On Boosting Sparse Parities

Lev Reyzin
Department of Mathematics, Statistics, & Computer Science
University of Illinois at Chicago
Chicago, IL 60607
lreyzin@math.uic.edu

Abstract

While boosting has been extensively studied, considerably less attention has been devoted to the task of designing good weak learning algorithms. In this paper we consider the problem of designing weak learners that are especially adept to the boosting procedure and specifically the AdaBoost algorithm.

First we describe conditions desirable for a weak learning algorithm. We then propose using sparse parity functions as weak learners, which have many of our desired properties, as weak learners in boosting. Our experimental tests show the proposed weak learners to be competitive with the most widely used ones: decision stumps and pruned decision trees.

Introduction

The boosting approach to machine learning is a powerful technique that combines simple rules that are correct with probability slightly better than random guessing into strong predictors. The idea behind boosting originated in Schapire's theoretical result (1990) showing a reduction from strong to weak learnability. Schapire's reduction was eventually made into a practical algorithm called AdaBoost (Freund and Schapire 1997), which remains the most analyzed and most ubiquitous of the many boosting algorithms that have appeared since. AdaBoost works by presenting its input to a weak learning algorithm, getting a hypothesis, reweighing the distribution on its input, and repeating the process again, with the idea that each successive weak learner focuses on the examples that had previously been hardest to classify.

Much effort has been expanded in various directions around boosting. Experimentally, AdaBoost was discovered to be very effective for classification, oftentimes beating concurrent state-of-the-art algorithms. It was also noticed that AdaBoost tended to perform better than initial theory suggested (we discuss this further in the following section), and this observation led to considerable theoretical and experimental investigation. AdaBoost also became the basis for many successful applications. A recent summary of various boosting results appears in (Schapire and Freund 2012).

To optimize for different objectives, new versions of boosting appeared, including LogitBoost (Friedman, Hastie, and Tibshirani 1998), BrownBoost (Freund 2001), LPBoost (Demiriz, Bennett, and Shawe-Taylor 2002), etc. One aspect these boosting algorithms have in common is the **weak learning assumption** – that a weak learning algorithm, used by boosting as a subroutine, can always do better than random guessing on the distributions presented to it. Without this property, boosting would fail to work both in theory and in practice, so ensuring that a boosting algorithm uses a good weak learner is essential to the success of the overall algorithm.

Despite the centrality of weak learning in boosting, the theory of choosing a good weak learner has been given considerably less effort compared to the plethora of work surrounding boosting's other facets. In this work, we aim to draw attention to the important problem of designing good weak learners, which we hope will be the subject of future study, and we give some theory that points to what would make a good weak learner in practice.

We go on to propose a weak learner suggested by the theory – the weak learner based on basic Fourier analysis of boolean functions, namely the low degree characters, or sparse parities. Finally, we experimentally test our learner and compare it to the most widely-used weak learners. While parities are not complicated functions, nor new to learning theory, combining them with boosting turns out to be an interesting and powerful tool.

The goal of this paper, however, is not to advertise using parities per-se, but to present a well-researched position that more serious study of this area is needed.

AdaBoost

AdaBoost, like all boosting algorithms, combines moderately inaccurate prediction rules, taking a weighted majority vote to form a single classifier. On each round, a boosting algorithm generates a new prediction rule and then places more weight on the examples classified incorrectly. Hence, boosting constantly focuses on correctly classifying the hardest examples. A nice overview of boosting appears in (Schapire 2003).

AdaBoost, the most ubiquitous of the boosting algorithms, will be the focus of our paper, though much of this paper is relevant to other boosting algorithms. The

Algorithm 1 AdaBoost (Freund and Schapire 1997)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$,
where $x_i \in X$, $y_i \in Y = \{-1, +1\}$.

Initialize $D_1(i) = 1/m$.

for $t = 1, \ldots, T$ **do**
Train base learner using distribution D_t.
Get base classifier $h_t : X \to \{-1, +1\}$.
Let:
$$\gamma_t = \sum_i D_t(i) y_i h_t(x_i).$$
Choose:
$$\alpha_t = \frac{1}{2} \ln \frac{1 + \gamma_t}{1 - \gamma_t}.$$
Update:
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).
end for

Output the final classifier:
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

AdaBoost algorithm is given in Algorithm 1.

