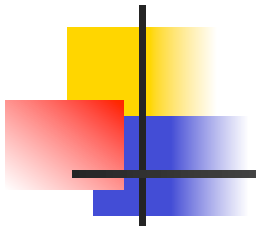
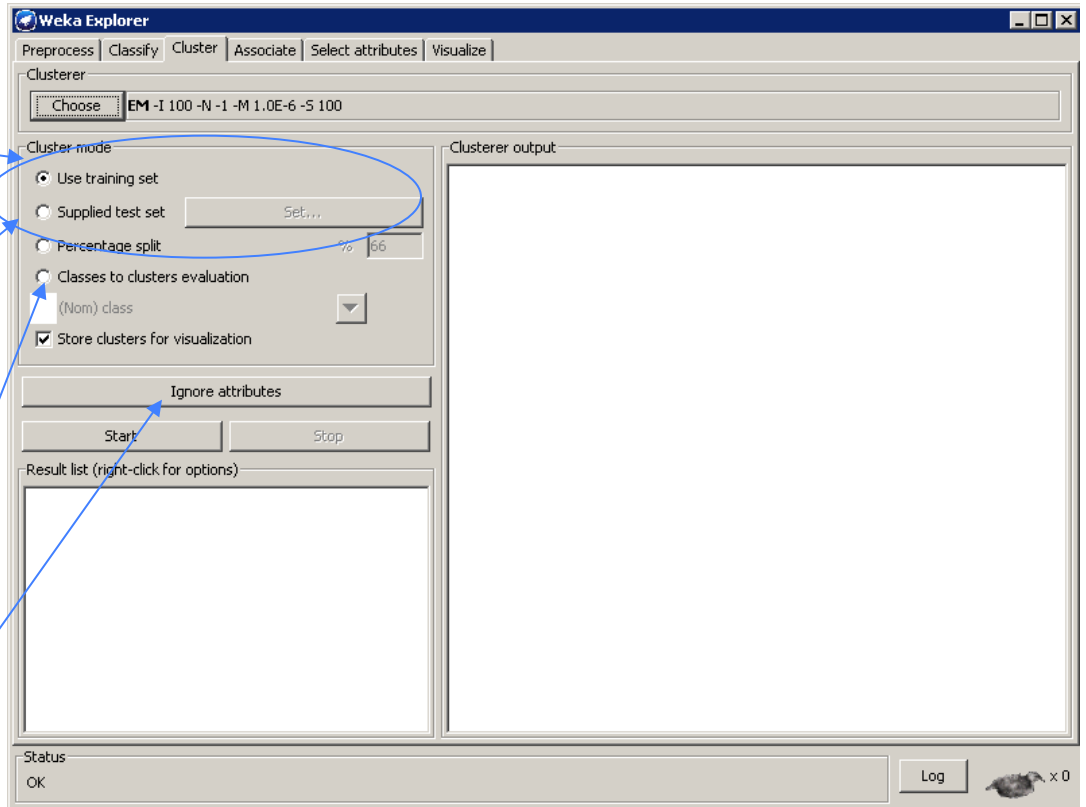


Clustering with WEKA

Prof. Pietro Ducange

Clustering

The Cluster mode box is used to choose what to cluster and how to evaluate the results.

