IS 733 Lesson 5

Knowledge Representation

Some slides based on those by Witten et al., Han et al., James Foulds, & Vandana Janeja

Announcements

- Reminder: **Homework 2** is due Friday, 3/5/2021

To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

A "divide and conquer" approach to the problem of learning leads to a style of representation called a _____.

Decision table

Decision tree

Linear model

Decision list

Instance-based representation.

0

To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

_____ rules are no different from _____ rules except that they can predict any attribute, and combinations of attributes.

Association rules, classification rules

Classification rules, association rules

0

To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

In ___ learning, all the real work is done when the time comes to classify a new instance rather than when the training set is processed.

Linear model

Decision tree

Decision list

Instance-based

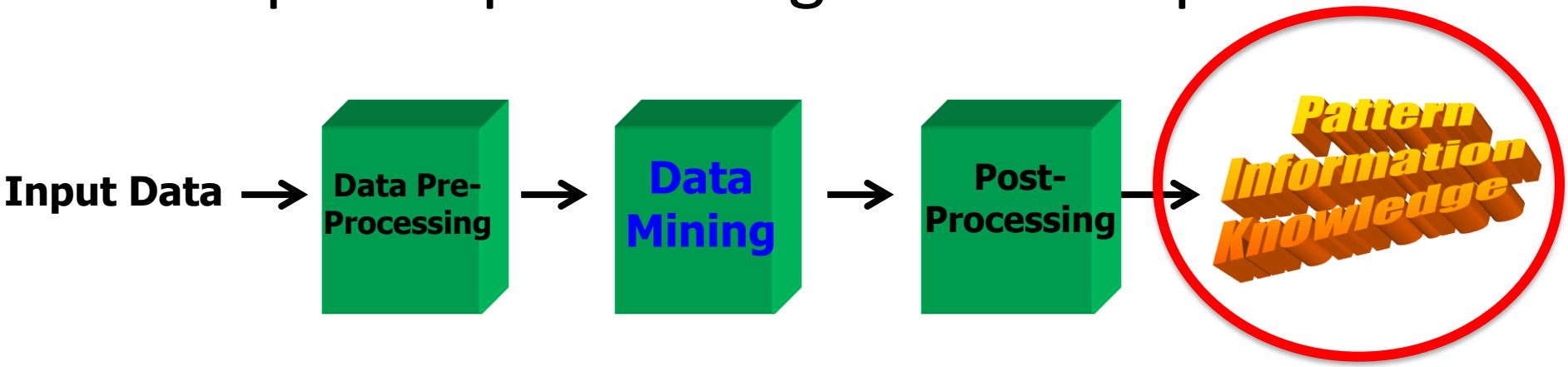
0

Learning outcomes

By the end of the lesson, you should be able to:

- **Discuss** how different styles of classification representations lead to different types of **decision boundaries**
- **Construct** **decision trees** by hand for simple functions
- **Convert** simple **decision trees** to **classification rules**
- **Construct** **decision boundaries** for **nearest neighbor** classification problems

Output: representing structural patterns



- Many different ways of representing **patterns**, a.k.a. **knowledge representation**
  - Decision trees, rules, instance-based, …
- Different types of output for different learning problems (e.g., classification, regression, …)

Output: Knowledge representation

Possible types of representation, i.e. the output of our data mining / machine learning methods, include:

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

Decision tables

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

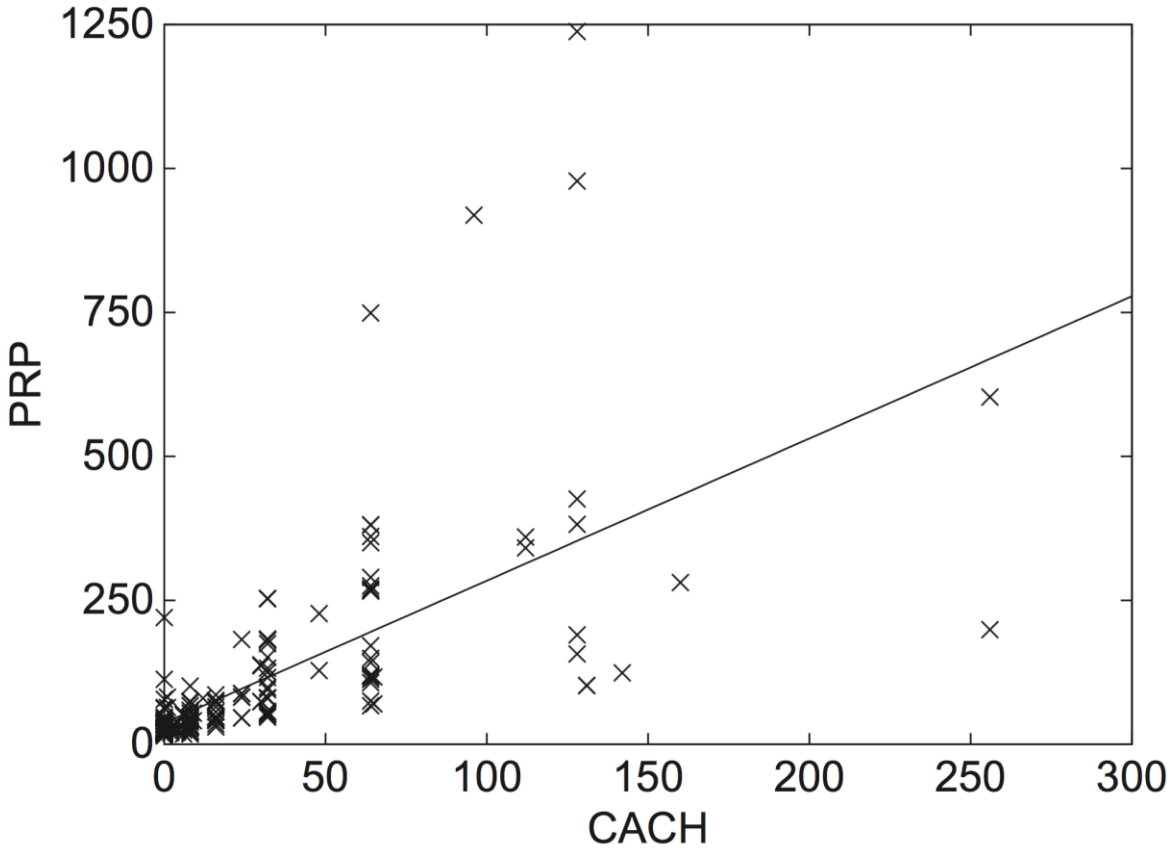
Outlook	Humidity	Play
Sunny	High	No
Sunny	Normal	Yes
Overcast	High	Yes
Overcast	Normal	Yes
Rainy	High	No
Rainy	Normal	No

- Problems:
  - selecting the right attributes
  - Table size exponential in # attributes
  - How to generalize to new data?

