More Data Mining with Weka

Class 3 – Lesson 1

Decision trees and rules

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

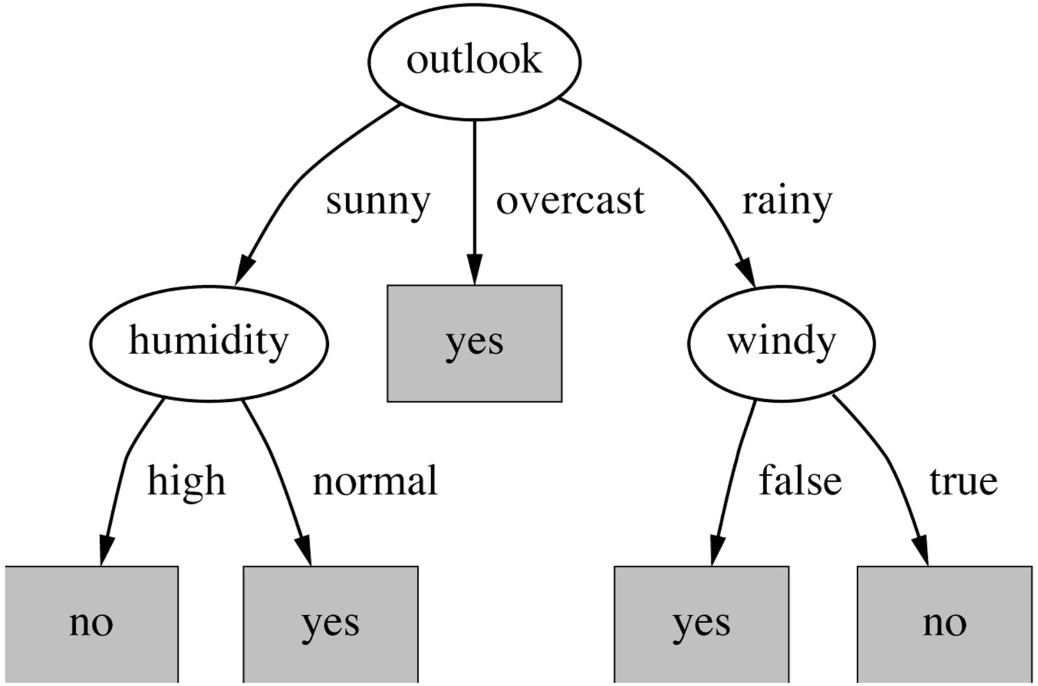
weka.waikato.ac.nz

Lesson 3.1: Decision trees and rules

- Class 1 **Exploring Weka's interfaces; working with big data**
- Class 2 **Discretization and text classification**
- Class 3 **Classification rules, association rules, and clustering**
  - **Lesson 3.1 Decision trees and rules**
  - Lesson 3.2 Generating decision rules
  - Lesson 3.3 Association rules
  - Lesson 3.4 Learning association rules
  - Lesson 3.5 Representing clusters
  - Lesson 3.6 Evaluating clusters
- Class 4 **Selecting attributes and counting the cost**
- Class 5 **Neural networks, learning curves, and performance optimization**

Lesson 3.1: Decision trees and rules

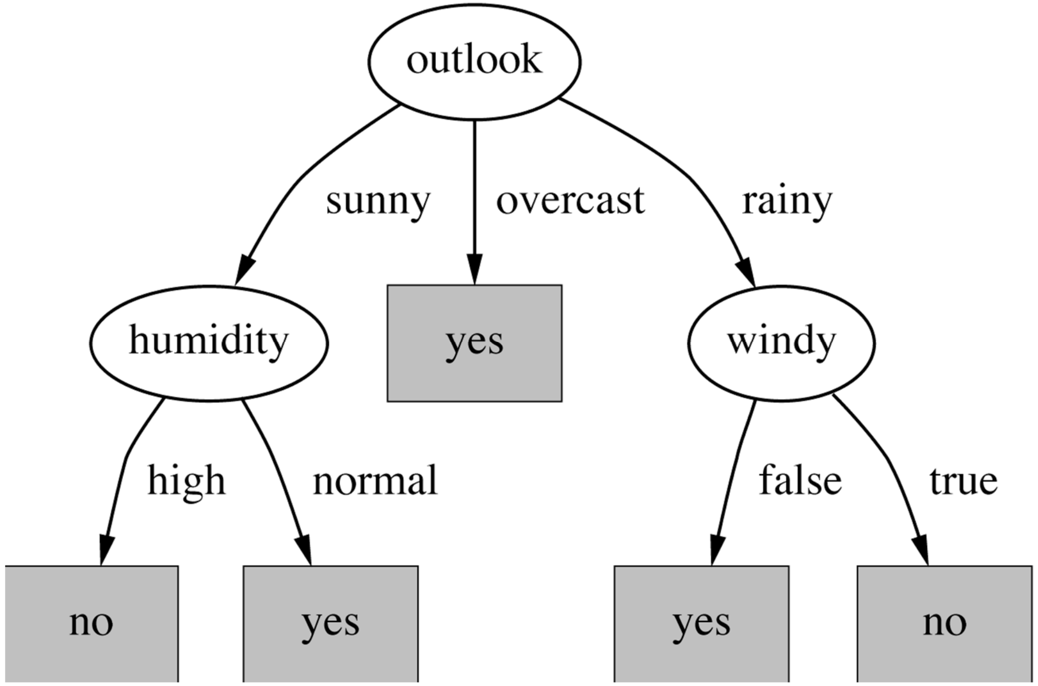
For any decision tree you can read off an equivalent set of rules