A key parameter of the boosting algorithm is the **edge**, γ, the weak learner can achieve on any distribution AdaBoost generates. If $\epsilon_t < 1/2$ is the error rate of the weak learner on round t on distribution D_t, then

$$\begin{aligned} \gamma_t &\doteq \sum_i D_t(i) y_i h_t(x_i) \\ &= 1 - 2\epsilon_t \\ &\geq \gamma, \end{aligned}$$

which we can lower bound by a global γ.

If AdaBoost is run for $T = \lceil \frac{\ln m}{2\gamma^2} \rceil$ rounds, its generalization error, denoted $\text{err}(H)$, can be shown to be bounded as follows

$$\text{err}(H) \leq \tilde{O}\left(\frac{\ln |\mathcal{H}|}{m\gamma^2} + \ln\left(\frac{1}{\delta}\right) \right), \qquad (1)$$

where δ is the failure probability, m the number of examples, and $\mathcal{H}$ is the hypothesis class used by the weak learning algorithm. It is important to note that the bound above is only one of many explanations for AdaBoost's performance. It was observed that, contrary to the bound above, AdaBoost tends not to **overfit** training data with more rounds of boosting, despite that the combined classifier's complexity increases with every round.

This led to the following bound based on the margin of the prediction of boosting's weighted vote – a quantity that can be seen as measuring the confidence (and correctness) in the prediction of the combined classifier on the training examples. From this angle, Schapire et al. (1998) derived another bound on the generalization error of boosting: for all $\theta > 0$,

$$\text{err}(H) \leq \Pr_S[yH(x) \leq \theta] + \tilde{O}\left(\sqrt{\frac{\ln |\mathcal{H}|}{m\theta^2}} \right), \qquad (2)$$

with the margin represented by the left term to the right of the inequality, in which $\Pr_S[\ldots]$ is the empirical probability with respect to the training sample that the margin is below the threshold θ.

For more detail, we refer the reader to the rich amount of work on alternative explanations of boosting's performance and resistance to overfitting, including, notably, the margins theory itself (Grove and Schuurmans 1998; Schapire et al. 1998; Mason, Bartlett, and Baxter 1998; Reyzin and Schapire 2006).

Properties of a good weak learning algorithm

We can think of the weak learning algorithm working by doing Empirical Risk Minimization (ERM) over some hypothesis class $\mathcal{H}$. That is to say that the algorithm, upon receiving a (possibly weighted) labeled sample returns a hypothesis $h \in \mathcal{H}$ such that

$$h = \arg\min_{h \in \mathcal{H}} \left(\Pr_{x \sim S}[h(x) \neq y] \right).$$

This criterion can be relaxed to finding an *approximate* ERM hypothesis.

With this view in mind, we argue the following properties are desirable for an effective weak learner.

Diversity: A good weak learner will have many hypotheses that *disagree* on a large fraction of examples in their predictions. One reason for this is that AdaBoost's update rule makes the distribution D_{t+1} be such that

$$\begin{aligned} \Pr_{(x,y) \sim D_{t+1}}[h_t(x) = y] &= \Pr_{(x,y) \sim D_{t+1}}[h_t(x) \neq y] \\ &= 1/2. \end{aligned}$$

Hence, all hypotheses highly correlated with h_t will likely have a small edge as well, making the weak learning condition harder to satisfy.

Coverage: We want all parts of the hypothesis space to be "covered" by a weak learner. For instance, we do not want features that are not considered, areas that are sparsely covered, etc. With high coverage, diversity is easier to achieve.

Simplicity: If $\mathcal{H}$ is finite, we want $|\mathcal{H}|$ to be as small as possible. If $\mathcal{H}$ is infinite, we want its VC dimension (or other measures) to be small.

Error: On the one hand, we need that over all distributions D_i generated by AdaBoost, for a fixed $\gamma > 0$,

$$\min_{h \in \mathcal{H}} \text{err}_{D_t}(\mathcal{H}) \leq 1/2 - \gamma,$$

On the other hand, perhaps counter-intuitively, we need h to have nonzero *training error*, otherwise AdaBoost terminates after the first round, in which case boosting is not used at all.

