Matt Zhang Presents:

WHAT THE HELL IS ADABOOST?

AN ALGORITHMS GROUP PRODUCTION

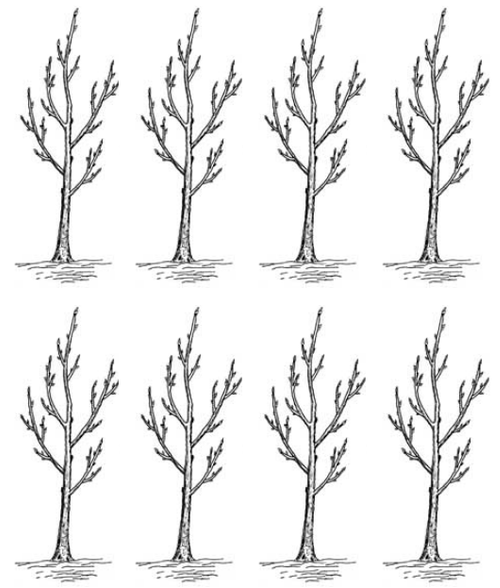
Boosting is a trick for combining weak learners to make them stronger. It is the fascism of machine learning techniques.

Weak tree - runs on all the training data and performs poorly

Training data weighted so that mispredicted events are weighted more highly

Another weak tree is trained

Rinse and repeat



Tree results are weighted based on how well they do with training data. Final classifier is weighted average of many trees.

AdaBoost is one of the first and most popular boosting techniques.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

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

Fig. 1 The boosting algorithm AdaBoost.

D is the weight distribution
t is the index of the tree
calculate error (what percent of the time the tree screws up)

α is positive when the error is below 0.5, and negative when error is larger

y*h is positive when the tree guesses correctly, and negative otherwise. when your tree does well, you want to increase weight for missed events and decrease weight for correct ones, for future training. when your tree is doing really bad you want to back off on the gas and do the opposite.

α is also what you use for weighting your trees in the end

It can be proved that given the weak learning condition (all trees have error less than 0.5), the training error converges to zero in O(log m) training rounds, where m is the number of training events.

Having zero training error is fine, but for most algorithms this leads to overfitting. As we'll see, one of the advantages of AdaBoost is its surprising and mysterious resilience to overfitting (not demonstrated in figure below).

