

Machine Learning

Chapter 3. Output

3

Output: Knowledge representation

- ❖ Decision tables
- ❖ Decision trees
- ❖ Decision rules
- ❖ Association rules
- ❖ Rules with exceptions
- ❖ Rules involving relations
- ❖ Linear regression
- ❖ Trees for numeric prediction
- ❖ 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 same format as 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

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

❖ Nominal:

number of children usually equal to number values

⇒ attribute won't get tested more than once

❑ Other possibility: division into two subsets

❖ Numeric:

test whether value is greater or less than constant

⇒ 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 A: assign instance to most popular branch
 - ❑ Solution B: split instance into pieces
 - Pieces receive weight according to fraction of training instances that go down each branch
 - Classifications from leaf nodes are combined using the weights that have percolated to them

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

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

From rules to trees

- ❖ More difficult: transforming a rule set into a tree

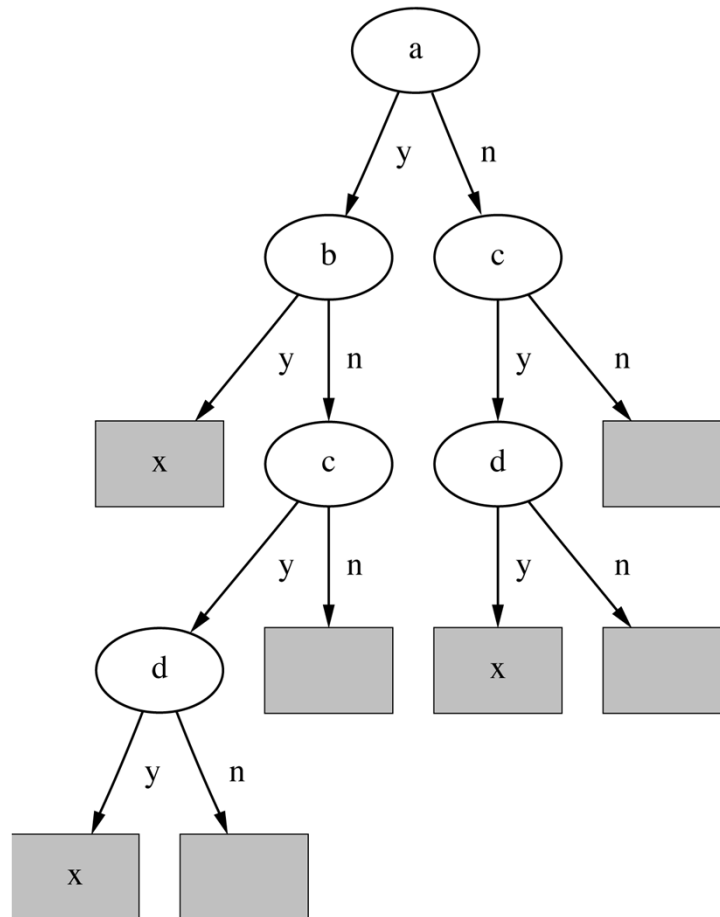
- ❑ Tree cannot easily express disjunction between rules

