Machine Learning

Russell Chap. 18, 19, 20

Luger Chap. 10

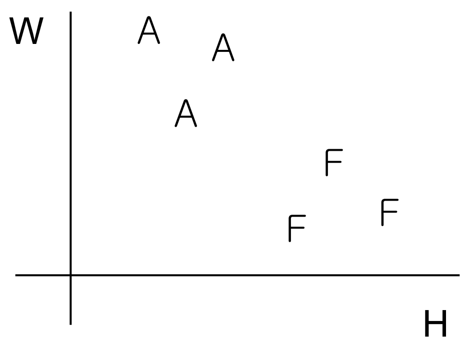
Machine Learning

- Definition
  - Changes in a system that enable it to perform better on repetition of same task
- Types of learning
  - Supervised learning
    - Given: **<problem instance, answer>** examples (labeled)
    - Learning: model - rules, trees, neural nets to give the right answer
  - Unsupervised learning
    - Given: **<problem instance>** examples (unlabeled)
    - Learning: clusters, probability distributions
  - Reinforcement learning
    - Given: **<action, reward>** experiences
    - Learning: rules for right action

Machine Learning

- Supervised learning

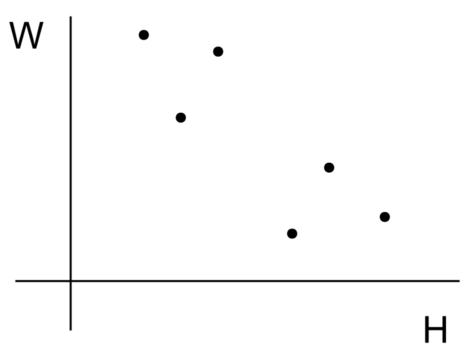
H	W	Grade
185	65	F
162	80	A
175	70	F
165	92	A
...	...	...



<187, 68> → ?

- Unsupervised learning

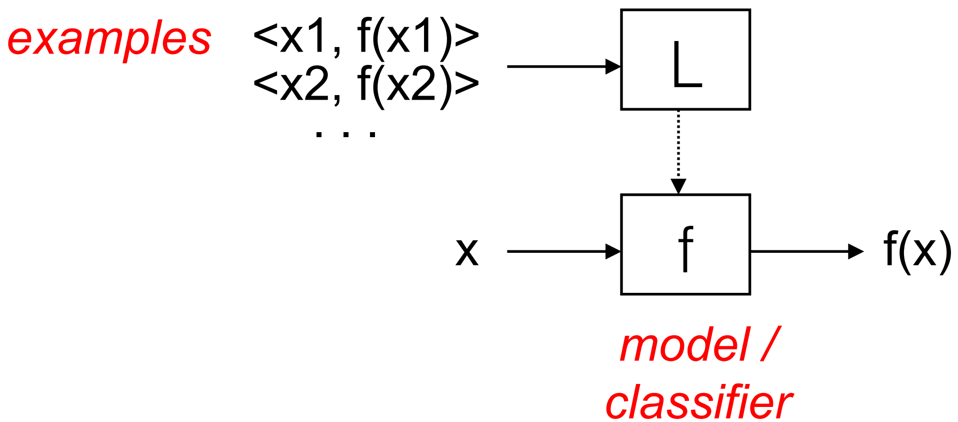
H	W
185	65
160	90
180	70
165	95
...	...



Group ?

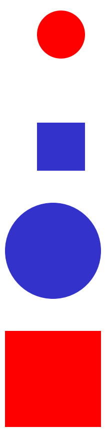
Supervised Learning

- Example: <x, f(x)>
  - X: input, f(X): output (f: target function)
- Learning
  - Given a set of examples
  - Find f (model or classifier)



Supervised Learning

- x = <size, color, shape>
- f(x) = Y/N (good obj?)



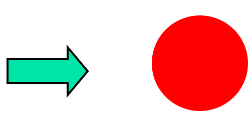
<small, red, ball> → Y

<small, blue, cube> → N

<large, blue, ball> → Y

<large, red, cube> → N

learning



<large, red, ball> →

If (shape = ball)
then Y
If (shape = cube)
then N

→ Y

Learning Concepts

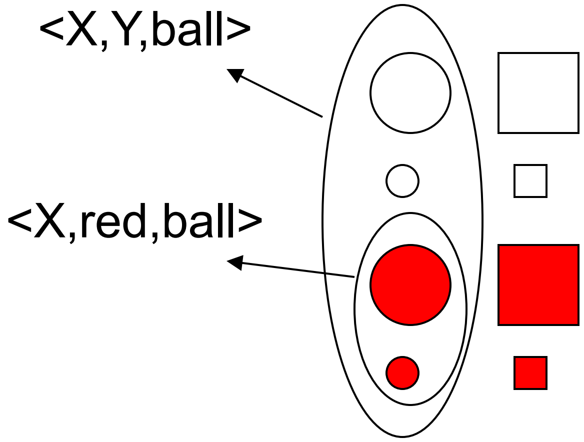
- Learning
  - Search through a concept space
- Concept representation
  - Conjunctive logical description

    $\forall Y$, color(Y, red) $\wedge$ shape(Y, ball) → $\boxed{\text{<X, red, ball>}}$
- Concept space
  - a set of descriptions with a partial ordering by generality

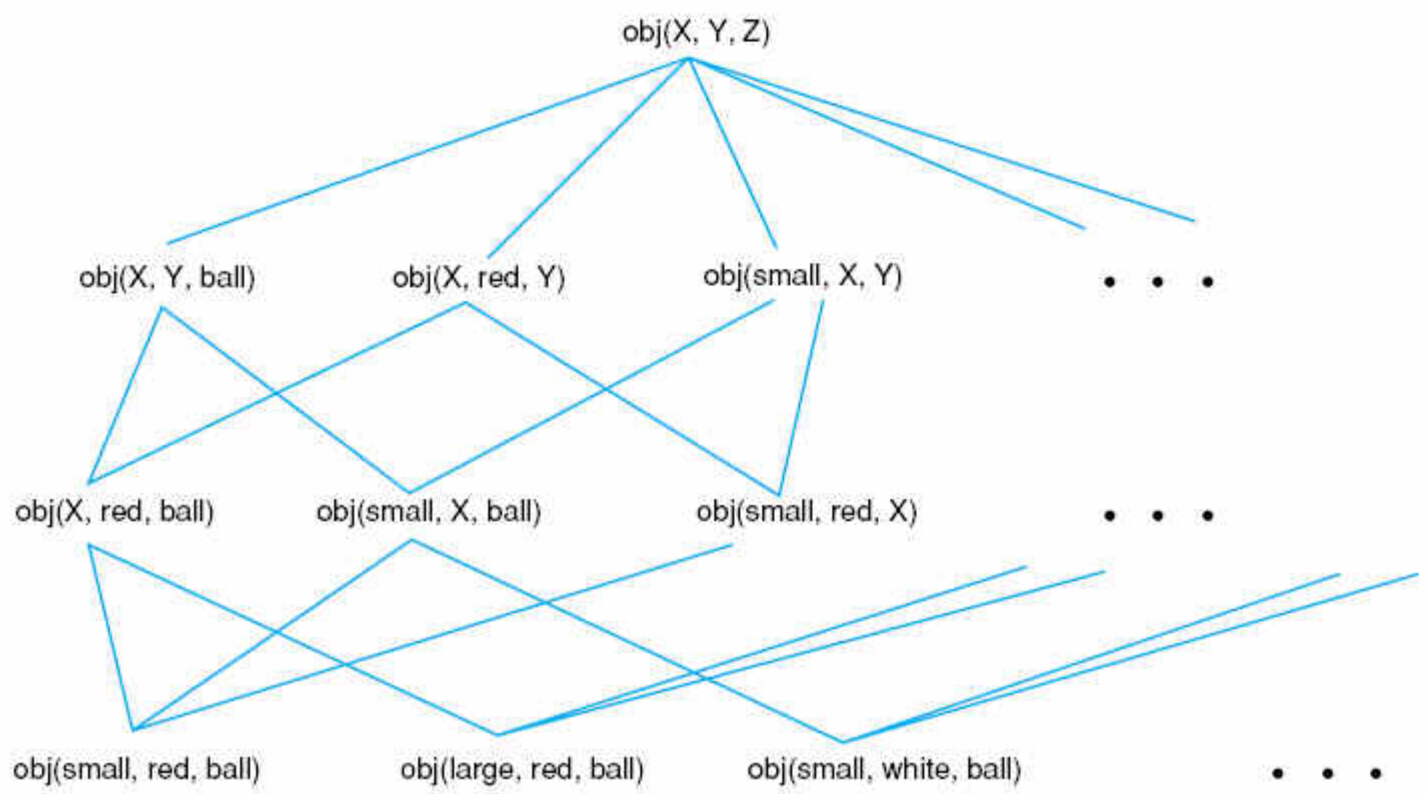
    <X, Y, ball> is more general than <X, red, ball>

Concept Representation

More general



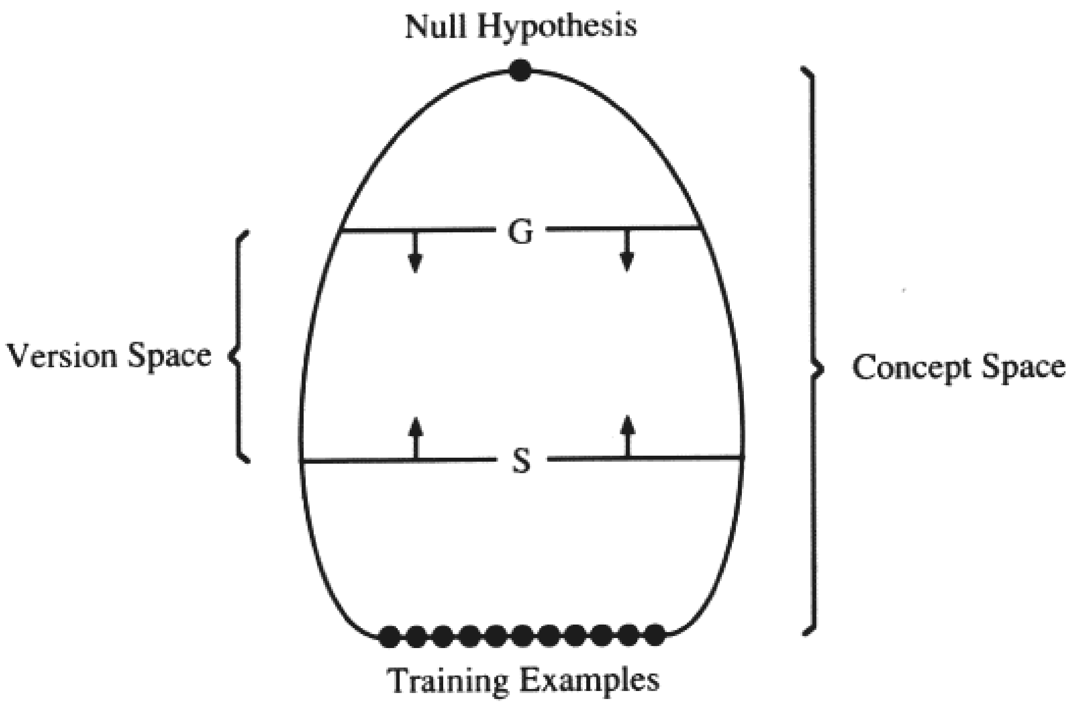
Concept Space



Version Space

- Version space
  - a set of descriptions consistent with example concepts
- Consistent concept example
  - Training examples
    - <small, red, ball> → Y
    - <large, red, ball> → Y
    - <small, red, cube> → N
  - Consistency
    - <X, red, ball>, <X, Y, ball>, … : Consistent (gives correct answer)
    - <X, red, Z>, <small, red, Z>, … : Inconsistent

Version Space Search



Version Space Search

- Algorithm

G:	Most general description consistent with examples
S:	Most specific descriptions consistent with examples

  - Start with G = most general, S = $\emptyset$
  - For positive examples: Make S more general
  - For negative examples: Make G more specific
  - Until G = S

⇨ ***Candidate elimination***

Version Space Search

- Generalization by a positive example

<small, Y, ball>
↑
<small, red, ball> ← +: <small, blue, ball>

- Specialization by a negative example

<X, Y, ball>
↙ ↘
<small, Y, ball> <X, red, ball> ← -: <large, blue, ball>

Example

G: <X, Y, Z>
S: ∅

+: <small, red, ball>

G: <X, Y, Z>
S: <small, red, ball>

-: <small, blue, ball>

G: ~~<large, Y, Z>~~ <X, red, Z> ~~<X, Y, cube>~~
S: <small, red, ball>

+: <large, red, ball>

G: <X, red, Z>
S: <X, red, ball>

-: <large, red, cube>

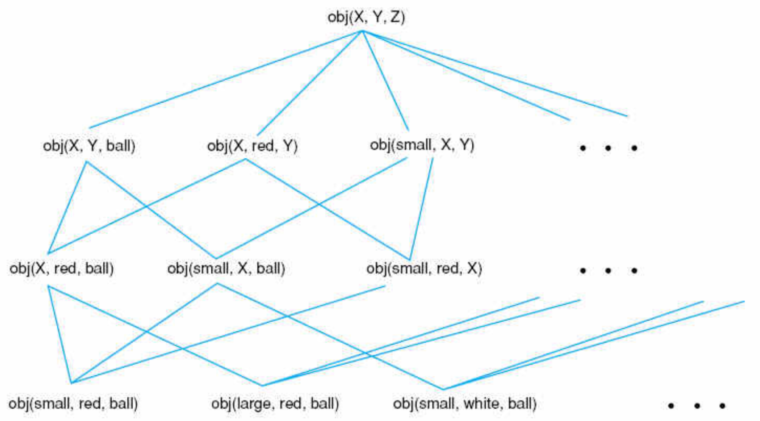
G: ~~<small, red, Z>~~ <X, red, ball>
S: <X, red, ball>

Limitation

- Binary function
  - Learning Y(concept)/N(not concept) function only

    Ex> f(x): x → A or B or C ?
- Conjunctive function description
  - Learning conjunctive description only

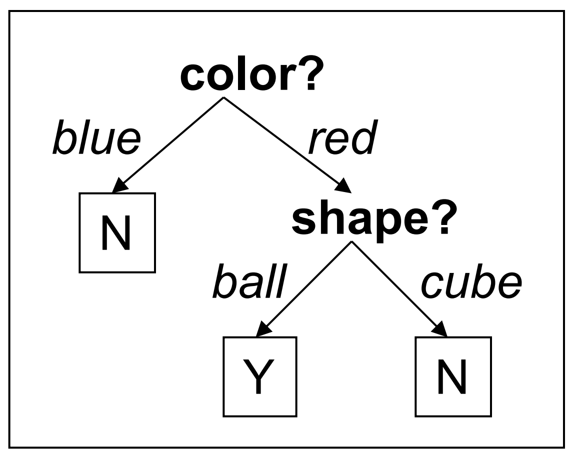
    Ex> $f(x) = (a \wedge b) \vee c$?



Decision Tree

- Decision tree
  - Representing the model(classifier) as a tree
    - $X : (A_1, A_2, \ldots, A_n)$, $f(X) = C_1$ or C_2 or … C_m
    - Internal nodes: attributes (A_i)
    - Edges: different attribute values
    - Leaves: answer (C_k)

X = <large, red, ball> →

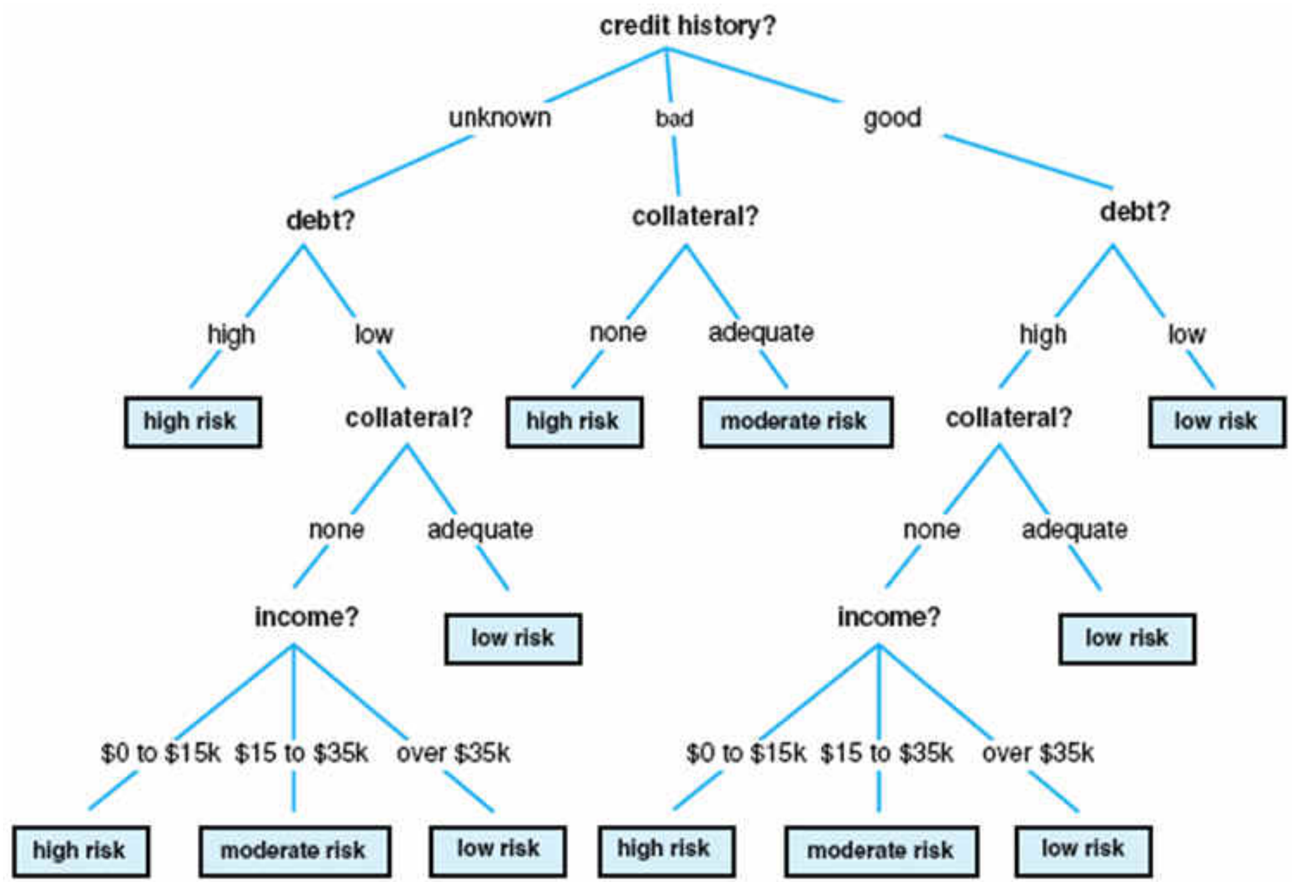


→ f(X) = Y

Example

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	\$0 to \$15k
2.	high	unknown	high	none	\$15 to \$35k
3.	moderate	unknown	low	none	\$15 to \$35k
4.	high	unknown	low	none	\$0 to \$15k
5.	low	unknown	low	none	over \$35k
6.	low	unknown	low	adequate	over \$35k
7.	high	bad	low	none	\$0 to \$15k
8.	moderate	bad	low	adequate	over \$35k
9.	low	good	low	none	over \$35k
10.	low	good	high	adequate	over \$35k
11.	high	good	high	none	\$0 to \$15k
12.	moderate	good	high	none	\$15 to \$35k
13.	low	good	high	none	over \$35k
14.	high	bad	high	none	\$15 to \$35k

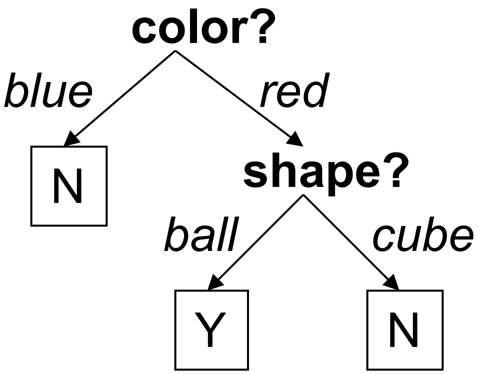
Example


credit history?
unknown
bad
good
debt?
collateral?
debt?
high
low
none
adequate
high
low
high risk
collateral?
high risk
moderate risk
collateral?
low risk
none
adequate
none
adequate
income?
low risk
income?
low risk
$0 to $15k
$15 to $35k
over $35k
$0 to $15k
$15 to $35k
over $35k
high risk
moderate risk
low risk
high risk
moderate risk
low risk

Decision Tree Learning

- Goal
  - Build the (smallest) decision tree consistent with examples

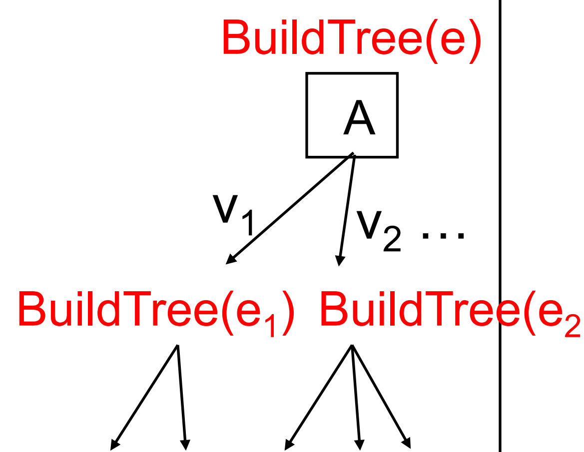
<small, red, ball> → Y
<small, blue, ball> → N
<large, red, ball> → Y
<large, red, cube> → N



- Method
  - Recursively choose an attribute and split examples

Decision Tree Learning

function **BuildTree** (e: set of examples)
  if (all examples $\in C_k$) return a leaf node C_k
  if (no example left) return majority class from parent
  if (no attribute left) return majority class
  else
    root ← choose an attribute A
    for each value v_i of A
      e_i ← examples with A = v_i
      subtree ← **BuildTree** (e_i)
      add branch v_i and subtree
    return (root)



Example

NO.	RISK	CREDIT HISTORY	DEBT	COLLATERAL	INCOME
1.	high	bad	high	none	$0 to $15k
2.	high	unknown	high	none	$15 to $35k
3.	moderate	unknown	low	none	$15 to $35k
4.	high	unknown	low	none	$0 to $15k
5.	low	unknown	low	none	over $35k
6.	low	unknown	low	adequate	over $35k
7.	high	bad	low	none	$0 to $15k
8.	moderate	bad	low	adequate	over $35k
9.	low	good	low	none	over $35k
10.	low	good	high	adequate	over $35k
11.	high	good	high	none	$0 to $15k
12.	moderate	good	high	none	$15 to $35k
13.	low	good	high	none	over $35k
14.	high	bad	high	none	$15 to $35k

Example

H: 1, 2, 4, 7, 11, 14
M: 3, 8, 12
L: 5, 6, 9, 10, 13

income?

$0-$15

H: 1, 4, 7, 11
M:
L:

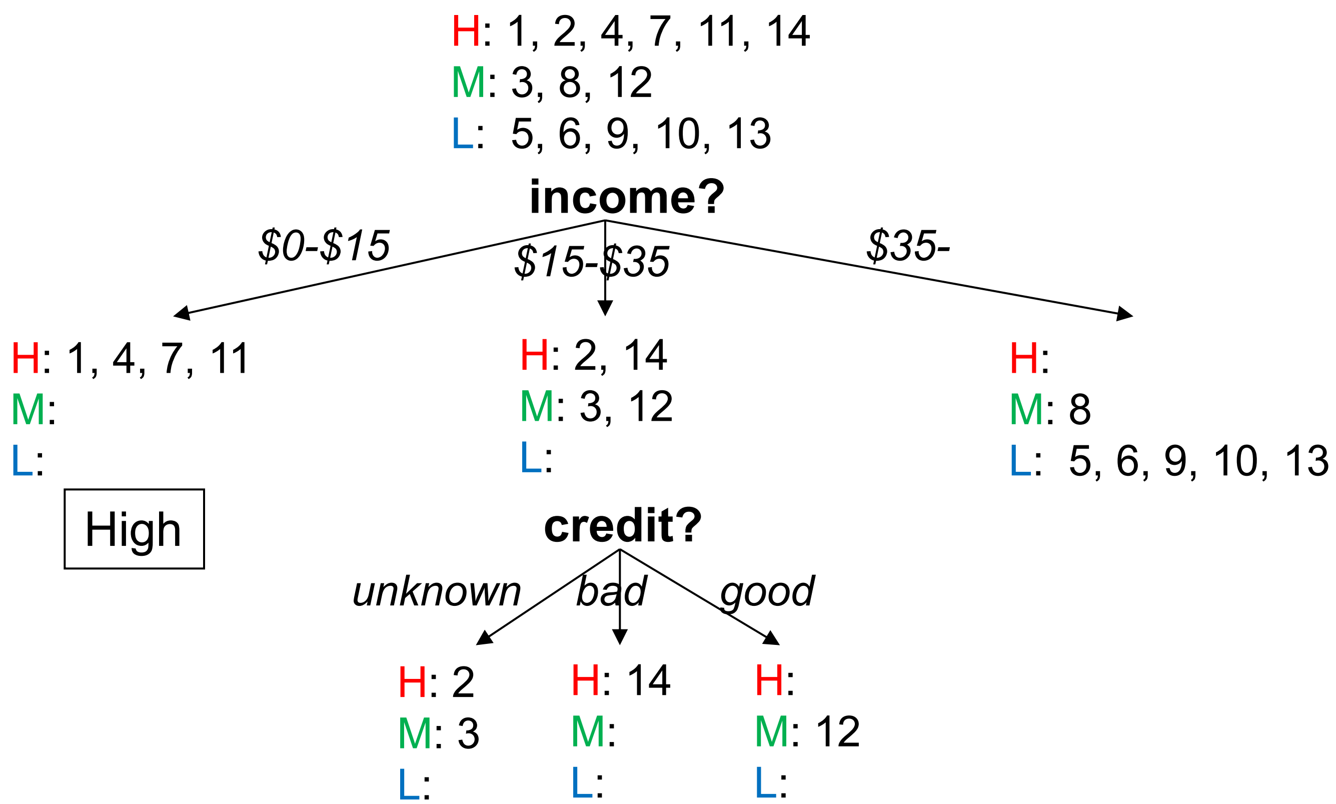
$15-$35

H: 2, 14
M: 3, 12
L:

$35-

H:
M: 8
L: 5, 6, 9, 10, 13

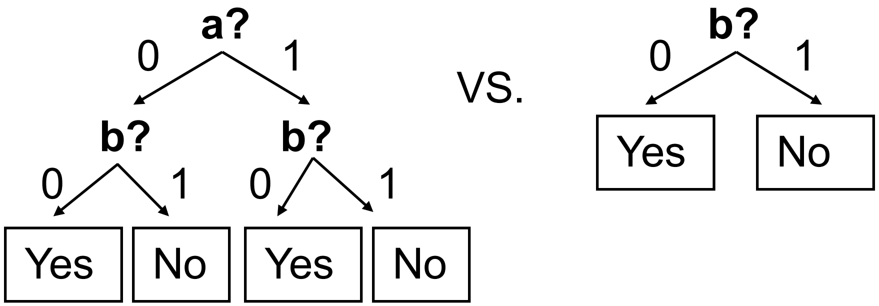
Example



H: 1, 2, 4, 7, 11, 14
M: 3, 8, 12
L: 5, 6, 9, 10, 13
income?
$0-$15
$15-$35
$35-
H: 1, 4, 7, 11
M:
L:
High
H: 2, 14
M: 3, 12
L:
H:
M: 8
L: 5, 6, 9, 10, 13
credit?
unknown
bad
good
H: 2
M: 3
L:
H: 14
M:
L:
H:
M: 12
L:

Selection of Attributes

- Different selection → different tree
  - (a=0, b=0) → Yes
    (a=0, b=1) → No
    (a=1, b=0) → Yes
    (a=1, b=1) → No



- Rule
  - Select A that classifies most examples
  - Selection based on information theory

Information Gain

- Amount of information
  - Depend on the probability
    - Example> Rolling a die

      Knowing that output is 2 (prob.=1/6) → most information

      Knowing that output is even (prob.=1/2)

      Knowing that output is 1~6 (prob.=1) → zero information
  - Low prob. → high info.
  - Prob. 1 → 0 info.

$$\boxed{I(m) = \log_2\left(\frac{1}{p(m)}\right) = -\log_2(p(m))}$$

Information Gain

- Entropy of a set
  - M = {m_1, m_2, … , m_n}, m_i has probability $p(m_i)$
  - E(M) = expected amount of information

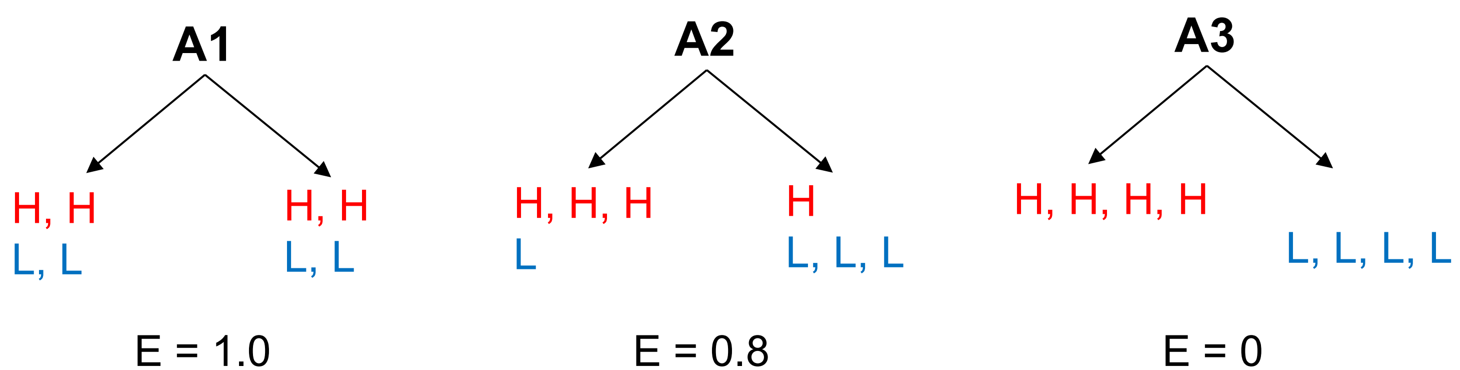
$$\boxed{E(M) = \sum p(m_i) I(m_i) = \sum -p(m_i)\log(p(m_i))}$$

  - Examples
    - M: {H(1,2), L(3,4)} → $E(M) = \left(-\frac{1}{2}\log\frac{1}{2}\right) + \left(-\frac{1}{2}\log\frac{1}{2}\right) = 1.0$
    - M: {H(1,2,3), L(4)} → $E(M) = \left(-\frac{3}{4}\log\frac{3}{4}\right) + \left(-\frac{1}{4}\log\frac{1}{4}\right) = 0.81$
    - M: {H(1,2,3,4), L()} → $E(M) = \left(-\frac{4}{4}\log\frac{4}{4}\right) + \left(-\frac{0}{4}\log\frac{0}{4}\right) = 0.0$

Information Gain

H, H, H, H
L, L, L, L

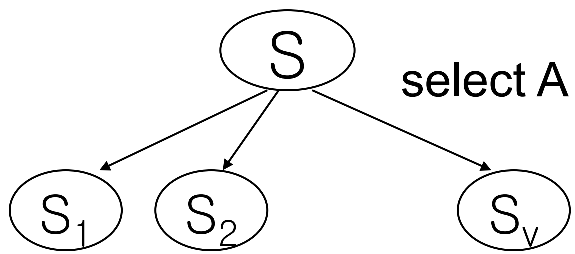
E = -1/2log1/2 -1/2log1/2 = 1.0



- A1 < A2 < A3 reduces more entropy
- A1 < A2 < A3 has more information for H/L decision

Information Gain

- Attribute selection
  - Select A that reduces entropy most
    - Entropy = 0 → all examples are in one category → leaf
  - Gain = current entropy – expected entropy after check A
  - Select A with largest gain



$$Gain(A) = E(S) - \sum \frac{|S_i|}{|S|} E(S_i)$$

Example

S = {H(1,2,4,7,11,14), M(3,8,12), L(5,6,9,10,13)}

$$\mathbf{E(S)} = \left(-\frac{6}{14}\log\frac{6}{14}\right)+\left(-\frac{3}{14}\log\frac{3}{14}\right)+\left(-\frac{5}{14}\log\frac{5}{14}\right)=1.53$$

If we select A = income,

income?

H: 1,4,7,11	H: 2,14	H:
M:	M: 3,12	M: 8
L:	L:	L: 5,6,9,10,13

$$\mathbf{E(S_1)} = \left(-\frac{4}{4}\log\frac{4}{4}\right)+0+0=0.0$$

$$\mathbf{E(S_2)} = \left(-\frac{2}{4}\log\frac{2}{4}\right)+\left(-\frac{2}{4}\log\frac{2}{4}\right)+0=1.0$$

$$\mathbf{E(S_3)} = 0+\left(-\frac{1}{6}\log\frac{1}{6}\right)+\left(-\frac{5}{6}\log\frac{5}{6}\right)=0.65$$

$$\mathbf{Gain(A{=}income)} = 1.53-\left(\frac{4}{14}\times 0.0+\frac{4}{14}\times 1.0+\frac{6}{14}\times 0.65\right)=0.97$$

Example

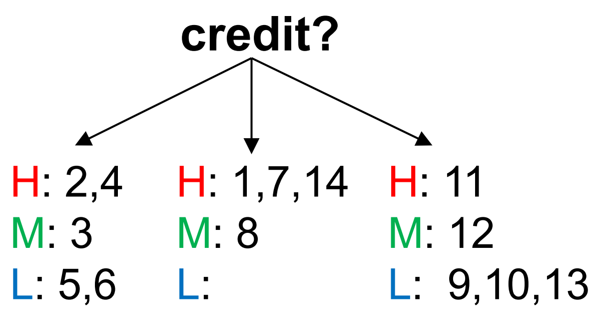
S = {H(1,2,4,7,11,14), M(3,8,12), L(5,6,9,10,13)}

Gain(A=income) = 0.97

Gain(A=credit) =0.27

Gain(A=debt) = 0.58

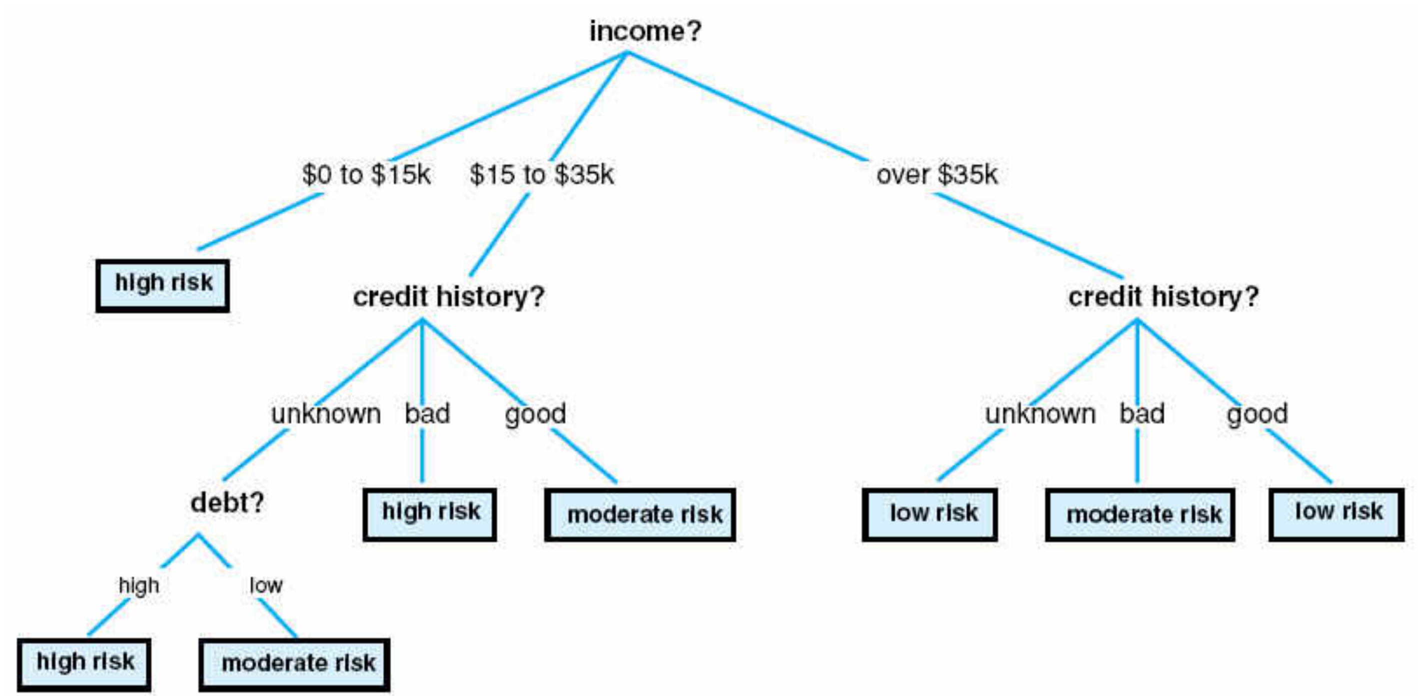
Gain(A=collateral) = 0.76



⇨ Select ‘income’ for partition

Example

- Result tree



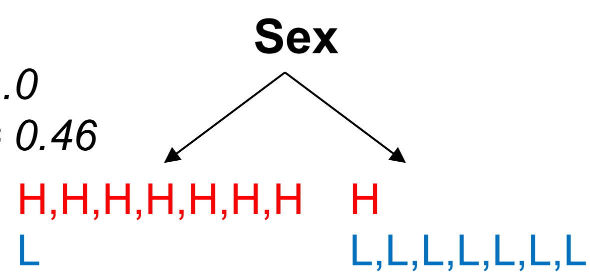
Gain Ratio

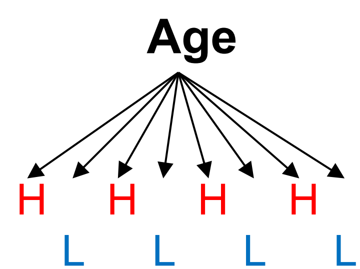
- Problem
  - Information gain measure is biased towards attributes with a large number of values
- GainRatio
  - Normalization to information gain

$$SplitInfo_A(S) = -\sum_{j=1}^{v} \frac{|S_j|}{|S|} \times \log\left(\frac{|S_j|}{|S|}\right)$$

$$GainRatio(A) = Gain(A) / SplitInfo_A$$

Gain = 0.46
SplitInfo = 1.0
GainRatio = 0.46

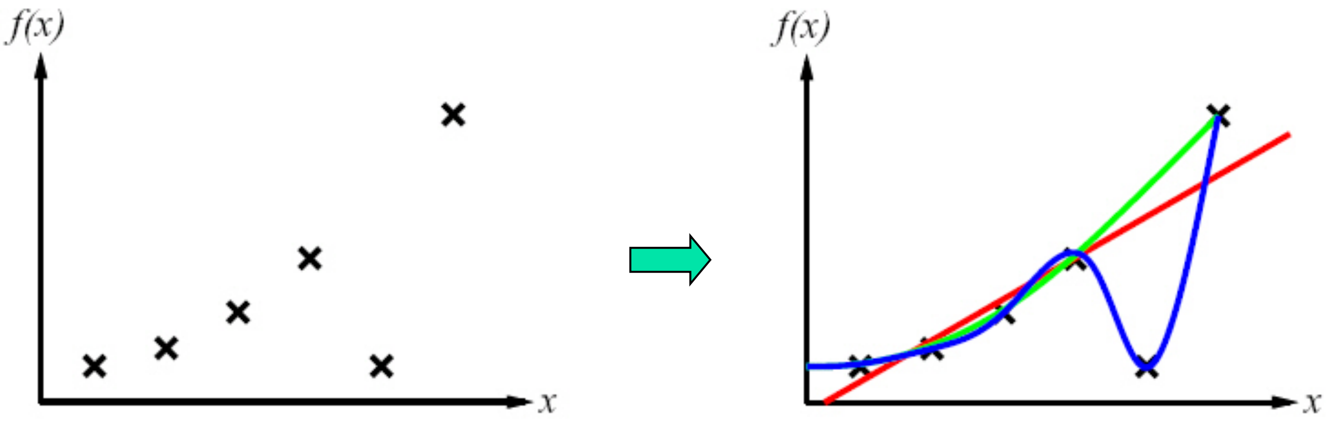




Gain = 1.0
SplitInfo = 3.0
GainRatio = 0.33

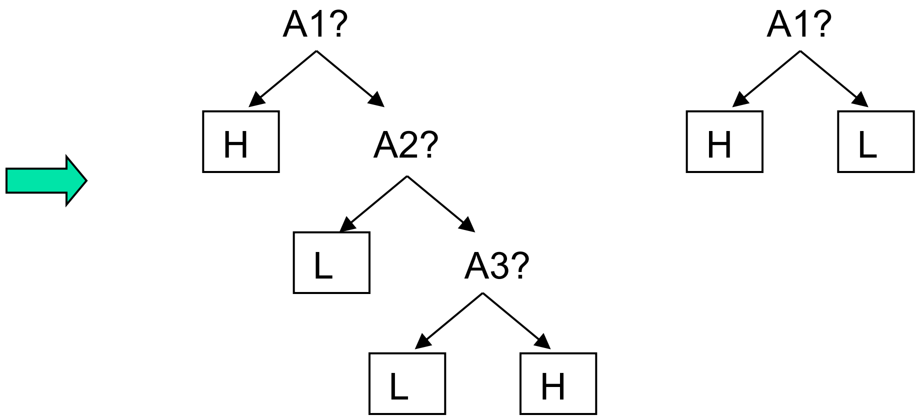
Overfitting

- The generated model may overfit to the training data
  - Too many branches, some may reflect anomalies due to *noise* or *outliers*
  - Result is in poor accuracy for unseen samples



Pruning

A1	A2	A3	Class
0	0	0	H
0	0	1	H
0	1	0	H
0	1	1	H
1	0	0	L
1	0	1	L
1	1	0	L
1	1	1	H



- Prepruning
  - Halt tree construction early
  - Do not split a node if this would result in the goodness measure falling below a threshold
- Postpruning
  - Remove branches from a "fully grown" tree
  - If pruning a node lead to a smaller error rate (with test set), prune it

Performance of Learning

- Performance measure
  1. Collect set of examples
  2. Divide it into training set / test set
  3. Learning by using training set
  4. Measure the % of correct answer on test set
- K-fold cross-validation
  - Divide the data set into *k* subsets
  - Use k-1 subsets as training data and 1 subset as test data
  - Repeat k times, and average the accuracy

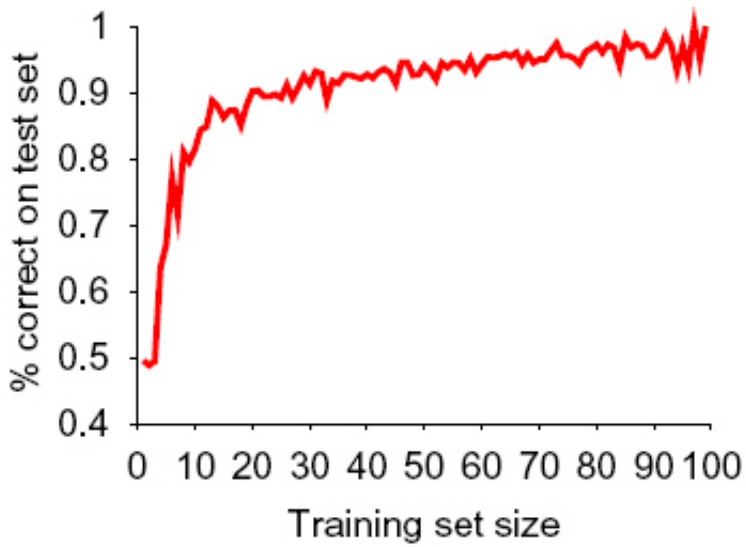
S_1, S_2, S_3, S_4, S_5

$\rightarrow S_1, S_2, S_3, S_4, S_5$

$\rightarrow S_1, S_2, S_3, S_4, S_5$

Performance of Learning

- The learning curve
  - Measure the % of correct classification with different size of randomly selected training sets

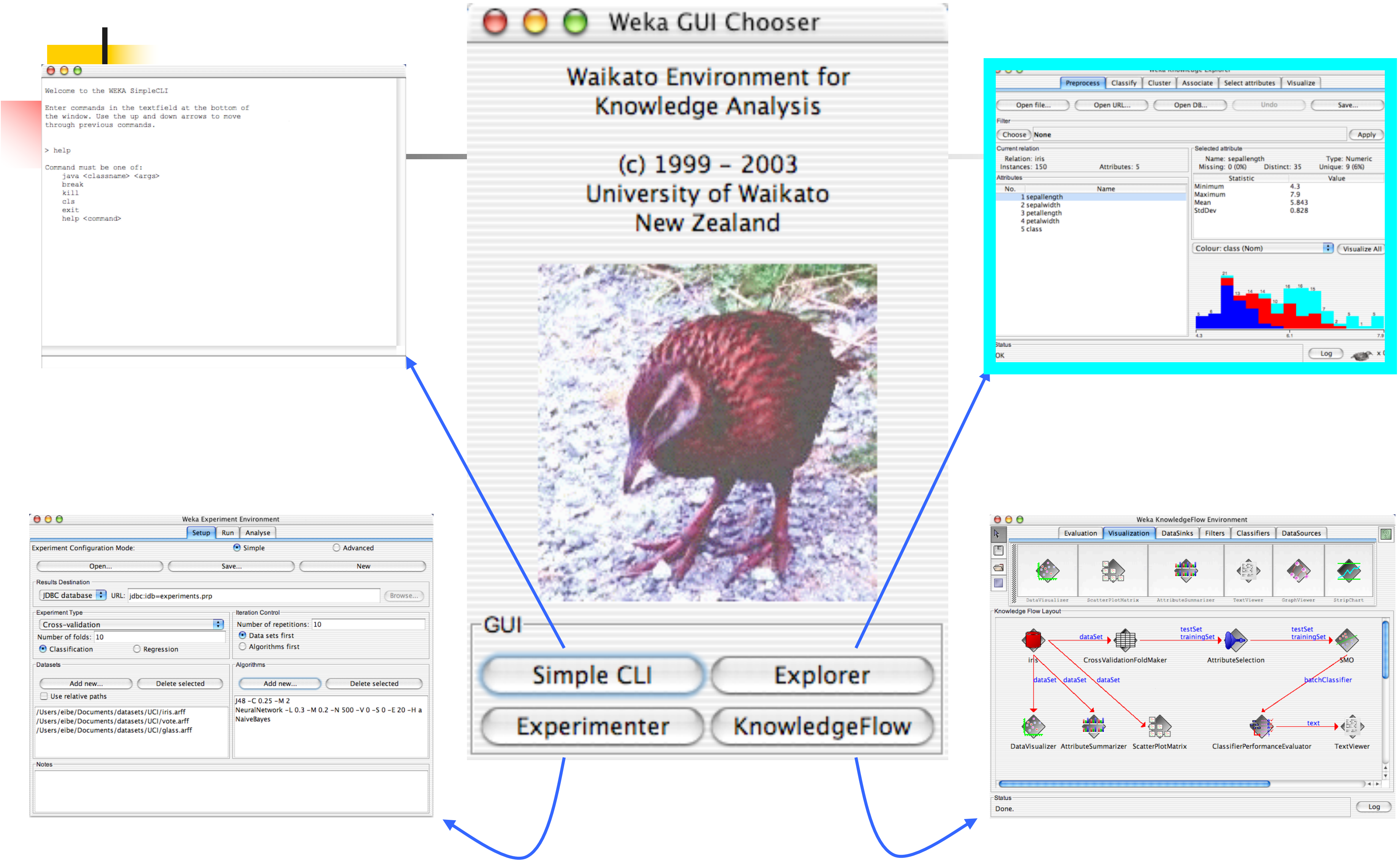


WEKA

- Machine learning/data mining software written in Java
  - http://www.cs.waikato.ac.nz/ml/weka
  - Used for research, education, and applications
  - Complements "Data Mining" by Witten & Frank
- Main features
  - Comprehensive set of data pre-processing tools, learning algorithms and evaluation methods
  - Graphical user interfaces (incl. data visualization)
  - Environment for comparing learning algorithms

Data Files (ARFF)

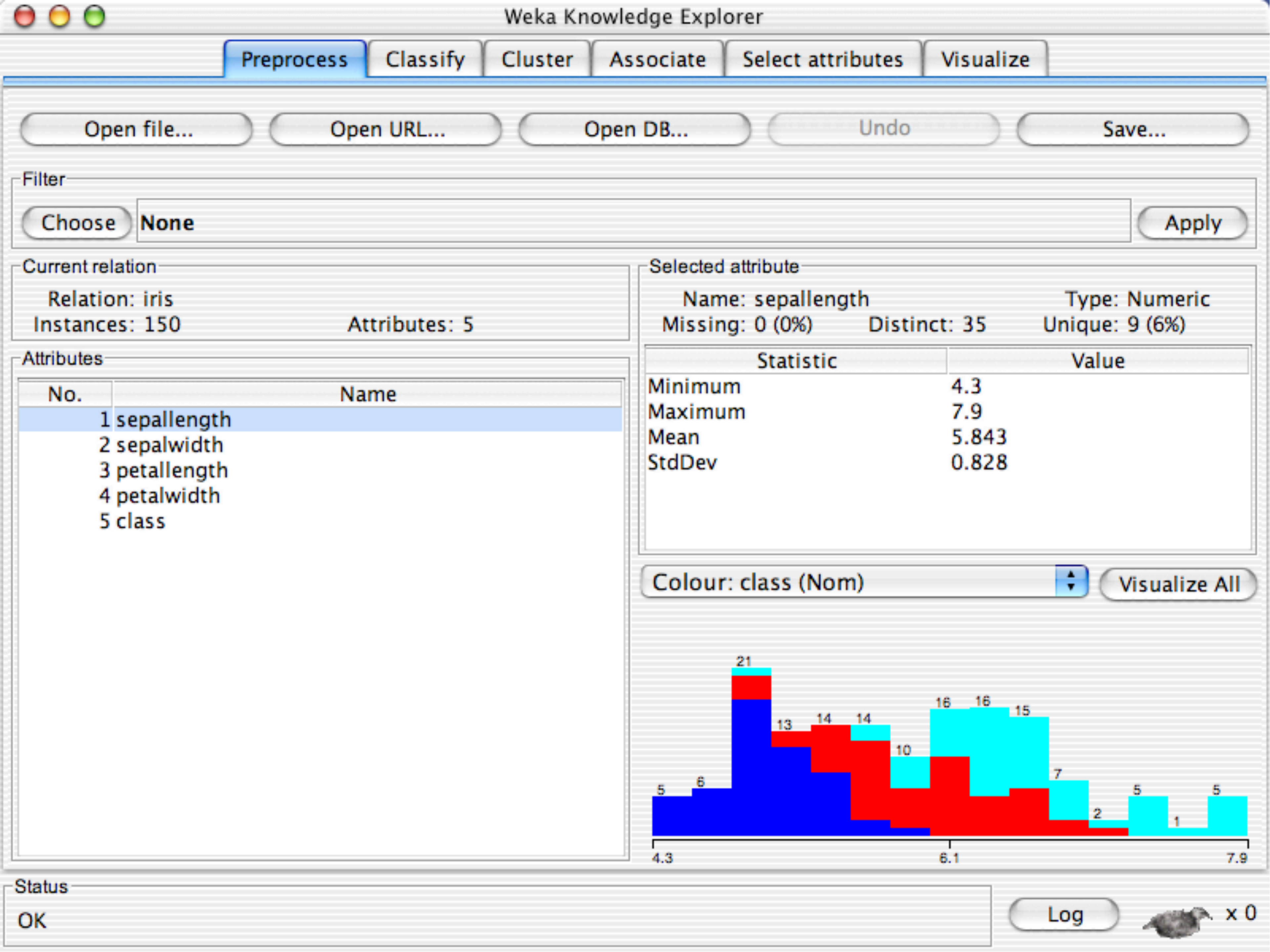
```
@relation heart-disease-simplified

@attribute age numeric
@attribute sex {female, male}
@attribute chest_pain_type {typ_angina, asympt, non_anginal,
                                                   typ_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina {no, yes}
@attribute class {present, not_present}

@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
...
```



Weka GUI Chooser
Waikato Environment for Knowledge Analysis
(c) 1999 – 2003
University of Waikato
New Zealand
GUI
Simple CLI
Explorer
Experimenter
KnowledgeFlow
Welcome to the WEKA SimpleCLI
Enter commands in the textfield at the bottom of the window. Use the up and down arrows to move through previous commands.
> help
Command must be one of:
java <classname> <args>
break
kill
cls
exit
help <command>
Weka Experiment Environment
Setup
Run
Analyse
Experiment Configuration Mode:
Simple
Advanced
Open...
Save...
New
Results Destination
JDBC database
URL: jdbc:idb=experiments.prp
Browse...
Experiment Type
Cross-validation
Number of folds: 10
Classification
Regression
Iteration Control
Number of repetitions: 10
Data sets first
Algorithms first
Datasets
Add new...
Delete selected
Use relative paths
/Users/eibe/Documents/datasets/UCI/iris.arff
/Users/eibe/Documents/datasets/UCI/vote.arff
/Users/eibe/Documents/datasets/UCI/glass.arff
Algorithms
J48 -C 0.25 -M 2
NeuralNetwork -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
NaiveBayes
Notes
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Undo
Save...
Filter
Choose
None
Apply
Current relation
Relation: iris
Instances: 150
Attributes: 5
Selected attribute
Name: sepallength
Type: Numeric
Missing: 0 (0%)
Distinct: 35
Unique: 9 (6%)
Attributes
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Colour: class (Nom)
Visualize All
Status
OK
Log
Weka KnowledgeFlow Environment
Evaluation
Visualization
DataSinks
Filters
Classifiers
DataSources
Knowledge Flow Layout
iris
dataSet
CrossValidationFoldMaker
testSet
trainingSet
AttributeSelection
SMO
batchClassifier
DataVisualizer
AttributeSummarizer
ScatterPlotMatrix
ClassifierPerformanceEvaluator
text
TextViewer
Status
Done.
Log

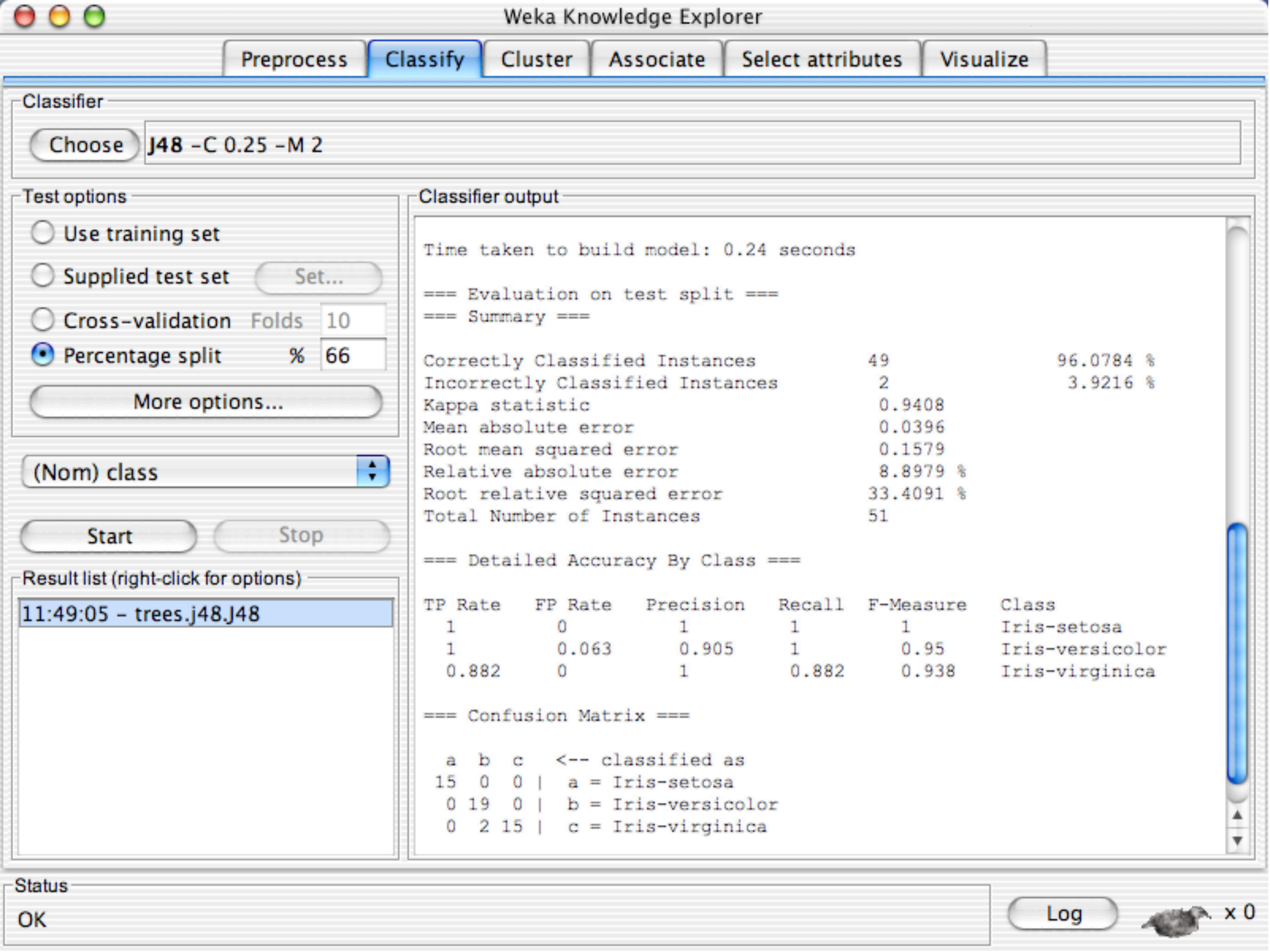
Explorer: pre-processing

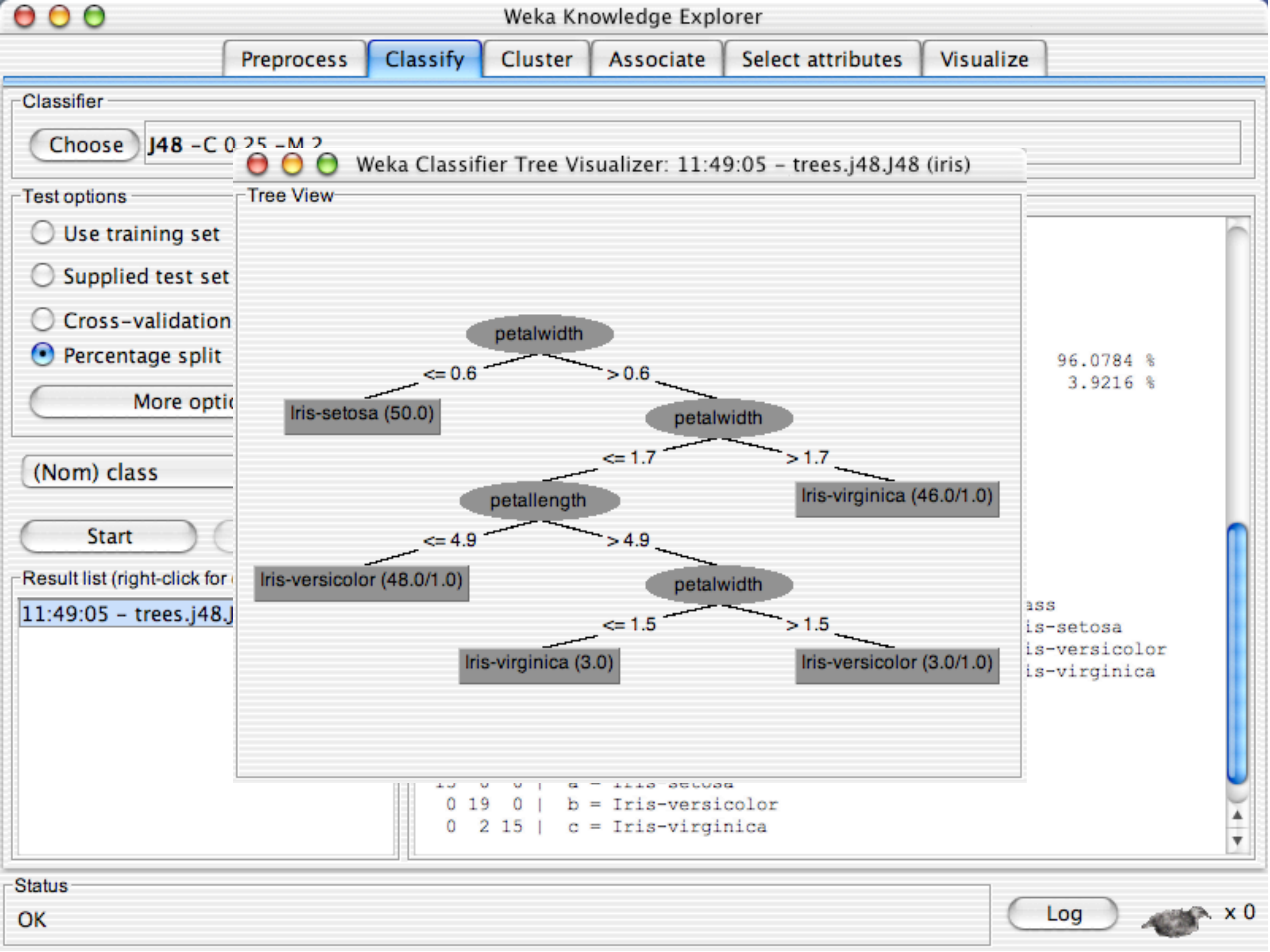
- Source
  - Data can be imported from a file in various formats: ARFF, CSV, C4.5, binary
  - Data can also be read from a URL or from an SQL database (using JDBC)
- Pre-processing tools
  - Called “filters”
  - Discretization, normalization, resampling, attribute selection, transforming and combining attributes, …

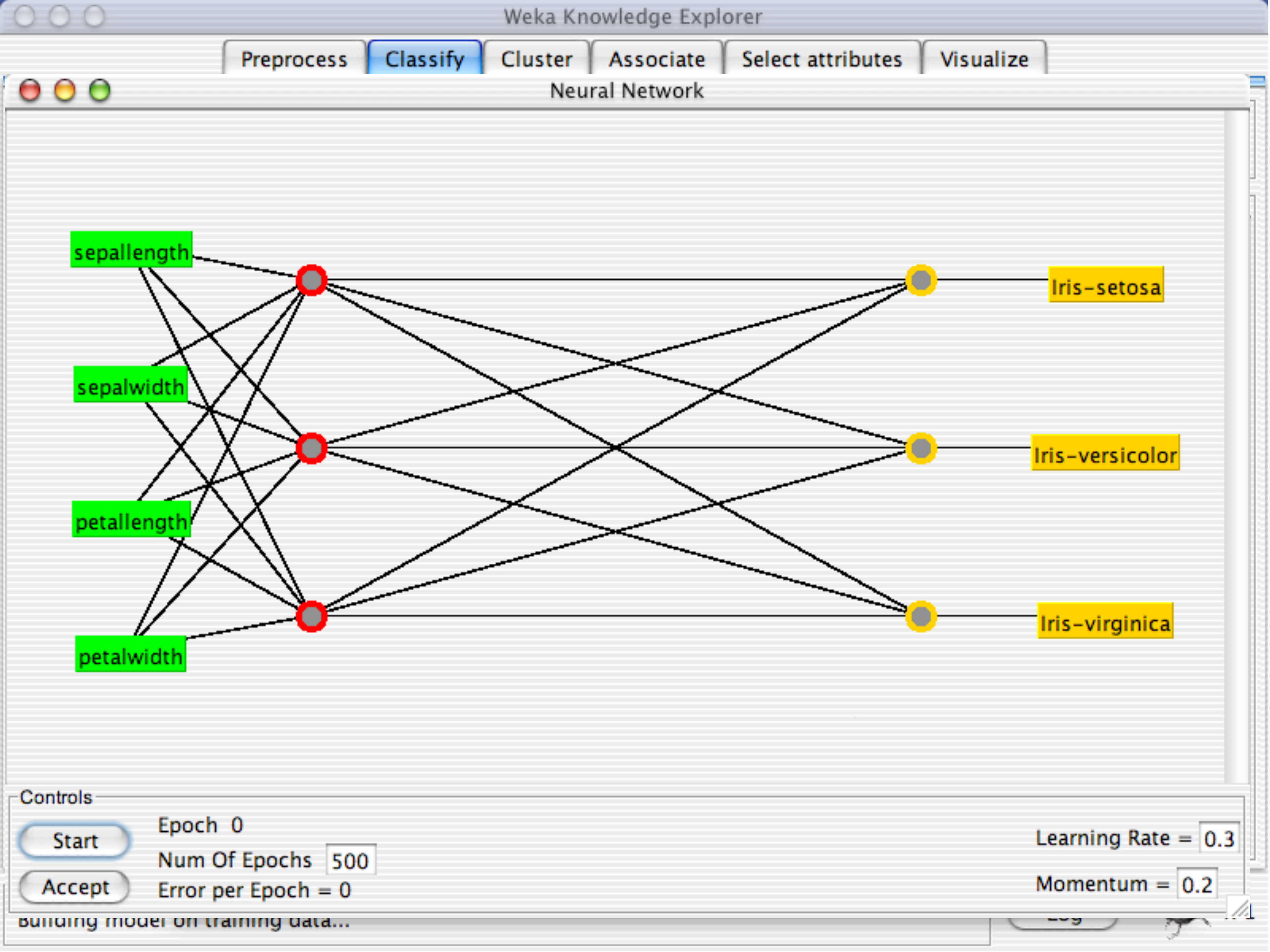

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Undo
Save...
Filter
Choose
None
Apply
Current relation
Relation: iris
Instances: 150
Attributes: 5
Attributes
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Selected attribute
Name: sepallength
Type: Numeric
Missing: 0 (0%)
Distinct: 35
Unique: 9 (6%)
Statistic
Value
Minimum 4.3
Maximum 7.9
Mean 5.843
StdDev 0.828
Colour: class (Nom)
Visualize All
5
6
21
13
14
14
10
16
16
15
7
2
5
1
5
4.3
6.1
7.9
Status
OK
Log
x 0

Explorer: building "classifiers"

- Classifiers in WEKA are models for predicting nominal or numeric quantities
- Implemented learning schemes include:
  - Decision trees and lists, instance-based classifiers, support vector machines, multi-layer perceptrons, logistic regression, Bayes' nets, …
- "Meta"-classifiers include:
  - Bagging, boosting, stacking, error-correcting output codes, locally weighted learning, …


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
11:49:05 - trees.j48.J48
Classifier output
Time taken to build model: 0.24 seconds
=== Evaluation on test split ===
=== Summary ===
Correctly Classified Instances 49 96.0784 %
Incorrectly Classified Instances 2 3.9216 %
Kappa statistic 0.9408
Mean absolute error 0.0396
Root mean squared error 0.1579
Relative absolute error 8.8979 %
Root relative squared error 33.4091 %
Total Number of Instances 51
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure Class
1 0 1 1 1 Iris-setosa
1 0.063 0.905 1 0.95 Iris-versicolor
0.882 0 1 0.882 0.938 Iris-virginica
=== Confusion Matrix ===
a b c <-- classified as
15 0 0 | a = Iris-setosa
0 19 0 | b = Iris-versicolor
0 2 15 | c = Iris-virginica
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
Test options
Use training set
Supplied test set
Cross-validation
Percentage split
(Nom) class
Start
11:49:05 - trees.j48.J
Weka Classifier Tree Visualizer: 11:49:05 - trees.j48.J48 (iris)
Tree View
petalwidth
<= 0.6
> 0.6
Iris-setosa (50.0)
petalwidth
<= 1.7
> 1.7
petallength
Iris-virginica (46.0/1.0)
<= 4.9
> 4.9
Iris-versicolor (48.0/1.0)
petalwidth
<= 1.5
> 1.5
Iris-virginica (3.0)
Iris-versicolor (3.0/1.0)
96.0784 %
3.9216 %
0 19 0 | b = Iris-versicolor
0 2 15 | c = Iris-virginica
Status
OK
Log
x 0

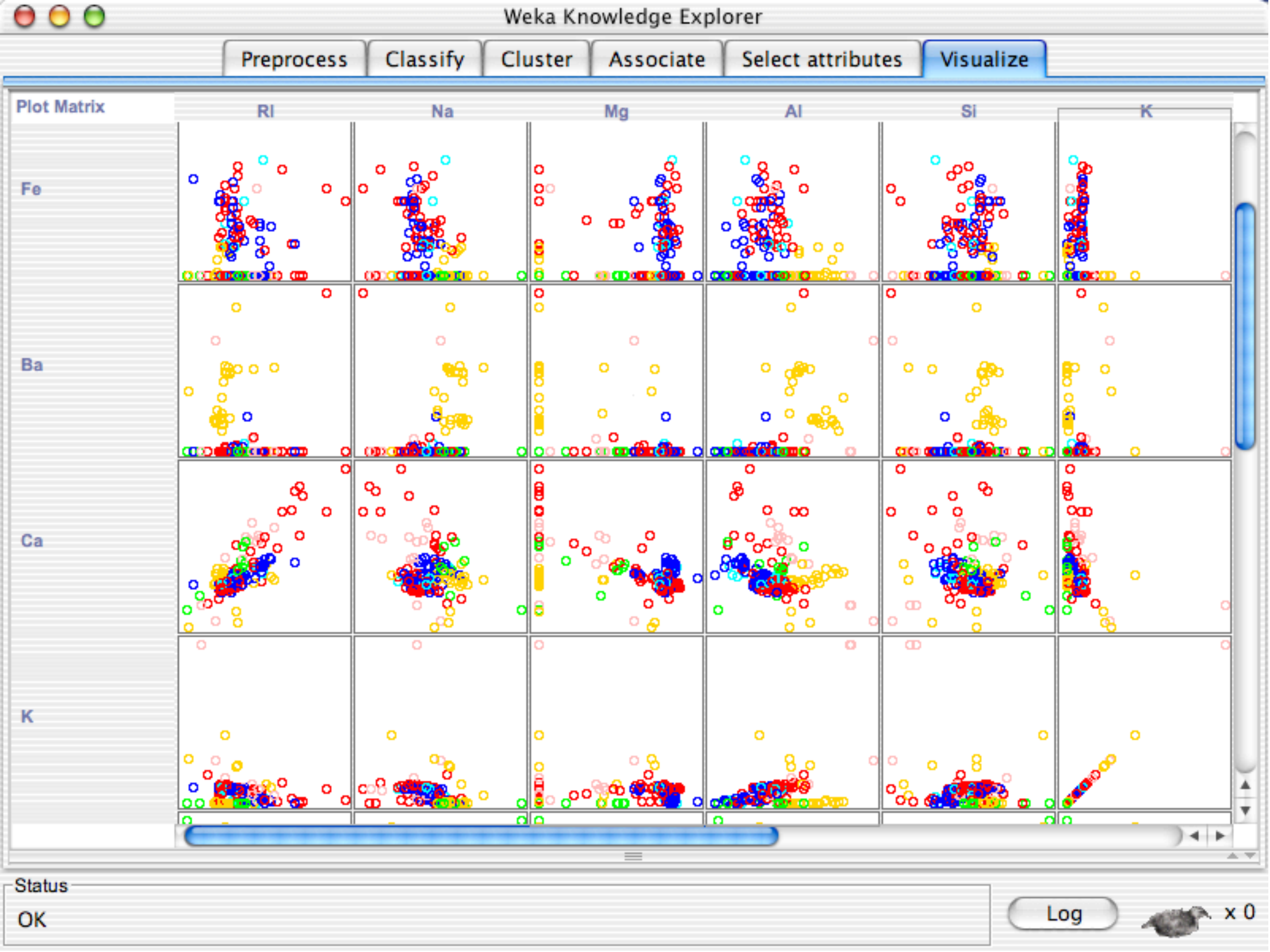

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Neural Network
sepallength
sepalwidth
petallength
petalwidth
Iris-setosa
Iris-versicolor
Iris-virginica
Controls
Start
Accept
Epoch 0
Num Of Epochs
500
Error per Epoch = 0
Learning Rate =
0.3
Momentum =
0.2

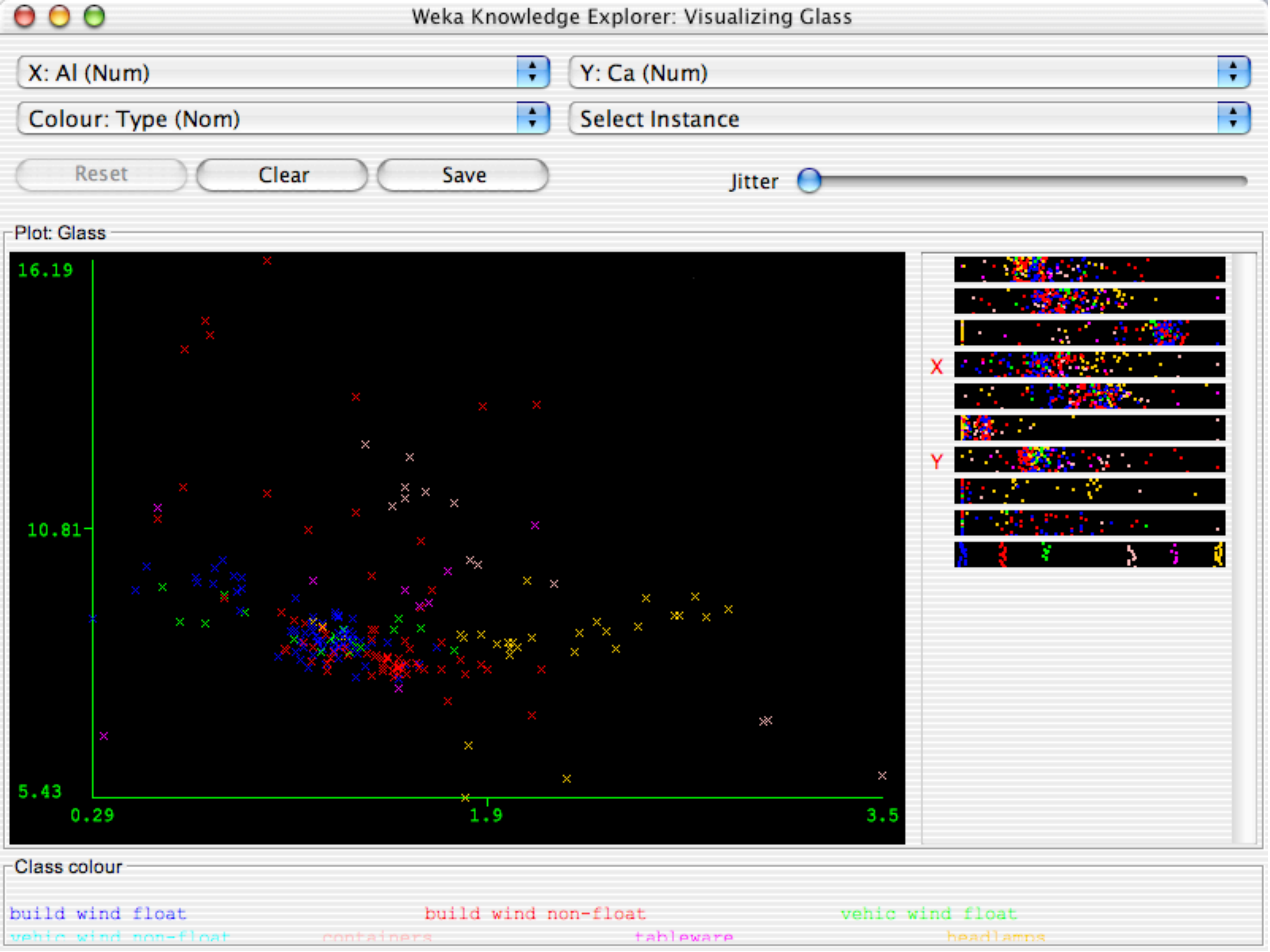
Explorer: clustering data

- WEKA contains “clusterers” for finding groups of similar instances in a dataset
- Implemented schemes are:
  - *k*-Means, EM, Cobweb, *X*-means, FarthestFirst
- Clusters can be visualized and compared to “true” clusters (if given)
- Evaluation based on loglikelihood if clustering scheme produces a probability distribution

Explorer: data visualization

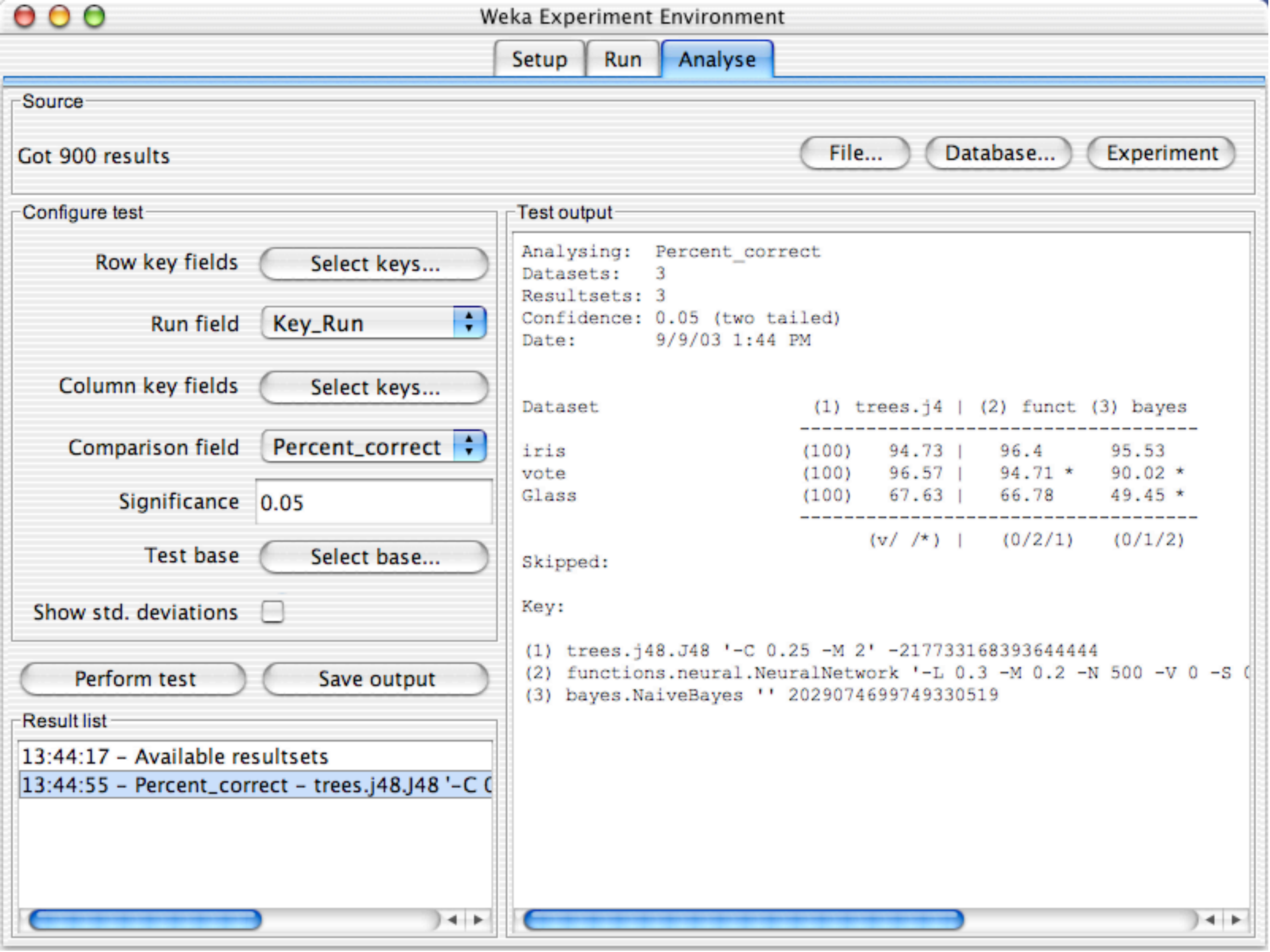
- Visualization very useful in practice: e.g. helps to determine difficulty of the learning problem
  - WEKA can visualize single attributes (1-d) and pairs of attributes (2-d)
  - Color-coded class values
  - “Jitter” option to deal with nominal attributes (and to detect “hidden” data points)
  - “Zoom-in” function


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Plot Matrix
RI
Na
Mg
Al
Si
K
Fe
Ba
Ca
K
Status
OK
Log
x 0


Weka Knowledge Explorer: Visualizing Glass
X: Al (Num)
Y: Ca (Num)
Colour: Type (Nom)
Select Instance
Reset
Clear
Save
Jitter
Plot: Glass
16.19
10.81
5.43
0.29
1.9
3.5
X
Y
Class colour
build wind float
build wind non-float
vehic wind float
vehic wind non-float
containers
tableware
headlamps

Performing experiments

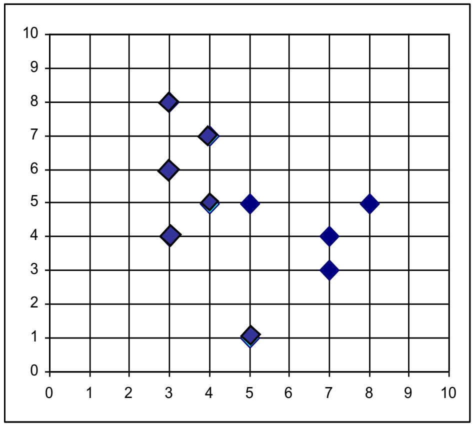
- Experimenter makes it easy to compare the performance of different learning schemes
  - For classification and regression problems
  - Results can be written into file or database
  - Evaluation options: cross-validation, learning curve, hold-out
  - Can also iterate over different parameter settings
  - Significance-testing built in!


Weka Experiment Environment
Setup
Run
Analyse
Source
Got 900 results
File...
Database...
Experiment
Configure test
Row key fields
Select keys...
Run field
Key_Run
Column key fields
Select keys...
Comparison field
Percent_correct
Significance
0.05
Test base
Select base...
Show std. deviations
Perform test
Save output
Result list
13:44:17 – Available resultsets
13:44:55 – Percent_correct – trees.j48.J48 '-C (
Test output
Analysing: Percent_correct
Datasets: 3
Resultsets: 3
Confidence: 0.05 (two tailed)
Date: 9/9/03 1:44 PM
Dataset (1) trees.j4 | (2) funct (3) bayes
iris (100) 94.73 | 96.4 95.53
vote (100) 96.57 | 94.71 * 90.02 *
Glass (100) 67.63 | 66.78 49.45 *
(v/ /*) | (0/2/1) (0/1/2)
Skipped:
Key:
(1) trees.j48.J48 '-C 0.25 -M 2' -217733168393644444
(2) functions.neural.NeuralNetwork '-L 0.3 -M 0.2 -N 500 -V 0 -S (
(3) bayes.NaiveBayes '' 2029074699749330519

Unsupervised Learning

- Class labels are not provided

< (3, 4), ~~Y~~ >
< (5, 1), ~~N~~ >
< (4, 7), ~~Y~~ >
< (5, 5), ~~Y~~ >
…

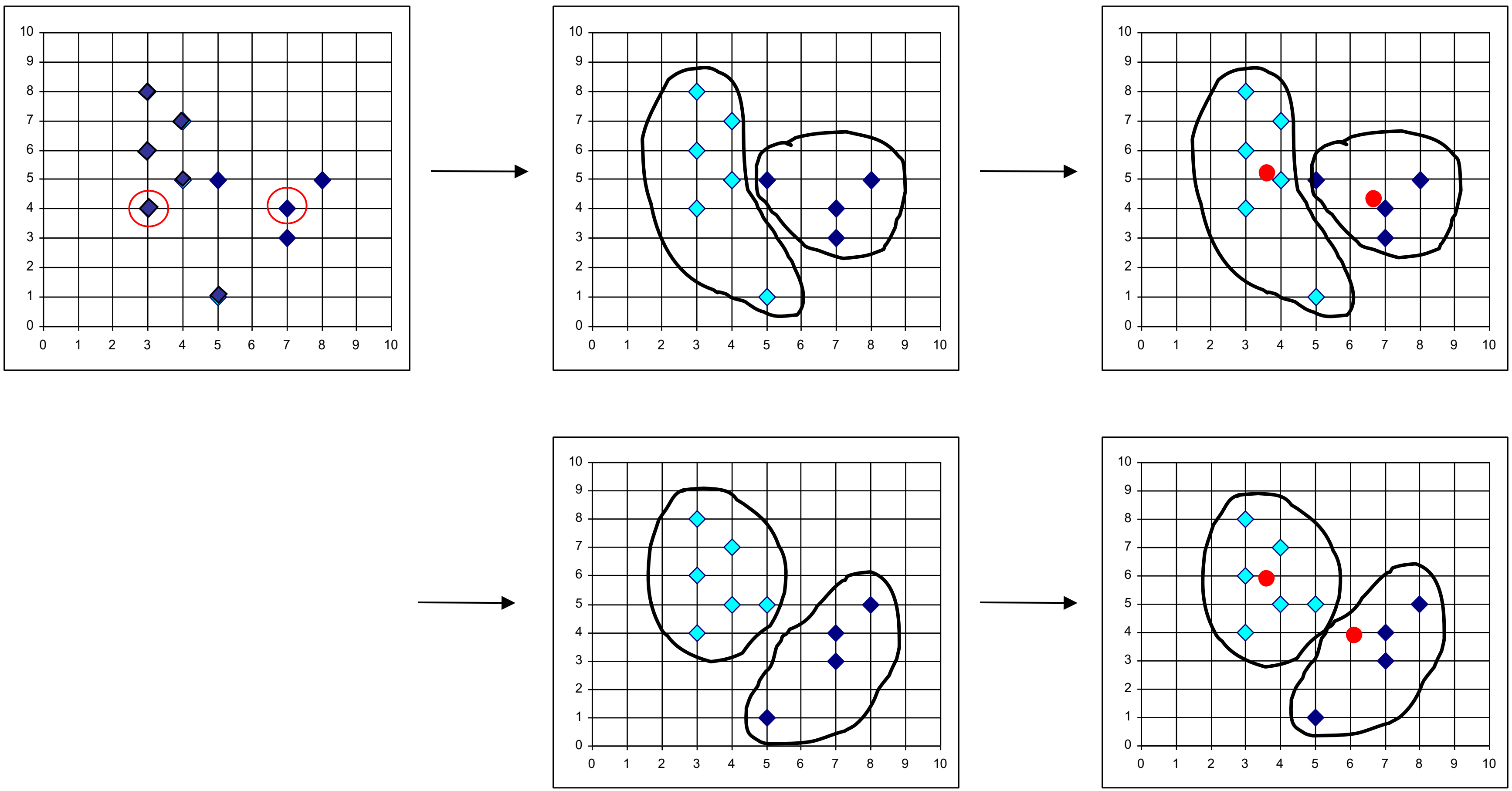


Clustering

K-Means Method

- K-means
  - Input: a set of unclassified data objects
  - Output: k clusters
- Algorithm
  1. **Choose k objects** as initial cluster centers
  2. **Assign each object to the cluster** with the nearest center
  3. **Update cluster centers** as the mean point of the cluster
  4. Go back to Step 2, stop when there is no change

K-Means Method

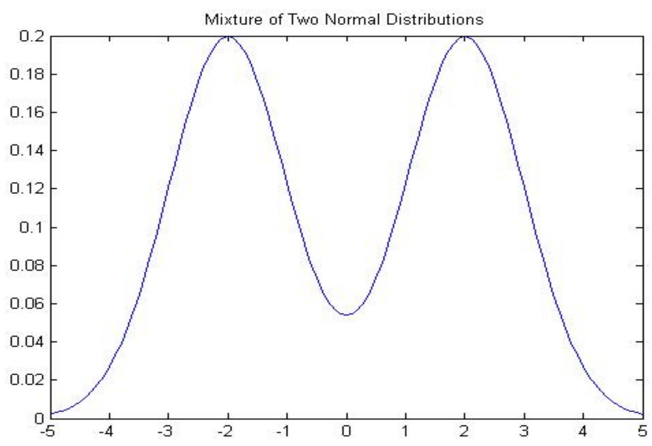


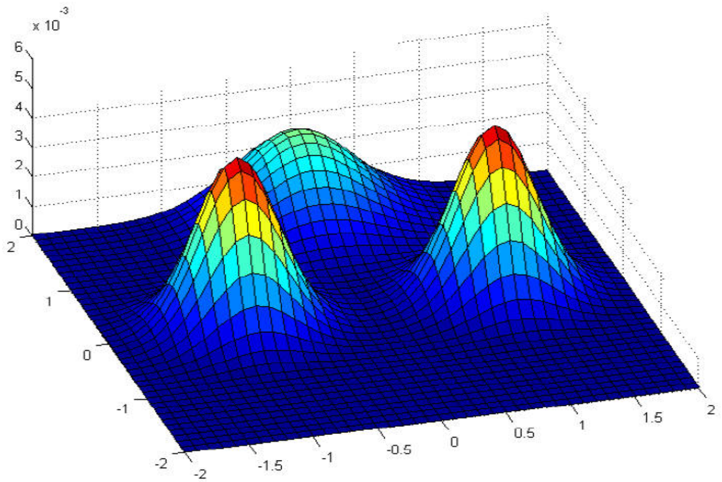
EM Algorithm

- Assumption
  - Data are generated from a mixture distribution with k components ($C_1, \ldots, C_k$)

$$P(x) = \sum_i P(C_i) P(x \mid C_i)$$

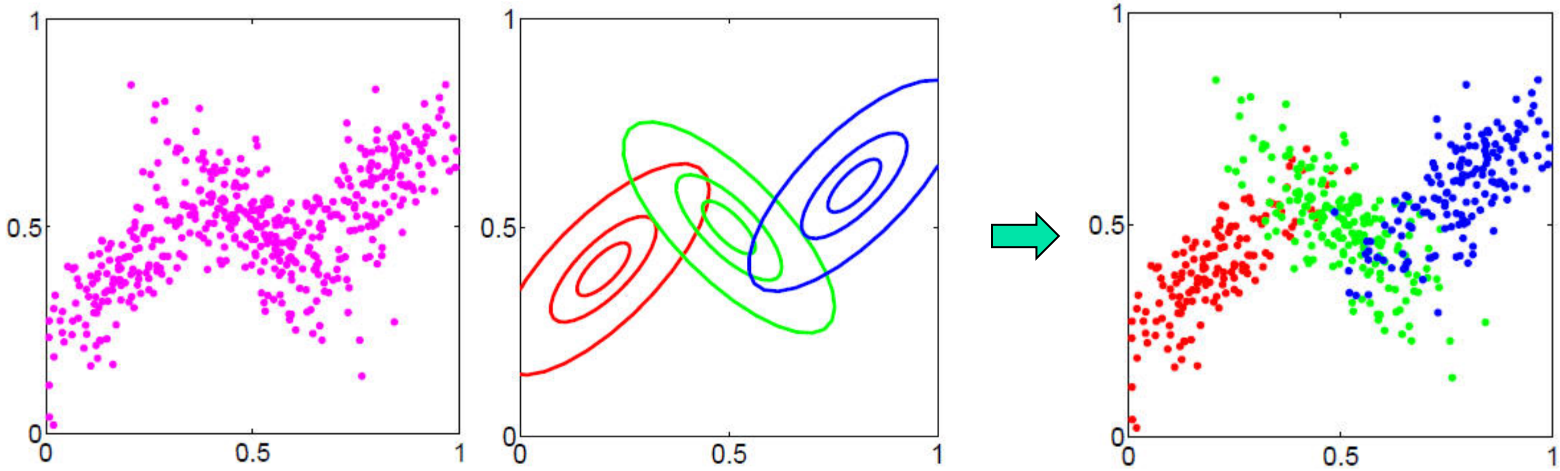
  - EX> Mixture Gaussian





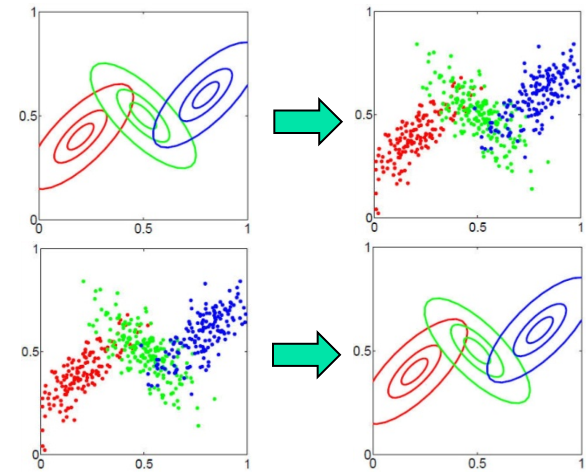
EM Algorithm

- Clustering = finding the unknown distribution
  - If we assume mixture Gaussian, find
    - w_i : $P(C_i)$
    - μ_i : mean of C_i
    - σ_i : variance of C_i



EM Algorithm

- Finding C_i
  - If we know C_i (w_i, μ_i, σ_i)
    → We can find which C_i generate x
  - If we know which C_i generate x
    → We can find C_i (w_i, μ_i, σ_i)
- Basic idea
  - Assume C_i
  - Repeat:
    1. Assign cluster (according to $p(C_i \mid x)$) → Expectation
    2. Compute new C_i (w_i, μ_i, σ_i) → Maximization



EM Algorithm

- Algorithm (Mixture Gaussian)
  - Initialize C_i (w_i, μ_i, σ_i) arbitrarily
  - Repeat:
    - E-step

$$p_{ij} = P(C_i \mid x_j) = \alpha\, P(x_j \mid C_i) P(C_i)$$

    - M-step

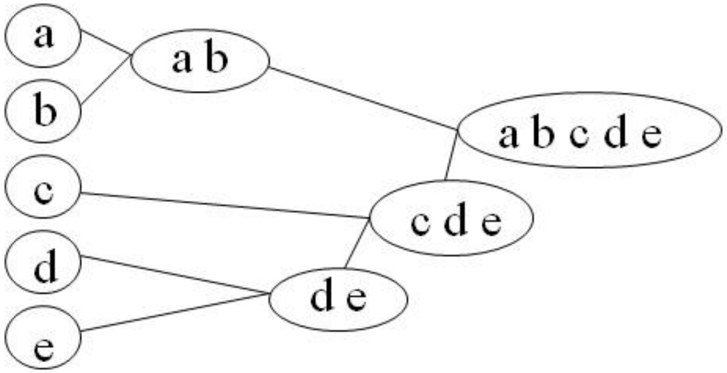
$$w_i = p_i = \sum_j p_{ij}$$

$$\mu_i = \sum_j \frac{p_{ij}}{p_i} x_j$$

$$\sigma_i = \sum_j \frac{p_{ij}}{p_i} (x_j - \mu_i)^2$$

Hierarchical Clustering

- Find a hierarchical cluster structure



- Conceptual clustering
  - Input: a set of unclassified objects
  - Output: a hierarchy of classes (clusters)
  - Goal: maximize the similarity of objects in the same class

Hierarchical Clustering

- COBWEB
  - Use Category utility
    - $C = \{C_1 .. C_p\}$ categories
    - $A = \{A_1 .. A_n\}$ attributes, each A_i has $\{V_{i1} .. V_{im}\}$ values

$$CU(C) = \frac{1}{p}\sum_k P(C_k)\left[\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2\right]$$

P(A=V | C) : prob. of correctly guessing the value
when we know the category labels

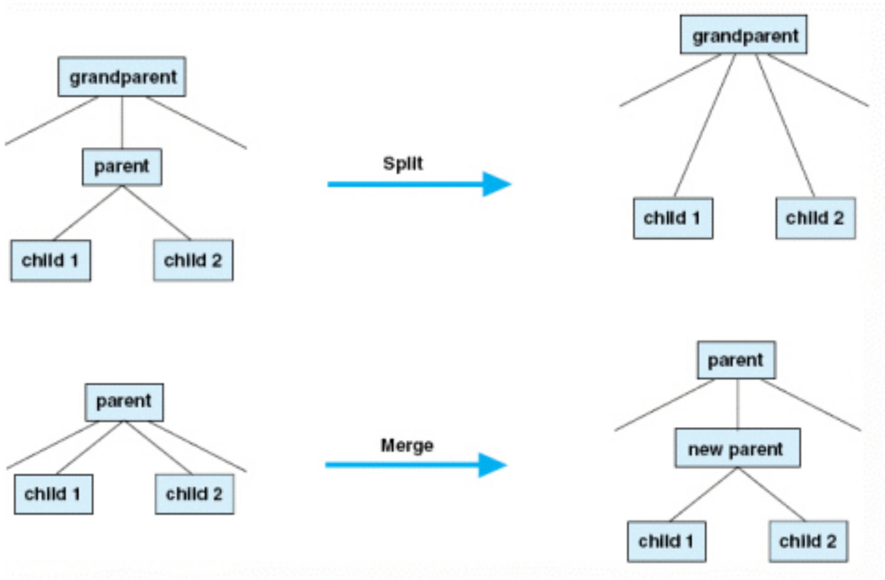
P(A=V) : prob. of correctly guessing the value
without the knowledge of category structure

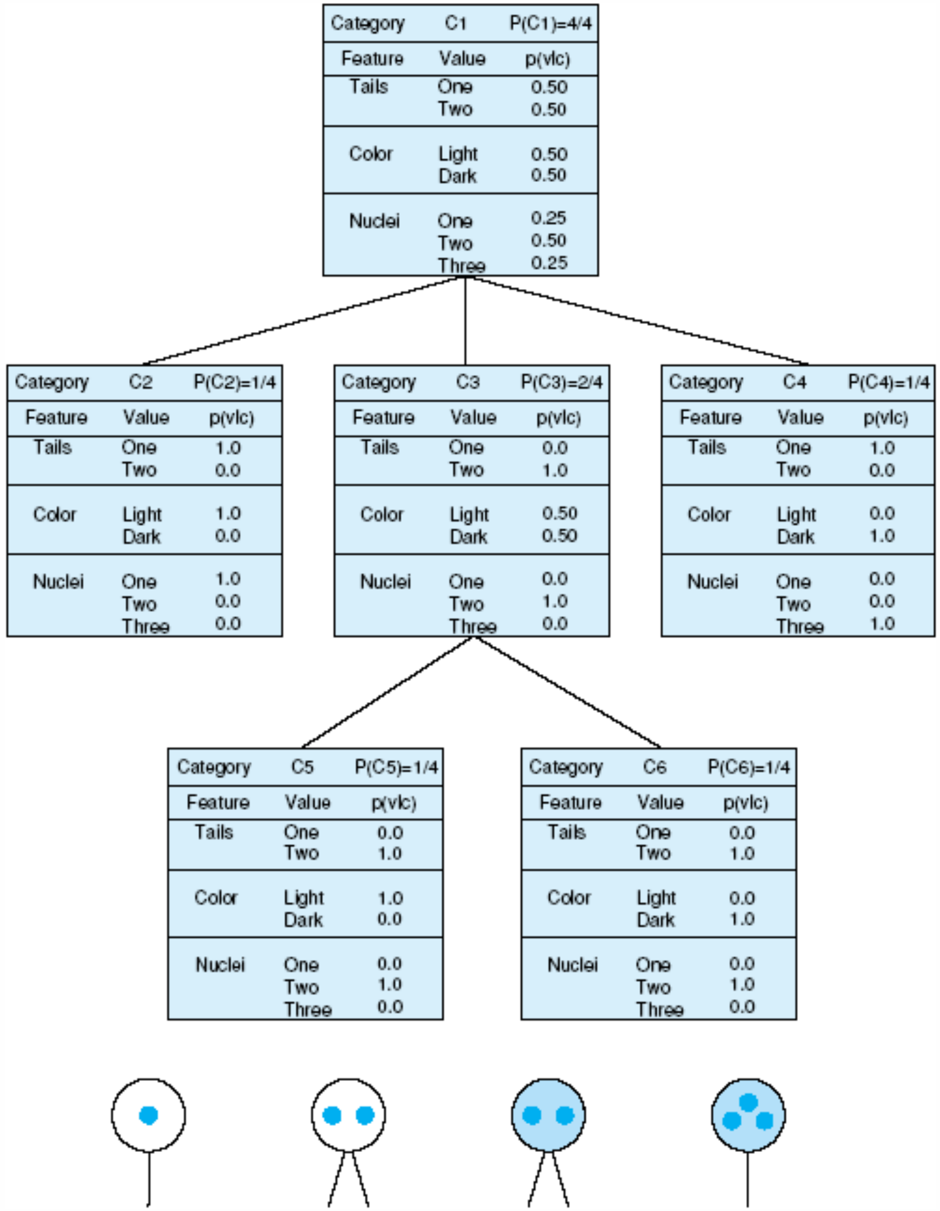
Hierarchical Clustering

COBWEB(C, i)

Compute CU of each case, and choose best

1. Put i in one subclass C1 → COBWEB(C1, i)
2. Split a subclass C1 → COBWEB(C, i)
3. Merge two subclass to Cm → COBWEB(Cm, i)
4. Make a new subclass Cn




Category C1 P(C1)=4/4
Feature Value p(v|c)
Tails One 0.50
Two 0.50
Color Light 0.50
Dark 0.50
Nuclei One 0.25
Two 0.50
Three 0.25
Category C2 P(C2)=1/4
Feature Value p(v|c)
Tails One 1.0
Two 0.0
Color Light 1.0
Dark 0.0
Nuclei One 1.0
Two 0.0
Three 0.0
Category C3 P(C3)=2/4
Feature Value p(v|c)
Tails One 0.0
Two 1.0
Color Light 0.50
Dark 0.50
Nuclei One 0.0
Two 1.0
Three 0.0
Category C4 P(C4)=1/4
Feature Value p(v|c)
Tails One 1.0
Two 0.0
Color Light 0.0
Dark 1.0
Nuclei One 0.0
Two 0.0
Three 1.0
Category C5 P(C5)=1/4
Feature Value p(v|c)
Tails One 0.0
Two 1.0
Color Light 1.0
Dark 0.0
Nuclei One 0.0
Two 1.0
Three 0.0
Category C6 P(C6)=1/4
Feature Value p(v|c)
Tails One 0.0
Two 1.0
Color Light 0.0
Dark 1.0
Nuclei One 0.0
Two 1.0
Three 0.0

Density-Based Methods

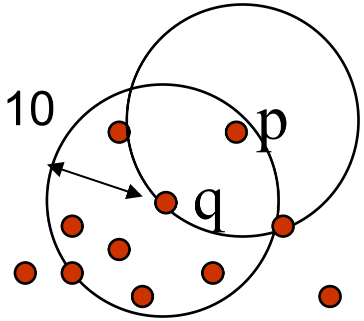
- Group objects in dense region
  - Density parameters

    Radius ε : distance to determine the neighborhood

    MinPts : Minimum number of points in neighborhood
- Core object
  - ε -neighborhood contains ***MinPts*** objects
- Directly density-reachable
  - ***p*** is directly density-reachable from ***q***

    if ***q*** is a core object, and

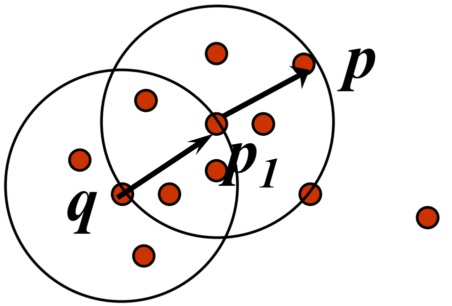
    p is ε -neighborhood of ***q***



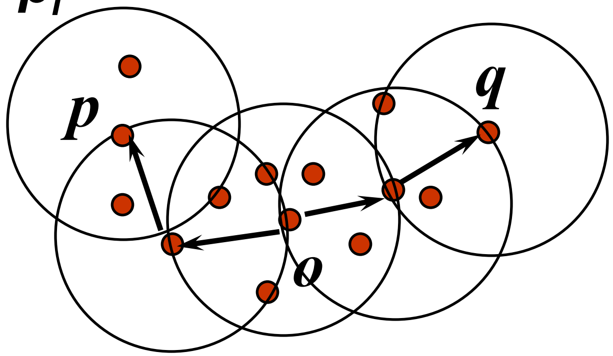
ε = 10
MinPts = 5

Density-Based Methods

- Density-reachable
  - ***p*** is density-reachable from ***q*** if
    there are objects $p_1(=q), p_2, \ldots p_n$
    such that
    p_{i+1} is directly density-reachable from p_i

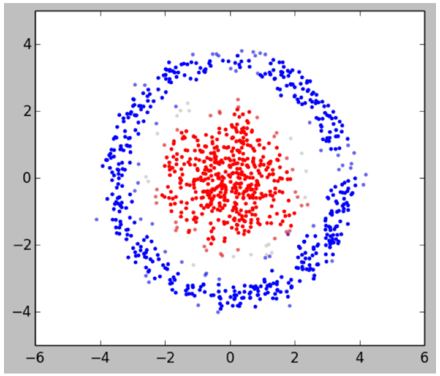


- Density-connected
  - ***p*** is density-connected to ***q*** if
    there is an object ***o***
    such that
    p and ***q*** are density-reachable from ***o***

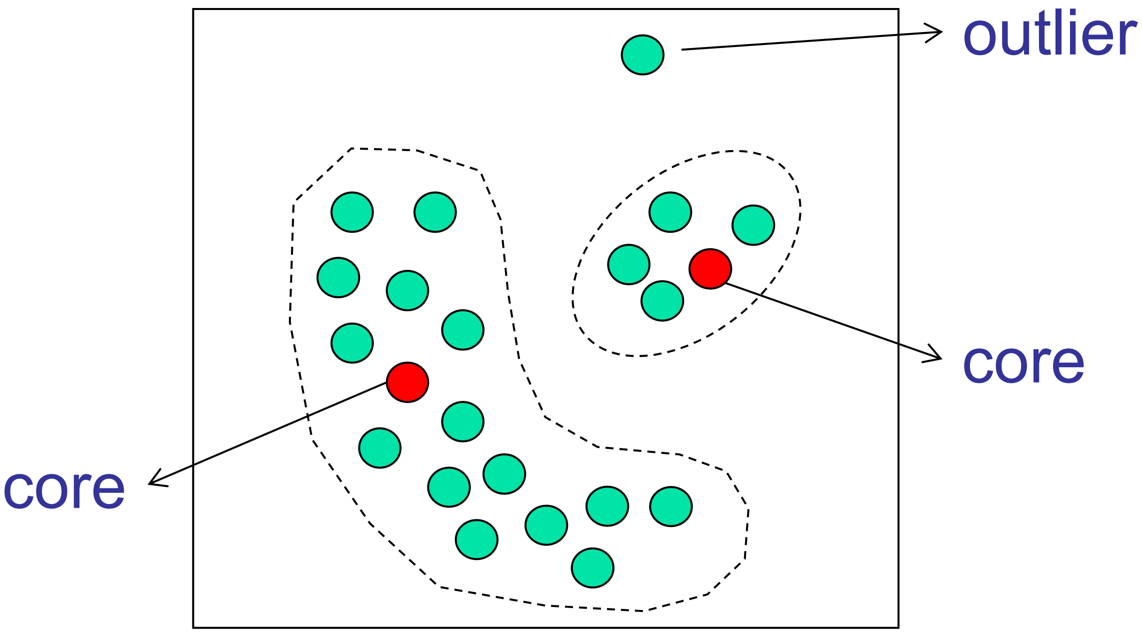


Density-Based Methods

- DBSCAN
  - Cluster - a maximal set of density-connected points
    1. Arbitrary select a point ***p*** and retrieve all ε-neighborhood
    2. If ***p*** is a core object, a cluster is formed
    3. From each core object ***p***, iteratively collects directly density-reachable objects (may merge clusters)
    4. Continue the process until no new points can be added
- Major features
  - Discover clusters of arbitrary shape
  - Handle noise
  - Problem: selecting parameters ε and ***MinPts***



Density-Based Methods



MinPts = 5