

The power of voting

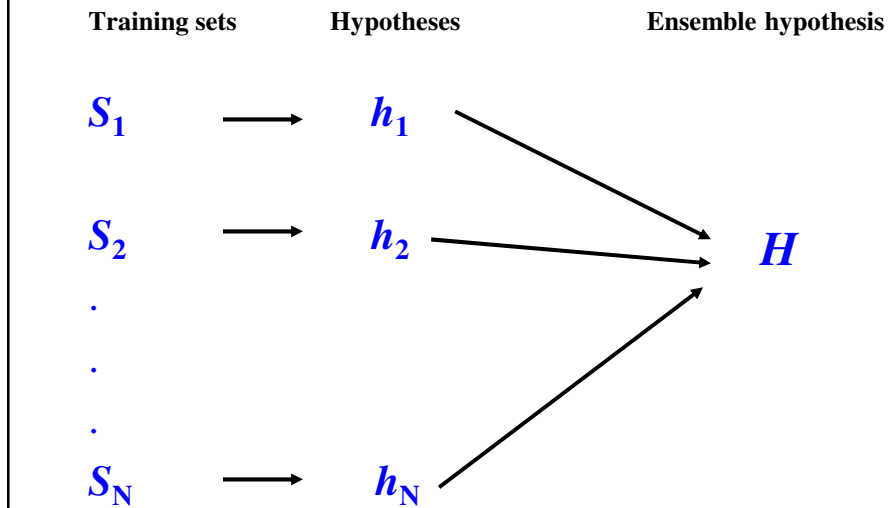
What movie won the Oscar for best picture in 2008?

- (a) Atonement
- (b) Juno
- (c) There Will Be Blood
- (d) No Country for Old Men
- (e) Michael Clayton

What year did Albert Einstein publish his special theory of relativity?

- (a) 1901
- (b) 1905
- (c) 1918
- (d) 1924
- (e) 1931

Ensemble learning



Advantages of ensemble learning

- Can be very effective at reducing generalization error! (E.g., by voting.)
- Ideal case: the h_i have independent errors

Example: Given three hypotheses, h_1, h_2, h_3 .

Suppose each h_i has 40% generalization error, and assume errors are independent.

Now suppose we classify x as the majority vote of h_1, h_2 , and h_3 . What is probability that the classification is correct?

In general, if hypotheses h_1, \dots, h_n all have generalization error $>1/2$, what is probability that a majority vote will be correct?

Possible problems with ensemble learning

- Errors most likely are not independent
- Training time and classification time are increased by a factor of N .
- Hard to explain how ensemble hypothesis does classification.
- How to get enough data to create N separate data sets, S_1, \dots, S_N ?

1797 items

0 (a)	1 (b)	2 (c)	3 (d)	4 (e)	5 (f)	6 (g)	7 (h)	8 (i)	9 (j)	<-classified as
160				3	1			6		(a): class 0
	118	8	1	4	9	6	1	19	16	(b): class 1
1	16	148	5		1		1	4	1	(c): class 2
2	7	6	123	2	1		15	10	17	(d): class 3
1	9			148	2	5	14	1	1	(e): class 4
1	4	7		2	148	4	3	3	10	(f): class 5
2	1	1	1	8		167		1		(g): class 6
2	9	2		4	2		129	7	24	(h): class 7
1	24	17	1	6				123	2	(i): class 8
2	9	2	7	4	7		3	3	143	(j): class 9

- Three popular methods:

– **Voting:**

- Train classifier on n different training sets S_i to obtain n different classifiers h_i . Assume h_i returns +1 or -1.
- For a new instance x , define $H(x)$ as:

$$H(x) = \operatorname{sgn}\left(\sum_{i=1}^n h_i(x)\right)$$

– **Bagging (Breiman, 1990s):**

- Same as voting, but to create S_i , create “bootstrap replicates” of original training set S

– **Boosting (Schapire & Freund, 1990s)**

- To create S_i , reweight examples in original training set S as a function of whether or not they were misclassified on the previous round.

Bagging
 (“Bootstrap Aggregation”)
L. Breiman, 1994

- **Problem with voting:** Usually don’t have enough data for N independent data sets S_i .
- **Solution:** Use “bootstrapping” as way to simulate this from single data set S .

Bootstrapping

- Create N training sets from original training set S as follows:
 - For each S_i , select m examples at random with replacement.
- Each of these new training sets has approximately the same distribution as that underlying S .

