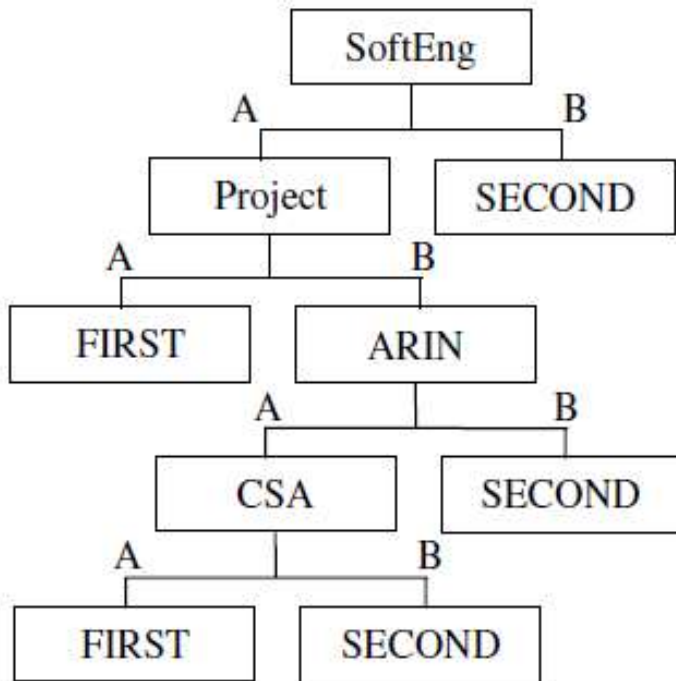

Computer Science 477

Inducing Modular Rules for
Classification

Lecture 16

Conversion to Rules



- IF SoftEng = A AND Project = B AND ARIN = A AND CSA = A THEN Class = FIRST
- IF SoftEng = A AND Project = A THEN Class = FIRST
- IF SoftEng = A AND Project = B AND ARIN = A AND CSA = B THEN Class = SECOND
- IF SoftEng = A AND Project = B AND ARIN = B THEN Class = SECOND
- IF SoftEng = B THEN Class = SECOND

