Output: knowledge representation

Most of these slides (used with permission) are based on the book:

Data Mining: Practical Machine Learning Tools and Techniques
by I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal

Output: Knowledge representation

- Tables
- Linear models
- Trees
- Rules
- Classification rules
- Association rules
- Rules with exceptions
- More expressive rules
- Instance-based representation
- Clusters

2

Output: representing structural patterns

- Many different ways of representing patterns
  - Decision trees, rules, instance-based, …
- Also called "knowledge" representation
- Representation determines inference method
- Understanding the output is the key to understanding the underlying learning methods
- Different types of output for different learning problems (e.g., classification, regression, …)

3

3

Decision tables

- Simplest way of representing output:
  - Use the format that is used for representing the input!
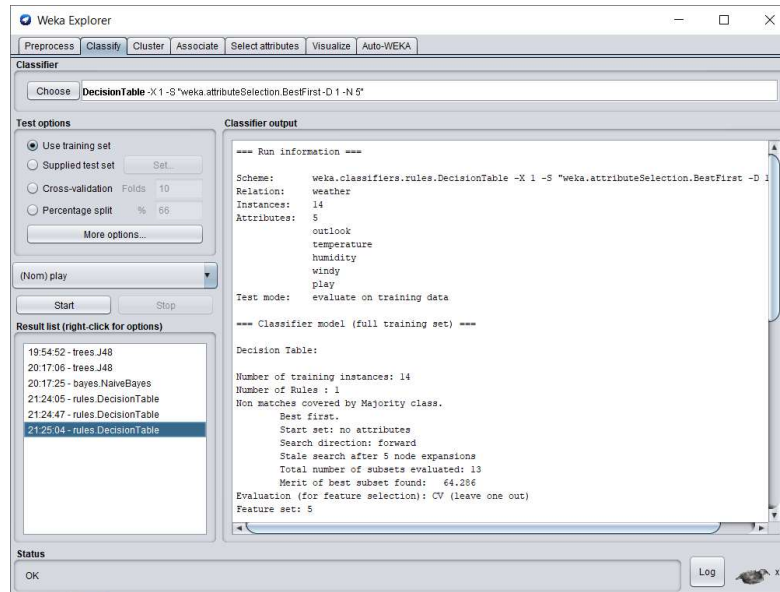- Decision table for the weather problem:

Outlook	Humidity	Play
Sunny	High	No
Sunny	Normal	Yes
Overcast	High	Yes
Overcast	Normal	Yes
Rainy	High	No
Rainy	Normal	No

- Main problem: selecting the right attributes

4

4

# Decision table classifier – uses majority class

Linear models

- Another simple representation
- Traditionally primarily used for regression:
  - Inputs (attribute values) and output are all numeric
- Output is the sum of the weighted input attribute values
- The trick is to find good values for the weights
- There are different ways of doing this, which we will consider later; the most famous one is to minimize the squared error

