CHAPTER 4

Algorithms: The Basic Methods

Outline

- Inferring rudimentary rules: 1R
- Statistical modeling: Naive Bayes
- Divide-and-Conquer: constructing decision trees
- Covering algorithms: constructing rules
- Mining association rules
- Linear models
- Instance-based learning
- Clustering: k-means
- Agglomerative Clustering

Inferring Rudimentary Rules: 1R (1/6)

Find rules of one attribute only

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Attribute	Rules	Errors	Total Errors
outlook*	sunny -> no	2/5	4/14
	overcast -> yes	0/4	
	rainy -> yes	2/5	
temperature	hot -> no*	2/4	5/14
	mild -> yes	2/6	
	cool -> yes	1/4	
humidity	high -> no	3/7	4/14
	normal -> yes	1/7	
windy	false -> yes	2/8	5/14
	true -> no*	3/6	

5 training instances follow
& 2 false classification

one-level decision tree

Inferring Rudimentary Rules: 1R (2/6)

```
Scheme:weka.classifiers.rules.OneR -B 6
Relation:     weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:evaluate on training data

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

outlook:
        sunny    -> no
        overcast        -> yes
        rainy    -> yes
(10/14 instances correct)

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances         4               28.5714 %
Kappa statistic                          0.3778
Mean absolute error                      0.2857
Root mean squared error                  0.5345
Relative absolute error                 61.5385 %
Root relative squared error            111.4773 %
Total Number of Instances               14

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                   0.778     0.4       0.778     0.778     0.778       0.689    yes
                   0.6       0.222     0.6       0.6       0.6         0.689    no
Weighted Avg.      0.714     0.337     0.714     0.714     0.714       0.689

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 2 3 | b = no
```

Success Rate: (TP+TN)/(TP+TN+FP+FN)

Confusion matrix of a two-class prediction			
		Predicted class	
		yes	no
Actual class	yes	true positive (TP)	false negative (FN)
	no	false positive (FP)	true negative (TN)

Inferring Rudimentary Rules: 1R (3/6)

```
For each attribute,
  For each value of that attribute, make a rule as follows:
    count how often each class appears
    find the most frequent class
    make the rule assign that class to this attribute value.
  Calculate the error rate of the rules.
Choose the rules with the smallest error rate.
```

FIGURE 4.1

Pseudocode for 1R.

Inferring Rudimentary Rules: 1R (4/6)

- Simple rules frequently achieve surprisingly high accuracy on real-world DB
- Special handling
  - Missing values
    - Treated as another attribute value “missing”
  - Numeric attributes
    - Convert numeric attributes into nominal ones using a simple discretization method

set minimum of the majority class = 1

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

64.5 66.5 70.5 72 77.5 80.5 84

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

64.5 66.5 70.5 71.5 77.5 80.5 84

Table 1.3 Weather Data with Some Numeric Attributes

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	false	no
Sunny	80	90	true	no
Overcast	83	86	false	yes
Rainy	70	96	false	yes
Rainy	68	80	false	yes
Rainy	65	70	true	no
Overcast	64	65	true	yes
Sunny	72	95	false	no
Sunny	69	70	false	yes
Rainy	75	80	false	yes
Sunny	75	70	true	yes
Overcast	72	90	true	yes
Overcast	81	75	false	yes
Rainy	71	91	true	no

set minimum of the majority class = 3

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

yes yes no

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

yes no

64	65	68	69	70	71	72	72	75	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	yes	yes	no	yes	yes	no

if temperature <= 77.5 then play = yes
if temperature > 77.5 then play = no

```
=== Run information ===

Scheme:weka.classifiers.rules.OneR -B 6
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:evaluate on training data

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

outlook:
        sunny    -> no
        overcast        -> yes
        rainy    -> yes
(10/14 instances correct)


Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances         4               28.5714 %
```

For 1R, overfitting is likely to occur whenever an attribute has a large number of possible values.

minBucketSize -- The minimum bucket size used for discretizing numeric attributes.

Statistical Model: Naive Bayes (1/10)

- Supervised classification technique
- Assumption
  - All input attributes are equally important and independent of one another
- Bayes theorem

P(play = yes | outlook = sunny, temperature = cool, humidity = high, windy = true)

$$P(H \mid E) = \frac{P(E \mid H) \times P(H)}{P(E)}$$

where H is the hypothesis to be tested — Output

E is the evidence associated with H — Input attributes

Statistical Model: Naive Bayes (2/10)

Bayes' Theorem

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A,B)}{P(B)}$$

$$\therefore P(A,B) = P(A \mid B) * P(B)$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A) * P(A)}{P(B \mid A)P(A) + P(B \mid \overline{A})P(\overline{A})}$$

$$P(A,B,C) = P(A \mid B,C) * P(B,C) = P(A \mid B,C) * P(B \mid C) * P(C)$$

$$\cdots$$

$$P(a_1, a_2, \cdots, a_m) = \prod_{i=1}^{m} P(a_i \mid a_{i+1}, \cdots, a_m)$$

Statistical Model: Naive Bayes (3/10)

P(play = yes | outlook = sunny, temperature = cool, humidity = high, windy = true)
= P(outlook = sunny, temperature = cool, humidity = high, windy = true | play = yes)
* P(play = yes)
/P(outlook = sunny, temperature = cool, humidity = high, windy = true)

where

P(outlook = sunny, temperature = cool, humidity = high, windy = true | play = yes)
* P(play = yes)
= P(outlook = sunny | play = yes) * P(temperature = cool | play = yes)
* P(humidity = high | play = yes) * P(windy = true | play = yes) * P(play = yes)
= (2/9)(3/9)(3/9)(3/9)(9/14) = 0.0053 (20.5%)

Similarly

P(E | play = no) * P(play = no) = (3/5)(1/5)(4/5)(3/5)(5/14) = 0.0206 (79.5%)

Table 1.2 Weather Data

Outlook	Temperature	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

Table 4.2 Weather Data with Counts and Probabilities

Outlook			Temperature			Humidity			Windy			Play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

P(Outlook=sunny|Play=yes)

P(Play=yes)

Statistical Model: Naive Bayes (5/10)

- Zero-Valued Attribute Counts

  Laplace estimator

  $$\frac{n+1}{d+m}$$ For each attribute's value, adding 1 to its count

  where m is the number of possible values for the attribute

P(E | play=yes) = (2+1)/(9+3) * 4/12 * 4/11 * 4/11

P(E | play=no) = (3+1)/(5+3) * 2/8 * 5/7 * 4/7

Statistical Model: Naive Bayes (6/10)

- Missing data
  - In training set
    - Not included in the frequency counts
  - For unknown example
    - Ignore this attribute

Give a probability value of 1.0 to this attribute

Outlook = unknown
Temperature = cool
Humidity = high
Windy = true
Play = ?

Table 4.2 Weather Data with Counts and Probabilities

Outlook			Temperature			Humidity			Windy			Play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

$P(E \mid play=yes) = (4/12)\ (4/11)\ (4/11)$

$P(E \mid play=no) = (2/8)\ (5/7)\ (4/7)$

Numeric Data

Table 1.3 Weather Data with Some Numeric Attributes

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	false	no
Sunny	80	90	true	no
Overcast	83	86	false	yes
Rainy	70	96	false	yes
Rainy	68	80	false	yes
Rainy	65	70	true	no
Overcast	64	65	true	yes
Sunny	72	95	false	no
Sunny	69	70	false	yes
Rainy	75	80	false	yes
Sunny	75	70	true	yes
Overcast	72	90	true	yes
Overcast	81	75	false	yes
Rainy	71	91	true	no

Table 4.4 Numeric Weather Data with Summary Statistics

Outlook			Temperature			Humidity			Windy			Play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3		83	85		86	85	false	6	2	9	5
overcast	4	0		70	80		96	90	true	3	3		
rainy	3	2		68	65		80	70					
				64	72		65	95					
				69	71		70	91					
				75			80						
				75			70						
				72			90						
				81			75						
sunny	2/9	3/5	*mean*	73	74.6	*mean*	79.1	86.2	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	*std. dev.*	6.2	7.9	*std. dev.*	10.2	9.7	true	3/9	3/5		
rainy	3/9	2/5											

Assuming normal or Gaussian probability distribution

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where

e = the *exponential function*
μ = the class *mean* for the given numerical attribute
σ = the class *standard deviation* for the attribute
x = the attribute value

If a particular attribute follows some other distribution, that distribution can be used instead

Statistical Model: Naive Bayes (8/10)

$$f(temperature = 66 \mid yes) = \frac{1}{\sqrt{2\pi} \times 6.2} e^{-\frac{(66-73)^2}{2\times 6.2^2}} = 0.0340$$

$$f(humidity = 90 \mid yes) = 0.0221$$

```
Naive Bayes Classifier

                    Class
Attribute             yes       no
                    (0.63)    (0.38)
=============================
outlook
  sunny               3.0       4.0
  overcast            5.0       1.0
  rainy               4.0       3.0
  [total]            12.0       8.0

temperature
  hot                 3.0       3.0
  mild                5.0       3.0
  cool                4.0       2.0
  [total]            12.0       8.0

humidity
  high                4.0       5.0
  normal              7.0       2.0
  [total]            11.0       7.0

windy
  TRUE                4.0       4.0
  FALSE               7.0       3.0
  [total]            11.0       7.0

Time taken to build model: 0 seconds

=== Predictions on training set ===

inst#,     actual, predicted, error, probability distribution
     1       2:no       2:no          0.296 *0.704
     2       2:no       2:no          0.153 *0.847
     3      1:yes      1:yes         *0.737  0.263
     4      1:yes      1:yes         *0.554  0.446
     5      1:yes      1:yes         *0.867  0.133
     6       2:no      1:yes      +  *0.737  0.263
     7      1:yes      1:yes         *0.913  0.087
     8       2:no       2:no          0.412 *0.588
     9      1:yes      1:yes         *0.786  0.214
    10      1:yes      1:yes         *0.845  0.155
    11      1:yes      1:yes         *0.568  0.432
    12      1:yes      1:yes         *0.667  0.333
    13      1:yes      1:yes         *0.925  0.075
    14       2:no       2:no          0.348 *0.652

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          13               92.8571 %
Incorrectly Classified Instances         1                7.1429 %
```

10/16

Table 4.2 Weather Data with Counts and Probabilities

Outlook			Temperature			Humidity			Windy			Play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

$$\frac{n+1}{d+m}$$

where m is the number of possible values for the attribute

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

yes:

(3/12) * (3/12) * (4/11) * (7/11) * 0.63 = 0.00911

no:

(4/8) * (3/8) * (5/7) * (3/7) * 0.38 = 0.02181

0.02181/(0.00911+0.02181) = 0.705

```
Naive Bayes Classifier

                   Class
Attribute            yes      no
                   (0.63)  (0.38)
=============================
outlook
  sunny              3.0     4.0
  overcast           5.0     1.0
  rainy              4.0     3.0
  [total]           12.0     8.0

temperature
  mean           72.9697 74.8364
  std. dev.       5.2304   7.384
  weight sum           9       5
  precision       1.9091  1.9091

humidity
  mean           78.8395 86.1111
  std. dev.       9.8023  9.2424
  weight sum           9       5
  precision       3.4444  3.4444

windy
  TRUE               4.0     4.0
  FALSE              7.0     3.0
  [total]           11.0     7.0


Time taken to build model: 0 seconds

=== Predictions on training set ===

inst#,     actual, predicted, error, probability distribution
     1       2:no       2:no          0.157 *0.843
     2       2:no       2:no          0.175 *0.825
     3      1:yes      1:yes         *0.75   0.25
     4      1:yes      1:yes         *0.539  0.461
     5      1:yes      1:yes         *0.796  0.204
     6       2:no      1:yes      +  *0.761  0.239
     7      1:yes      1:yes         *0.944  0.056
     8       2:no       2:no          0.392 *0.608
     9      1:yes      1:yes         *0.848  0.152
    10      1:yes      1:yes         *0.787  0.213
    11      1:yes      1:yes         *0.694  0.306
    12      1:yes      1:yes         *0.744  0.256
    13      1:yes      1:yes         *0.908  0.092
    14       2:no       2:no          0.441 *0.559

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          13               92.8571 %
Incorrectly Classified Instances         1                7.1429 %
```

Divide-and-Conquer: constructing decision trees (1/14)

- Method

```
DT(root);

DT(node) {
    IF having identical class THEN stop
    ELSE {
        select the most appropriate attribute A at the node ;
        FOR each value a of A DO DT(a);
    }
}
```

- How to determine which attribute to split on?
- Top-down approach

Divide-and-Conquer: constructing decision trees (2/14)

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

FIGURE 4.2

Tree stumps for the weather data: (a) outlook, (b) temperature, (c) humidity, and (d) windy.