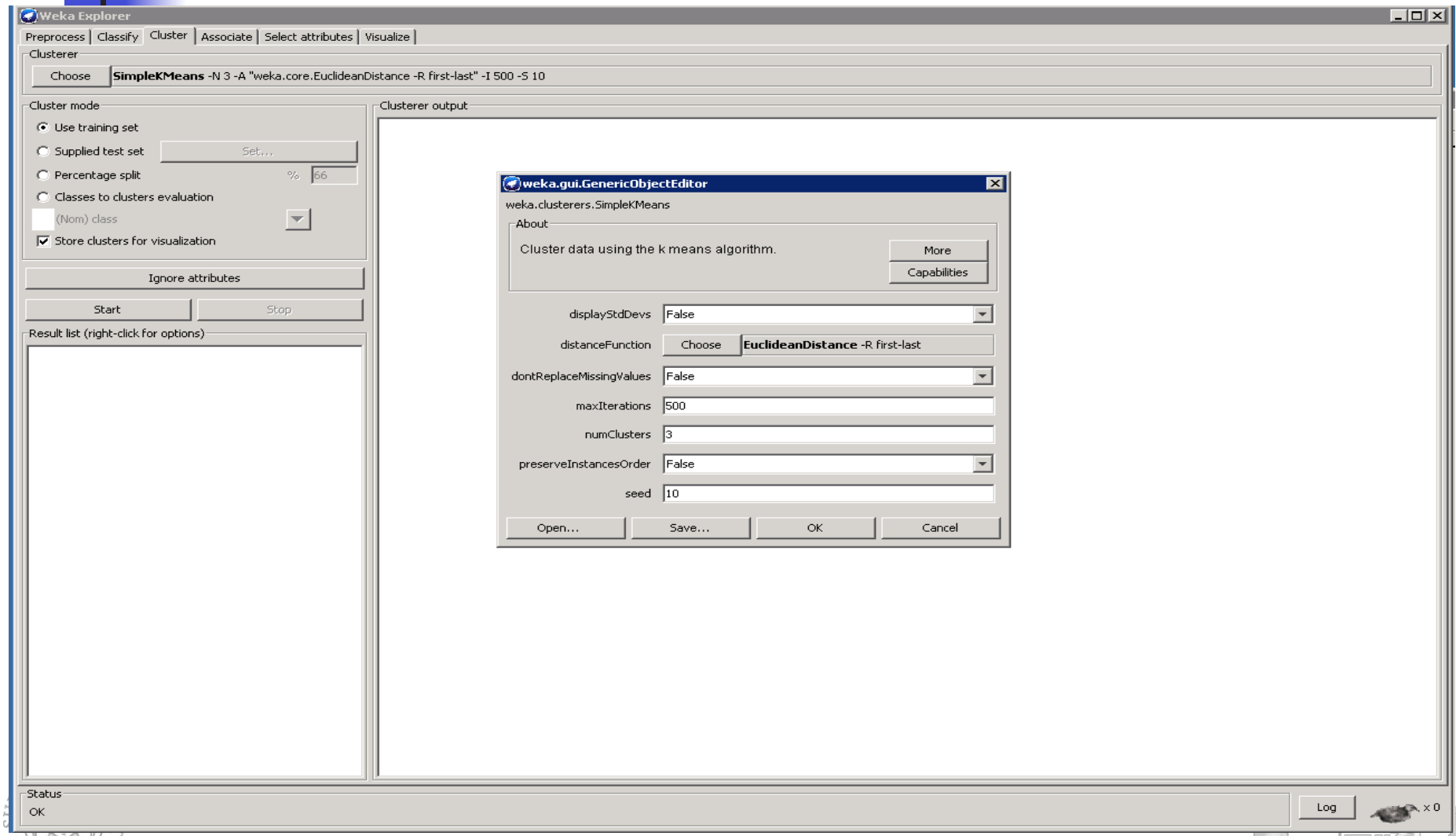


First three options: Use training set, Supplied test set and Percentage split. Now the data is assigned to clusters instead of trying to predict a specific class.

The fourth mode compares how well the chosen clusters match up with a pre-assigned class in the data. The drop-down box below this option selects the class, just as in the Classify panel.

Often, some attributes in the data should be ignored when clustering. The Ignore attributes button brings up a small window that allows you to select which attributes are ignored

K-means


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
Clusterer output
weka.gui.GenericObjectEditor
weka.clusterers.SimpleKMeans
About
Cluster data using the k means algorithm.
More
Capabilities
displayStdDevs
False
distanceFunction
Choose
EuclideanDistance -R first-last
dontReplaceMissingValues
False
maxIterations
500
numClusters
3
preserveInstancesOrder
False
seed
10
Open...
Save...
OK
Cancel
Status
OK
Log
x 0

Clustering: K-means example (0): training set

```
=== Model and evaluation on training set ===
Cluster centroids:
                         Cluster#
Attribute      Full Data         0         1         2
                 (150)      (61)      (50)      (39)
==========================================================
sepallength       5.8433    5.8885     5.006    6.8462
sepalwidth         3.054    2.7377     3.418    3.0821
petallength       3.7587    4.3967     1.464    5.7026
petalwidth        1.1987     1.418     0.244    2.0795
Time taken to build model (full training data) : 0 seconds

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

Clustered Instances

0       61 ( 41%)
1       50 ( 33%)
2       39 ( 26%)
```

