RN, Chapter 18 – 18.3

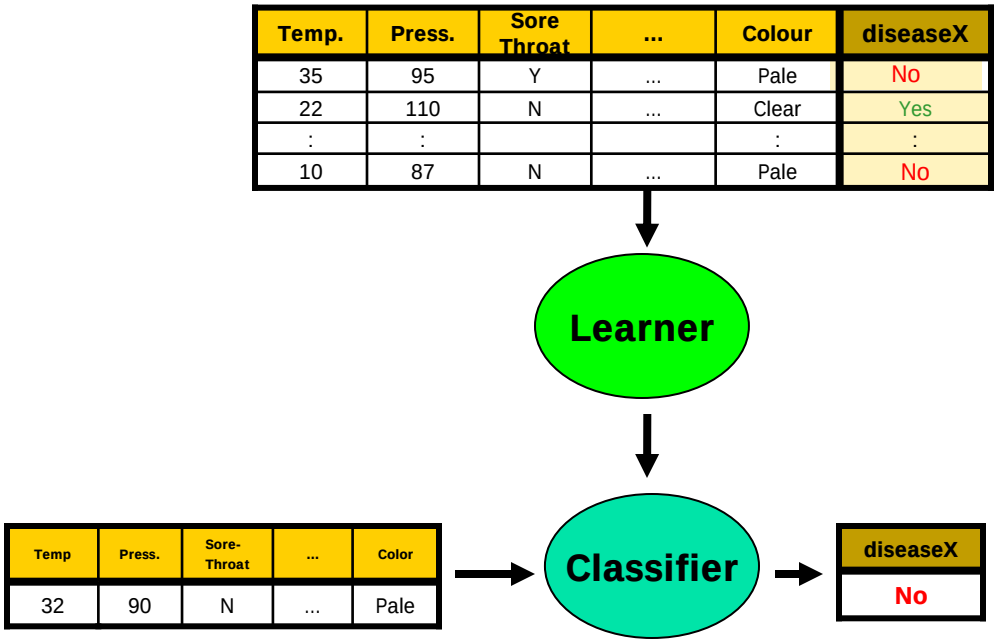
Learning from Observations

Learning Decision Trees

- Framework
  - Classification Learning
  - Bias
  - Def'n: Decision Trees
- Algorithm for Learning Decision Trees
  - Entropy, Inductive Bias (Occam's Razor)
- Evaluating Classifier
  - CrossValidation
- Overfitting
  - Def'n, MDL, χ^2, PostPruning
- Topics:
  - k-ary attribute values
  - Real attribute values
  - Other splitting criteria
  - Attribute Cost
  - Missing Values
  - ...

Inductive Inference

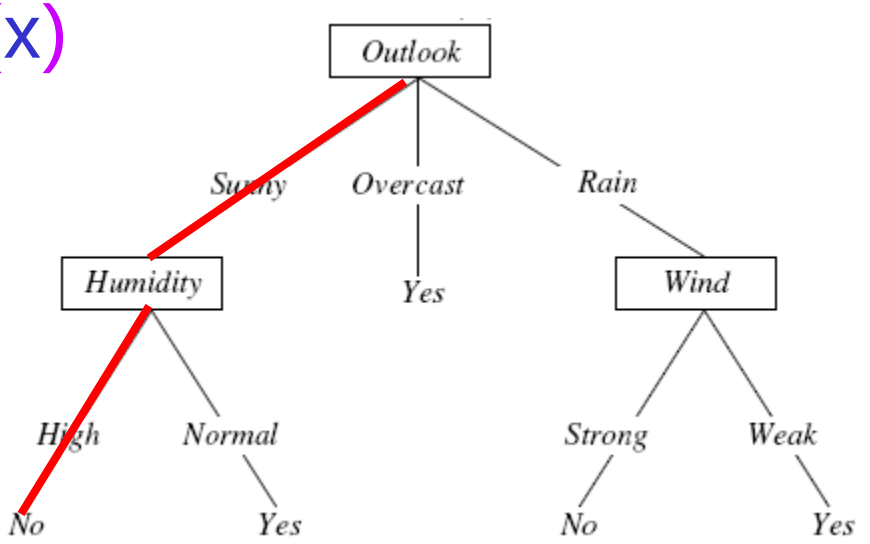
- Learning Element
- Performance Element



DecisionTree Hypothesis Space

- Internal nodes
  - test value of features x_j
  - branch according to results of test
- Leaf nodes specify class h(x)
- Instance:

Outlook	= Sunny
Temperature	= Hot
Humidity	= High
Wind	= Strong



  classified as "No"
  - (Temperature, Wind: irrelevant)
- Easy to use in Classification
  - Divide-and-Conquer
  - . . . keep splitting based on some test

Training Examples

Day	Outlook	Temp.	Humidity	Wind	P.Tennis
d_1	Sunny	Hot	High	Weak	No
d_2	Sunny	Hot	High	Strong	No
d_3	Overcast	Hot	High	Weak	Yes
d_4	Rain	Mild	High	Weak	Yes
d_5	Rain	Cool	Normal	Weak	Yes
d_6	Rain	Cool	Normal	Strong	No
d_7	Overcast	Cool	Normal	Strong	Yes
d_8	Sunny	Mild	High	Weak	No
d_9	Sunny	Cool	Normal	Weak	Yes
d_{10}	Rain	Mild	Normal	Weak	Yes
d_{11}	Sunny	Mild	Normal	Strong	Yes
d_{12}	Overcast	Mild	High	Strong	Yes
d_{13}	Overcast	Hot	Normal	Weak	Yes
d_{14}	Rain	Mild	High	Strong	No

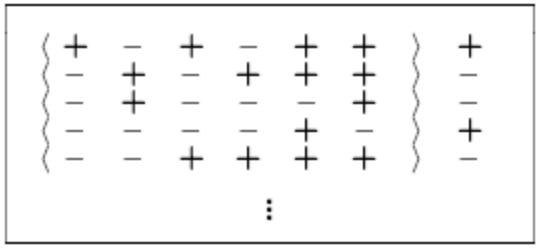
- 4 discrete-valued attributes
- "Yes/No" classification

Want: Decision Tree

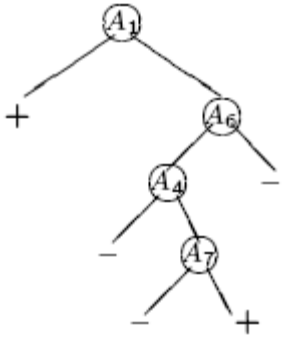
DT_{PT} (*Out, Temp, Humid, Wind*) $\in$ { Yes, No }

Learning Decision Trees – Easy?

- Learn: Data $\mapsto$ DecisionTree



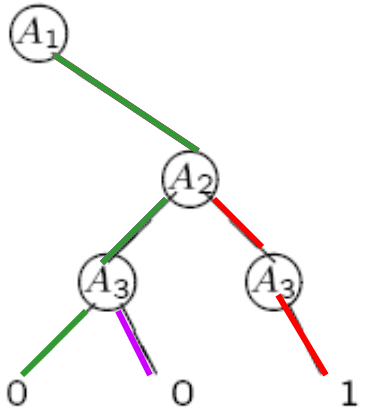
$\Rightarrow$



- Option 1: Just store training data

$A1$	$A2$	$A3$	$LABEL$
+	+	+	1
+	−	+	0
+	−	−	0

$\Rightarrow$



- But …

Learn ?Any? Decision Tree

- Just produce "path" for each example
  - May produce large tree
  - Any generalization? (what of other instances?)
    $\langle -, + + \rangle$?
  - Noise in data

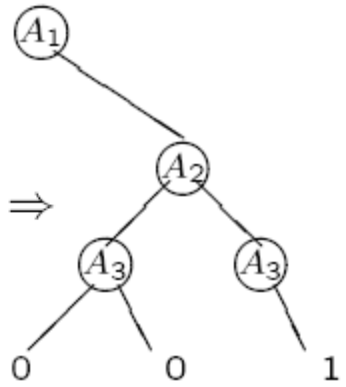
    $\langle \langle +, -, - \text{ i}, 0 \rangle$

    mis-recorded as

    $\langle \langle +, \mathbf{+}, - \rangle, 0 \rangle$

    $\langle \langle +, -, - \rangle, \mathbf{1} \rangle$

A1	A2	A3	LABEL
+	+	+	1
+	−	+	0
+	−	−	0



- Intuition:

  Want SMALL tree

  ... to capture "regularities" in data ...