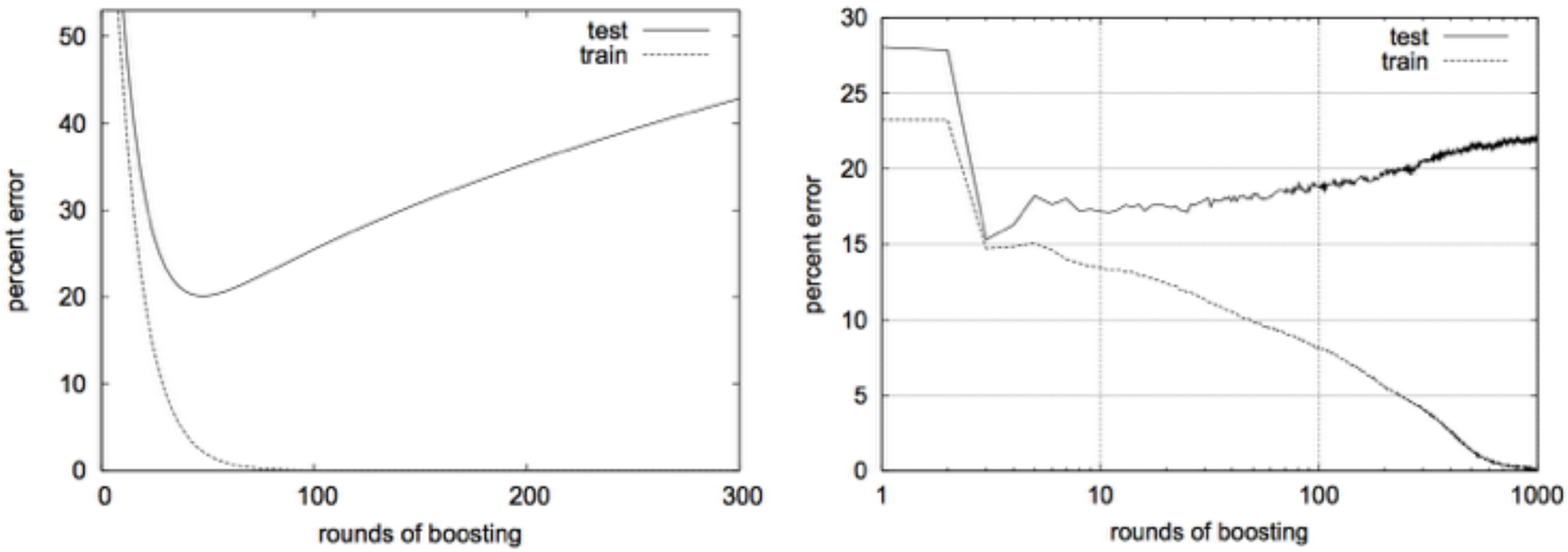
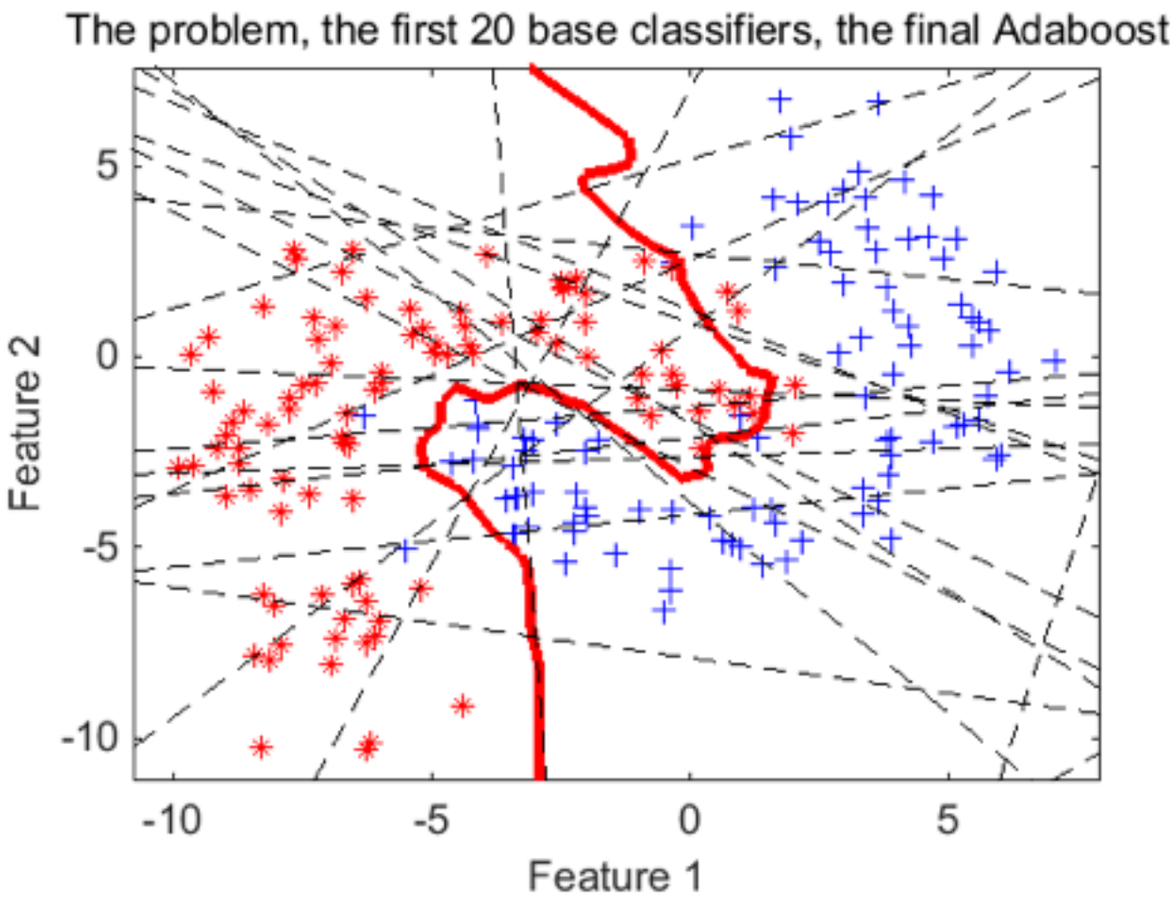
