CSCI 556 Data Analysis & Visualization

Output: knowledge representation

Instructor: Dr. Jinoh Kim

Some contents have been imported/modified from the Weka materials.

Output: Knowledge representation

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

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, …)

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

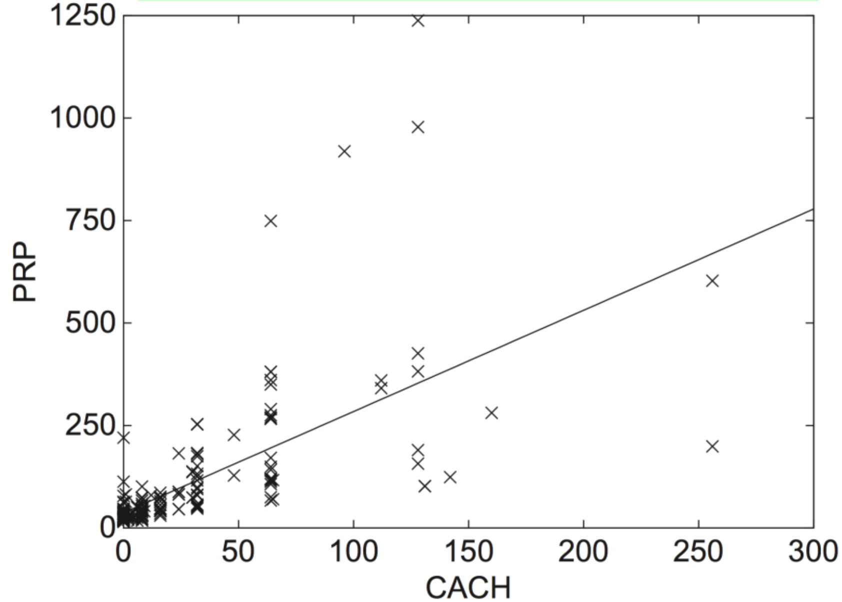
- Main problem: selecting the right attributes

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, the most famous one is to minimize the squared error

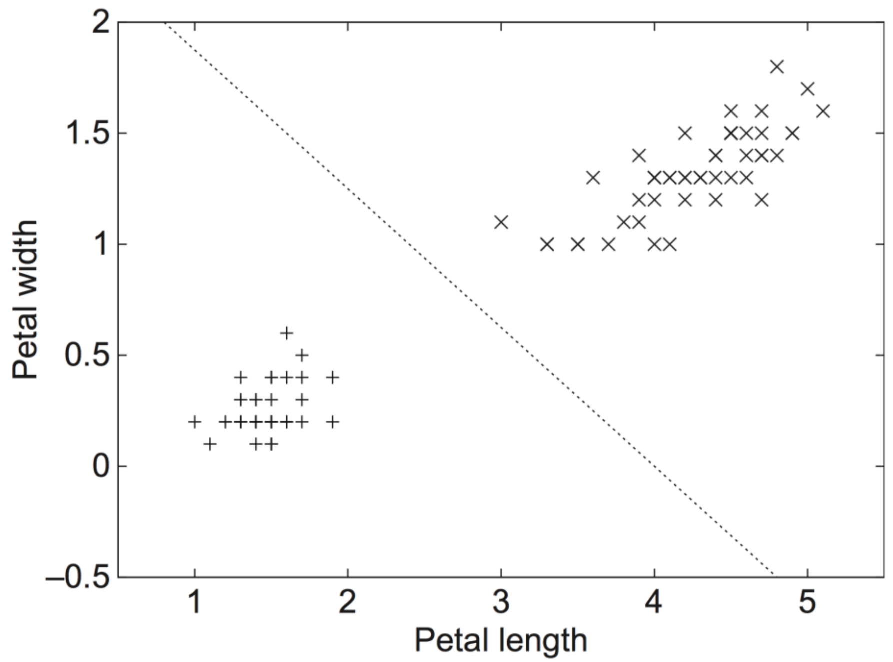
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

```
y = 37.06 + 2.47x
```