If outlook = sunny and humidity = high then no
If outlook = sunny and humidity = normal then yes
if outlook = overcast then yes
if outlook = rainy and windy = false then yes
if outlook = rainy and windy = true then no

Lesson 3.1: Decision trees and rules

For any decision tree you can read off an equivalent set of ordered rules ("decision list")



If outlook = sunny and humidity = high then no
If outlook = sunny ~~and humidity = normal~~ then yes
if outlook = overcast then yes
if ~~outlook = rainy and~~ windy = false then yes
~~if outlook = rainy and windy = true then~~ no

but rules from the tree are overly complex:

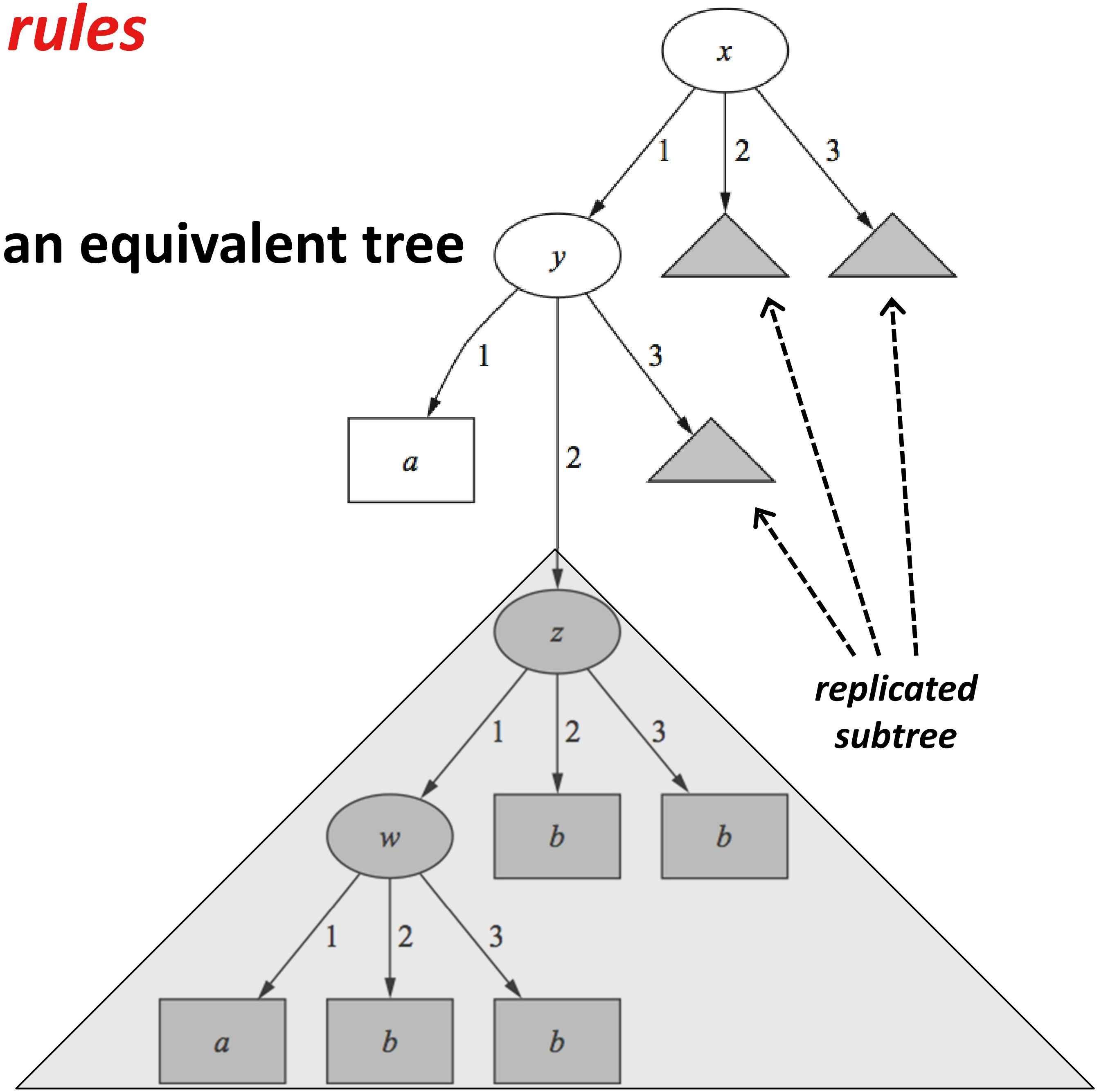
If outlook = sunny and humidity = high then no
if outlook = rainy and windy = true then no
otherwise yes

Lesson 3.1: Decision trees and rules

For any set of rules there is an equivalent tree

but it might be very complex

if x = 1 and y = 1 then a
if z = 1 and w = 1 then a
otherwise b



Lesson 3.1: Decision trees and rules

- Theoretically, rules and trees have equivalent “descriptive power”
- But practically they are very different

  … because rules are usually expressed as a decision list, to be executed sequentially, in order, until one “fires”

- People like rules: they’re easy to read and understand
- It’s tempting to view them as independent “nuggets of knowledge”
- … but that’s misleading
  - *when rules are executed sequentially*
    each one must be interpreted in the context of its predecessors

Lesson 3.1: Decision trees and rules

- **Create a decision tree (top-down, divide-and-conquer); read rules off the tree**
  - *One rule for each leaf*
  - *Straightforward, but rules contain repeated tests and are overly complex*
  - *More effective conversions are not trivial*