- ❖ Example: rules which test different attributes

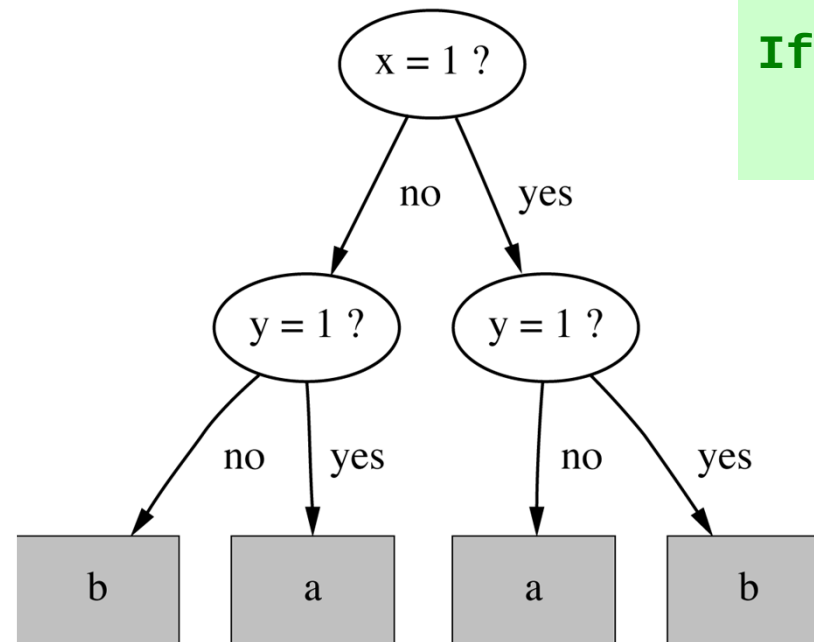
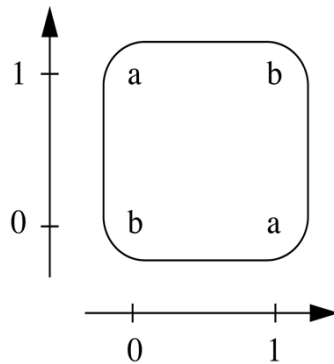
```
If a and b then x  
If c and d then x
```

- ❖ Symmetry needs to be broken
- ❖ Corresponding tree contains identical subtrees ($\Rightarrow$ “replicated subtree problem”)

A tree for a simple disjunction



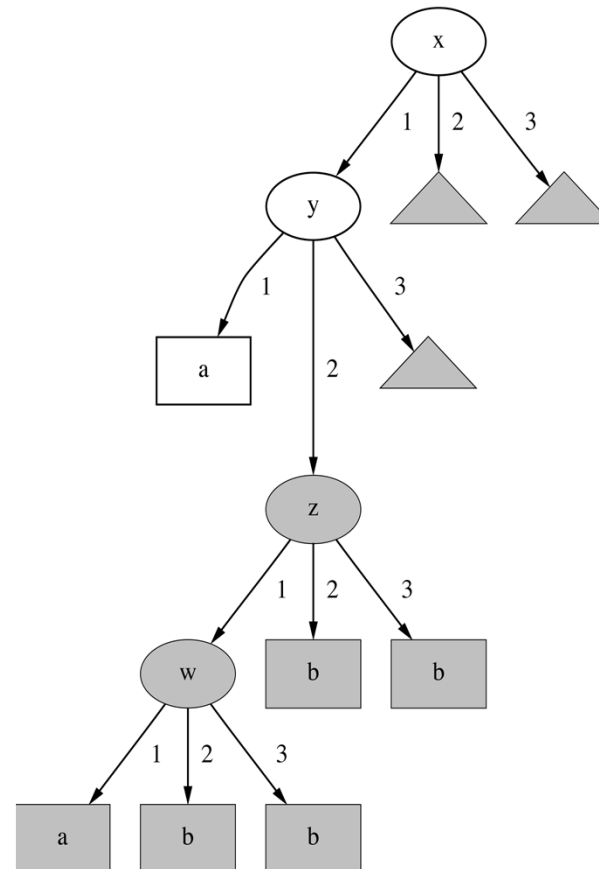
The exclusive-or problem



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

A tree with a replicated subtree

If $x = 1$ and $y = 1$
then class = a
If $z = 1$ and $w = 1$
then class = a
Otherwise 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!
- ❖ Rule can be written in *disjunctive normal form*

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 => 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
⇒ Support = 4, confidence = 100%
- ❖ Normally: minimum support and confidence pre-specified (e.g. 58 rules with support ≥ 2 and confidence $\geq 95\%$ for weather data)

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

❖ However, it means that the following also holds:

if humidity = high and windy = false and play = no
then outlook = sunny

Rules with exceptions

❖ Idea: allow rules to have exceptions

❖ Example: rule for iris data

```
if petal-length  $\geq$  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	Iris-setosa

❖ Modified rule:

```
if petal-length  $\geq$  2.45 and petal-length < 4.45  
then Iris-versicolor EXCEPT if petal-width < 1.0 then  
Iris-setosa
```

A more complex example

❖ Exceptions to exceptions to exceptions ...

