CSI-777
Principles of Knowledge Mining

Class 8
Adv. Instance-Based Linear Models

William G. Kennedy,
PhD, CAPT, USN (Ret.)
Center for Social Complexity
Computational and Data Sciences Dept.
College of Science

Review of Previous Class

Review of Homework

Implementations

- Intro to more advanced techniques
- Decision Trees (C4.5)
- Classification Rules
- Association Rules

Intro to More Adv. Techniques

- Last week: Decision Trees & Rules
- This week: Instance-based & Linear models

Decision Trees

- For a categorized dataset:

  Practical, fast, intelligible, learnable
- Build using divide and conquer based on info gain
- How to handle numeric values & missing
- Then pruning

Decision Trees

- C4.5 (Quinlan,1993)
- “See5” (Windows OS) and “C5.0” (Linux): 2002/3

Decision Trees

- C4.5 (Quinlan,1993)
  - Handle numeric values for an attribute:
    - Break points in range of values
    - Based on info gain
    - After establishing a break, change tree & process each child node
    - (sort values once…)

Decision Trees

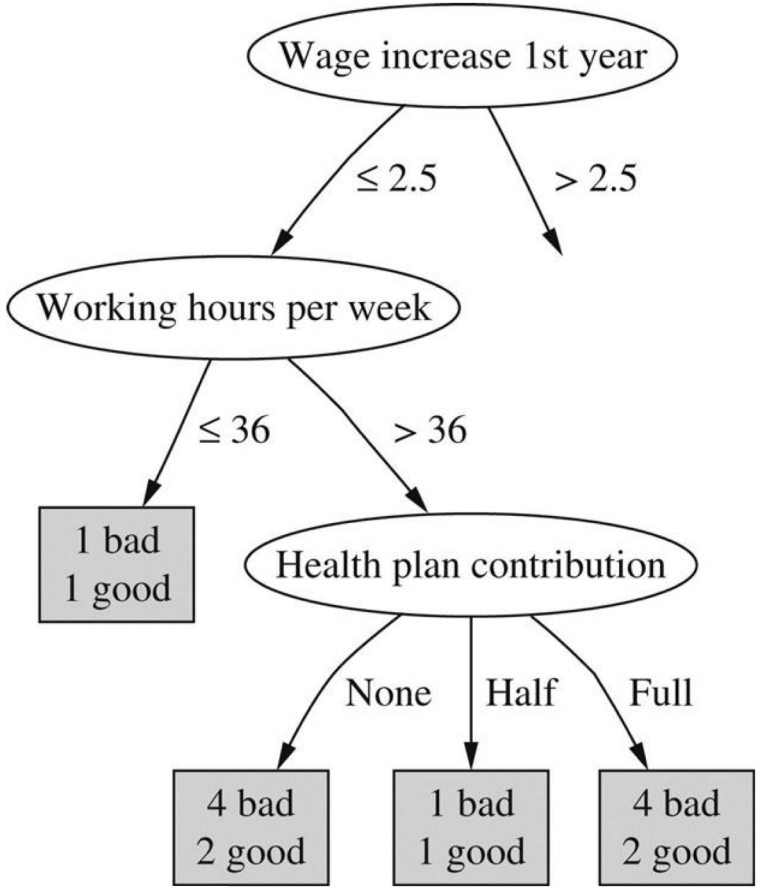
- C4.5 (Quinlan,1993)
  - Handle numeric values for an attribute
  - Handle missing attributes:
    - Treat as an “value”, if significant?
    - Else, ignore instances, or carry on w/o attribute

Decision Trees

- C4.5 (Quinlan,1993)
  - Handle numeric values for an attribute
  - Handle missing attributes
  - Pruning:
    - Post-pruning vs. pre-pruning
    - Post: subtree replacement (leaf) and raising