Stable versus unstable learning procedures

- Note that bagging will be most effective when small changes in S produce significant changes in h . (“Unstable learning procedures”.)
- “The evidence, both experimental and theoretical, is that bagging can push a good but unstable procedure a significant step towards optimality. On the other hand, it can slightly degrade the performance of stable procedures.” (Breiman, 1994)

Sources of Classifier Error: Bias, Variance, and Noise

Bias:

- Classifier cannot learn the correct hypothesis (no matter what training data is given), and so incorrect hypothesis h is learned. The **bias** is the average error of h over all possible training sets.

Variance:

- Training data is not representative enough of all data, so the learned classifier h varies from one training set to another.

• Noise:

- Training data contains errors, so incorrect hypothesis h is learned.

Why does bagging work?

Two different explanations:

1. Reduces variance (Breiman, 1996)
 - Error is due to noise, bias, and variance
 - Bootstrap sampling simulates sampling variance
 - Averaging over these samples can reduce error from variance, especially for unstable learning procedures

Boosting

- Main idea:
 - Often hard to learn a single good hypothesis or model of data.
 - Easier to learn “rough rules of thumb” (“weak hypotheses”) that are only moderately accurate.
- Boosting:
 - A method for combining different weak hypotheses (training error close to but less than 50%) to produce a strong hypothesis (training error close to 0%)

Sketch of algorithm

- Repeatedly run learning algorithm on training sets S_t to produce h_1, h_2, \dots, h_T .
- At each step, derive S_t from S by choosing examples probabilistically according to probability distribution D_t . Use S_t to learn h_t .
- At each step, derive D_{t+1} by giving more probability to examples that were misclassified at step t .
- The final ensemble classifier H is a weighted sum of the h_t 's, with each weight being a function of the corresponding h_t 's error on its training set.

Adaboost algorithm

- Given $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ where $\mathbf{x} \in X, y_i \in \{+1, -1\}$
- Initialize $D_1(i) = 1/m$. (Uniform distribution over training examples)

- For $t = 1, \dots, T$:
 - Learn base hypothesis h_t using training set S_t (select examples, with replacement, from S according to D_t).
 - Let the error of h_t be denoted as ϵ_t :

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

- If $\epsilon_t \geq 0.5$, set T to $T-1$ and break from loop.

- Let

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

– 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 probability distribution.

- At the end of T iterations of this algorithm, we have

$$h_1, h_2, \dots, h_T$$

We also have

$$\alpha_1, \alpha_2, \dots, \alpha_T, \text{ where } \alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

- Ensemble classifier:

$$H(\mathbf{x}) = \text{sgn} \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$$

- Note that hypotheses with lower training error on their training sets are weighted more strongly.

A Simple Example

S	D_1	S_1	$h_1(x)$	D_2
$x_1, +1$	1/4	$x_1, +1$	+1	
$x_2, -1$	1/4	$x_1, +1$	+1	
$x_3, -1$	1/4	$x_3, -1$	+1	
$x_4, -1$	1/4	$x_4, -1$	-1	

$$\epsilon_1 = 0.25$$

$$\alpha_1 = \frac{1}{2} \ln \left(\frac{1 - \epsilon_1}{\epsilon_1} \right) = 0.55$$

$$D_2(i) = \frac{D_1(i) \exp(-\alpha_1 y_i h_1(x_i))}{Z_1}$$

Why does Adaboost work?

- Decreases variance **and** bias
- But...why doesn't increasing number of classifiers in ensemble (i.e., increasing complexity of ensemble classifier) lead to overfitting?
- Hypothesis: Adaboost avoids overfitting by increasing "margin distribution".
- Note: This is a different (but related) notion of "margin" from that in SVMs.

How Boosting Avoids Overfitting: Margin Theory (Schapire et al.)

Result of boosting:

$$H(x) = \text{sgn}\left(\sum_t \alpha_t h_t(\mathbf{x})\right), \text{ where}$$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right),$$

and ε_t is the error on the training set at time step t .

- Definition of **margin** of example (\mathbf{x}, y) :

$$\text{margin}(\mathbf{x}, y) = \frac{y \sum_t \alpha_t h_t(\mathbf{x})}{\sum_t \alpha_t}$$

- The magnitude of the margin is the strength of agreement of base classifiers, and the sign indicates whether the ensemble produced a correct classification.