```
default: Iris-setosa
except if petal-length  $\geq$  2.45 and petal-length  $<$  5.355
      and petal-width  $<$  1.75
  then Iris-versicolor
      except if petal-length  $\geq$  4.95 and petal-width  $<$  1.55
        then Iris-virginica
        else if sepal-length  $<$  4.95 and sepal-width  $\geq$  2.45
          then Iris-virginica
  else if petal-length  $\geq$  3.35
    then Iris-virginica
      except if petal-length  $<$  4.85 and sepal-length  $<$  5.95
        then Iris-versicolor
```

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 don’t offer this advantage

More on exceptions

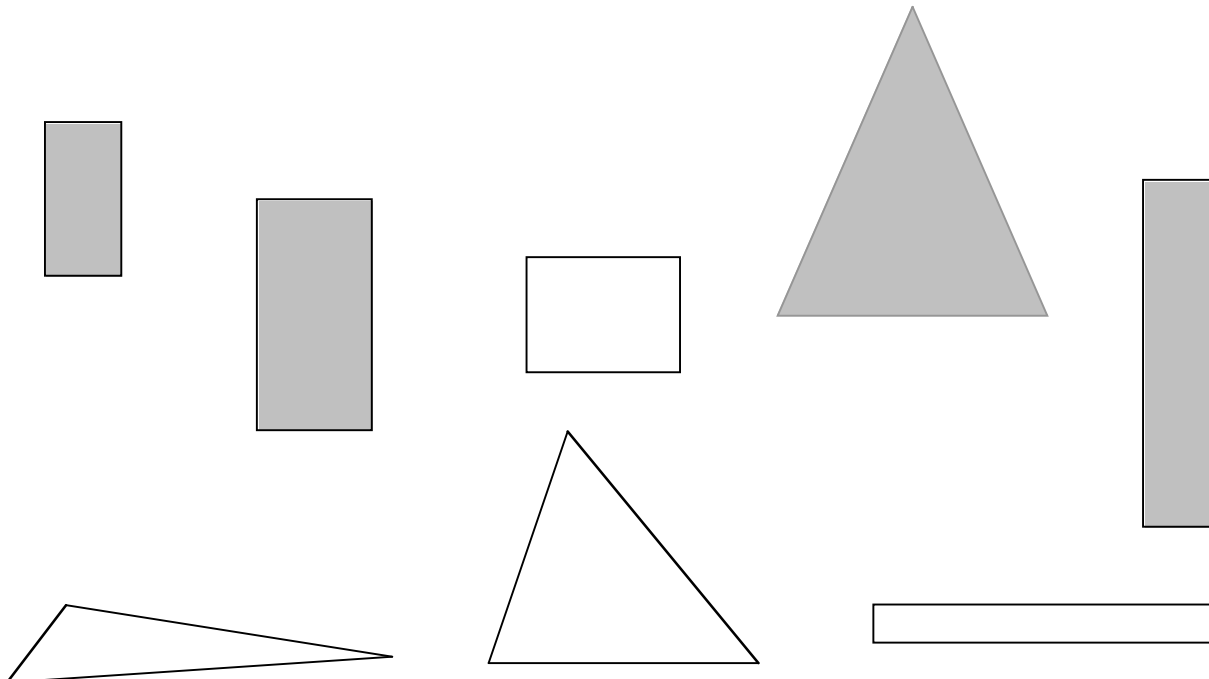
- ❖ “Default ... except if ... then ...”
is logically equivalent to
“if ... then ... else”
(where the else specifies what the default did)
- ❖ But: exceptions offer a psychological advantage
 - ❑ Assumption: defaults and tests early on apply more widely than exceptions further down
 - ❑ Exceptions reflect special cases

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*



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

A relational solution

- ❖ Comparing attributes with each other

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

- ❖ Generalizes better to new data
- ❖ Standard relations: =, <, >
- ❖ But: learning relational rules is costly
- ❖ Simple solution: add extra attributes
(e.g. a binary attribute *is width < height?*)

Rules with variables

- ❖ Using variables and multiple relations:

```
If height_and_width_of(x,h,w) and h > w  
then standing(x)
```

- ❖ The top of a tower of blocks is standing:

```
If height_and_width_of(x,h,w) and h > w  
and is_top_of(x,y)  
then standing(x)
```

- ❖ The whole tower is standing:

```
If is_top_of(x,z) and  
height_and_width_of(z,h,w) and h > w  
and is_rest_of(x,y) and standing(y)  
then standing(x)  
If empty(x) then standing(x)
```

- ❖ Recursive definition!

Inductive logic programming

- ❖ Recursive definition can be seen as logic program
- ❖ Techniques for learning logic programs stem from the area of “inductive logic programming” (ILP)
- ❖ But: recursive definitions are hard to learn
 - ❑ Also: few practical problems require recursion
 - ❑ Thus: many ILP techniques are restricted to non-recursive definitions to make learning easier

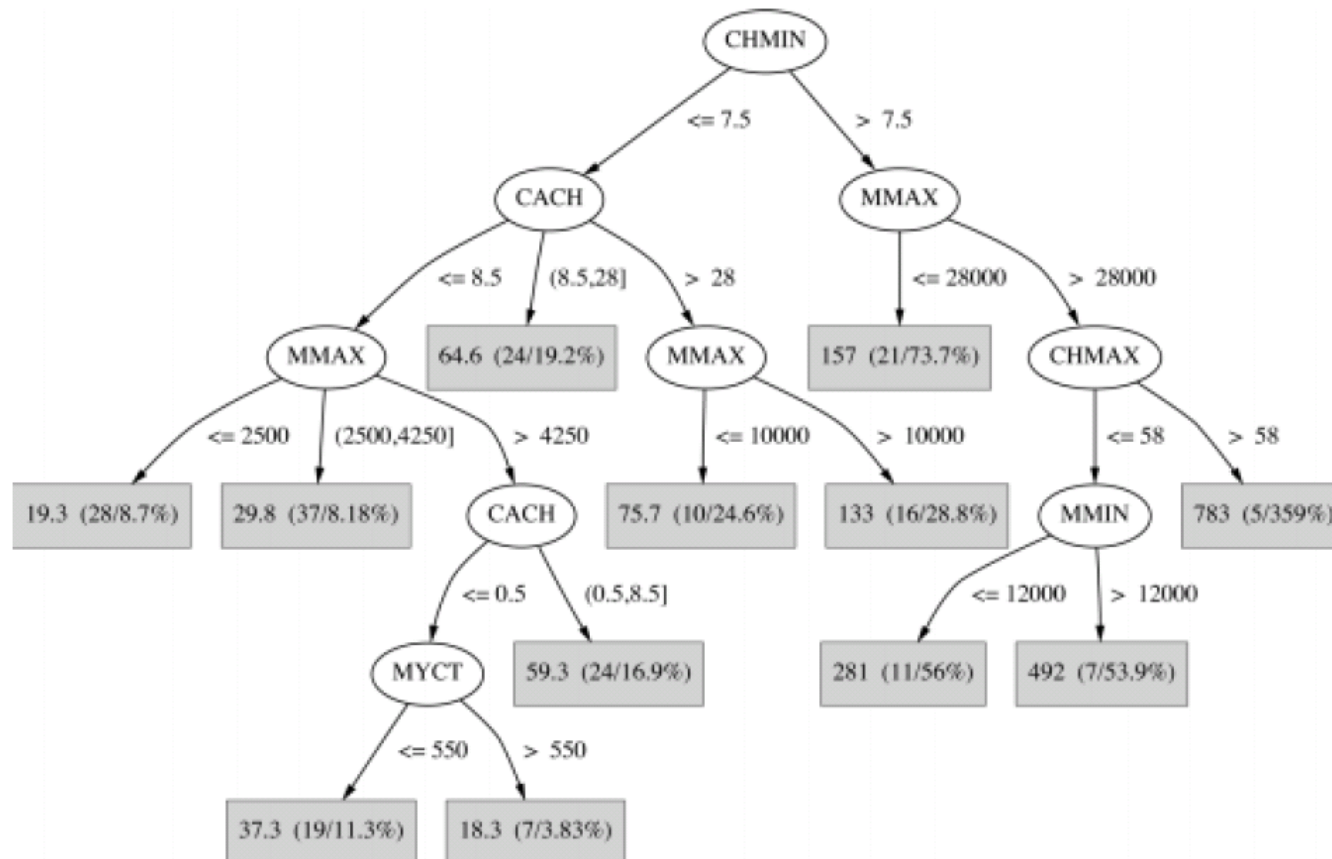
Trees for numeric prediction

- ❖ *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

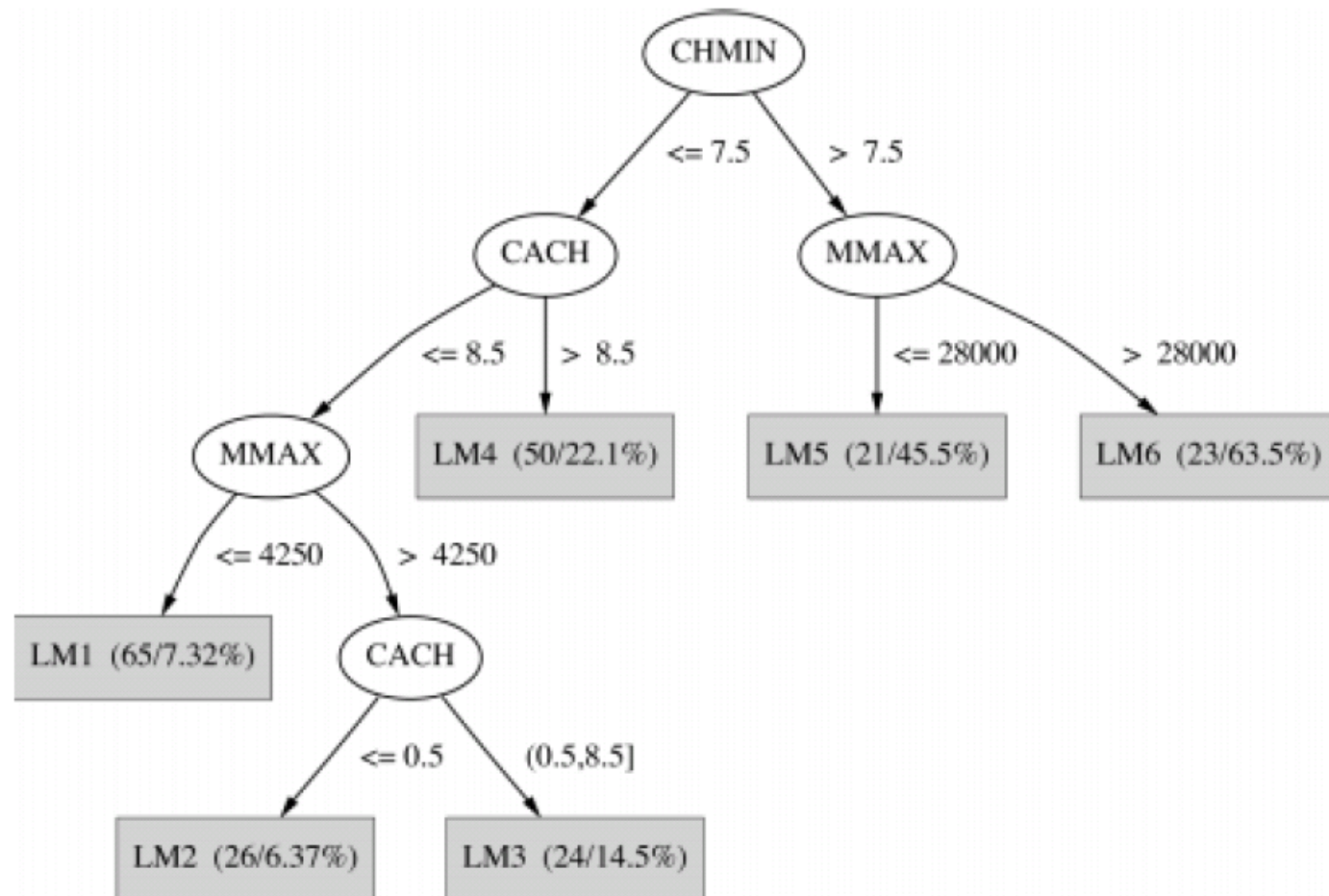
Linear regression for the CPU data