Rule Post-pruning

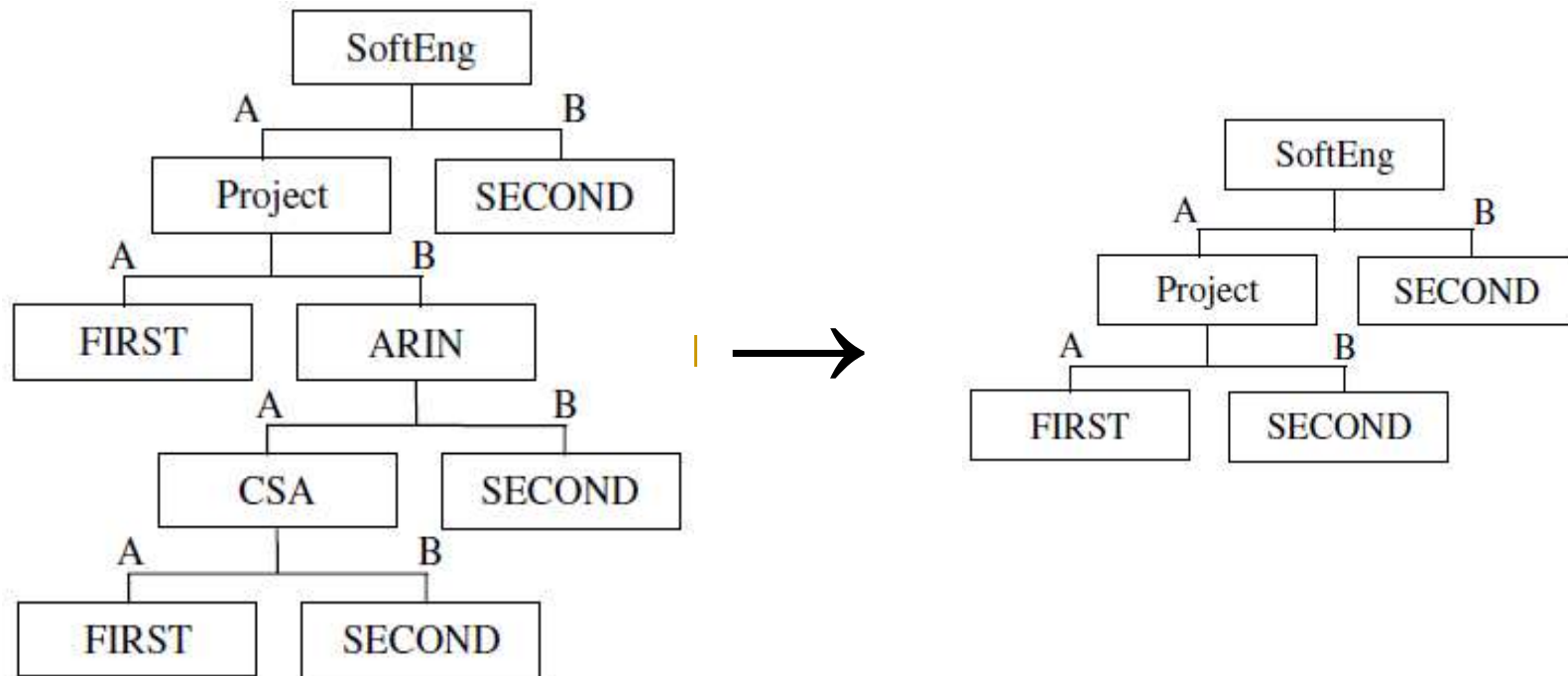
- Convert a decision tree to an equivalent set of rules
- Examine rule to simplify them
 - Without any loss of (preferably with a gain in) predictive accuracy.
- For rule
 - IF SoftEng = A AND Project = B AND ARIN = A AND CSA = A THEN Class = FIRST
- Consider the four terms
 - 'SoftEng = A', 'Project = B', 'ARIN = A' and 'CSA = A'.

Term Elimination

- Need some criterion for whether removing a term from a ruleset increases or decreases classification accuracy.
- Suppose we have one.
- The term elimination process terminates when removing a term from a rule decreases classification accuracy.
- Needs *three* datasets
 - Training, Pruning and Test
- Otherwise leads to overfitting

