

Machine Learning in Real World: C4.5

Industrial-strength algorithms

- For an algorithm to be useful in a wide range of real-world applications it must:
 - Permit numeric attributes
 - Allow missing values
 - Be robust in the presence of noise
 - Be able to approximate arbitrary concept descriptions (at least in principle)
- Basic schemes need to be extended to fulfill these requirements

C4.5 History

- ID3, CHAID – 1960s
- C4.5 innovations (Quinlan):
 - permit numeric attributes
 - deal sensibly with missing values
 - pruning to deal with for noisy data
- C4.5 - one of best-known and most widely-used learning algorithms
 - Last research version: C4.8, implemented in Weka as J4.8 (Java)
 - Commercial successor: C5.0 (available from Rulequest)

3

Numeric attributes

- Standard method: binary splits
 - E.g. temp < 45
- Unlike nominal attributes, every attribute has many possible split points
- Solution is straightforward extension:
 - Evaluate info gain (or other measure) for every possible split point of attribute
 - Choose "best" split point
 - Info gain for best split point is info gain for attribute
- Computationally more demanding

witten & eibe

4

Weather data – nominal values

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	Normal	False	Yes
...

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

Weather data - numeric

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

```
If outlook = sunny and humidity > 83 then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity < 85 then play = yes
If none of the above then play = yes
```

Example

- Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- E.g. temperature < 71.5 : yes/4, no/2
temperature ≥ 71.5 : yes/5, no/3
- Info([4,2],[5,3])
= $6/14 \text{ info}([4,2]) + 8/14 \text{ info}([5,3])$
= 0.939 bits
- Place split points halfway between values
- Can evaluate all split points in one pass!

witten & eibe

7

Avoid repeated sorting!

- Sort instances by the values of the numeric attribute
 - Time complexity for sorting: $O(n \log n)$
- Q. Does this have to be repeated at each node of the tree?**
- A: No! Sort order for children can be derived from sort order for parent
 - Time complexity of derivation: $O(n)$
 - Drawback: need to create and store an array of sorted indices for each numeric attribute

witten & eibe

8

More speeding up

- Entropy only needs to be evaluated between points of different classes (Fayyad & Irani, 1992)

value	64	65	68	69	70	71	72	72	75	75	80	81	83	85
class	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

Potential optimal breakpoints

Breakpoints between values of the same class cannot be optimal

9

Missing as a separate value

- Missing value denoted "?" in C4.X
- Simple idea: treat missing as a separate value
- Q: When this is not appropriate?
- A: When values are missing due to different reasons
 - Example 1: gene expression could be missing when it is very high or very low
 - Example 2: field **IsPregnant**=missing for a male patient should be treated differently (no) than for a female patient of age 25 (unknown)

10

Missing values - advanced

Split instances with missing values into pieces

- A piece going down a branch receives a weight proportional to the popularity of the branch
- weights sum to 1
- Info gain works with fractional instances
 - use sums of weights instead of counts
- During classification, split the instance into pieces in the same way
 - Merge probability distribution using weights

Pruning

- Goal: Prevent overfitting to noise in the data
- Two strategies for “pruning” the decision tree:
 - ◆ *Postpruning* - take a fully-grown decision tree and discard unreliable parts
 - ◆ *Prepruning* - stop growing a branch when information becomes unreliable
- Postpruning preferred in practice—prepruning can “stop too early”

Post-pruning

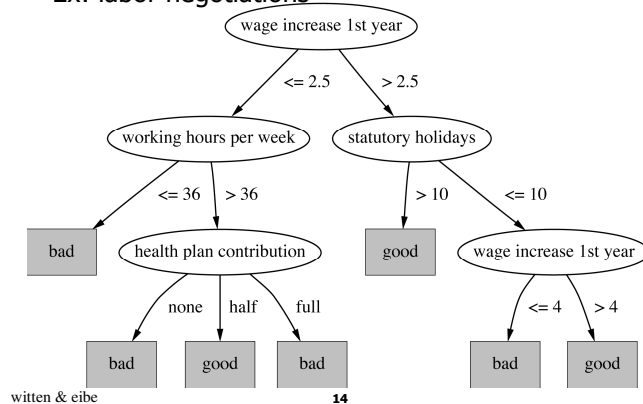
- First, build full tree
- Then, prune it
 - Fully-grown tree shows all attribute interactions
- Two pruning operations:
 1. *Subtree replacement*
 2. *Subtree raising*
- Possible strategies:
 - error estimation
 - significance testing
 - MDL principle

