POLITECNICO DI MILANO

Classification Rules

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

Prof. Pier Luca Lanzi

The Weather Dataset

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

A Rule Set to Classify the Data

- IF (humidity = high) and (outlook = sunny)
  THEN play=no (3.0/0.0)
- IF (outlook = rainy) and (windy = TRUE)
  THEN play=no (2.0/0.0)
- OTHERWISE play=yes (9.0/0.0)

Let's Check the First Rule

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

IF (humidity = high) and (outlook = sunny) THEN play=no (3.0/0.0)

Then, The Second Rule

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

IF (outlook = rainy) and (windy = TRUE) THEN play=no (2.0/0.0)

Finally, the Third Rule

Outlook	Temp	Humidity	Windy	Play
[redacted]				
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
[redacted]				
Overcast	Cool	Normal	True	Yes
[redacted]				
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
[redacted]				

IF (outlook = rainy) and (windy = TRUE) THEN play=no (2.0/0.0)

A Simpler Solution

- IF (outlook = sunny) THEN play IS no
  ELSE IF (outlook = overcast) THEN play IS yes
  ELSE IF (outlook = rainy) THEN play IS yes
  (6/14 instances correct)

What is a Classification Rule?

What Is A Classification Rules? Why Rules?

- They are IF-THEN rules
  - The IF part states a condition over the data
  - The THEN part includes a class label

- What types of conditions?
  - Propositional, with attribute-value comparisons
  - First order Horn clauses, with variables

- Why rules?
  - Because they are one of the most expressive and most human readable representation for hypotheses

Coverage and Accuracy

- IF (humidity = high) and (outlook = sunny)
  THEN play=no (3.0/0.0)

- ncovers = number of examples covered by the rule
- ncorrect = number of examples correctly classified by the rule

- coverage(R) = ncovers/size of the |training data set
- accuracy(R) = ncorrect/ncovers

Conflict Resolution

- If more than one rule is triggered, we need conflict resolution
- Size ordering
  - Assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the most attribute test)
- Class-based ordering
  - Decreasing order of prevalence or misclassification cost per class
- Rule-based ordering (decision list)
  - Rules are organized into one long priority list, according to some measure of rule quality or by experts

Two Approaches for Rule Learning

- Direct Methods
  - Directly learn the rules from the training data
- Indirect Methods
  - Learn decision tree, then convert to rules
  - Learn neural networks, then extract rules

IR Classifier

Inferring Rudimentary Rules

- 1R Classifier learns a simple rule involving one attribute
  - Assumes nominal attributes
  - The rule tests all the values of one particular attribute
- Basic version
  - One branch for each value
  - Each branch assigns most frequent class
  - Attribute performance is measured using the error rate computed as the proportion of instances that don't belong to the majority class of their corresponding branch
  - Choose attribute with lowest error rate
  - "missing" is treated as a separate value

Pseudo-Code for OneRule


```
For each attribute,
For each value of the attribute,
make a rule as follows:
       count how often each class appears
       find the most frequent class
       make the rule assign that class to
       this attribute-value
Calculate the error rate of the rules

Choose the rules with the smallest error rate
```

Evaluating the Weather Attributes

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny => No	2/5	4/14
	Overcast => Yes	0/4	
	Rainy => Yes	2/5	
Temp	Hot => No	2/4	5/14*
	Mild => Yes	2/6	
	Cool => Yes	1/4	
Humidity	High => No	3/7	4/14
	Normal => Yes	1/7	
Windy	False => Yes	2/8	5/14*
	True => No	3/6	

*** indicates a tie**

Contact Lens Data

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
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	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

1R Applied to Contact Lens Data

- IF tear-prod-rate=normal THEN soft
  ELSE IF tear-prod-rate=reduced THEN none

  (17/24 instances correct)

How can we deal with numerical attributes using 1R?