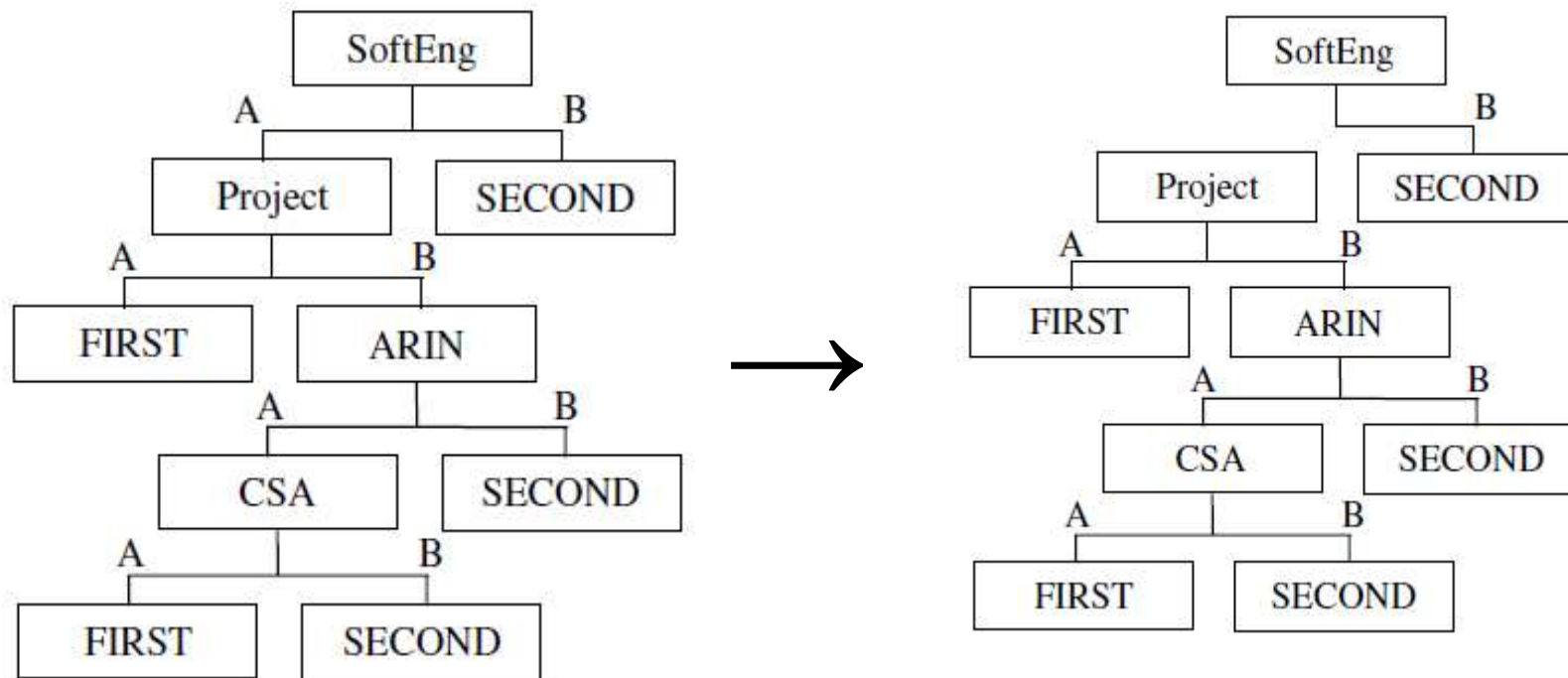
Conflict Resolution

- Bottom-up pruning method in Chapter 8 has virtue:
 - Resulting structure still a decision tree.
- Might lead to test on ARIN with the single node



Non-bottom-up Pruning

- Suppose in rule pruning, remove the link corresponding to 'SoftEng' = A
- No longer a tree
 - Two disconnected trees



Rule Reduction

- IF SoftEng = A AND Project = B AND ARIN = A AND CSA = A THEN Class = FIRST
 - IF SoftEng = A AND Project = A THEN Class = FIRST
 - IF SoftEng = A AND Project = B AND ARIN = A AND CSA = B THEN Class = SECOND
 - IF SoftEng = A AND Project = B AND ARIN = B THEN Class = SECOND
 - IF SoftEng = B THEN Class = SECOND
- First four rules have changed.
 - IF Project = B AND ARIN = A AND CSA = A THEN Class = FIRST
 - IF Project = A THEN Class = FIRST
 - IF Project = B AND ARIN = A AND CSA = B THEN Class = SECOND
 - IF Project = B AND ARIN = B THEN Class = SECOND
 - IF SoftEng = B THEN Class = SECOND

Consequence

- Rule *fires* if its condition part is satisfied for a given instance.
- A set of rules derived directly from a tree structure
 - Only one rule that can fire for any instance
- Suppose an instance:
 - SoftEng=B, Project=A, ARIN=A and CSA=A,
- Both rules fire:
 - IF Project = B AND ARIN = A AND CSA = A THEN
Class = FIRST AND CSA = B THEN Class =
SECOND
 - IF Project = B AND ARIN = B THEN Class = SECOND

Another Example – Another Dataset

- IF $x = 4$ THEN Class = a
 - IF $y = 2$ THEN Class = b
 - What if, for an instance, $x=4$, $y=2$
 - More elaborately:
 - IF $w = 9$ and $k = 5$ THEN Class = b
 - IF $x = 4$ THEN Class = a
 - IF $y = 2$ THEN Class = b
 - IF $z = 6$ and $m = 47$ THEN Class = b
 - Instance:
 - $w = 9$, $k = 5$, $x = 4$, $y = 2$, $z = 6$ and $m = 47$
 - One rule gives class a
 - The other three give b
-