witten & eibe

13

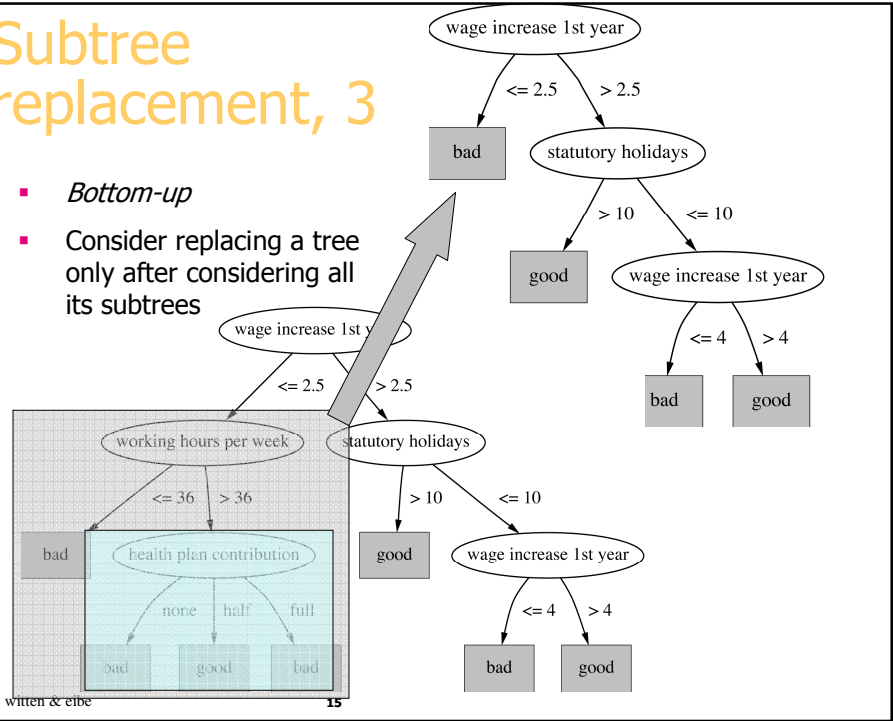
Subtree replacement, 1

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees
- Ex: labor negotiations



Subtree replacement, 3

- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees

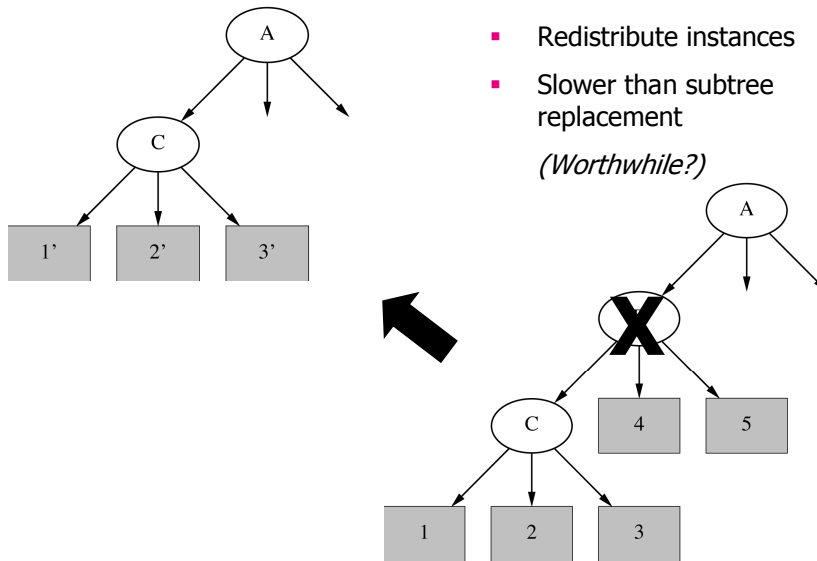


witten & eibe

15

*Subtree raising

- Delete node
- Redistribute instances
- Slower than subtree replacement
(Worthwhile?)



witten & eibe

16

Estimating error rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
 - Q: Why it would result in very little pruning?*
- Use hold-out set for pruning ("reduced-error pruning")
- C4.5's method
 - Derive confidence interval from training data
 - Use a heuristic limit, derived from this, for pruning
 - Standard Bernoulli-process-based method
 - Shaky statistical assumptions (based on training data)

Expected Error Pruning

- Approximate expected error assuming that we prune at a particular node.
- Approximate backed-up error from children assuming we did not prune.
- If expected error is less than backed-up error, prune.