Clustering: K-means example (1) Percentage split (66%)

=== Model and evaluation on test split ===

kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 4.7314953964193025
Missing values globally replaced with mean/mode

Cluster centroids:

		Cluster#		
Attribute	Full Data	0	1	2
	(99)	(35)	(36)	(28)
sepallength	5.8313	5.0514	6.725	5.6571
sepalwidth	3.0586	3.4543	3.0139	2.6214
petallength	3.6848	1.4771	5.4389	4.1893
petalwidth	1.1657	0.2571	1.9139	1.339

Time taken to build model (percentage split) : 0 seconds

Clustered Instances

0 15 (29%)
1 20 (39%)
2 16 (31%)

Clustering: K-means example (2)
Classes to clusters evaluation

Cluster centroids:

Attribute	Full Data (150)	Cluster# 0 (61)	1 (50)	2 (39)
sepallength	5.8433	5.8885	5.006	6.8462
sepalwidth	3.054	2.7377	3.418	3.0821
petallength	3.7587	4.3967	1.464	5.7026
petalwidth	1.1987	1.418	0.244	2.0795

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

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

Clustered Instances

0 61 (41%)
1 50 (33%)
2 39 (26%)

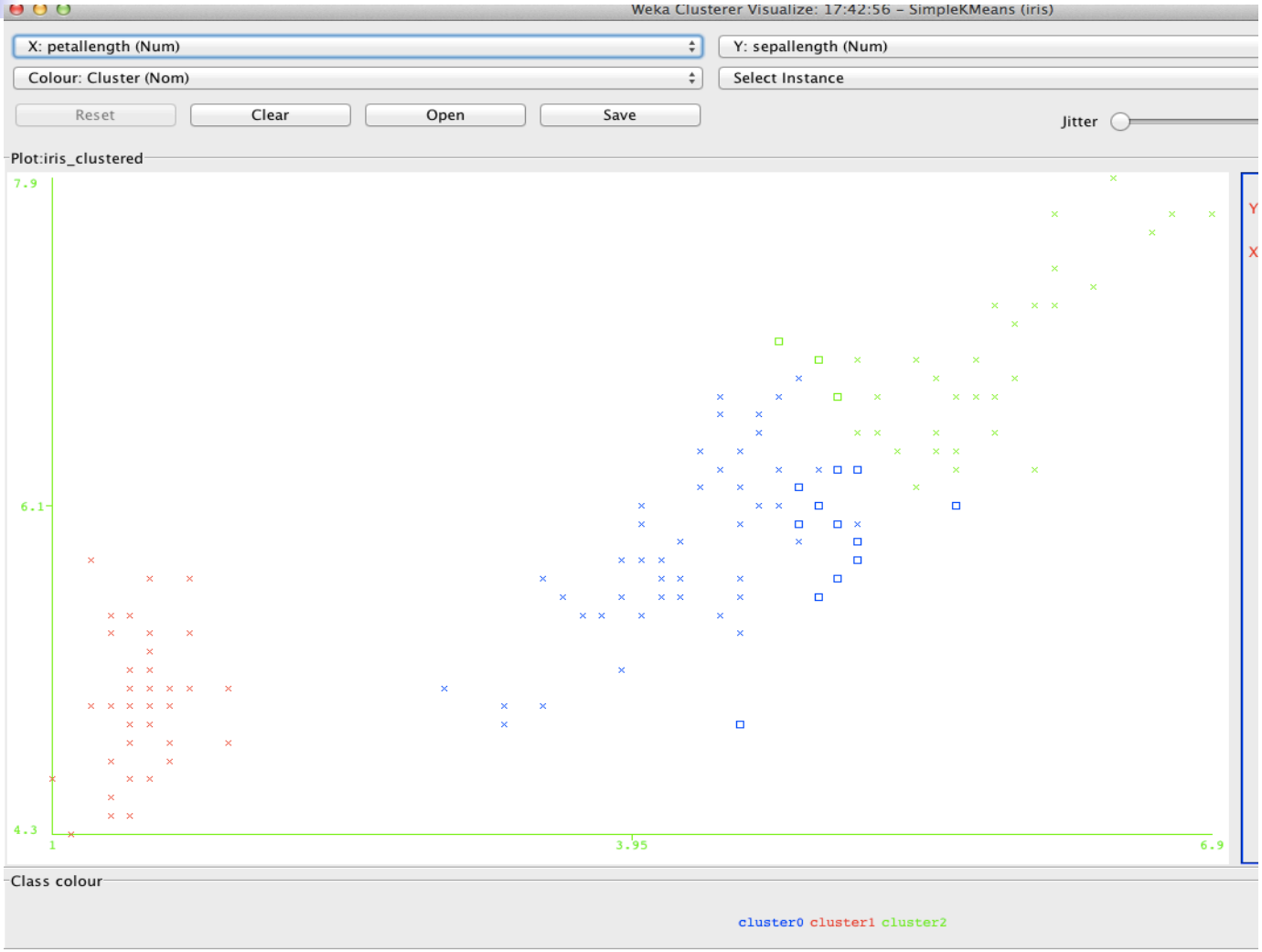
Class attribute: class
Classes to Clusters:

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

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

Incorrectly clustered instances : 17.0
11.3333 %

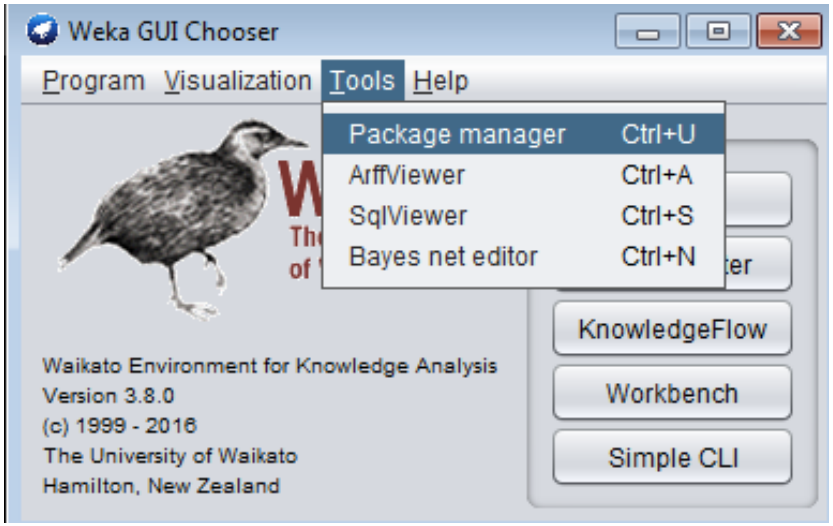
Cluster Visualization



DB Scan clustering algorithm

In the most recent version of WEKA the DB Scan algorithm is not available in the basic version of software

The algorithm must be added from the "Package manager" menu (please search for optics_dbScan algorithm and install it)



Clustering:DB scan example (1) epsilon=0.9, minPts=6

Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

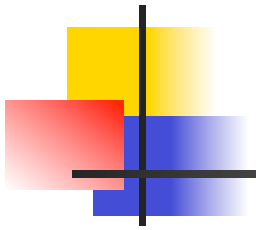
0 150 (100%)

Class attribute: class
Classes to Clusters:

0 <-- assigned to cluster
50 | Iris-setosa
50 | Iris-versicolor
50 | Iris-virginica

Cluster 0 <-- Iris-setosa

Incorrectly clustered instances : 100.0 66.6667 %



Exercise

- Use the BD scan algorithm with the following parameters:

  Minpoints = 5

  Epsilon = 0.2

- Discuss the generated results using the Classes to clusters evaluation mode

Clustering:DB scan example (1) epsilon=0.2, minpoints=5

Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

0 49 (33%)
1 98 (67%)