Gain(outlook) = 0.247

Gain(temperature) = 0.029

Gain(humidity) = 0.152

Gain(windy) = 0.048

Entropy

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

average amount of information needed to identify the class label of a tuple in D

$$where\ p_i = \frac{|C_{i,D}|}{|D|}$$ is the nonzero probability that an arbitrary tupe in D belongs to class C_i

ID3

Information Gain

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

expected information required to classify a tuple from D based on the partitioning by A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute having maximum information gain is selected

Info([2,3]) = -(2/5)log$_2$(2/5)-(3/5)log$_2$(3/5) = 0.971

Info([4,0]) = -(0/4)log$_2$(0/4)-(4/4)log$_2$(4/4) = 0

Info([3,2]) = -(3/5)log$_2$(3/5)-(2/5)log$_2$(2/5) = 0.971

Info([3,3]) = -(3/6)log$_2$(3/6)-(3/6)log$_2$(3/6) = 1

Info$_{outlook}$([2,3],[4,0],[3,2]) = (5/14)*0.971+(4/14)*0+(5/14)*0.971 = 0.693

Info([9,5]) = -(9/14)log$_2$(9/14)-(5/14)log$_2$(5/14) = 0.940

Gain(outlook) = 0.940-0.693 = 0.247

Gain(temperature) = 0.029

Gain(humidity) = 0.152

Gain(windy) = 0.048

Divide-and-Conquer: constructing decision trees (4/14)

Gain(temperature) = 0.571

Gain(humidity) = 0.971

Gain(windy) = 0.020

FIGURE 4.3

Expanded tree stumps for the weather data: (a) temperature, (b) humidity, and (c) windy.

Divide-and-Conquer: constructing decision trees (5/14)

FIGURE 4.4

Decision tree for the weather data.

Divide-and-Conquer: constructing decision trees (6/14)

nominal attributes only

```
Scheme:weka.classifiers.trees.Id3
Relation:     weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:evaluate on training data

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

Id3


outlook = sunny
|   humidity = high: no
|   humidity = normal: yes
outlook = overcast: yes
outlook = rainy
|   windy = TRUE: no
|   windy = FALSE: yes

Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
```

FIGURE 4.4

Decision tree for the weather data.

Divide-and-Conquer: constructing decision trees (7/14)

Information gain measure tends to prefer attributes with large numbers of possible values

FIGURE 4.5

Tree stump for the *ID code* attribute.

Info([0,1]) = $-(0/1)\log_2(0/1)-(1/1)\log_2(1/1) = 0$

Info([0,1],[0,1],[1,0],...) = $(1/14)*0 + \ldots = 0$

Info([9,5]) = 0.940

Gain(ID code) = 0.940- 0 = 0.940

Gain(outlook) = 0.247

Gain(temperature) = 0.029

Gain(humidity) = 0.152

Gain(windy) = 0.048

Divide-and-Conquer: constructing decision trees (8/14)

C4.5

Gain Ratio

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

Attribute having maximum gain ratio is selected

Gain(ID code) = 0.940

SplitInfo([1,1,...] = -(1/14)$\log_2$(1/14)*14 = 3.807

GainRatio(ID code) = 0.940/3.807 = 0.247

filter it out by using ad hoc test

Gain(outlook) = 0.940-0.693 = 0.247

SplitInfo([5,4,5] = -(5/14)$\log_2$(5/14)-(4/14)$\log_2$(4/14)-(5/14)$\log_2$(5/14) = 1.577

GainRatio(outlook) = 0.247/1.577 = 0.156

GainRatio(temperature) = 0.019

GainRatio(humidity) = 0.152

GainRatio(windy) = 0.049

Divide-and-Conquer: constructing decision trees (9/14)

```
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:evaluate on training data

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

J48 pruned tree
------------------

outlook = sunny
|   humidity = high: no (3.0)
|   humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :     5

Size of the tree :      8


Time taken to build model: 0.01 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          14
Incorrectly Classified Instances         0
```

minimum number of instances per leaf

batchSize	100
binarySplits	False
collapseTree	True
confidenceFactor	0.25
debug	False
doNotCheckCapabilities	False
doNotMakeSplitPointActualValue	False
minNumObj	2
numDecimalPlaces	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False
useMDLcorrection	True

```
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:evaluate on training data

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

J48 pruned tree
------------------

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :     5

Size of the tree :      8


Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          14               100      %
Incorrectly Classified Instances         0                 0      %
```

Relation: weather

No.	outlook Nominal	temperature Numeric	humidity Numeric	windy Nominal	play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

Gini Index

- If a data set D contains examples from n classes, gini index, gini(D) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the gini index gini(D) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)

$gini([4,0]) = 1 - (4/4)^2 - (0/4)^2 = 0$
$gini([2,2]) = 1 - (2/4)^2 - (2/4)^2 = 0.5$
$1 - n*(1/n)^2 = 1 - 1/n$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$\begin{aligned} gini_{income \in \{low,medium\}}(D) &= (\frac{10}{14})gini(D_1) + (\frac{4}{14})gini(D_2) \\ &= \frac{10}{14} * \left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14} * \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) \\ &= 0.443 \\ &= Gini_{income \in \{high\}}(D) \end{aligned}$$

but $gini_{\{low\}}$ is 0.30 and $gini_{\{medium\}}$ is 0.4583

So, pick the separation {low} and {high, medium}

Divide-and-Conquer: constructing decision trees (13/14)

- Methods for building Decision Trees
  - *C4.5*
    - Quinlan's most recent noncommercial decision tree building algorithm
    - Weka: J48
  - ID3
    - Quinlan
    - Precursor to C4.5
    - Weka: Id3

Divide-and-Conquer: constructing decision trees (14/14)

- CART (commercial)
  - Breiman et al., 1984
  - Building binary trees usually
  - Weka: SimpleCart (minimum cost-complexity pruning)
- CHAID (commercial)
  - Kass, 1980
  - SPSS and SAS
  - For categorical attributes only

Covering Algorithms: constructing rules (1/8)

if x>1.2 then class=a (6/11)

if x>1.2 and y>2.6 then class=a (5/5)

if x>1.4 and y<2.4 then class=a (1/2)

if x<=1.2 then class=b (9/9)

if x>1.2 and y<=2.6 then class=b (5/6)

FIGURE 4.6

Covering algorithm: (a) covering the instances, and (b) decision tree for the same problem.

Covering Algorithms: constructing rules (2/8)

PRISM

Take each class in turn and seek a way of covering all instances in it

```
Initialize E to the instance set
For each class C
  ~~Initialize E to the instance set~~
  While E contains instances in class C
    Create a rule R with an empty left-hand side that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value v,
        Consider adding the condition A = v to the LHS of R
        Select A and v to maximize the accuracy p/t
          (break ties by choosing the condition with the largest p)
      Add A = v to R
    Remove the instances covered by R from E
```

FIGURE 4.8

Pseudocode for a basic rule learner.

Covering Algorithms: constructing rules (4/8)

if ? then contact-lenses=hard

age = young (2/8)
age = pre-presbyopic (1/8)
age = presbyopic (1/8)
spectacle-prescrip = myope (3/12)
spectacle-prescrip = hypermetrope (1/12)
astigmatism = no (0/12)
astigmatism = yes (4/12)*
tear-prod-rate = reduced (0/12)
tear-prod-rate = normal (4/12)

Viewer

Relation: contact-lenses

No.	age Nominal	spectacle-prescrip Nominal	astigmatism Nominal	tear-prod-rate Nominal	contact-lenses Nominal
4	young	myope	yes	normal	hard
8	young	hypermetrope	yes	normal	hard
12	pre-presbyopic	myope	yes	normal	hard
20	presbyopic	myope	yes	normal	hard
1	young	myope	no	reduced	none
3	young	myope	yes	reduced	none
5	young	hypermetrope	no	reduced	none
7	young	hypermetrope	yes	reduced	none
9	pre-presbyopic	myope	no	reduced	none
11	pre-presbyopic	myope	yes	reduced	none
13	pre-presbyopic	hypermetrope	no	reduced	none
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
21	presbyopic	hypermetrope	no	reduced	none
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none
2	young	myope	no	normal	soft
6	young	hypermetrope	no	normal	soft
10	pre-presbyopic	myope	no	normal	soft
14	pre-presbyopic	hypermetrope	no	normal	soft
22	presbyopic	hypermetrope	no	normal	soft

if astigmatism=yes then contact-lenses=hard (4/12)

Covering Algorithms: constructing rules (5/8)

if astigmatism=yes and
?
then contact-lenses=hard

age = young (2/4)
age = pre-presbyopic (1/4)
age = presbyopic (1/4)
spectacle-prescrip = myope (3/6)
spectacle-prescrip = hypermetrope (1/6)
tear-prod-rate = reduced (0/6)
tear-prod-rate = normal (4/6) *

Viewer

Relation: contact-lenses

No.	age Nominal	spectacle-prescrip Nominal	astigmatism Nominal	tear-prod-rate Nominal	contact-lenses Nominal
4	young	myope	yes	normal	hard
8	young	hypermetrope	yes	normal	hard
12	pre-presbyopic	myope	yes	normal	hard
20	presbyopic	myope	yes	normal	hard
1	young	myope	no	reduced	none
3	young	myope	yes	reduced	none
5	young	hypermetrope	no	reduced	none
7	young	hypermetrope	yes	reduced	none
9	pre-presbyopic	myope	no	reduced	none
11	pre-presbyopic	myope	yes	reduced	none
13	pre-presbyopic	hypermetrope	no	reduced	none
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
21	presbyopic	hypermetrope	no	reduced	none
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none
2	young	myope	no	normal	soft
6	young	hypermetrope	no	normal	soft
10	pre-presbyopic	myope	no	normal	soft
14	pre-presbyopic	hypermetrope	no	normal	soft
22	presbyopic	hypermetrope	no	normal	soft

if astigmatism=yes and
tear-prod-rate=normal
then contact-lenses=hard (4/6)

Covering Algorithms: constructing rules (6/8)

if astigmatism=yes and
tear-prod-rate=normal and
?
then contact-lenses=hard

age = young (2/2)
age = pre-presbyopic (1/2)
age = presbyopic (1/2)
spectacle-prescrip = myope (3/3)*
spectacle-prescrip = hypermetrope (1/3)

if astigmatism=yes and
tear-prod-rate=normal and
spectacle-prescrip=myope
then contact-lenses=hard (3/3)

Viewer

Relation: contact-lenses

No.	age Nominal	spectacle-prescrip Nominal	astigmatism Nominal	tear-prod-rate Nominal	contact-lenses Nominal
4	young	myope	yes	normal	hard
8	young	hypermetrope	yes	normal	hard
12	pre-presbyopic	myope	yes	normal	hard
20	presbyopic	myope	yes	normal	hard
1	young	myope	no	reduced	none
3	young	myope	yes	reduced	none
5	young	hypermetrope	no	reduced	none
7	young	hypermetrope	yes	reduced	none
9	pre-presbyopic	myope	no	reduced	none
11	pre-presbyopic	myope	yes	reduced	none
13	pre-presbyopic	hypermetrope	no	reduced	none
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
21	presbyopic	hypermetrope	no	reduced	none
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none
2	young	myope	no	normal	soft
6	young	hypermetrope	no	normal	soft
10	pre-presbyopic	myope	no	normal	soft
14	pre-presbyopic	hypermetrope	no	normal	soft
22	presbyopic	hypermetrope	no	normal	soft

Covering Algorithms: constructing rules (7/8)

Follow the same process, we find another rule for the class “hard”

if age = young and
astigmatism = yes and
tear-prod-rate = normal
then contact-lenses=hard (1/1)