Linear models for classification

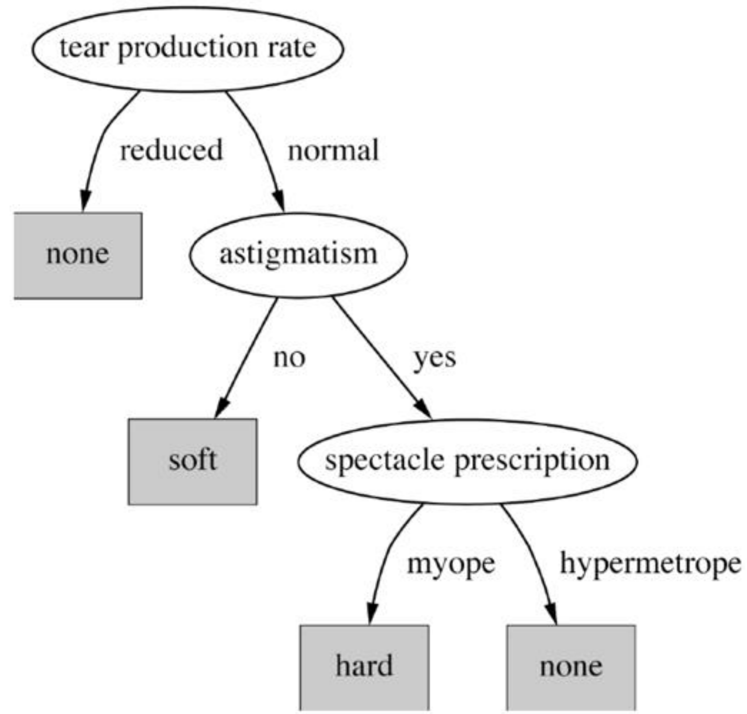
- Binary classification
- Line separates the two classes
  - Decision boundary - defines where the decision changes from one class value to the other
- Prediction is made by plugging in observed values of the attributes into the expression
  - Predict one class if output ≥ 0, and the other class if output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




```
2.0 – 0.5x – 0.8y = 0
```