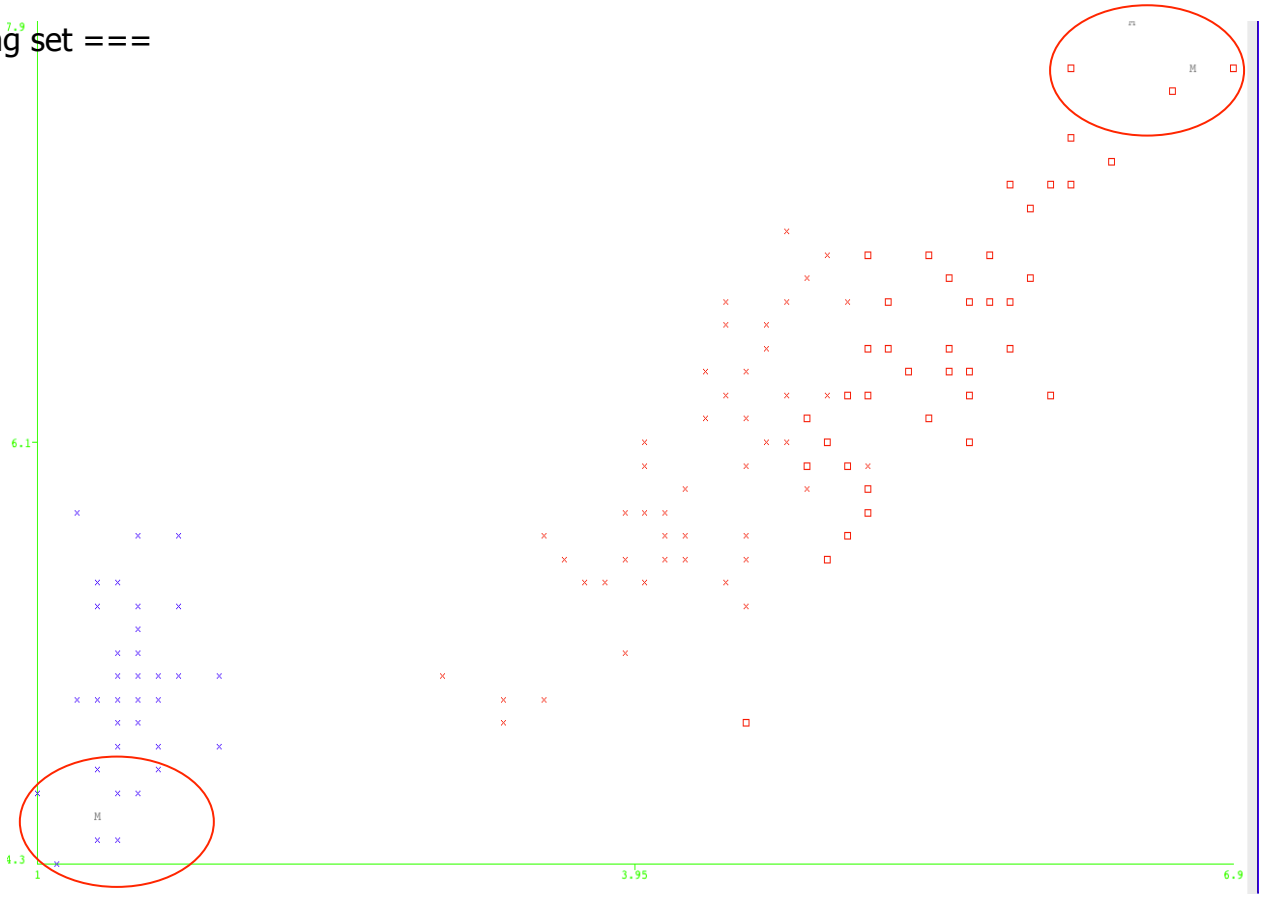
Unclustered instances : 3
(one setosa and two virginicas)

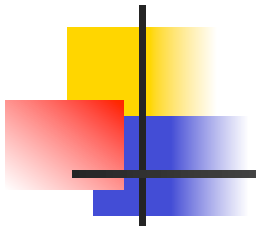
Class attribute: class
Classes to Clusters:

0	1	<-- assigned to cluster
49	0	Iris-setosa
0	50	Iris-versicolor
0	48	Iris-virginica

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

Incorrectly clustered instances :
48.0 32 %





Exercise

- Use the BD scan algorithm with the following parameters (classes to clusters evaluation mode):

  Minpoints = 2

  Epsilon = 0.2

- Discuss the generated results

Clustering:DB scan example (1) epsilon=0.2, minpoints=2

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

Clustered Instances

0 49 (33%)
1 98 (66%)
2 2 (1%)