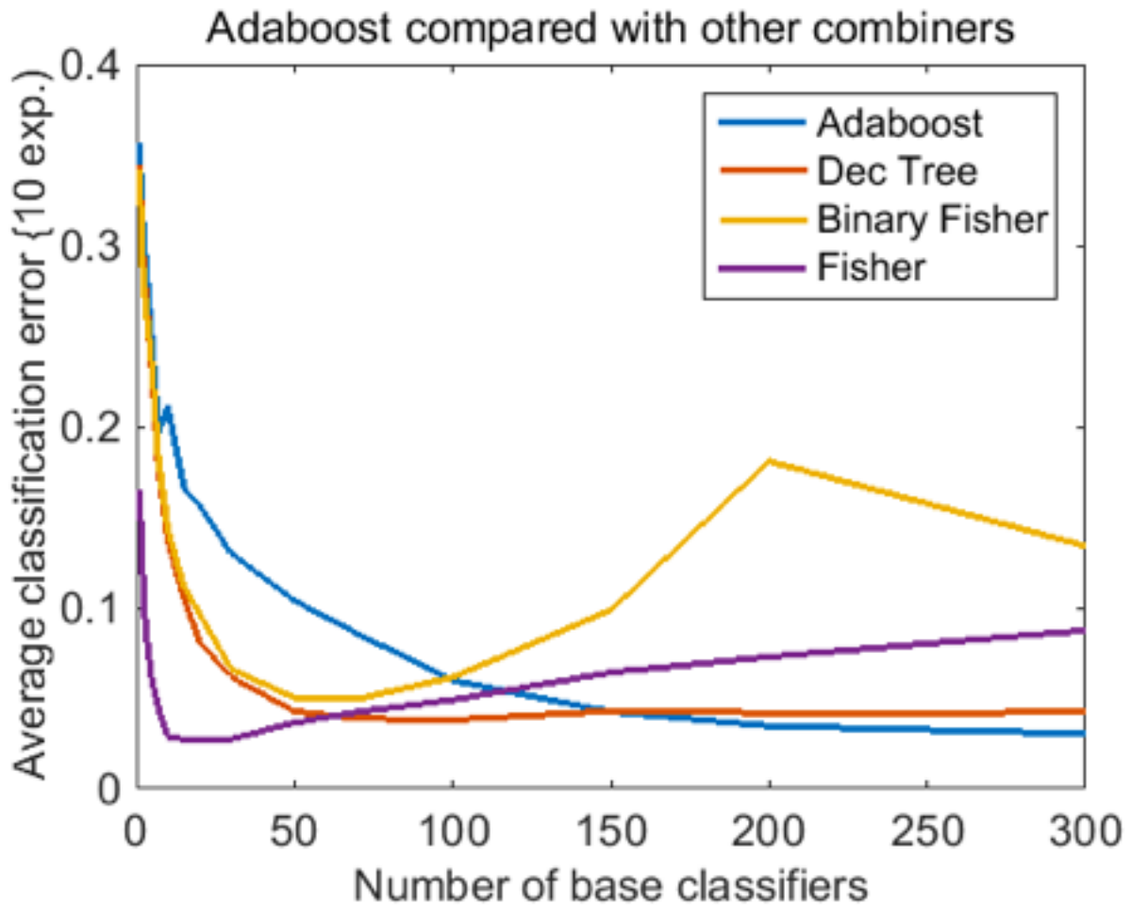