$$\begin{aligned} \text{PRP} = & - 56.1 \\ & + 0.049 \text{ MYCT} \\ & + 0.015 \text{ MMIN} \\ & + 0.006 \text{ MMAX} \\ & + 0.630 \text{ CACH} \\ & - 0.270 \text{ CHMIN} \\ & + 1.46 \text{ CHMAX} \end{aligned}$$

Regression tree for the CPU data



Model tree for the CPU data



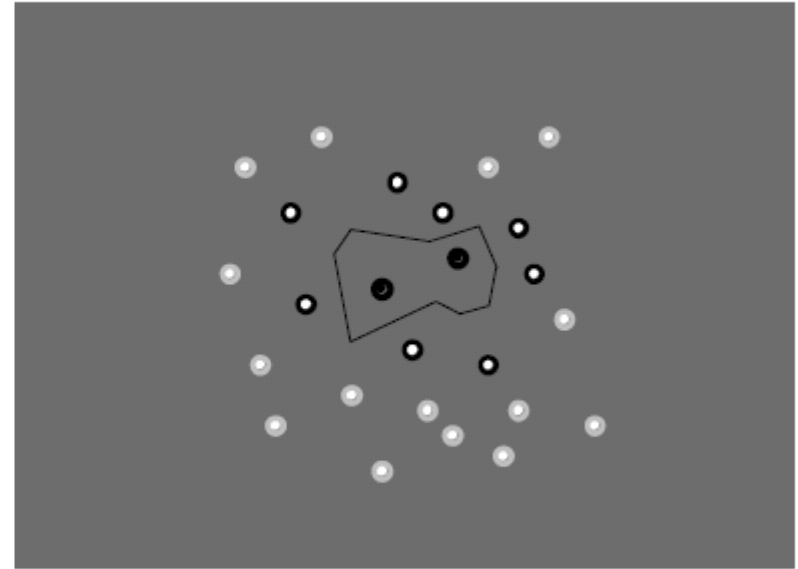
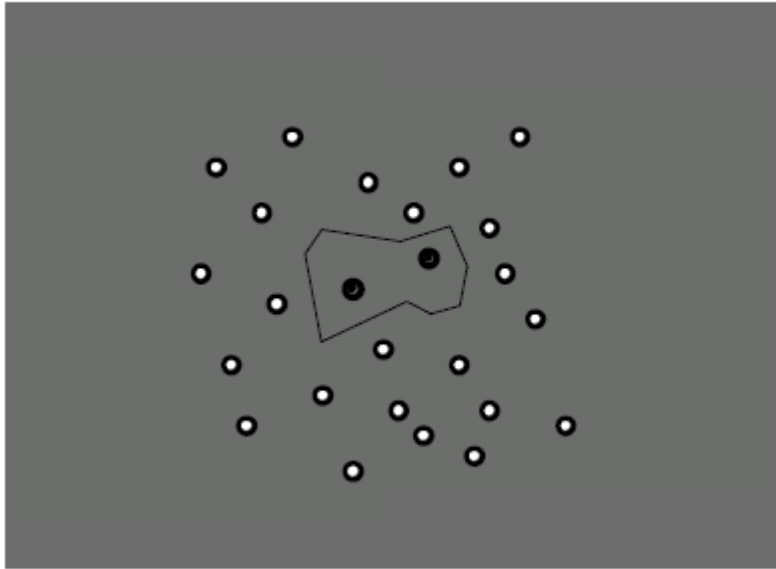
Instance-based representation

- ❖ 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

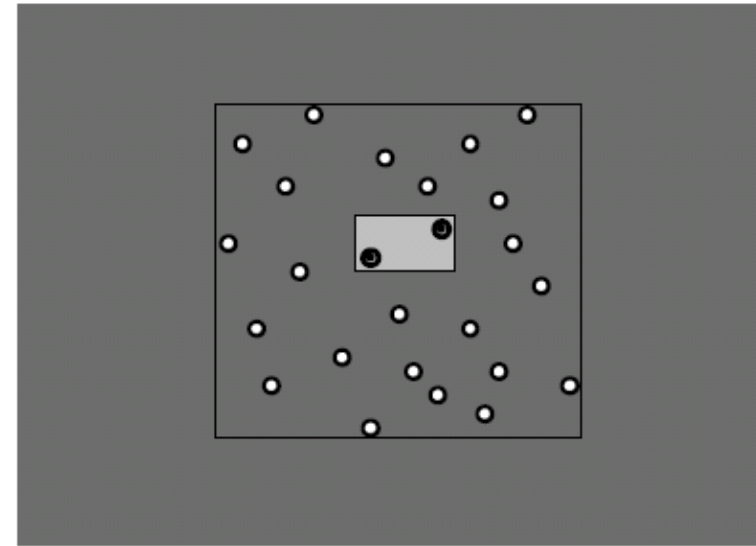
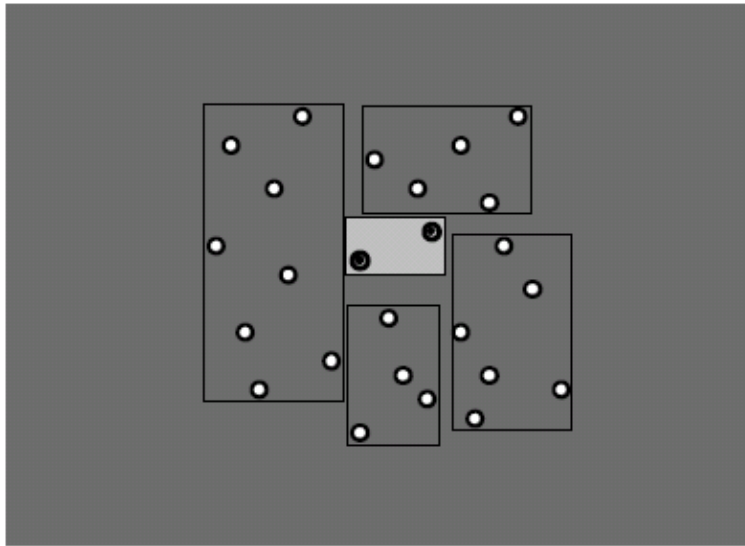
