"Boosting" Stumps from Positive Only Data

Andrew R. Mitchell[*]
Artificial Intelligence Group
School of Computer Science and Engineering
University of New South Wales Sydney 2052, NSW, Australia
Email: andrewm@cse.unsw.edu.au

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

[*]Supported by APA scholarship.

Abstract

Most current learning algorithms require both positive and negative data. This is also the case for many of the recent ensemble learning techniques. Applications of *Boosting*, for example, rely on both positive and negative data to produce a hypothesis with high predictive accuracy.

In this technical report, a learning methodology is presented that does not rely on negative examples. A learning method in this framework is described which shows remarkable similarities to boosting stumps. This is all the more surprising because learning from positive data has traditionally turned out to be very difficult.

Empirical results show that this technique successfully boosts stumps from positive data by paying only a small price in accuracy compared to learners that have access to both positive and negative data.

Some theoretical justification of the results is also provided.

1 TRADITIONAL LEARNING MODEL

In the traditional learning model, a learner is presented with a finite set of examples with their corresponding class labels. From this the learner is required to produce a decision procedure, which, given an unlabelled (unclassified) example, returns, with a high degree of accuracy, the correct class label [Qui86, FS96].

Decision trees have been very effectively used to represent classifiers, and several systems have been built to produce decision trees from positive and negative data [FS96]. Ensemble methods such as bagging and boosting [Qui96, Bre96] combine the outcome of several different classifiers to form a new (and hopefully improved) classifier. In most cases, the dataset is modified after each classifier is created so as to create a different classifier each time.

Boosting [FS96] is one such ensemble technique. In this method, the first classifier is created from the full data set, each example being equally weighted. After the initial classifier has been built, the weights of the examples are modified so that the original classifier performs no better than chance on the new, weighted data set. This process is continued to create a series of classifiers. The classifiers are assigned weights according to their estimated error and trial number and are voted together on any new unclassified data.

Given the power of boosting to improve the classification accuracy of a learner, one does not need to begin with a very good learner. *Decision Stumps* [IL92] are examples of such a weak learner whose accuracy has been shown to improve by application of boosting. A decision stump is a decision tree of depth one; i.e., a stump consists of a single decision node.

In boosting, as presented by Freund and Schapire [FS96], there does not appear to be any direct approach to applying boosting to positive only data, as the error of a hypothesis on the training data is not very meaningful in this case. For example, the hypothesis that covers all the data instances will have zero training error, but will most likely be a very poor predictor. In the present technical report, we describe an algorithm that may be considered a modification of boosting applied to decision stumps in such a way that boosting still occurs despite the learner receiving only positive data.

2 PRELIMINARIES OF AN ALTERNATE METHODOLOGY

For the purposes of learning from positive-only data, we assume the following model:

The learner receives a finite set of positive only examples of some concept drawn according to some probability distribution. From this data, the learner is required to come up with a procedure that, given any unclassified instance, returns a confidence value [SS98, KS90] in the range $(0, \infty)$ that the given instance is in the concept.

To simplify our treatment we assume that the instance space is n-dimensional and the domain of each attribute is the real interval $[0, 1]$. We also assume that the examples are drawn from a probability distribution function, P, which is bounded, i.e., there exists a bound c such that for each instance $\vec{x}$, $P(\vec{x}) \leq c$.

2.1 A LEARNING METHODOLOGY

When learning from positive only data, we are to some extent trying to figure out the probability distribution function that produced the original data. The following proposition provides a basis for achieving this.

Proposition 1 Let P be any distribution satisfying the assumptions noted above. Let $f : [0, 1]^n \mapsto [0, 1]^n$ be a monotonic function which maps P to the uniform distribution. Then for all points x where P is continuous, $f'(x) = P(x)$.

The proposition is illustrated in Figure 1.

