Ensembles
(Bagging, Boosting, and all that)

Old View

- Learn **one** good model

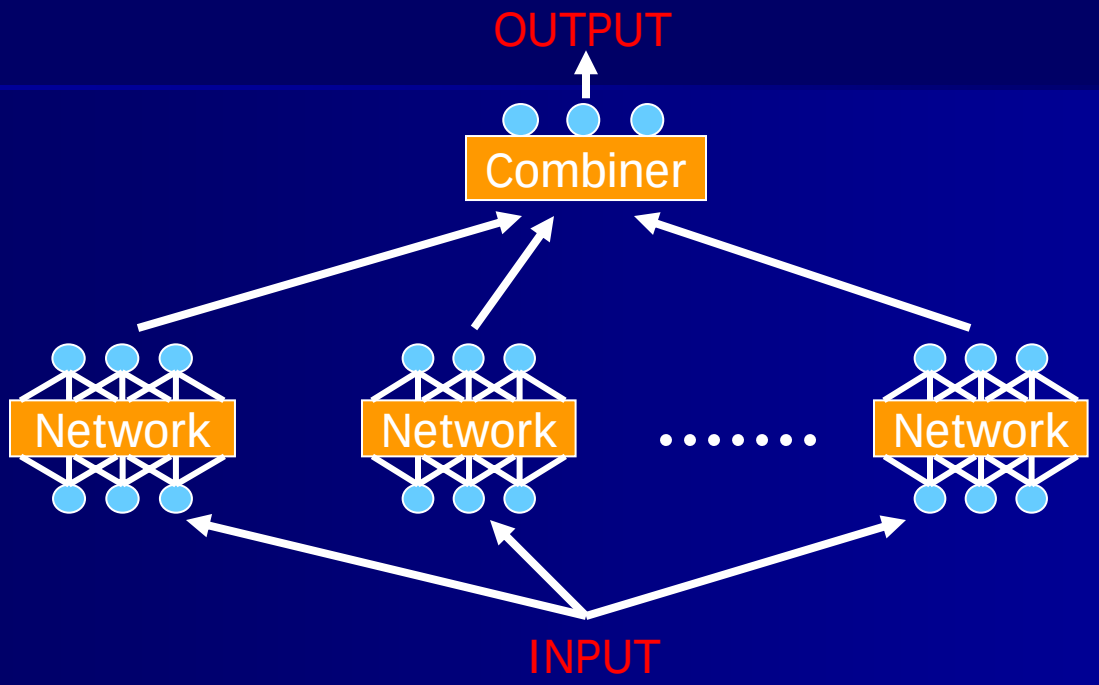
Naïve Bayes, k-NN, neural net
d-tree, SVM, etc

New View

- Learn a good **set** of models

Probably best example of interplay between "theory & practice" in Machine Learning

Ensembles of Neural Networks
(or any supervised learner)



- Ensembles often produce accuracy gains of 5-10 percentage points!
- Can combine "classifiers" of various types
  - E.g., decision trees, rule sets, neural networks, etc.

Combining Multiple Models

Three ideas

1. Simple (unweighted) votes
   - Standard choice
2. Weighted votes
   - e.g., weight by tuning-set accuracy
3. Train a combining function
   - Prone to overfitting?
   - "Stacked generalization" (Wolpert)

Some Relevant Early Papers

- Hansen & Salamen, PAMI:20, 1990
  - If (a) the combined predictors have errors that are independent from one another
  - And (b) prob any given model correct predicts any given testset example is > 50%, then

$$\lim_{N \to \infty} (\text{test set error rate of } N \text{ predictors}) = 0$$

  - Think about flipping *N* coins, each with prob > ½ of coming up heads – what is the prob *more than half* will come up heads?

Some Relevant Early Papers

- Schapire, MLJ:5, 1990 ("Boosting")
  - If you have an algorithm that gets > 50% on any distribution of examples, you can create an algorithm that gets > (100% - ε), for any $\varepsilon > 0$
  - Need an infinite (or very large, at least) source of examples
    - Later extensions (eg, AdaBoost) address this weakness
- Also see Wolpert, "Stacked Generalization," *Neural Networks*, 1992

Some Methods for Producing "Uncorrelated" Members of an Ensemble

- *k* times randomly choose (with replacement) *N* examples from a training set of size *N*
  - give each training set to a std ML algo
  - "Bagging" by Brieman (MLJ, 1996)
  - Want unstable algorithms

- Reweight examples each cycle (if wrong, increase weight; else decrease weight)
  - "AdaBoosting" by Freund & Schapire (1995, 1996)

Some More Methods for Producing "Uncorrelated" Members of an Ensemble

- Directly optimize accuracy + diversity
  - Opitz & Shavlik (1995; used genetic algo's)
  - Melville & Mooney (2004-5; DECORATE algo)
- Different number of hidden units in a neural network, different *k* in *k*-NN, tie-breaking scheme, example ordering, etc
  - Various people
  - See 2005 and 2006 papers of Caruana's group for large-scale empirical study of ensembles

Variance / Diversity Creating Methods (cont.)