Evaluability: We need the weak learner to be efficiently (e.g. in polynomial-time) evaluable for it to be usable in practice.

Richness: Because `AdaBoost` takes a weighted vote over hypotheses from $\mathcal{H}$, it is desirable that a linear combination of hypotheses from $\mathcal{H}$ is able to represent a rich set of functions.

Optimizability: Finding an approximate ERM over the weak learners should be tractable.

Domingos (2012) argues that "learning = representation + evaluation + optimization." It is unsurprising similar conditions appear to help for weak learning.

We note that some of these conditions reinforce one another – e.g. as in the case of diversity and coverage, where one good way of achieving diversity is to ensure coverage. On the other hand, some of these conditions are in tension – e.g. simplicity and diversity. Designing a good weak learner requires striking a *balance* among these. For instance, the set of all boolean functions, while achieving great diversity, coverage, richness, and efficiency fails the error and simplicity requirements, making for an ineffective weak learner.

Common weak learners

Here we look at two of the most common weak learners used in boosting.

Decision stumps

Decision stumps are simple weak learners that examine only one (binary) feature value and predict either it or its negation. Hence, for n features, there are $2n$ different stumps. These can be thought of as 1-node decision trees.

Despite their simplicity, they have proven useful in practice, perhaps most notably in the celebrated Viola-Jones face-detection algorithm (2004). Decision stumps have had extensive empirical and theoretical experimentation, e.g. (Caruana and Niculescu-Mizil 2006; Reyzin and Schapire 2006). One practical advantage of using decision stumps is that optimization can be done via enumeration, given the small number of hypotheses in the entire class.

Yet, decision stumps are not without drawbacks. Their limited scope can often fail to find high edges, which can yield overall worse performance than SVMs or boosted decision trees on many tasks (Caruana and Niculescu-Mizil 2006).

A related limitation of decision stumps is that because vote of `AdaBoost` (or any ensemble) can, instead of a sum over rounds, be rewritten as a sum over all the weak learners in the finite hypothesis space, with coefficients $\lambda > 0$:

$$H(x) = \text{sign}\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right) = \text{sign}\left(\sum_{j=1}^{|H|} \lambda_j h_j(x)\right),$$

is clearly a linear function in the features when stumps are used as the weak learner.

Hence, many functions (e.g. an XOR on just two features) simply cannot be predicted with reasonable accuracy via a linear combination of any number of features. We note that attempts have also been made to overcome this limitation of stumps by modifying the boosting algorithm itself (Kégl and Busa-Fekete 2009).

Classification and regression trees

Classification and regression trees (CART) (Breiman et al. 1984) form a more complicated weak learner, and together with trees based on other splitting criteria, they comprise perhaps the most popular general weak learner. CART trees in particular use the popular Gini index, G, which measures impurity at a node. The goal is to locally minimize:

$$G(A_i) = \sum_{j=1}^{M_i} p(A_{ij}) G(C|A_{ij})$$

where

$$G(C|A_{ij}) = 1 - \sum_{k=1}^{J} p^2(C_k|A_{ij}),$$

with A_i being the feature on which to branch, J the number of classes, M_i the number of features, and $p(A_{ij})$ the probability of $A_i = j$th value, and $p(C_k|A_{ij})$ as the probability an example is in C_k given its attribute $A_i = j$. See Galiano et al. (2003) for more details on splitting rules, including the Gini index.

Decision trees, used as weak learners in boosting are competitive with state of the art solutions for a wide array of problems, generally outperforming decision stumps (Caruana and Niculescu-Mizil 2006). Decision trees, however, also present their own difficulties. In order to avoid overfitting the training data, decision trees need to be pruned to a limited number of nodes. However, the number of k-node binary decision trees grows exponentially in k, presenting theoretical, if not practical, difficulty in Occam's Razor bounds.

Moreover, even pruned trees, or trees whose size is controlled by some stopping criteria, can be problematic for controlling complexity, and certain boosting algorithms can overfit the data using pruned decision trees as weak learners (Reyzin and Schapire 2006). With this in mind, we are motivated to design a weak learner without these downsides that can be competitive with the state of the art.

