Classification Algorithms – Rules

Outline

- Generating Rules
- The Covering Algorithm
- Rules VS Decision Lists

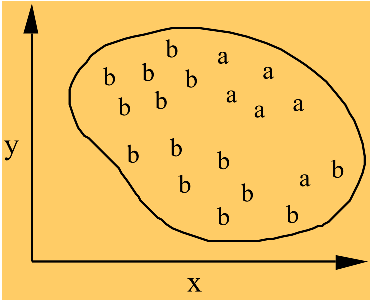
Generating Rules

- Decision tree can be converted into a rule set
- Straightforward conversion:
  - each path to the leaf becomes a rule – makes an overly complex rule set
- More effective conversions are not trivial
  - (e.g. C4.5 tests each node in root-leaf path to see if it can be eliminated without loss in accuracy)

Covering algorithms

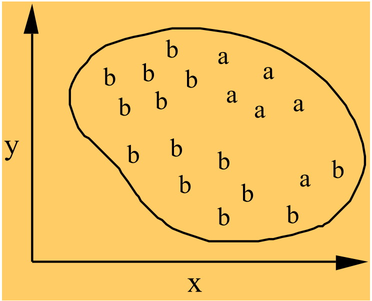
- Strategy for generating a rule set directly: for each class in turn find rule set that covers all instances in it (excluding instances not in the class)
- This approach is called the ***covering*** approach because at each stage a rule is identified that covers some of the instances

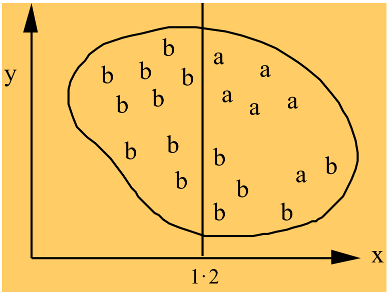
Example: generating a rule



If true then class = a

Example: generating a rule, II

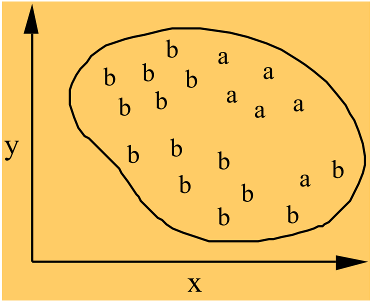


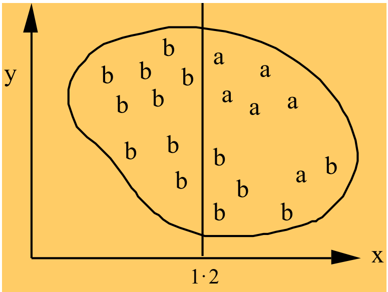


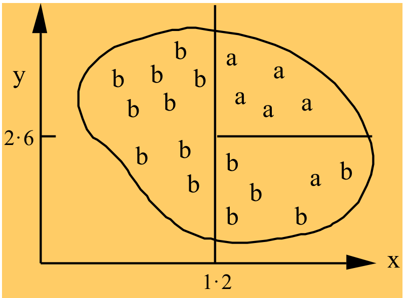
If true then class = a

If x > 1.2 then class = a

Example: generating a rule, III





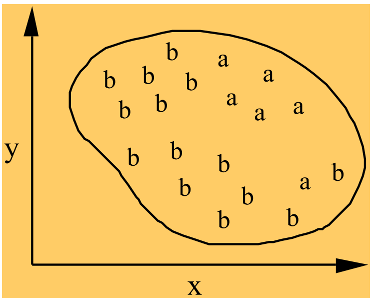


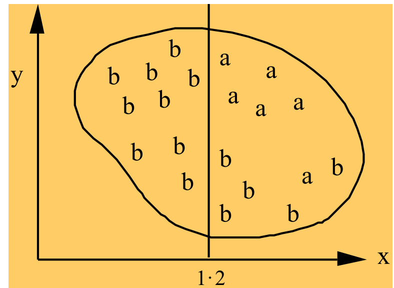
If true then class = a

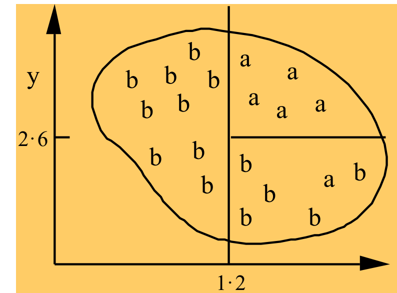
If $x > 1.2$ and $y > 2.6$ then class = a

If $x > 1.2$ then class = a

Example: generating a rule, IV







If true then class = a

If x > 1.2 and y > 2.6 then class = a

If x > 1.2 then class = a

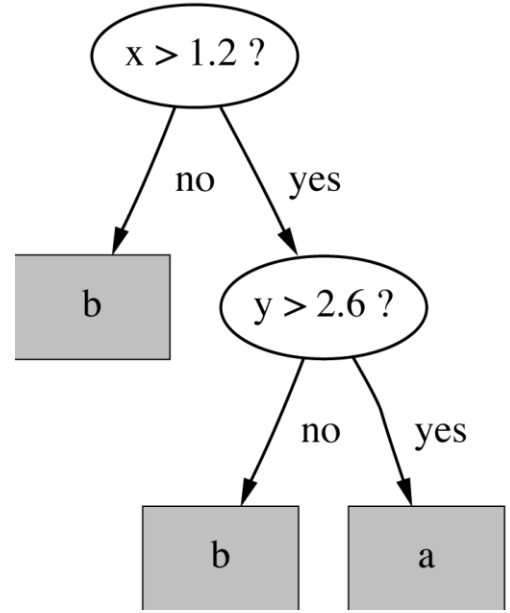
- Possible rule set for class "b":