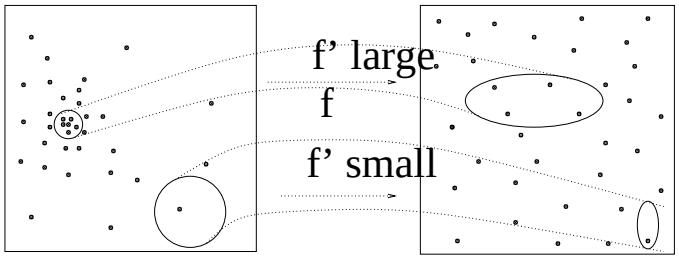

Figure 1: $f' = P$

Proof: Consider the single dimensional case. Assume that we have a monotonic function, $f : [0,1] \mapsto [0,1]$, which maps P to the uniform distribution, U. For any point, x in $[0,1)$, and $\epsilon > 0$ (such that $x + \epsilon < 1$),

$$\int_x^{x+\epsilon} P(t)dt = \int_{f(x)}^{f(x+\epsilon)} U(t)dt$$

as f is monotonic (any point between x and $x+\epsilon$ maps to a point between $f(x)$ and $f(x+\epsilon)$, and vice versa). However,

$$\int_{f(x)}^{f(x+\epsilon)} U(t)dt = f(x+\epsilon) - f(x)$$

Hence

$$\lim_{\epsilon \mapsto 0} \int_x^{(x+\epsilon)} (P(t)/\epsilon)dt = \lim_{\epsilon \mapsto 0} (f(x+\epsilon) - f(x))/\epsilon$$

so $P(x) = f'(x)$ as required. The multi-dimensional case follows similarly. ∎

The above proposition can be used as a basis for a general method of learning which we call SPRAL (spread and learn)! The idea is that given infinitely many examples drawn according to distribution P, we can approximate f whose derivative yields P. This is summarised in the following proposition.

Proposition 2 Given a continuous distribution, P, satisfying above assumptions, there is an algorithm which will, from t examples drawn according to P, produce an f_t, such that as $t \mapsto \infty$, $f_t \mapsto f$ where f is as above (i.e., such that $f'(x) = P(x)$

The proof is omitted due to a lack of space.

2.1.1 Finite Data

For practical applications, however, we are restricted to learning from finite data. This means that we need a method for determining whether a finite amount of data is uniform. For this, we need some prior information on possible non-uniform probability distribution functions that we are looking for in the data, as any set of points of size n is equally likely to come from a uniform distribution.

In machine learning, we tend to examine conjecture domains where similarly valued data points have similar classes. Hence we say that data is more uniform if it is less likely to come from any other smooth, simple probability distribution function, which occurs when the data is spread out more. Hence, relying on this prior, we shall henceforth interchange the terms uniform and well spread. This is in keeping with Friedman's work in projection pursuit [FT74] where data is considered interesting if it is not spread.

2.2 ENSEMBLE CLASSIFIERS - SPRALEN

Suppose we have an algorithm that produces a function f from any data set E such that f transforms E "closer" to uniform. We would like to improve the performance of our algorithm using an ensemble type method.

Suppose we use our algorithm to find an f_1 from the original data set. Suppose we modify the data by applying f_1 to each data point, and again use our original algorithm on this transformed data set to find f_2. Then $f_2(f_1)$ should take as input the original distribution and produce something even closer to uniform. Hence $f_2(f_1)$ would be a better f. We can repeat this process as many times as we wish (see Figure 2).

We call this ensemble method SPRALEN (Spread and learn ensemble). This relates very closely to Friedman's "structure removal" idea [Fri87].

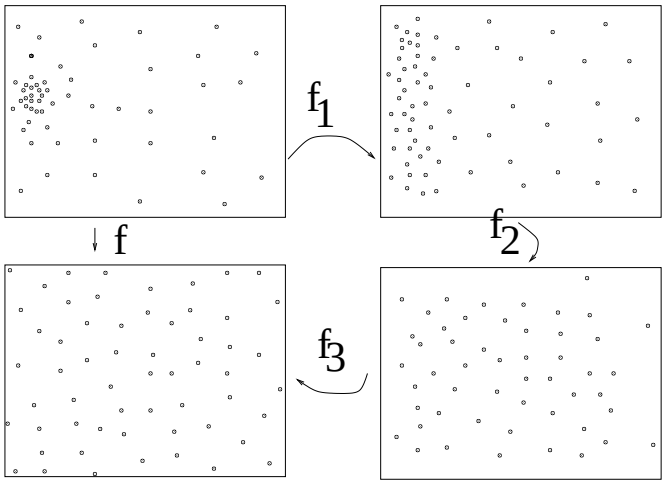

Figure 2: SPRALEN: f improves on f_1

The idea can be captured in the following simple proposition:

Proposition 3 Suppose we have some measure of uniformness, s [FT74], and a distribution, P on $[0,1]^n$. Suppose we have functions $f_1, f_2, \ldots, f_n$ and that (for each i from 1 to n), $f_{i-1}f_{i-2}\ldots f_1 P$ is less s uniform than $f_i f_{i-1}\ldots f_1 P$, then

$$f_n f_{n-1}\ldots f_1 P$$

is more s uniform than $f_1 P$.