Fig. 2 *Left:* A plot of the theoretical training and test percent errors for AdaBoost, as predicted by the arguments of Section 2. *Right:* The training and test percent error rates obtained using boosting on the Cleveland heart-disease benchmark dataset. (Reprinted from [30] with permission of MIT Press.)

Example of AdaBoost not overfitting.





Base classifier = linear percoptron

Why does AdaBoost tend not to overfit, even when it becomes massively complex? Lots of people have tried to answer this question, but the solution is still unclear. Here are some things they tried…

Margin minimization

Even after training error goes to zero, margin improves with additional training. (Margin = % correct classifiers - % incorrect classifiers).

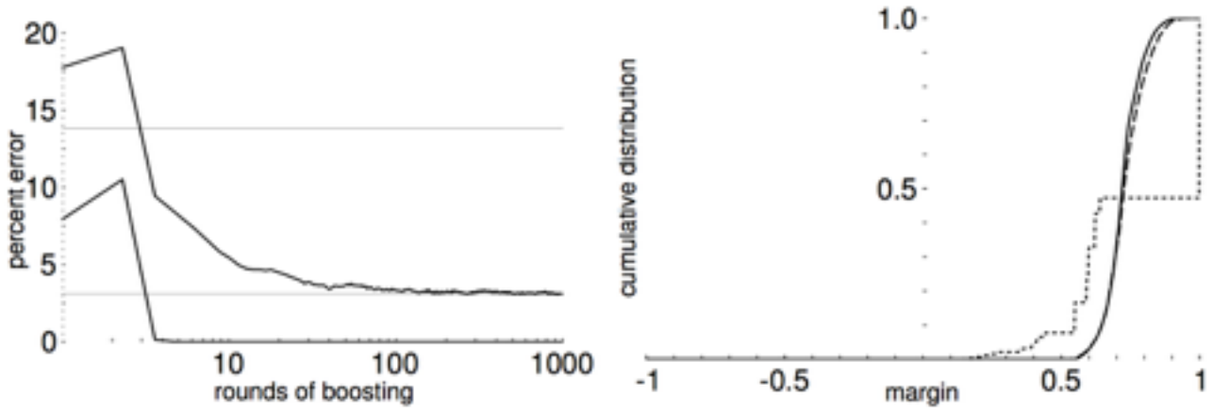
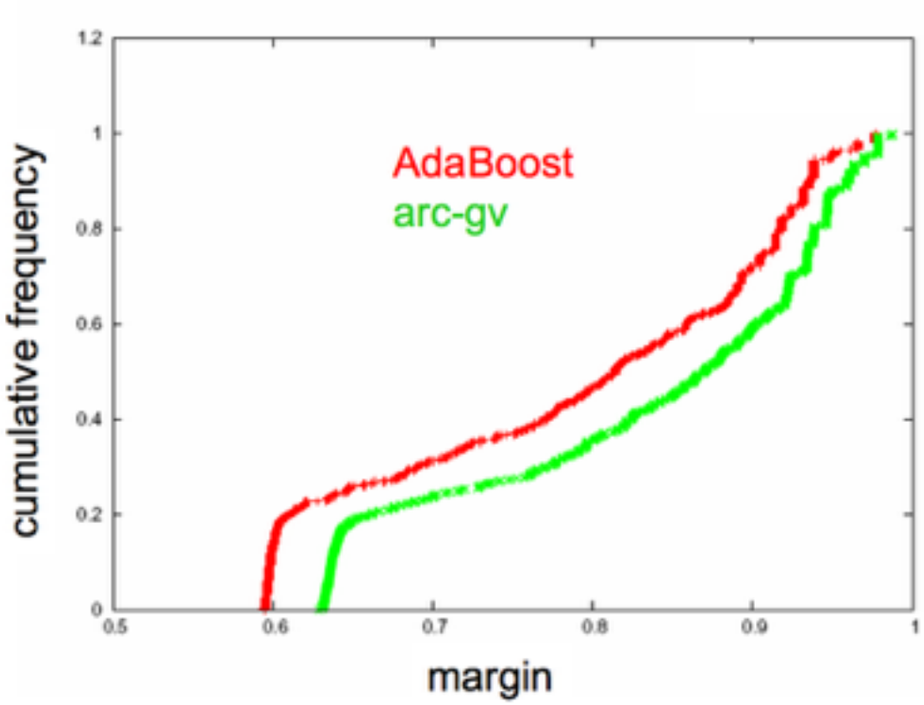