Example:

E.g.,

$$.4 h_1(\mathbf{x})$$

$$.2 h_2(\mathbf{x})$$

$$.1 h_3(\mathbf{x})$$

Test set and results:

$$y(\mathbf{x}_1) = -1$$

$$h_1(\mathbf{x}_1) = 1, h_2(\mathbf{x}_1) = 1, h_3(\mathbf{x}_1) = -1$$

$$y(\mathbf{x}_2) = 1$$

$$h_1(\mathbf{x}_2) = -1, h_2(\mathbf{x}_2) = -1, h_3(\mathbf{x}_2) = 1$$

$$y(\mathbf{x}_3) = 1$$

$$h_1(\mathbf{x}_3) = 1, h_2(\mathbf{x}_3) = 1, h_3(\mathbf{x}_3) = 1$$

Calculate margin of each training example.

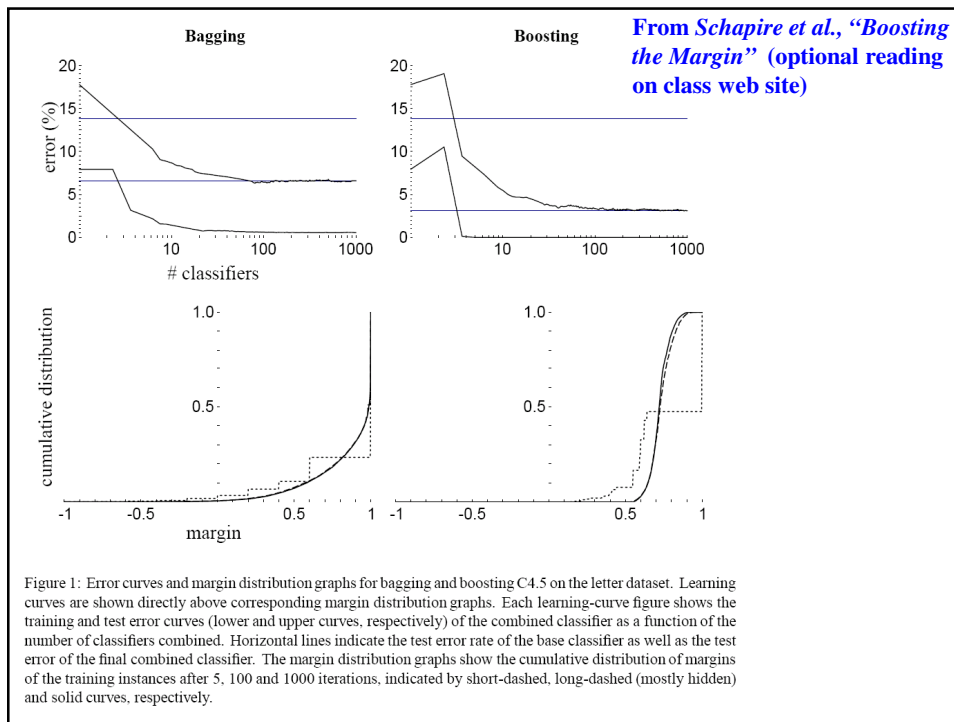
- Schapire et al. proved a new bound on generalization error, based on margins:

For any θ , $0 < \theta$,

$$\Pr_D(H(x) \neq y) \leq \Pr_S[\text{margin}(\mathbf{x}, y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{VC(H)}{m\theta^2}}\right)$$

Note: does not depend on T!

- When θ is large, second term is minimized, but that tends to increase first term. So want big margins! In particular, want minimum margin to be large.



- Leo Brieman designed a variation of adaboost called "arc-gv", that explicitly maximizes minimum margin. He compared arc-gv with adaboost on boosting decision trees, keeping the complexity (size) of the trees constant.
- Margin theory would predict that arc-gv should produce ensemble classifiers with smaller generalization error than adaboost.
- But....

- Indeed the minimum margin was higher with arc-gv:

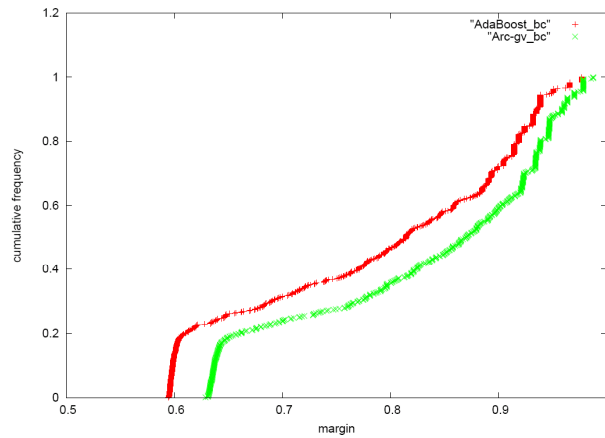
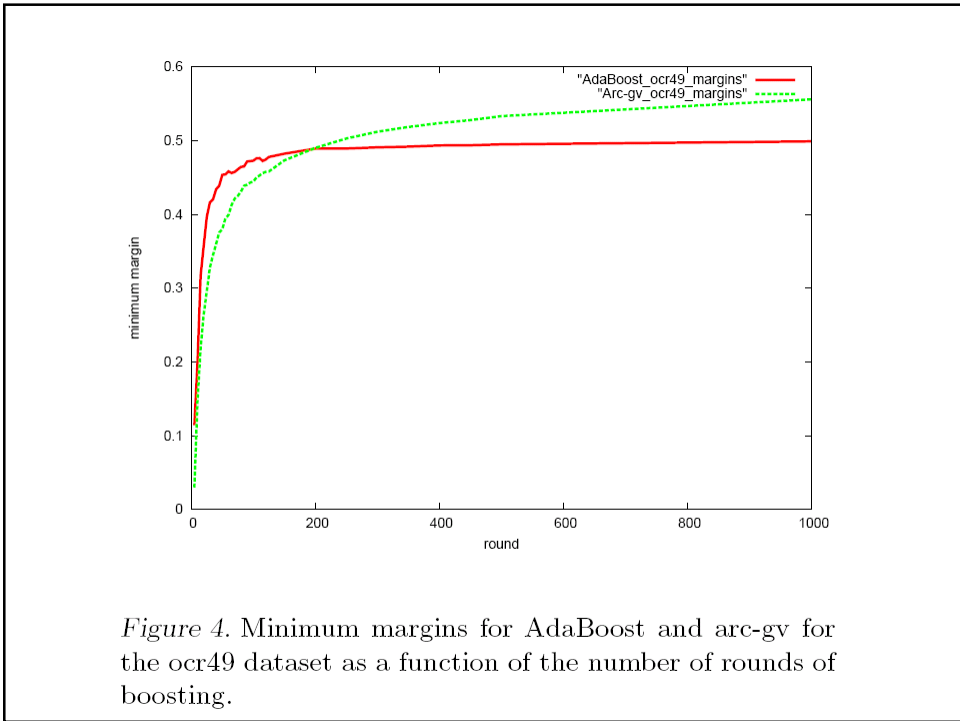
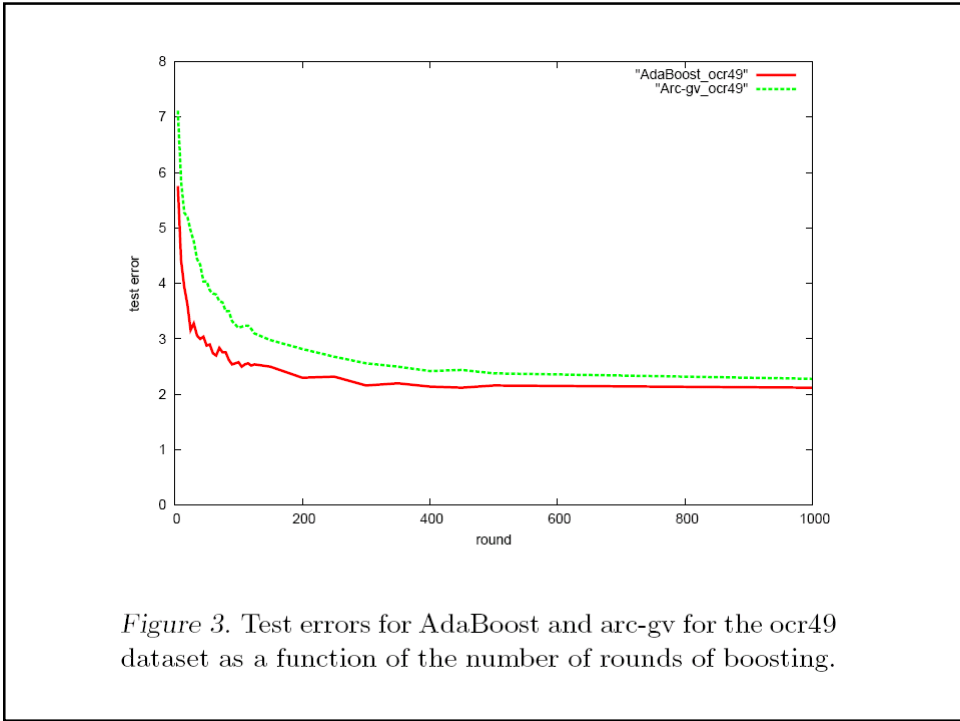


Figure 2. Cumulative margins for AdaBoost and arc-gv for the breast cancer dataset after 500 rounds of boosting.