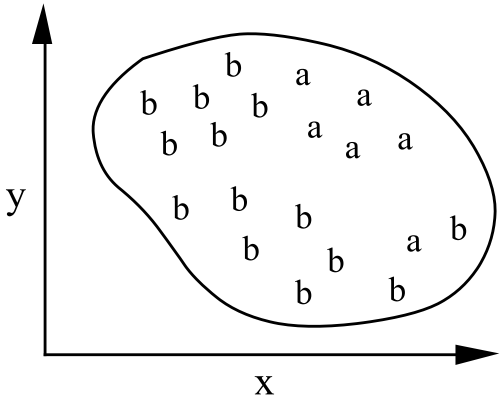
- **Alternative: covering method (bottom-up, separate-and-conquer)**
  - *For each class in turn find rules that cover all its instances (excluding instances not in the class)*

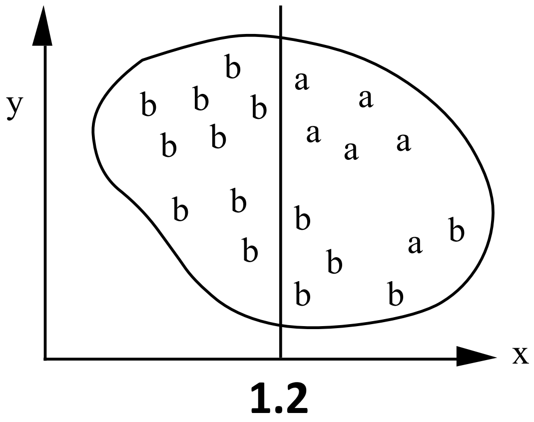
1. *Identify a useful rule*
2. *Separate out all the instances it covers*
3. *Then "conquer" the remaining instances in that class*

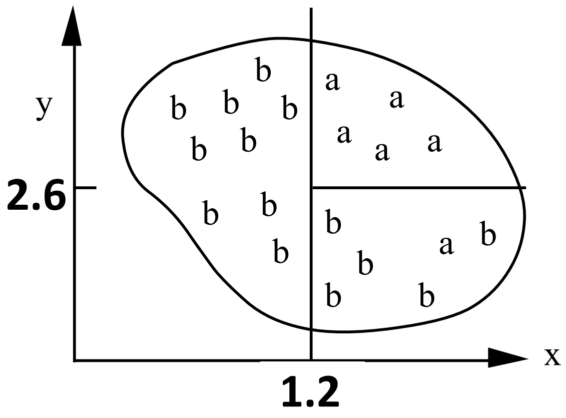
Lesson 3.1: Decision trees and rules

Generating a rule

- Generating a rule for class *a*







if *true*
  then class = *a*

if x > 1.2
  then class = *a*

if x > 1.2 and y > 2.6
  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*

- Could add more rules, get "perfect" rule set

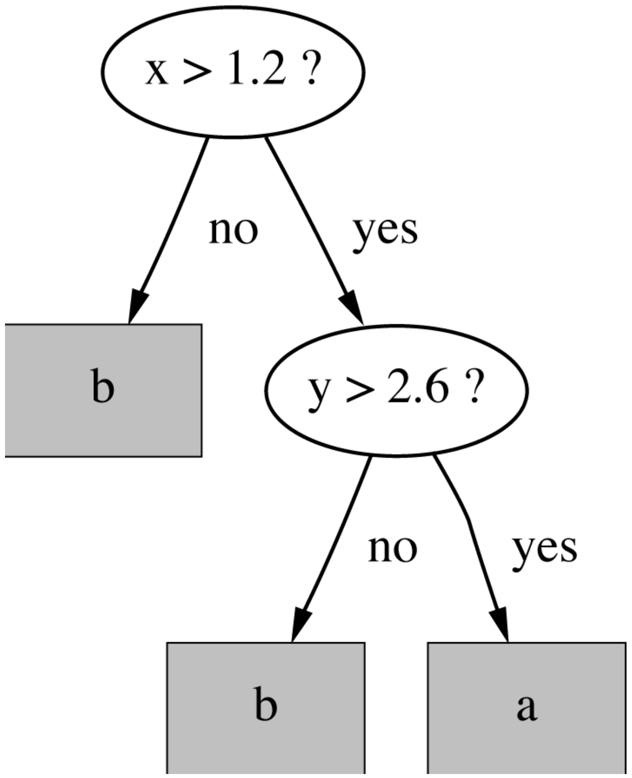
Lesson 3.1: Decision trees and rules

Rules vs. trees

❖ Corresponding decision tree
  – *produces exactly the same predictions*

❖ Rule sets *can* be more perspicuous
  – *E.g. when decision trees contain replicated subtrees*

❖ Also: in multiclass situations,
  – *covering algorithm concentrates on one class at a time*
  – *decision tree learner takes all classes into account*

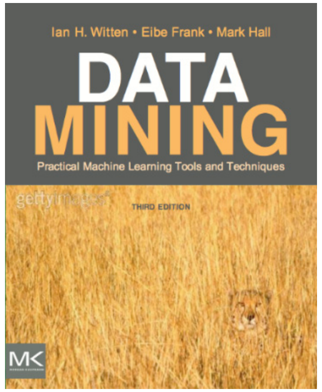


Simple bottom-up covering algorithm for creating rules: PRISM

```
For each class C
    Initialize E to the instance set
    While E contains instances in class C
        Create a rule R that predicts class C
            (with empty left-hand side)
        Until R is perfect
            (or there are no more attributes to use)
            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
                    (break ties by choosing the condition with the largest p)
            Add A = v to R
        Remove the instances covered by R from E
```