  ... easier to understand, faster to execute, ...

(Simplified) Algorithm for Learning Decision Tree

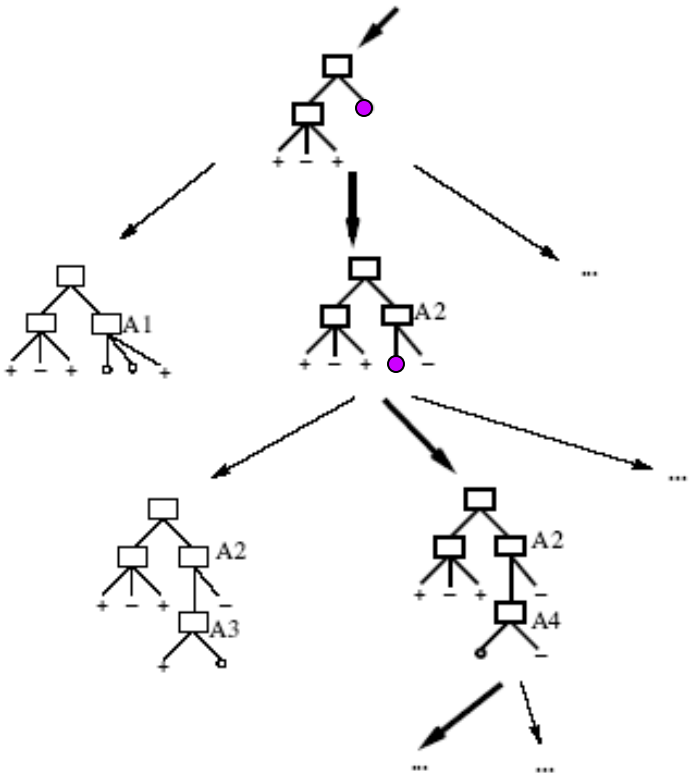
```
GrowTree(S)
    if (y = 0) for all ⟨x, y⟩ ∈ S return new leaf(0)
    if (y = 1) for all ⟨x, y⟩ ∈ S return new leaf(1)
    /* else */

    x_j = ChooseBestAttribute(S)
    S_0 = all ⟨x, y⟩ ∈ S with x_j = 0
    S_1 = all ⟨x, y⟩ ∈ S with x_j = 1
    return new node(x_j, GrowTree(S_0), GrowTree(S_1) )
```

- Many fields independently discovered this learning alg...
- Issues
  - no more attributes
  - > 2 labels
  - continuous values
  - oblique splits
  - pruning

Search for Good Decision Tree

- Local search
  - expanding one leaf at-a-time
  - no backtracking
- Trivial to find tree that perfectly "fits" training data*
- but… this is NOT necessarily best tree
  - NP-hard to find smallest tree that fits data



* noise-free data

Issues in Design of Decision Tree Learner

- What attribute to split on?
- When to stop?
- Should tree by pruned?
- How to evaluate classifier (decision tree) ? ... learner?

Desired Properties

- Score for split M(D, x_i) related to

$$S\begin{pmatrix} \#(x_i = t, Y = +) \\ \#(x_i = t, Y = -) \end{pmatrix} \qquad S\begin{pmatrix} \#(x_i = f, Y = +) \\ \#(x_i = f, Y = -) \end{pmatrix}$$

- Score S(.) should be
  - Score is BEST for [+0, –200]
  - Score is WORST for [+100, –100]
  - Score is “symmetric"

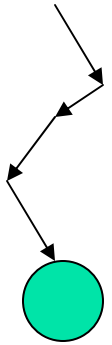
    Same for [+19, –5] and [+5, –19]
  - Deals with any number of values

v_1	7
v_2	19
:	:
v_k	2

Prefer Low Entropy Leaves

- Use decision tree h(.) to classify (unlabeled) test example x

  … Follow path down to leaf r
  … What classification?



- Consider training examples that reached r:
  - If all have same class c
    ⇒ label x as c (ie, entropy is 0)

    +200, – 0
  - If ½ are +; ½ are –
    ⇒ label x as ??? (ie, entropy is 1)

    +100, – 100
- On reaching leaf r with entropy H_r, uncertainty w/label is H_r

  (ie, need H_r more bits to decide on class)

  ⇒ prefer leaf with LOW entropy

Remaining Uncertainty

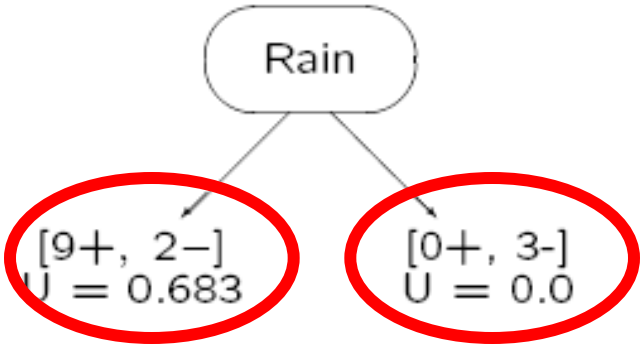
$Uncert(S, A)$ = remaining expected entropy after splitting on A

$$\equiv \sum_{v_i \in Values(A)} \frac{|S_{v_i}|}{|S|} \text{Entropy}(S_{v_i})$$

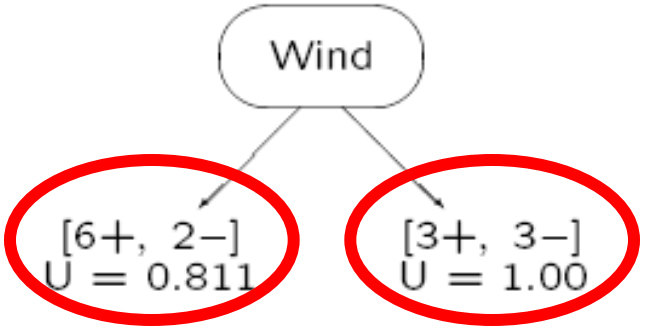
$$\equiv \sum_{i=1}^{v} \frac{p_i^{(A)} + n_i^{(A)}}{p+n} H\left(\frac{p_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}, \frac{n_i^{(A)}}{p_i^{(A)} + n_i^{(A)}}\right)$$

Day	Rain	Temp.	Humidity	Wind	P.Tennis
d_1	No	Hot	High	Weak	No
d_2	No	Hot	High	Strong	No
d_3	No	Hot	High	Weak	Yes
d_4	No	Mild	High	Weak	Yes
d_5	No	Cool	Normal	Weak	Yes
d_6	Yes	Cool	Normal	Strong	No
d_7	No	Cool	Normal	Strong	Yes
d_8	Yes	Mild	High	Weak	No
d_9	No	Cool	Normal	Weak	Yes
d_{10}	No	Mild	Normal	Weak	Yes
d_{11}	No	Mild	Normal	Strong	Yes
d_{12}	No	Mild	High	Strong	Yes
d_{13}	No	Hot	Normal	Weak	Yes
d_{14}	Yes	Mild	High	Strong	No

S : [9+, 5−]



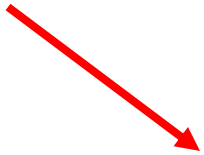
S : [9+, 5−]



Uncert(S, Rain)
$= \frac{11}{14}\ 0.683 + \frac{3}{14}\ 0.0$
$= 0.536$

Uncert(S, Wind)
$= \frac{8}{14}\ 0.811 + \frac{6}{14}\ 1.00$
$= 0.892$

Learning Decision Trees



- Framework
- Algorithm for Learning Decision Trees
- Evaluating Classifier
  - CrossValidation
- Overfitting
- Topics:

Quality of Learned Classifier h?

Sample error of h wrt

labeled data sample S = { $\langle x_i, f(x_i) \rangle$ }

≡ proportion of examples h misclassifies

$$\widehat{err}_S(h) \equiv \frac{1}{n} \sum_{\langle x, f(x) \rangle \in S} \mathcal{I}\{ f(x) \neq h(x) \}$$

where $\mathcal{I}\{ a \neq b \} = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{otherwise} \end{cases}$

- EASY to compute. . .
- But... what sample S?
  - Perhaps the one used for TRAINING?
  - TRIVIAL to get 0% error: JUST REUSE DATA !
- Goal is determining

  quality of response on NEW (unseen) DATA!