- ❖ Simplest case: one numeric attribute
 - ❑ Distance is the difference between the two attribute values involved (or a function thereof)
- ❖ Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- ❖ Nominal attributes: distance is set to 1 if values are different,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

Learning prototypes



- ❖ Only those instances involved in a decision need to be stored
- ❖ Noisy instances should be filtered out
- ❖ Idea: only use *prototypical* examples

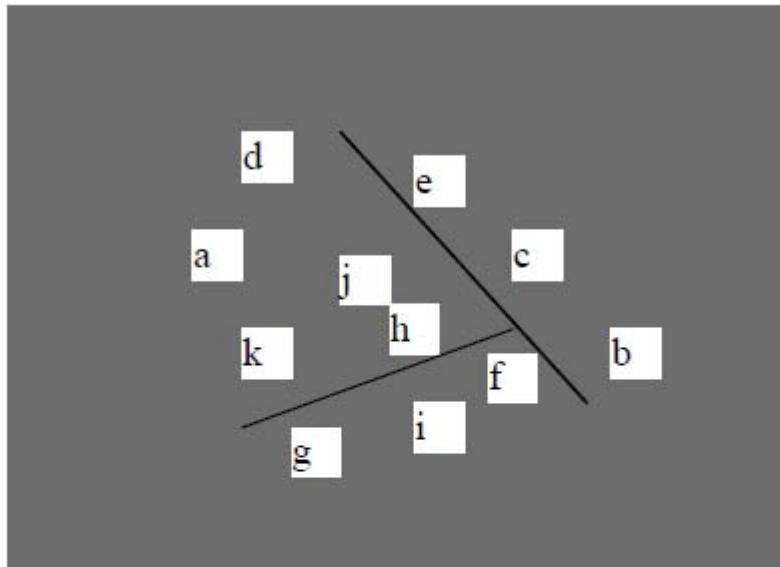