Decision Trees

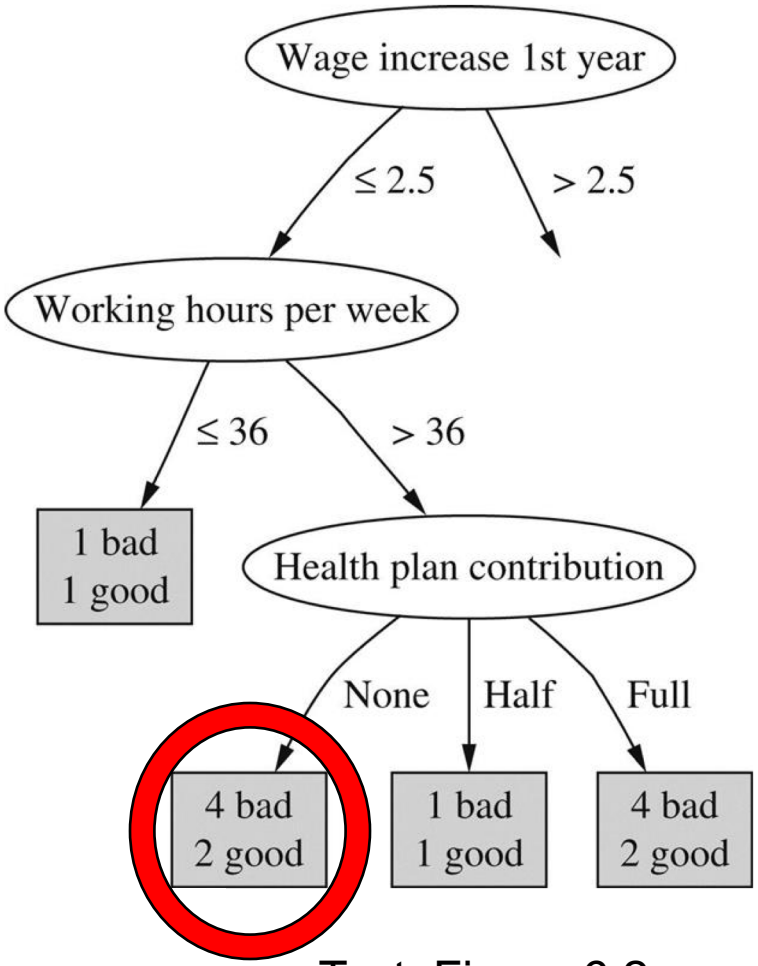
C4.5 Pruning Example



Decision Trees

C4.5 Pruning Example

Pr[calc > z] = .25
z = 0.69
2 errors, N=6
calc error=0.47

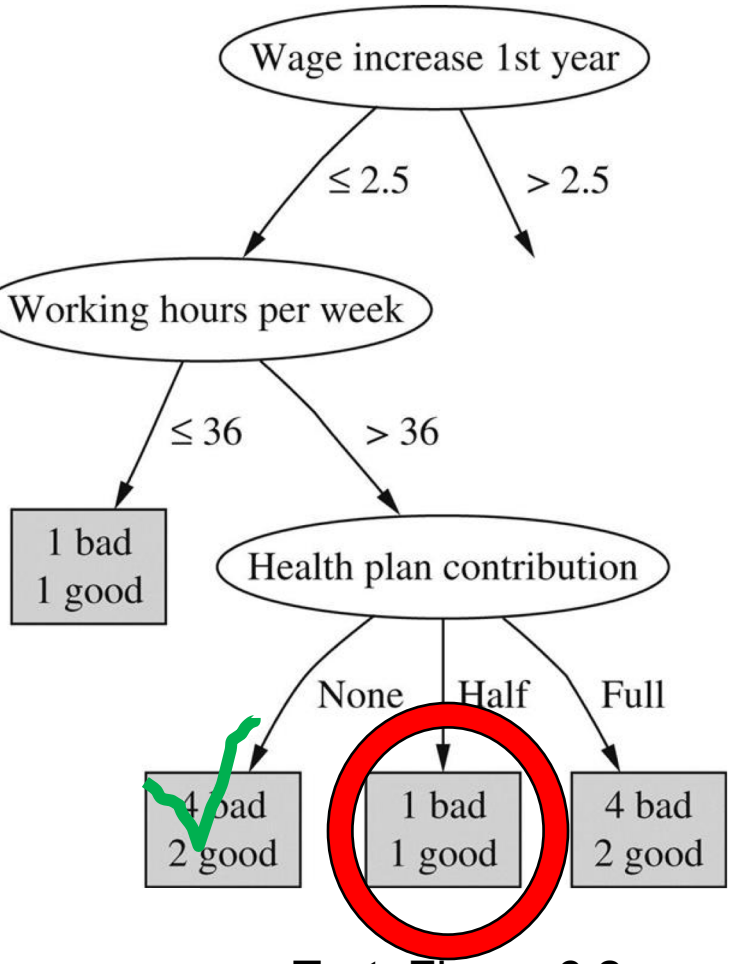


Text, Figure 6.2

Decision Trees

C4.5 Pruning Example

Pr[calc > z] = .25
z = 0.69
1 errors, N=2
calc error=0.72

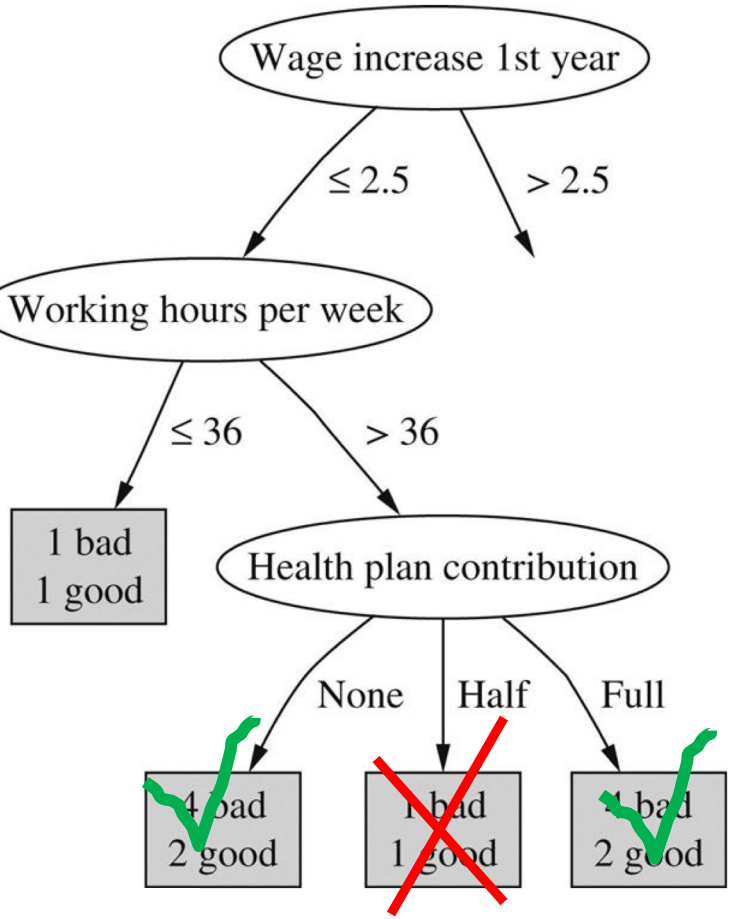


Text, Figure 6.2

Decision Trees

C4.5 Pruning Example

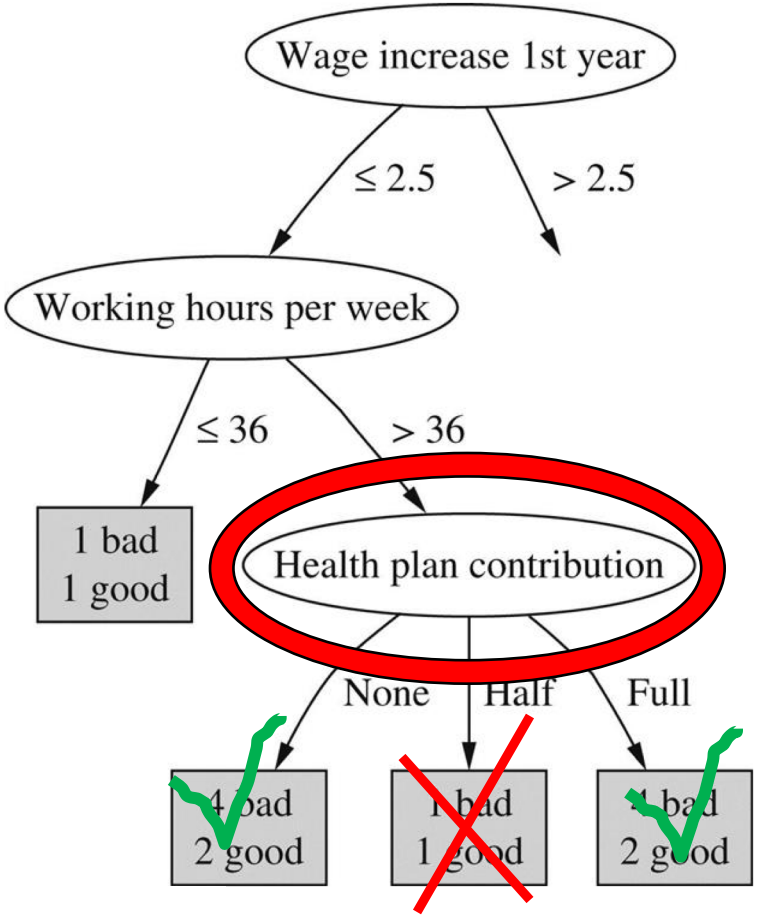
0.47, 0.72, 0.47
6:2:6
calc error=0.51



Text, Figure 6.2

Decision Trees

C4.5 Pruning Example

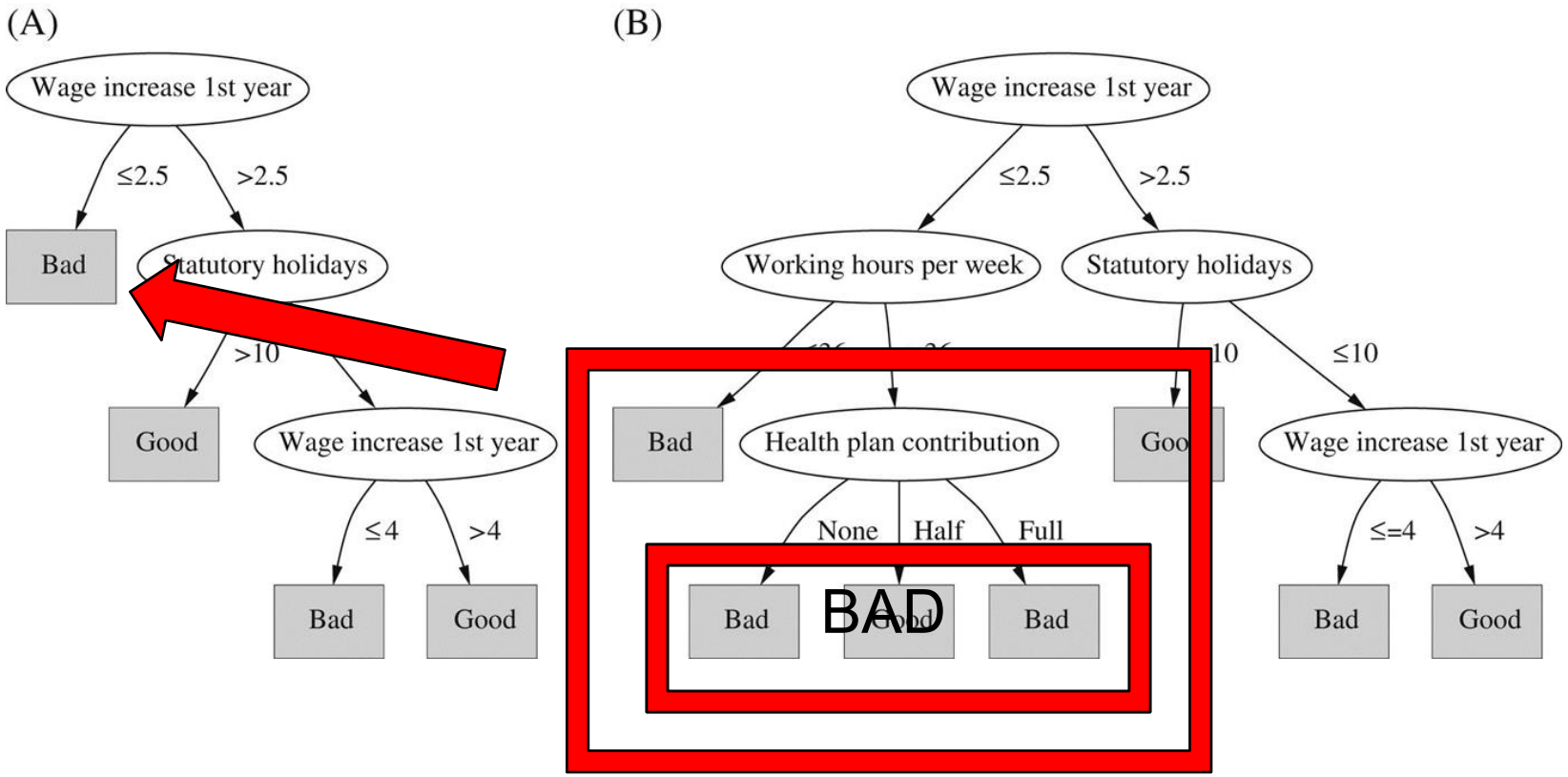


5 errors, N=14
calc =0.46
Parent 0.46, with children 0.51, so, prune children

Text, Figure 6.2

Decision Trees

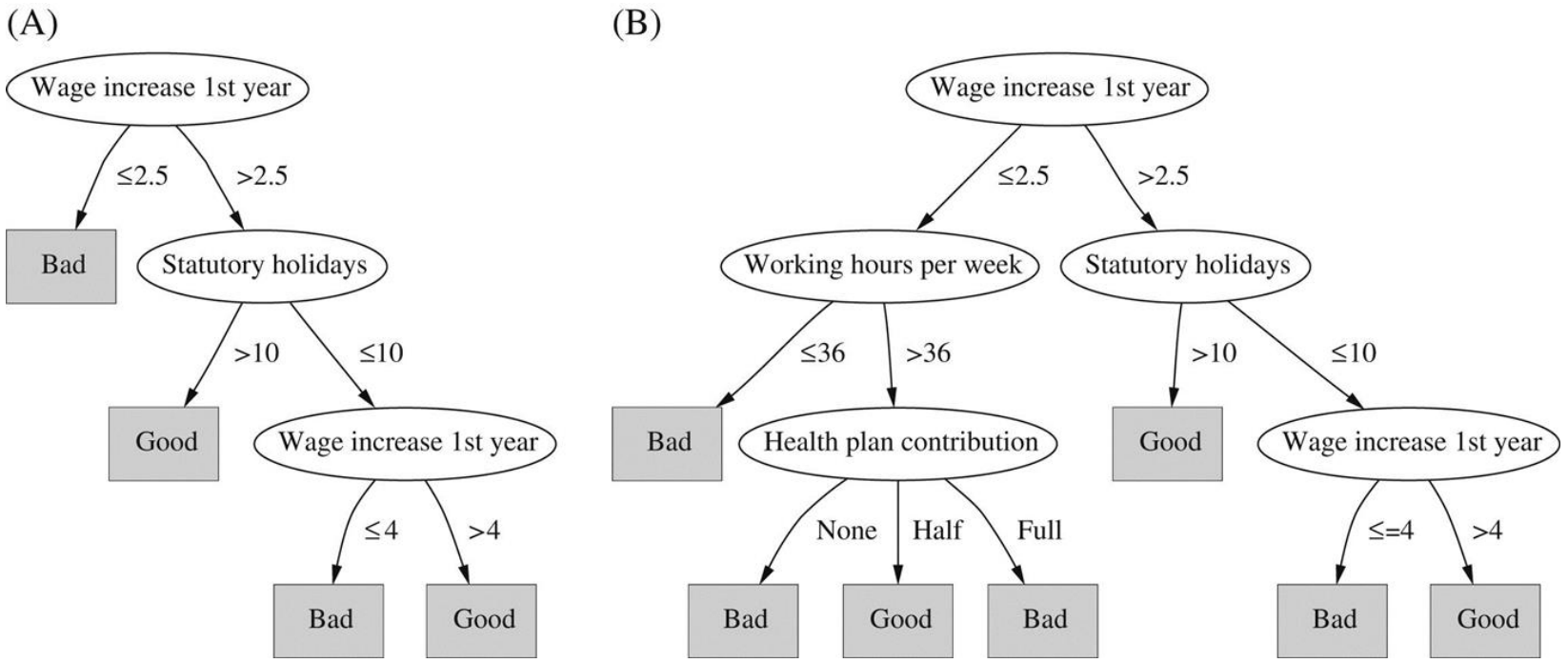
Subtree replacement



Text, Figure 1.3

Decision Trees

Subtree replacement



Text, Figure 1.3

Decision Trees

Subtree raising (in C4.5)

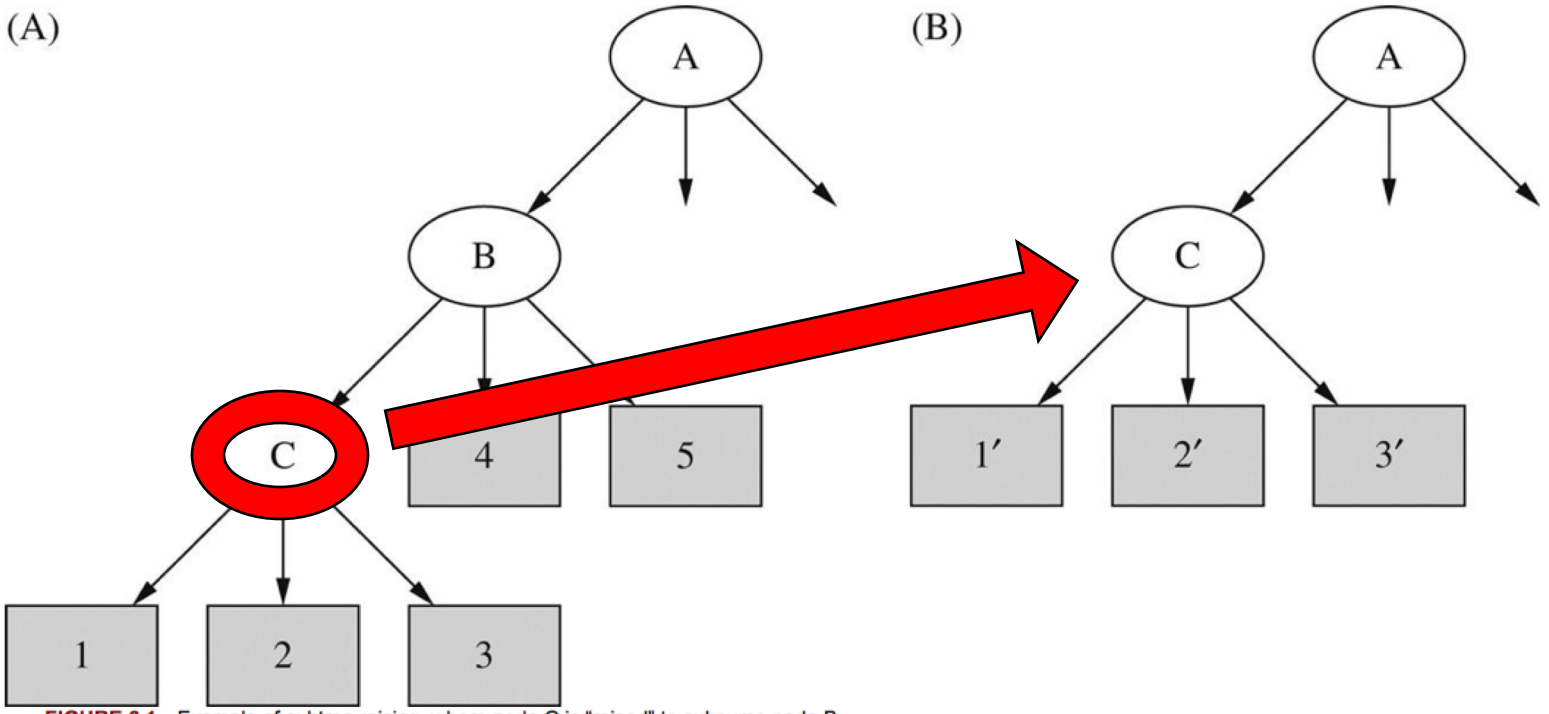


FIGURE 6.1 Example of subtree raising, where node C is "raised" to subsume node B.

Text, Figure 6.1

Decision Trees

- C4.5
  - Estimating Error Rates from Pruning
  - Use testing methods
  - C4.5 uses training set
    - Works well in practice
    - Uses confidence of 25% (50% band)

Decision Trees

Complexity in inducing trees

- Say, *n* instances, *m* attributes
- Assume depth of tree O(log *n*) i.e., *n* leaves
- Assuming most instances diff, *m* tests enough to decide

Decision Trees

Complexity in inducing trees

- Computational cost:
  - @ each depth, all n instances to be considered
  - For each attribute: O(n log n)