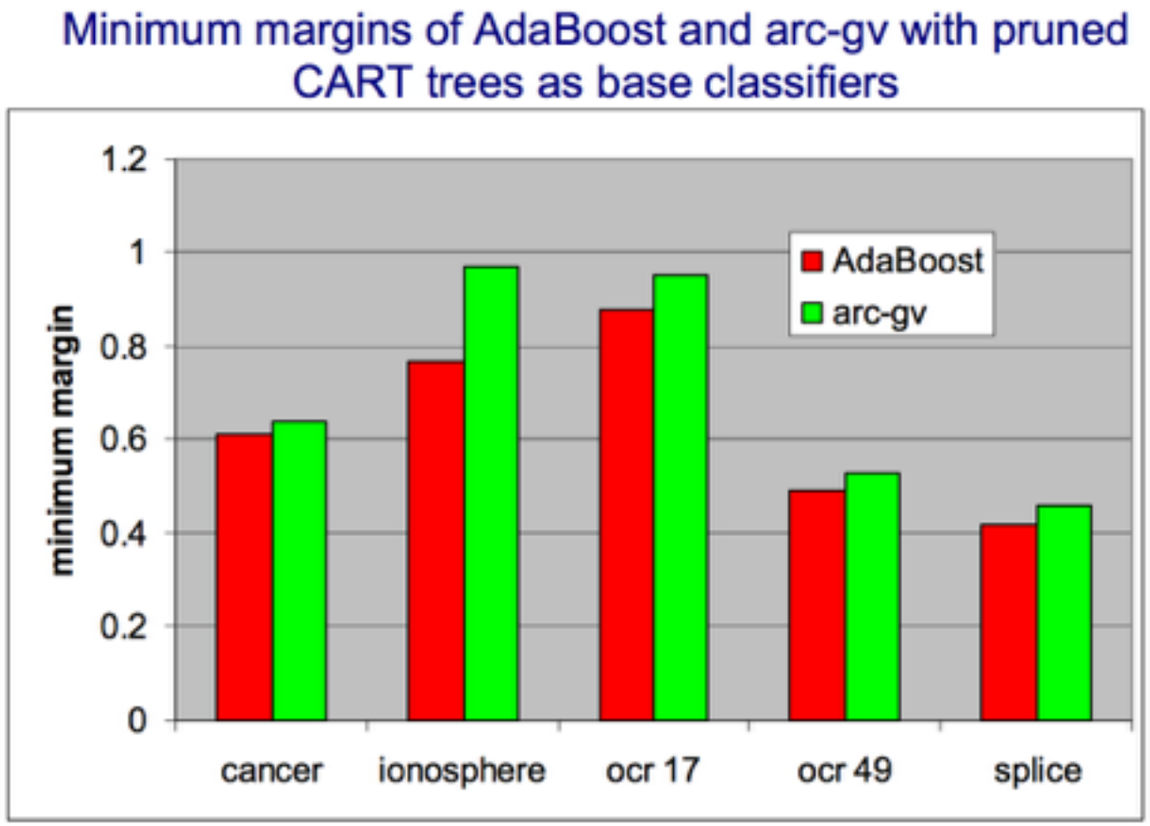
Fig. 3 *Left:* The training and test percent error rates obtained using boosting on an OCR dataset with C4.5 as the base learner. The top and bottom curves are test and training error, respectively. The top horizontal line shows the test error rate using just C4.5. The bottom line shows the final test error rate of AdaBoost after 1000 rounds. *Right:* The margin distribution graph for this same case showing the cumulative distribution of margins of the training instances after 5, 100 and 1000 iterations, indicated by short-dashed, long-dashed (mostly hidden) and solid curves, respectively. (Both figures are reprinted from [31] with permission of the Institute of Mathematical Statistics.)

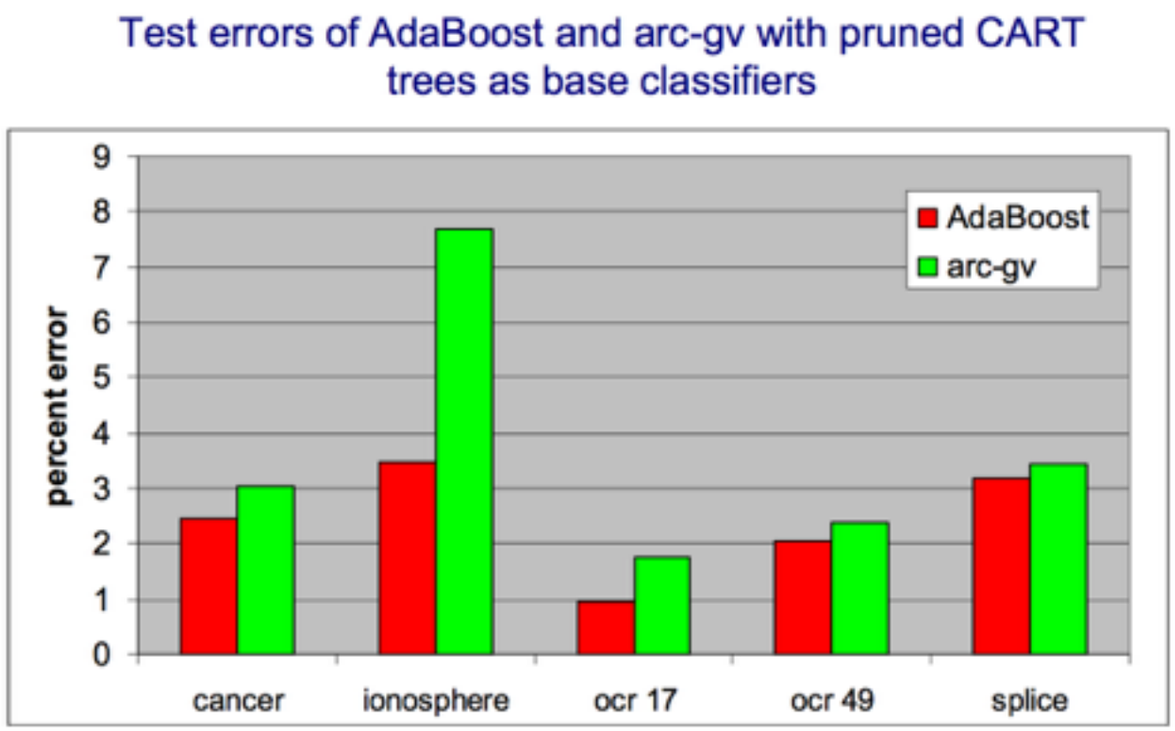
Hypothesis: weak learners with a high margin = low generalization error