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



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*

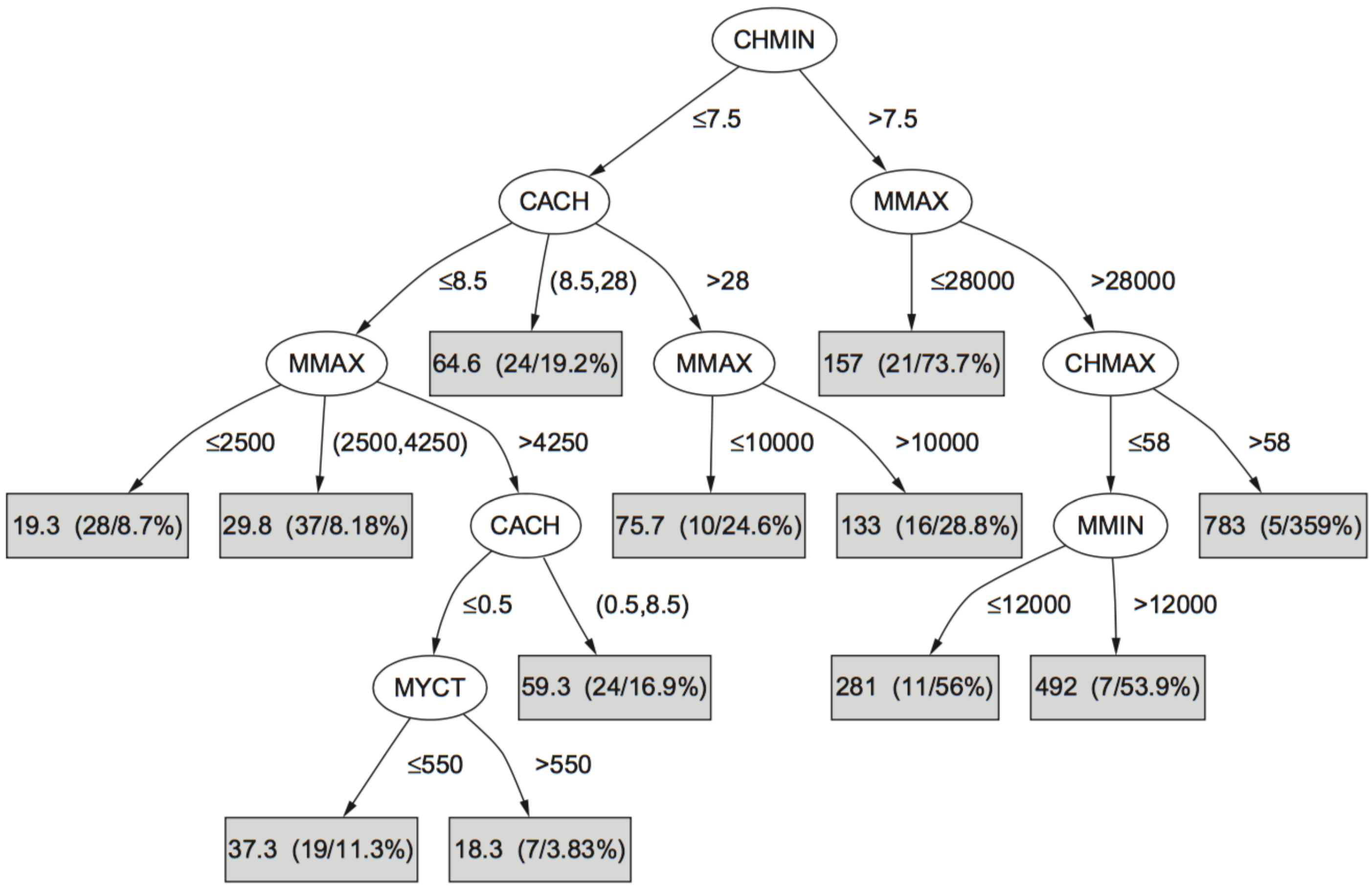
Missing values

- Does absence of value have some significance?
- Yes $\Rightarrow$ "missing" is a separate value
- No $\Rightarrow$ "missing" must be treated in a special way
  - E.g., assign instance to most popular branch

Trees for numeric prediction

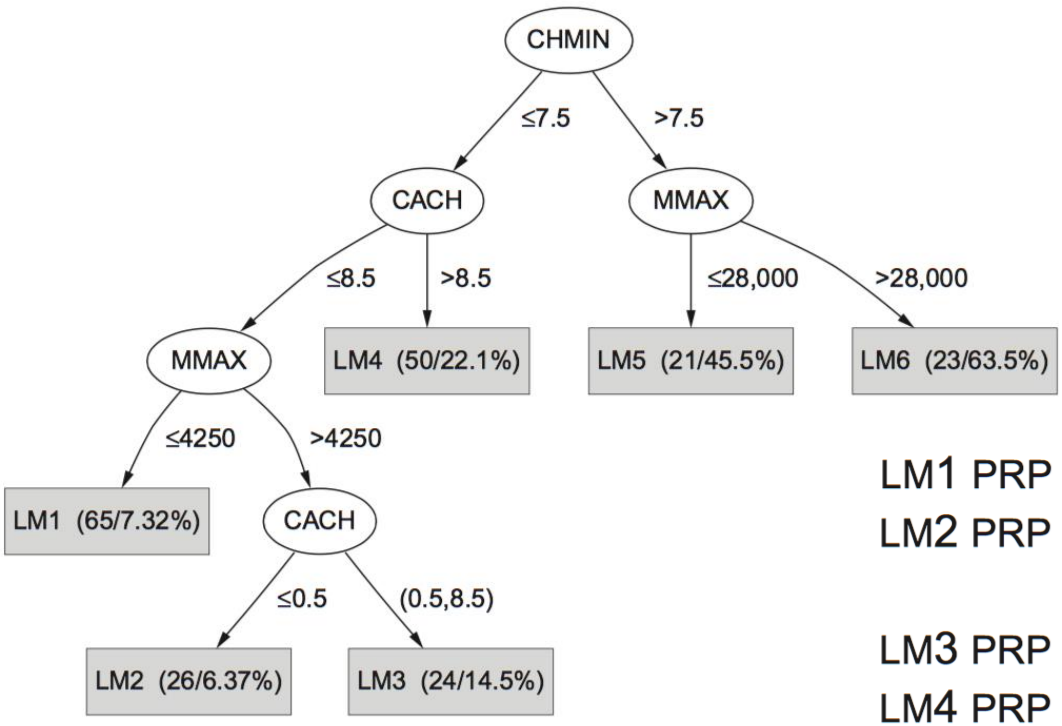
- Regression: the process of computing an expression that predicts a numeric quantity
- Regression tree: “decision tree” where each leaf predicts a numeric quantity
  - Predicted value is average value of training instances that reach the leaf
- Model tree: “regression tree” with linear regression models at the leaf nodes
  - Linear patches approximate continuous function

Regression tree for the CPU data



Predicted value is average value of training instances that reach the leaf.