  - Then, each node, all attributes: O(mn log n)
  - Sorting (once per numeric attribute): O(n log n)
  - Pruning by subtree replacement: O(n)
  - Pruning by subtree lifting: O(n (log n)^2)

Bottom line: O(mn log n) + O(n (log n)^2)

So what?

theoretical performance vs. measured performance

Decision Trees

Trees to Rules

- 1 for each leaf w/conditions back to root
- Consider error rate if each condition was removed
- Could produce dupes to be removed
- Above is greedy, other approaches may be better, but will be more expensive
- Computational costs: evaluate on training set… slow

Decision Trees

C4.5 settings

- C5.0 (commercial version) faster at rule generation
- Default confidence value of .25 should be lower (more pruning) (Test error rate of sys in use?)
- Test min. # instances, default 2… raise if data noisy
- Candidate splits must effect min of (10% or 25 inst.)
- Includes a pre-pruning based on info gain

- C4.5 Release 8 implemented in Weka

Decision Trees

Remaining Points on Trees

- Top-down induction of trees a researched area
- May be basically done…
- Description in text based on single attributes
- Linear combination possible
  - called ”multivariate” decision trees
  - CART supports as an option (Classification and Regression Trees)

Classification Rules

- Previously described basic covering algorithm for generating rules
- More than just correctness-based
- General problem: overfitting & not generalizing
- Industrial-strength rule learners:
  - Separate-and-conquer with global optimization
  - Incremental learner using partial decision-trees

Classification Rules

Testing rules:

- Previously, cover max instances correctly
- Alternatively, an information-theory-based:

$$p[\log (p/t) - \log (P/T)]$$

p, t =instances (new rule)

P,T =instances satisfying rule before new conditional

“info gained” by added conditional * instances

Classification Rules

Handling missing and numeric values:

- Not much new
- Missing treated as not matching any
- Exceptions treated late in the process (advantage of covering algorithms)
- Numeric values, same: sort & thresholds for binary divisions considered

Classification Rules

Generating Good Rules:

- Not perfect, but “sensible” rules w/o overfitting (potentially better)

Reference Dataset: Contact Lens

Age	Spectacle Prescription	Astigmatism	Tear Production Rate	Recommended Lenses
Young	Myope	No	Reduced	None
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Reduced	None
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	Hard
Prepresbyopic	Myope	No	Reduced	None
Prepresbyopic	Myope	No	Normal	Soft
Prepresbyopic	Myope	Yes	Reduced	None
Prepresbyopic	Myope	Yes	Normal	Hard
Prepresbyopic	Hypermetrope	No	Reduced	None
Prepresbyopic	Hypermetrope	No	Normal	Soft
Prepresbyopic	Hypermetrope	Yes	Reduced	None
Prepresbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Reduced	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Classification Rules

Generating Good Rules:

- 2 rules: 4/6 and 2/2 correct. Which better?
- Pre-prune or post-prune?
- Separate growing & pruning datasets (2:1)
- “Reduced-error” pruning (also incremental…)
- Simple measure of “better” difficult

Classification Rules

Generating Good Rules

(incremental, reduced error-pruning):

```
Initialize E to the instance set
Split T into Grow and Prune ration 2:1
For each class C with instance in both G&P
  Create best perfect rule for C
  Calc worth of rule, w(R) using Prune and with
latest condition omitted, w(R-)
  While w(R-)>R(w), remove latest condition & re-eval
#have rules for all classes
Select rule with largest w(R)
Remove instances covered by rule from E
Continue
```

Based on Figure 6.3

Classification Rules

Generating Good Rules (practical advice):

- Algorithm faster if process one class at a time
- Maybe work smallest to largest class
- Terminate when rule below threshold (not best, MDL?)

Classification Rules

Global Optimization:

- Experiments: worthwhile to optimize resulting rules
- Optimization complicated
- Example of “elaborate industrial-strength rule learners”: Ripper (Cohen 1995) Repeated Incremental Pruning to Produce Error Reduction, a variation of sequential covering algorithm

Classification Rules

Rules from Partial Decision Trees:

- Avoids global optimization
- Divide & conquer combined with separate & conquer
- To make single rule, build pruned decision tree for current instances, leaf with largest coverage => rule
- Key idea: build partial decision tree (with undefined subtrees)

Classification Rules

Rules from Partial Decision Trees:

```
Expand-subset(S)
  Use test T to split examples into subsets
  Sort subsets by increasing ave. entropy
  While (subset X not yet expanded) AND (subsets
         not all leaves):  expand-subset(X)
  if (all expanded subsets leaves) AND (est. error
      for subtree ≥ est. error for node):
         undo expansion and make node a leaf
```

Classification Rules

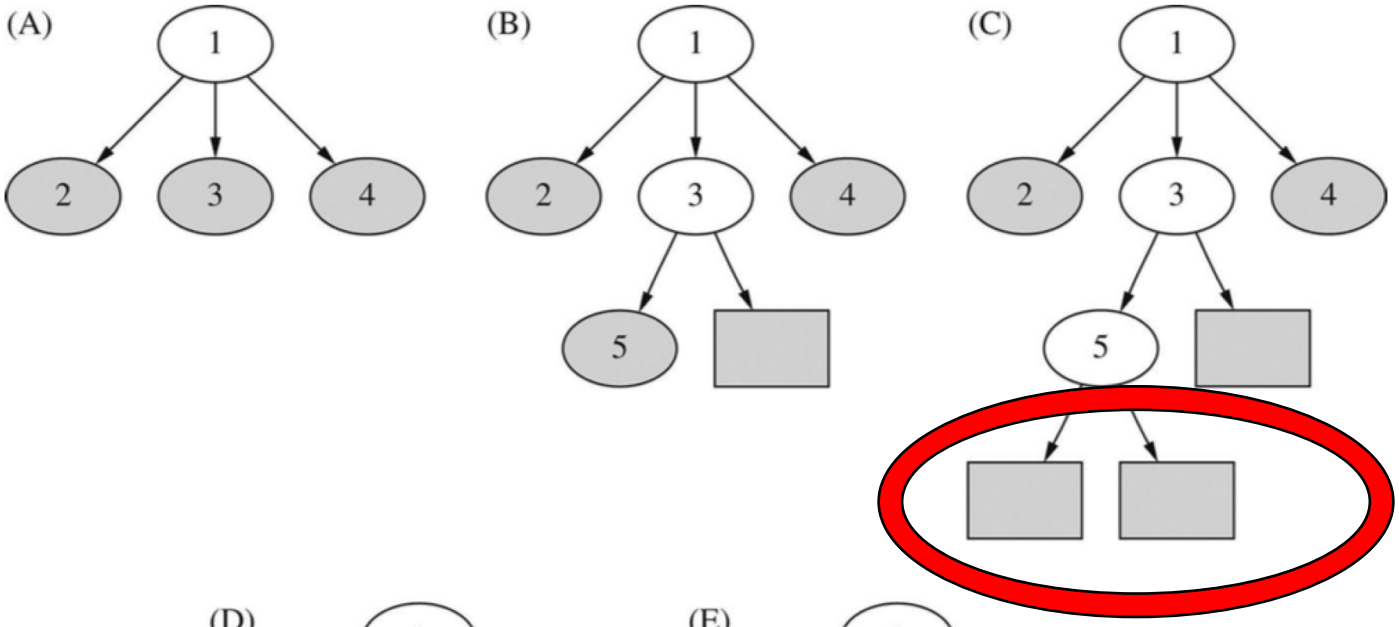


Fig. 6.6

Classification Rules

Rules from Partial Decision Trees:

- Next step: build 1 rule from partial tree
- Pick best leaf (lowest entropy)
- Experiments: pick rule covering most instances

Notes:

- If data noise-free w/ enough instances to prevent pruning, just full decision tree to be explored=> best gain
- With numeric attributes: time complexity driven by sorting makes developing full tree effective
- Missing values processed proportional to training instances covered

Classification Rules

Rules with Exceptions:

- Start with top level, default rule: T/F
- If either subset of instances has more then 1 class, repeat recursively
- Proceeds until singletons, if necessary

Classification Rules

Rules with Exceptions:

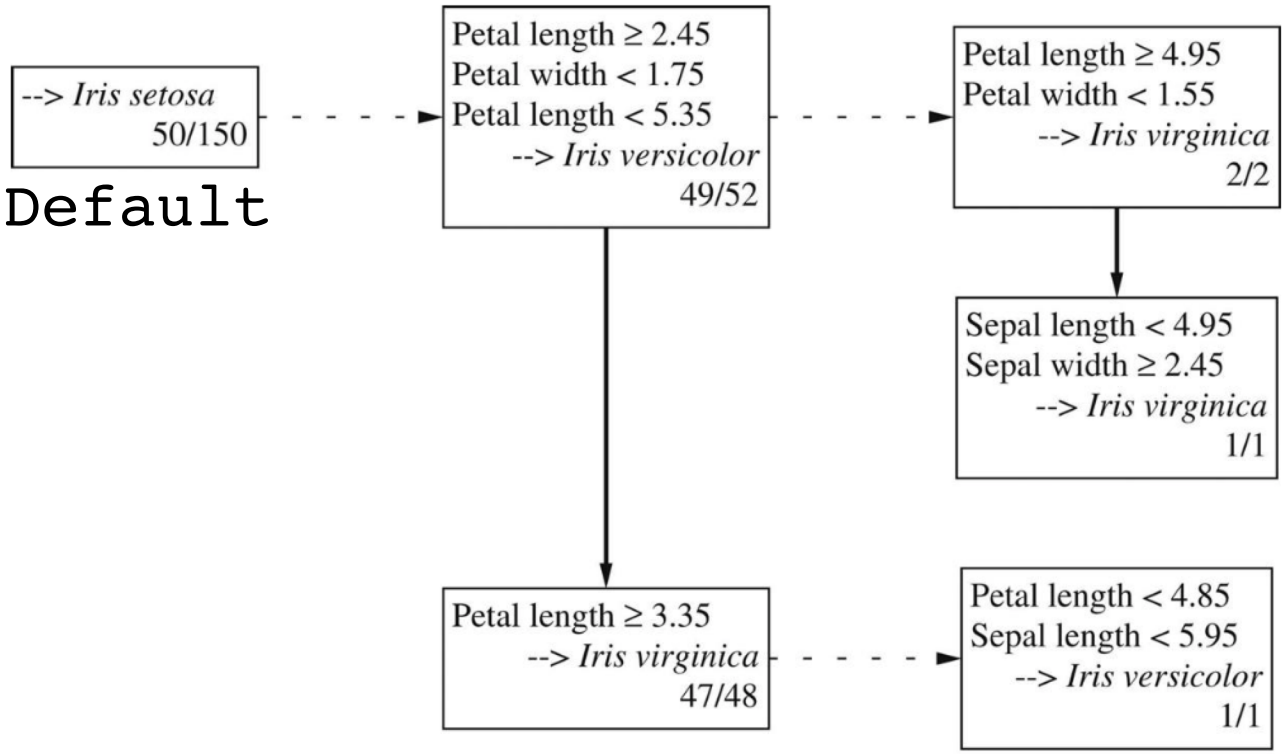


Fig 6.7

Classification Rules

Rules with Exceptions:

```
Default: Iris-setosa
except if petal-length ≥ 2.45 and petal-length < 5.355
          and petal-width < 1.75
       then Iris-versicolor
            except if petal-length ≥ 4.95 and petal-width < 1.55
                   then Iris-virginica
                   else if sepal-length < 4.95 and sepal-width ≥ 2.45
                        then Iris-virginica
       else if petal-length ≥ 3.35
            then Iris-virginica
                 except if petal-length < 4.85 and sepal-length < 5.95
                        then Iris-versicolor
```

FIGURE 3.8 Rules for the iris data.