arc-gv algorithm developed to specifically increase margin.



Unfortunately, arc-gv shows worse test performance, despite having better margins.





arc-gv runs into problems, like tending to create more complex single classifiers (trees with more depth), thus reducing their generality.

Loss minimization

Some people have noticed that AdaBoost is an algorithm which greedily minimizes the exponential loss function:

$$\frac{1}{m}\sum_{i=1}^{m}\exp(-y_i F(x_i)) \qquad\qquad F(x)=\sum_{t=1}^{T}\alpha_t h_t(x).$$

Does AdaBoost's power come from its choice of loss function?

Mease and Wyner generated 10,000-dimensional vectors, with each dimension having value +1 or -1. 1,000 training samples and 10,000 test samples **x** were randomly generated in this space. The truth value **y** is set equal to either +1 or -1 based on majority vote of three designated coordinates in **x**.

Three methods tested: AdaBoost, gradient descent, and random AdaBoost (instead of finding the best weak learner at each step, you just pick a random one).

	% test error [# rounds]					
exp. loss	exhaustive AdaBoost		gradient descent		random AdaBoost	
10^{-10}	0.0	[94]	40.7	[5]	44.0	[24,464]
10^{-20}	0.0	[190]	40.8	[9]	41.6	[47,534]
10^{-40}	0.0	[382]	40.8	[21]	40.9	[94,479]
10^{-100}	0.0	[956]	40.8	[70]	40.3	[234,654]

Table 1 Results of the experiment described in Section 4. The numbers in brackets show the number of rounds required for each algorithm to reach specified values of the exponential loss. The unbracketed numbers show the percent test error achieved by each algorithm at the point in its run at which the exponential loss first dropped below the specified values. All results are averaged over ten random repetitions of the experiment. (Reprinted from [30] with permission of MIT Press.)

All three methods minimize exponential loss, but only AdaBoost performs well. AdaBoost's results can't be explained just by loss function minimization.

Regularization

It looks like the simplicity of AdaBoost is a big part of its performance. What if we did function minimization again, but attempt regularization by keeping the classifier weights low?