Static Expected Error

- If we prune a node, it becomes a leaf labeled, C
- What will be the expected classification error at this leaf?

$$E(S) = \frac{N - n + k - 1}{N + k}$$

S is the set of examples in a node
 k is the number of classes
 N examples in S
 C the majority class in S
 n out of N examples in S belong to C

This is called Laplace error estimate – it is based on the assumption that the distribution of probabilities that examples will belong to different classes is uniform.

19

Backed-Up Error

- For a non-leaf node
- Let children of Node be Node1, Node2, etc

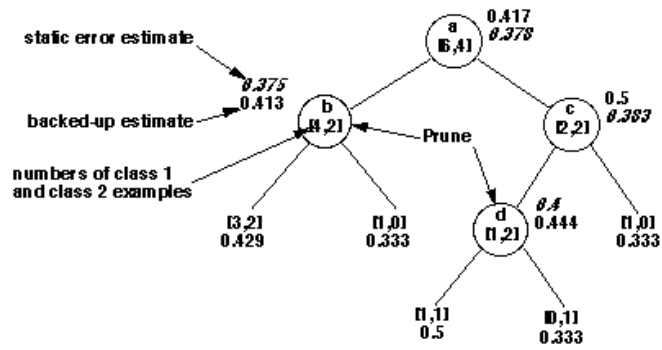
$$BackedUpError(Node) = \sum_i P_i \times Error(Node_i)$$

Probabilities can be estimated by relative frequencies of attribute values in sets of examples that fall into child nodes

$$Error(Node) = \min(E(Node), BackedUpError(Node))$$

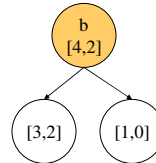
20

Example



21

Example Calculation



Error Calculation for Pruning Example

- Left child of b has class frequencies $[3, 2]$

$$E = \frac{N - n + k - 1}{N + k} = \frac{5 - 3 + 2 - 1}{5 + 2} = 0.429$$

- Right child has error of 0.333, calculated in the same way
- Static error estimate $E(b)$ is 0.375, again calculated using the Laplace error estimate formula, with $N=6$, $n=4$, and $k=2$.
- Backed-up error is:

$$\text{BackedUpError}(b) = (5/6) \times 0.429 + (1/6) \times 0.333 = 0.413$$

(5/6 and 1/6 because there are 4+2=6 examples handled by node b , of which 3+2=5 go to the left subtree and 1 to the right subtree.

- Since backed-up estimate of 0.413 is greater than static estimate of 0.375, we prune the tree and use the static error of 0.375

22

*Complexity of tree induction

- Assume
 - m attributes
 - n training instances
 - tree depth $O(\log n)$
- Building a tree $O(m n \log n)$
- Subtree replacement $O(n)$
- Subtree raising $O(n (\log n)^2)$
 - Every instance may have to be redistributed at every node between its leaf and the root: $O(n \log n)$
 - Cost for redistribution (on average): $O(\log n)$
- Total cost: $O(m n \log n) + O(n (\log n)^2)$

witten & eibe

23

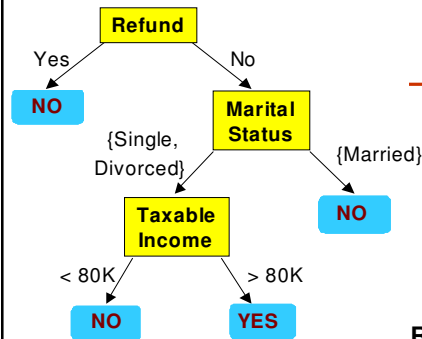
Decision Trees to Rules

- It is easy to derive a rule set from a decision tree: write a rule for each path in the decision tree from the root to a leaf.
- In that rule the left-hand side is easily built from the label of the nodes and the labels of the arcs.
- The resulting rules set can be simplified:
 - Let LHS be the left hand side of a rule.
 - Let LHS' be obtained from LHS by eliminating some conditions.
 - We can certainly replace LHS by LHS' in this rule if the subsets of the training set that satisfy respectively LHS and LHS' are equal.
 - A rule may be eliminated by using meta-conditions such as "if no other rule applies".