Classification Rules

Discussion - well studied area

- Basic classification rule system “PRISM” 1987
- With noisy data, can’t have perfect rules
- Incremental pruning: RIPPER 1994 & 95

Fig 6.7

Association Rules

- Previously, generated assoc. rules for min. support & confidence threshold
- Generate & Test lengthening conditions with many runs through full dataset
- Can be reduced to two runs through building frequent-pattern trees

Association Rules

- In first pass, count occurrences of att-value pairs
- In second pass, instances not meeting thresholds: not included in tree
- E.g., with support threshold of ≥6, counts:

play = yes	9		
windy = false	8	play=yes & windy=false	6
humidity = normal	7	play=yes & humidity=normal	6
humidity = high	7		
windy = true	6		
temp = mild	6		

Association Rules

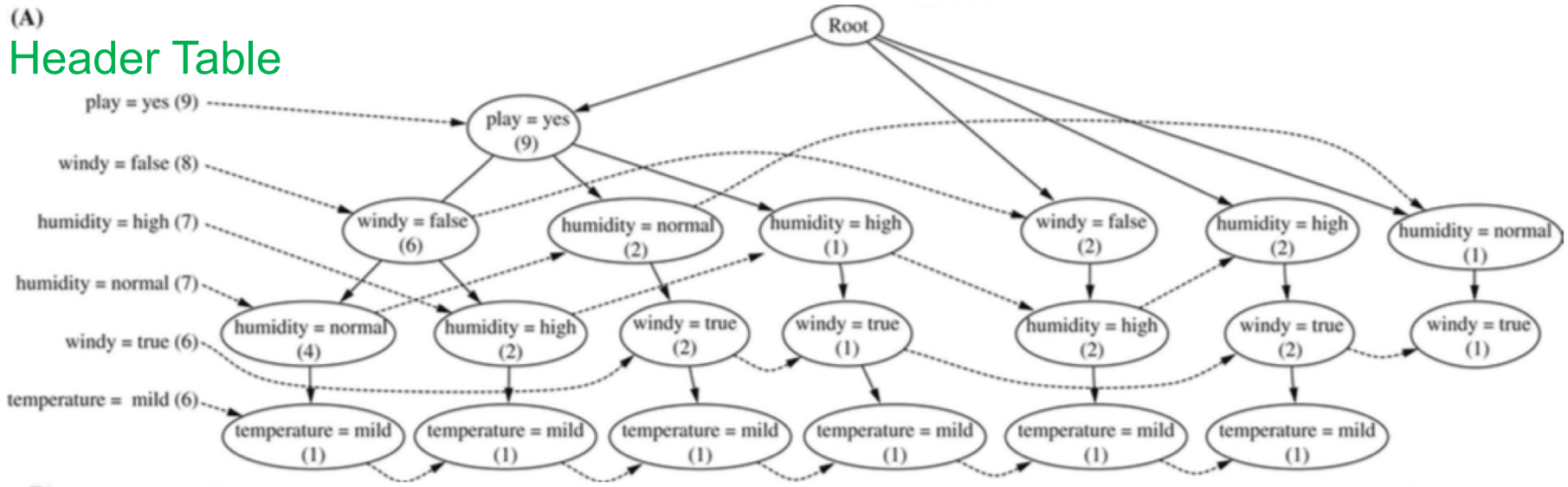


Figure 6.8a (full tree solid arrows)
Min support of 6

Association Rules

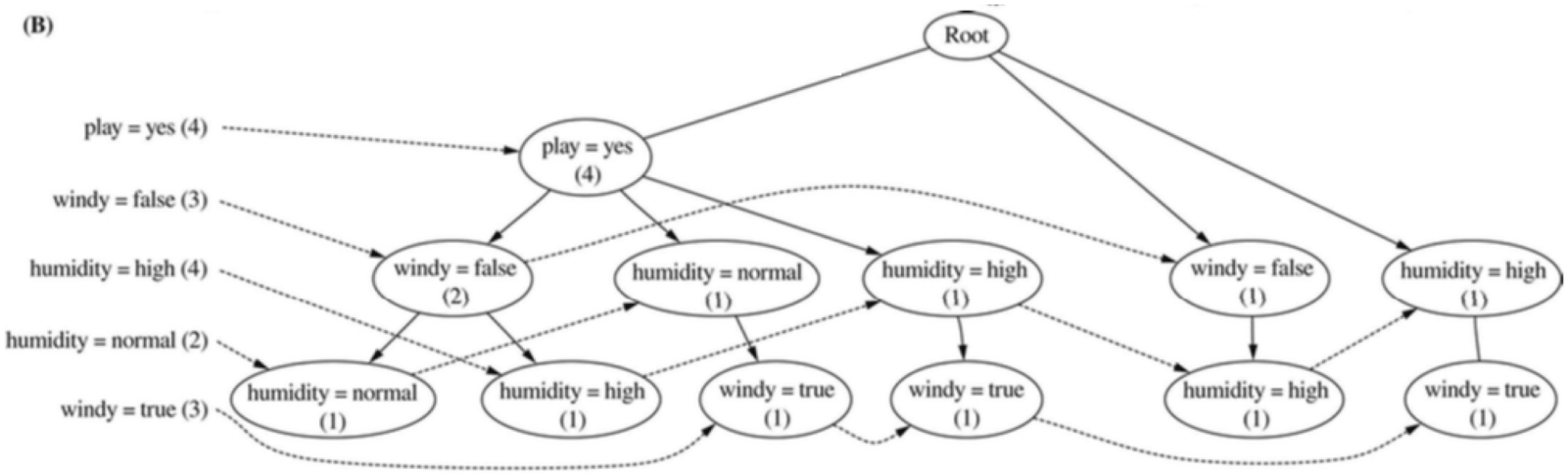


Figure 6.8b (cond on temp=mild)

Association Rules

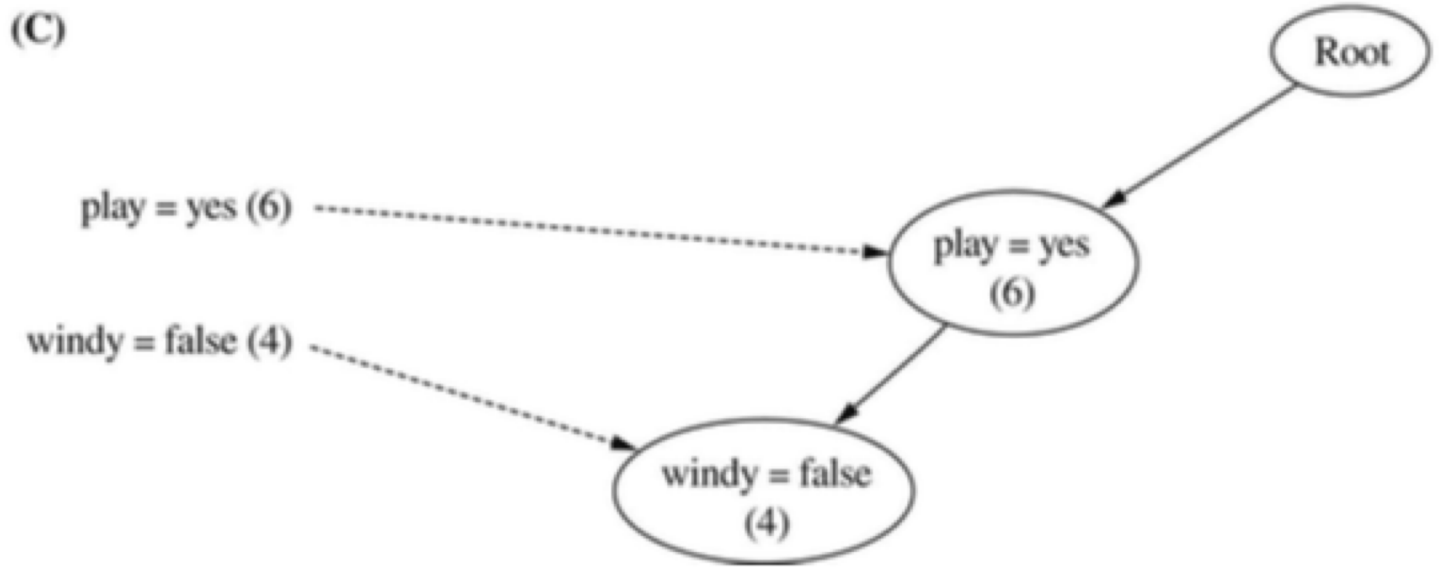


Figure 6.8c (cond on humidity=normal)