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

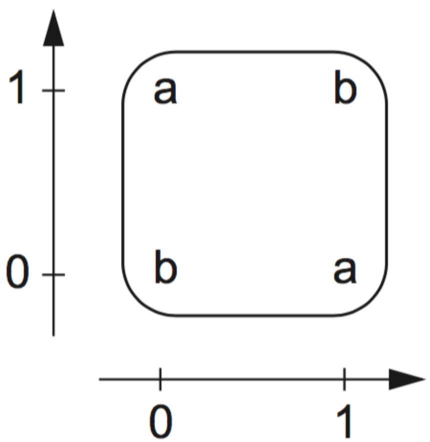
LM6 PRP = – 65.8 + 0.03 MMIN – 2.94 CHMIN
+ 4.98 CHMAX

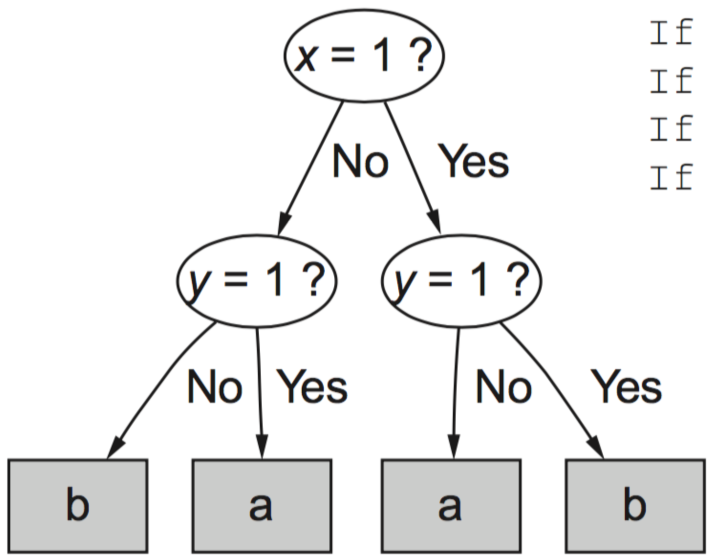
Each leaf has a linear regression models.

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

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

“Nuggets” of knowledge

- Are rules independent pieces of knowledge? (It seems easy to add a rule to an existing rule base.)
- Problem: ignores how rules are executed
- Two ways of executing a rule set:
  - Ordered set of rules (“decision list”)
    - Order is important for interpretation
  - Unordered set of rules
    - Rules may overlap and lead to different conclusions for the same instance

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?
  - …

Special case: Boolean class

- Assumption: if instance does not belong to class “yes”, it belongs to class “no”
- Trick: only learn rules for class “yes” and use default rule for “no”

```
If x = 1 and y = 1 then class = a
If z = 1 and w = 1 then class = a
Otherwise class = b
```

- Order of rules is not important. No conflicts!

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 (class=Iris-setosa):

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

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

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

Using relations between attributes

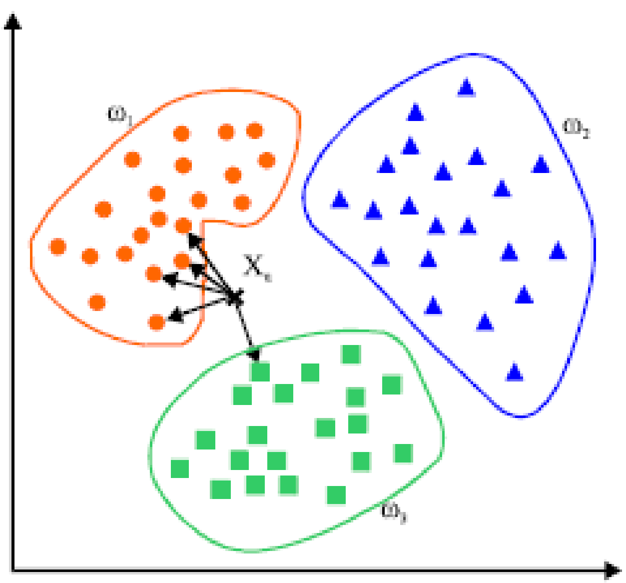
- Comparing attributes with each other enables rules like this:

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

- This description generalizes better to new data
- Standard relations: =, <, >
- Searching for relations between attributes may 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 (aka distance function) defines what's "learned"
- Instance-based learning is *lazy* learning
- Methods: *nearest-neighbor*, *k-nearest-neighbor*, …

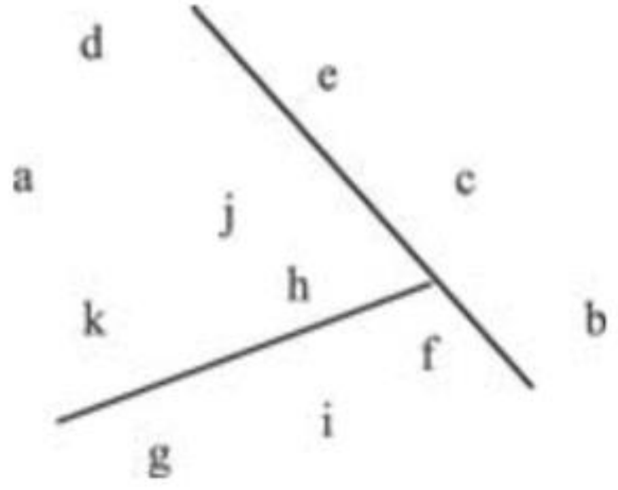


Clusters

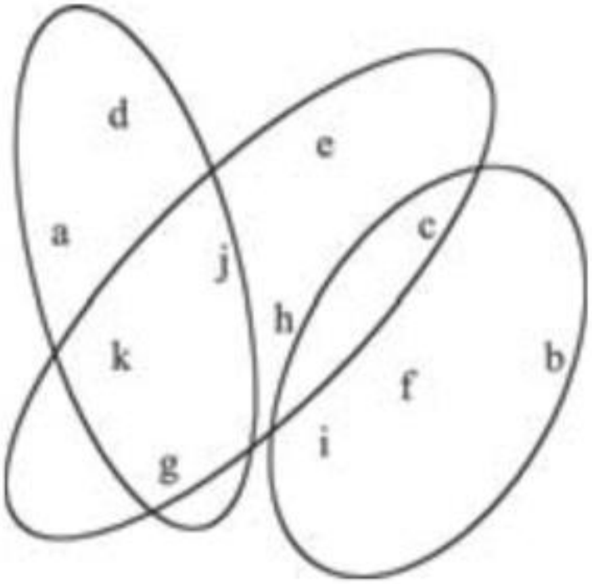
- Output takes the form of a diagram showing how the instances fall into clusters
  - A cluster number for each instance
- Some clustering algorithms allow one instance to belong to 1+ clusters
- The probability can be given rather than the categorical info
- Hierarchical structure of clusters (dendograms)

Representing clusters

Simple 2-D representation



Venn diagram

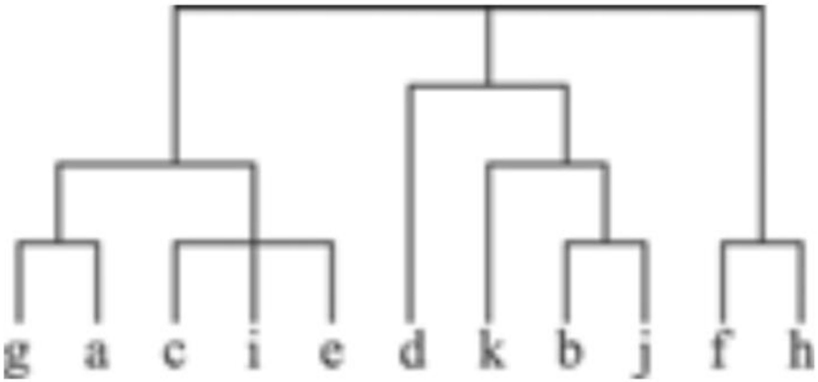


Representing clusters (cont'd)

Probabilistic assignment

	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



Summary

- Output – knowledge representation
- Linear models
- Decision trees
- Rules
- Instance-based representation
- Clustering