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

A linear regression function for the CPU performance data

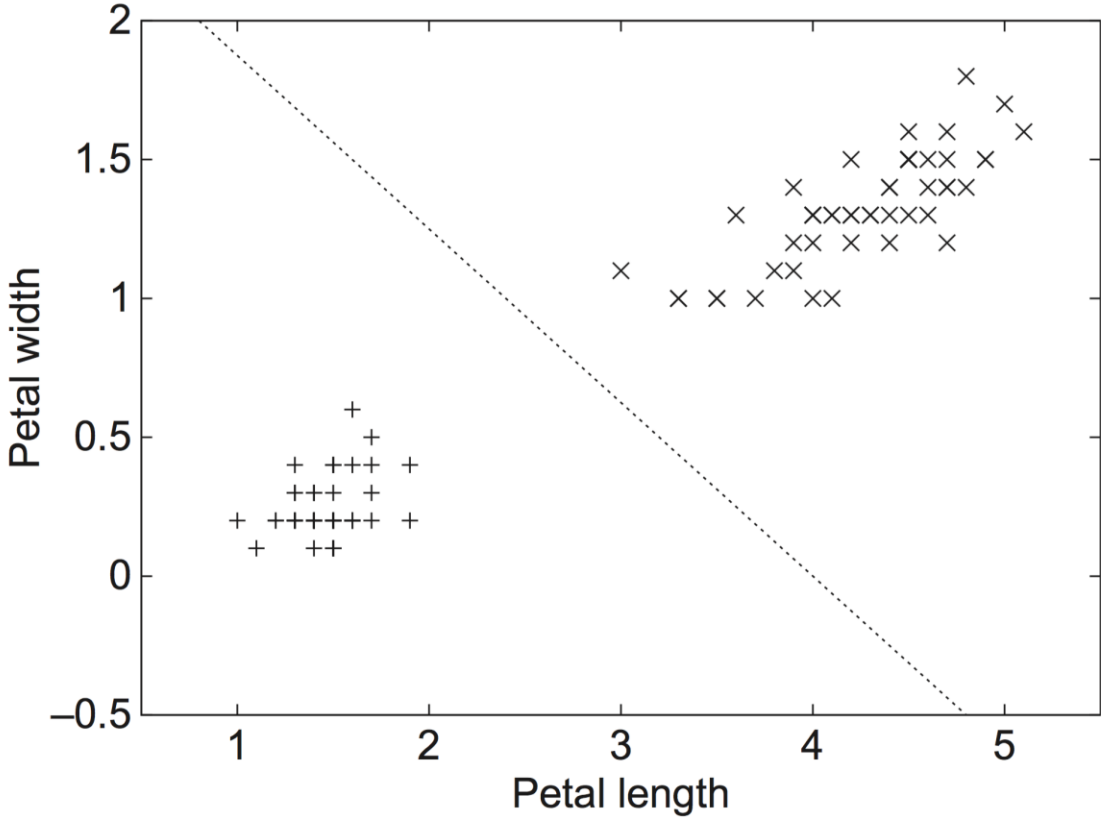


PRP = 37.06 + 2.47CACH

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

Separating setosas from versicolors



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

To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

Consider the task of representing binary Boolean logical operators as a classifier, where true and false are represented numerically as 1 and 0. How many of the following operators can be represented with a linear decision boundary? AND(X,Y), OR(X,Y), XOR(X,Y), IMPLIES(X,Y), IFF(X,Y)

1

2

3

4

5

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

Decision trees' decision boundary

- Recursively partitions the space in axis-parallel directions

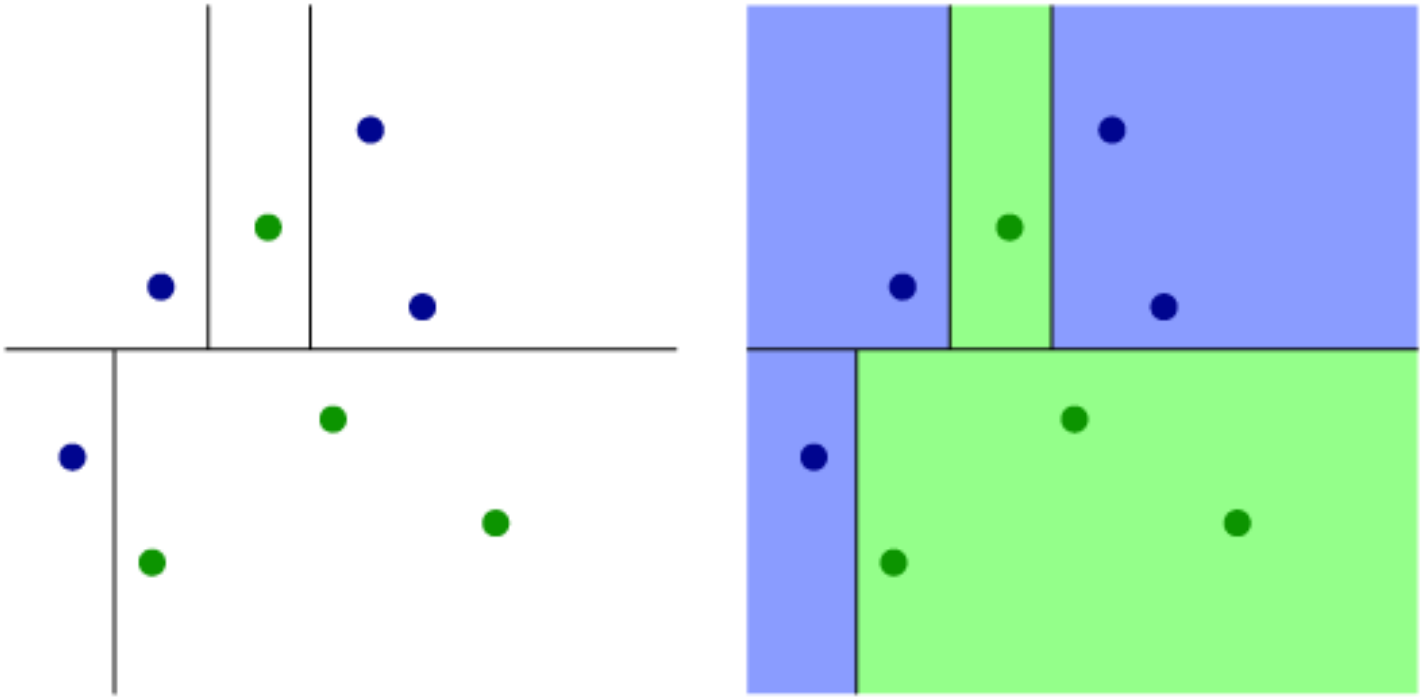
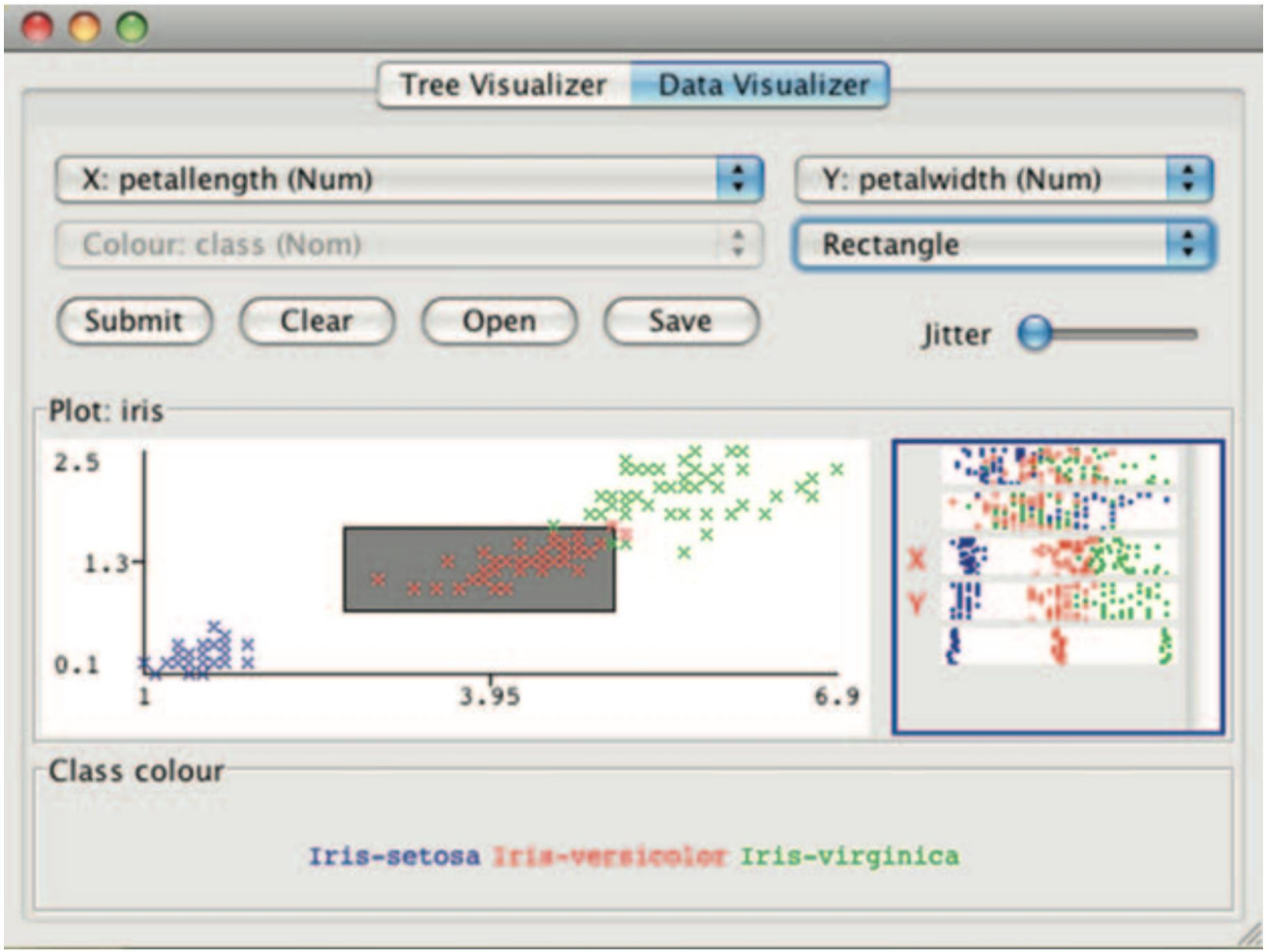