Sparse parities

Herein, we propose using sparse parity functions, the low-degree characters, as weak learners. Their proposed use in boosting, while a straightforward idea, is to our knowledge novel.[1] This section assumes binary features and binary classification. We then discuss how to possibly circumvent both these limitations in future work.

[1] One exception is a theoretical result of Jackson (1997) who uses an extended Kushilevitz and Mansour (1993) type Fourier-finding algorithm as a boosting subroutine in order to prove a result for learning DNF under the uniform distribution with membership queries. Follow-up works by (Bshouty and Feldman 2002) and (Jackson, Klivans, and Servedio 2002) explicitly considered the sparsity of the parities used in learning.

Theory

A basic fact of Fourier analysis is that any Boolean function $f : \{0,1\}^n \to \{-1,1\}$ can be uniquely written in the Fourier basis, taking the form

$$f(x) = \sum_{S \in \{0,1\}^n} \hat{f}_S \chi_S(x), \tag{3}$$

where

$$\chi_S(x) = (-1)^{S \cdot x}.$$

The χ_S are known as the **characters** over $\mathbb{GF}(2)$, the Galois field consisting of two elements. One may, for simplicity, just think of this function as changing with the parity of $S \cdot x$. Hence, in the PAC learning literature, these χ_Ss are usually known as parity functions, with the **sparsity** of a parity referring to the degree of the corresponding character, i.e. a d-parity corresponds to a character of degree d. We shall use these terms interchangeably.

The scalar quantities $\hat{f}_S$ are known as the **Fourier coefficients** of the corresponding characters of f, and specifying all 2^n of them is sufficient to reconstruct f. The **degree** of a coefficient $\hat{f}_S$ is simply equal to $||S||_1$. Functions with all non-zero coefficients having low degree are called low degree functions.

The **weight** of the coefficients of degree $\leq d$ is

$$\mathbf{W}^{\leq d}[f] = \sum_{S \in \{0,1\}^n \text{ s.t. } ||S||_1 \leq d} \hat{f}_S^2.$$

Our proposed technique can approximately recover targets whose Fourier-weight is concentrated on the low degree characters, i.e. f such that

$$\mathbf{W}^{\leq d}[f] \geq 1 - \epsilon_0,$$

for a suitable choice of ϵ_0 as a function of the target error rate.

While low degree functions over $\{0,1\}^n$ seem quite limited, they are surprisingly powerful. We have already seen that when the degree restriction is $d = 1$, then the class is precisely the linear functions (parities of single features are just the features themselves), which has already proven itself to be quite effective in machine learning.

Use in boosting

It is not hard to see that the form of Equation 3 is similar to boosting's goal of finding a function H such that

$$H(x) = \text{sign}\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right).$$

If we set the weak learner class $\mathcal{H}_d$ to be the d-parities, or characters of degree $\leq d$, as well as their negations, we get a correspondence between the αs and the $\hat{f}$s, as well as between the hs and the χs.[2] Hence, boosting will try to find a low degree Fourier approximation of the target function, if it exists.

Now we analyze which of the desired properties such a weak learner will have.

First, we note that $|\mathcal{H}_d| = 2\binom{n}{d} \leq 2n^d$ when the degree bound is d. Using the generalization error bounds in Equations 1 and 2, we get bounds of

$$\text{err}(H) \leq \left(\frac{d \ln n}{m\gamma^2} + \ln\left(\frac{1}{\delta}\right)\right) \tag{4}$$

and

$$\text{err}(H) \leq \Pr_S[yf(x) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d \ln n}{m\theta^2}}\right). \tag{5}$$

Both these bounds have a contribution of $d \ln n$ stemming from $|\mathcal{H}_d|$, which are small for constant d. However, these functions are able to represent a rich class of functions.