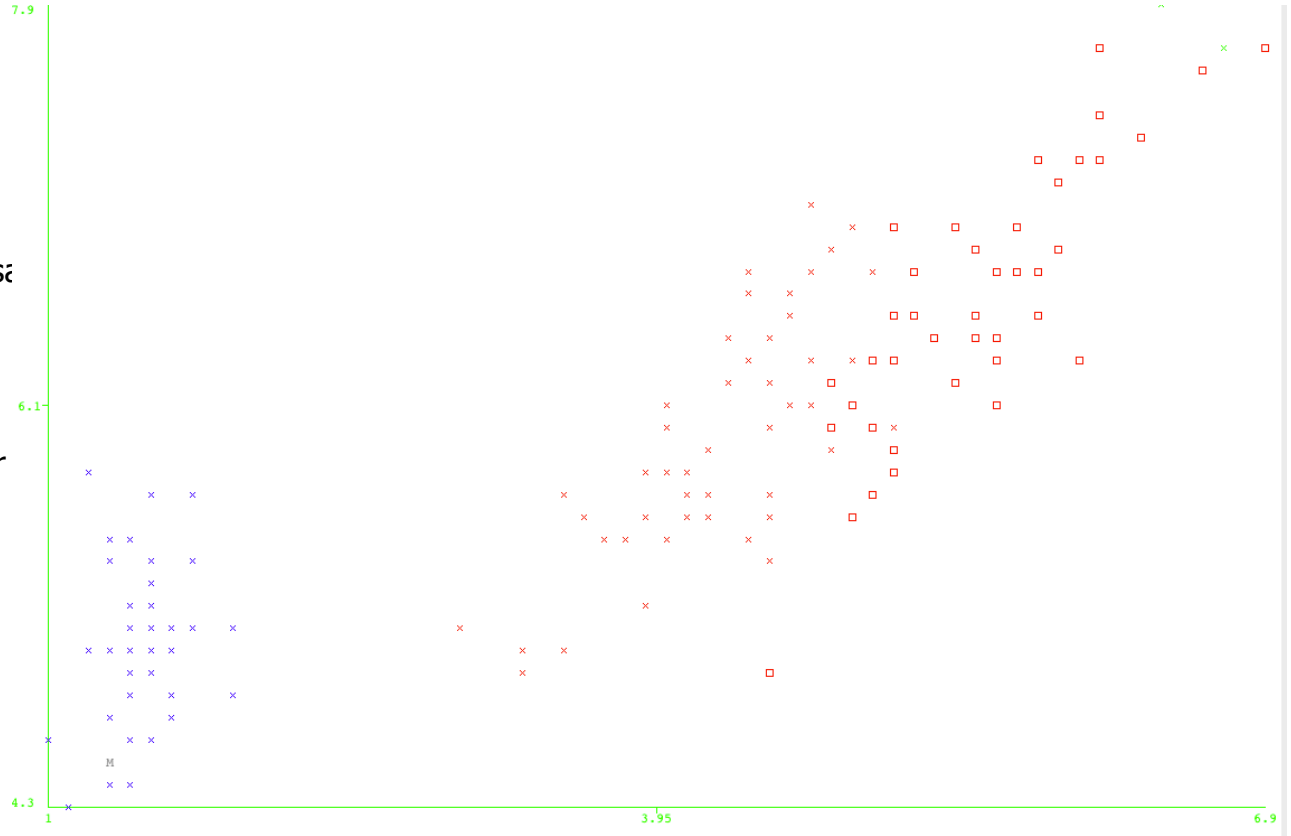
Unclustered instances : 1 (setosa

Class attribute: class
Classes to Clusters:

0	1	2	<-- assigned to cluster
49	0	0	Iris-setosa
0	50	0	Iris-versicolor
0	48	2	Iris-virginica