6

A linear regression function for the CPU performance data

PRP: Performance   CACH: Cache
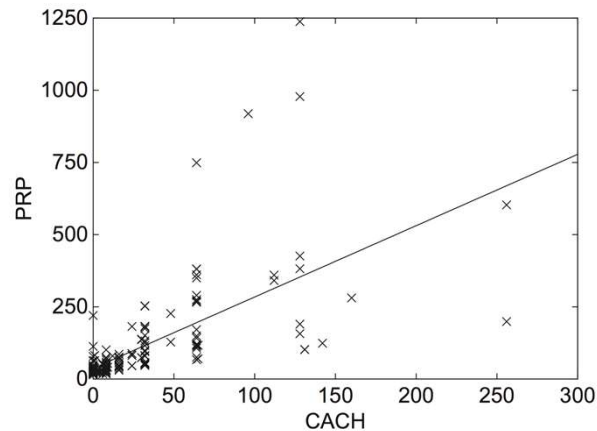

PRP = 37.06 + 2.47CACH

7

Linear models for classification

- Binary classification (Logistic regression)
- Line *separates* the two classes
  - Decision boundary - defines where the decision changes from one class value to the other
- Prediction is made by plugging in observed values of the attributes into the expression
  - Predict one class if output ≥ 0, and the other class if output < 0
- Boundary becomes a high-dimensional plane (*hyperplane*) when there are multiple attributes
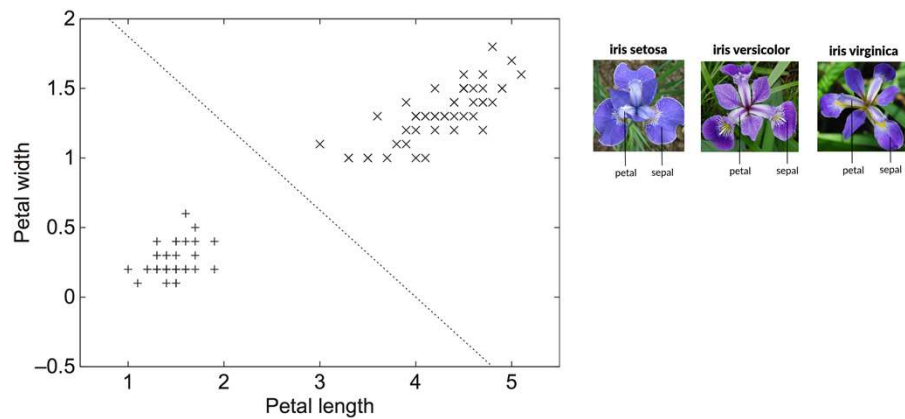
8

Logistic regression for classification
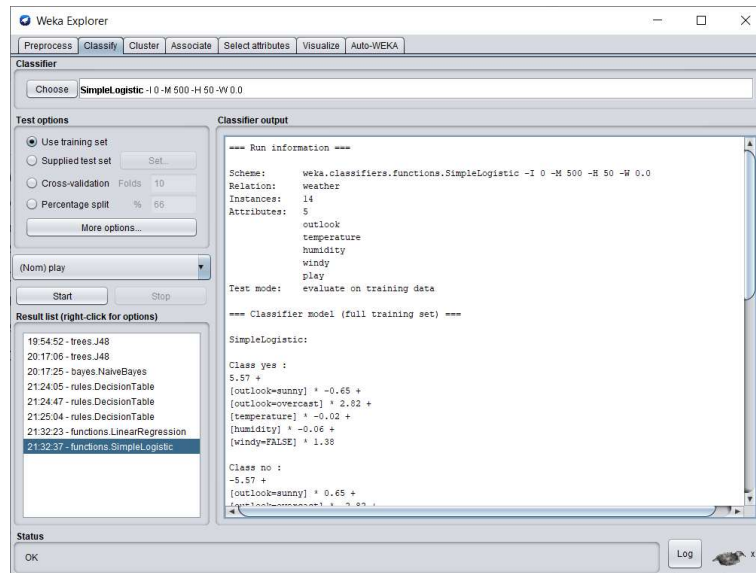
9

Separating setosas from versicolors

$$2.0 - 0.5\text{PETAL-LENGTH} - 0.8\text{PETAL-WIDTH} = 0$$

10

Separating setosas from versicolors – iris2class.arff
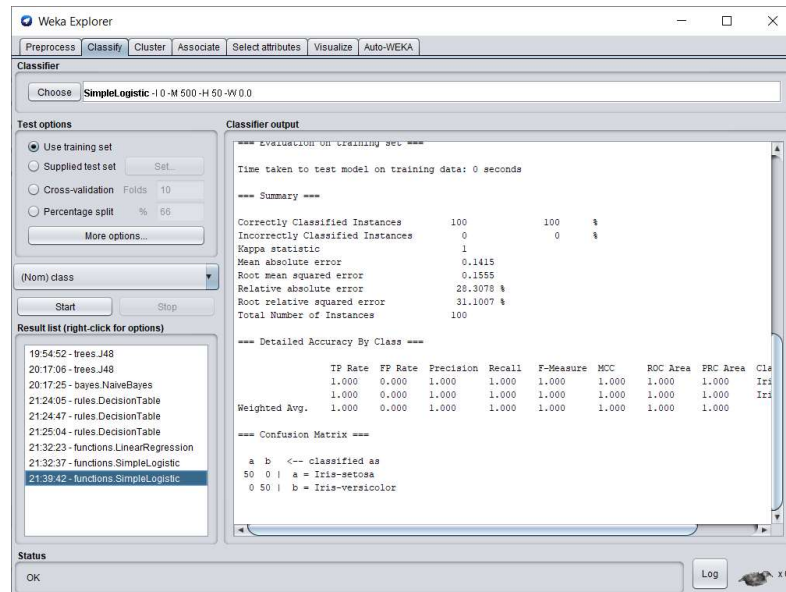
Decision trees