Lesson 3.1: Decision trees and rules

- Decision trees and rules have the same expressive power
  … but either can be more perspicuous than the other
- Rules can be created using a bottom-up covering process
- Rule sets are often “decision lists”, to be executed in order
  - if rules assign different classes to an instance, the first rule wins
  - rules are not really independent “nuggets of knowledge”
- Still, people like rules and often prefer them to trees



Course text

- Section 4.4 *Covering algorithms: constructing rules*

More Data Mining with Weka

Class 3 – Lesson 2

Generating decision rules

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 3.2: Generating decision rules

Class 1 **Exploring Weka's interfaces; working with big data**

Class 2 **Discretization and text classification**

Class 3 **Classification rules, association rules, and clustering**

Class 4 **Selecting attributes and counting the cost**

Class 5 **Neural networks, learning curves, and performance optimization**

Lesson 3.1 Decision trees and rules

Lesson 3.2 Generating decision rules

Lesson 3.3 Association rules

Lesson 3.4 Learning association rules

Lesson 3.5 Representing clusters

Lesson 3.6 Evaluating clusters

Lesson 3.2: Generating decision rules

1. Rules from partial decision trees: PART

- Make a rule
- Remove the instances it covers
- Continue, creating rules for the remaining instances

Separate
and conquer

To make a rule, build a tree!

- Build and prune a decision tree for the current set of instances
- Read off the rule for the largest leaf
- Discard the tree (!)

(can build just a partial tree, instead of a full one)

Lesson 3.2: Generating decision rules

2. Incremental reduced-error pruning

Split the instance set into *Grow* and *Prune* in the ratio 2:1
For each class *C*
  While *Grow* and *Prune* both contain instances in *C*
    On *Grow*, use PRISM to create the best perfect rule for *C*
    Calculate the worth $w(R)$ for the rule on *Prune*,
      and of the rule with the final condition omitted $w(R–)$
      While $w(R–) > w(R)$, remove the final condition from the rule
        and repeat the previous step
    Print the rule; remove the instances it covers from *Grow* and *Prune*

"worth":
  success rate?
  something more complex?

… followed by a fiendishly complicated global optimization step – RIPPER

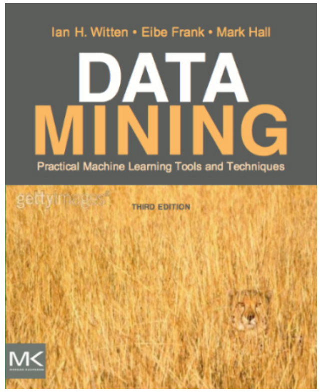
Diabetes dataset

- J48 74% 39-node tree
- PART 73% 13 rules (25 tests)
- JRip 76% 4 rules (9 tests)

plas ≥ 132 and mass ≥ 30 –> tested_positive
age ≥ 29 and insu ≥ 125 and preg ≤ 3 –> tested_positive
age ≥ 31 and pedi ≥ 0.529 and preg ≥ 8 and mass ≥ 25.9 –> tested_positive
–> tested_negative

Lesson 3.2: Generating decision rules

- PART is quick and elegant
  - repeatedly constructing decision trees and discarding them is less wasteful than it sounds
- Incremental reduced-error pruning is a standard technique
  - using Grow and Prune sets
- Ripper (JRip) follows this by complex global optimization
  - makes rules that classify all class values except the majority one
  - last rule is a default rule, for the majority class
  - usually produces fewer rules than PART



Course text

- Section 6.2 *Classification rules*


SHARKS
CROCODILES
STINGRAYS
ANGRY NATIVES
STEVE IRWIN
BACKPACKER MURDERERS
GIANT RATS
POISONOUS SNAKES
NOTHING
MANEATING KOALAS
SCORPIONS
MOSQUITOES
BABYKILLER DINGOS
SCORCHING DESERT
SHARKS
SHARKS
DANNI MINOGUE
FIRES
SHARKS WITH FRICKIN' LASERS
STINGING JELLYFISH
GIANT SPIDERS
CONVICTS
DERANGED

More Data Mining with Weka

Class 3 – Lesson 3

Association rules

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 3.3: Association rules

- Class 1 **Exploring Weka's interfaces; working with big data**
- Class 2 **Discretization and text classification**
- Class 3 **Classification rules, association rules, and clustering**
- Class 4 **Selecting attributes and counting the cost**
- Class 5 **Neural networks, learning curves, and performance optimization**

- **Lesson 3.1 Decision trees and rules**
- **Lesson 3.2 Generating decision rules**
- **Lesson 3.3 Association rules**
- **Lesson 3.4 Learning association rules**
- **Lesson 3.5 Representing clusters**
- **Lesson 3.6 Evaluating clusters**

Lesson 3.3: Association rules

- With association rules, there is no “class” attribute
- Rules can predict any attribute, or combination of attributes
- Need a different kind of algorithm: “**Apriori**”

Here are some association rules for the weather data:

1. outlook = overcast	==>	play = yes
2. temperature = cool	==>	humidity = normal
3. humidity = normal & windy = false	==>	play = yes
4. outlook = sunny & play = no	==>	humidity = high
5. outlook = sunny & humidity = high	==>	play = no
6. outlook = rainy & play = yes	==>	windy = false
7. outlook = rainy & windy = false	==>	play = yes
8. temperature = cool & play = yes	==>	humidity = normal
9. outlook = sunny & temperature = hot	==>	humidity = high
10. temperature = hot & play = no	==>	outlook = sunny

