_{ij}=-1]}$
      $w_j^+ \leftarrow w_j^+ + d_j \mathbf{1}_{[u_{ij}=+1]}$
      $\alpha_j = \frac{1}{2}\log\frac{w_j^+}{w_j^-}$
      $d_{j+1} = d_j e^{-\alpha_j u_{ij}}$
    end if
  end for
  update $z_i = \sum_{k=1}^{n}\alpha_k^2$ by updating coordinates $1..j$
end for
Output: $\alpha_1, ..., \alpha_n$

5. Experiments

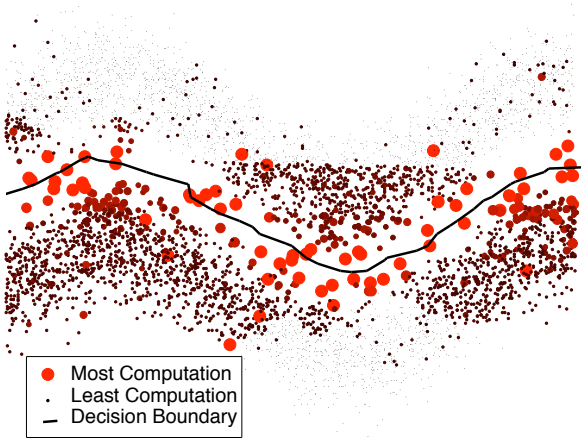
We set two experiments to test the speed and accuracy of Curtailed Online Boosting. The first a synthetic experiment which enables us to see how the algorithm spends it computational power. The second, a real world experiment - the MNIST dataset, which shows the speed advantage of Curtailed Online Boosting.

5.1. Synthetic data

The synthetic experiment was set up to test the computational efficiency of Curtailed Online Boosting. Two translated sin waves were sampled from to create a set of positive and negative examples. These examples were then split randomly to the training and test sets. Each of the sets contains $100,000$ examples. First AdaBoost was trained on the sets to obtain a set of 100 features. The features we used are thresholded planes. These features are not online-learnable, and therefore were set beforehand. Figure 2(a) shows a random subset of 5000 examples from the test set and the resulting decision boundary found by Curtailed Online Boosting. We set $\theta = 0, \alpha = 0.95, \beta = 0.8$. The algorithm fully calculated the margin of only 1233 examples out of the $100,000$ which is about 1% of the training set. The average number of hypotheses evaluated per example is 18 out of 100 which is a $5x$ speedup over Online Boosting. Throughout the learning process, only 282 examples actually had a margin below the required threshold of 0, out of which 6% were not fully evaluated due to early stopping. This is far below the type II error rate of 20% that we set. Figure 2(b) shows an efficiency map of the algorithm. Plotted are 5000 randomly chosen training points. The figure



(a) The synthetic two sin waves learning problem. Curtailed Online Boosting is trained with 100 boosting stumps and run over $100,000$ examples. The stumps used are thresholded linear separators which were selected using AdaBoost. Plotted is part of the test set, in blue the positive class, in red the negative class, and the learned decision boundary in black.



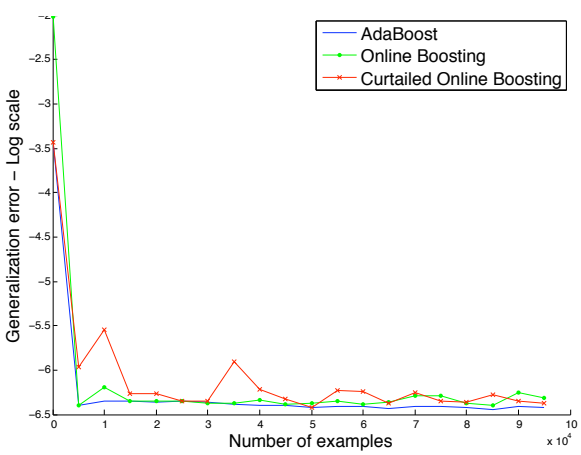
(b) **Computational Efficiency Plot.** The size and color of each point shows the amount of features evaluated by Curtailed Online Boosting throughout the learning process. The larger and more red a point is, the more features were evaluated and updated while processing it. The figure shows that the algorithm puts the most computational effort when processing examples by the decision boundary. A subset of 5000 random training points are plotted. The jumps in the computation allocation are due to the structure of the decision surface.