The proof is trivial.

The result of SPRALEN is a confidence procedure that, in general, is less simple than the original f_1, but fits the data better (just like traditional ensemble methods).

3 APPLYING SPRALEN to STUMPS

We now examine methods of finding an f from a given data set. As with boosting, the initial learner need not be very good. We first describe a 2-piecewise linear method for finding an initial candidate for f and argue that this is essentially a decision stump. We then describe how SPRALEN can be employed to "boost" decision stumps.

3.1 THE 2-PIECEWISE LINEAR METHOD

A function is 2-piecewise linear on $[0, 1]$ if it is linear on $[0, a]$ and on $[a, 1]$ for some a.

By some method, select an attribute a and a value p in the range of a.

Choose by some method a point q in the domain of a.

Find the linear map f_1 and f_2 such that $f_1(0) = 0$, $f_1(p) = q$,$f_2(p) = q$ and $f_2(1) = 1$.

Let $f : [0, 1]^n \mapsto [0, 1]^n$ be such that $f(x) = f_1(x_a)$ if $x_a < p$ and $f(x) = f_2(x_a)$ otherwise, where x_a is the value of x on attribute a.

3.2 WHY IS THE ABOVE A STUMP?

Consider the derivative of the above f. In the range $[0, p]$, the derivative is q/p. In the range $[p, 1]$, the derivative is $(1 - q)/(1 - p)$. So, on one side of the split, any unlabelled example receives confidence (or voting power for a positive example) of q/p and on the other side of $(1 - q)/(1 - p)$.

The reader should note that our learning method returns a confidence value rather than the class label returned by a decision tree. This can, however, be converted to a decision procedure by introducing a cut-off c where any higher confidence than c causes the example to be considered positive, and any lower confidence negative.

In the case of this 2-piecewise linear method, a cutoff lower than $min(q/p, (1 - q)/(1 - p))$ causes all examples to be considered positive, a value between q/p and $(1 - q)/(1 - p)$ causes all examples on one side of p to be considered positive, and on the other negative. A value greater than both q/p and $(1 - q)/(1 - p)$ causes every example to be considered negative. Each of these decision procedures is, however, equivalent to a stump with split point p.

This shows the equivalence of using SPRAL to find a 2-piecewise linear function and a cutoff to that of using a single stump. That is, finding a 2-piecewise linear function for f and a cutoff value could be done by finding an appropriate stump. Similarly, any stump could be represented by a 2-piecewise linear function and a cutoff value.

Next we compare the method of boosting stumps with SPRALEN, where the functions f are calculated using a 2-piecewise linear method and a cutoff value.

3.3 SPRALEN USING A 2-PIECEWISE LINEAR METHOD AND A CUTOFF VALUE

A *confidence stump* is a stump where the leaves represent confidence values of the example being positive.

As seen above this is in some way equivalent to SPRAL with a 2-piecewise linear method. With a cutoff value, it becomes a traditional stump.

Suppose we create n functions,

$$f_1, f_2, \ldots f_n$$

using SPRALEN and a 2-piecewise linear method. The composed function, f, is:

$$f_n(f_{n-1}(\ldots f_1))\ldots)$$

As each f_i is piecewise linear, $f'(x)$ is

$$f_1'(x)f_2'(f_1(x))\ldots f_n'(f_{n-1}(f_{n-2}(\ldots(f_1(x))\ldots)))$$

As each f_i is linear, this is equivalent to multiplying the results of n confidence stumps. For example, the ith confidence stump would split at $f_{i-1}(f_{i-2}(\ldots f_1(p))\ldots)$ where p was the split in f_i.