- "Divide-and-conquer" approach produces tree
- Nodes involve testing a particular attribute
- Usually, attribute value is compared to constant
- Other possibilities:
  - Comparing values of two attributes
  - Using a function of one or more attributes
- Leaves assign classification, set of classifications, or probability distribution to instances
- Unknown instance is routed down the tree

Interactive tree construction I
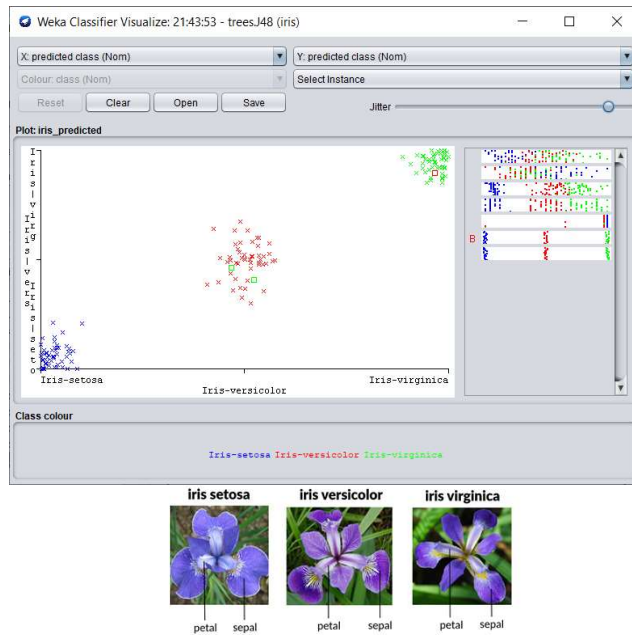
13

Interactive tree construction II
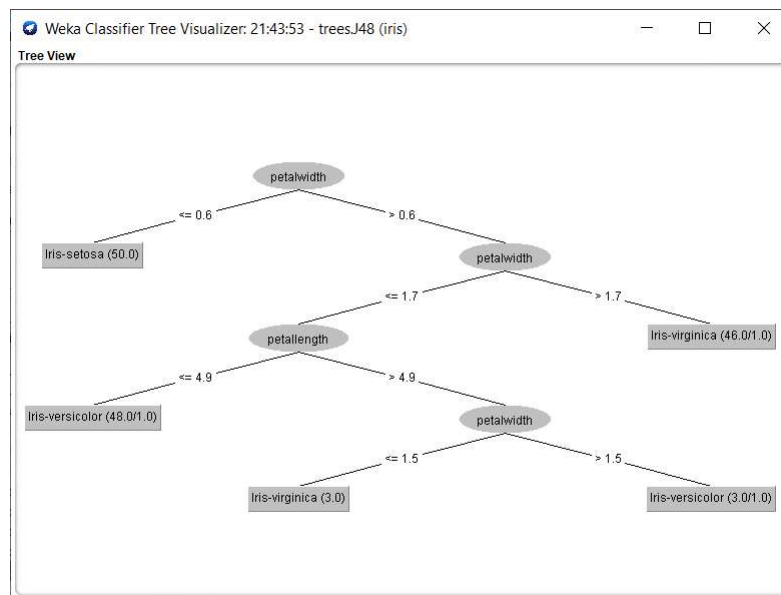
14

Nominal and numeric attributes in trees

- Nominal:
  number of children usually equal to number values
  $\Rightarrow$ attribute won't get tested more than once
- Other possibility: division into two subsets
- Numeric:
  test whether value is greater or less than constant
  $\Rightarrow$ attribute may get tested several times
  - Other possibility: three-way split (or multi-way split)
    - Integer: *less than, equal to, greater than*
    - Real: *below, within, above*

Missing values

- Does absence of value have some significance?
- Yes $\Rightarrow$ "missing" is a separate value
- No $\Rightarrow$ "missing" must be treated in a special way
  - Solution: assign instance to most popular branch
  - Solution: Use Weka filter for filling in missing values (See Weka presentation)