One still has to analyze the achievable γ for the bound, and this will depend on the target data distribution. However, the following facts should help us

1. Observing that $\mathcal{H}_1$ is the class of decision stumps, $\forall d \geq 1$, $\mathcal{H}_1 \subseteq \mathcal{H}_d$. Hence, we can expect the characters to achieve higher γs.
2. On the uniform distribution, all of the hypotheses used by the weak learner are orthogonal. Namely, let $h_1, h_2 \in \mathcal{H}$ s.t. $h_1 \neq h_2$; it is not hard to see that

   $$\Pr_{x \sim U}[h_1(x) = h_2(x)] = 1/2.$$

   This suggests that as a hypothesis $h \in \mathcal{H}_d$ is found to satisfy the weak learning criterion, `AdaBoost` always renormalizes the distribution to be orthogonal to h. If the starting distribution is close to uniform, the renormalization should leave the error rates of the remaining hypotheses in $\mathcal{H}$ nearly unaffected.

This suggests that the γs produced by this weak learner should be at least as good as for decision stumps. Moreover, because $\mathcal{H}$ is symmetric in $x_1 \ldots x_n$ and its hypotheses are evaluable in linear time, the sparse parities clearly meet the diversity, coverage, simplicity, efficiency, error and richness.

The optimizability criterion, however, is harder to meet, as naively one would need to try all the $\binom{n}{d}$ elements. This is the focus of the following section.

Optimizing over parity functions

Because of the orthogonality property of the class, for a long time no algorithm better than brute force was known for learning parities with noise, which is present in any real-world dataset. Even this would not be a major impediment to learning via low-degree coefficients, e.g. $d = 2$ or 3, however recent advances give hope for using larger classes of weak learners.

Blum et al. (2003), in a theoretical breakthrough separating the learning complexity classes $\eta-$PAC and SQ, produced essentially the first advance on the learning parity problem, showing that arbitrary parities over n variables can be learned in time approximately $2^{\frac{n}{\log n}}$, as opposed to the

[2] It has been observed that variants of boosting can be used as Goldreich-Levin (1989) analogues for finding heavy Fourier coefficients, with recently an explicit connection being made between boosting and quadratic Fourier coefficients (Tulsiani and Wolf 2011). Of course, we leave this task for the weak learner.

brute force solution of checking all 2^n functions. Their solution remains state of the art for the general case.

The first progress on the sparse version of the noisy parity problem was tackled by Grigorescu et al. (2011), who gave a $\tilde{O}(\sqrt{n^d})$ algorithm for learning d-parities. Subsequently, Valiant (2012) gave a $O(n^{\cdot 8d})$ algorithm, but with a significantly better dependence[3] on the noise rate η, which has been omitted from the statements of the bounds. We include the (Grigorescu, Reyzin, and Vempala 2011) algorithm, Algorithm 2, herein for completeness. The algorithm of (Valiant 2012) more efficiently finds the correlations from step 3.

Algorithm 2 Approach of Grigorescu et al. (2011) for learning d-parities $(d, n, \epsilon, \delta, \eta)$

1: Given a set $\hat{X} = \{x_1, \ldots, x_m\}$ of examples drawn from D with noise $\eta < 1/2$,
where $m \geq \frac{d \log(n/\delta)\omega(1)}{(\epsilon' - \eta)^2}$ and $\epsilon' = \epsilon + \eta - 2\epsilon\eta$.

2: For each $\frac{d}{2}$-parity c, evaluate it on $\hat{X}$ to obtain the corresponding points

$$\langle c \cdot x_1, c \cdot x_2, \ldots, c \cdot x_m \rangle \in \{0,1\}^m$$

and

$$\langle c \cdot x_1 + \ell(x_1), c \cdot x_2 + \ell(x_2), \ldots, c \cdot x_m + \ell(x_m) \rangle \in \{0,1\}^m.$$

Let $\mathcal{H}$ be the set of all these $2 \cdot \binom{n}{d/2}$ points on the Hamming cube.

3: Run the Approximate Closest Pair algorithm from (Andoni and Indyk 2006) on $\mathcal{H}$ with the approximation parameter $\rho = \epsilon'/\eta$, to obtain the closest pair of points in $\{0,1\}^m$ with corresponding $\frac{d}{2}$-parities c_1 and c_2, respectively.

4: Return $c_1 + c_2$.

Employing these algorithms alone is not enough because of their reliance on random noise and other assumptions. However, Feldman et al. (2009) showed how to, in general, extend such results to work in the adversarial noise setting without asymptotically sacrificing on running time.