- However, the generalization errors did not seem to agree with the margin theory

Table 2. Test errors, averaged over 10 trials, of AdaBoost and arc-gv, run for 500 rounds using CART decision trees pruned to 16 leaf nodes as base classifiers.

	cancer	ion	ocr 17	ocr 49	splice
AdaBoost	2.46	3.46	0.96	2.04	3.18
arc-gv	3.04	7.69	1.76	2.38	3.45



Reyzin and Schapire's explanation

From paper: "How boosting the margin can also boost classifier complexity" (optional reading on class website)

- The bound on generalization error involves more than simply minimum margin:

$$\Pr_D(H(x) \neq y) \leq \Pr_S[\text{margin}(\mathbf{x}, y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{VC(H)}{m\theta^2}}\right)$$

- It involves entire margin distribution, training set size, and VC dimension of hypothesis space.
- Breiman controlled VC dimension by keeping decision trees at same size. However, he did not control *depth* of the trees. Deeper trees tend to overfit more than shallow trees.
- Reyzin and Schapire found that arc-gv generates significantly deeper trees than AdaBoost.

	test error		minimum margin		tree depth	
	arc-gv	AdaBoost	arc-gv	AdaBoost	arc-gv	AdaBoost
breast cancer	3.04	2.46	0.64	0.61	9.71	7.86
ionosphere	7.69	3.46	0.97	0.77	8.89	7.23
ocr 17	1.76	0.96	0.95	0.88	7.47	7.41
ocr 49	2.38	2.04	0.53	0.49	7.39	6.70
splice	3.45	3.18	0.46	0.42	7.12	6.67

Table 3. Test errors, minimum margins, and tree depths, averaged over 10 trials, of AdaBoost and arc-gv, run for 500 rounds using CART decision trees pruned to 16 leaf nodes as base classifiers. (For 100 rounds, we also saw arc-gv producing deeper trees on average.)

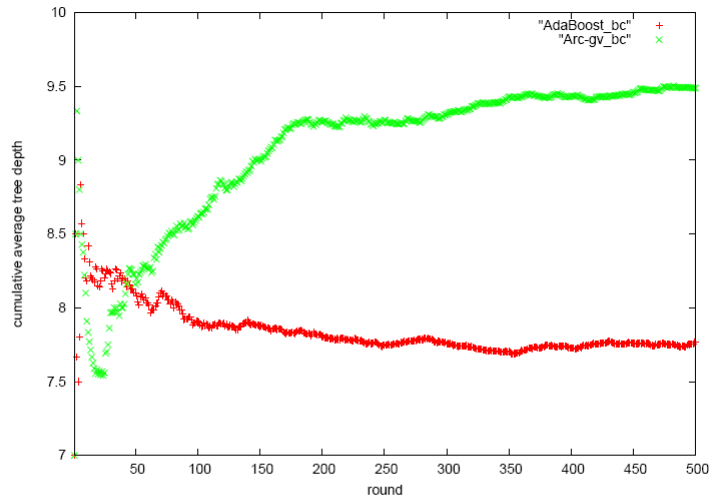


Figure 5. Cumulative average of decision tree depth for AdaBoost and arc-gv for the breast cancer set for 500 rounds of boosting.

Table 4. Percent test and training errors per generated tree, and their differences, averaged over all CART decision trees generated in 500 rounds of boosting, over 10 trials.

	AdaBoost			arc-gv		
	test	train	diff	test	train	diff
cancer	13.2	9.7	3.5	10.4	6.3	4.1
ion	19.8	10.9	8.9	12.5	2.6	9.9
ocr 17	5.6	3.7	1.9	2.6	0.6	2.0
ocr 49	24.8	21.1	3.7	21.9	17.8	4.1
splice	27.7	23.4	4.3	23.9	19.2	4.7

Claim: arc-gv is overfitting to the training data, due to higher complexity of trees

- New experiment: Control both size and depth of decision trees by using “decision stumps” – one-level binary decision trees.

	test error		minimum margin		average margin	
	arc-gv	AdaBoost	arc-gv	AdaBoost	arc-gv	AdaBoost
cancer	4.15	4.29	-.01	-.06	.07	.27
ionosphere	10.27	9.58	.01	.03	.09	.20
ocr 17	1.12	1.10	.03	.06	.14	.36
ocr 49	6.38	6.28	-.02	-.07	.05	.20
splice	7.22	6.79	-.01	-.07	.06	.21

Table 5. Test errors, minimum margins, and average margins averaged over 100 trials, of AdaBoost and arc-gv, run for 100 rounds using decision stumps as weak learners.