Association Rules

- Once large sets identified, create rules
- Condition sets with support:

  play=yes, windy=false, temp=mild, &

  (humidity=normal & play=yes)
- Rules

  play=yes: if (windy=false) then (play=yes) (6/9)

  windy=false: if (humidity=normal) then (windy=false) (4/6)

  …

Implementations

- Decision Trees (C4.5)
- Classification Rules
- Association Rules

New Material

Advanced Approaches 2

1. Instance-based Learning
2. Extending Linear Models
3. Local Linear Models

Instance-Based Learning

- Practical problems with basic k-NN:
  - Slow for large datasets, one pass through for each test instance
  - Doesn’t handle noisy data well
  - Presumes balanced impact of attributes
  - Doesn’t explicitly generalize

Instance-Based Learning

Which exemplars to keep?

- Keep all… unnecessary
- Discard correctly classified?
  - Ideal: 1 exemplar per class
  - But may need “errors” to adapt, but …
- Keep errors?
  - If noisy, keeping noise…

Instance-Based Learning

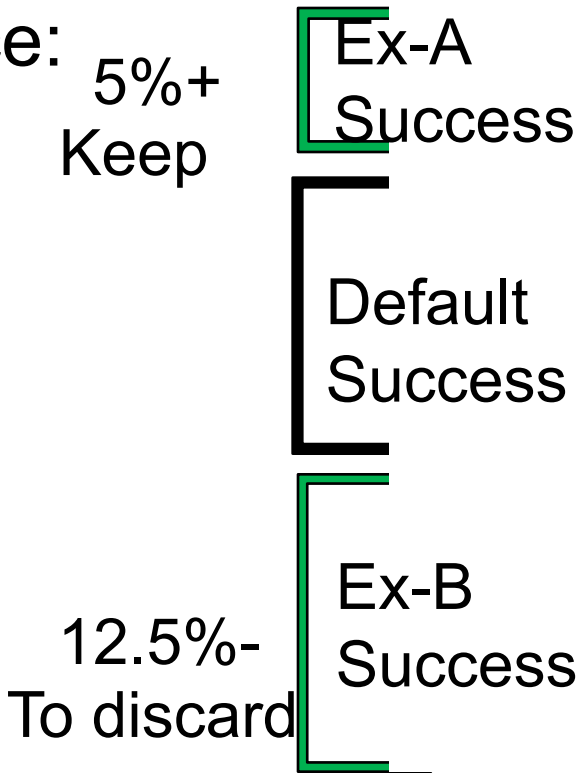
Noisy exemplars

- Keeping them causes problems
- 1st option: Keep k NN, rather than 1, and vote
- Noise ↑ need k ↑, explore several values…

Instance-Based Learning

Noisy exemplars

- Keep stats on performance:
  - Too low, discard
  - High, use
  - Between, keep around
- Confidence limits
  - Level & bounds
  - IB3



Instance-Based Learning

Weighting Attributes

- Basic: same distance measure for all but…
- Learn relevance of each attribute (+/-)
- Updated with each training instance

Instance-Based Learning

Generalized Attributes

- Rectangular exemplars (“hyperrectangles”)
- +: Generalized by merging with nearest
- -: Shrunk away from new instance
- Overgeneralize by growing allowed Y/N

Instance-Based Learning

Distance Functions

- With generalized exemplars, simplify measure
- 0 or dist. to closest part of rectangle
- Generalized distance measure: use all transformations to a similar class

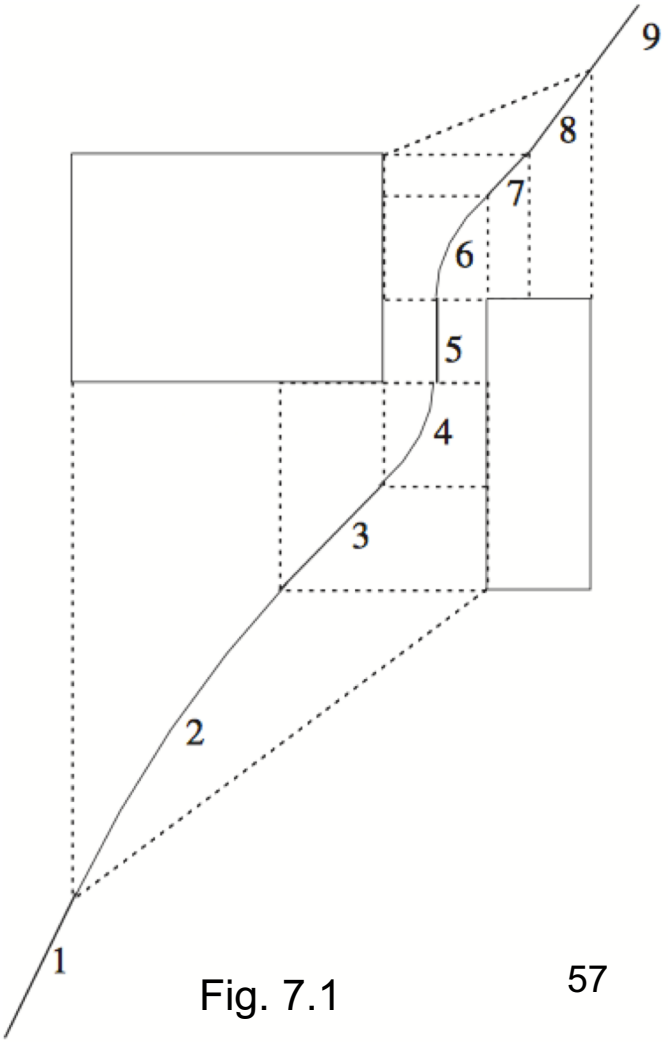


Fig. 7.1

Extending Linear Models

- LM weakness: only linear boundaries
- SVM use LM to implement non-linear boundaries (non-linear mapping of input):

$$x = w_1 a_1^3 + w_2 a_1^2 a_2 + w_3 a_1 a_2^2 + w_4 a_2^3$$

(2 attributes, 4 weights)

- But …Computational complexity: 10a w/5 factors=>2,000 coef.
- and … Overfitting

Extending Linear Models

- SVM address both weaknesses
- Max margin hyperplane = max separation
- Set of SVs define max margin hyperplane
- Finding SVs for training instances solved optimization problem
- Text describes how SVM solve both weaknesses

Extending Linear Models

Support Vector Regression: from classification to prediction

- Find function that minimizes prediction error (predicting training data)
- Tradeoff prediction error ε and tube's flatness

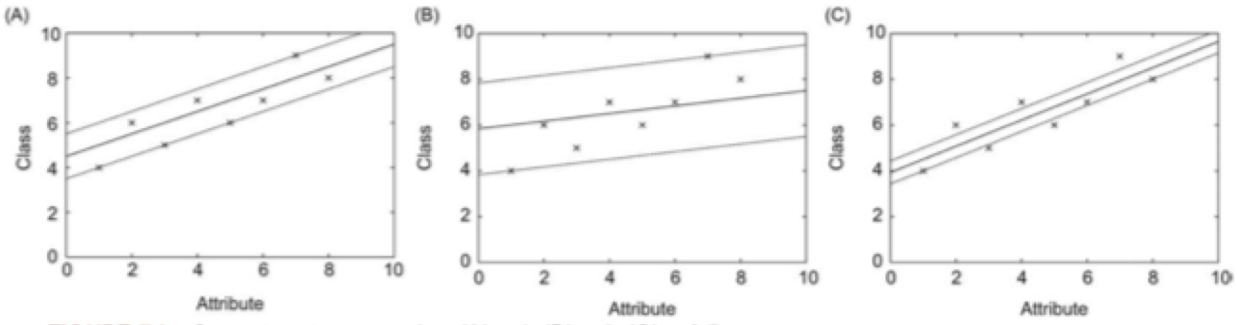
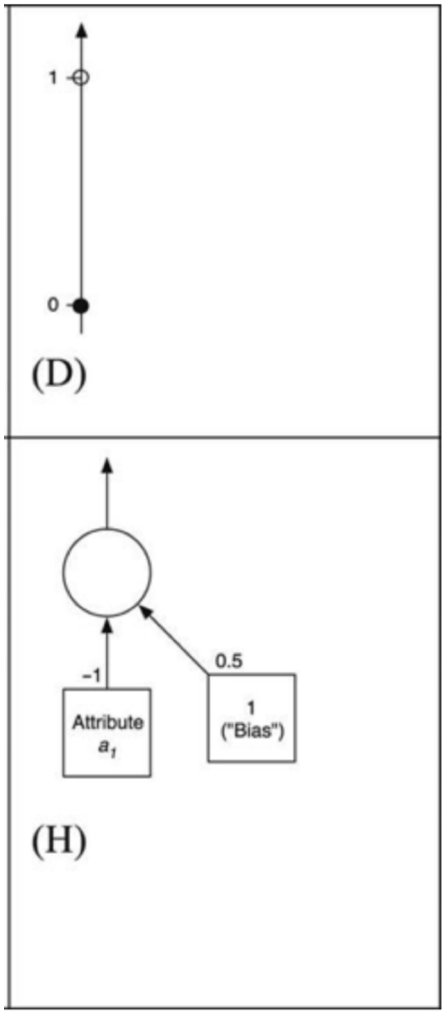


FIGURE 7.3 Support vector regression: (A) ε=1; (B) ε=2; (C) ε=0.5.