If we take this and apply a cutoff value, c, we find that we are multiplying together n confidence stumps and using a cutoff value. Each stump will multiply the confidence of all examples on one side of the split

by a constant $c_{i,1}$ and those on the other by another ($c_{i,2}$). We can, however multiply $c_{i,1}$ and $c_{i,2}$ by a constant without affecting the final classifier's result on any input, so long as we also multiply c by this constant. Suppose we do this for each multiplicative stump such that $c_{i,1}c_{i,2} = 1$.

Taking logarithms of the operation, we find that this learner is equivalent to adding values together, which can be represented as voting stumps together (where each stump has some weight and an appropriate cutoff value is chosen).

Friedman [Fri99] has recently used boosting of decision trees to approximate functions. We are doing something slightly similar here in that we are using something that is "equivalent" to the technique of boosting stumps to approximate a probability distribution function.

We next shed some more light on why our method can be viewed as boosting stumps from only positive data.

3.3.1 Closer similarities to boosting

We now draw more similarities between SPRALEN on 2-piecewise linear functions with a cutoff and boosting of stumps. Let us closely examine the (slightly generalised) boosting algorithm.

> F_0 is the first learner found (from the original data)
>
> **For** $m = 1$ to M do:
>
> > Examine how F_{m-1} acts on the current data.
> > Modify the data such that F_{m-1} performs no better than chance.
> > Find a learner, F_m for the new data.
>
> For unlabelled examples, vote $F_1, F_2, \ldots, F_M$ together, giving each learner a weight.

This algorithm is, however, the same algorithm that we are using when we use SPRALEN on 2-piecewise linear functions with a cutoff, where voting is taken as logarithms (given the equivalence shown previously).

The main differences between boosting and our algorithm are as follows:

> Our algorithm uses only positive data and requires the selection of a cutoff value (these two are closely related).
>
> Different methods must be used to find the F_is.
>
> The methods "modify the data" differently.

We have yet to say how we modify the data when using SPRALEN. When we do, we shall show that we modify the data so that the previous learner does "no better than chance".

Due to these similarities, we claim that, in essence, our algorithm does "boost" the learner.

3.4 HOW TO FIND EACH WEAK LEARNER

To complete the definition of our learning algorithm for positive only data we must describe a method for finding p and q—the split point for our "stump" and the point which it is moved to.

Firstly, concerning a method for finding q.

As we are trying to spread the data, the intuitive way to find q is such that the average density of examples on each side of the moved-split-point is the same. As our "stump" merely changes the relative values of the two sides of the split, this is in some sense the best that is possible.

As a bonus, this method modifies the data so that the previous learner does no better than chance in that if the same split point was chosen again, the average densities on each side would be the same, and hence the average confidences on each side would be the same. Such a split would give us no information gain.

This leaves the problem of finding p.

As a first approach, so that our results could be easily compared to Ross Quinlan's *C4.5*, we used the same algorithm as *C4.5* to determine a split (that is, we added uniform negative data and asked *C4.5* where it would split the data). We termed this method *PolarC4.5* (Positive only learner *C4.5*). This method performed well, and was a good base to work from. It was not, however, theoretically "nice". The method that we chose next was to find the split where the number of points below (or equivalently above) the split point was most different from what one would expect from a uniform distribution. We termed this algorithm *Polar* (Positive only learner).

3.5 USING SPRALEN ON DATA THAT CONTAINS BOTH POSITIVE AND NEGATIVE EXAMPLES

As we would like to explore the applicability of our method on more that just positive only data sets, we investigated how we might use this method on multi-class problems.

To do this, our approach is to learn each class separately, to create one confidence procedure per class. For any unlabelled example, we then compare the confidences of the confidence procedures. We then classify the data as the class for which the confidence procedure gave the highest confidence. Hence we do not need a cutoff value when there is more than one class involved. The confidence values from the confidence procedures are reasonably comparable as the confidences relate directly to the densities of the examples, once prior probabilities (average densities over the entire space) were taken care of. We call this process *parallelising*.

Although this method of learning from positive and negative examples theoretically works, the model that SPRALEN is using (positive-only examples) makes the job much more difficult for the learner. The confidence predictor found by a positive only learner should be able to differentiate the positive concept learned with any other concept as well as it can. Learners given positive and negative data, on the other hand, merely need to separate the given classes.