Covering Algorithms: constructing rules (8/8)

```
Scheme:weka.classifiers.rules.Prism
Relation:     contact-lenses
Instances:    24
Attributes:   5
              age
              spectacle-prescrip
              astigmatism
              tear-prod-rate
              contact-lenses
Test mode:evaluate on training data

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

Prism rules
----------
If astigmatism = no
   and tear-prod-rate = normal
   and spectacle-prescrip = hypermetrope then soft
If astigmatism = no
   and tear-prod-rate = normal
   and age = young then soft
If age = pre-presbyopic
   and astigmatism = no
   and tear-prod-rate = normal then soft
If astigmatism = yes
   and tear-prod-rate = normal
   and spectacle-prescrip = myope then hard
If age = young
   and astigmatism = yes
   and tear-prod-rate = normal then hard
If tear-prod-rate = reduced then none
If age = presbyopic
   and tear-prod-rate = normal
   and spectacle-prescrip = myope
   and astigmatism = no then none
If spectacle-prescrip = hypermetrope
   and astigmatism = yes
   and age = pre-presbyopic then none
If age = presbyopic
   and spectacle-prescrip = hypermetrope
   and astigmatism = yes then none


Time taken to build model: 0 seconds
```

nominal attributes only

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

Correctly Classified Instances          24              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               24

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                 1         0          1          1         1           1        soft
                 1         0          1          1         1           1        hard
                 1         0          1          1         1           1        none
Weighted Avg.    1         0          1          1         1           1

=== Confusion Matrix ===

  a  b  c   <-- classified as
  5  0  0 |  a = soft
  0  4  0 |  b = hard
  0  0 15 |  c = none
```

Mining Association Rules (1/14)

- Affinity analysis（吸引力分析）
  - The general process of determining which things go together
  - One of market basket analysis
    - The output of the market basket analysis is a set of associations about customer-purchase

Mining Association Rules (2/14)

- "Support" for a rule（支持度） coverage
  - The minimum percentage of instances in the database that contain all items listed in a given association rule
- Rule "Confidence"（信賴度） accuracy
  - "If A then B", rule confidence is the conditional probability that B is true when A is known to be true
- Frequent k-itemset $I = \{i_1, \ldots, i_k\}$
  Satisfying the support

$$support(A \Rightarrow B) = P(A \cup B)$$

包含所有A跟B的items

$$confidence(A \Rightarrow B) = P(B \mid A) = \frac{P(A \cup B)}{P(A)} = \frac{support(A \cup B)}{support(A)}$$

Mining Association Rules (3/14)

Transaction id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Let $sup_{min} = 50\%$, $conf_{min} = 80\%$

Frequent itemsets:

{A:3, B:3, D:4, E:3, AD:3}

Association rules:

$A \Rightarrow D$ (3/5=60%, 3/3=100%)

~~$D \Rightarrow A$ (3/5=60%, 3/4=75%)~~

Mining Association Rules (4/14)

- Apriori algorithm
  - Step 1: find all frequent itemsets satisfying minimum support
  - Step 2: create a set of (strong) association rules
    - Strong association rule: satisfy both minimum support and minimum confidence

Mining Association Rules (5/14)

TID	Items
1	Bread, Milk
2	Beer, Bread, Diapers, Eggs
3	Beer, Cola, Diapers, Milk
4	Beer, Bread, Diapers, Milk
5	Bread, Cola, Diapers, Milk

Support: 60%

minimal support count: 5*0.6 = 3

1-itemset	Count
Beer	3
Bread	4
Cola	2
Diapers	4
Milk	4
Eggs	1

2-itemset	count
Beer, Bread	2
Beer, Diapers	3
Beer, Milk	2
Bread, Diapers	3
Bread, Milk	3
Diapers, Milk	3

3-itemset	count
Bread, Diapers, Milk	2

Mining Association Rules (6/14)

Support-based pruning

All nonempty subsets of a frequent itemset must also be frequent.

frequent itemset

Each k-frequent itemset comes from the union of two k-1-frequent itemsets

All supersets of an infrequent itemset must also be infrequent.

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Table 4.10 Item Sets for Weather Data with Coverage 2 or Greater

	One-Item Sets		Two-Item Sets		Three-Item Sets		Four-Item Sets	
1	outlook = sunny	5	outlook = sunny temperature = mild	2	outlook = sunny temperature = hot humidity = high	2	outlook = sunny temperature = hot humidity = high play = no	2
2	outlook = overcast	4	outlook = sunny temperature = hot	2	outlook = sunny temperature = hot play = no	2	outlook = sunny humidity = high windy = false play = no	2
3	outlook = rainy	5	outlook = sunny humidity = normal	2	outlook = sunny humidity = normal play = yes	2	outlook = overcast temperature = hot windy = false play = yes	2
4	temperature = cool	4	outlook = sunny humidity = high	3	outlook = sunny humidity = high windy = false	2	outlook = rainy temperature = mild windy = false play = yes	2
5	temperature = mild	6	outlook = sunny windy = true	2	outlook = sunny humidity = high play = no	3	outlook = rainy humidity = normal windy = false play = yes	2
6	temperature = hot	4	outlook = sunny windy = false	3	outlook = sunny windy = false play = no	2	temperature = cool humidity = normal windy = false play = yes	2
7	humidity = normal	7	outlook = sunny play = yes	2	outlook = overcast temperature = hot windy = false	2		
8	humidity = high	7	outlook = sunny play = no	3	outlook = overcast temperature = hot play = yes	2		
9	windy = true	6	outlook = overcast temperature = hot	2	outlook = overcast humidity = normal play = yes	2		
10	windy = false	8	outlook = overcast humidity = normal	2	outlook = overcast humidity = high play = yes	2		
11	play = yes	9	outlook = overcast humidity = high	2	outlook = overcast windy = true play = yes	2		
12	play = no	5	outlook = overcast windy = true	2	outlook = overcast windy = false play = yes	2		
13			outlook = overcast windy = false	2	outlook = rainy temperature = cool humidity = normal	2		
…			…		…			
38			humidity = normal windy = false	4	humidity = normal windy = false play = yes	4		
39			humidity = normal play = yes	6	humidity = high windy = false play = no	2		
40			humidity = high windy = true	3				
…			…					
47			windy = false play = no	2				

Mining Association Rules (8/14)

- Generating (strong) associate rules

for each frequent itemset I
    for all nonempty subset S of I
        output "s ⇒ (l-s)" if

$$\frac{support_count(l)}{support_count(s)} \geq \min_conf$$

Mining Association Rules (9/14)

1-itemset	Count
Beer	3
Bread	4
Cola	2
Diapers	4
Milk	4
Eggs	1

2-itemset	count
Beer, Bread	2
Beer, Diapers	3
Beer, Milk	2
Bread, Diapers	3
Bread, Milk	3
Diapers, Milk	3

1-item consequent of the rule	confidence
Diapers -> Beer	75%
Beer -> Diapers	100%
Diapers -> Bread	75%
Bread -> Diapers	75%
Milk -> Bread	75%
Bread -> Milk	75%
Diapers -> Milk	75%
Milk -> Diapers	75%

Mining Association Rules (10/14)

Confidence-based pruning

hold

supp(bcd) <= supp(cd)
So, supp(abcd)/supp(bcd) >= supp(abcd)/supp(cd)

Starting from a single item for the left side of a rule

Mining Association Rules (11/14)

```
humidity = normal, windy = false, play = yes
```

This set leads to seven potential rules:

```
If humidity = normal and windy = false then play = yes     4/4
If humidity = normal and play = yes then windy = false     4/6
If windy = false and play = yes then humidity = normal     4/6
If humidity = normal then windy = false and play = yes     4/7
If windy = false then humidity = normal and play = yes     4/8
If play = yes then humidity = normal and windy = false     4/9
If - then humidity = normal and windy = false and play = yes    4/14
```

Table 4.11 Association Rules for Weather Data

	Association Rule	Coverage	Accuracy
1	humidity = normal windy = false ⇒ play = yes	4	100%
2	temperature = cool ⇒ humidity = normal	4	100%
3	outlook = overcast ⇒ play = yes	4	100%
4	temperature = cool play = yes ⇒ humidity = normal	3	100%
5	outlook = rainy windy = false ⇒ play = yes	3	100%
6	outlook = rainy play = yes ⇒ windy = false	3	100%
7	outlook = sunny humidity = high ⇒ play = no	3	100%
8	outlook = sunny play = no ⇒ humidity = high	3	100%
9	temperature = cool windy = false ⇒ humidity = normal play = yes	2	100%
10	temperature = cool humidity = normal windy = false ⇒ play = yes	2	100%
11	temperature = cool windy = false play = yes ⇒ humidity = normal	2	100%
12	outlook = rainy humidity = normal windy = false ⇒ play = yes	2	100%
13	outlook = rainy humidity = normal play = yes ⇒ windy = false	2	100%
14	outlook = rainy temperature = mild windy = false ⇒ play = yes	2	100%
15	outlook = rainy temperature = mild play = yes ⇒ windy = false	2	100%
16	temperature = mild windy = false play = yes ⇒ outlook = rainy	2	100%
17	outlook = overcast temperature = hot ⇒ windy = false play = yes	2	100%
18	outlook = overcast windy = false ⇒ temperature = hot play = yes	2	100%

Table 4.11 *Continued*

	Association Rule	Coverage	Accuracy
19	temperature = hot play = yes ⇒ outlook = overcast windy = false	2	100%
20	outlook = overcast temperature = hot windy = false ⇒ play = yes	2	100%
21	outlook = overcast temperature = hot play = yes ⇒ windy = false	2	100%
22	outlook = overcast windy = false play = yes ⇒ temperature = hot	2	100%
23	temperature = hot windy = false play = yes ⇒ outlook = overcast	2	100%
24	windy = false play = no ⇒ outlook = sunny humidity = high	2	100%
25	outlook = sunny humidity = high windy = false ⇒ play = no	2	100%
26	outlook = sunny windy = false play = no ⇒ humidity = high	2	100%
27	humidity = high windy = false play = no ⇒ outlook = sunny	2	100%
28	outlook = sunny temperature = hot ⇒ humidity = high play = no	2	100%
29	temperature = hot play = no ⇒ outlook = sunny humidity = high	2	100%
30	outlook = sunny temperature = hot humidity = high ⇒ play = no	2	100%
31	outlook = sunny temperature = hot play = no ⇒ humidity = high	2	100%
…	…	…	…
58	outlook = sunny temperature = hot ⇒ humidity = high	2	100%

nominal attributes only

outputItemSets=true

```
Apriori
=======

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

Large Itemsets L(1):
outlook=sunny 5
outlook=overcast 4
outlook=rainy 5
temperature=hot 4
temperature=mild 6
temperature=cool 4
humidity=high 7
humidity=normal 7
windy=TRUE 6
windy=FALSE 8
play=yes 9
play=no 5

Size of set of large itemsets L(2):

Large Itemsets L(2):
outlook=sunny temperature=hot 2
outlook=sunny temperature=mild 2
outlook=sunny humidity=high 3
outlook=sunny humidity=normal 2
```

supp(outlook=overcast)

supp(outlook=overcast & play=yes)

car	False	
classIndex	-1	
delta	0.05	support
doNotCheckCapabilities	False	
lowerBoundMinSupport	0.1	support
metricType	Confidence	
minMetric	0.9	confidence
numRules	10	
outputItemSets	True	
removeAllMissingCols	False	
significanceLevel	-1.0	
treatZeroAsMissing	False	
upperBoundMinSupport	1.0	support
verbose	False	

```
Large Itemsets L(4):
outlook=sunny temperature=hot humidi
outlook=sunny humidity=high windy=FA
outlook=overcast temperature=hot win
outlook=rainy temperature=mild windy=FALSE play=yes 2
outlook=rainy humidity=normal windy=FALSE play=yes 2
temperature=cool humidity=normal windy=FALSE play=yes 2

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)
```

outputItemSets=false

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
=== Associator model (full training set) ===


Apriori
=======

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)
 2. temperature=cool 4 ==> humidity=normal 4    conf:(1)
 3. humidity=normal windy=FALSE 4 ==> play=yes 4    conf:(1)
 4. outlook=sunny play=no 3 ==> humidity=high 3    conf:(1)
 5. outlook=sunny humidity=high 3 ==> play=no 3    conf:(1)
 6. outlook=rainy play=yes 3 ==> windy=FALSE 3    conf:(1)
 7. outlook=rainy windy=FALSE 3 ==> play=yes 3    conf:(1)
 8. temperature=cool play=yes 3 ==> humidity=normal 3    conf:(1)
 9. outlook=sunny temperature=hot 2 ==> humidity=high 2    conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2    conf:(1)
```