True Error

True error of hypothesis h wrt

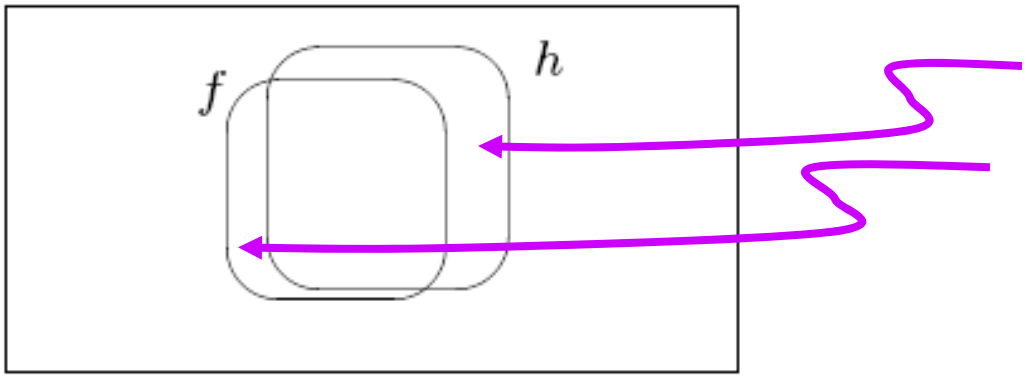
- target function f
- distribution D

≡ probability that h will misclassify (wrt f) an instance drawn from D

- $err_{D;f}(h) \equiv Pr_{x \in D}[f(x) \neq h(x)]$ $= \Sigma \, Pr(x) * \mathcal{I}[f(x) \neq h(x)]$

  Expected Performance

  over entire distribution of instances



Comparing $\underline{err}_S(h)$ vs $err_{D;f}(h)$

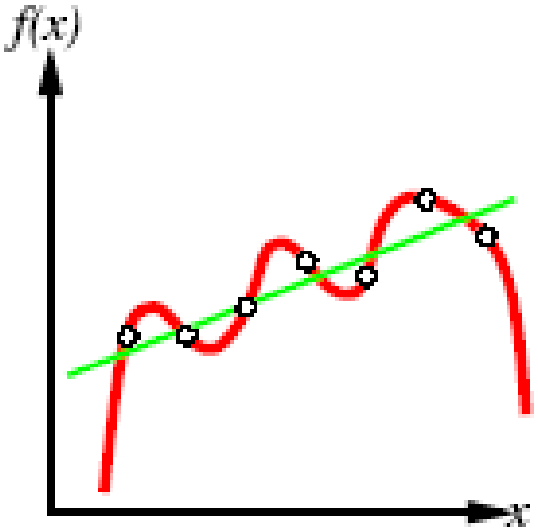
- Need $err_{D;f}(h)$
  h's Generalization Error over distribution D
  - to evaluate classifier
  - to decide among possible classifiers
  - to evaluate learner
- But depends on D, f: not known!
- Q: Why not use $\underline{err}_S(h_S)$?
- Good news
  - Easy to compute
  - If different S', then $\underline{err}_{S'}(h_S) \approx err_{D;f}(h)$
- Bad news. . .

But ...

- $\underline{err}_S(h_S) \neq err_{D;f}(h_S)$

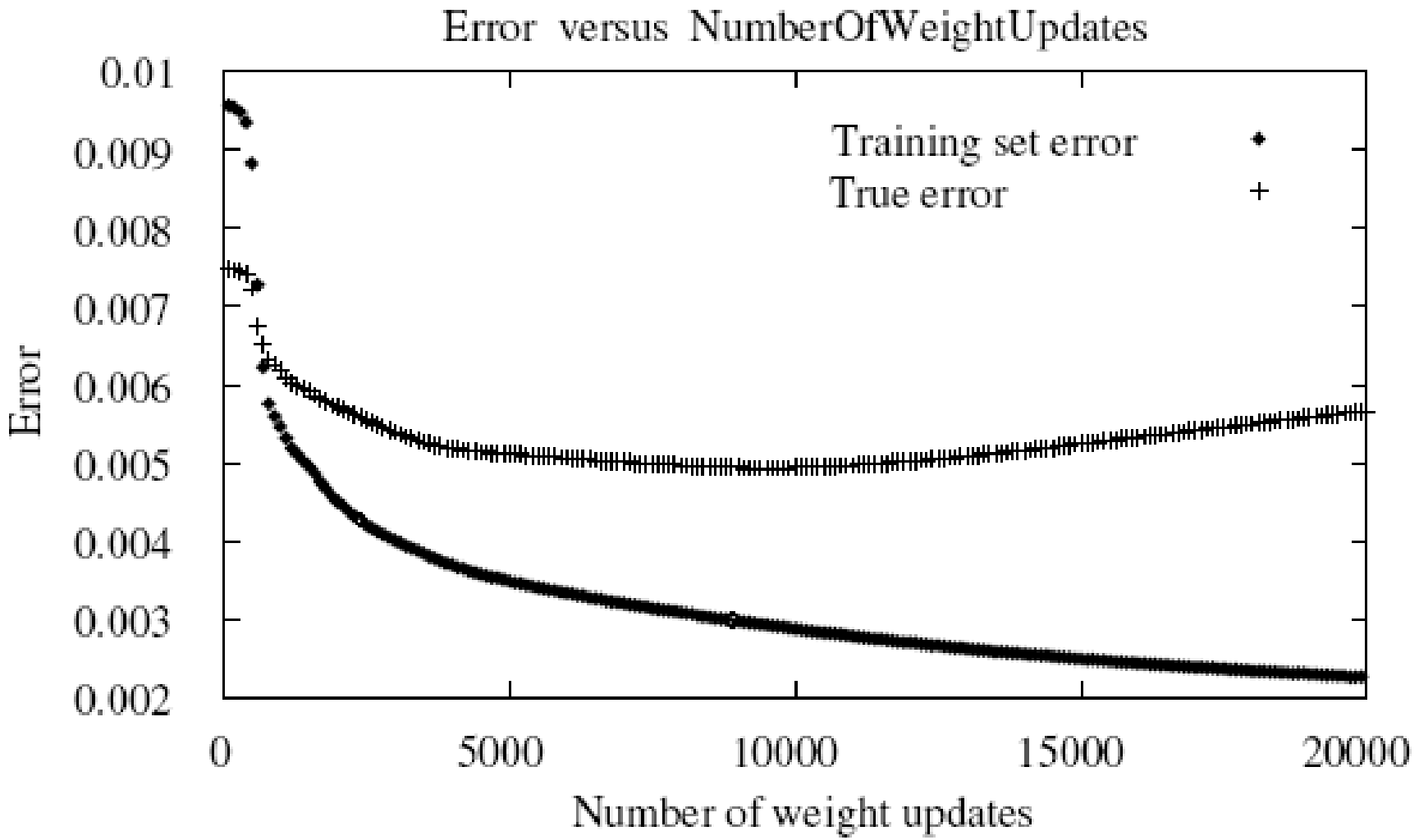
- $\underline{err}_S(h_S) \equiv$
  Eval h_S on training set S
  - only approximation to $err_{D;f}(h_S)$
  - ... can be TOO optimistic!

- "Cheating"
  Like being evaluated on test
  after seeing SAME test. . .



$\underline{err}_S(h) = 0$
$\underline{err}S(h) > 0$

$\underline{err}_S(h_S)$ vs $err_{D;f}(h_S)$



Estimating Error

- SetUp: Learner L,
  - using labeled training data S
  - produces classifier $h_S = L(S)$
- Estimating $err_{D;f}(h_S)$

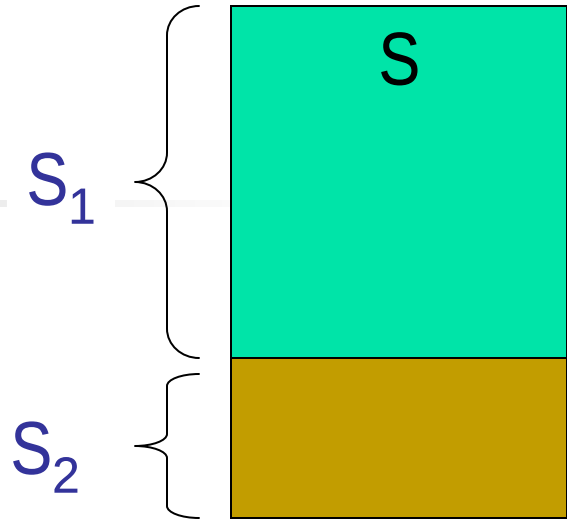
  $\equiv$ (true) error of h_S over distribution D