24

24

From Decision Trees To Rules



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single, Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single, Divorced}, Taxable Income>80K) ==> Yes

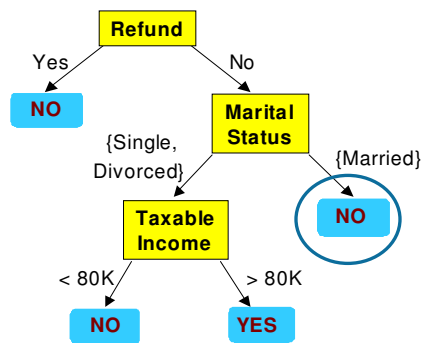
(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive
Rule set contains as much information as the tree

25

25

Rules Can Be Simplified



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Initial Rule: (Refund=No) \wedge (Status=Married) \rightarrow No

Simplified Rule: (Status=Married) \rightarrow No

26

26

Evaluation Methodology

- Standard methodology:
 1. Collect a large set of examples (all with correct classifications!).
 2. Randomly divide collection into two disjoint sets: training and test.
 3. Apply learning algorithm to training set giving hypothesis H
 4. Measure performance of H w.r.t. test set ([cross-validation](#))
- Important: keep the training and test sets disjoint!
- To study the efficiency and robustness of an algorithm, repeat steps 2-4 for different training sets and sizes of training sets.
- If you improve your algorithm, start again with step 1 to avoid evolving the algorithm to work well on just this collection.

27

27

Ensemble Learning

- So far we only focused on learning one hypotheses for making a prediction.
- Depending on the true hypotheses it might get very difficult.
- Ensemble learning takes several hypotheses and combines them appropriately (generate 100 Decision Trees from the samples).

28

AdaBoost

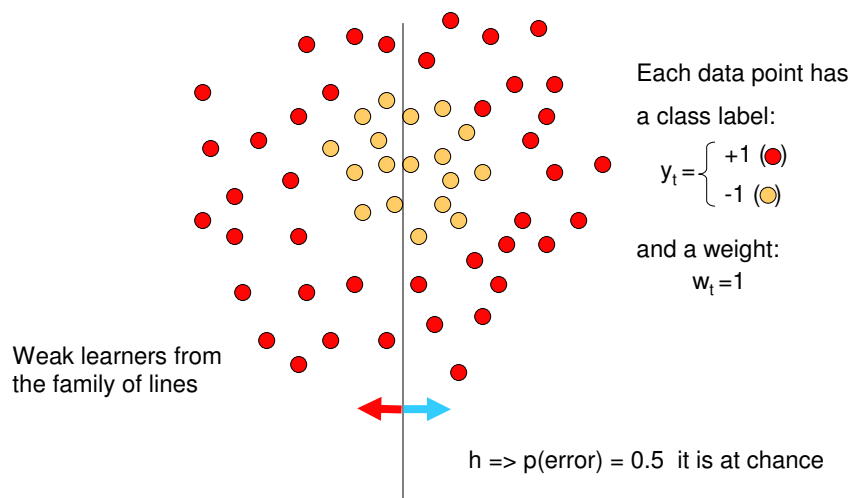
- Idea:
 - Complex hypothesis are hard to design without overfitting
 - Simple hypothesis cannot explain all data points
 - Combine many simple hypothesis into a complex one
- Issues:
 - How do we generate simple hypotheses?
 - How do we combine them?