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

Lesson 3.3: Association rules

- **Support:** number of instances that satisfy a rule
- **Confidence:** proportion of instances that satisfy the left-hand side for which the right-hand side also holds
- Specify minimum confidence, seek the rules with greatest support??

			support	confidence
1. outlook = overcast	==>	play = yes	4	100%
2. temperature = cool	==>	humidity = normal	4	100%
3. humidity = normal & windy = false	==>	play = yes	4	100%
4. outlook = sunny & play = no	==>	humidity = high	3	100%
5. outlook = sunny & humidity = high	==>	play = no	3	100%
6. outlook = rainy & play = yes	==>	windy = false	3	100%
7. outlook = rainy & windy = false	==>	play = yes	3	100%
8. temperature = cool & play = yes	==>	humidity = normal	3	100%
9. outlook = sunny & temperature = hot	==>	humidity = high	2	100%
10. temperature = hot & play = no	==>	outlook = sunny	2	100%

Lesson 3.3: Association rules

- **Itemset** set of attribute-value pairs, e.g.

  humidity = normal & windy = false & play = yes — support = 4

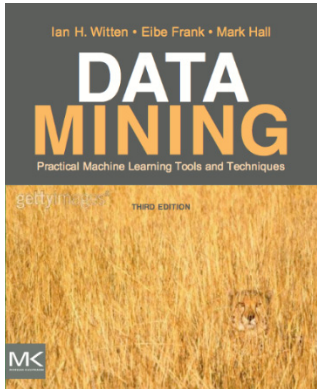
- 7 potential rules from this itemset:

Rule	support	confidence
If humidity = normal & windy = false ==> play = yes	4	4/4
If humidity = normal & play = yes ==> windy = false	4	4/6
If windy = false & play = yes ==> humidity = normal	4	4/6
If humidity = normal ==> windy = false & play = yes	4	4/7
If windy = false ==> humidity = normal & play = yes	4	4/8
If play = yes ==> humidity = normal & windy = false	4	4/9
==> humidity = normal & windy = false & play = yes	4	4/14

- Generate high-support itemsets, get several rules from each
- Strategy: iteratively reduce the minimum support until the required number of rules is found with a given minimum confidence

Lesson 3.3: Association rules

- There are far more association rules than classification rules
  - need different techniques
- *Support* and *Confidence* are measures of a rule
- Apriori is the standard association-rule algorithm
- Want to specify minimum confidence value and seek rules with the most support
- Details? – see next lesson



Course text

- Section 4.5 *Mining association rules*

More Data Mining with Weka

Class 3 – Lesson 4

Learning association rules

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 3.4: Learning association rules

Class 1 **Exploring Weka's interfaces; working with big data**

Class 2 **Discretization and text classification**

Class 3 **Classification rules, association rules, and clustering**

Class 4 **Selecting attributes and counting the cost**

Class 5 **Neural networks, learning curves, and performance optimization**

Lesson 3.1 Decision trees and rules

Lesson 3.2 Generating decision rules

Lesson 3.3 Association rules

Lesson 3.4 Learning association rules

Lesson 3.5 Representing clusters

Lesson 3.6 Evaluating clusters

Lesson 3.4: Learning association rules

Strategy

- *specify minimum confidence*
- *iteratively reduce support until enough rules are found with > this confidence*

7 potential rules from a single itemset:

	support	confidence
If humidity = normal & windy = false ==> play = yes	4	4/4
If humidity = normal & play = yes ==> windy = false	4	4/6
If windy = false & play = yes ==> humidity = normal	4	4/6
If humidity = normal ==> windy = false & play = yes	4	4/7
If windy = false ==> humidity = normal & play = yes	4	4/8
If play = yes ==> humidity = normal & windy = false	4	4/9
==> humidity = normal & windy = false & play = yes	4	4/14

1. Generate itemsets with support 14 (none)
2. find rules with > min confidence level (Weka default: 90%)
3. continue with itemsets with support 13 (none)

… and so on, until sufficient rules have been generated

Lesson 3.4: Learning association rules

- Weather data has 336 rules with confidence 100%!
  - *but only 8 have support ≥ 3, only 58 have support ≥ 2*
- Weka: specify minimum confidence level (minMetric, default 90%)

  number of rules sought (numRules, default 10)
- Support is expressed as a proportion of the number of instances
- Weka runs Apriori algorithm several times

  starts at upperBoundMinSupport (usually left at 100%)

  decreases by delta at each iteration (default 5%)

  stops when numRules reached

  … or at lowerBoundMinSupport (default 10%)

Lesson 3.4: Learning association rules

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

- 17 cycles of Apriori algorithm:
  - *support = 100%, 95%, 90%, …, 20%, 15%*
  - *14, 13, 13, …, 3, 2 instances*
  - *only 8 rules with conf > 0.9 & support ≥ 3*
- to see itemsets, set outputItemSets
  - *they're based on the final support value, i.e. 2*

12 one-item sets with support ≥ 2

outlook = sunny 5
outlook = overcast 4
...
play = no 5

47 two-item sets with support ≥ 2

outlook = sunny & temperature = hot 2
outlook = sunny & humidity = high 3
...

39 three-item sets with support ≥ 2

outlook = sunny & temperature = hot & humidity = high 2
outlook = sunny & humidity = high & play = no 3
outlook = sunny & windy = false & play = no 2
...