- **Option1:** Use h_S's empirical error on S

  $\underline{err}_S(h_S)$ (Resubstitution Error)

  $\Rightarrow$ Very Optimistic

Estimating Error: Hold-Out Set



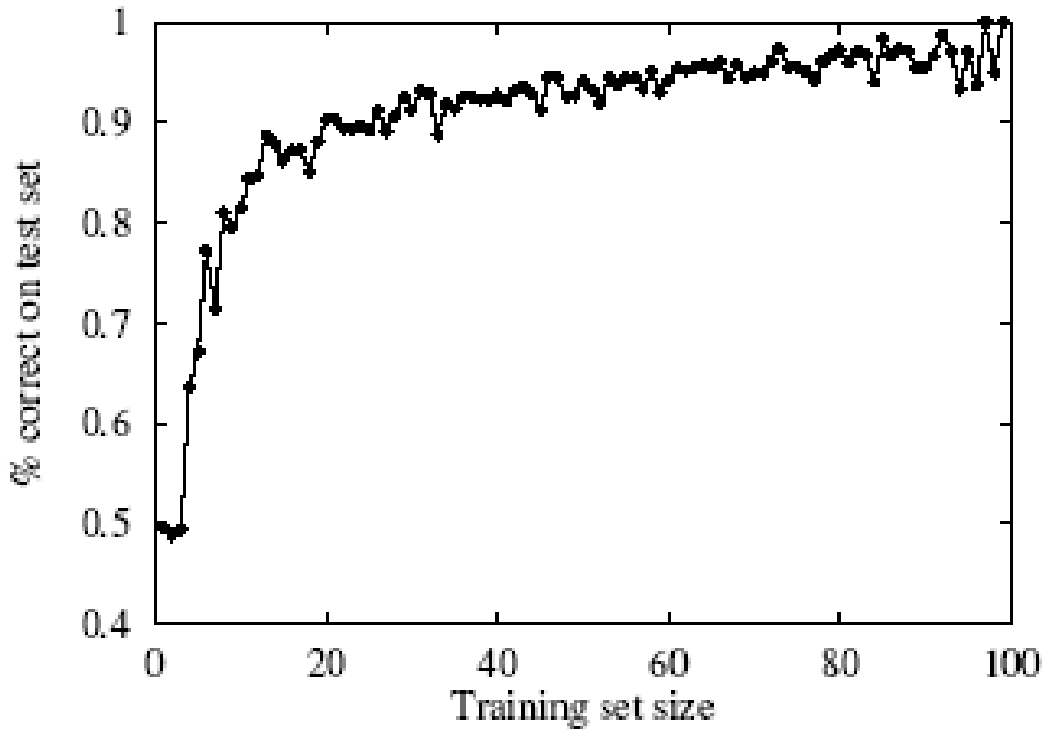
- **Option2**: Divide S into disjoint S_1, S_2
  - Train on S_1: computing $h_{S_1} = L(S_1)$
  - Test on S_2: $\underline{err}_{S_2}(h_{S_1})$
- As $S_1 \approx S$, $L(S_1) \approx L(S)$
  But $L(S_1)$ not as good as $L(S)$
  (Learning curve: $L(S)$ improves as $|S|$ increases)
  $\Rightarrow$ want S_1 to be as large as possible
- $\underline{err}_{S_2}(h_{S_1})$ is estimate of $err_{D;f}(h_S)$
  This estimate improves as S_2 gets larger
  $\Rightarrow$ want S_2 to be as large as possible
- As $S = S_1 \cup S_2$, must trade off
  quality of classifier $h_{S_1} = L(S_1)$
  with
  accuracy of estimate $\underline{err}_{S_2}(h_{S_1})$

More Data ⇒ Better Accuracy

- w/high prob

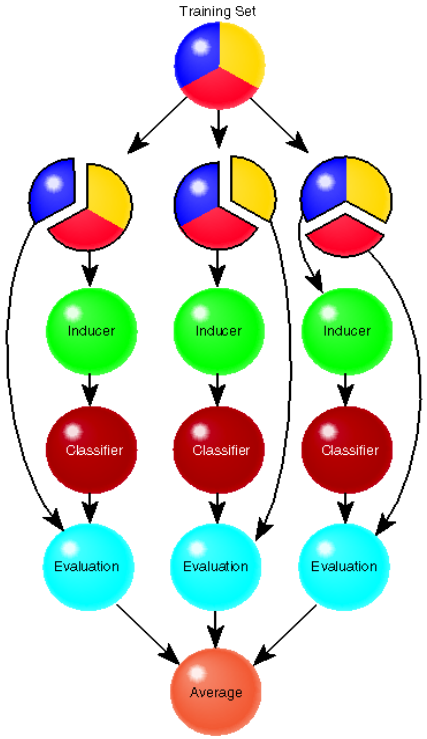
$$| \, err_S(h) - err_{D;f}(h) | \approx \alpha / \sqrt{(|S|)}$$

- Learning curve. . .



Estimating Error: Cross Validation

- **Option3:** "Cross-Validation"



CV(data S, alg L, int k)
  Divide S into k disjoint sets { S_1, S_2, …, S_k }
  For i = 1..k do
    Run L on $S_{-1} = S - S_i$
    obtain $L(S_{-i}) = h_i$
    Evaluate h_i on S_i
      $err_{S_i}(h_i) = 1/|S_i| \sum_{\langle x,y \rangle \in S_i} I(h_i(x) \neq y)$
  Return Average $1/k \sum_i err_{S_i}(h_i)$

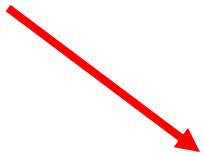
$\Rightarrow$ Less Pessimistic

as train on $(k - 1)/k$ $|S|$ of the data

Comments on Cross-Validation

- Every point used as
  Test 1 time, Training k -1 times
- Computational cost for *k*-fold Cross-validation ... linear in *k*
- Should use “balanced CV"
  If class c_i appears in m_i instances,
  insist each S_k include $\approx(k-1)/k\ m_i/|S|$ such instances
- Use **CV(S, L, k)** as ESTIMATE of true error of L(S)
  Return L(S) and CV(S, L, k)
- Leave-One-Out-Cross-Validation k = m !
  - eg, for Nearest-Neighbor
- Notice different folds are correlated
  as training sets overlap: (k-2)/k unless k=2
- 5 × 2-CV
  - Run 2-fold CV, 5 times. . .

Learning Decision Trees



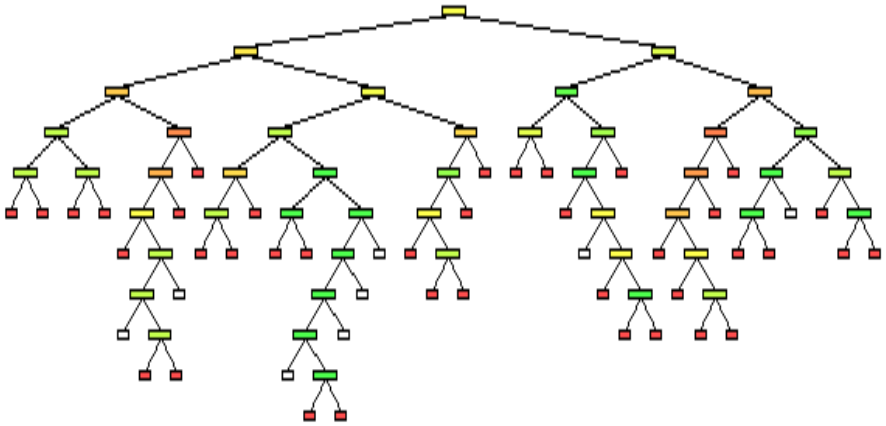
- Framework
- Algorithm for Learning Decision Trees
- Evaluating Classifier
- Overfitting
  - Def'n
  - MDL, χ^2
  - PostPruning
- Topics:

Example of Overfitting

- 25% have butterfly-itis
- ½ of patients have $F_1 = 1$
  - Eg: “odd birthday”
- ½ of patients have $F_2 = 1$
  - Eg: “even SSN”
- ... for 10 features
- Decision Tree results
  - over 1000 patients (using these silly features) ...