  If x ≤ 1.2 then class = b

  If x > 1.2 and y ≤ 2.6 then class = b

- More rules could be added for "perfect" rule set

Rules vs. trees

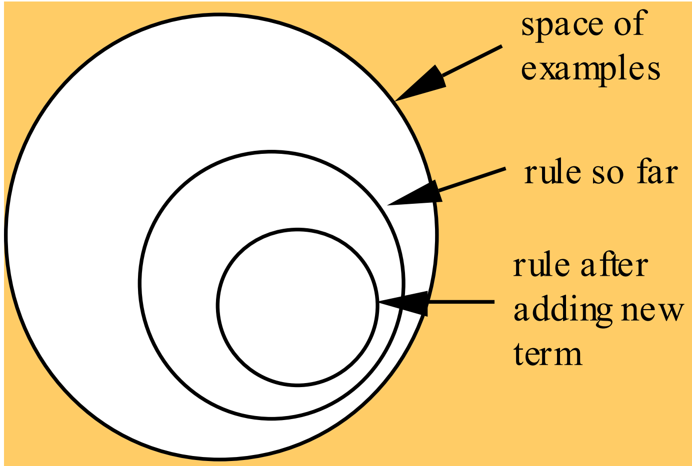


- Corresponding decision tree:
  (produces exactly the same
  predictions)
- But: rule sets *can* be more clear when decision trees suffer from replicated subtrees
- Also: in multi-class situations, covering algorithm concentrates on one class at a time whereas decision tree learner takes all classes into account

A simple covering algorithm

- Generates a rule by adding tests that maximize rule's accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
  - But: decision tree inducer maximizes overall purity
- Each new test reduces

  rule's coverage:



Selecting a test

- Goal: maximize accuracy
  - t total number of instances covered by rule
  - p positive examples of the class covered by rule
  - $t - p$ number of errors made by rule
  - $\Rightarrow$ Select test that maximizes the ratio p/t
- We are finished when $p/t = 1$ or the set of instances can't be split any further

Example: contact lens data, 1

- Rule we seek:

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

- Possible tests:

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

Example: contact lens data, 2

- 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

Modified rule and resulting data

- 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

Further refinement, 1

- Current state:

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

- Possible tests:

```
Age = Young                                      2/4
Age = Pre-presbyopic
Age = Presbyopic
Spectacle prescription = Myope
Spectacle prescription = Hypermetrope
Tear production rate = Reduced
Tear production rate = Normal
```

Further refinement, 2

- 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

Modified rule and resulting data

- 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
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Further refinement, 3

- Current state:

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

- Possible tests:

```
Age = Young
Age = Pre-presbyopic
Age = Presbyopic
Spectacle prescription = Myope
Spectacle prescription = Hypermetrope
```

Further refinement, 4

- 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 result

- Final rule:

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

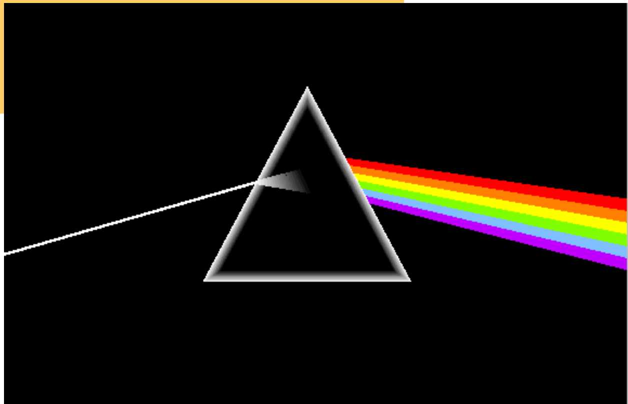
- Second rule for recommending "hard lenses":
  (built from instances not covered by first rule)

```
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
```



Rules vs. decision lists

- PRISM with outer loop removed generates a decision list for one class
  - Subsequent rules are designed for rules that are not covered by previous rules
  - But: order doesn't matter because all rules predict the same class
- Outer loop considers all classes separately
  - No order dependence implied
- Problems: overlapping rules, default rule required

Separate and conquer

- Methods like PRISM (for dealing with one class) are *separate-and-conquer* algorithms:
  - First, a rule is identified
  - Then, all instances covered by the rule are separated out
  - Finally, the remaining instances are "conquered"
- Difference to divide-and-conquer methods:
  - Subset covered by rule doesn't need to be explored any further

Conclusions

- Basically, two ways of learning decision rules:
  - from decision trees (straightforward, more effectively)
  - directly from data
- The covering approach
- The PRISM algorithm
- Decision rules vs. Decision lists