- Assign each category an *error correcting code*, and train on each bit separately

  Dietterich et al. (ICML 1995)

Cat1 =	1	1	1	0	1	1	1
Cat2 =	1	1	0	1	1	0	0
Cat3 =	1	0	1	1	0	1	0
Cat4 =	0	1	1	1	0	0	1

Predicting 5 of 7 bits correctly suffices

Want: Large Hamming distance between rows
Large Hamming distance between columns

Random Forests
(Breiman, MLJ 2001; related to Ho, 1995)

A variant of BAGGING

Algorithm

Let N = # of examples
F = # of features
i = some number << F

Repeat k times

(1) Draw with replacement N examples, put in train set

(2) Build d-tree, but in each recursive call

- Choose (w/o replacement) i features
- Choose best of these i as the root of this (sub)tree

(3) Do NOT prune

More on Random Forests

- Increasing *i*
  - Increases correlation among individual trees (BAD)
  - Also increases accuracy of individual trees (GOOD)
- Can use tuning set to choose good setting for *i*
- Overall, random forests
  - Are very fast (e.g., 50K examples, 10 features, 10 trees/min on 1 GHz CPU in 2004)
  - Deal with large # of features
  - Reduce overfitting substantially
  - Work very well in practice

AdaBoosting
(Freund & Schapire)

$W_{i,j}$ = weight on ex_j on cycle i

Initially weight all ex's equally (ie, 1/N, N=#examples)

1. Let H_i = concept/hypothesis learned on current weighted train set
2. Let ε_I = weighted error of H_i on current train set
3. If ε_i>1/2, return {H_1, H_2, ..., H_{i-1}} (all previous hypotheses)
4. Reweight correct ex's:

   Note: since ε_i<1/2, w_{i+1}< w_i

$$w_{i+1,j} = \frac{\varepsilon_i}{1-\varepsilon_i} \times w_{i,j}$$

5. Renormalize, so sum wgts = 1
6. i ← i+1, goto 1

Using the Set of Hypothesis Produced by AdaBoost

Output for example x =

$$\underset{y \in categories}{\arg\max} \sum_{i=1}^{\#hypo's} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right) \times \delta\left(h_i(x) = y\right)$$

where δ(false) $\equiv$ 0, δ(true) $\equiv$ 1

- ie, count *weighted* votes for hypotheses that predict category y for input x

Dealing with Weighted Examples in an ML Algo

Two approaches

1. Sample from this probability distribution and train as normal (ie, create prob dist from wgts, then sample to create an unweighted train set)
2. Alter learning algorithm so it counts weighted examples and not just examples

   eg) from accuracy = # correct / # total

   to weighted accuracy = $\sum w_i$ of correct / $\sum w_i$ of all

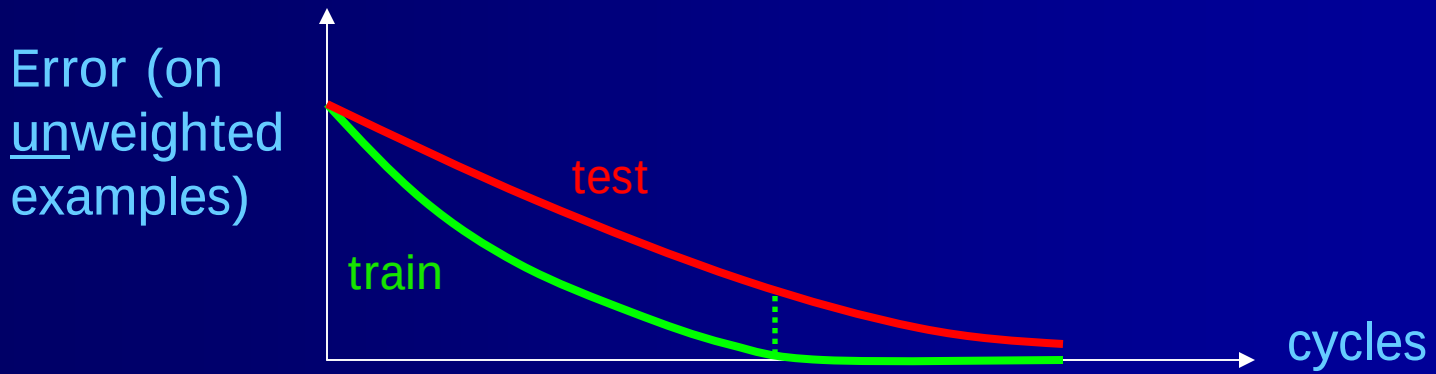
#2 preferred – avoids sampling effects

AdaBoosting & ID3

- Apply to PRUNED trees* – otherwise no trainset error! (Can avoid $\varepsilon_i = 0$ via *m*-est's)
- ID3's calc's all based on weighted sums, so easy to extend to weighted examples