Strategies for Conflict Resolution

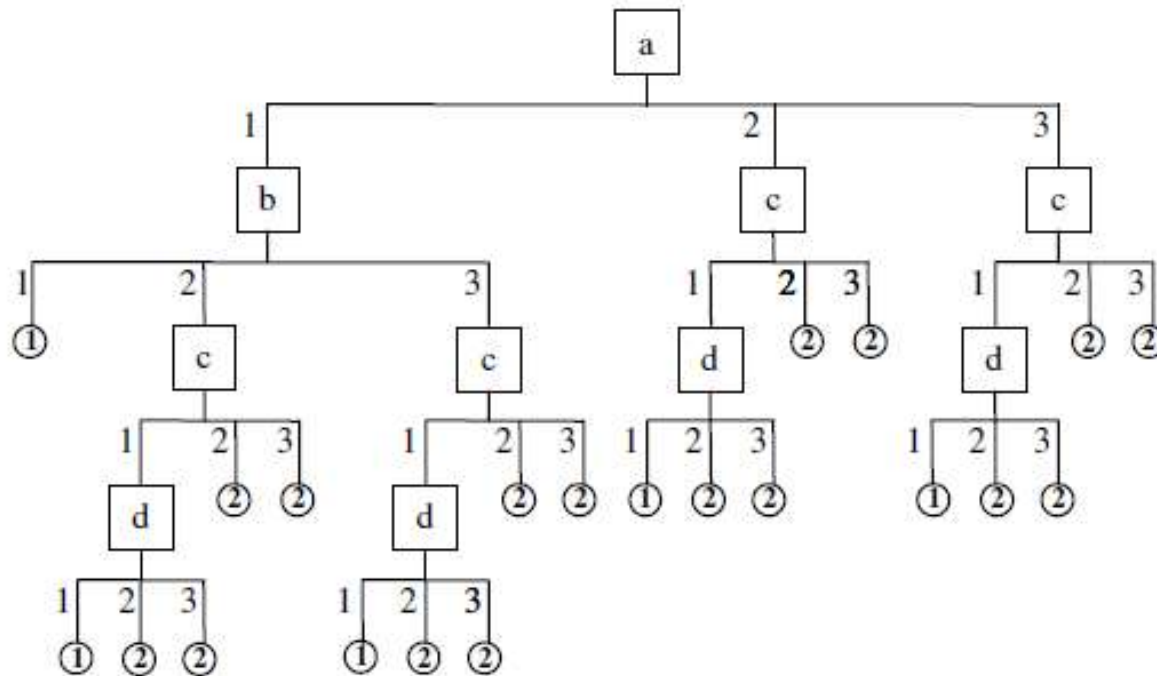
- 'Majority voting' (e.g. there are three rules predicting class *b* and only one predicting class *a*, so choose class *b*)
- Giving priority to certain types of rule or classification (e.g. rules with a small number of terms or predicting a rare classification might have a higher weighting than other rules in the voting)
- Using a measure of the 'interestingness' of each rule give priority to the most interesting rule.
 - Of which more anon
- Most conflict resolution strategies require the condition that *all* the rules to be tested for each unseen instance, so that all the rules that fire are known before the strategy is applied.
- With trees one need only work through the rules generated from a decision tree until the first one fires (as we know no others can).
- Possibly dispense with decision tree generation, proceed directly to rules.