Linear Model: linear regression (1/12)

numeric prediction

Supervised technique for a set of numeric input attributes and a single numeric output attribute

For training instance $x^{(1)}=(a_1^{(1)}, a_2^{(1)} \ldots a_k^{(1)})$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

$$\min \sum_{i=1}^{n} (y^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)})^2$$

Usually assume $a_0=1$

Linear Model: linear regression (2/12)

- LinearRegression
- SimpleLinearRegression
  - Learn a linear regression model based on a <u>single</u> attribute
  - <u>Automatically</u> pick the attribute that results in the lowest squared error

Relation: District-Office-Building

No.	space Numeric	offices Numeric	entrances Numeric	age Numeric	value Numeric
1	2310.0	2.0	2.0	20.0	142000.0
10	2517.0	4.0	4.0	55.0	169000.0
11	2540.0	2.0	3.0	22.0	149000.0
2	2333.0	2.0	2.0	12.0	144000.0
3	2356.0	3.0	1.5	33.0	151000.0
4	2379.0	3.0	2.0	43.0	150000.0
5	2402.0	2.0	3.0	53.0	139000.0
6	2425.0	4.0	2.0	23.0	169000.0
7	2448.0	2.0	1.5	99.0	126000.0
8	2471.0	2.0	2.0	34.0	142900.0
9	2494.0	3.0	3.0	23.0	163000.0

output predictions

```
Linear Regression Model

value =

     27.6414 * space +
  12529.7682 * offices +
   2553.2107 * entrances +
   -234.2372 * age +
  52317.8306

Time taken to build model: 0.04 seconds

=== Predictions ontraining set===

inst#,     actual, predicted, error
     1 142000      141650.65     -349.35
     2 144000      144160.299     160.299
     3 151000      151130.233     130.233
     4 150000      150700.219     700.219
     5 139000      139017.042      17.042
     6 169000      169186.234     186.234
     7 126000      125683.82     -316.18
     8 142900      142821.593     -78.407
     9 163000      161116.932   -1883.068
    10 169000      169340.074     340.074
    11 149000      150092.905    1092.905

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient                  0.9984
Mean absolute error                    477.6373
Root mean squared error                716.8197
Relative absolute error                  4.8363 %
Root relative squared error              5.7026 %
Total Number of Instances               11
```

predicted - actual

correlation coefficient 相關係數

-1	—	0	—	1
negatively perfect		no correlation		positively perfect

Table 5.8 Performance Measures for Numeric Prediction

Mean-squared error	$\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}$
Root mean-squared error	$\sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$
Mean-absolute error	$\frac{\lvert p_1 - a_1\rvert + \ldots + \lvert p_n - a_n\rvert}{n}$
Relative-squared error*	$\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \ldots + (a_n - \bar{a})^2}$
Root relative-squared error*	$\sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \ldots + (a_n - \bar{a})^2}}$
Relative-absolute error*	$\frac{\lvert p_1 - a_1\rvert + \ldots + \lvert p_n - a_n\rvert}{\lvert a_1 - \bar{a}\rvert + \ldots + \lvert a_n - \bar{a}\rvert}$
Correlation coefficient** Pearson's correlation coefficient	$\frac{S_{PA}}{\sqrt{S_P S_A}}$, where $S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n-1}$, covariance（共變異數） variance（變異數） $S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1}$, $S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1}$

*Here, $\bar{a}$ is the mean value over the training data.
**Here, $\bar{a}$ is the mean value over the test data.

```
Scheme:weka.classifiers.functions.SimpleLinearRegression
Relation:     District-Office-Building
Instances:    11
Attributes:   5
              space
              offices
              entrances
              age
              value
Test mode:evaluate on training data

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

Linear regression on offices

14245.83 * offices + 111979.17


Time taken to build model: 0 seconds

=== Predictions on training set ===

inst#,     actual, predicted, error
     1 142000     140470.833  -1529.167
     2 144000     140470.833  -3529.167
     3 151000     154716.667   3716.667
     4 150000     154716.667   4716.667
     5 139000     140470.833   1470.833
     6 169000     168962.5       -37.5
     7 126000     140470.833  14470.833
     8 142900     140470.833  -2429.167
     9 163000     154716.667  -8283.333
    10 169000     168962.5       -37.5
    11 149000     140470.833  -8529.167

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient                  0.8742
Mean absolute error                   4431.8182
Root mean squared error               6102.803
Relative absolute error                 44.8745 %
Root relative squared error             48.5507 %
Total Number of Instances               11
```

Picks the attribute that results in the lowest squared error

```
Linear Regression Model

value =

     27.6414 * space +
  12529.7682 * offices +
   2553.2107 * entrances +
   -234.2372 * age +
  52317.8306

Time taken to build model: 0.04 seconds

=== Predictions ontraining set===

inst#,     actual, predicted, error
     1 142000     141650.65    -349.35
     2 144000     144160.299    160.299
     3 151000     151130.233    130.233
     4 150000     150700.219    700.219
     5 139000     139017.042     17.042
     6 169000     169186.234    186.234
     7 126000     125683.82    -316.18
     8 142900     142821.593    -78.407
     9 163000     161116.932  -1883.068
    10 169000     169340.074    340.074
    11 149000     150092.905   1092.905

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient                  0.9984
Mean absolute error                    477.6373
Root mean squared error                716.8197
Relative absolute error                  4.8363 %
Root relative squared error              5.7026 %
Total Number of Instances               11
```

```
Scheme:weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8
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: 0.01 seconds

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient                  0.93
Mean absolute error                     37.9748
Root mean squared error                 58.9899
Relative absolute error                 39.592  %
Root relative squared error             36.7663 %
Total Number of Instances              209
```

Linear Model: linear regression (7/12)

- M5P
  - M5 model/regression tree learner
  - Nonlinear model

Linear Model: linear regression (8/12)

Model Tree

- Regression tree:
  average of the output attribute
  for all instances in that branch
- Model trees: fewer tree levels
  linear regression equation

Scheme:weka.classifiers.trees.M5P -M 4.0

The minimum number of instances to allow at a leaf node.

Relation: cpu

No.	MYCT Numeric	MMIN Numeric	MMAX Numeric	CACH Numeric	CHMIN Numeric	CHMAX Numeric	class Numeric
1	125.0	256.0	6000.0	256.0	16.0	128.0	198.0
10	23.0	32000.0	64000.0	128.0	32.0	64.0	1144.0
100	480.0	96.0	512.0	0.0	1.0	1.0	6.0
101	203.0	1000.0	2000.0	0.0	1.0	5.0	24.0
102	115.0	512.0	6000.0	16.0	1.0	6.0	45.0
103	1100.0	512.0	1500.0	0.0	1.0	1.0	7.0
104	1100.0	768.0	2000.0	0.0	1.0	1.0	13.0
105	600.0	768.0	2000.0	0.0	1.0	1.0	16.0
106	400.0	2000.0	4000.0	0.0	1.0	1.0	32.0
107	400.0	4000.0	8000.0	0.0	1.0	1.0	32.0
108	900.0	1000.0	1000.0	0.0	1.0	2.0	11.0
109	900.0	512.0	1000.0	0.0	1.0	2.0	11.0
11	400.0	1000.0	3000.0	0.0	1.0	2.0	38.0
110	900.0	1000.0	4000.0	4.0	1.0	2.0	18.0
111	900.0	1000.0	4000.0	8.0	1.0	2.0	22.0
112	900.0	2000.0	4000.0	0.0	3.0	6.0	37.0
113	225.0	2000.0	4000.0	8.0	3.0	6.0	40.0
114	225.0	2000.0	4000.0	8.0	3.0	6.0	34.0
115	180.0	2000.0	8000.0	8.0	1.0	6.0	50.0
116	185.0	2000.0	16000.0	16.0	1.0	6.0	76.0
117	180.0	2000.0	16000.0	16.0	1.0	6.0	66.0
118	225.0	1000.0	4000.0	2.0	3.0	6.0	24.0

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

M5 pruned model tree:
(using smoothed linear models)

CHMIN <= 7.5 : LM1 (165/12.903%)
CHMIN >  7.5 :
|   MMAX <= 28000 :
|   |   MMAX <= 13240 :
|   |   |   CACH <= 81.5 : LM2 (6/18.551%)
|   |   |   CACH >  81.5 : LM3 (4/30.824%)
|   |   MMAX >  13240 : LM4 (11/24.185%)
|   MMAX >  28000 : LM5 (23/48.302%)

LM num: 1
class =
        -0.0055 * MYCT
        + 0.0013 * MMIN
        + 0.0029 * MMAX
        + 0.8007 * CACH
        + 0.4015 * CHMAX
        + 11.0971

LM num: 2
class =
        -1.0307 * MYCT
        + 0.0086 * MMIN
        + 0.0031 * MMAX
        + 0.7866 * CACH
        - 2.4503 * CHMIN
        + 1.1597 * CHMAX
        + 70.8672

LM num: 3
class =
        -1.1057 * MYCT
        + 0.0086 * MMIN
        + 0.0031 * MMAX
        + 0.7995 * CACH
        - 2.4503 * CHMIN
        + 1.1597 * CHMAX
        + 83.0016

LM num: 4
class =
        -0.8813 * MYCT
        + 0.0086 * MMIN
        + 0.0031 * MMAX
        + 0.6547 * CACH
        - 2.3561 * CHMIN
        + 1.1597 * CHMAX
        + 82.5725

LM num: 5
class =
        -0.4882 * MYCT
        + 0.0218 * MMIN
        + 0.003 * MMAX
        + 0.3865 * CACH
        - 1.3252 * CHMIN
        + 3.3671 * CHMAX
        - 51.8474

Number of Rules : 5
```

instances

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

Correlation coefficient                  0.9728
Mean absolute error                     21.9246
Root mean squared error                 37.5095
Relative absolute error                 22.8582 %
Root relative squared error             23.3784 %
Total Number of Instances              209
```

Scheme:weka.classifiers.trees.M5P -M 5.0

-M 5.0

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

Correlation coefficient                  0.971
Mean absolute error                     22.4639
Root mean squared error                 38.6636
Relative absolute error                 23.4205 %
Root relative squared error             24.0977 %
Total Number of Instances              209
```

-M 4.0

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

Correlation coefficient                  0.9728
Mean absolute error                     21.9246
Root mean squared error                 37.5095
Relative absolute error                 22.8582 %
Root relative squared error             23.3784 %
Total Number of Instances              209
```

Regression tree

buildRegressionTree = True

```
M5 pruned regression tree:
(using smoothed linear models)

CHMIN <= 7.5 :
|   MMAX <= 6100 :
|   |   MMAX <= 2500 : LM1 (28/7.705%)
|   |   MMAX >  2500 : LM2 (48/7.038%)
|   MMAX >  6100 :
|   |   CACH <= 28 :
|   |   |   CACH <= 0.5 : LM3 (20/10.029%)
|   |   |   CACH >  0.5 : LM4 (43/14.815%)
|   |   CACH >  28 : LM5 (26/29.236%)
CHMIN >  7.5 :
|   MMAX <= 28000 : LM6 (21/53.261%)
|   MMAX >  28000 :
|   |   CHMAX <= 58 :
|   |   |   MMIN <= 12000 : LM7 (11/41.55%)
|   |   |   MMIN >  12000 : LM8 (7/46.208%)
|   |   CHMAX >  58 :
|   |   |   MYCT <= 40 : LM9 (3/68.181%)
|   |   |   MYCT >  40 : LM10 (2/28.047%)

LM num: 1
class =
        + 33.1423

LM num: 2
class =
        + 39.2596

LM num: 3
class =
        + 53.0392

LM num: 4
class =
        + 65.3038

LM num: 5
class =
        + 90.8891

LM num: 6
class =
        + 192.2838
```

model tree

```
LM num: 7
class =
        + 292.7961

LM num: 8
class =
        + 311.4276

LM num: 9
class =
        + 366.2654

LM num: 10
class =
        + 355.1497

Number of Rules : 10

Time taken to build model: 0.09 seconds

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient                  0.8232
Mean absolute error                     42.1911
Root mean squared error                101.2808
Relative absolute error                 43.9878 %
Root relative squared error             63.1247 %
Total Number of Instances              209
```