Boosting & Overfitting

Often get better test-set results, even when (and after) train error is ZERO



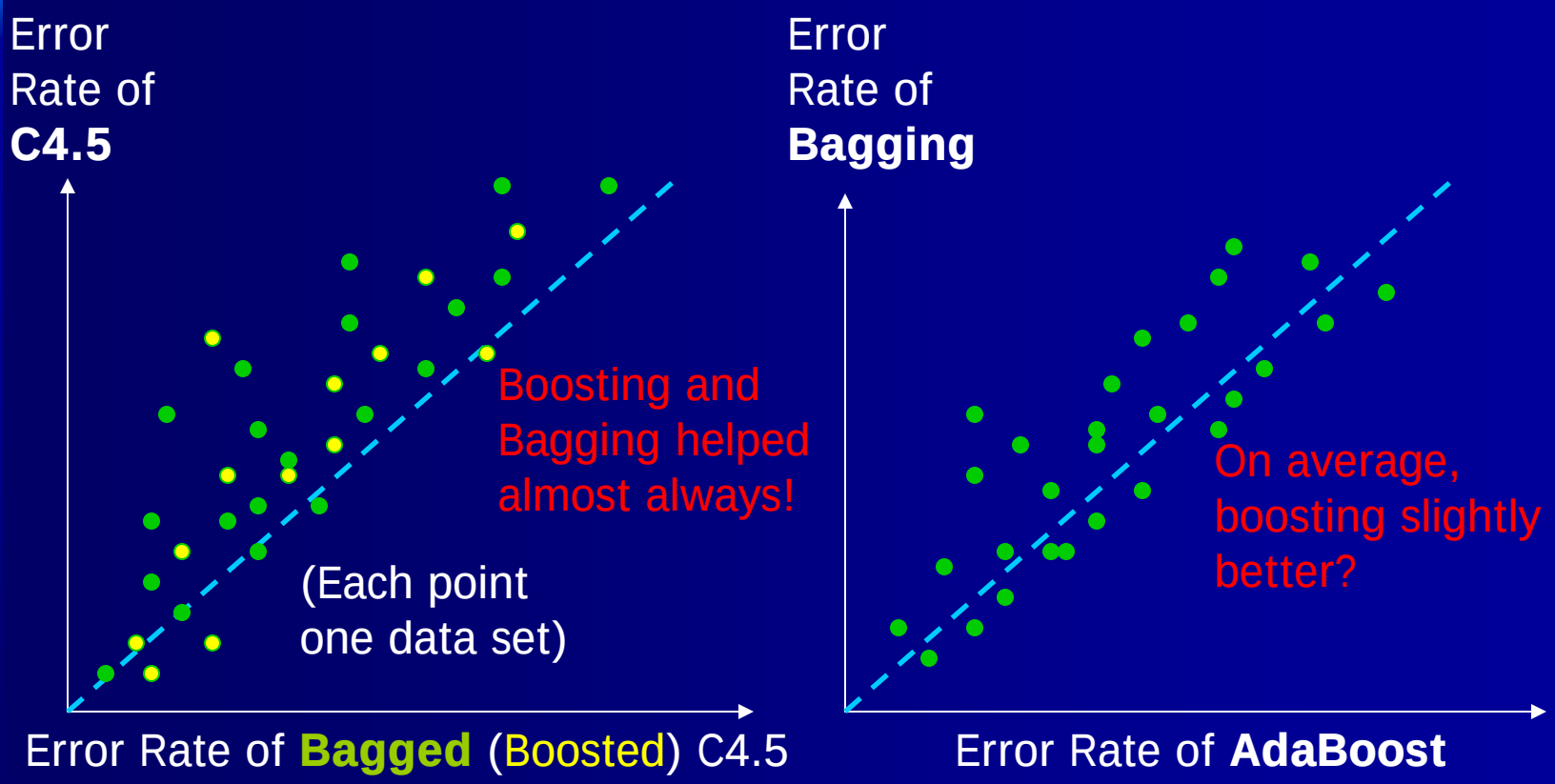
Hypothesis (see papers by Schurmanns or Schapire)

Still improving number/strength of votes even though getting all train-set ex's correct

Wider "margins" between pos and neg ex's – relates to SVM's

Empirical Studies

(from Freund & Schapire; reprinted in Dietterich's AI Mag paper)



Large Empirical Study of Bagging vs. Boosting

Opitz & Maclin (UW CS PhD's),

JAIR Vol 11, pp 169-198, 1999

www.jair.org/abstracts/opitz99a.html

- Bagging almost always better than single D-tree or ANN (artificial neural net)
- Boosting can be much better than Bagging
- However, boosting can sometimes be harmful (too much emphasis on "outliers"?)

Thought Experiment
(possible class project)

- Consider a *learning curve*
- Plot, averaged across many datasets, error rates of
  - *Best single model*
  - *Some ensemble method*
- We know that for many #examples, ensembles have lower error rates
- What happens as #examples $\rightarrow \infty$?

Boosting / Bagging / Etc Wrapup

- An easy to use and usually highly effective technique
  - always consider it (bagging, at least) when applying ML to practical problems
- Does reduce "comprehensibility" of models
  - see work by Craven & Shavlik though ("rule extraction")
- Also an increase in runtime, but cycles usually much cheaper than examples