Trees for numeric prediction

- *Regression*: the process of computing an expression that predicts a numeric quantity
- *Regression tree*: "decision tree" where each leaf predicts a numeric quantity
  - Predicted value is average value of training instances that reach the leaf
- *Model tree:* "regression tree" with linear regression models at the leaf nodes

17

Predicting CPU performance

- Example: 209 different computer configurations

	Cycle time (ns)	Main memory (Kb)		Cache (Kb)	Channels		Performance
	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
...							
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

- Linear regression function

```
PRP = –56.1 + 0.049 MYCT + 0.015 MMIN + 0.006 MMAX
      + 0.630 CACH – 0.270 CHMIN + 1.46 CHMAX
```

18

Predicting CPU performance

Weka Explorer — □ ✕

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize | Auto-WEKA

Classifier

Choose | LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4

Test options
- ⦿ Use training set
- ○ Supplied test set   Set...
- ○ Cross-validation Folds 10
- ○ Percentage split   % 66
- More options...

(Num) class ▾

Start   Stop

Result list (right-click for options)

19:54:52 - trees.J48
20:17:06 - trees.J48
20:17:25 - bayes.NaiveBayes
21:24:05 - rules.DecisionTable
21:24:47 - rules.DecisionTable
21:25:04 - rules.DecisionTable
21:32:23 - functions.LinearRegression
21:32:37 - functions.SimpleLogistic
21:39:42 - functions.SimpleLogistic
21:43:09 - functions.SimpleLogistic
21:43:53 - trees.J48
21:48:29 - functions.LinearRegression

Classifier output

```
Scheme:        weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 4
Relation:      cpu
Instances:     209
Attributes:    7
               MYCT
               MMIN
               MMAX
               CACH
               CHMIN
               CHMAX
               class
Test mode:     evaluate on training data

=== Classifier model (full training set) ===


Linear Regression Model

class =

      0.0491 * MYCT +
      0.0152 * MMIN +
      0.0056 * MMAX +
      0.6298 * CACH +
      1.4599 * CHMAX +
    -56.075

Time taken to build model: 0.07 seconds

=== Evaluation on training set ===
```

Status

OK   Log   x 0
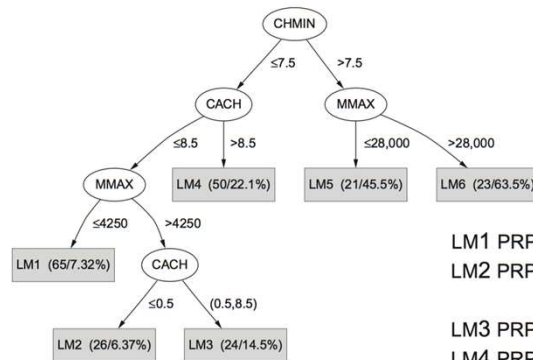
19

Regression tree for the CPU data
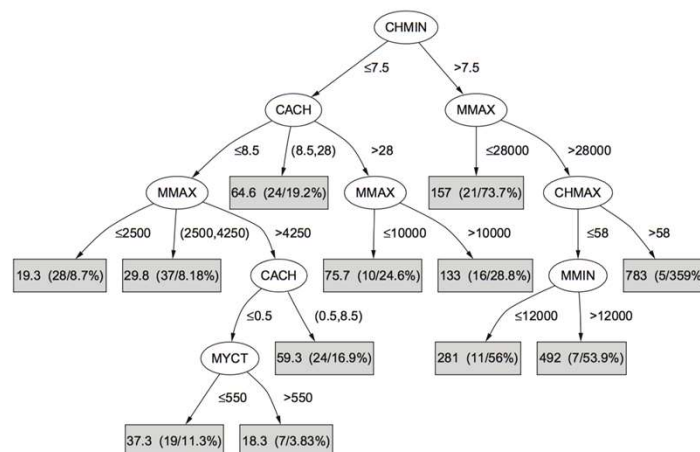
20

Model tree for the CPU data

$$LM1 \; PRP = 8.29 + 0.004 \; MMAX + 2.77 \; CHMIN$$
$$LM2 \; PRP = 20.3 + 0.004 \; MMIN - 3.99 \; CHMIN$$
$$+ 0.946 \; CHMAX$$
$$LM3 \; PRP = 38.1 + 0.012 \; MMIN$$
$$LM4 \; PRP = 19.5 + 0.002 \; MMAX + 0.698 \; CACH$$
$$+ 0.969 \; CHMAX$$
$$LM5 \; PRP = 285 \; 1.46 \; MYCT + 1.02 \; CACH$$
$$- 9.39 \; CHMIN$$
$$LM6 \; PRP = -65.8 + 0.03 \; MMIN - 2.94 \; CHMIN$$
$$+ 4.98 \; CHMAX$$