We can discretize them before applying 1R

We can directly use 1R discretization

1R and Numerical Attributes

- Applies simple supervised discretization
- Sort instances according to attribute's values
- Place breakpoints where class changes (majority class)
- This procedure is however very sensitive to noise since one example with an incorrect class label may produce a separate interval. This is likely to lead to overfitting.
- In the case of the temperature,

```
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
```

- To limit overfitting, enforce minimum number of instances in majority class per interval.
- For instance, in the case of the temperature, if we set the minimum number of majority class instances to 3, we have

```
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
```

join the intervals to get at least 3 examples

```
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
```

join the intervals with the same majority class

```
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
```

1R Applied to the Numerical Version of the Weather Dataset

Attribute	Rules	Errors	Total errors
Outlook	Sunny => No	2/5	4/14
	Overcast => Yes	0/4	
	Rainy => Yes	2/5	
Temperature	≤ 77.5 => Yes	3/10	5/14
	> 77.5 => No*	2/4	
Humidity	≤ 82.5 => Yes	1/7	3/14
	> 82.5 and ≤ 95.5 => No	2/6	
	> 95.5 => Yes	0/1	
Windy	False => Yes	2/8	5/14
	True => No*	3/6	

Discussion of 1R

- 1R was described in a paper by Holte (1993) "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets"
- Contains an experimental evaluation on 16 datasets (using cross-validation to estimate classification accuracy on fresh data)
- Required minimum number of instances in majority class was set to 6 after some experimentation
- 1R's simple rules performed not much worse than much more complex decision trees
- The takehome message is simplicity first can pay off on practical datasets
- Note that 1R does not perform as well on more recent, more sophisticated benchmark datasets