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

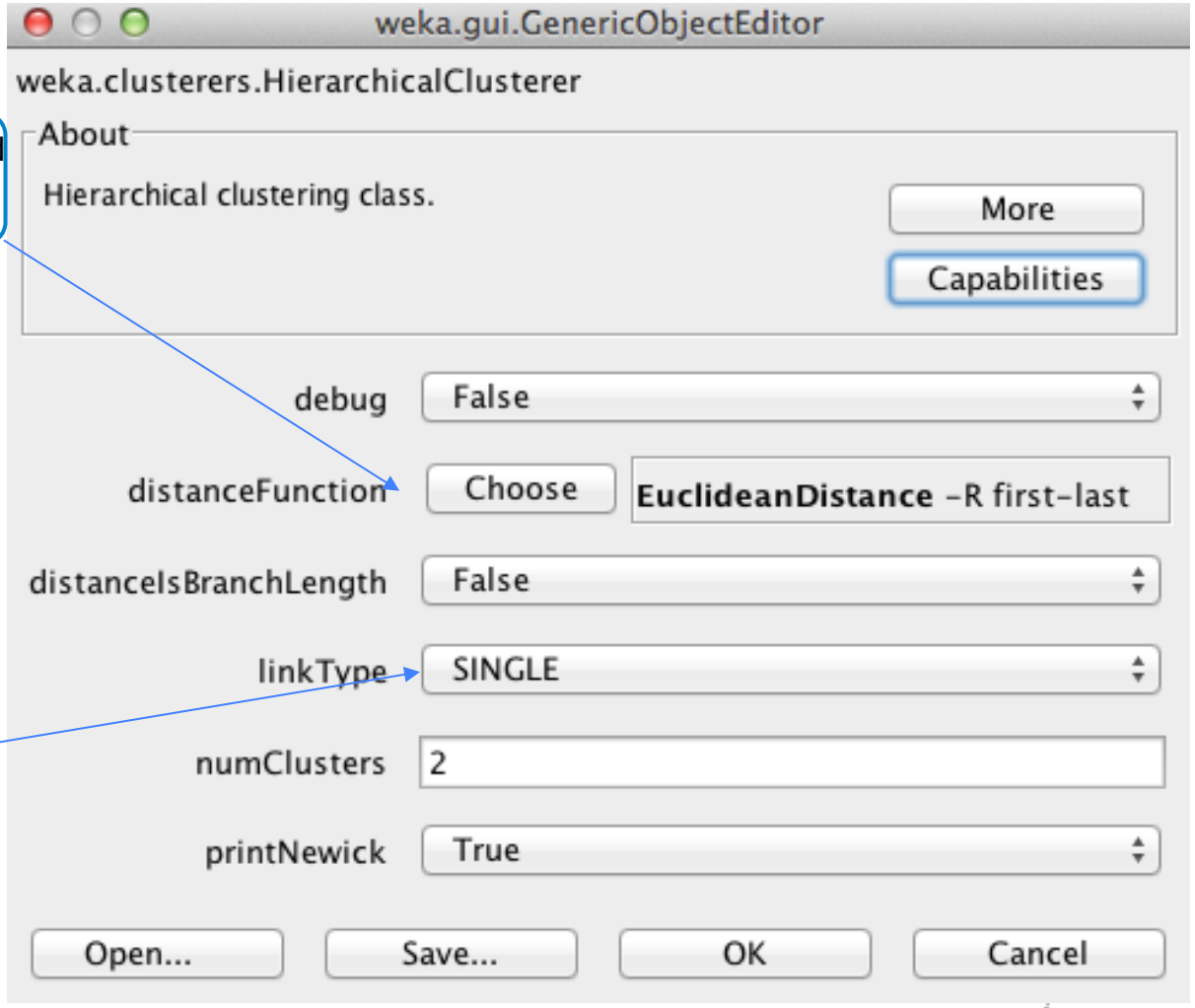
Incorrectly clustered instances :
48.0 32 %



Hierarchical Agglomerative Clustering

Different Distance Functions can be used and also re-defined

Find single link distance i.e. the minimum link, which is the closest distance between any item in cluster_i and any item in cluster_j



Hierarchical Agglomerative Clustering: an Example

We consider a dataset with 1023 instances and 32 attributes. 16 clusters must be identified.

Clustered Instances Using the Hierarchical Clustering

0	63 (6%)
1	64 (6%)
2	64 (6%)
3	64 (6%)
4	64 (6%)
5	64 (6%)
6	64 (6%)
7	64 (6%)
8	64 (6%)
9	64 (6%)
10	64 (6%)
11	64 (6%)
12	64 (6%)
13	64 (6%)
14	64 (6%)
15	64 (6%)

Clustered Instances Using the simpleKmeans

0	1 (0%)
1	63 (6%)
2	192 (19%)
3	64 (6%)
4	1 (0%)
5	1 (0%)
6	61 (6%)
7	64 (6%)
8	128 (13%)
9	13 (1%)
10	64 (6%)
11	64 (6%)
12	51 (5%)
13	64 (6%)
14	64 (6%)
15	128 (13%)