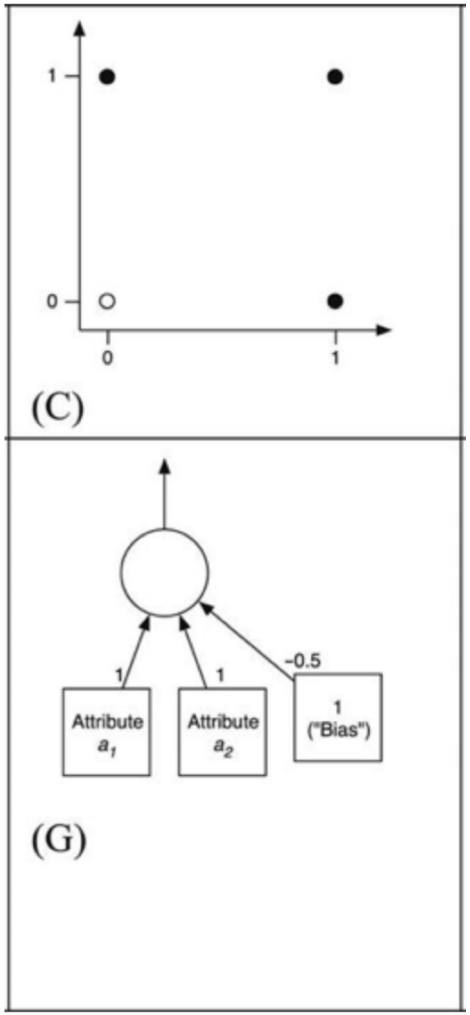
Extending Linear Models

Multi-layer Perceptrons



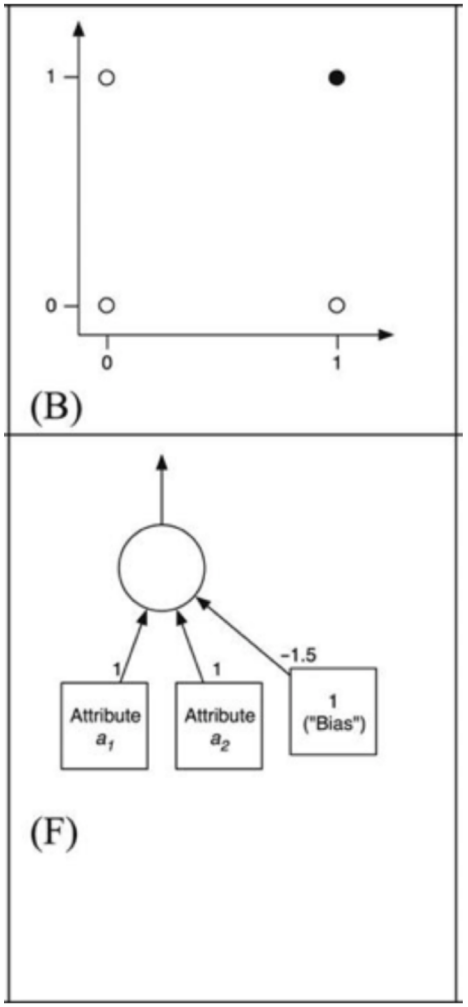
Extending Linear Models

Multi-layer Perceptrons



Extending Linear Models

Multi-layer Perceptrons

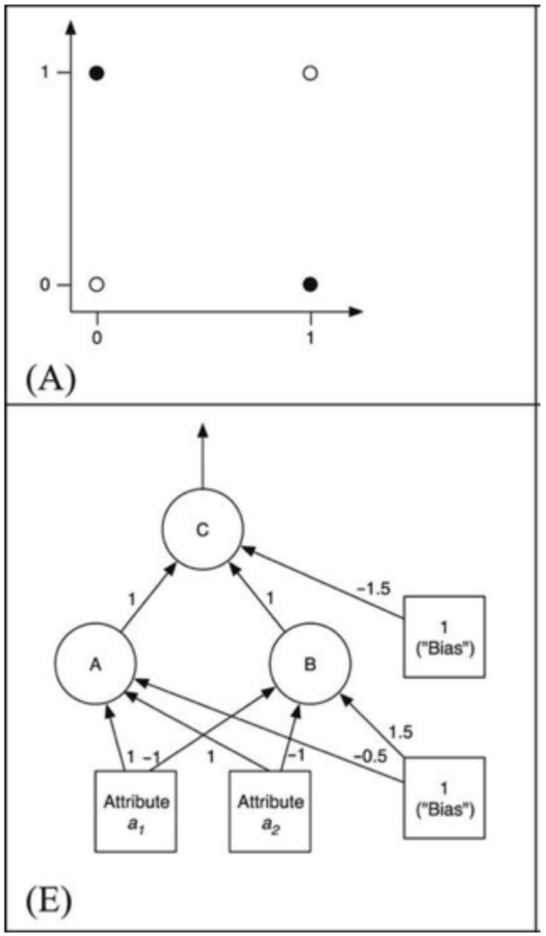


Extending Linear Models

Multi-layer Perceptrons

- Discussed XOR difficulty
- Multi-layer fixes XOR
- Train adjusting weights with “BackProp”

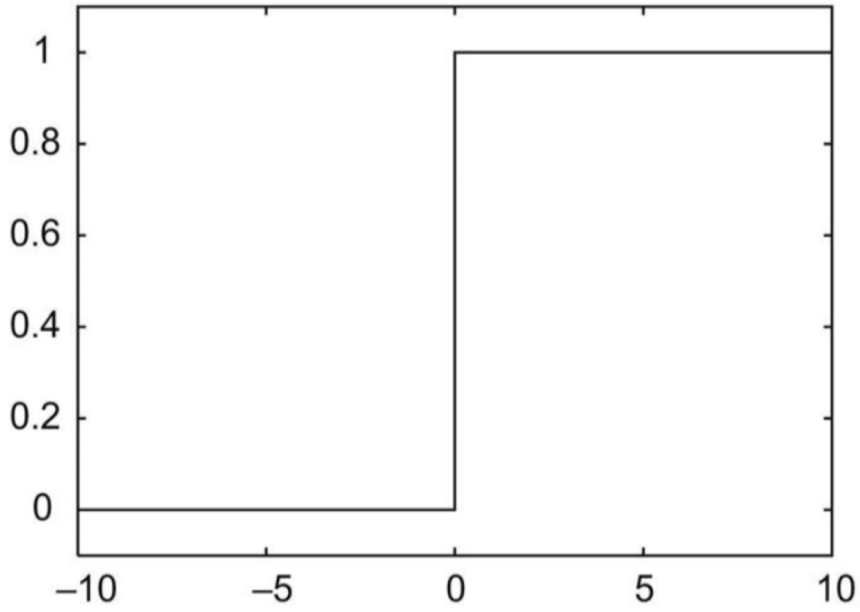
A is OR, B is AND, C is NAND.