Hence we would expect that using this method to make a positive-only learner learn a multi-class problem should perform worse than a dedicated multi-class learner. The advantage in using parallelism is that we can see how well a positive-only learner does in dealing with the difficult problem of only having positive data. That is, the closer the results of a parallelised, positive only learner are to those of the best (comparable) multiclass learner, the better the learner has overcome the problem. Another advantage is that it allows us to compare positive-only learners, easily, on well studied data sets where we can easily calculate how well the learner is performing.

4 TESTING ON REAL DATA

We would like to examine how our method(s) compare with other methods on real data. As noted in the previous section, we have the following positive only learners:

- *Polar* (our main algorithm, similar to boosting stumps),
- *PolarC4.5* (where *C4.5* was used to select split points),

There were no positive only learners at our disposal to compare *Polar* to, so we wondered if we could turn *C4.5* or *C4.5b*[Qui96] (*C4.5* with boosting, by Ross Quinlan) into a positive-only learner. The method we used for this was to take the positive-only data, and add uniformly distributed negative data, ask *C4.5*

Figure 3: Percentage Error on UCI data

		Positive and negative data						Pos only		
	Number of attributes	**C4.5 boost**	C4.5	C4.5 stump boost	C4.5 boost	C4.5	C4.5 stump boost	**Polar**	Polar C4.5	**C4.5b po**[4]
		Original data			Continuous only					Orig.
pendigits	10	**0.5**	3.4	79.5	0.6	3.6	79.5	**12.4**	13.2	**90**
page-blocks	5	**2.8**	3	6.8	2.4	3.1	6.8	**17.1**	12.2	**70**
pima	2	**25**	25	24.1	25.8	26.2	24.6	**29.4**	30.2	**33**
vis	7	**1.3**	3.2	71.4	1.4	3.4	71.4	**24.7**	75.8	**85**
ionosphere	2	**5.8**	10.7	6.8	6.7	11.7	8.2	**14.6**	13.3	**36**
breast	2	**2.8**	4.9	3.9	2.8	4.8	3.8	**2.4**	3.0	**29**
letterAB	2	**0.0**	0.9	0.0	0.0	1.2	0.1	**3.9**	4.0	**15**
vehicle	4	**21.4**	26.4	60.2	22.5	27.1	59.9	**38.9**	40.2	**66**

to learn this, then use the classifier produced as the positive only learner. In the case that more than an example was classified positive by no classifier, or by more than one, a random class was selected.

Polar performed much better than this learner, which we term *C4.5bpo* (see Figure 3). Indeed, *C4.5bpo* performed little better than chance. Unfortunately our figures for *C4.5bpo* are only approximate due to technical reasons [1].

Using the method of parallelism as described above, we tried our learners on some of the UCI datasets [CBM98]. We also tried the more traditional learning methods of *C4.5b* and *C4.5bstumps* (our modification of *C4.5b* as a stump learner). We used various numbers of trials[2], from 1 (that is, no boosting) to 200 trials [3]. We used 10 trial 10-fold cross validation. The datasets are all standard except for the *letterAB* set, which is the letter data set restricted to the letters A and B. *C4.5bstumps* is a modification of *C4.5b* to only allow stumps, and hence does not boost multiclass problems well as error rates above fifty percent are common after one stump.

Some of the data in the data sets was not created by a bounded probability distribution function, as the attribute values were integral. For these attributes we needed to spread the data a little to bound the probability distribution function. We achieved this by adding a random number $[0, 1]$ to each example in each such attribute where the data was integral. This process would make learning more difficult in general as the data is noisier. To see how much more difficult the learning became, we also ran *C4.5* through this "continuous only" data. All *Polar* results used this data.

In general, the range of the attributes was not $[0, 1]$ as *Polar* requires. Hence *Polar* was modified to scale the given data linearly to $[0.2, 0.8]$. The confidence given to any test data outside the $[0, 1]$ range was "penalised" (although this was rare).

For the table we present (Figure 3), we show the best results for each data set and learning method. Due to space limitations, we represent the data corresponding to the number of boosting trials (of 1, 3, 5, 10, 20, 40, 60, 100, 200) that gave us the least error in our 10 times 10-fold cross validation tests involving boosting. This was the 200 boost trial in almost all cases, although the error rates appeared to be near their lower limits by this stage.