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

↑ ↑ ↑ ↑
Strong classifier Features vector Weight Weak classifier

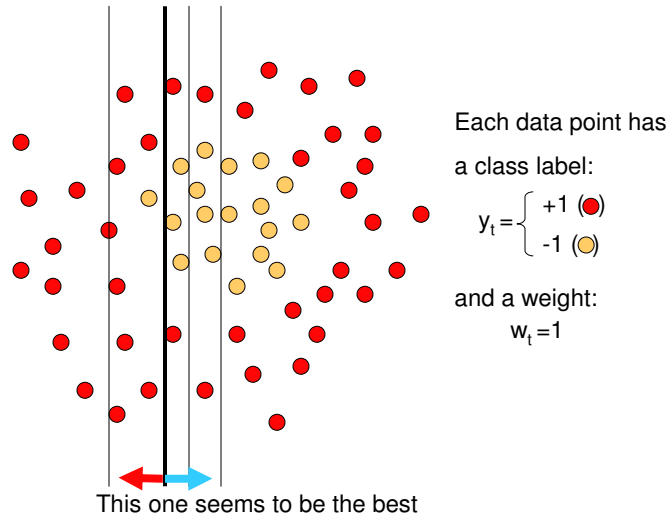
29

Toy Example — taken from Torralba @MIT



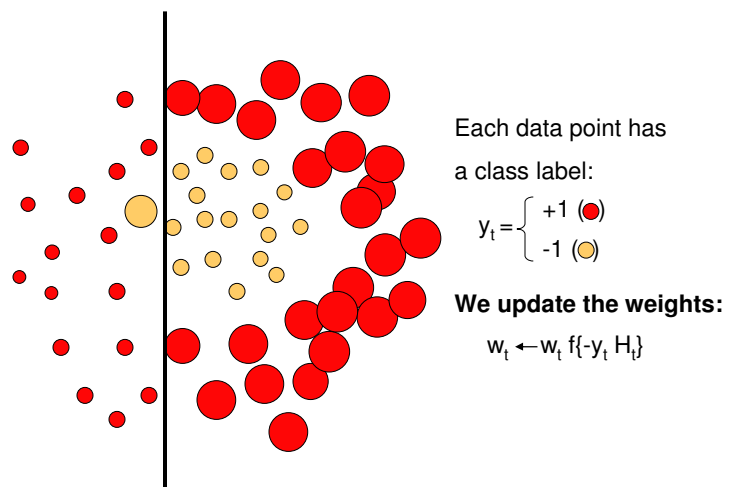
30

Toy example



This is a '**weak classifier**': It performs slightly better than chance.

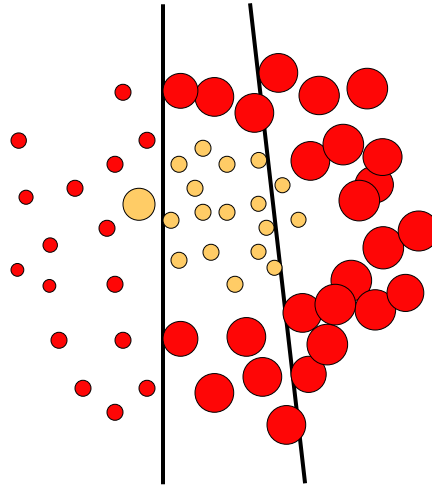
Toy example



We set a new problem for which the previous weak classifier performs at chance again

32

Toy example



Each data point has
a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

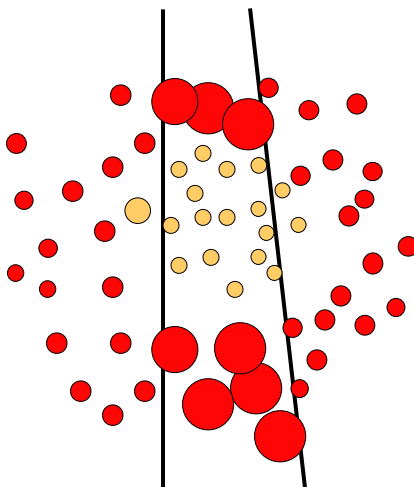
We update the weights:

$$w_t \leftarrow w_t f\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

33

Toy example



Each data point has
a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

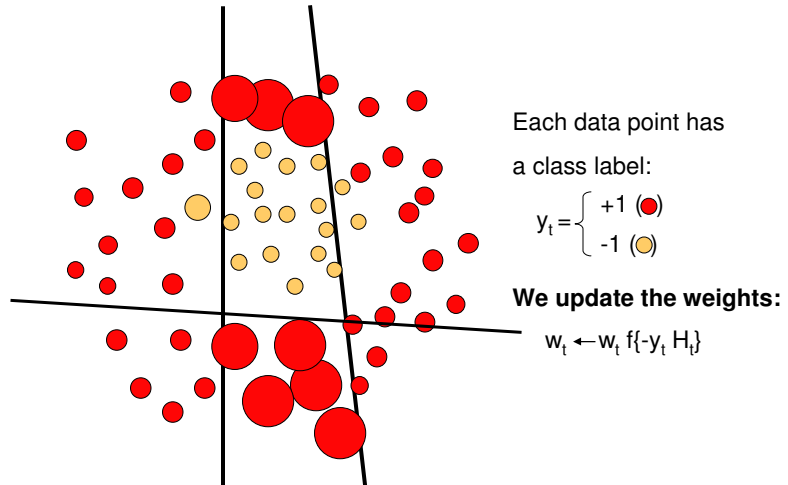
We update the weights:

$$w_t \leftarrow w_t f\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

34

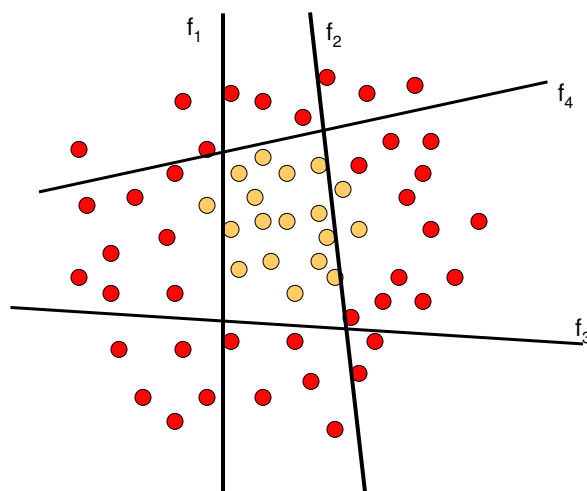
Toy example



We set a new problem for which the previous weak classifier performs at chance again

35

Toy example



The strong (non-linear) classifier is built as the combination of all the weak (linear) classifiers.

Formal Procedure of AdaBoost

- given training set $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
 - construct distribution D_t on $\{1, \dots, m\}$
 - find weak classifier (“rule of thumb”)

$$h_t : X \rightarrow \{-1, +1\}$$
 with small error ϵ_t on D_t :

$$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

37

function ADABOOST(*examples*, L , M) **returns** a weighted-majority hypothesis
inputs: *examples*, set of N labelled examples $(x_1, y_1), \dots, (x_N, y_N)$
 L , a learning algorithm
 M , the number of hypotheses in the ensemble
local variables: \mathbf{w} , a vector of N example weights, initially $1/N$
 \mathbf{h} , a vector of M hypotheses
 \mathbf{z} , a vector of M hypothesis weights

```

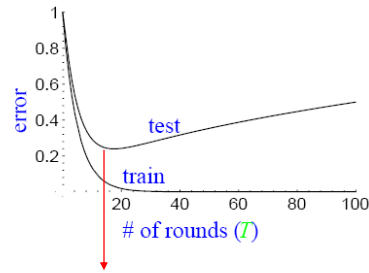
for  $m = 1$  to  $M$  do
   $\mathbf{h}[m] \leftarrow L(\text{examples}, \mathbf{w})$ 
   $\text{error} \leftarrow 0$ 
  for  $j = 1$  to  $N$  do
    if  $\mathbf{h}[m](x_j) \neq y_j$  then  $\text{error} \leftarrow \text{error} + \mathbf{w}[j]$ 
  for  $j = 1$  to  $N$  do
    if  $\mathbf{h}[m](x_j) = y_j$  then  $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot \text{error}/(1 - \text{error})$ 
   $\mathbf{w} \leftarrow \text{NORMALIZE}(\mathbf{w})$ 
   $\mathbf{z}[m] \leftarrow \frac{1}{2} \ln((1 - \text{error})/\text{error})$ 
return WEIGHTED-MAJORITY( $\mathbf{h}$ ,  $\mathbf{z}$ )
  
```

Figure 18.10 The ADABOOST variant of the boosting method for ensemble learning. The algorithm generates hypotheses by successively reweighting the training examples. The function WEIGHTED-MAJORITY generates a hypothesis that returns the output value with the highest vote from the hypotheses in \mathbf{h} , with votes weighted by \mathbf{z} .

38

But we are NOT interested in Training set

- Will Adaboost screw up with a fat complex classifier finally?



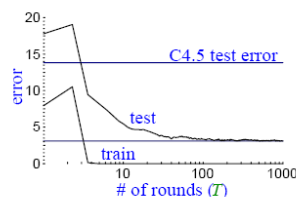
Occam's razor – simple is the best

Over fitting

Shall we stop before over fitting? If only over fitting happens.

39

Actual Typical Run



(boosting C4,5 on "letter" dataset)

- test error does not increase, even after 1000 rounds
 - (total size > 2,000,000 nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

40