Altogether, the recent results above give concrete algorithms for efficiently using higher constant-degree characters, allowing for richer classes of weak learners to be used in practice.

Non-binary features and multiclass prediction

An assumption underlying the use of parities in the manner described is that the features and classifications are both binary. One straightforward extension to the parity function is that for p classes, one can use the function $x \cdot c \pmod p$. The features x can then take integer values (though this splits the numerical values of x into p equivalence classes).

[3]Namely, the algorithm of Grigorescu et al. (2011) has an exponential dependence on η, and the bound degrades to n^d as η approaches $1/2$. Valiant's (2012) bound, which more precisely is $O(n^{\frac{3\omega}{4}+\epsilon})$, does not.

One can, of course, be more clever in extending these functions past binary, however, we will leave this for future work, as our goal is simply to study the basic properties of this weak learner and to argue that this is an area ripe for future study.

Experimental results

In this section we experimentally test our proposed weak learner against two commonly used weak learners: decision stumps and pruned decision trees.

Data

We considered the following datasets: *census*, *splice*, *ocr17*, and *ocr49*, *breast cancer*, *heart*, and *ecoli*, all available from the UCI repository. The splice dataset was modified to collapse the two splice categories into one to create binary-labeled data. The ecoli dataset was similarly modified, merging three classes, to create binary-labeled data.

The following table shows the number of features in each dataset, as well as how many points we used for training and test. 1000 training examples were used unless the data set was too small to accommodate such a large training set, in which case we used fewer examples for training. Each run of the experiment randomly divided the data into training and test sets, and the errors are averages over 20 runs. The divisions used in the experiments, though random, were the same across different algorithms to minimize variance in comparisons.

	# training examples	# test examples	# binary features
oct17	1000	5000	403
ocr49	1000	5000	403
splice	1000	2175	240
census	1000	5000	131
cancer	500	199	82
ecoli	100	236	356
heart	100	170	371

Table 1: Dataset sizes, and numbers of features, for training and test.

Comparing to decision stumps

Because the functions χ_S are a generalization of decision stumps, it is natural to ask how they compare in generalization error. We compared decision stumps (which are characters with a degree bound of 1) to parities bounded by degree 2 and 3 – for these small values it was even possible to do brute-force search over the entire hypothesis space in the optimization.

As we can see, these results show that sparse parities compare favorably against stumps as weak learners. The data also illustrate that stumps also sometimes outperform higher degree parities – this is unsurprising, one pays for the extra hypotheses in the class by the d term in the bounds in

	decision stumps	2-parities	3-parities
oct17	1.09	0.68	**0.62**
ocr49	6.08	2.80	**2.77**
splice	7.37	**4.64**	4.99
census	**18.50**	20.44	22.00
cancer	4.45	4.02	**3.64**
ecoli	7.06	6.84	**5.87**
heart	**22.24**	23.88	22.94

Table 2: Error rates of decision stumps, 2-parities, and 3-parities used as weak learners for `AdaBoost` run for 250 rounds, averaged over 20 trials.

Equations 4 and 5, which may not be compensated for with higher edges.

Overall, these results indicate that the parity-based weak learner more than competes with decision stumps. However, it is also important to compare with decision trees, which as we have mentioned, generally tend to outperform decision stumps.

CART trees

Now we compare our weak learner to a more-widely used and arguably better weak learner – pruned decision trees, namely CART trees pruned to 16 nodes.

Because, modulo the splice dataset, 2-parities are outperformed either by decision stumps or 3-parities, we exclude them from the following comparison table.

	decision stumps	3-parities	CART-16 trees
oct17	1.09	**0.62**	1.11
ocr49	6.08	2.77	**2.16**
splice	7.37	4.99	**3.18**
census	**18.50**	22.00	22.00
cancer	4.45	3.64	**3.19**
ecoli	7.06	**5.87**	8.98
heart	**22.24**	22.94	24.06

Table 3: Error rates of decision stumps, 3-parities, and CART-16 trees used as weak learners for `AdaBoost` run for 250 rounds, averaged over 20 trials.