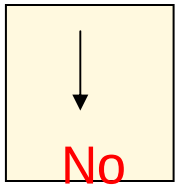
Decision Tree Results

- Standard decision tree learner:



- Error Rate:
  - Train data: 0%
  - New data: 37%

- Optimal decision tree:



- Error Rate:
  - Train data: 25%
  - New data: 25%

Overfitting

- Often "meaningless regularity" in data
  due to coincidences in the noise
  $\Rightarrow$ bad generalization behavior
  "Overfitting"
- Consider error in hypothesis h over ...
  - training data S: $err_S(h)$
  - entire distribution D of data: $err_{D,f}(h)$
- Hypothesis $h \in H$ **overfits** training data if
  $\exists$ alternative hypothesis $h' \in H$ s.t.
  $err_S(h) < err_S(h')$
  but
  $err_{D,f}(h) > err_{D,f}(h')$

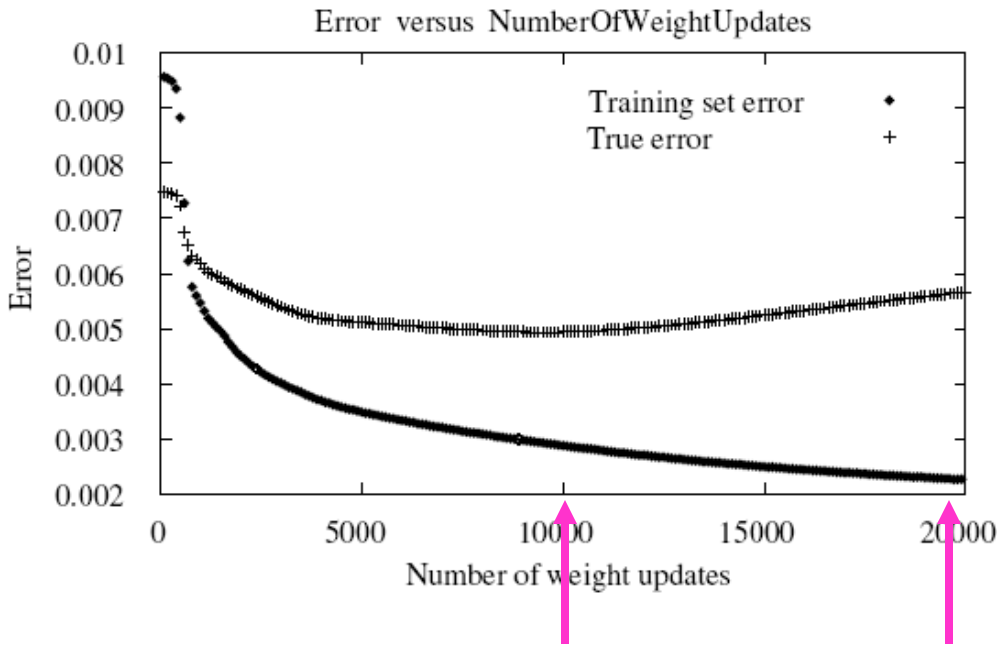
Fit-to-Data ≠ Generalization

- h_k = hyp after k updates

  $err_S(h_{20000}) < err_S(h_{10000})$

  but

  $err_{D,f}(h_{20000}) > err_{D,f}(h_{10000})$



- "Overfitting"

  Best "fit-to-data" will often find meaningless regularity in data

  (coincidences in the noise)

  $\Rightarrow$ bad generalization behavior

Example of Overfitting

- Spse 10 binary attributes (uniform),

  but class is random: { N w/prob p = 0.75 ; Y w/prob 1–p = 0.25 }

- C4.5 builds nonsensical tree w/ 119 nodes!
  ⇒ Should be SINGLE NODE!
- Error rate (hold-out): 35%
  ⇒ Should be 25% (just say "No")
- Why? Tree assigns leaf "N" w/prob 1–p, "Y" w/prob p
  - Tree sends instances to arbitrary leaves
  - Mistake if
    - Y-instance reaches N-leaf: p x (1–p)
    - N-instance reaches Y-leaf: (1–p) x p
  - Total prob of mistake = 2 x p x (1–p) = 0.375

- Overfitting happens for EVERY learner ... not just DecTree !!

How to Avoid Overfitting (Decision Trees)

- When to act
  - Use more stringent STOPPING criterion while growing tree

    . . . only allow statistically significant splits ...
  - Grow full tree, then post-prune
- To evaluate tree, measure performance over ...
  - training data
  - separate validation data set
- How to represent classifier?
  - as Decision Tree
  - as Rule Set

Avoid Overfitting #1

(StopEARLY, Training-Data, DecTree)

- Add more stringent STOPPING criterion while growing tree
  - At leaf n_r (w/ instances S_r)
    spse optimal split is based on attribute A
  - Apply statistical test to compare
    - T_0: leaf at r (majority label) vs
    - T_A: split using A
  - Is error of T_A statistically better than T_0?
  - (Note: resubstitution error of $T_A \leq T_0$)

A. Use χ^2 **test**, on data S_r

B. **MDL**: minimize

size(tree) + size(misclassifications(tree))

χ^2 Test for Significance

- Null hypothesis $\mathbf{H_0}$:
  Attribute A is irrelevant in context of r
  Ie, distr of class labels at node $n_r \equiv$ distr after splitting on A
- Observe some difference between these distr's.
  What is prob (under H_0) of observing this difference, given m = $|S_r|$ iid samples?
- Defn: { p / n } of S_r are { positive / negative }

  After splitting on A, get *k* subsets
  wrt *A = i* : p_i positives, n_i negatives
- If $\mathbf{H_0}$ (A irrelevant), would have
  - $p_i^{\sim} = p \times (p_i+n_i)/p+n$ positives
  - $n_i^{\sim} = n \times (p_i+n_i)/p+n$ negatives

χ^2 Test – con't

- If H_0 (A irrelevant), would have
  $\hat{p}_i = p \times \frac{p_i+n_i}{p+n}$ positives
  $\hat{n}_i = n \times \frac{p_i+n_i}{p+n}$ negatives

- Measure of deviation:

$$D = \sum_{i=1}^{k} \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

Under H_0, $D \sim \chi^2_{(k-1)(2-1)}$
(k = size of A's domain; 2 = # classes)

Q? Is D value within tolerences?

Don't add iff $D < T_{\alpha,d}$

$T_{\alpha,d}$ is threshold:
α = prob of making mistake
(rejecting null hyp, when A is irrelevant)
d = degree of freedom

- Obvious extension when > 2 classes

$\sum$ (exp – obs)^2 / exp

Results: Empirically, not reliable. . .

χ^2 Table

For χ^2 test, with

$\alpha = 0.05$

various "degrees of freedom"

DOF	$T_{\alpha,DOF}$	DOF	$T_{\alpha,DOF}$
1	3.841	16	26.296
2	5.991	17	27.587
3	7.815	18	28.869
4	9.488	19	30.144
5	11.070	20	31.410
6	12.592	21	32.671
7	14.067	22	33.924
8	15.507	23	35.172
9	16.919	24	36.415
10	18.307	25	37.652
11	19.675	26	38.885
12	21.026	27	40.113
13	22.362	28	41.337
14	23.685	29	42.557
15	24.996	30	43.773

Minimum Description Length

A wants to transmit to B classification function c()

- simplified to:
  - A and B agree on instances $\langle x_1, \ldots, x_M \rangle$
  - What should A send, to allow B to determine M bits:

    $\langle c(x_1), \ldots, c(x_M) \rangle$
- Option#1: A can send M "bits"
- Option#2: A sends "perfect" decision tree d

  s.t. $c(x_i) = d(x_i)$ for each x_i
- Option#3: A sends "imperfect" decision tree d'

  + set of indices of K exceptions B = $\{ x_{i1}, \ldots, x_{iK} \}$

$$c(x_i) = \begin{cases} \neg d(x_i) & \text{if } x_i \in B \\ d(x_i) & \text{otherwise} \end{cases}$$

- So... Increase tree-size

  IF (significant) reduction in #exceptions

Avoid overfitting#2: PostPruning

- Grow tree, to “purity”, then PRUNE it back!

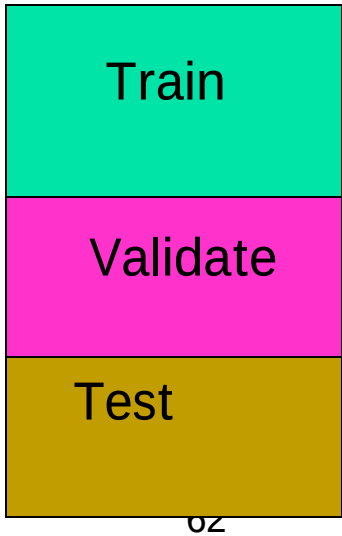
Build “complete" decision tree h, from train
For each penultimate node: n_i
  Let h_i be tree formed by “collapsing” subtree under n_i, into single node
  If h_i better than h
    Reset $h \leftarrow h_i$, . . .