Rectangular generalizations



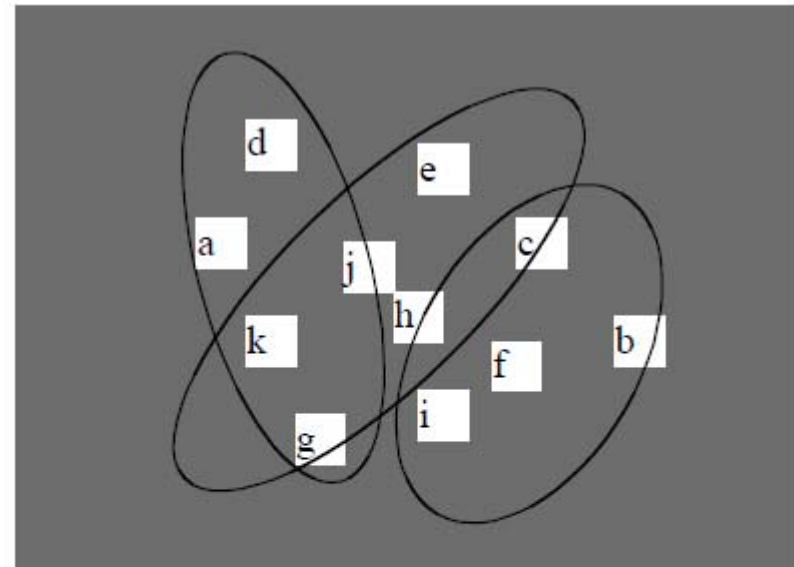
- ❖ Nearest-neighbor rules is used outside rectangles
- ❖ Rectangles are rules! (But they can be more conservative than “normal” rules.)
- ❖ Nested rectangles are rules with exceptions

Representing clusters I

Simple 2-D representation



Venn diagram



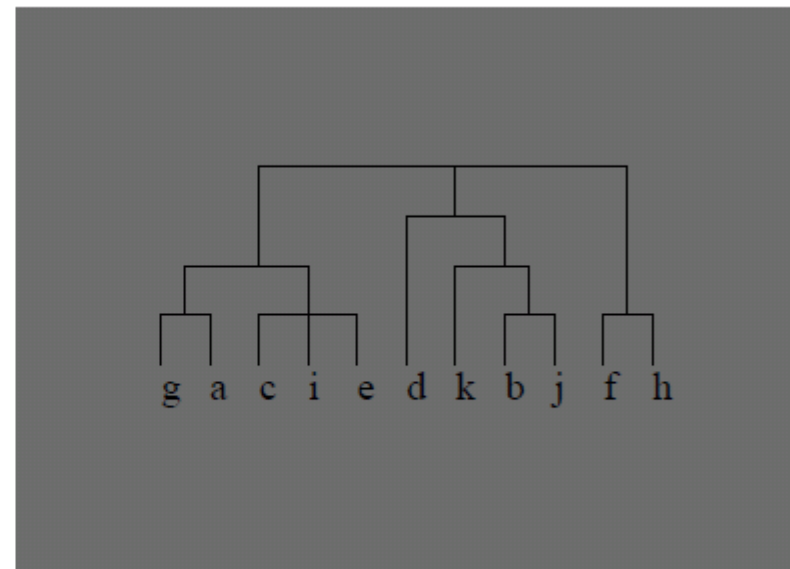
Overlapping clusters

Representing clusters II

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



NB: dendron is the Greek word for tree