Figure 2. Synthetic experiment.

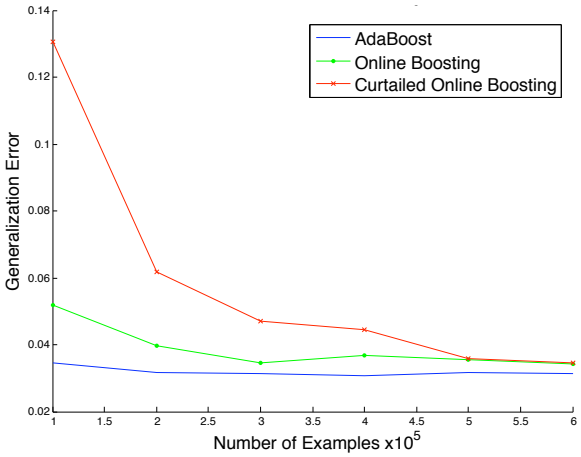
shows how most of the computation is allocated to examples that are on the decision boundary, and substantially less computation is allocated to examples that are easily classifiable. Figure 3(a) compares the test error of AdaBoost, Online Boosting, and Curtailed Online Boosting as training progresses. All three algorithms had the weak hypotheses fixed throughout training for comparison.

5.2. MNIST

The MNIST dataset consists of 28×28 images of the digits $[0, 9]$. The dataset is split into a training set which includes $60,000$ images, and a test set which includes $10,000$ images. All the digits are represented approximately in equal amount in each set. Similarly to the synthetic experiment, we trained a classifier in an offline manner with sampling to find a set of weak hypotheses. When training we normalized the images to have zero mean and unit variance. We used $h_j(x) = sign(\|x_j - x\|_2 - \tau)$ as our weak hypothesis. The weak learner found for every boosting round the vector x_j and threshold τ that create a weak hypothesis which minimizes the training error. As candidates for x_j we used all the examples that were sampled from the training set at that boosting round. We partitioned the multiclass problem into 10 one-versus-all problems, and defined a meta-rule for deciding the digit number as the index of the classifier that produced the highest vote. Each of the digit classifiers was trained with $1,000$ features. We set $\alpha = 0.95, \beta = 0.8$, and θ was set individually for each classifier in the range $\theta \in [0, -0.5]$. The generalization error rate of the combination rule using each of the methods can be seen in figure 3(b). The generalization error rates for each classifier can be seen in table 1. At the beginning of the training process the votes α_j are a very bad estimate of the end votes. This causes the random walk to be highly unreliable, and the curtailment process inefficient. We therefore initialized our model with a small batch of the first $5,000$ examples to avoid these estimation errors. The efficiency results in table 1 exclude the full margin computation of the initial $5,000$ out of $60,000$ examples. If we were to add these examples then the average number of features evaluated would increase by about 66 features. The filtering error rate is not affected since we fully compute the margin of these examples. The results show that an order of magnitude speedup is possible while maintaining generalization



(a) Synthetic: Test error for the different boosting algorithms as more examples are trained on. Online Boosting and Curtailed Online Boosting approach the accuracy of batch AdaBoost as more examples are presented. However, Curtailed Online Boosting performs a full model update only on 1% of the examples.



(b) MNIST: Combined classifier test error as the number of training examples is increased. Online Boosting and Curtailed Online Boosting converge to the same error rate, however Curtailed Online Boosting is about 7 times faster.

Figure 3. Generalization Error on synthetic and MNIST data sets.