- How to decide if h_i better than h?

1. Test on Hold-Out data?
   - 3 sets: training, *VALIDATION*, testing

   Problematic if small total # of samples
2. Pessimistic Pruning

   . . . re-use training samples . . .

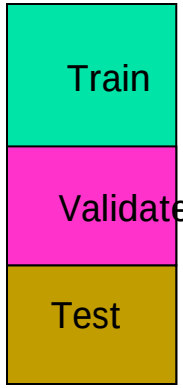


Avoid Overfitting#2.1 “Reduced-Error Pruning"

- Split data into *training* and *validation* set

Alg: Do until further pruning is harmful:

1. Evaluate impact on *validation* set...
   of pruning each possible node (plus those below it)
2. Greedily remove the node that most improves accuracy on *validation* set



- Produces small version of accurate subtree
- What if data is limited?

Avoid Overfitting#2.2 "Pessimistic Pruning"

- Assume N training samples reach leaf r; ... which makes E mistakes
  so (resubstitution) error is E/N

- For confidence level (eg, 1-sided $\alpha = 0.25$),
  can estimate upper bound on # of errors:
  Number Of Errors = $N \times [E/N \pm EB_{\alpha}(N,E)]$

- Let $U_{\alpha}(N,E) = E/N + EB_{\alpha}(N, E)$
  $EB_{\alpha}(E;N)$ based on binomial distribution

  ~ Normal distribution: $z_{\alpha} \times \sqrt{p(1-p)/N}$

  $p \approx E/N$

- Laplacian correction... to avoid "divide by 0" problems:

  Use $p = (E+1)/(N+2)$ not E/N

- For $\alpha = 0.25$, use $z_{0.25} = 1.53$
  (recall 1-sided)

Pessimistic Pruning (example)

- Eg, spse A has 3 values: { v_1 v_2 v_3 }
- If split on A, get
  - A= v_1 – return Y (6 cases, 0 errors)
  - A= v_2 – return Y (9 cases, 0 errors)
  - A= v_3 – return N (1 cases, 0 errors)

  So 0 errors if split on A

W/prob α,

$$\text{For } A = \left\{ \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} \right\}, \text{ error rate is under } \left\{ \begin{array}{c} U_\alpha(6,0) \\ U_\alpha(9,0) \\ U_\alpha(1,0) \end{array} \right\}$$

- For α=0.25:

  #errors $\leq 6 \times U_{0.25}(6,0) + 9 \times U_{0.25}(9,0) + 1 \times U_{0.25}(1,0)$
  $= 6 \times 0.206 + 9 \times 0.143 + 1 \times 0.75 = 3.273$
- If replace A-subtree w/ simple "Y"-leaf: (16 cases, 1 error)

  #errors $\leq 16 \times U_{0.25}(16,1) = 16 \times 0.1827 = 2.923$
- As 2.923 < 3.273, prune A-subtree to single "Y" leaf

  Then recur – going up to higher node

Pessimistic Pruning: Notes

- Results: Pruned trees tend to be
  - more accurate
  - smaller
  - easier to understand than original tree
- Notes:
  - Goal: to remove irrelevant attributes
  - Seems inefficient to grow subtree, only to remove it
  - This is VERY ad hoc, and WRONG statistically
    but works SO WELL in practice it seems essential
  - Resubstitution error will go UP, but generalization error, down...
  - Could replace n_i with single node,
    or with most-frequently used branch

Avoid Overfitting #3
Using Rule Post-Pruning

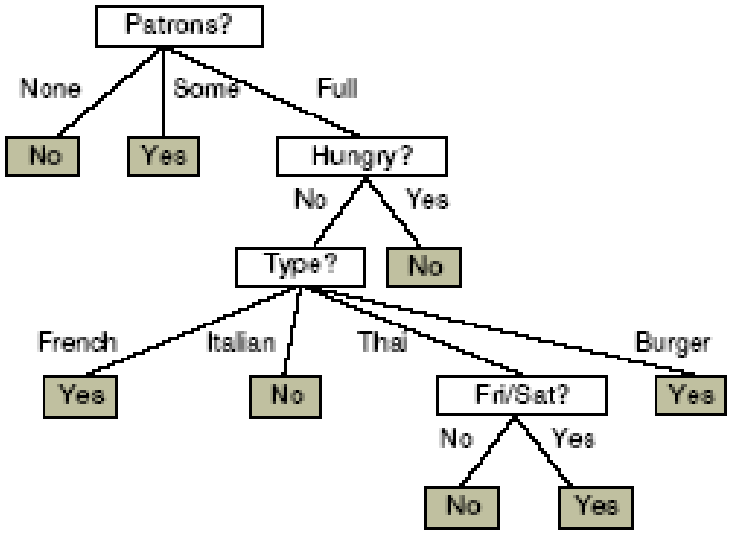
1. Grow decision tree.
   Fit data as well as possible.
   Allow overfitting.
2. Convert tree to equivalent set of rules:
   - One rule for each path from root to leaf.
3. Prune each rule independently of others.
   - ie, delete preconditions that improve its accuracy
4. Sort final rules into desired sequence for use depending on accuracy.

---

5. Use ordered sequence for classification.

Converting Trees to Rules

- Every decision tree corresponds to set of rules:
  - IF (Patrons = None)
    THEN WillWait = No
  - IF (Patrons = Full)
    & (Hungry = No)
    &(Type = French)
    THEN WillWait = Yes
  - ...



- Why? (Small) RuleSet MORE expressive
  small DecTree ≈ small RuleSet
  (DecTree is subclass of ORTHOGONAL DNF)

Learning Decision Trees

- Framework
- Algorithm for Learning Decision Trees
- Overfitting
- Topics:
  - k-ary attribute values
  - Real attribute values
  - Other splitting criteria
  - Attribute Cost
  - Missing Values
  - ...

Attributes with Many Values

- Problem: *Gain* prefers attribute with many values.
  Entropy ≈ ln(k) . . .
  Eg, imagine using
  Date = Jun 3 1996
  Name = Russ
- One approach: use *GainRatio* instead

$$\text{GainRatio}(S,A) = \frac{\text{Gain}(S,A)}{\text{SplitInformation}(S,A)}$$

$$\text{SplitInformation}(S,A) \equiv -\sum_{i=1}^{k} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

- Issues:
  - Construct a multiway split?
  - Test one value versus all of the others?
  - Group values into two disjoint subsets?

Continuous Valued Attributes

Create a discrete attribute to test continuous

- *Temperature* = 82.5
- (Temperature > 72.3) ∈ { t, f }

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- Note: need only consider splits between "class boundaries"

  Eg, between 48 / 60; 80 / 90

Finding Split for Real-Valued Features

- Best threshold θ_j for attribute j
  - Sort training instance by $x_{i,j}$
  - For each $\theta \in \Re$,

		where $< \theta$	where $\geq \theta$
labeled	A	$n_{A,\ell}(\theta)$	$n_{A,r}(\theta)$
	B	$n_{B,\ell}(\theta)$	$n_{B,r}(\theta)$
	(either)	$n_\ell(\theta)$	$n_r(\theta)$

y_i	A	A	B	A	B	B	A	B	B
$x_{i,j}$	0.2	0.4	0.7	1.1	1.3	1.7	1.9	2.4	2.9

$n_{A,\ell} = 3$	$n_{A,r} = 1$
$n_{B,\ell} = 1$	$n_{B,r} = 4$

Mutual information $= 0.2294$

  - Sequentially set θ to value of $\frac{x_{i,j}+x_{i,j+1}}{2}$ Compute mutual information

$$\frac{n_\ell(\theta)}{n_\ell(\theta)+n_r(\theta)}\left[\frac{n_{A,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)}\ln\frac{n_{A,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)} + \frac{n_{B,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)}\ln\frac{n_{B,\ell}(\theta)}{n_{A,\ell}(\theta)+n_{B,\ell}(\theta)}\right]$$
$$+$$
$$\frac{n_r(\theta)}{n_\ell(\theta)+n_r(\theta)}\left[\frac{n_{A,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)}\ln\frac{n_{A,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)} + \frac{n_{B,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)}\ln\frac{n_{B,r}(\theta)}{n_{A,r}(\theta)+n_{B,r}(\theta)}\right]$$

  - Just need to consider j s.t. $y_j \neq y_{j+1}$!