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

Correlation coefficient                  0.9728
Mean absolute error                     21.9246
Root mean squared error                 37.5095
Relative absolute error                 22.8582 %
Root relative squared error             23.3784 %
Total Number of Instances              209
```

Linear Model: linear regression (12/12)

Logistic Regression

Relation: CreditCardPromotionLogistic

No.	Income Range Numeric	Magazine Promotion Numeric	Watch Promotion Numeric	Life-In-Promo Nominal	Credit Card Insurance Numeric	Sex Numeric	Age Numeric
1	40000.0	1.0	0.0	No	0.0	1.0	45.0
2	30000.0	1.0	1.0	Yes	0.0	0.0	40.0
3	40000.0	0.0	0.0	No	0.0	1.0	42.0
4	30000.0	1.0	1.0	Yes	1.0	1.0	43.0
5	50000.0	1.0	0.0	Yes	0.0	0.0	38.0
6	20000.0	0.0	0.0	No	0.0	0.0	55.0
7	30000.0	1.0	0.0	Yes	1.0	1.0	35.0
8	20000.0	0.0	1.0	No	0.0	1.0	27.0
9	30000.0	1.0	0.0	No	0.0	1.0	43.0
10	30000.0	1.0	1.0	Yes	0.0	0.0	41.0
11	40000.0	0.0	1.0	Yes	0.0	0.0	43.0
12	20000.0	0.0	1.0	Yes	0.0	1.0	29.0
13	50000.0	1.0	1.0	Yes	0.0	0.0	39.0
14	40000.0	0.0	1.0	No	0.0	1.0	55.0
15	20000.0	0.0	0.0	Yes	1.0	0.0	19.0

Relation: CreditCardPromotionNet

No.	Income Range Numeric	Magazine Promotion Numeric	Watch Promotion Numeric	Life Ins Promotion Numeric	Credit Card Insurance Numeric	Sex Numeric	Age Numeric
1	40000.0	1.0	0.0	0.0	0.0	1.0	45.0
2	30000.0	1.0	1.0	1.0	0.0	0.0	40.0
3	40000.0	0.0	0.0	0.0	0.0	1.0	42.0
4	30000.0	1.0	1.0	1.0	1.0	1.0	43.0
5	50000.0	1.0	0.0	1.0	0.0	0.0	38.0
6	20000.0	0.0	0.0	0.0	0.0	0.0	55.0
7	30000.0	1.0	0.0	1.0	1.0	1.0	35.0
8	20000.0	0.0	1.0	0.0	0.0	1.0	27.0
9	30000.0	1.0	0.0	0.0	0.0	1.0	43.0
10	30000.0	1.0	1.0	1.0	0.0	0.0	41.0
11	40000.0	0.0	1.0	1.0	0.0	0.0	43.0
12	20000.0	0.0	1.0	1.0	0.0	1.0	29.0
13	50000.0	1.0	1.0	1.0	0.0	0.0	39.0
14	40000.0	0.0	1.0	0.0	0.0	1.0	55.0
15	20000.0	0.0	0.0	1.0	1.0	0.0	19.0

```
Scheme:weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8
Relation:     CreditCardPromotionNet-weka.filters.unsupervised.attr
Instances:    15
Attributes:   5
              Income Range
                   Life Ins Promotion
                    Credit Card Insurance
              Sex
              Age
Test mode:evaluate on training data

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


Linear Regression Model

     Life Ins Promotion =

      0.5909 *       Credit Card Insurance +
     -0.5455 * Sex +
      0.7727

Time taken to build model: 0 seconds

=== Predictions on training set ===

inst#,    actual, predicted, error
     1      0          0.227      0.227
     2      1          0.773     -0.227
     3      0          0.227      0.227
     4      1          0.818     -0.182
     5      1          0.773     -0.227
     6      0          0.773      0.773
     7      1          0.818     -0.182
     8      0          0.227      0.227
     9      0          0.227      0.227
    10      1          0.773     -0.227
    11      1          0.773     -0.227
    12      1          0.227     -0.773
    13      1          0.773     -0.227
    14      0          0.227      0.227
    15      1          1.364      0.364
```

Linear Model: logistic regression (2/6)

classification

non-linear

$$p(y=1\,|\,x) = \frac{1}{1+e^{-(ax+c)}} = \frac{e^{ax+c}}{1+e^{ax+c}}$$

$$x = (x_1, x_2, x_3 \ldots x_n)$$

$$ax + c = a_1x_1 + a_2x_2 + a_3x_3 + \ldots\ldots a_nx_n + c$$

odd ratio

logit transformation

$$\log_e\left(\frac{p(y=1\,|\,x)}{1-p(y=1\,|\,x)}\right) = ax + c$$

two classes

$$\text{or} \quad \frac{p(y=1\,|\,x)}{1-p(y=1\,|\,x)} = e^{ax+c}$$

```
Scheme:weka.classifiers.functions.Logistic -R 1.0E-8
Relation:     CreditCardPromotionLogistic-weka.filte
Instances:    15
Attributes:   5
              Income Range
              Life-In-Promo
                    Credit Card Insurance
              Sex
              Age
Test mode:evaluate on training data

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

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
                                          Class
Variable                                    Yes
===========================================
Income Range                             0.0001
      Credit Card Insurance             21.2336
Sex                                     -8.3143
Age                                     -0.4151
Intercept                               17.6907


Odds Ratios...
                                          Class
Variable                                    Yes
===========================================
Income Range                             1.0001
      Credit Card Insurance    1665837985.6492
Sex                                      0.0002
Age                                      0.6602


Time taken to build model: 0.01 seconds
```

Relation: CreditCardPromotionLogistic

No.	Income Range Numeric	Magazine Promotion Numeric	Watch Promotion Numeric	Life-In-Promo Nominal	Credit Card Insurance Numeric	Sex Numeric	Age Numeric
1	40000.0	1.0	0.0	No	0.0	1.0	45.0
2	30000.0	1.0	1.0	Yes	0.0	0.0	40.0
3	40000.0	0.0	0.0	No	0.0	1.0	42.0
4	30000.0	1.0	1.0	Yes	1.0	1.0	43.0
5	50000.0	1.0	0.0	Yes	0.0	0.0	38.0
6	20000.0	0.0	0.0	No	0.0	0.0	55.0
7	30000.0	1.0	0.0	Yes	1.0	1.0	35.0
8	20000.0	0.0	1.0	No	0.0	1.0	27.0
9	30000.0	1.0	0.0	No	0.0	1.0	43.0
10	30000.0	1.0	1.0	Yes	0.0	0.0	41.0
11	40000.0	0.0	1.0	Yes	0.0	0.0	43.0
12	20000.0	0.0	1.0	Yes	0.0	1.0	29.0
13	50000.0	1.0	1.0	Yes	0.0	0.0	39.0
14	40000.0	0.0	1.0	No	0.0	1.0	55.0
15	20000.0	0.0	0.0	Yes	1.0	0.0	19.0

$$p(y=1\mid x)=\frac{1}{1+e^{-(0.0001*40000+21.2336*0-8.3143*1-0.4151*45+17.6907)}}$$

$$=\frac{1}{1+e^{-(-5.3031)}}=0.00495$$

```
=== Predictions on training set ===

inst#,     actual, predicted, error, probability distribution
     1       2:No       2:No          0.007 *0.993
     2      1:Yes      1:Yes         *0.987  0.013
     3       2:No       2:No          0.024 *0.976
     4      1:Yes      1:Yes         *1      0
     5      1:Yes      1:Yes         *0.999  0.001
     6       2:No       2:No          0.049 *0.951
     7      1:Yes      1:Yes         *1      0
     8       2:No      1:Yes      +  *0.584  0.416
     9       2:No       2:No          0.005 *0.995
    10      1:Yes      1:Yes         *0.981  0.019
    11      1:Yes      1:Yes         *0.985  0.015
    12      1:Yes       2:No      +   0.38  *0.62
    13      1:Yes      1:Yes         *0.999  0.001
    14       2:No       2:No          0     *1
    15      1:Yes      1:Yes         *1      0
```

```
Scheme:weka.classifiers.functions.Logistic -R 1.0E-8 -M -1
Relation:     CreditCardPromotion-weka.filters.unsupervised.attribu
Instances:    15
Attributes:   5
              Income-Range
              Life-In-Promo
              Credit-Card-Ins
              Sex
              Age
Test mode:evaluate on training data

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

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
                           Class
Variable                     Yes
===================================
Income-Range=20-30000      -2.9466
Income-Range=30-40000      -0.8905
Income-Range=40-50000      -1.0868
Income-Range=50-60000       8.5381
Credit-Card-Ins           -38.0683
Sex                         7.8789
Age                        -0.4032
Intercept                  52.2156


Odds Ratios...
                           Class
Variable                     Yes
===================================
Income-Range=20-30000       0.0525
Income-Range=30-40000       0.4104
Income-Range=40-50000       0.3373
Income-Range=50-60000    5105.7916
Credit-Card-Ins                 0
Sex                      2640.8747
Age                         0.6682


Time taken to build model: 0.05 seconds

=== Predictions ontraining set===

inst#,     actual, predicted, error, probability distribution
     1       2:No       2:No          0.006 *0.994
     2      1:Yes      1:Yes         *0.993  0.007
     3       2:No       2:No          0.02  *0.98
     4      1:Yes      1:Yes         *1      0
     5      1:Yes      1:Yes         *1      0
     6       2:No       2:No          0.043 *0.957
     7      1:Yes      1:Yes         *1      0
     8       2:No      1:Yes      +  *0.578  0.422
     9       2:No       2:No          0.017 *0.983
    10      1:Yes      1:Yes         *0.99   0.01
    11      1:Yes      1:Yes         *0.973  0.027
    12      1:Yes       2:No      +   0.379 *0.621
    13      1:Yes      1:Yes         *1      0
    14       2:No       2:No          0     *1
    15      1:Yes      1:Yes         *1      0

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          13               86.6667 %
Incorrectly Classified Instances         2               13.3333 %
```

Relation: CreditCardPromotion-weka.filters.unsupervised.attribute.R

No.	Income-Range Nominal	Life-In-Promo Nominal	Credit-Card-Ins Nominal	Sex Nominal	**Age** Numeric
1	40-50000	No	No	Male	45.0
10	30-40000	Yes	No	Female	41.0
11	40-50000	Yes	No	Female	43.0
12	20-30000	Yes	No	Male	29.0
13	50-60000	Yes	No	Female	39.0
14	40-50000	No	No	Male	55.0
15	20-30000	Yes	Yes	Female	19.0
2	30-40000	Yes	No	Female	40.0
3	40-50000	No	No	Male	42.0
4	30-40000	Yes	Yes	Male	43.0
5	50-60000	Yes	No	Female	38.0
6	20-30000	No	No	Female	55.0
7	30-40000	Yes	Yes	Male	35.0
8	20-30000	No	No	Male	27.0
9	30-40000	No	No	Male	43.0

$$\frac{p(y=1|x)}{p(y=3|x)} = e^{a^{(1)}x+c^{(1)}} \quad \Rightarrow p(y=1|x) = (e^{a^{(1)}x+c^{(1)}}) * p(y=3|x)$$

$$\frac{p(y=2|x)}{p(y=3|x)} = e^{a^{(2)}x+c^{(2)}} \quad \Rightarrow p(y=2|x) = (e^{a^{(2)}x+c^{(2)}}) * p(y=3|x)$$

$$\because \; p(y=1|x) + p(y=2|x) + p(y=3|x) = 1$$

$$\therefore \; (e^{a^{(1)}x+c^{(1)}}) * p(y=3|x) + (e^{a^{(2)}x+c^{(2)}}) * p(y=3|x) + p(y=3|x) = 1$$

$$p(y=3|x) = \frac{1}{1+e^{a^{(1)}x+c^{(1)}} + e^{a^{(2)}x+c^{(2)}}}$$

$$p(y=1|x) = \frac{e^{a^{(1)}x+c^{(1)}}}{1+e^{a^{(1)}x+c^{(1)}} + e^{a^{(2)}x+c^{(2)}}}$$

$$p(y=2|x) = \frac{e^{a^{(2)}x+c^{(2)}}}{1+e^{a^{(1)}x+c^{(1)}} + e^{a^{(2)}x+c^{(2)}}}$$

$$p_j(x_i) = \frac{e^{x_i B_j}}{1+\sum_{j=1}^{k-1} e^{x_i B_j}} \quad \textit{for} \text{ the first k-1 classes}$$

$$p_k(x_i) = 1 - \sum_{j=1}^{k-1} p_j(x_i)$$

$$= 1 - \frac{\sum_{j=1}^{k-1} e^{x_i B_j}}{1+\sum_{j=1}^{k-1} e^{x_i B_j}}$$

$$= \frac{1}{1+\sum_{j=1}^{k-1} e^{x_i B_j}}$$