Here, we see a more mixed picture. Which algorithm is best is fairly evenly split among the classifiers, with sparse parities occasionally strongly outperforming the others, see especially the ecoli dataset. However, mixed results are to be expected of many new algorithms, and we present all our data in full for a fairer comparison. In designing a weak learner competitive with those currently commonly used, we can claim sparse parities should be considered as a choice of weak learners, at least for some domains. One especially interesting observation is that sparse parities were never the worst performing classifier.

Discussion

In this paper we discussed some desirable properties of weak learners, as well as the strengths and weaknesses of decision stumps and decision trees. Our theory pointed to a new, and simple, weak learner – sparse parity functions, which is quite well studied in other contexts in the theory community, but not previously empirically tested in the context of boosting. Empirically, our weak learner stayed competitive with decision stumps and decision trees, sometimes achieving much lower and sometimes higher error rates.

Another main goal of this work is to raise the question of principled design of weak learners, which we hope will become more widely studied.

Acknowledgements

We thank Shmuel Friedland for the use of his computing resources in our experiments. We thank the anonymous referees of both this version of the paper and of a previous version for helpful comments and suggestions.

References

Andoni, A., and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, 459–468.

Blum, A.; Kalai, A.; and Wasserman, H. 2003. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* 50(4):506–519.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Wadsworth.

Bshouty, N. H., and Feldman, V. 2002. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research* 2:359–395.

Caruana, R., and Niculescu-Mizil, A. 2006. An empirical comparison of supervised learning algorithms. In *ICML*, 161–168.

Demiriz, A.; Bennett, K. P.; and Shawe-Taylor, J. 2002. Linear programming boosting via column generation. *Mach. Learn.* 46(1-3):225–254.

Domingos, P. 2012. A few useful things to know about machine learning. *Commun. ACM* 55(10):78–87.

Feldman, V.; Gopalan, P.; Khot, S.; and Ponnuswami, A. K. 2009. On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput.* 39(2):606–645.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55(1):119–139.

Freund, Y. 2001. An adaptive version of the boost by majority algorithm. *Mach. Learn.* 43(3):293–318.

Friedman, J.; Hastie, T.; and Tibshirani, R. 1998. Additive logistic regression: a statistical view of boosting. *Ann. Stat.* 28:2000.

Galiano, F. B.; Cubero, J. C.; Cuenca, F.; and Martín-Bautista, M. J. 2003. On the quest for easy-to-understand splitting rules. *Data Knowl. Eng.* 44(1):31–48.

Goldreich, O., and Levin, L. A. 1989. A hard-core predicate for all one-way functions. In *STOC*, 25–32.

Grigorescu, E.; Reyzin, L.; and Vempala, S. 2011. On noise-tolerant learning of sparse parities and related problems. In *ALT*, 413–424.

Grove, A. J., and Schuurmans, D. 1998. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, 692–699.

Jackson, J. C.; Klivans, A.; and Servedio, R. A. 2002. Learnability beyond ac0. In *STOC*, 776–784.

Jackson, J. C. 1997. An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. *J. Comput. Syst. Sci.* 55(3):414–440.

Kégl, B., and Busa-Fekete, R. 2009. Boosting products of base classifiers. In *ICML*, 63.

Kushilevitz, E., and Mansour, Y. 1993. Learning decision trees using the fourier spectrum. *SIAM J. Comput.* 22(6):1331–1348.

Mason, L.; Bartlett, P. L.; and Baxter, J. 1998. Direct optimization of margins improves generalization in combined classifiers. In *NIPS*, 288–294.

Reyzin, L., and Schapire, R. E. 2006. How boosting the margin can also boost classifier complexity. In *ICML*, 753–760.

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

Schapire, R. E.; Freund, Y.; Bartlett, P.; and Lee, W. S. 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. *Ann. Stat.* 26(5):1651–1686.

Schapire, R. E. 1990. The strength of weak learnability. *Mach. Learn.* 5:197–227.

Schapire, R. E. 2003. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*. Springer.

Tulsiani, M., and Wolf, J. 2011. Quadratic goldreich-levin theorems. In *FOCS*, 619–628.

Valiant, G. 2012. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *FOCS*, 11–20.

Viola, P. A., and Jones, M. J. 2004. Robust real-time face detection. *Int. J. Comp. Vision* 57(2):137–154.