Desired Properties

Repeat!

- Score for split M(D, x_i) related to

$$S\begin{pmatrix} \#(x_i = t, Y = +) \\ \#(x_i = t, Y = -) \end{pmatrix} \qquad S\begin{pmatrix} \#(x_i = f, Y = +) \\ \#(x_i = f, Y = -) \end{pmatrix}$$

- Score S(.) should be
  - Score is BEST for [+0, –200]
  - Score is WORST for [+100, – 100]
  - Score is “symmetric"

    Same for [+19, – 5] and [+5, –19]
  - Deals with any number of values

v_1	7
v_2	19
:	:
v_k	2

Other Splitting Criteria

- Why use *Gain* as splitting criterion?

Want: Large "use me" value if split is

⟨ 85, 0, 0, ..., 0 ⟩

Small "avoid me" value if split is

⟨ 5, 5, 5, ..., 5 ⟩

- True of *Gain*, *GainRatio*... also for. . .
- Statistical tests: χ^2

For each attr A, compute deviation:

$$D(A) = \sum_{i=1}^{k} \frac{(p_i^{(A)} - \hat{p}_i^{(A)})^2}{\hat{p}_i^{(A)}} + \frac{(n_i^{(A)} - \hat{n}_i^{(A)})^2}{\hat{n}_i^{(A)}}$$

- GINI index: $GINI(A) = \sum_i \sum_{j \neq i} p_i\, p_j = 1 - \sum_i p_i^2$

- Others: “Marshall Correction”
  - “G” statistic
  - Probabilities (rather than statistic)

Example of χ^2

- Attributes T_1, T_2 class c

Data:

$T_1 =$	Yes	No	$c =$
Y	0	3	3
N	5	2	7
	5	5	10

$T_2 =$	Yes	No	$c =$
a	3	2	5
b	1	2	3
c	1	1	2
	5	5	10

χ^2: For T_1:

$$\frac{(0-1.5)^2}{1.5} + \frac{(3-1.5)^2}{1.5} + \frac{(2-3.5)^2}{3.5} + \frac{(5-3.5)^2}{3.5} = 4.29$$

For T_2:

$$\frac{(3.0-2.5)^2}{2.5} + \frac{(2-2.5)^2}{2.5} + \frac{(1-1.5)^2}{1.5} + \frac{(2-1.5)^2}{1.5} + \frac{(1-1)^2}{1} + \frac{(1-1)^2}{1} = 0.533$$

- As 4.29 (T_1) > 0.533 (T_2), ... use T_1 (less likely to be irrelevant)

Example of GINI

- Attributes T_1, T_2 class c

Data:

$T_1 =$	$c =$ Yes	No	
Y	0	3	3
N	5	2	7
	5	5	10

$T_2 =$	$c =$ Yes	No	
a	3	2	5
b	1	2	3
c	1	1	2
	5	5	10

Scores: $GINI(T_1) =$

$$\left[1 - \left(\tfrac{5}{10}\right)^2 - \left(\tfrac{5}{10}\right)^2\right] - \left[\tfrac{3}{10}\left(1 - \left(\tfrac{0}{3}\right)^2 - \left(\tfrac{3}{3}\right)^2\right) + \tfrac{7}{10}\left(1 - \left(\tfrac{5}{7}\right)^2 - \left(\tfrac{2}{7}\right)^2\right)\right]$$

$$= \tfrac{1}{10}\left(\left(\tfrac{0^2}{3} + \tfrac{3^2}{3} + \tfrac{5^2}{7} + \tfrac{2^2}{7}\right) - \left(\tfrac{5^2}{10} + \tfrac{5^2}{10}\right)\right)$$

$$= 0.21429$$

$$GINI(T_2) = \tfrac{1}{10}\left(\left(\tfrac{3^2}{5} + \tfrac{2^2}{5} + \tfrac{1^2}{3} + \tfrac{2^2}{3} + \tfrac{1^2}{2} + \tfrac{1^2}{2}\right) - \left(\tfrac{5^2}{10} + \tfrac{5^2}{10}\right)\right)$$

$$= 0.026667$$

- As $GINI(T_1) > GINI(T_2)$, split on T_1
- As "$\left(\tfrac{5^2}{10} + \tfrac{5^2}{10}\right)$" is subtracted from both, can just use first term. . . and ignore $\tfrac{1}{10}$ multiplier

Cost-Sensitive Classification ... Learning

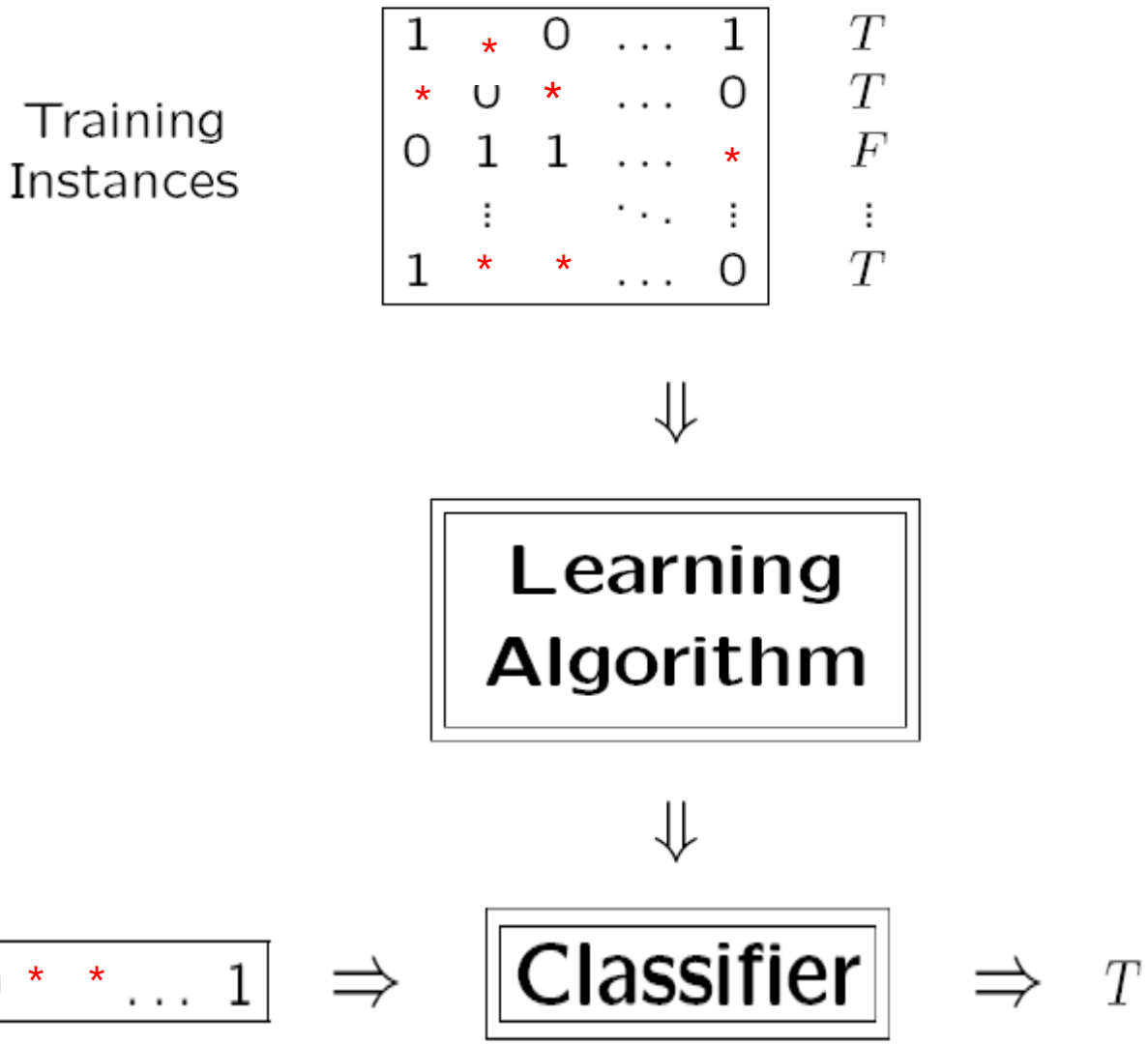
- So far, only considered **ACCURACY**

  In gen'l, may want to consider **COST** as well
  - medical diagnosis: BloodTest costs $150
  - robotics: Width_from_1ft costs 23 sec
- Learn a consistent tree with low expected cost? . . . perhaps replace *InfoGain(S,A)* by
  - $Gain^2(S,A) / Cost(A)$ [Tan/Schlimmer'90]
  - $[2^{Gain(S,A)} - 1] / [Cost(A)+1]^w$