Extending Linear Models

Multi-layer Perceptrons

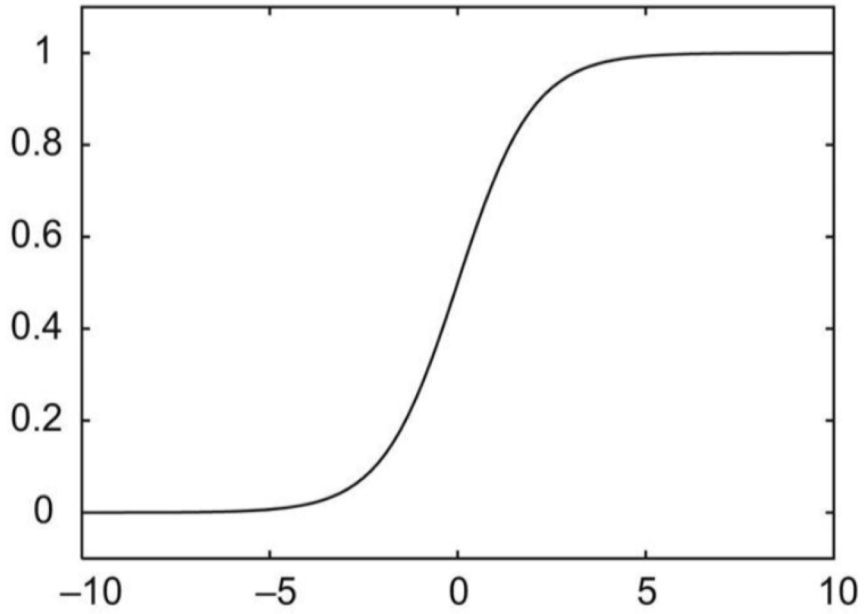
- Discussed XOR difficulty
- Multi-layer fixes XOR
- Train adjusting weights with “BackProp”



Extending Linear Models

Multi-layer Perceptrons

- Discussed XOR difficulty
- Multi-layer fixes XOR
- Train adjusting weights with “BackProp”



$$f(x) = \frac{1}{1 + e^{-x}}$$

Extending Linear Models

Stochastic Gradient Decent

- Gradient decent general purpose optimization
- Can learn linear models such as SVM & logistic regression
- Not at all what I expected…

Numeric Prediction with Local Linear Models

- Trees with numeric prediction: leaves either a class or ave. value of instances.
- “Regression” tree or “model” tree
- Decision tree down to class or linear model
- Pruning involved in tree

Numeric Prediction with Local Linear Models

Model Tree:

- Smooth over linear models to cover gaps
- Back up to the root through LMs at each node

$$p' = (np + kq) / (n + k)$$

p' : passed up
q : prediction at this node
p : prediction from below
n : number of instances covered below
k : smoothing factor

Numeric Prediction with Local Linear Models

Building a Model Tree:

- Splitting training set at each node based on the attribute w/ max error reduction
- StdDev used as error measure (portion of StdDev from this training set)
- Split until reach too small improvement (5%) or too few instances left
- (convert nominal attributes into numeric attributes)
- (modify error measure to account for missing)
- (surrogate splitting … class value as heuristic)

Numeric Prediction with Local Linear Models

Pruning a Model Tree:

- Estimated error rate at each node adjusted by number of instances and parameters in LM
- Simplify (reducing attributes) until est. error increases
- Prune nodes below if error of lower nodes exceeds current node

Numeric Prediction with Local Linear Models

Algorithm for Model Tree:

```
MakeModelTree (instances)
{
   SD = sd(instances)
   for each k-valued nominal attribute
      convert into k-1 binary attributes
   root = newNode
   root.instances = instances
   split(root)
   prune(root)
   printTree(root)
}
```

Numeric Prediction with Local Linear Models

Rules from Model Trees:

- Just like other trees: pick leaf, build rule, remove instances, repeat
- Rules have LM on RHS (then clause)

Numeric Prediction with Local Linear Models

Locally Weighted Linear Regression:

- For subdivided instance space, create LM for each
- Increase wt of instances close test instance
- Select smoothing factor, controls # instances to consider (like *k*NN)
- Locally weighted naïve Bayes algorithm > naïve Bayes and kNN

Advanced Approaches 2

1. Instance-based Learning
2. Extending Linear Models
3. Local Linear Models

The Project

The Project

Knowledge Mining project: 25%

The knowledge mining modeling project is intended to have students apply their knowledge of the subject by developing an analysis of a large dataset their choosing. Projects will be done individually or in teams of no more than 2 and presented to the class near the end of the classes. Students will propose a project (5 pts) and the instructor will provide feedback on scope and projected level of difficulty. Presented projects will be graded demonstrated knowledge mining operation (7 pts), usefulness of patterns reported (7 pts), and explanation of results, (6 pts).

The Project

Knowledge Mining project proposal: (5pts)

1. A title
2. Research question(s) (ends with “?”)
3. Identification of the source dataset (source, size, complexity)
4. Identification of methods & tools to be applied
5. Expected results

The Project

Presented projects will be graded on:

- demonstrated knowledge mining operation (7 pts),
- usefulness of patterns reported (7 pts), and
- explanation of results, (6 pts).

SAVE THE DATE

PhD Student Lunch with Interim Dean Andalibi

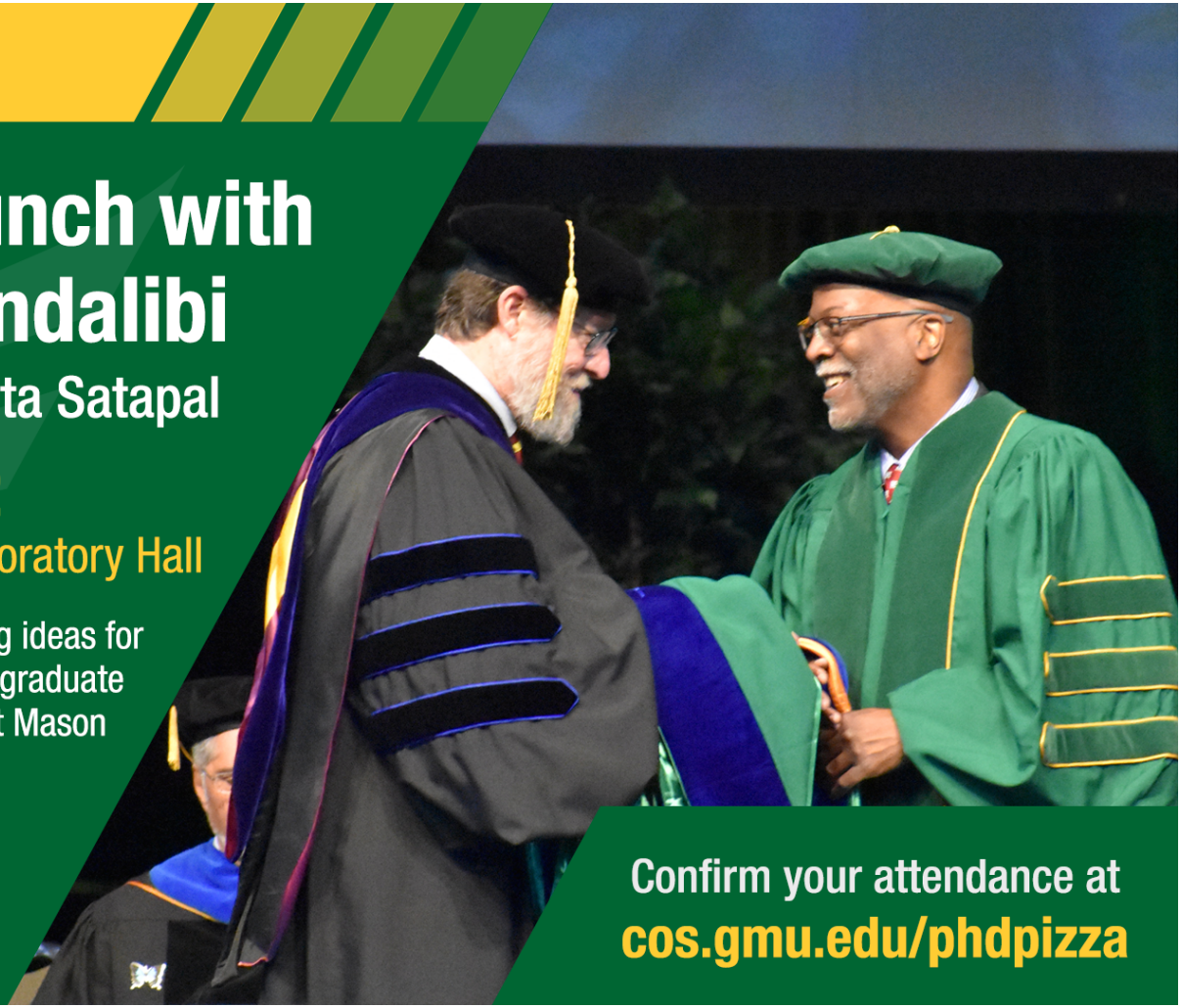
and Dean's Fellow, Shobita Satapal

Tuesday, October 22

12:15 to 1:30 p.m., 3301 Exploratory Hall

Meet fellow PhD students while sharing ideas for PhD career support and improving the graduate school experience in the college and at Mason

Pizza and beverages provided



GEORGE MASON UNIVERSITY | College of Science

Confirm your attendance at
cos.gmu.edu/phdpizza

For next week

- Read: text chap. 6*
- **Ex4:** Do one of the following (A or B):
- A: Using the iris data, evaluate C4.5 using the training set and cross-validation. Which is more realistic and why?
- B: Using the weather data, generate and count all the rules for combinations of minimum confidence 0.7-0.9 and support 0.1-0.3.

Your Questions?