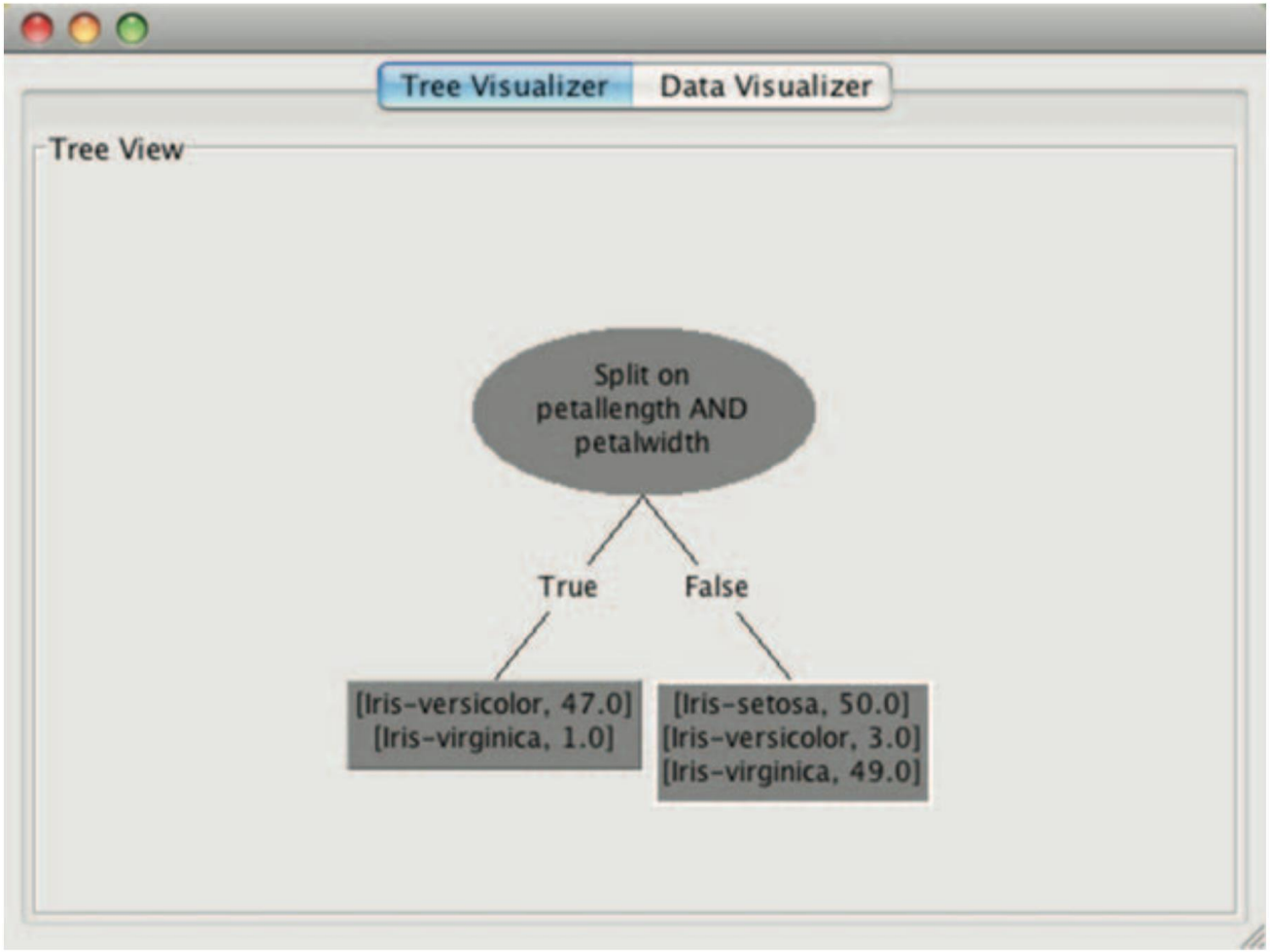
Figure from https://shapeofdata.wordpress.com/2013/07/02/decision-trees/

Interactive tree construction I



Classifiers -> Trees -> UserClassifier. Disable cross-validation so that you don't have to draw 10 trees!

Interactive tree construction II



Classifiers -> Trees -> UserClassifier. Disable cross-validation so that you don't have to draw 10 trees!

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 (interval)*

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 leave nodes are combined using the weights that have percolated to them