6 four-item sets with support ≥ 2

outlook = sunny & humidity = high & windy = false
& play = no 2
...

Lesson 3.4: Learning association rules

Other parameters in Weka implementation

- car: always produce rules that predict the class attribute
  - *set the class attribute using classIndex*
- significanceLevel: filter rules according to a statistical test (χ^2)
  - *unreliable because with so many tests, significant results will be found just by chance*
  - *the test is inaccurate for small support values*
- metricType: different measures for ranking rules
  - *Confidence*
  - *Lift*
  - *Leverage*
  - *Conviction*
- removeAllMissingCols: removes attribute whose values are all “missing”

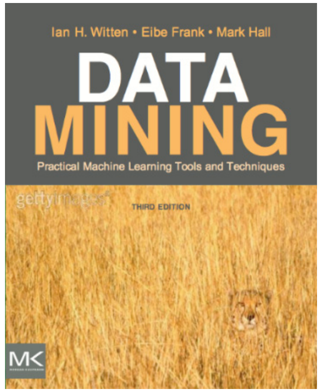
Lesson 3.4: Learning association rules

Market basket analysis

- Look at supermarket.arff
  - *collected from an actual New Zealand supermarket*
- 4500 instances, 220 attributes; 1M attribute values
- Missing values used to indicate that the basket did not contain that item
- 92% of values are missing
  - *average basket contains 220×8% = 18 items*
- Most popular items: bread-and-cake (3330), vegetables (2961), frozen foods (2717), biscuits (2605)

Lesson 3.4: Learning association rules

- Apriori makes multiple passes through the data
  - generates 1-item sets, 2-item sets, … with more than minimum support
  - turns each one into (many) rules and checks their confidence
- Fast and efficient (provided data fits into main memory)
- Weka invokes Apriori several times gradually reducing the support until sufficient high-confidence rules have been found
  - there are parameters to control this
- Activity: supermarket data



Course text

- Section 11.7 *Association-rule learners*

More Data Mining with Weka

Class 3 – Lesson 5

Representing clusters

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 3.5: Representing clusters

- Class 1 **Exploring Weka's interfaces; working with big data**
- Class 2 **Discretization and text classification**
- Class 3 **Classification rules, association rules, and clustering**
- Class 4 **Selecting attributes and counting the cost**
- Class 5 **Neural networks, learning curves, and performance optimization**

Lesson 3.1 Decision trees and rules

Lesson 3.2 Generating decision rules

Lesson 3.3 Association rules

Lesson 3.4 Learning association rules

Lesson 3.5 Representing clusters

Lesson 3.6 Evaluating clusters

Lesson 3.5: Representing clusters

- With clustering, there is no “class” attribute
- Try to divide the instances into natural groups, or “clusters”

Example

- Examine iris.arff in the Explorer
- Imagine deleting the class attribute
- Could you recover the classes by clustering the data?