Sequential Covering

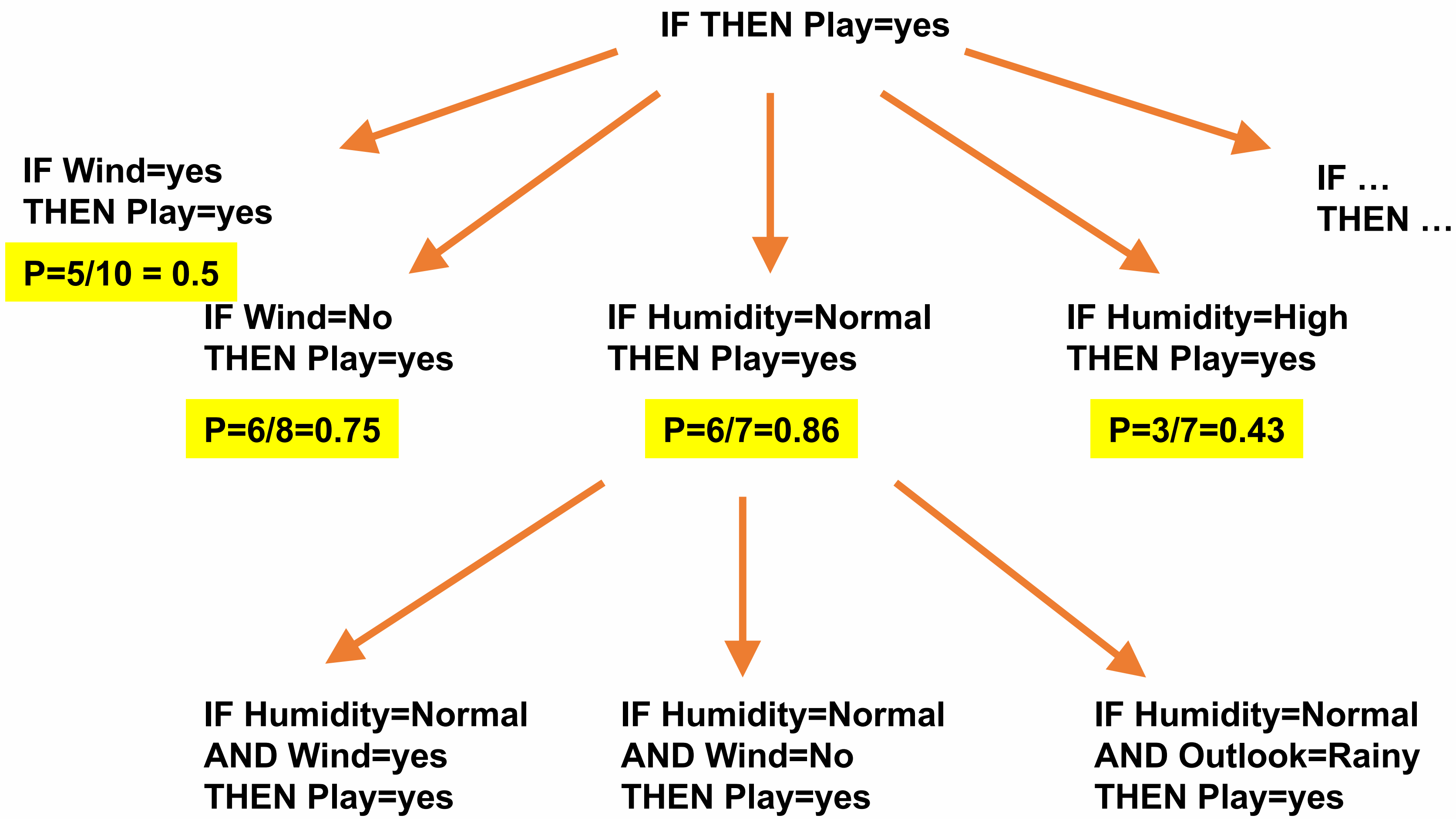
Sequential Covering Algorithms

- Consider the set E of positive and negative examples
- Repeat
  - Learn one rule with high accuracy, any coverage
  - Remove positive examples covered by this rule
- Until all the examples are covered

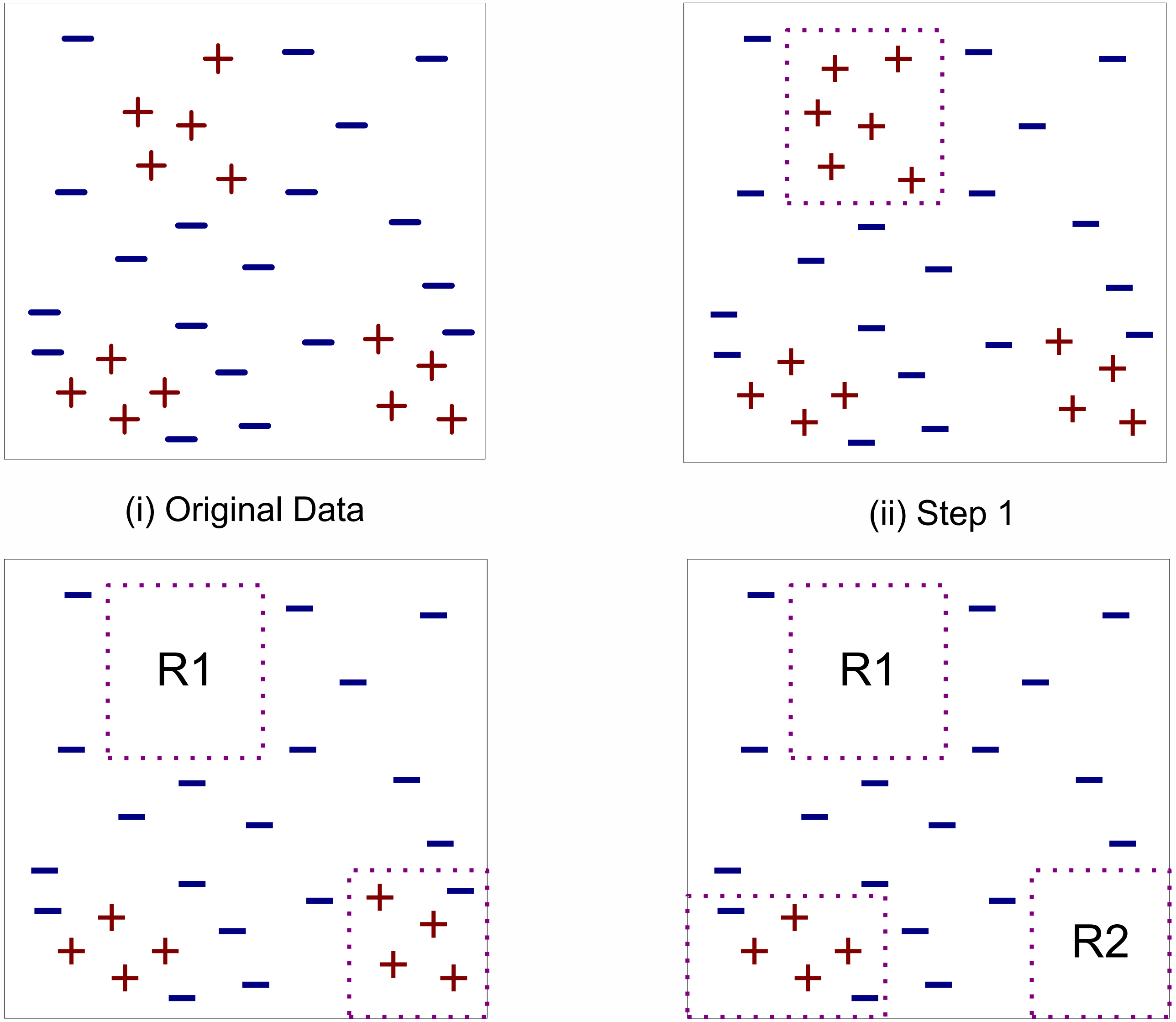
Basic Sequential Covering Algorithm

```
procedure Covering (Examples, Classifier)
input: a set of positive and negative examples for class c

// rule set is initially empty
classifier = {}

while PositiveExamples(Examples)!={}
  // find the best rule possible
  Rule = FindBestRule(Examples)
  // check if we need more rules
  if Stop(Examples, Rule, Classifier) breakwhile
  // remove covered examples and update the model
  Examples = Examples\Cover(Rule,Examples)
  Classifier = Classifier U {Rule}
Endwhile
// post-process the rules (sort them, simplify them, etc.)
Classifier = PostProcessing(Classifier)
output: Classifier
```