Table 1. MNIST test error in % for each classifier, and curtailment efficiency

Classification Error in %										
Digit	0	1	2	3	4	5	6	7	8	9
AdaBoost, 1000 features	0.31	0.19	0.8	0.89	0.9	1	0.47	0.79	1.6	1.3
Online Boosting, 1000 features	0.35	0.27	0.79	1	0.85	1	0.55	0.97	1.8	1.4
Online Boosting, 200 features	0.47	0.28	1.55	1.69	1.60	1.61	0.85	1.31	3.61	2.34
Curtailed Online Boosting	0.43	0.31	0.84	1.1	1.1	0.99	0.74	0.94	1.4	1.4
Curtailed Online Boosting Computational Efficiency										
Avg. Num. Features Evaluated	145	175	135	153	108	141	107	101	202	150
Speedup	7	6	7	7	9	7	9	10	5	7
Type II errors in %	22	20	21	19	11	18	23	20	15	14

accuracy.

6. Discussion and Future Work

We showed that by using Sequential Analysis we can come up with a very simple rule to speed up online algorithms which maintains accuracy. However, we did not estimate the expected remaining margin accurately, which leaves room for improvement of the testing procedure. By making a few assumptions on the weak hypotheses and the margin distribution, we believe that a bound on the average number of features evaluated can be obtained. Furthermore, by obtaining upper and lower bounds on the random walk, we'll be able to use both of Wald's stopping rules. Another improvement would be to extend the inequalities to deal with absolute scores instead of signed margins, thereby making the algorithm more robust to noise.

References

[1] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition*, volume 2, pages 236–243, Washington, DC, USA, 2005. IEEE Computer Society. 2

[2] J. K. Bradley and R. E. Schapire. Filterboost: Regression and classification on large datasets. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Neural Information Processing Systems*, pages 185–192, Cambridge, MA, 2008. MIT Press. 3

[3] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, 7:1205–1230, 2006. 2

[4] D. Cohn, R. Ladner, and A. Waibel. Improving generalization with active learning. In *Machine Learning*, pages 201–221, 1994. 2

[5] K. C. Crammer, O. Dekel, J. Keshet, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006. 1

[6] S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *In COLT*, pages 249–263, 2005. 2

[7] T. R. Fleming, D. P. Harrington, and P. C. O'Brien. Designs for group sequential tests. *Controlled Clinical Trials*, 5(4, Supplement 1):348 – 361, 1984. 5

[8] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001. 3

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. 2

[10] Y. Freund, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. In *Machine Learning*, pages 133–168, 1997. 2

[11] M. Jones and P. Viola. Face recognition using boosted local features. In *Interantional Conference on Computer Vision*, 2003. 2

[12] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997. 1

[13] K. G. Lan and D. L. DeMetz. Discrete sequential boundaries for clinical trials. *Biometrika*, 70(3):659–663, 1983. 5

[14] K. G. Lan, R. Simon, and M. Halperin. Stochastically curtailed tests in long-term clinical trials. *Sequential Analysis*, 1(3):207–219, 1982. 1, 4

[15] P. M. Long and R. A. Servedio. Martingale boosting. *Learning Theory*, pages 79–94, 2005. 3

[16] P. M. Long and R. A. Servedio. Adaptive martingale boosting. In *Neural Information Processing Systems*, 2008. 3

[17] N. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics*, pages 105–112. Morgan Kaufmann, 2001. 1

[18] R. Pelossof, M. Jones, I. Vovsha, and C. Rudin. Online coordinate boosting. Arxiv, 2008. 5

[19] S. J. Pocock. Group sequential methods in the design and analysis of clinical trials. *Biometrika*, 64(2):191–199, 1977. 5

[20] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 1

[21] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. 5

[22] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 2

[23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, 2001. 2

[24] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. 1, 4

[25] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *International Conference on Computer Vision*, page 709, Washington, DC, USA, 2003. IEEE Computer Society. 2