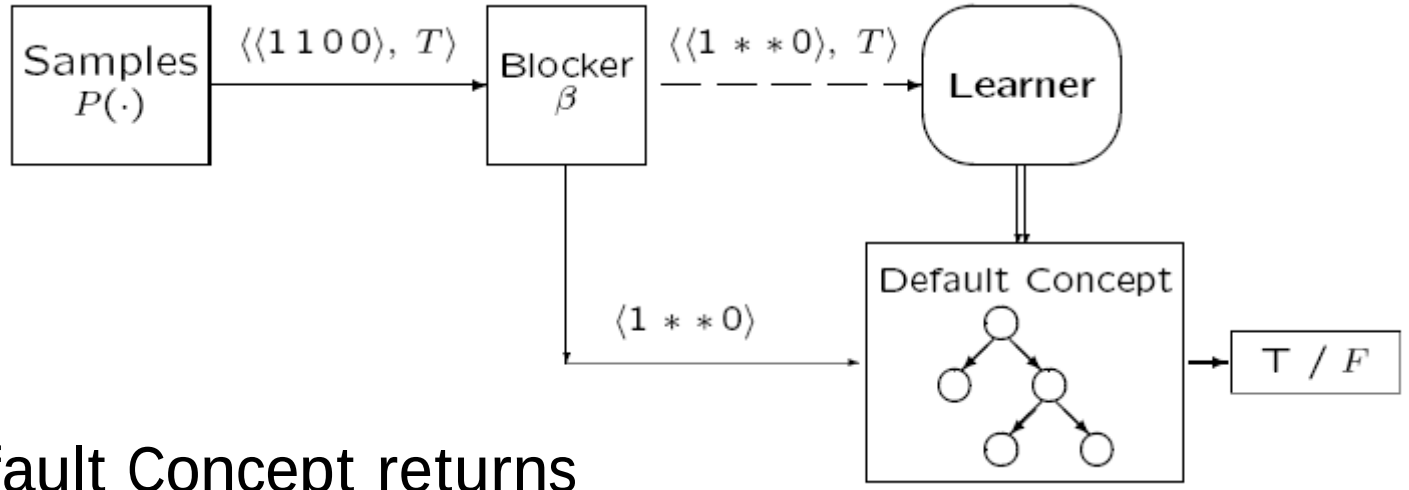
    where $w \in [0, 1]$ determines importance of cost [Nunez'88]
- General utility (arb rep'n)

  $E[\sum_i cost(A_i) +$ Misclass penalty $]$ [Greiner/Grove/Roth'96]

Dealing with Missing Information



Formal Model



- Default Concept returns Categorical Label {T, F } even when given partial instance
  - . . . even ⟨ *,*, ..., * ⟩ !
- Blocker β : { 0, 1}$^n \rightarrow_{\text{Stoch}}$ {0, 1, * }
- N.b., β
  - does NOT map 0 to 1
  - does NOT change class label
  - may reveal different attributes on different instances (on same instance, different times)

Unknown Attribute Values

- Q: What if some examples are **incomplete**
  . . . missing values of some attributes?

When learning:

A1: Throw out all incomplete examples?
  ... May throw out too many. . .

A2: Fill in most common value ("imputation")
  - May miss correlations with other values
  - If impute wrt attributes: may require high order statistics

A3: Follow all paths, w/ appropriate weights
  - Huge computational cost if missing MANY values

When classifying

- Similar ideas . . .
- ISSUE: Why are values missing?
  - Transmission Noise
  - "Bald men wear hats"
  - "You don't care"

  See [Schuurmans/Greiner'94]

Handling Missing Values: Proportional Distribution

- Associate weight w_i with example $\langle x_i, y_i \rangle$
  At root, each example has weight 1.0
- Modify mutual information computations:
  use weights instead of counts
- When considering test on attribute j,
  only consider examples that include x_{ij}
- When splitting examples on attribute j:
  - p_L = prob. non-missing example sent left
    p_R = prob. non-missing example sent right
  - For each example $\langle x_i, y_i \rangle$ missing attribute j:
    send it to both children;
    - to left w/ $w_i := w_i \times p_L$
    - to right w/ $w_i := w_i \times p_R$
  - To classify example missing attribute j:
    - Send it down left subtree; $P(y\tilde{}_L \mid x)$ = resulting prediction
    - Send it down left subtree; $P(y\tilde{}_R \mid x)$ = resulting prediction
    - Return $p_L \times P(y\tilde{}_L \mid x) + p_R \times P(y\tilde{}_R \mid x)$

Handling Missing Values: Surrogate Splits

- **Choose** attribute j and splitting threshold θ_j using all examples that include j

$$u_i = \begin{cases} \text{L if } \langle x_i, y_i \rangle \text{ sent to LEFT subtree} \\ \text{R if } \langle x_i, y_i \rangle \text{ sent to RIGHT subtree} \end{cases}$$

- For each other attribute q, find splitting threshold θ_q that best predicts u_i
  - Sort q by predictive power
  - Called "surrogate splits"
- Sort via "surrogate splits"

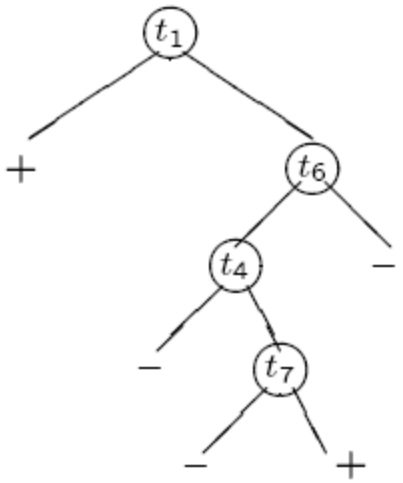
  To handle $\langle x_i, y_i \rangle$ where $x_{ij} = *$:
  - go thru surrogate splits q until finding one NOT missing
  - Use q, θ_q to decide which child gets x_i

Learning Task

Input:

P#1:	⟨	+				⟩	+
P#2:	⟨	−	−	+	+	⟩	−
P#3:	⟨	−		−		⟩	−
P#4:	⟨	−	−	+	−	⟩	+
P#5:	⟨	−	+	+		⟩	−
			⋮				

Output: (small) decision tree consistent with data



Learning Decision Trees … with “You Don't Care” Omissions

- No known algorithm for PAC-learning gen'l Decision Trees

  given all attribute values
- … but Decision Trees are TRIVIAL to learn,

  if superfluous values are omitted:

Algorithm GrowDT

Collect "enough" labeled (blocked) instances

*Let **root** = never-blocked instance xi*

Split instances by $x_i = 1$ vs $x_i = 0$,

and recur (until purity)

Motivation

- Most learning systems work best when
  - few attribute values are missing
  - missing values randomly distributed
- but. . . [Porter,Bareiss,Holte'90]
  - many datasets missing > ½ values!
  - not randomly missing but . . .

    "[missing] when they are known to be irrelevant for classication or redundant with features already present in the case description"

  $\Rightarrow$ Our Situation!!
- Why Learn? . . . when experts
  - not available, or
  - unable to articulate classification process

Comments on Decision Trees

- "Decision Stumps" (1-level DT)
  seem to work surprisingly well
- Efficient algorithms for learning optimal "depth-k decision trees“
  ... even if continuous variables
- Oblique Decision Trees

  Not just "$x_3 > 5$", but "$x_4 + x_8 < 91$"
- Use of prior knowledge
  - Incremental Learners ("Theory Revision")
  - "Relevance" info
- Applications:
  - Gasoil ≈ 2500 rules
    designing gas-oil separation for offshore oil platforms
  - Learning to fly Cessna plane
- Software Systems:
  - C5.0 (from ID3, C4.5) [Quinlan'93]
  - CART
  - . . .

Decision Tree Evaluation

Criterion	LMS	Logistic	LDA	DecTree
Mixed data	No	No	No	Yes
Missing values	No	No	Yes	Yes
Outliers	No	Yes	No	Yes
Monotone transformations	No	No	No	Yes
Scalability	Yes	Yes	Yes	Yes
Irrelevant inputs	No	No	No	Somewhat
Linear combinations	Yes	Yes	Yes	No
Interpretable	Yes	Yes	Yes	Yes
Predictive power	Yes	Yes	Yes	No