Finding the Best Rule Possible



And Another Viewpoint



(i) Original Data

(ii) Step 1

(iii) Step 2

(iv) Step 3

```
LearnOneRule(Attributes, Examples, k)
   init BH to the most general hypothesis
   init CH to {BH}

   while CH not empty Do
      Generate Next More Specific CH in NCH
      // check all the NCH for an hypothesis that
      // improves the performance of BH
      Update BH
      Update CH with the k best NCH
   endwhile
   return a rule “IF BH THEN prediction”
```

An Example Using Contact Lens Data

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
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	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

- Rule we seek:

```
If ?
    then recommendation = hard
```

- Possible tests:

```
Age = Young                                  2/8
Age = Pre-presbyopic                         1/8
Age = Presbyopic                             1/8
Spectacle prescription = Myope               3/12
Spectacle prescription = Hypermetrope        1/12
Astigmatism = no                             0/12
Astigmatism = yes                            4/12
Tear production rate = Reduced               0/12
Tear production rate = Normal                4/12
```

Adding the First Clause

- Rule with best test added,

```
If astigmatism = yes
     then recommendation = hard
```

- Instances covered by modified rule,

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

- Current state,

```
If astigmatism = yes
    and ?
  then recommendation = hard
```

- Possible tests,

Age = Young	2/4
Age = Pre-presbyopic	1/4
Age = Presbyopic	1/4
Spectacle prescription = Myope	3/6
Spectacle prescription = Hypermetrope	1/6
Tear production rate = Reduced	0/6
Tear production rate = Normal	4/6

The Second Rule

- Rule with best test added:

```
If astigmatism = yes
    and tear production rate = normal
then recommendation = Hard
```

- Instances covered by modified rule

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	Hard
Prepresbyopic	Myope	Yes	Normal	Hard
Prepresbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Adding the Third Clause

- Current state:

```
If astigmatism = yes
     and tear production rate = normal
     and ?
   then recommendation = hard
```

- Possible tests:

```
Age = Young                                2/2
Age = Pre-presbyopic                       1/2
Age = Presbyopic                           1/2
Spectacle prescription = Myope             3/3
Spectacle prescription = Hypermetrope      1/3
```

- Tie between the first and the fourth test, we choose the one with greater coverage

The Final Result

- Final rule:

```
If astigmatism = yes
   and tear production rate = normal
   and spectacle prescription = myope
   then recommendation = hard
```

- Second rule for recommending “hard lenses”:
(built from instances not covered by first rule)

```
If age = young and astigmatism = yes
   and tear production rate = normal
   then recommendation = hard
```

- These two rules cover all “hard lenses”:
- Process is repeated with other two classes

Pseudo-code for PRISM

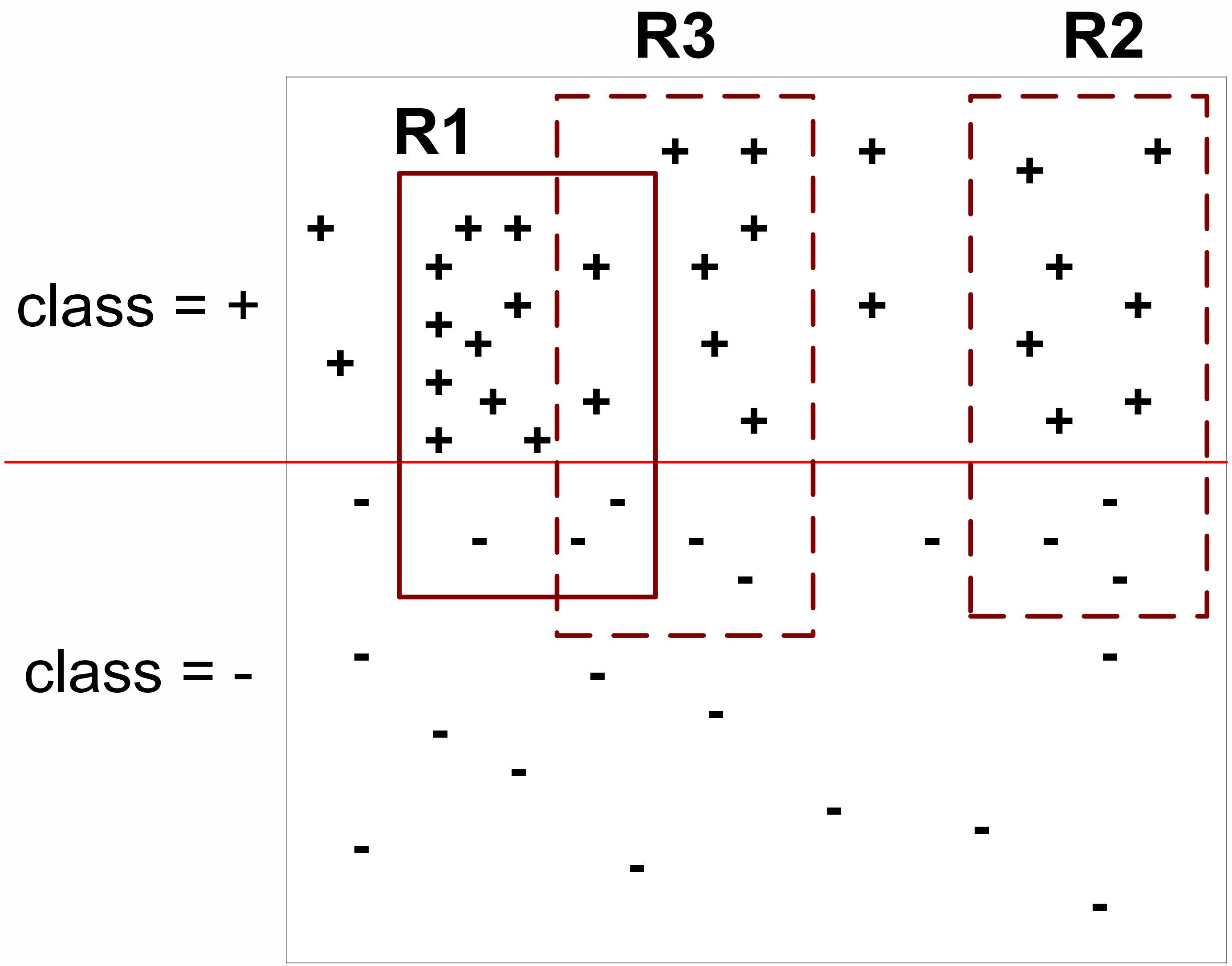

```
For each class C
  Initialize E to the instance set
  While E contains instances in class C
    Create a rule R with an empty left-hand side that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value v,
        Consider adding the condition A = v to the left-hand side of R
        Select A and v to maximize the accuracy p/t
          (break ties by choosing the condition with the largest p)
      Add A = v to R
    Remove the instances covered by R from E
```

Testing for the Best Rule

- Measure 1: Accuracy (p/t)
  - t total instances covered by rule
    pnumber of these that are positive
  - Produce rules that do not cover negative instances, as quickly as possible
  - May produce rules with very small coverage —special cases or noise?
- Measure 2: Information gain p (log(p/t) – log(P/T))
  - P and T the positive and total numbers before the new condition was added
  - Information gain emphasizes positive rather than negative instances
- These measures interact with the pruning mechanism used

- Why do we need to eliminate instances?
  - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
  - To ensure that the next rule is different
  - Prevent overestimating the accuracy of rule
  - So that we have a more robust estimate of accuracy (why?)
- Why do we remove negative instances?
  - Prevent underestimating the accuracy of rule
  - Compare rules R2 and R3 in the following diagram

Eliminating Instances



Missing Values and Numeric Attributes

- Missing values usually fail the test

- Covering algorithm must either
  - Use other tests to separate out positive instances
  - Leave them uncovered until later in the process

- In some cases it is better to treat "missing" as a separate value (i.e., if "missing" has a special significance")

- Numeric attributes are treated just like they are in decision trees, with binary split points

- Split points are found by optimizing test selection criterion, similar to what happens when finding a split in decision trees

Stopping Criterion and Rule Pruning

- The process usually stops when there is no significant improvement by adding the new rule
- Rule pruning is similar to post-pruning of decision trees
- Reduced Error Pruning:
  - Remove one of the conjuncts in the rule
  - Compare error rate on validation set
  - If error improves, prune the conjunct

Pruning rules

- Two main strategies
  - Incremental pruning
  - Global pruning
- Pruning criterion
  - Error on hold-out set (reduced-error pruning)
  - Statistical significance
  - MDL principle

Using a Pruning Set

- For statistical validity, must evaluate measure on data not used for growing the tree:
  - This requires a growing set and a pruning set
  - The full training set is split, randomly, into these two sets
- Reduced-error pruning: build full rule set on growing set and then prune it
- Incremental reduced-error pruning: simplify each rule as soon as it has been built
  - Can re-split data after rule has been pruned
- Stratification is advantageous when applying reduced-error pruning, so that class proportions are preserved

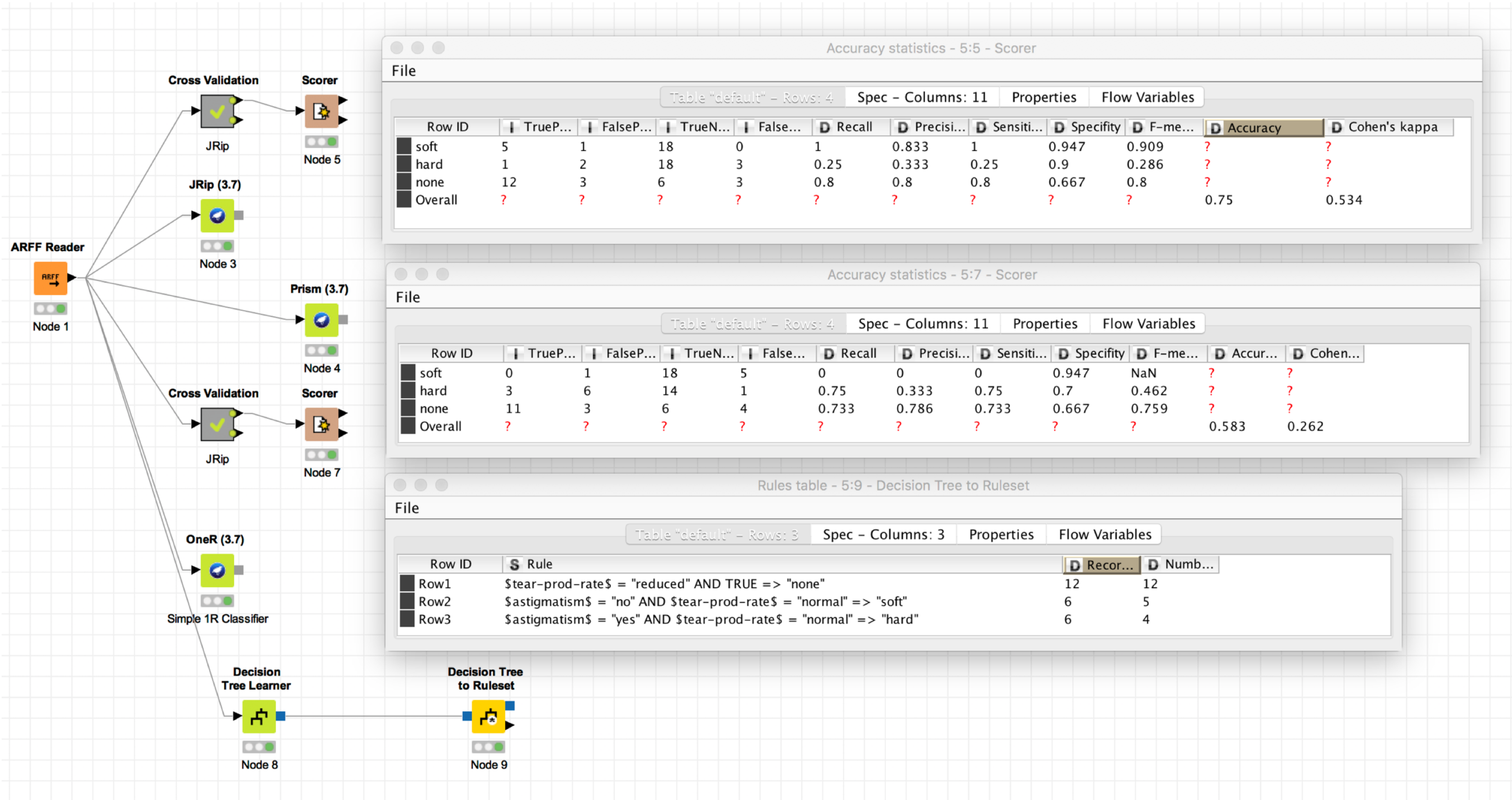
run the KNIME workflows on decision rules