21

Classification rules

- Popular alternative to decision trees
- *Antecedent* (pre-condition): a series of tests (just like the tests at the nodes of a decision tree)
- Tests are usually logically ANDed together (but may also be general logical expressions)
- *Consequent* (conclusion): classes, set of classes, or probability distribution assigned by rule
- Individual rules are often logically ORed together
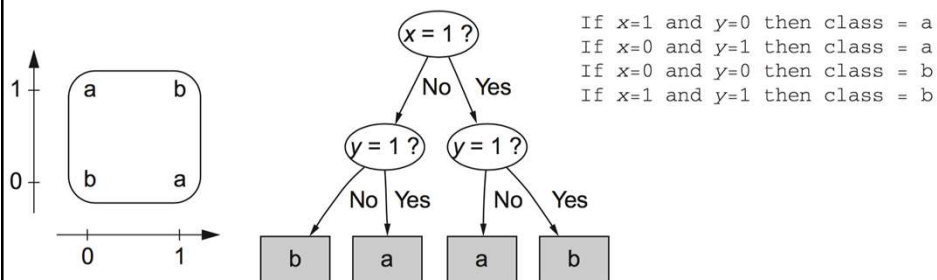 - Conflicts arise if different conclusions apply

22

From trees to rules

- Easy: converting a tree into a set of rules
  - One rule for each leaf:
    - Antecedent contains a condition for every node on the path from the root to the leaf
    - Consequent is class assigned by the leaf
- Produces rules that are unambiguous
  - Doesn't matter in which order they are executed
- But: resulting rules are unnecessarily complex
  - Pruning to remove redundant tests/rules

23

23

From rules to trees – not straightforward
The exclusive-or problem


```
If x=1 and y=0 then class = a
If x=0 and y=1 then class = a
If x=0 and y=0 then class = b
If x=1 and y=1 then class = b
```

- Even if the rule involves two attributes, split on one attribute first to get a sub-tree

24

24

Interpreting rules

- What if two or more rules conflict?
  - Give no conclusion at all?
  - Go with rule that is most popular on training data?
  - …
- What if no rule applies to a test instance?
  - Give no conclusion at all?
  - Go with class that is most frequent in training data?
  - …

Association rules

- Association rules…
  - … can predict any attribute and combinations of attributes
  - … are not intended to be used together as a set
- Problem: immense number of possible associations
  - Output needs to be restricted to show only the most predictive associations
    $\Rightarrow$ only those with high *support* and high *confidence*

Support and confidence of a rule

- Support: number of instances predicted correctly

- Confidence: number of correct predictions, as proportion of all instances that rule applies to

- Example: 4 cool days with normal humidity

  ```
  If temperature = cool then
  humidity = normal
  ```

  $\Rightarrow$ Support = 4, confidence = 100%

- Normally: minimum support and confidence pre-specified (e.g. 58 rules with support $\geq$ 2 and confidence $\geq$ 95% for weather data)

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

27

Interpreting association rules

- Interpretation is not obvious:

  ```
  If windy = false and play = no then outlook = sunny
                                   and humidity = high
  ```