Problems with Decision Trees

- Rule 1: IF $a = 1$ AND $b = 1$ THEN Class = 1
 - Rule 2: IF $c = 1$ AND $d = 1$ THEN Class = 1
 - Suppose that Rules 1 and 2 cover all instances of Class 1 and all other instances are of Class 2.
 - These two rules cannot be represented by a single decision tree
 - The root node of the tree must split on a single attribute, and
 - No attribute which is common to both rules.
 - Simplest decision tree necessarily adds an extra term to one of the rules
 - Would in turn require at least one extra rule to cover instances excluded by the addition of that extra term.
-

Converting to a decision tree

- IF $a = 1$ AND $b = 1$ THEN Class = 1
- IF $a = 1$ AND $b = 2$ AND $c = 1$ AND $d = 1$ THEN Class = 1
- IF $a = 1$ AND $b = 3$ AND $c = 1$ AND $d = 1$ THEN Class = 1
- IF $a = 2$ AND $c = 1$ AND $d = 1$ THEN Class = 1



Problems with Decision Trees

- Trees branch on a single attribute
- Can lead to overfitting
- Demands that values of all attributes known for all instances
- Medical context
 - Require tests high in cost
 - Or risk
- Induce rules directly from data
- Not mediated by decision tree

Prism Algorithm

- The aim is to induce modular classification rules directly from the training set.
 - Assumes that all the attributes are categorical.
 - When there are continuous attributes they can first be converted to categorical ones
 - Algorithm can be extended to deal with continuous attributes in much the same way as was described for TDIDT
 - Generates the rules concluding each of the possible classes in turn.
 - Each rule is generated term by term, with each term of the form 'attribute = value'.
 - The attribute/value pair added at each step is chosen to maximize the probability of the target 'outcome class'.
-

Prism Algorithm

- For each classification (class = i) in turn and starting with the complete training set each time:
 - 1. Calculate the probability that class = i for each attribute/value pair.
 - 2. Select the pair with the largest probability and create a subset of the training set comprising all the instances with the selected attribute/value combination (for all classifications).
 - 3. Repeat 1 and 2 for this subset until a subset is reached that contains only instances of class i .
- The induced rule is then the conjunction of all the attribute/value pairs selected.
- 4. Remove all instances covered by this rule from the training set.
- *Repeat 1–4 until all instances of class i have been removed*

Lens24 Dataset

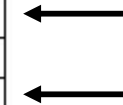
- Comprises 24 instances

age	specRx	astig	tears	class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

Calculate probability: class = i for attribute/value pairs

- Probabilities for class 1:

Attribute/value pair	Frequency for class = 1	Total frequency (out of 24 instances)	Probability
age = 1	2	8	0.25
age = 2	1	8	0.125
age = 3	1	8	0.125
specRx = 1	3	12	0.25
specRx = 2	1	12	0.083
astig = 1	0	12	0
astig = 2	4	12	0.33
tears = 1	0	12	0
tears = 2	4	12	0.33



- Maximum probability
 - astig = 2 or tears = 2
- Choose astig = 2
 - Arbitrarily
- Rule thus far:
 - IF astig = 2 ... then class = 1

Subsetting the Dataset

- All the instances where $astig = 2$


age	specRx	astig	tears	class
1	1	2	1	3
1	1	2	2	1
1	2	2	1	3
1	2	2	2	1
2	1	2	1	3
2	1	2	2	1
2	2	2	1	3
2	2	2	2	3
3	1	2	1	3
3	1	2	2	1
3	2	2	1	3
3	2	2	2	3

- Subset covered by incomplete rule
 - $astig = 2$

Attribute/value pairs for subset

- Calculate probabilities for subset
 - Not involving astig

Attribute/value pair	Frequency for class = 1	Total frequency (out of 12 instances)	Probability
age = 1	2	4	0.5
age = 2	1	4	0.25
age = 3	1	4	0.25
specRx = 1	3	6	0.5
specRx = 2	1	6	0.17
tears = 1	0	6	0
tears = 2	4	6	0.67

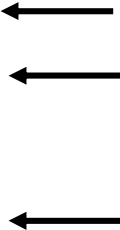


- Rule thus far
 - IF astig = 2 and tears = 2 THEN class = 1

Subset of Training Set Covered

- Items covered by rule:

age	specRx	astig	tears	class
1	1	2	2	1
1	2	2	2	1
2	1	2	2	1
2	2	2	2	3
3	1	2	2	1
3	2	2	2	3