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

PRP =

– 56.1

+ 0.049 MYCT

+ 0.015 MMIN

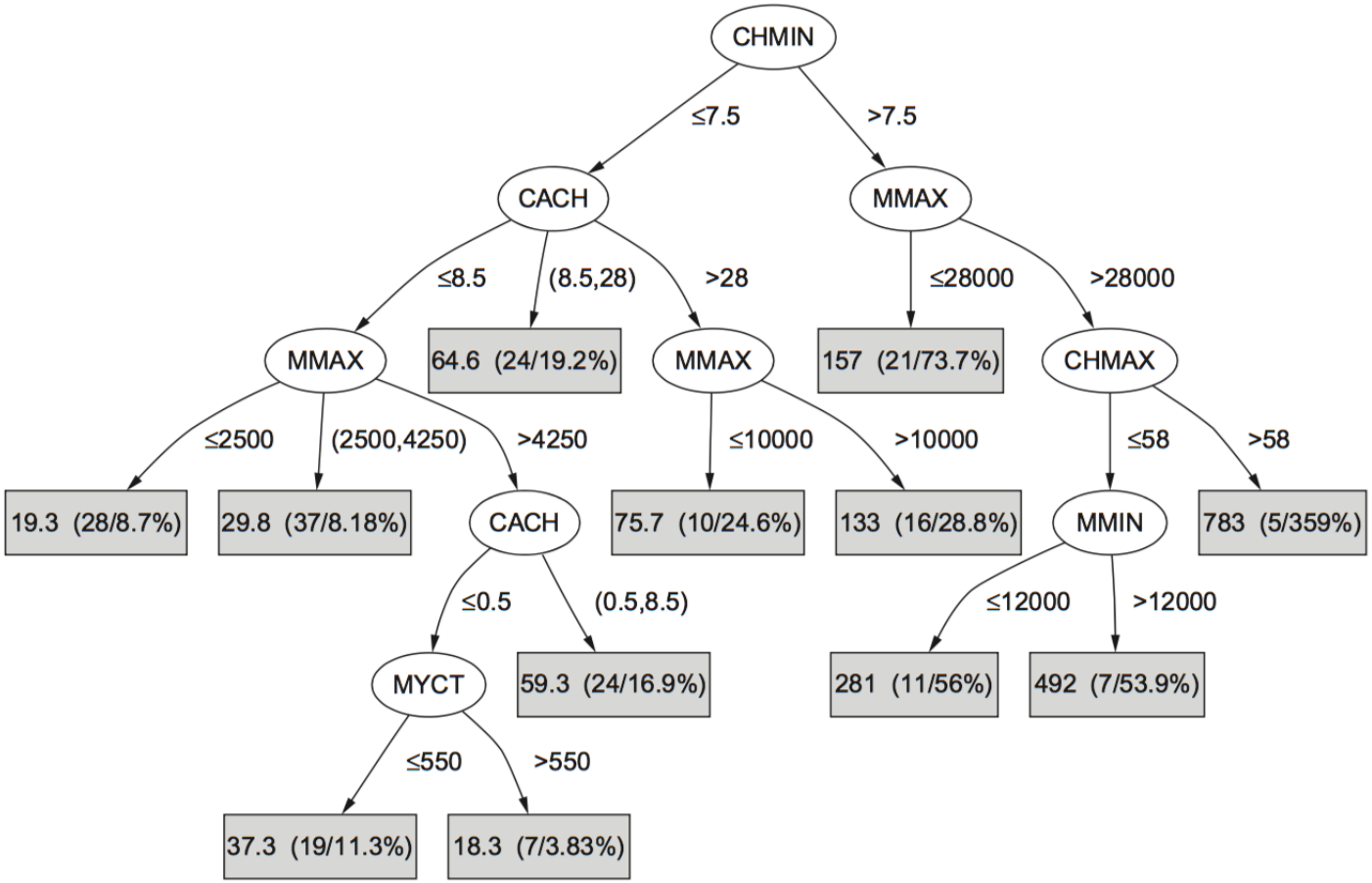
+ 0.006 MMAX

+ 0.630 CACH

– 0.270 CHMIN

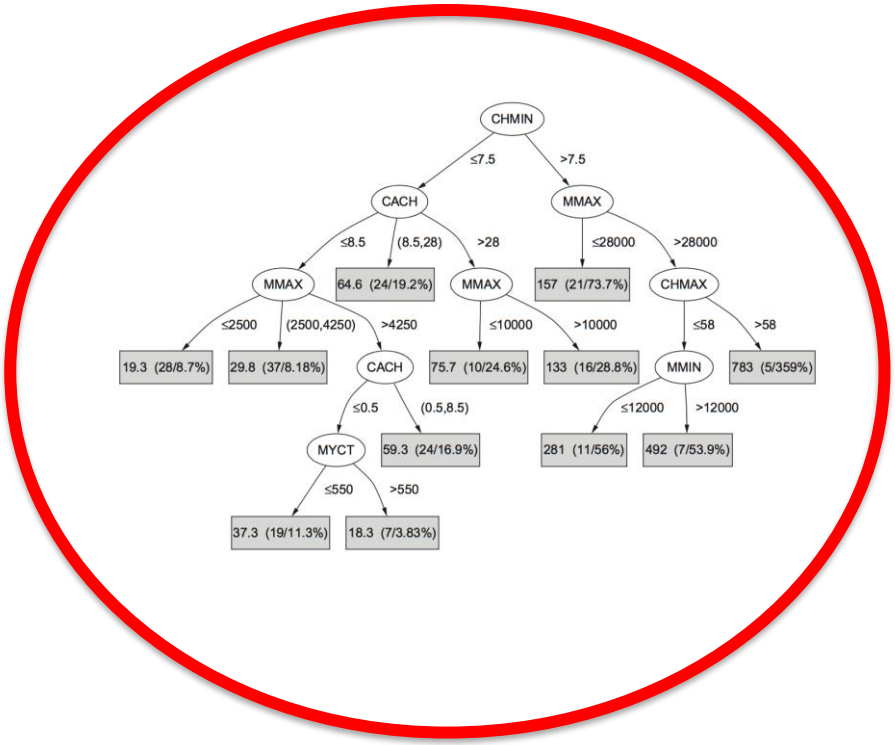
+ 1.46 CHMAX

Regression tree for the CPU data



Trees for numeric prediction

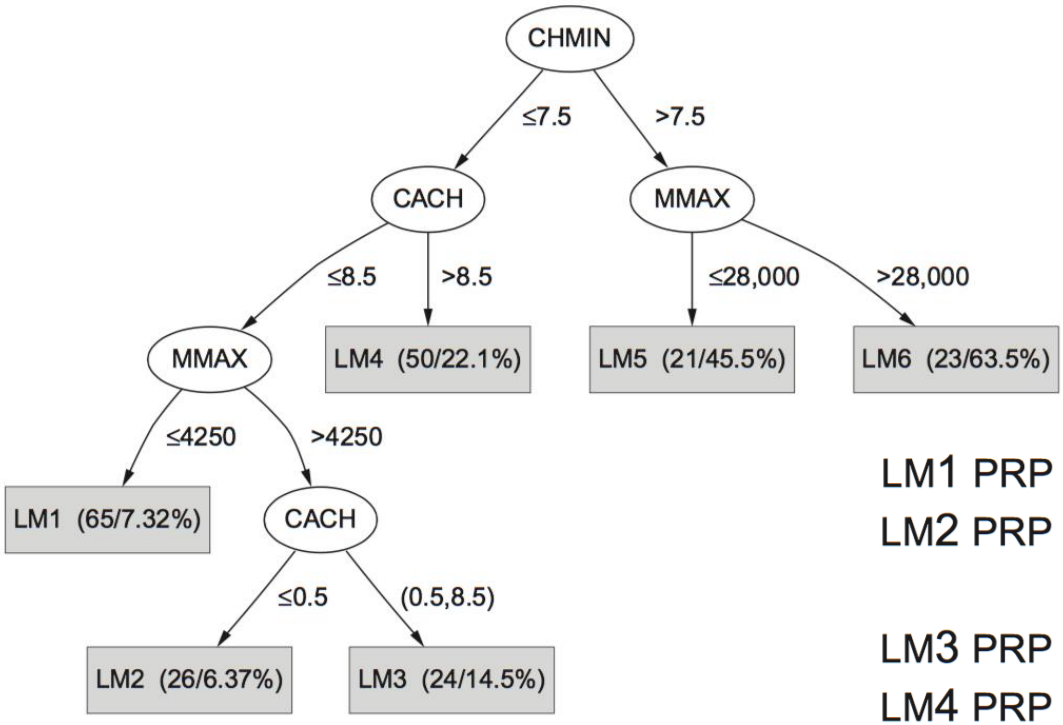
- ***Model tree:*** "regression tree" with linear regression models at the leaf nodes
  - Linear patches approximate continuous function



+

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

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

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)

If **age = young** and **tear-production = high** then **lenses = soft**

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

If **age = young** and **tear-production = high** ther **lenses = soft**

- Individual rules are often logically ORed together
  - Conflicts arise if different conclusions apply

To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

Can any decision tree be encoded using a set of classification rules?

Yes

No

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

To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

Can any set of classification rules be encoded as a decision tree?

Yes

No

0

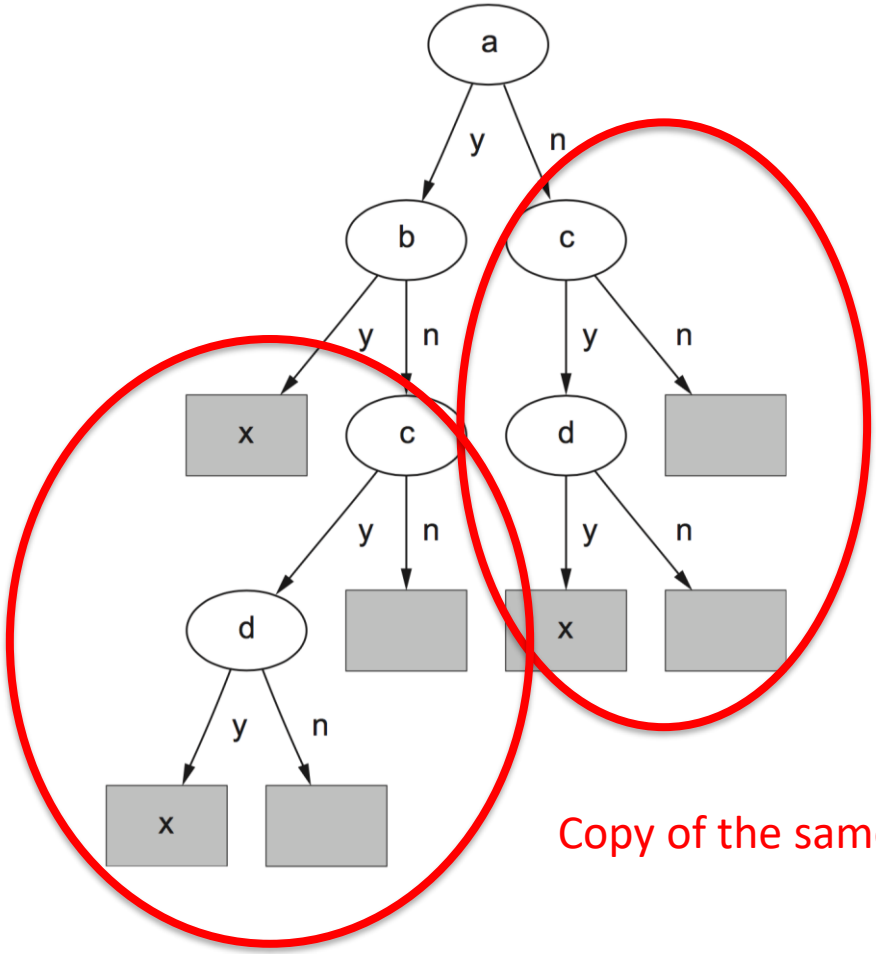
From rules to trees

- 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")