  is *not* the same as

  ```
  If windy = false and play = no then outlook = sunny
  If windy = false and play = no then humidity = high
  ```
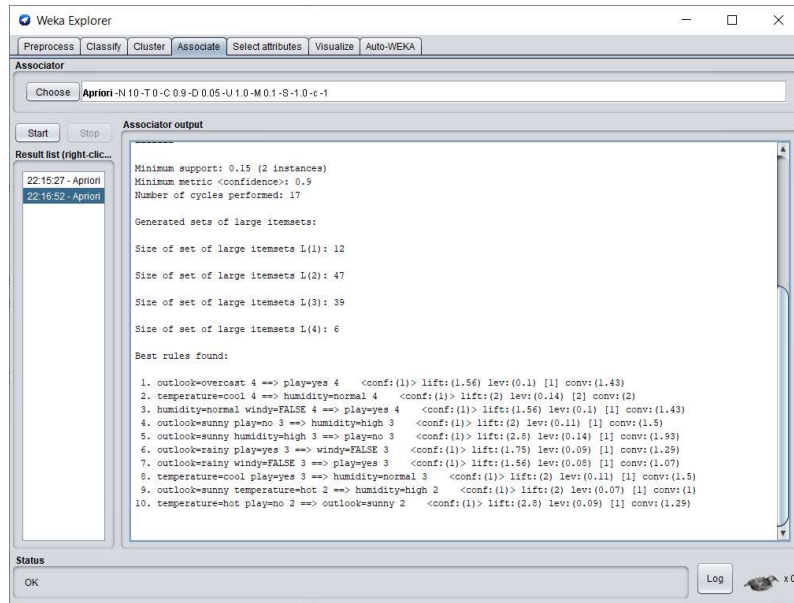
- It means that the following also holds:

  ```
  If humidity = high and windy = false and play = no
      then outlook = sunny
  ```

28

Association rules – weather.arff

Weka Explorer — Associator output:

```
Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 47

Size of set of large itemsets L(3): 39

Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4        <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4   <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4   <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3    <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3    <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3    <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3    <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3   <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2   <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2    <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)
```

29

Rules with exceptions

- Idea: allow rules to have *exceptions*
- Example: rule for iris data

`If petal-length ≥ 2.45 and petal-length < 4.45 then Iris-versicolor`

- New instance:

Sepal Length	Sepal Width	Petal Length	Petal Width	Type
5.1	3.5	2.6	0.2	?

- Modified rule:

`If petal-length ≥ 2.45 and petal-length < 4.45 then Iris-versicolor`
`  EXCEPT if petal-width < 1.0 then Iris-setosa`

30

30

Advantages of using exceptions

- Rules can be updated incrementally
  - Easy to incorporate new data
  - Easy to incorporate domain knowledge
- People often think in terms of exceptions
- Each conclusion can be considered just in the context of rules and exceptions that lead to it
  - Locality property is important for understanding large rule sets
  - "Normal" rule sets do not offer this advantage

31

31

More on exceptions

- `Default...except if...then...`

  is logically equivalent to

  `if...then...else`

  (where the "else" specifies what the "default" does)
- But: exceptions offer a psychological advantage
  - Assumption: defaults and tests early on apply more widely than exceptions further down
  - Exceptions reflect special cases

32

32

Rules involving relations

- So far: all rules involved comparing an attribute-value to a constant (e.g. temperature < 45)
- These rules are called "propositional" because they have the same expressive power as propositional logic
- What if problem involves relationships between examples (e.g. family tree problem from above)?
    - Can't be expressed with propositional rules
    - More expressive representation required

The shapes problem

- Target concept: *standing up*
- Shaded: *standing*
  Unshaded: *lying*

Shaded: *standing*
Unshaded: *lying*
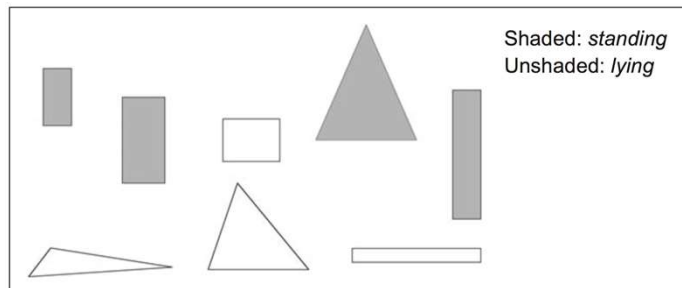
A propositional solution

Width	Height	Sides	Class
2	4	4	Standing
3	6	4	Standing
4	3	4	Lying
7	8	3	Standing
7	6	3	Lying
2	9	4	Standing
9	1	4	Lying
10	2	3	Lying

```
If width ≥ 3.5 and height < 7.0
   then lying
If height ≥ 3.5 then standing
```

Using relations between attributes

- Comparing attributes with each other enables rules like this:

```
If width > height then lying
If height > width then standing
```

- This description generalizes better to new data
- Standard relations: =, <, >
- But: searching for relations between attributes can be costly
- Simple solution: add extra attributes
  (e.g., a binary attribute "*is width < height?*")