- Every instance where $\text{astig} = 2$ and $\text{tears} = 2$
 - Class 1
- Remove these instances only from the training set

Calculate Probabilities

- Remaining classes are age and specRx
- Frequency of attribute value pairs for class = 1

Attribute/value pair	Frequency for class = 1	Total frequency (out of 6 instances)	Probability
age = 1	2	2	1.0
age = 2	1	2	0.5
age = 3	1	2	0.5
specRx = 1	3	3	1.0
specRx = 2	1	3	0.33



- Incomplete rule thus far:
 - IF astig = 2 and tears = 2 and age = 1 THEN class = 1

Check Incomplete Rule

- Subset covered by incomplete rule:

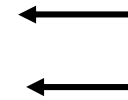
age	specRx	astig	tears	class
1	1	2	2	1
1	2	2	2	1

- All instances are class one
 - So rule is final

Continuing

- Remove two instances covered by first final rule
 - Leaves 22 instances
- Attribute/value probabilities for class 1

Attribute/value pair	Frequency for class = 1	Total frequency (out of 22 instances)	Probability
age = 1	0	6	0
age = 2	1	8	0.125
age = 3	1	8	0.125
specRx = 1	2	11	0.18
specRx = 2	0	11	0
astig = 1	0	12	0
astig = 2	2	10	0.2
tears = 1	0	12	0
tears = 2	2	10	0.2



- Maximum when astig = 2 and tears = 2
- Chose astig = 2
 - Arbitrarily

Elaborate rule

- Incomplete rule
 - IF astig = 2 THEN class = 1
- Subset of training set covered:

age	specRx	astig	tears	class
1	1	2	1	3
1	2	2	1	3
2	1	2	1	3
2	1	2	2	1
2	2	2	1	3
2	2	2	2	3
3	1	2	1	3
3	1	2	2	1
3	2	2	1	3
3	2	2	2	3

In-class Exercise

- Current training set:

age	specRx	astig	tears	class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

Modifications of Prism Algorithm

- In case of ties
- The basic algorithm can be improved slightly by choosing between attribute/value pairs which have equal probability not arbitrarily as above
- Taking the one with the highest total frequency

Clashes in Training Data

- Clashes in Training Data
- Step 3 or original algorithm
 - Repeat 1 and 2 for this subset until a subset is reached that contains only instances of class i .
- Add: 'or a subset is reached which contains instances of more than one class, although values of all the attributes have already been used in creating the subset'
- Majority class does not fit directly into the Prism framework.

Clashes

- If a clash occurs while generating the rules for class i :
 - 1. Determine the majority class for the subset of instances in the clash set.
 - 2. If this majority class is class i , then complete the induced rule by assigning all the instances in the clash set to class i .
- If not, discard the rule.

Resolving Clashes

- Where you have mixed leaf but no more attributes:
- If a clash occurs while generating the rules for class i :
 - 1. Determine the majority class for the subset of instances in the clash set.
 - 2(a). If this majority class is class i , then complete the induced rule by assigning all the instances in the clash set to class i .
 - 2(b). If not, discard the rule.

Comparing Prism with TDIDT

- “The experiments presented here suggest that the Prism algorithm for generating modular rules gives classification rules which are at least as good as those obtained from the widely used TDIDT algorithm.”
- Generally fewer rules with fewer terms per rule, which is likely to aid their comprehensibility to domain experts and users.
- This result would seem to apply even more strongly when there is noise in the training set.
- Classification accuracy on unseen test data - there appears to be little to choose between the two algorithms for noise-free datasets, including ones with a significant proportion of clash instances in the training set.
- The main difference is that Prism generally has a preference for leaving a test instance as ‘unclassified’ rather than giving it a wrong classification.

Rules

- What about rules of the form?
- If $\text{att1} = a$ or $\text{att2} = b$, then $\text{class} = l$
- Short exercise by the end of the day.