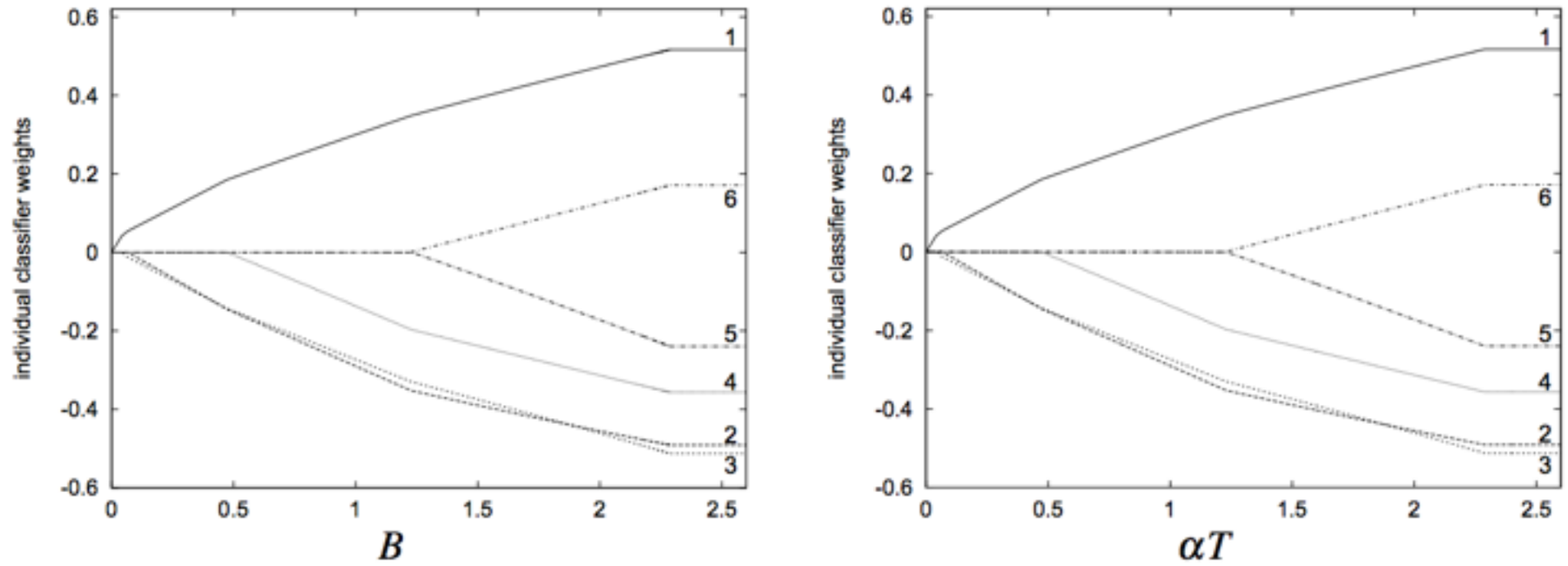


Fig. 4 The trajectories of the weight vectors computed on a benchmark dataset using only six possible weak hypotheses. Trajectories are plotted for ℓ_1-regularized exponential loss as the parameter B varies (left), and for a variant of AdaBoost in which $\alpha_t = \alpha = 10^{-6}$ on every round (right). Each figure includes one curve for each of the six weak hypotheses showing its associated weight as a function of the total weight added. (Reprinted from [30] with permission of MIT Press.)

It looks like AdaBoost “accidentally” regularizes when you halt it after a limited number of rounds. However, this explanation breaks down with increasing rounds, and does not explain why you can train indefinitely and still not overtrain.

Furthermore, this is only true in a modified version of AdaBoost where α is held constant.

So it looks like AdaBoost “accidentally” increases margins, decreases the error function, and performs regularization, but algorithms built to do these things specifically don’t end up doing as well as AdaBoost.

How to break AdaBoost

Adding just a little bit of noise to the training set (mislabelling truth outputs) can completely destroy the effectiveness of AdaBoost. For example, Long and Servedio created an example where a 1% noise caused AdaBoost to have a classifier error of greater than 50%.