```
Scheme:weka.classifiers.functions.Logistic -R 1.0E-8 -M -1
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:evaluate on training data

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

Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
                         Class
Variable           Iris-setosa  Iris-versicolor
===================================================
sepallength            21.8065           2.4652
sepalwidth              4.5648           6.6809
petallength           -26.3083          -9.4293
petalwidth             -43.887         -18.2859
Intercept               8.1743           42.637


Odds Ratios...
                         Class
Variable           Iris-setosa  Iris-versicolor
===================================================
sepallength    2954196659.8892          11.7653
sepalwidth             96.0426         797.0304
petallength                  0           0.0001
petalwidth                   0                0


Time taken to build model: 0.05 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         148               98.6667 %
Incorrectly Classified Instances         2                1.3333 %
```

Linear Model: perceptron (1/3)

classification

>0	output 1 (C_1)
<=0	output 0 (C_2)

```
Set all weights to zero
UNTIL all instances in the training data are classified correctly
   FOR each instance I in the training data
      IF I is classified incorrectly by the perceptron
         IF I belongs to the first class add it to the weight vector
         ELSE subtract it from the weight vector
```

$(w_0+a_0)a_0+ (w_1+a_1)a_1+ (w_2+a_2)a_2+...+ (w_k+a_k)\ a_k$

$(w_0-a_0)a_0+ (w_1-a_1)a_1+ (w_2-a_2)a_2+...+ (w_k-a_k)\ a_k$

Linear Model: perceptron (2/3)

instance #	a_1	a_2	OR	
1	0	0	0	C1
2	0	1	1	C2
3	1	0	1	C2
4	1	1	1	C2

OR

w	0	0	0		
	1	0	0	0	X
	1	0	0		
	1	0	1	1	X
	0	0	-1		
	1	1	0	0	V
	1	1	1	-1	V
	1	0	0	0	X
	1	0	-1		
	1	0	1	0	V
	1	1	0	1	X
	0	-1	-1		
	1	1	1	-2	V
	1	0	0	0	X
	1	-1	-1		
	1	0	1	0	V
	1	1	0	0	V
	1	1	1	-1	V
	1	0	0	1	V

round 1

round 2

round 3

round 4

Linear Model: perceptron (3/3)

Linearly separable:

If the data can be separated perfectly into two groups using a hyperplane.

$$w_0a_0 + w_1a_1 + w_2a_2 + \ldots + w_ka_k = 0$$

Linear Model: Winnow (1/5)

For binary attribute value only

Set every w_i to any positive constant

```
While some instances are misclassified
  for every instance a
    classify a using the current weights
    if the predicted class is incorrect
      if a belongs to the first class
        for each a_i that is 1, multiply w_i by α
        (if a_i is 0, leave w_i unchanged)
      otherwise
        for each a_i that is 1, divide w_i by α
        (if a_i is 0, leave w_i unchanged)
```

$\alpha > 1$

$$class1: \quad w_0 a_0 + w_1 a_1 + \cdots + w_k a_k > \theta$$

usually set threshold to # of attributes

(a)

```
While some instances are misclassified
  for every instance a
    classify a using the current weights
    if the predicted class is incorrect
      if a belongs to the first class
        for each a_i that is 1,
          multiply w_i+ by α
          divide w_i- by α
        (if a_i is 0, leave w_i+ and w_i- unchanged)
      otherwise
          multiply w_i- by α
          divide w_i+ by α
        (if a_i is 0, leave w_i+ and w_i- unchanged)
```

multiply w_i^- by β

$$class1: \quad \left(w_0^+ - w_0^-\right) a_0 + \left(w_1^+ - w_1^-\right) a_1 + \cdots + \left(w_k^+ - w_k^-\right) a_k > \theta$$

(b)

FIGURE 4.11

The Winnow algorithm: (a) unbalanced version and (b) balanced version.

Linear Model: Winnow (2/5)

unbalanced version

```
=== Run information ===

Scheme:weka.classifiers.functions.Winnow -I 10 -A 2.0 -B 0.5 -H -1.
Relation:     OR
Instances:    4
Attributes:   3
              X
              Y
              OR
Test mode:evaluate on training data

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

Winnow

Attribute weights

w0 4.0
w1 4.0

Cumulated mistake count: 2



Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          4               100      %
Incorrectly Classified Instances        0                 0      %
```

alpha	2.0
balanced	False
beta	0.5
debug	False
defaultWeight	2.0
numIterations	1
seed	1
threshold	-1.0

# of attributes

$$? \ (4X+4Y>2)$$

$$4X+4Y-2>0$$

```
=== Run information ===

Scheme:weka.classifiers.functions.Winnow -I 10 -A 2.0 -B 0.5 -H -1.
Relation:     XOR
Instances:    4
Attributes:   3
              X
              Y
              XOR
Test mode:evaluate on training data

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

Winnow

Attribute weights

w0 2.0
w1 2.0

Cumulated mistake count: 30



Time taken to build model: 0 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          1                25      %
Incorrectly Classified Instances        3                75      %
```

Linear Model: Winnow (3/5)

instance #	a_0	a_1	OR	
1	0	0	0	C2
2	0	1	1	C1
3	1	0	1	C1
4	1	1	1	C1

$$\alpha = 2$$

$$\varpi_0 = \varpi_1 = 2$$

$$\theta = 2$$

2	2		
0	0	0	V
0	1	2	X
2	4		
1	0	2	X
4	4		
1	1	8	V
0	0	0	V
0	1	4	V
1	0	4	V

Linear Model: Winnow (4/5)

balanced version

```
Scheme:weka.classifiers.functions.Winnow -L -I 1 -A 2.0 -B 0.5 -H -1.0 -W 2.0 -S 1
Relation:     OR
Instances:    4
Attributes:   3
              X
              Y
              OR
Test mode:evaluate on training data

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

Winnow

Attribute weights

w0 p 4.0 n 1.0 d 3.0
w1 p 4.0 n 1.0 d 3.0

Cumulated mistake count: 1



Time taken to build model: 0 seconds

=== Predictions on training set ===

inst#,     actual, predicted, error, probability distribution
     1        1:F        1:F        *1       0
     2        2:T        2:T         0      *1
     3        2:T        2:T         0      *1
     4        2:T        2:T         0      *1

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances           4               100      %
Incorrectly Classified Instances         0                 0      %
```

Linear Model: Winnow (5/5)

instance #	a_0	a_1	OR	
1	0	0	0	C2
2	0	1	1	C1
3	1	0	1	C1
4	1	1	1	C1

$$\alpha = 2$$

$$\varpi_0^+ = \varpi_1^+ = 2$$

$$\varpi_0^- = \varpi_1^- = 2$$

$$\theta = 2$$

2 2	2 2		
0	0	0	V
0	1	0	X
2 2	4 1		
1	0	0	X
4 1	4 1		
1	1	6	V
0	0	0	V
0	1	3	V
1	0	3	V

Support Vector Machines (1/13)

$$(x_i, y_i),\ \ i = 1,\ldots,l,\ \ x_i \in R^n$$

$$y_i = \begin{cases} +1,\ x_i \in \text{ class1} \\ -1,\ x_i \in \text{ class2} \end{cases}$$

Support Vector Machines (2/13)

margin = $d_+ + d_-$

goal: Max{margin}

Support Vector Machines (3/13)

$$\begin{cases} w^T x + b \geq 1 & \text{if } y_i = +1 \\ w^T x + b \leq -1 & \text{if } y_i = -1 \end{cases}$$

$$\therefore \ \mathrm{y_i}(w^T x + b) \geq 1, \quad \text{for all i}$$

$$\mathrm{d_+} = \mathrm{d_-} = \frac{1}{\|w\|} = \frac{1}{\sqrt{w^T w}}$$

$$w_1 x_1 + w_2 x_2 + w_3 x_3 = b_1$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$ 法向量

$$w_1 x_1 + w_2 x_2 + w_3 x_3 = b_2$$ 平行線

$$\frac{|b_1 - b_2|}{\sqrt{w_1^2 + w_2^2 + w_3^2}} = \frac{|b_1 - b_2|}{\|w\|} = \frac{|b_1 - b_2|}{\sqrt{w^T w}}$$ 兩平行線距離

$$\boxed{\begin{aligned} &\min \ \Phi(w) = \frac{1}{2} w^T w \\ &\text{subject to } \ \mathrm{y_i}(w^T x + b) \geq 1, \quad \text{for all i} \end{aligned}}$$

Relation: AND

No.	X Nominal	Y Nominal	AND Nominal
1	F	F	F
2	F	T	F
3	T	F	F
4	T	T	T

```
Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N
Relation:     AND
Instances:    4
Attributes:   3
              X
              Y
              AND
Test mode:evaluate on training data

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

SMO

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

Classifier for classes: F, T

BinarySMO

Machine linear: showing attribute weights, not support vectors.

         0.5    * (normalized) X
 +       0.5    * (normalized) Y
 -       1.5

Number of kernel evaluations: 5 (16.667% cached)



Time taken to build model: 0.01 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances          3               75      %
Incorrectly Classified Instances        1               25      %
```

buildLogisticModels	False
c	1.0
checksTurnedOff	False
debug	False
epsilon	1.0E-12
filterType	Normalize training data
kernel	Choose **PolyKernel** -C 250007 -E 1.0
numFolds	-1
randomSeed	1
toleranceParameter	0.001

weka.classifiers.functions.supportVector.PolyKernel

About

The polynomial kernel : K(x, y) = <x, y>^p or K(x, y) = (<x, y>+1)^p

More

Capabilities

cacheSize	250007
checksTurnedOff	False
debug	False
exponent	1.0
useLowerOrder	False

Open... Save... OK Cancel

```
Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N
Relation:     AND
Instances:    4
Attributes:   3
              X
              Y
              AND
Test mode:evaluate on training data

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

SMO

Kernel used:
  Poly Kernel: K(x,y) = <x,y>^2.0

Classifier for classes: F, T

BinarySMO

      0.7997 * <1 1 > * X]
 -       0.3992 * <0 1 > * X]
 -       0.4005 * <1 0 > * X]
 -       1.3999

Number of support vectors: 3

Number of kernel evaluations: 9 (82% cached)



Time taken to build model: 0.01 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances           4               100      %
Incorrectly Classified Instances         0                 0      %
```

Support Vector Machines (6/13)

feature mapping $\phi : R^2 \rightarrow R^3$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1$$

$$\frac{1}{a^2} z_1 + 0 * z_2 + \frac{1}{b^2} z_3 = 1$$

Support Vector Machines (7/13)

Define a kernel function $k : R^2 * R^2 \rightarrow R$ such that

$$k(x, x') = < \phi(x), \phi(x') >$$

$$k(x, x') = < \phi(x), \phi(x') >$$
$$= < \phi(x_1, x_2), \phi(x'_1, x'_2) > = < (z_1, z_2, z_3), (z'_1, z'_2, z'_3) > = < (x_1^2, \sqrt{2}x_1x_2, x_2^2), (x'^2_1, \sqrt{2}x'_1 x'_2, x'^2_2) >$$
$$= x_1^2 x'^2_1 + 2x_1x_2x'_1 x'_2 + x_2^2 x'^2_2 = (x_1x'_1 + x_2x'_2)^2 = (< x, x' >)^2$$

Support Vector Machines (8/13)

Distance in the feature space

$$\|\phi(x)-\phi(x')\|^2 = (\phi(x)-\phi(x'))^T(\phi(x)-\phi(x'))$$

$$= (\phi(x)^T - \phi(x')^T)(\phi(x)-\phi(x'))$$

$$= \phi(x)^T\phi(x) - \phi(x)^T\phi(x') - \phi(x')^T\phi(x) + \phi(x')^T\phi(x')$$

$$= <\phi(x),\phi(x)> - <\phi(x),\phi(x')> - <\phi(x'),\phi(x)> + <\phi(x'),\phi(x')>$$

$$= k(x,x) - 2k(x,x') + k(x',x')$$

Angle in the feature space

$$\cos\theta = \frac{<\phi(x),\phi(x')>}{\|\phi(x)\|\bullet\|\phi(x')\|} = \frac{<\phi(x),\phi(x')>}{\sqrt{<\phi(x),\phi(x)>}\bullet\sqrt{<\phi(x'),\phi(x')>}}$$

$$= \frac{k(x,x')}{\sqrt{k(x,x)}\bullet\sqrt{k(x',x')}}$$

Training Samples:

feature mapping

$$\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset R^d \times \{1,-1\} \xrightarrow{\phi} \{(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n)\} \subset H \times \{1,-1\}$$

$$\operatorname{sgn} x = \begin{cases} -1 & : \quad x < 0 \\ 0 & : \quad x = 0 \\ 1 & : \quad x > 0 \end{cases}$$

$$c_+ = \frac{1}{m_+} \sum_{\{i|y_i=+1\}} \phi(x_i)$$

$$y = 1 \Leftrightarrow 0 \le \theta < \frac{\pi}{2} \Leftrightarrow 0 < \cos\theta \le 1 \Leftrightarrow 0 < \frac{<\phi(x)-c, w>}{\sqrt{|\phi(x)-c|^2}\sqrt{|w|^2}} \le 1 \Leftrightarrow <\phi(x)-c, w> \ \ge 0$$

$$y = -1 \Leftrightarrow \frac{\pi}{2} < \theta \le \pi \Leftrightarrow -1 \le \cos\theta < 0 \Leftrightarrow -1 \le \frac{<\phi(x)-c, w>}{\sqrt{|\phi(x)-c|^2}\sqrt{|w|^2}} < 0 \Leftrightarrow <\phi(x)-c, w> \le 0$$

$$\boxed{y = \operatorname{sgn}\left(<\phi(x)-c, w> \right)}$$

$\phi(x)$ can be ignored.

$$<\phi(x)-c, w>$$

$$= \frac{2m_+ - 1}{2m_+^2} \sum_{\{i|y_i=1\}} k(x_i, x_i) + \frac{1}{m_+ m_-} \sum_{\{i|y_i=1\}} \sum_{\{j|y_j=-1\}} k(x_i, x_j) + \frac{2m_- - 1}{2m_-^2} \sum_{\{i|y_i=-1\}} k(x_i, x_i)$$

Support Vector Machines (10/13)

- Given a feature mapping ϕ , we can find a corresponding kernel function k

$$k(x, x') = < \phi(x), \phi(x') >$$

- Given a kernel function k, we can construct a feature space H (i.e. a feature mapping ϕ) such that k computes the dot product in H

Support Vector Machines (11/13)

Some widely used kernel functions

- Linear Kernel

$$\kappa(x,z) = < x, z >$$

- Polynomial Kernel

$$\kappa(x,z) = (< x, z > +1)^r, \quad r \in Z^+$$

- RBF (Gaussian) Kernel

$$\kappa(x,z) = \exp\left(\frac{-\|x-z\|^2}{2\sigma^2}\right), \quad \sigma \in R - \{0\}$$

Support Vector Machines (12/13)

Given the samples $x_1,\ldots,x_m \in R^d$ and a kernel function $k : R^d * R^d \rightarrow R$ then

$$K = \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_m) \\ \vdots & \ddots & \vdots \\ k(x_m,x_1) & \cdots & k(x_m,x_m) \end{bmatrix}$$

is called the kernel matrix (Gram matrix) of k with respect to $x_1,\ldots,x_m$

Support Vector Machines (13/13)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{R}^2 \qquad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \in \mathcal{R}^2$$

$$\kappa(\mathbf{x}, \mathbf{z}) = (x_1 z_1 + x_2 z_2)^2$$

$k(x, z) = \langle x, z \rangle^2$

$$\mathbf{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$K = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) \end{bmatrix}$$

$$= \begin{bmatrix} (0 \times 0 + 0 \times 0)^2 & (0 \times 1 + 0 \times 1)^2 \\ (1 \times 0 + 1 \times 0)^2 & (1 \times 1 + 1 \times 1)^2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$$

Instance-based learning (1/10)

- Eager learners
  - Use models learned from a set of training data to classify a new instance
  - e.g. decision tree, rule-based, Bayesian classifier ...

Instance-based learning (2/10)

- Lazy learners
  - Store training data set and wait until it is given a test instance
  - Learning from your neighbors
  - Require efficient storage techniques

Instance-based learning (3/10)

Classification or Numeric prediction

Randomly pick one or select the closest one

Instance-based learning (4/10)

k^{th} Nearest Neighbor algorithm

Let k be the number of nearest neighbors and D be the set of training data;
FOR (each test instance z=(x', y')) {
  Compute d(x', x), the distance between z and every instance (x, y) in D;
  Select D', the subset of D, containing all k closest training data to z;

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D'} I(y_i = v)$$

}

$$I(x) = \begin{cases} 1 & \text{if x is true} \\ 0 & \text{otherwise} \end{cases}$$

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D'} w_i \times I(y_i = v)$$

where $w_i = \dfrac{1}{d(x', x_i)}$ or $w_i = 1 - d(x', x_i)$

Instance-based learning (5/10)

- Usually normalize all attribute values to lie between 0 and 1

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

- For nominal attributes

$$d(v_i, v_j) = \begin{cases} 0 & \text{if } v_i \text{ and } v_j \text{ are identical} \\ 1 & \text{otherwise} \end{cases}$$

- For missing values
  - nominal
    Difference is 0 only if they are not missing and both are the same
  - numeric

$$d(-,-) = 1$$

$$d(v_i,-) = \max\{|v_i|, 1 - |v_i|\}$$

Instance-based learning (6/10)

IB1 (basic instance-based learner)

Find the closest training instance and predict the same class as this training instance.
If several instances qualify as the closest, the first one found is used

Instance-based learning (7/10)

Relation: weather.symbolic

No.	outlook Nominal	temperature Nominal	humidity Nominal	windy Nominal	play Nominal	
1	sunny	hot	high	FALSE	no	ts4
2	sunny	hot	high	TRUE	no	
3	overcast	hot	high	FALSE	yes	
4	rainy	mild	high	FALSE	yes	
5	rainy	cool	normal	FALSE	yes	ts1
6	rainy	cool	normal	TRUE	no	
7	overcast	cool	normal	TRUE	yes	
8	sunny	mild	high	FALSE	no	ts3
9	sunny	cool	normal	FALSE	yes	ts2
10	rainy	mild	normal	FALSE	yes	
11	sunny	mild	normal	TRUE	yes	
12	overcast	mild	high	TRUE	yes	ts5
13	overcast	hot	normal	FALSE	yes	
14	rainy	mild	high	TRUE	no	

	ts1	ts2	ts3	ts4	ts5
tn2	4	3	2	1	2
tn3	3	3	2	1	2
tn4	2	3	1	2	2
tn6	1	2	4	4	3
tn7	2	2	4	4	2
tn10	1	2	2	3	3
tn11	3	2	2	3	2
tn13	2	2	3	2	3
tn14	3	4	2	3	1

```
Scheme:weka.classifiers.lazy.IB1
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R3-4
Instances:    150
Attributes:   3
              sepallength
              sepalwidth
              class
Test mode:split 90.0% train, remainder test

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

IB1 classifier

Time taken to build model: 0 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          10               66.6667 %
Incorrectly Classified Instances         5               33.3333 %
```

Instance-based learning (9/10)

IBk

k-nearest-neighbor

Generally k gets decided based on the square root of data points

```
Scheme:weka.classifiers.lazy.IBk -K 3 -W 0 -I -A "weka.core.neighboursearch.LinearNI
Relation:     CreditCardPromotion-weka.filters.unsupervised.attribute.Remove-R2-3
Instances:    15
Attributes:   5
              Income-Range
              Life-In-Promo
              Credit-Card-Ins
              Sex
              Age
Test mode:split 66.0% train, remainder test

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

IB1 instance-based classifier
using 3 inverse-distance-weighted nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Predictions on test split ===

inst#,     actual, predicted, error, probability distribution
     1       1:Yes       2:No      +   0.014 *0.986
     2       1:Yes      1:Yes         *0.971  0.029
     3        2:No       2:No          0.014 *0.986
     4       1:Yes      1:Yes         *0.999  0.001
     5        2:No       2:No          0.06  *0.94

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances           4               80      %
Incorrectly Classified Instances         1               20      %
```

Instance-based learning (10/10)

```
Scheme:weka.classifiers.lazy.IBk -K 5 -W 0 -I -A "weka.core.neighboursearch.LinearNN
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R3-4
Instances:    150
Attributes:   3
              sepallength
              sepalwidth
              class
Test mode:split 90.0% train, remainder test

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

IB1 instance-based classifier
using 5 inverse-distance-weighted nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          10               66.6667 %
Incorrectly Classified Instances         5               33.3333 %
```

Clustering:K-Means (1/10)

Unsupervised clustering

1. Choose a value for K, the total number of clusters
2. Randomly choose K points as cluster centers
3. Assign the remaining instances to their closest cluster center by using Euclidean distance
4. Use the instances in each cluster to calculate a new mean for each cluster
5. Repeat steps 3-5 until the cluster centers do not change

Clustering:K-Means (2/10)

Table 3.6 • **K-Means Input Values**

Instance	X	Y
1	1.0	1.5
2	1.0	4.5
3	2.0	1.5
4	2.0	3.5
5	3.0	2.5
6	5.0	6.0

Clustering:K-Means (3/10)

Table 3.6 • **K-Means Input Values**

Instance	X	Y
1	1.0	1.5
2	1.0	4.5
3	2.0	1.5
4	2.0	3.5
5	3.0	2.5
6	5.0	6.0

Iteration-1

K = 2;
Select (1.0, 1.5) as the center of C_1
(2.0, 1.5) as the center of C_2

Instance #	1	2	3	4	5	6
Distance from center of C_1	0	3	1	2.24	2.24	6.02
Distance from center of C_2	1	3.16	0	2	1.41	5.41

Class 1: 1, 2 mean (1.0, 3.0) = C_1
Class 2: 3, 4, 5, 6 mean (3.0, 3.375) = C_2

Iteration-2

Instance #	1	2	3	4	5	6
Distance from center of C_1	1.5	1.5	1.8	1.12	2.06	5
Distance from center of C_2	2.74	2.29	2.125	1.01	0.875	3.3

Class 1: 1, 2, 3 mean (1.33, 2.50) = C_1
Class 2: 4, 5, 6 mean (3.33, 4.00) = C_2

Clustering:K-Means (4/10)

$C_1 = (1.33, 2.50)$
$C_2 = (3.33, 4.00)$

Iteration-3

Instance #	1	2	3	4	5	6
Distance from center of C_1	1.05	2.03	1.2	1.2	1.67	5.07
Distance from center of C_2	3.42	2.38	2.83	1.42	1.54	2.61

Class 1: 1, 2, 3, 4 mean (1.50, 2.75)
Class 2: 5, 6 mean (4.00, 4.25)

Clustering:K-Means (5/10)

```
Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-c
Relation:     XYpoints
Instances:    6
Attributes:   2
              X
              Y
Test mode:    evaluate on training data


=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 0.908371913580247

Initial starting points (random):

Cluster 0: 2,3.5
Cluster 1: 1,1.5

Missing values globally replaced with mean/mode

Final cluster centroids:
                          Cluster#
Attribute    Full Data          0          1
                 (6.0)      (4.0)      (2.0)
==============================================
X                2.3333       2.75        1.5
Y                  3.25      4.125        1.5
```

$$SS = \sum_{k} \sum_{x_i \epsilon k} (x_i - \mu_k)^2$$

(4.5+3.5+2.5+6.0)/4

No.	Instance_number Numeric	X Numeric	Y Numeric	Cluster Nominal
1	0.0	1.0	1.5	cluster1
2	1.0	1.0	4.5	cluster0
3	2.0	2.0	1.5	cluster1
4	3.0	2.0	3.5	cluster0
5	4.0	3.0	2.5	cluster0
6	5.0	5.0	6.0	cluster0

Clustering:K-Means (6/10)

Run the algorithm several times with different initial choices and choose the best final result because the final clusters are quite sensitive to the initial cluster centers.

```
Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 100
Relation:     XYpoints
Instances:    6
Attributes:   2
              X
              Y
Test mode:evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 0.8559670781893004
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                   (6)        (3)        (3)
==========================================
X               2.3333     2.6667          2
Y                 3.25     4.6667     1.8333
```

```
Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 20
Relation:     XYpoints
Instances:    6
Attributes:   2
              X
              Y
Test mode:evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 0.5108024691358024
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                   (6)        (1)        (5)
==========================================
X               2.3333          5        1.8
Y                 3.25          6        2.7
```

better

Clustering:K-Means (7/10)

```
Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 16.237456311387238
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute       Full Data        0          1
                     (14)      (9)        (5)
==========================================
outlook             sunny    sunny   overcast
temperature       73.5714  75.8889       69.4
humidity          81.6429  84.1111       77.2
windy               FALSE    FALSE        TRUE
play                  yes      yes         yes


Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       9 ( 64%)
1       5 ( 36%)
```

Cluster mode

- ○ Use training set
- ○ Supplied test set — Set...
- ○ Percentage split — % 66
- ◉ Classes to clusters evaluation
  - (Nom) play
- ☑ Store clusters for visualization

Ignore attributes

Start — Stop

Result list (right-click for options)

- 10:54:59 - SimpleKMeans
- 11:09:00 - SimpleKMeans
- 11:11:25 - SimpleKMeans

Clusterer output

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
Ignored:
              play
Test mode:classes to clusters evaluation on training data
=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 11.237456311387234
Missing values globally replaced with mean/mode

Cluster centroids:
                           Cluster#
Attribute       Full Data          0          1
                     (14)        (9)        (5)
==========================================
outlook             sunny      sunny   overcast
temperature       73.5714    75.8889       69.4
humidity          81.6429    84.1111       77.2
windy               FALSE      FALSE       TRUE


Time taken to build model (full training data) : 0 s

=== Model and evaluation on training set ===

Clustered Instances

0       9 ( 64%)
1       5 ( 36%)


Class attribute: play
Classes to Clusters:

 0 1  <-- assigned to cluster
 6 3 | yes
 3 2 | no

Cluster 0 <-- yes
Cluster 1 <-- no

Incorrectly clustered instances :       6.0       42.8571 %
```

Clustering:K-Means (9/10)

Elbow method with sum of squared errors

Clustering:K-Means (10/10)

Silhouette

For data point $i \in C_i$ (data point i in the cluster C_i), let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

The mean s(i) over all data of the entire dataset is a measure of how appropriately the data have been clustered

Agglomerative Clustering (1/10)

bottom-up hierarchical clustering method

- Procedure
  1. Place each instance into a separate partition
  2. Until all instances are part of a single cluster
     - 2.1. Determine the two most similar clusters
     - 2.2. Merge the clusters chosen into a single cluster
  3. Choose a clustering formed by one of the step 2 iterations as a final result

Agglomerative Clustering (2/10)

Table 10.7 • **Five Instances from the Credit Card Promotion Database**

Instance	Income Range	Magazine Promotion	Watch Promotion	Life Insurance Promotion	Sex
I_1	40–50K	Yes	No	No	Male
I_2	25–35K	Yes	Yes	Yes	Female
I_3	40–50K	No	No	No	Male
I_4	25–35K	Yes	Yes	Yes	Male
I_5	50–60K	Yes	No	Yes	Female

Similarity score

total number of attribute-value matches is divided by total number of attributes

Table 10.8 • **Agglomerative Clustering: First Iteration**

	I_1	I_2	I_3	I_4	I_5
I_1	1.00				
I_2	0.20	1.00			
I_3	0.80	0.00	1.00		
I_4	0.40	0.80	0.20	1.00	
I_5	0.40	0.60	0.20	0.40	1.00

Table 10.8 • **Agglomerative Clustering: First Iteration**

	I_1	I_2	I_3	I_4	I_5
I_1	1.00				
I_2	0.20	1.00			
I_3	0.80	0.00	1.00		
I_4	0.40	0.80	0.20	1.00	
I_5	0.40	0.60	0.20	0.40	1.00

Table 10.9 • **Agglomerative Clustering: Second Iteration**

	I_1 I_3	I_2	I_4	I_5
I_1 I_3	0.80			
I_2	0.33	1.00		
I_4	0.47	0.80	1.00	
I_5	0.47	0.60	0.40	1.00

group average

{0.2 (I1&I2) + 0.8 (I1&I3) + 0.0 (I2&I3)} / 3

Third Iteration

	I1 I3	I2 I4	I5
I1 I3	0.8		
I2 I4	0.4	0.8	
I5	0.47	0.6	1

Fourth Iteration

	I1 I3	I2 I4 I5
I1 I3	0.8	
I2 I4 I5	0.4	0.6

Final Iteration

	I1 I2 I3 I4 I5
I1 I2 I3 I4 I5	0.4

Agglomerative Clustering (4/10)

- "cluster proximity" measures
  - MIN (single-linkage) 樂觀但計算容易

$$dist_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|p - p'|\}$$

  - MAX (complete-linkage) 保守

$$dist_{\max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{|p - p'|\}$$

  - Average-linkage

$$dist_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, p' \in C_j} |p - p'|$$

  - Centroid-linkage

$$dist_{mean}(C_i, C_j) = |m_i - m_j|$$

  - Group average

$$dist_{group-avg}(C_i, C_j) = \frac{1}{c(|C_i \cup C_j|, 2)} \sum_{p, p' \in C_i \cup C_j} |p - p'|$$

Agglomerative Clustering (5/10)

	X	Y
p1	0.4	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.3

	p1	p2	p3	p4	p5	p6
p1	0					
p2	0.24	0				
p3	0.22	0.15	0			
p4	0.37	0.2	0.15	0		
p5	0.34	0.14	0.28	0.29	0	
p6	0.23	0.25	$\boxed{0.11}$	0.22	0.39	0

P3, P6	0.11
P2, P5	0.14
P2, P3	0.15
P3, P4	0.15
P2, P4	0.20
P1, P3	0.22
P4, P6	0.22
P1, P6	0.23
P1, P2	0.24
P2, P6	0.25
P3, P5	0.28
P4, P5	0.29
P1, P5	0.34
P1, P4	0.37
P5, P6	0.39

Single-linkage

	p1	p2	p3 p6	p4	p5
p1	0				
p2	0.24	0			
p3 p6	0.22	0.15	0		
p4	0.37	0.2	0.15	0	
p5	0.34	0.14	0.28	0.29	0

	p1	p2 p3 p5 p6	p4
p1	0		
p2 p3 p5 p6	0.22	0	
p4	0.37	0.15	0

	p1	p2 p5	p3 p6	p4
p1	0			
p2 p5	0.24	0		
p3 p6	0.22	0.15	0	
p4	0.37	0.2	0.15	0

	p1	p2 p3 p4 p5 p6
p1	0	
p2 p3 p4 p5 p6	0.22	0

P3, P6	0.11
P2, P5	0.14
P2, P3	0.15
P3, P4	0.15
P2, P4	0.20
P1, P3	0.22
P4, P6	0.22
P1, P6	0.23
P1, P2	0.24
P2, P6	0.25
P3, P5	0.28
P4, P5	0.29
P1, P5	0.34
P1, P4	0.37
P5, P6	0.39

Complete-linkage

	p1	p2	p3 p6	p4	p5
p1	0				
p2	0.24	0			
p3 p6	0.23	0.25	0.11		
p4	0.37	0.2	0.22	0	
p5	0.34	0.14	0.39	0.29	0

	p1	p2 p5	p3 p6 p4
p1	0		
p2 p5	0.34	0.14	
p3 p6 p4	0.37	0.39	0.22

	p1	p2 p5	p3 p6	p4
p1	0			
p2 p5	0.34	0.14		
p3 p6	0.23	0.39	0.11	
p4	0.37	0.29	0.22	0

	p1 p2 p5	p3 p6 p4
p1 p2 p5	0.34	
p3 p6 p4	0.39	0.22

```
Scheme:weka.clusterers.HierarchicalClusterer -N 3 -L SINGLE -P -A "weka.core.EuclideanDistan
Relation:     Points
Instances:    6
Attributes:   2
              X
              Y
Test mode:evaluate on training data

=== Model and evaluation on training set ===

Cluster 1
((0.38:0.38853,0.41:0.38853):0.00465,(0.32:0.2766,0.3:0.2766):0.11658)


Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       1 ( 17%)
1       4 ( 67%)
2       1 ( 17%)
```

{P1} {P2, P3, P5, P6} {P4}

Single-linkage
SINGLE

6	{P1} {P2} {P3} {P4} {P5} {P6}
5	{P1} {P2} {P3, P6} {P4} {P5}
4	{P1} {P2, P5} {P3, P6} {P4}
3	{P1} {P2, P3, P5, P6} {P4}
2	{P1} {P2, P3, P4, P5, P6}

Complete-linkage
COMPLETE

6	{P1} {P2} {P3} {P4} {P5} {P6}
5	{P1} {P2} {P3, P6} {P4} {P5}
4	{P1} {P2, P5} {P3, P6} {P4}
3	{P1} {P2, P5} {P3, P4, P6}
2	{P1, P2, P5} {P3, P4, P6}

Centroid-linkage
CENTROID

6	{P1} {P2} {P3} {P4} {P5} {P6}
5	{P1} {P2} {P3, P6} {P4} {P5}
4	{P1} {P2, P5} {P3, P6} {P4}
3	{P1} {P2, P5} {P3, P4, P6}
2	{P1} {P2, P3, P4, P5, P6}

Average-linkage
AVERAGE

6	{P1} {P2} {P3} {P4} {P5} {P6}
5	{P1} {P2} {P3, P6} {P4} {P5}
4	{P1} {P2, P5} {P3, P6} {P4}
3	{P1} {P2, P5} {P3, P4, P6}
2	{P1} {P2, P3, P4, P5, P6}

Group-average
MEAN

6	{P1} {P2} {P3} {P4} {P5} {P6}
5	{P1} {P2} {P3, P6} {P4} {P5}
4	{P1} {P2, P5} {P3, P6} {P4}
3	{P1} {P2, P5} {P3, P4, P6}
2	{P1} {P2, P3, P4, P5, P6}

```
Scheme:weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     CreditCardPromotion-weka.filters.unsupervised.attribute.Remove-R5,7
Instances:    15
Attributes:   5
              Income-Range
              Magazine-Prom
              Watch-Promo
              Life-In-Promo
              Sex
Test mode:evaluate on training data

=== Model and evaluation on training set ===

Cluster 0
(((((((0.0:1,0.0:1):0,(0.0:1,0.0:1):0):0,0.0:1):0,(1.0:0,1.0:0):1):0,1.0:1):0,1.0:1):0,((0.0:1,0.0:1):0,0.0:1):0)

Cluster 1
((1.0:1,1.0:1):0.41421,1.0:1.41421)



Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       12 ( 80%)
1        3 ( 20%)
```

14	{1} {2, 10} {3} {4} {5} {6} {7} {8} {9} {11} {12} {13} {14} {15}
13	{1, 3} {2, 10} {4} {5} {6} {7} {8} {9} {11} {12} {13} {14} {15}
12	{1, 3} {2, 10} {4} {5} {6, 15} {7} {8} {9} {11} {12} {13} {14}
11	{1, 3} {2, 10} {4} {5} {6, 15} {7, 9} {8} {11} {12} {13} {14}
10	{1, 3, 7, 9} {2, 10} {4} {5} {6, 15} {8} {11} {12} {13} {14}
9	{1, 3, 4, 7, 9} {2, 10} {5} {6, 15} {8} {11} {12} {13} {14}
8	{1, 3, 4, 7, 9} {2, 10} {6, 15} {8, 12} {5} {11} {13} {14}
7	{1, 2, 3, 4, 7, 9, 10} {6, 15} {8, 12, 14} {5} {11} {13}
6	{1, 2, 3, 4, 7, 9, 10} {6, 15} {8, 12, 14} {5} {11} {13}
5	{1, 2, 3, 4, 7, 9, 10, 13} {6, 15} {8, 12, 14} {5} {11}
4	{1, 2, 3, 4, 5, 7, 9, 10, 13} {6, 15} {8, 12, 14} {11}
3	{1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14} {6, 15} {11}
2	{1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14} {6, 11, 15}