[1] Due to the structure of the *C4.5b* source, it was much easier to calculate the percentage error for each class, and estimate the overall error from this assuming some independence of mistakes

[2] For the parallelising case, the number of trials is measured per class, so that each data point is boosted the number of times corresponding to the number of boosting trials.

[3] 1, 3, 5, 10, 20, 40, 60, 100, 200 trials were used in each case.

[4] Results in this column are less accurate, for reasons described in the text

4.1 RESULTS

As could be expected from the model of learning (positive-only), *Polar* performed worse in general than the positive-negative data learner, *C4.5* (see Figure 3). The results, however, are very good when compared to our basic positive only learner, *C4.5bpo*. It would appear that the methodology of *Polar* does seem to solve most of the disadvantages of positive only data in practice.

One interesting note is how well *Polar* performed on the breast dataset. The result is significantly and consistently better than even *C4.5b*. We cannot explain why this dataset was so much more easily learnable by *Polar*.

Another practical advantage of *Polar* is that it can give, with its result, a degree of confidence of the result. In most of the cases where *Polar* came to the wrong conclusion, the difference between the two confidence values was small.

It was also interesting to examine how the confidence predictors performed when used as SPRALEN with 2-piecewise linear functions and a cutoff value (that is, *Polar* without the parallelisation). In the case of the breast data, such a learner was found to perform much better on the malignant data than the non-malignant data, a discovery which could be useful from a data mining standpoint.

Another curious fact that we noted was that in one dataset (*vis*), *Polar* (and its derivatives) performed significantly worse on higher numbers of iterations. In this case the error more than doubled between 40 and 200 iterations, consistently. At this stage we do not know why this occurred.

We were surprised how few "boosting" iterations were required to achieve near convergent result. Hence the classifiers which *Polar* produced were quite readable by humans. In general an almost optimal classifier from parallelised *Polar* could be represented by about 20 statements of the form "if attribute a is greater than b then there is [excellent || good || some] evidence that the example is [positive—negative], otherwise that it is [negative—positive]".

5 THEORETICAL SUPPORT

At this point we would have liked a nice theoretical result. However, we are unable to prove the following conjecture:

Conjecture 1 Suppose P is a bounded distribution on $[0, 1]$. For any $\epsilon > 0$ there exists n and t such that, given t examples drawn from P, there is at least $1 - \epsilon$ probability of *Polar* giving densities (confidence values) closer than ϵ on at least $1 - \epsilon$ of the space after n "boosts".

In other words:

Given a one-attribute classification problem where each class has elements drawn from a bounded probability distribution, parallel *Polar* will tend towards the correct solution as the number of "boosting" trials and number of examples becomes large.

The above conjecture is difficult to show due to the fact that moving the data in a 2-piecewise linear way will move previously moved split points. That is, the points we have taken care to move to the correct places in the past are now shifted slightly. Intuitively one would imagine that these would even out and converge after some time but this turns out to be difficult to show.

However, we have the following special case of the conjecture:

Proposition 4 If P is a one-dimensional 3-step function[4], the above conjecture holds.

[4] A 3-step function is a step function with at most 2 discontinuities

Proof sketch: Suppose that the discontinuities are at r_1 and r_2. The proof follows by showing the following steps:

Given any δ there exists some t such that p_1 and q_1 are each within δ of where they would be under conditions of infinite data (that is, p_1 is within δ of r_1 or r_2), with probability at least $1 - \delta$.

Given any δ and m there exists some t such that

$$p_1, q_1, p_2, q_2, \ldots p_m, q_m$$

are each within δ of where they would be under conditions of infinite data, with probability at least $1 - \delta$.

Given infinitely much data, *Polar* converges as required for 3-piecewise linear functions.

Find a number of boosting trials such that, under infinitely much data, *Polar* has converged within $\epsilon/2$.

Find an appropriate δ and t.

5.1 ALTERNATIVES

The next question we considered was whether we can alter *Polar* to gain better theoretical grounding. We examined this possibility by modifying the f we are looking for. Instead of looking for a 2-piecewise linear function on $[0, 1]$, we expanded our search to 2-piecewise linear functions on $[a, b]$ where $0 \leq a < p < b \leq 1$ (the function would be the identity elsewhere). We let a be the largest point below p (and b the least point above) such that a and b have already been used as values of q.