Instance-based representation

- Simplest form of learning: *rote learning*
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called *instance-based* learning
- Similarity function defines what's "learned"
- Instance-based learning is *lazy* learning
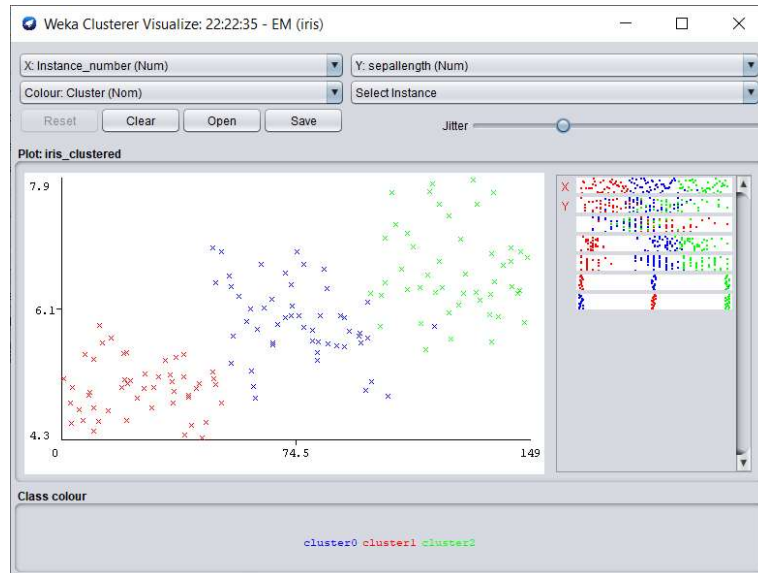- Methods: *nearest-neighbor, k-nearest-neighbor, …*

The distance function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
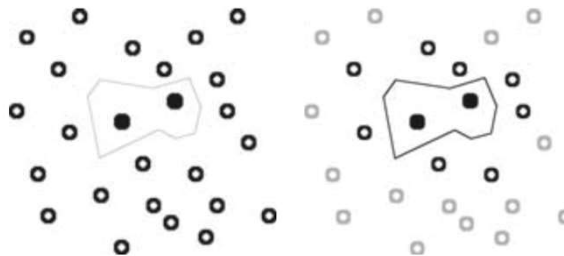 - Weighting the attributes might be necessary

IRIS data clusters

Learning prototypes

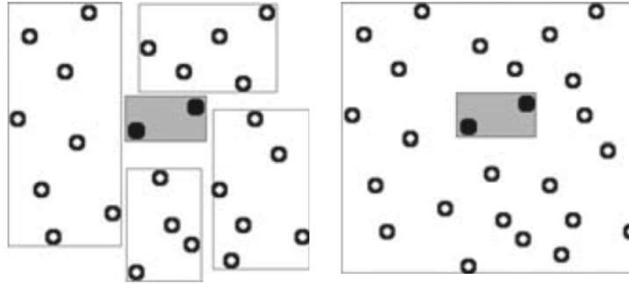- Only those instances involved in a decision need to be stored
- Noisy instances should be filtered out

Rectangular generalizations

- Nearest-neighbor rule is used outside rectangles
- Rectangles are rules! (But they can be more conservative than "normal" rules.)
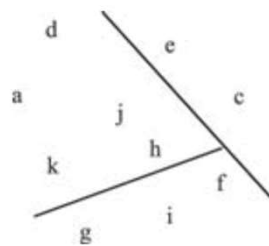- Nested rectangles are rules with exceptions
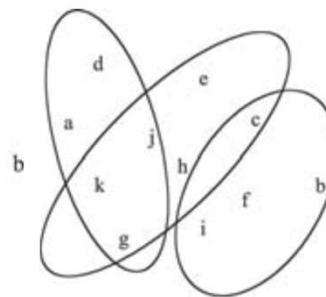
41

41

Representing clusters I

Simple 2-D representation

One cluster per example

Venn diagram

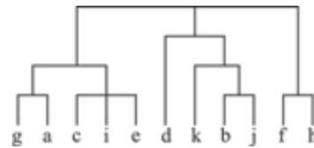Multiple clusters per example

42

42

21

Representing clusters II

Probabilistic assignment

Probability of belonging to each cluster

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1
…			

Dendrogram

Hierarchical clusters

g a c i e d k b j f h

43