Indirect Methods

Rules vs. Trees

- Rule sets can be more readable
- Decision trees suffer from replicated subtrees
- Rule sets are collections of local models, trees represent models over the whole domain
- The covering algorithm concentrates on one class at a time whereas decision tree learner takes all classes into account

run the KNIME workflows on decision rules



Row ID	TrueP...	FalseP...	TrueN...	False...	Recall	Precisi...	Sensiti...	Specifity	F-me...	Accuracy	Cohen's kappa
soft	5	1	18	0	1	0.833	1	0.947	0.909	?	?
hard	1	2	18	3	0.25	0.333	0.25	0.9	0.286	?	?
none	12	3	6	3	0.8	0.8	0.8	0.667	0.8	?	?
Overall	?	?	?	?	?	?	?	?	?	0.75	0.534

Row ID	TrueP...	FalseP...	TrueN...	False...	Recall	Precisi...	Sensiti...	Specifity	F-me...	Accur...	Cohen...
soft	0	1	18	5	0	0	0	0.947	NaN	?	?
hard	3	6	14	1	0.75	0.333	0.75	0.7	0.462	?	?
none	11	3	6	4	0.733	0.786	0.733	0.667	0.759	?	?
Overall	?	?	?	?	?	?	?	?	?	0.583	0.262

Row ID	Rule	Recor...	Numb...
Row1	$tear-prod-rate$ = "reduced" AND TRUE => "none"	12	12
Row2	$astigmatism$ = "no" AND $tear-prod-rate$ = "normal" => "soft"	6	5
Row3	$astigmatism$ = "yes" AND $tear-prod-rate$ = "normal" => "hard"	6	4

Summary

Summary

- Advantages of Rule-Based Classifiers
  - As highly expressive as decision trees
  - Easy to interpret
  - Easy to generate
  - Can classify new instances rapidly
  - Performance comparable to decision trees

- Two approaches: direct and indirect methods

Summary

- Direct Methods, typically apply sequential covering approach
  - Grow a single rule
  - Remove Instances from rule
  - Prune the rule (if necessary)
  - Add rule to Current Rule Set
  - Repeat
- Other approaches exist
  - Specific to general exploration (RISE)
  - Post processing of neural networks, association rules, decision trees, etc.

Homework

- Generate the rule set for the Weather dataset by repeatedly applying the procedure to learn one rule until no improvement can be produced or the covered examples are too few

- Check the problems provided in the previous exams and apply both OneRule and Sequential Covering to generate the first rule. Then, check the result with one of the implementations available in Weka

- Apply one or more sequential covering algorithms using the KNIME workflow. Compute the usual performance statistics.