This way the points that have already been fixed correctly by the algorithm remain fixed, and the above conjecture becomes trivial. Convergence is also much faster as no points are displaced. So, in theory, this method (which we shall call *PolarInf*) should learn more quickly, at least on very large data sets.

On small (for example real) data sets, however, the situation is slightly different. After a few "boosting" trials, a and b would get close together, meaning that the "learning" gained from any new split is concentrated in a more localised area. Indeed, using this method (of small localised learning) is quite reminiscent of a nearest-neighbour type technique.

We feel that the results of *PolarInf* are less readable than for *Polar* due to the presence of the variables a and b. However:

Proposition 5 Any confidence predictor produced by n-boost *PolarInf* can, by appropriate choices of p(s) and q(s), be represented by a confidence predictor produced by SPRALEN using n-boost 2-piecewise linear functions

That is, whatever can be produced by *PolarInf* can also be produced by "voting" the confidence stumps used in *Polar*

5.2 TESTING *P*olarInf ON REAL DATA

We tried *PolarInf* on the same datasets (see Figure 4), under the same conditions as for *Polar* in an earlier section. The results on average were slightly better than for *Polar*.

Figure 4: Percentage error of *PolarInf* on UCI data.

pendigits	10.3
page-blocks	12.7
pima	28.5
vis	24.5
ionosphere	14.7
breast	2.4
letterAB	4.1
vehicle	38.7

6 CONCLUSION

In this technical report, we have presented a learning method which works on positive-only data and outputs a procedure for determining confidence values which performed considerably better than a basic positive only learner developed from *C4.5b*. We believe that our method very closely captures the essence of boosting decision stumps. We also showed results in which our system performed well against systems which used both positive and negative data.

There are many avenues to explore in modifying this system. Having found a link between boosting and projection pursuit, positive-only data and stumps, further integration and comparison between these areas may be possible and this is well worth exploring. There may also be an interesting link with polygonal approximation [Pap85] methods and the Radon-Nikodyn derivative [Mal93]. We shall also consider decision trees as learners, and consider applications to cluster analysis.

We would also like to work toward more theoretical results, including conjecture 1.

7 ACKNOWLEDGEMENTS

This technical report is a response to a challenge by our advisor Arun Sharma to his students to find a way to make boosting work from positive only data. We would like to thank him for taking the time to help with the preparation of this technical report. We would also like to thank Professor Ross Quinlan for his encouragement and willingness to allow us the use of the *C4.5b* source code. Ross also pointed us to Jerome Friedman's work on projection pursuit. Helpful comments were also provided by Ivan Bratko, Eric McCreath, Jerome Friedman, Waleed Kadous, Michael Harries, Mark Reid and Margaret Wasko. The author is supported by an *Australian Postgraduate Award*.

References

[Bre96] L. Breiman. Bagging predictors. *Machine Learning 26*, pages 123–140, 1996.

[CBM98] E. Keogh C. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[Fri87] J. Friedman. Exploratory projection pursuit. *J. Am. Statist. Ass, 82(397)*, pages 249–266, March 1987.

[Fri99] J. Friedman. Greedy function approximation: A gradient boosting machine, February 1999. Available from `www-stat.stanford.edu/~jhf/`.

[FS96] Y. Freund and R. Shapire. Experiments in a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[FT74] J. Friedman and J Tukey. a projection persuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, pages 881–890, 1974.

[IL92] W. Iba and P. Langley. Induction of one-level decision trees. In *Proceedings of the Ninth international machine Learning Conference*, 1992.

[KS90] Michael J. Kearns and Robert E. Schapire. Efficient distribution-free learning of probabilistic concepts. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, page 389, 1990.

[Mal93] P. Malliavin. *Graduate Texts in Mathematics: Integration and Probability*. Springer-Verlag, 1993.

[Pap85] G. Papakonstantinou. Optimal polygonal approximation of digital curves. *SP*, 8:131–135, 1985.

[Qui86] R. Quinlan. Induction of decision trees. *Machine Learning, 1*, pages 81–106, 1986.

[Qui96] R. Quinlan. Boosting, bagging, and c4.5. In *AAAI Proceedings*, 1996.

[SS98] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh annual conference on Computational Learning Theory*, pages 80–91, 1998.