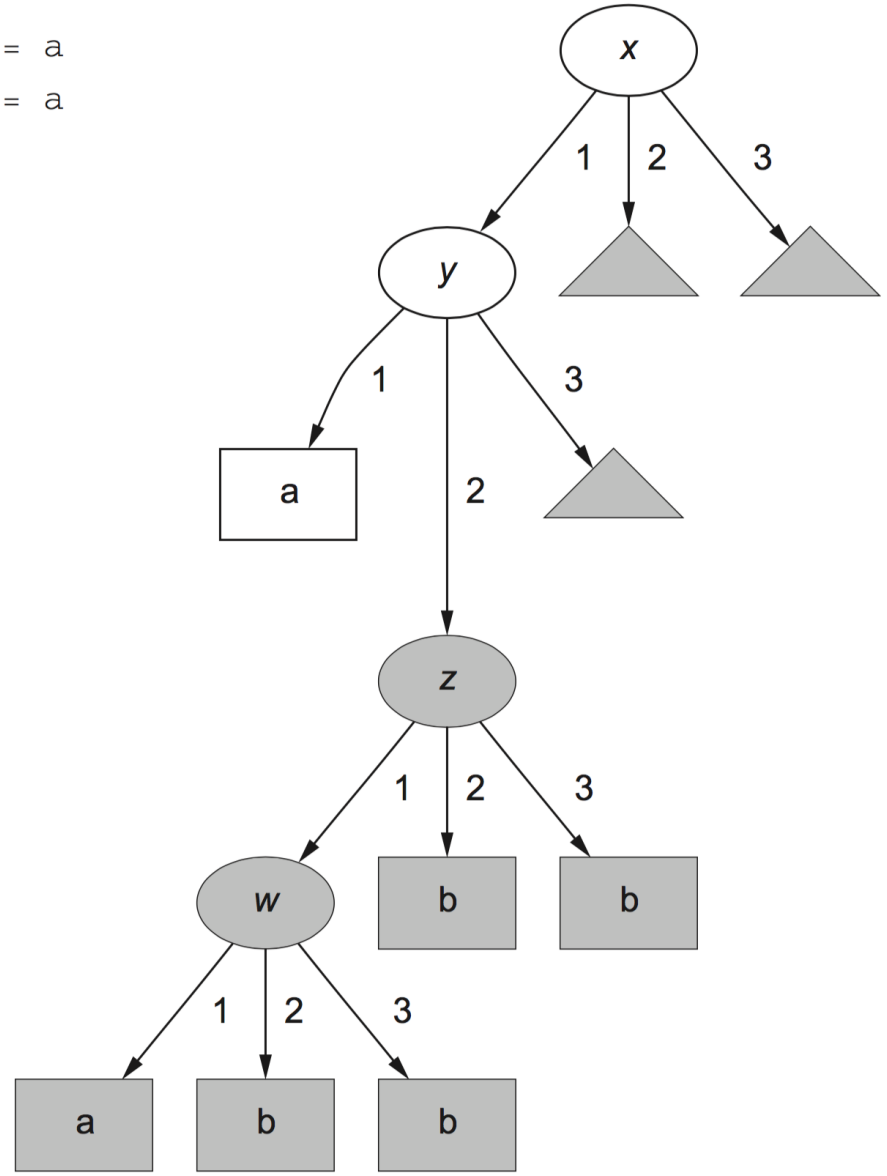
A tree for a simple disjunction



Copy of the same subtree!

A bigger 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
```



To show this poll

1

Install the app from pollev.com/app

2

Start the presentation

Still not working? Get help at pollev.com/app/help

or

Open poll in your web browser

Poll locked. Responses not accepted.

How many internal nodes (i.e. nodes that are not leaves) are needed for the smallest decision tree representing XOR (exclusive-or)?

3

4

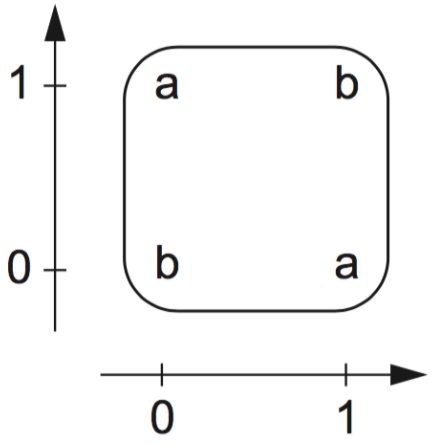
5

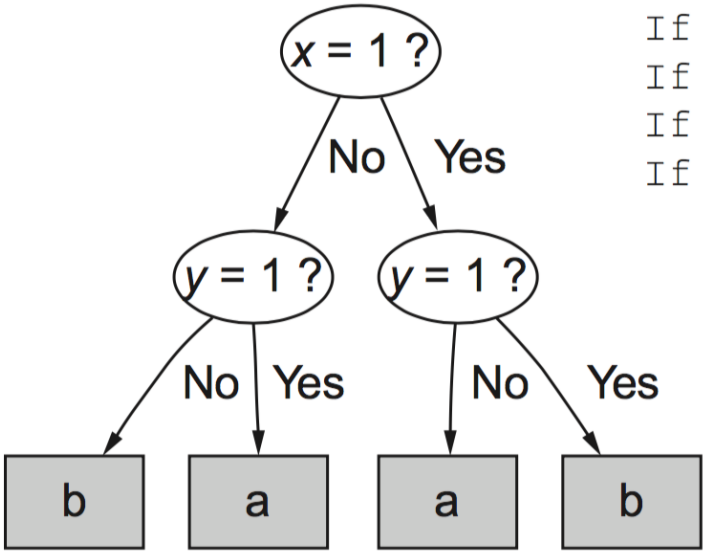
8

16

0

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!
- 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
    $\Rightarrow$ 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)

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
```

A more complex example

- Exceptions to exceptions to exceptions …

```
default: Iris-setosa
except if petal-length ≥ 2.45 and petal-length < 5.355
          and petal-width < 1.75
       then Iris-versicolor
            except if petal-length ≥ 4.95 and petal-width < 1.55
                   then Iris-virginica
                   else if sepal-length < 4.95 and sepal-width ≥ 2.45
                        then Iris-virginica
       else if petal-length ≥ 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 do not offer this advantage

More on exceptions

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

  is logically equivalent to

  `if...then...else`

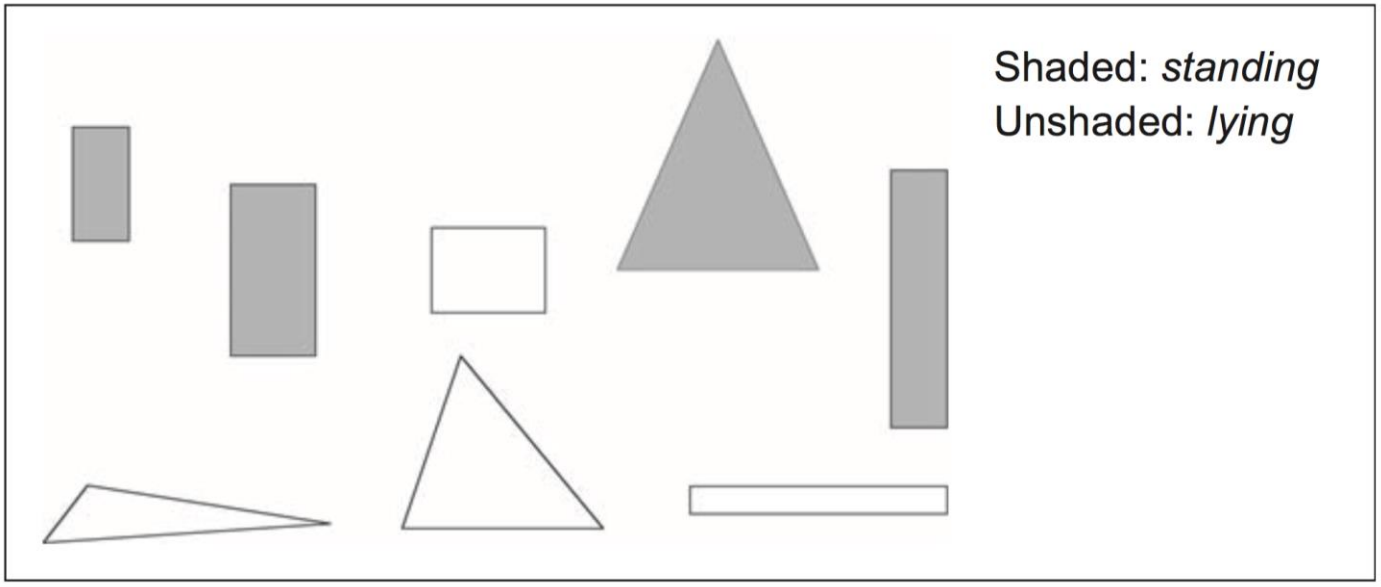
  (where the “else” specifies what the “default” does)
- But: exceptions offer a psychological advantage
  - Assumption: defaults and tests early on apply more widely than exceptions further down
  - Exceptions reflect special cases

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
```

Using relations between attributes

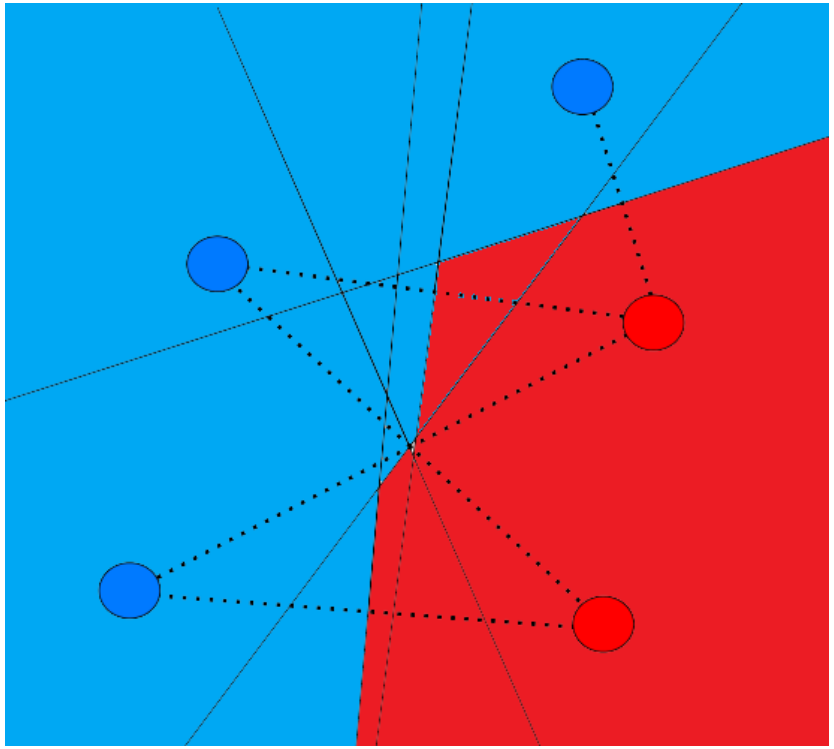
- Comparing attributes with each other enables rules like this:

```
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



Instance-based representation

- Instance-based learning is *lazy* learning
  - You don't have to do any work until you use the classifier! Nothing is trained

- Methods:
  - ***nearest-neighbor.*** Predict label of closest example
  - ***k-nearest-neighbor.*** Find *k* nearest examples, average their predictions

- Similarity function defines what's "learned"

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
  - Preprocessing: normalization to get the attributes on the same scale

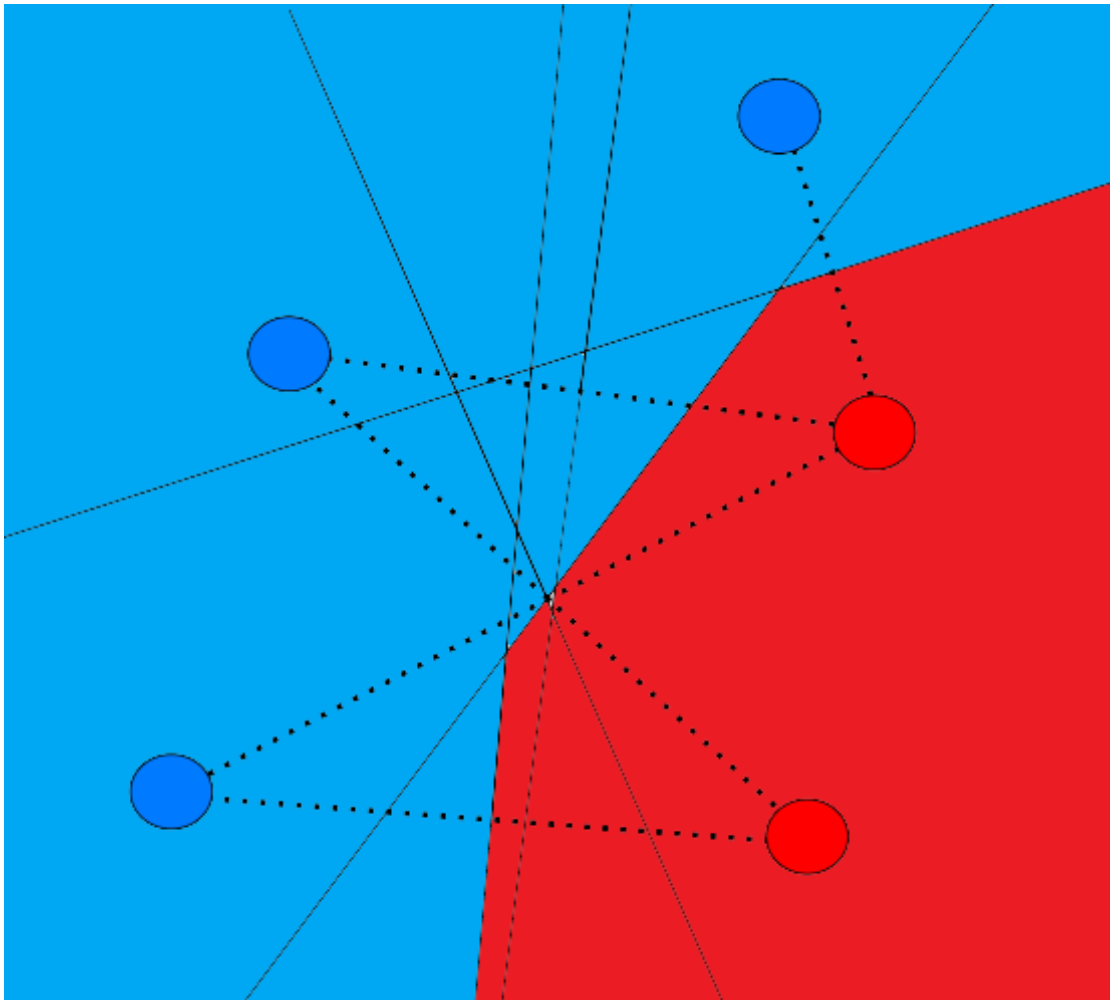
Nearest neighbor classifier

- Decision boundary (numeric data, Euclidean distance):

A subset of the lines in a Voronoi diagram.

Simpler computation:

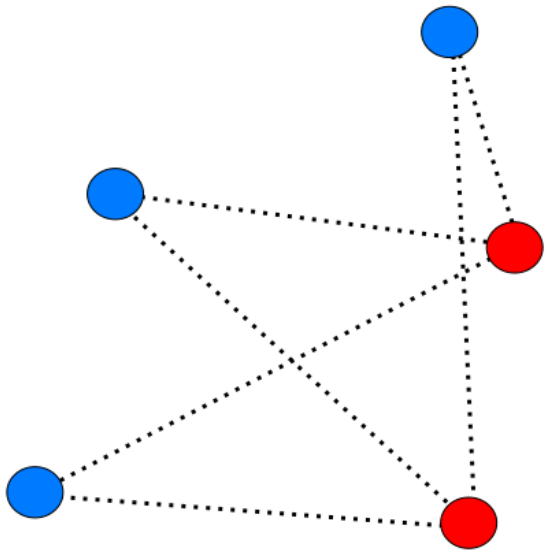
Space is divided by set of lines representing the **perpendicular bisectors** of pairs of data points from each class



Nearest neighbor classifier

- Decision boundary (numeric data, Euclidean distance):

- Step 1: draw lines connecting data points from different classes

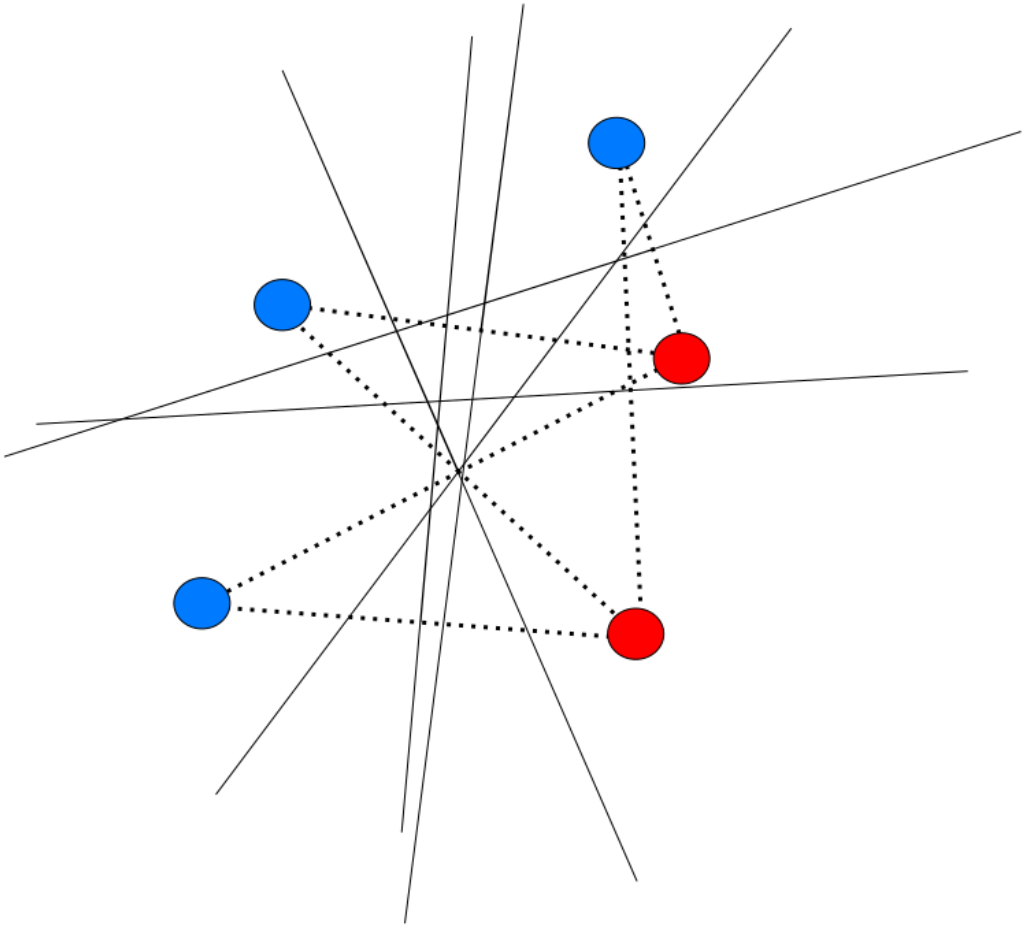


Nearest neighbor classifier

- Decision boundary (numeric data, Euclidean distance):

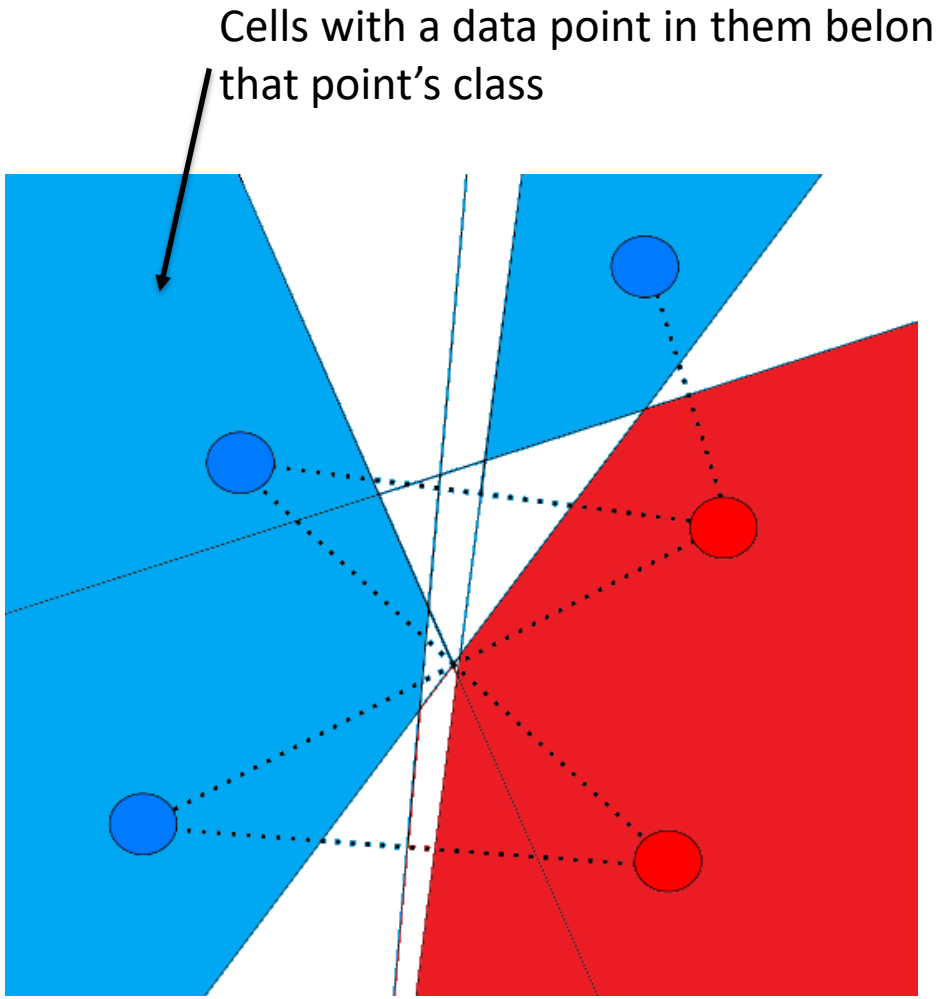
- Step 1: draw lines connecting data points from different classes

- Step 2: draw perpendicular bisectors through the middle of the connecting lines



Nearest neighbor classifier

- Decision boundary (numeric data, Euclidean distance):

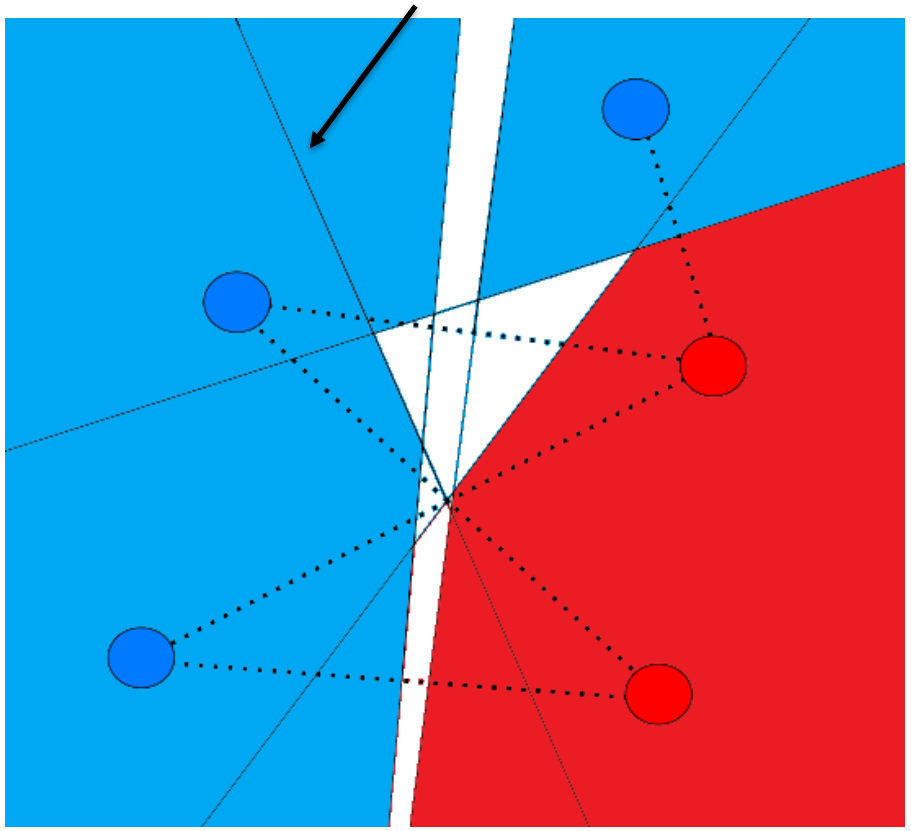


- Step 1: draw lines connecting data points from different classes
- Step 2: draw perpendicular bisectors through the middle of the connecting lines
- Step 3: associate each region with class of closest point. Decision boundary is the edges where the class swaps

Nearest neighbor classifier

- Decision boundary (numeric data, Euclidean distance):

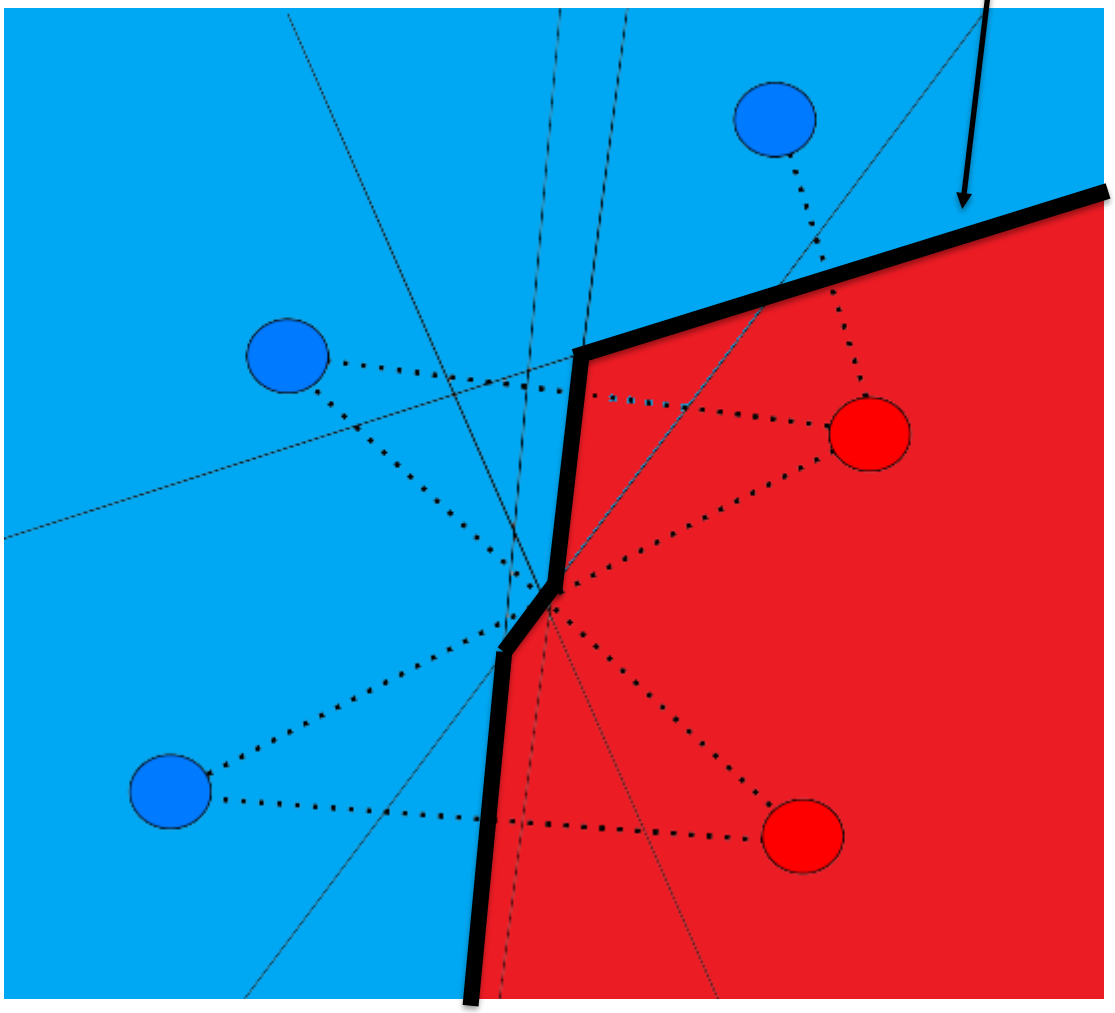
Expand the shaded region. Check whether the class flips when you cross the line.
(is the nearest point in the same class or not?)



- Step 1: draw lines connecting data points from different classes
- Step 2: draw perpendicular bisectors through the middle of the connecting lines
- Step 3: associate each region with class of closest point. Decision boundary is the edges where the class swaps

Nearest neighbor classifier

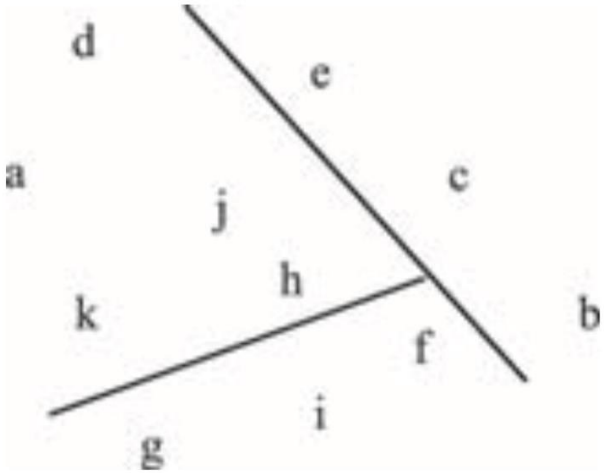
- Decision boundary (numeric data, Euclidean distance):



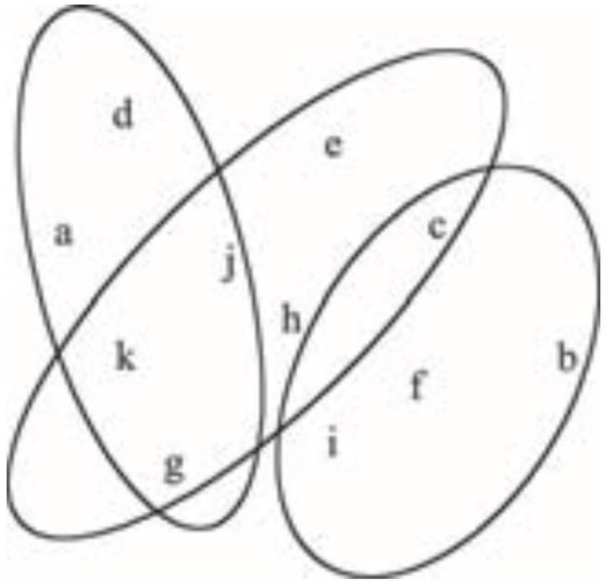
- Step 1: draw lines connecting data points from different classes
- Step 2: draw perpendicular bisectors through the middle of the connecting lines
- Step 3: associate each region with class of closest point. Decision boundary is the edges where the class swaps

Representing clusters I

Simple 2-D representation



Venn diagram

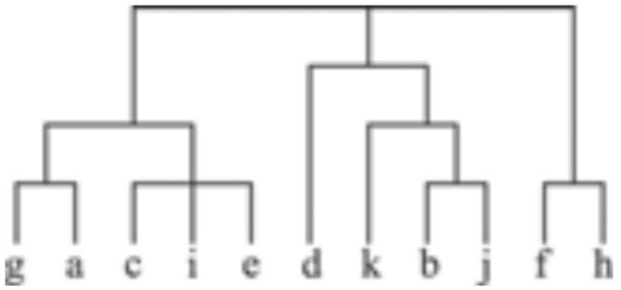


Representing clusters II

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



Think-pair-share: Whether to eat at a restaurant

- You are standing outside a restaurant, and you are deciding whether to go inside and eat.
  - What factors will you consider in your decision? (i.e. what attributes / features will you use)
  - Using these attributes, design a decision tree to encode your decision-making process
  - Now, design a linear model for making this decision. Pick the weights intuitively, relative to importance of the factors / attributes. Does it make similar decisions to your decision tree?