Iris Setosa

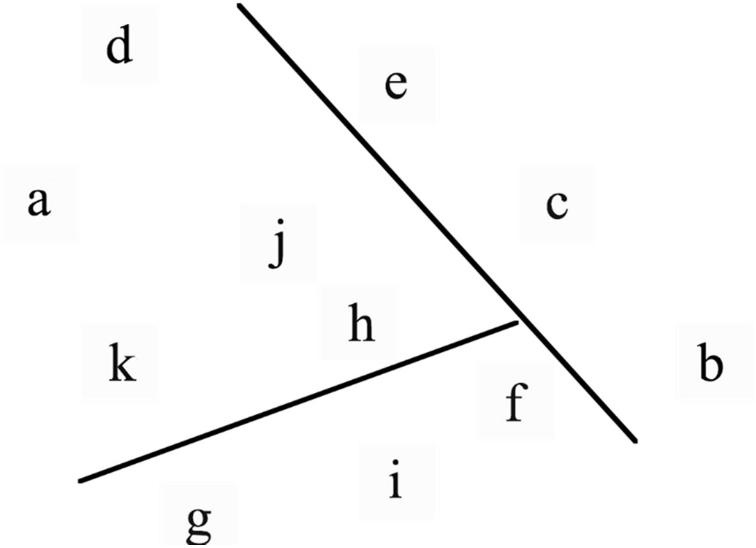
Iris Versicolor

Iris Virginica

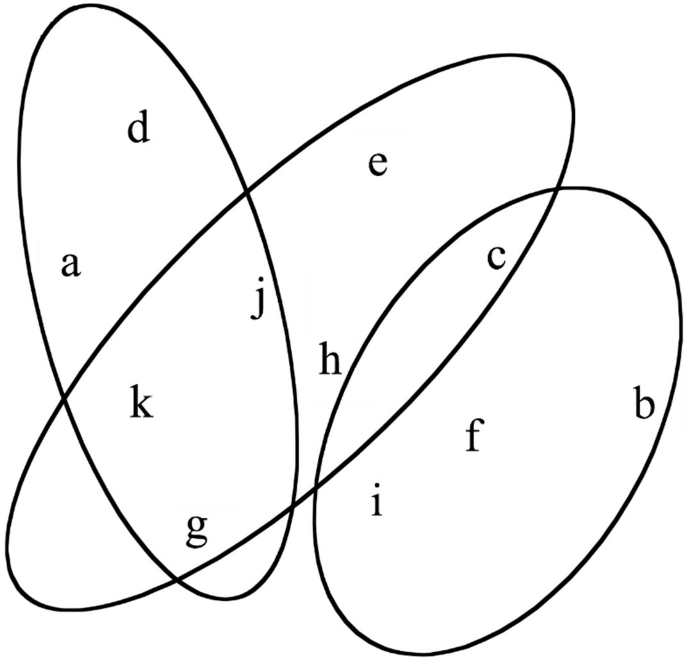
Lesson 3.5: Representing clusters

Cluster types

1. Disjoint sets



2. Overlapping sets



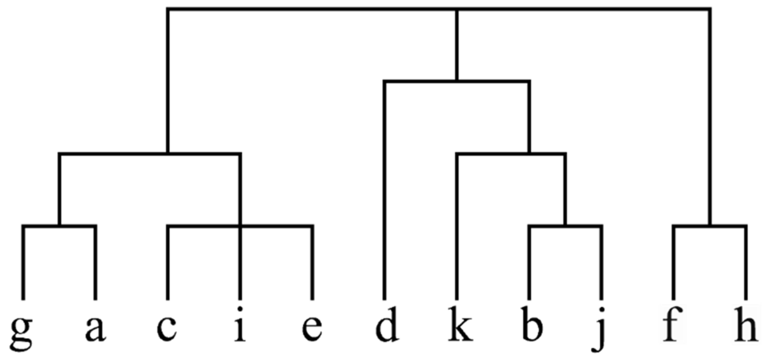
Lesson 3.5: Representing clusters

Cluster types

3. Probabilistic clusters

	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
…			

4. Hierarchical clusters



Lesson 3.5: Representing clusters

KMeans: Iterative distance-based clustering (disjoint sets)

1. Specify *k*, the desired number of clusters
2. Choose *k* points at random as cluster centers
3. Assign all instances to their closest cluster center
4. Calculate the centroid (i.e., mean) of instances in each cluster
5. These centroids are the new cluster centers
6. Continue until the cluster centers don’t change

Minimizes the total squared distance from instances to their cluster centers

Local, not global, minimum!

Lesson 3.5: Representing clusters

KMeans clustering

- Open weather.numeric.arff
- Cluster panel; choose SimpleKMeans
- Note parameters: numClusters, distanceFunction, seed (default 10)

- Two clusters, 9 and 5 members, total squared error 16.2

  {1/no, 2/no, 3/yes, 4/yes, 5/yes, 8/no, 9/yes, 10/yes, 13/yes} {6/no, 7/yes. 11/yes, 12/yes, 14/no}
- Set seed to 11
- Two clusters, 6 and 8 members, total squared error 13.6
- Set seed to 12
- Total squared error 17.3

Lesson 3.5: Representing clusters

XMeans: Extended version of KMeans

- Selects the number of clusters itself
- Can specify the min/max number of clusters
- Can specify four different distance metrics
- Can use kD-trees for speed

Cannot handle nominal attributes

- Ignore nominal attributes in weather data

  outlook, windy, play

Lesson 3.5: Representing clusters

EM clustering (probabilistic, uses "Expectation Maximization")

- Cluster panel; choose EM
- Change numClusters to 2 (–1 asks EM to determine the number)
- Note parameters: maxIterations, minStdDev, seed (default 100)

  restore nominal attributes

- Two clusters, prior probs 0.35 and 0.65
- Within each:

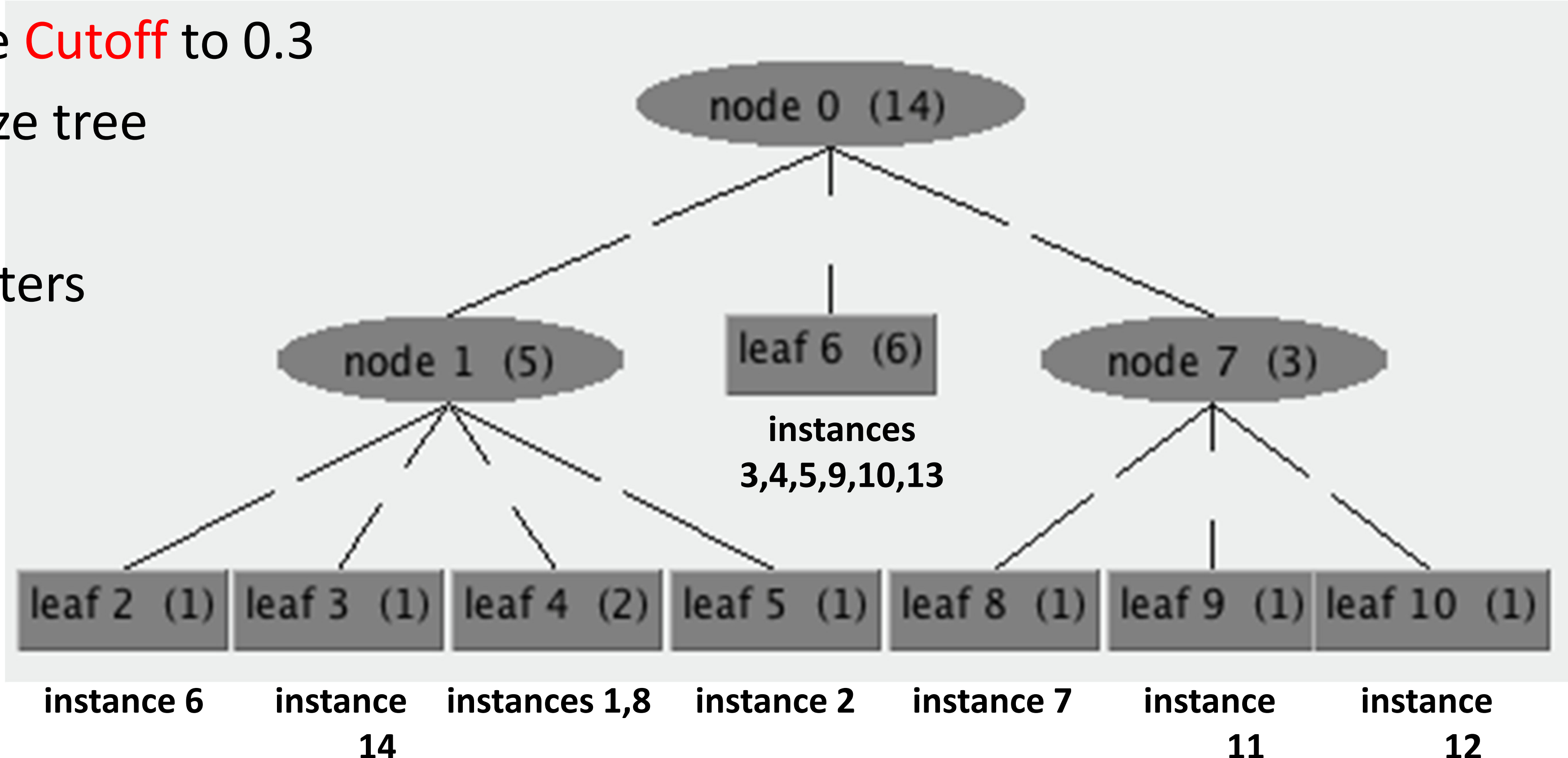
  nominal attributes: prob of each value
  numeric attributes: mean and std dev
- Can calculate the cluster membership prob for any instance
- Overall quality measure: log likelihood