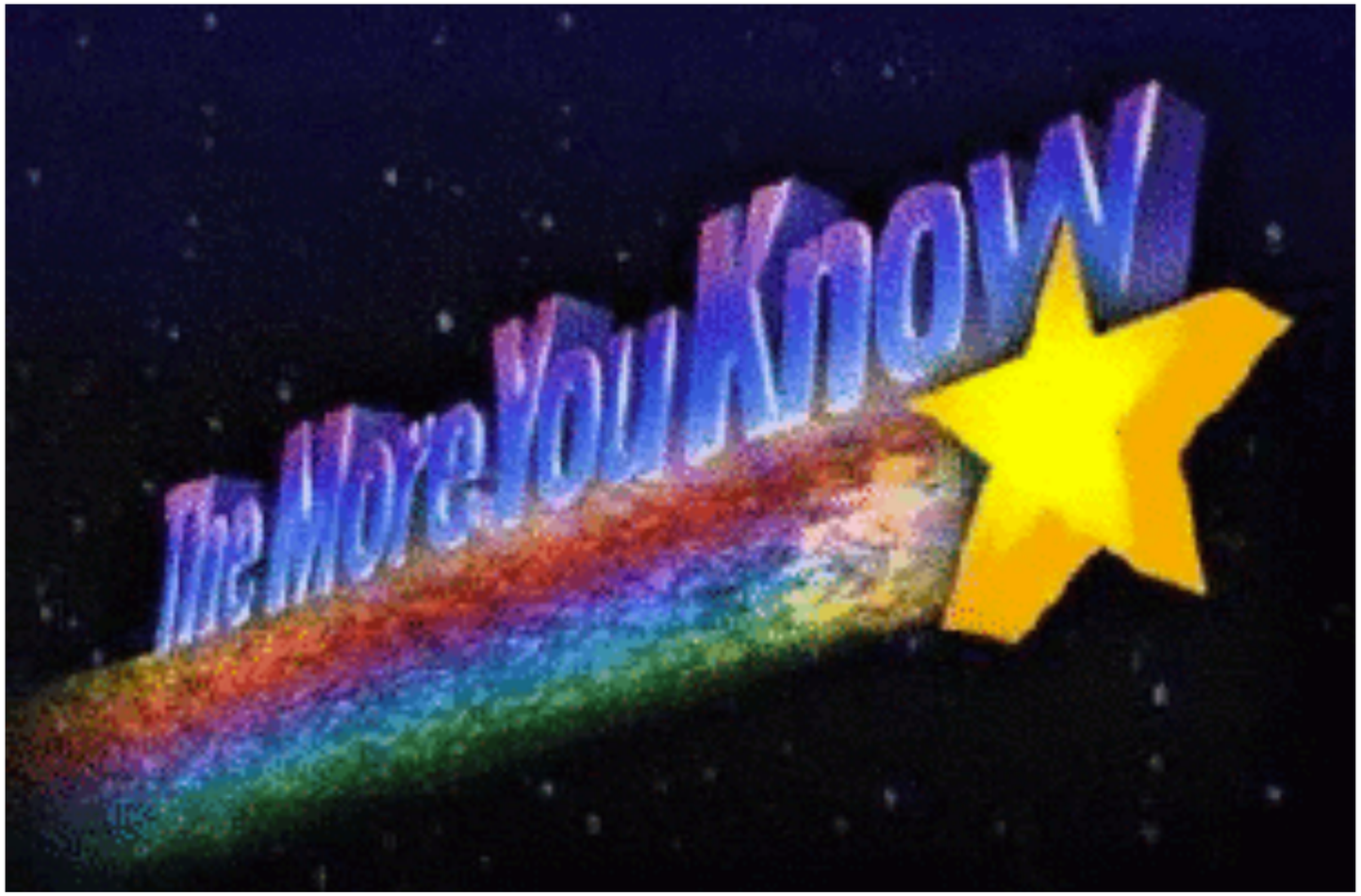
BrownBoost (shown on right) is like AdaBoost, but has the ability to "give up" on poorly fit events.

dataset	noise	AdaBoost	BrownBoost
letter	0%	3.7	4.2
	10%	10.8	7.0
	20%	15.7	10.5
satimage	0%	4.9	5.2
	10%	12.1	6.2
	20%	21.3	7.4

Table 2 The results of running AdaBoost and BrownBoost on the "letter" and "satimage" benchmark datasets. After converting to binary by combining the classes into two arbitrary groups, each dataset was split randomly into training and test sets, and corrupted for training with artificial noise at the given rates. The entries of the table show percent error on *uncorrupted* test examples. All results are averaged over 50 random repetitions of the experiment. (These experiments were conducted by Evan Ettinger, Sunsern Cheamanunkul and Yoav Freund, and were reported in [30].)

Lots of other extensions to AdaBoost - mostly focused on different regression methods and ways of calculating α.

To learn more, consult Wikipedia or your local public library.



References

Mostly stolen from “Explaining AdaBoost” by Robert E. Schapire - http://rob.schapire.net/papers/explaining-adaboost.pdf

Brief example of usage with BDT - https://indico.scc.kit.edu/indico/event/48/session/4/contribution/35/material/slides/0.pdf

AdaBoost compared with other algorithms - http://www.37steps.com/exam/adaboost_comp/html/adaboost_comp.html

“How Boosting the Margin Can Also Boost Classifier Complexity” by Reyzin and Schapire - http://www.levreyzin.com/presentations/ReyzinSch06_icml_presentation.pdf

Wikipedia article on AdaBoost