Lesson 3.5: Representing clusters

Cobweb clustering (hierarchical)

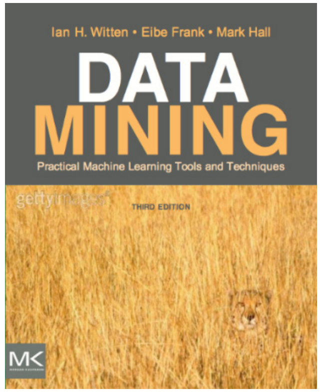
- Cluster panel; choose Cobweb
- Change Cutoff to 0.3
- Visualize tree

- 10 clusters



Lesson 3.5: Representing clusters

- Clustering: no class value
- Representations: disjoint sets, probabilistic, hierarchical
  - in Weka, SimpleKMeans (+XMeans), EM, Cobweb
- Kmeans: Iterative distance-based method
- Different distance metrics
- Hard to evaluate clustering



Course text

- Sections 4.8 and 6.8 *Clustering*

More Data Mining with Weka

Class 3 – Lesson 6

Evaluating clusters

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 3.6: Evaluating clusters

Class 1 **Exploring Weka's interfaces; working with big data**

Class 2 **Discretization and text classification**

Class 3 **Classification rules, association rules, and clustering**

Class 4 **Selecting attributes and counting the cost**

Class 5 **Neural networks, learning curves, and performance optimization**

Lesson 3.1 Decision trees and rules

Lesson 3.2 Generating decision rules

Lesson 3.3 Association rules

Lesson 3.4 Learning association rules

Lesson 3.5 Representing clusters

Lesson 3.6 Evaluating clusters

Lesson 3.6: Evaluating clusters

Visualizing clusters

- Iris data, SimpleKMeans, specify 3 clusters
  3 clusters with 50 instances each
- Visualize cluster assignments (right-click menu)
  Plot Cluster against Instance_number to see what the errors are
- Perfect? – surely not!
  Ignore class attribute; 3 clusters, with 61, 50, 39 instances

Which instances does a cluster contain?

- Use the AddCluster unsupervised attribute filter
- Try with SimpleKMeans; Apply and click Edit

Lesson 3.6: Evaluating clusters

Classes-to-clusters evaluation

- Iris data, SimpleKMeans, specify 3 clusters
- Classes to clusters evaluation

SimpleKMeans (3 clusters)

```
 0  1  2  <-- assigned to cluster
 0 50  0 | Iris-setosa
47  0  3 | Iris-versicolor
14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances:  17  11%
```

EM (3 clusters)

```
 0  1  2  <-- assigned to cluster
 0 50  0 | Iris-setosa
50  0  0 | Iris-versicolor
14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances:  14  9%
```

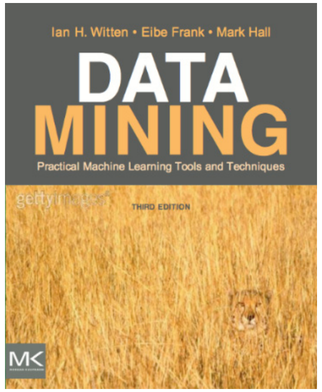
Lesson 3.6: Evaluating clusters

ClassificationViaClustering meta-classifier

- Create a classifier:
  - *Ignore classes*
  - *cluster*
  - *assign to each cluster its most frequent class*
- Obviously not competitive with other classification techniques
- Good way of comparing clusterers

Lesson 3.6: Evaluating clusters

- Hard to evaluate clustering
  - SimpleKMeans: Within-cluster sum of squared errors
  - Should really be evaluated with respect to an application
- Visualization
- AddCluster filter shows the instances in each cluster
- Classes to clusters evaluation
- Classification via clustering



Course text

- Section 11.2, under *Clustering and association rules*
- Section 11.6 *Clustering algorithms*

More Data Mining with Weka

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz