大数据基础与应用

BIG DATA FOUNDATION AND APPLICATION

数据分析算法

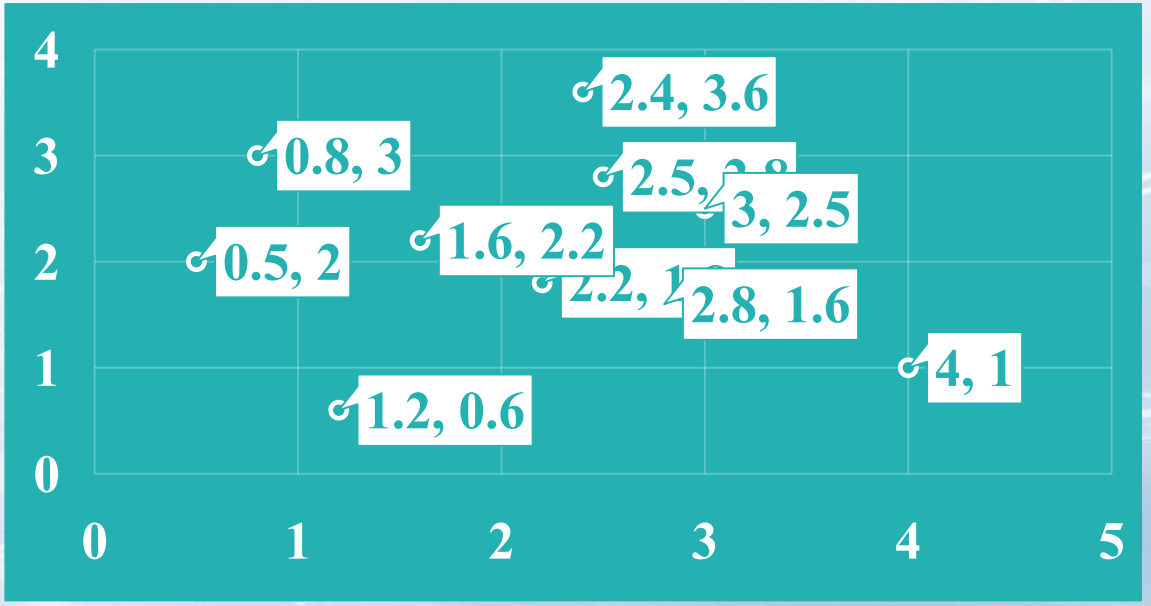
北京理工大学计算机学院　孙新

2019年1月

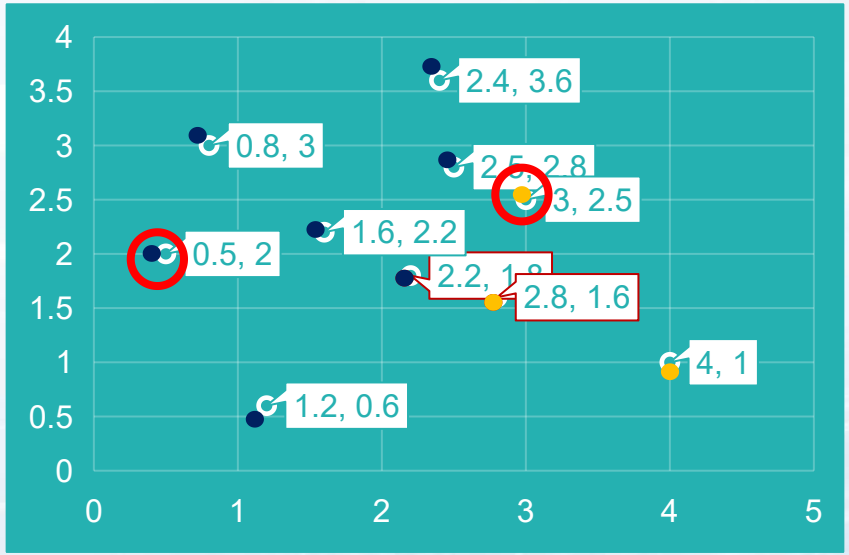
K-means聚类算法实例求解过程

U	x	y
1	0.5	2
2	0.8	3
3	1.2	0.6
4	1.6	2.2
5	2.2	1.8
6	2.4	3.6
7	2.5	2.8
8	2.8	1.6
9	3	2.5
10	4	1

给出数据对象集合如左图所示
簇的数量k=2
样本点分布如下图



K-means聚类算法实例求解过程



(1)任意选取两个点作为两个簇的

如C1=（2.2，1.8），C2=（2.8,

(2)对剩余的每个对象，根据其与

它赋给最近的簇对

例如，对点(0.5,2)

$$d1 = \sqrt{(0.5-2.2)^2+(}$$

$$d2 = \sqrt{(0.5-2.8)^2+(2-}$$

显然 d1<d2，故将点(0.5,2)分配

对点(3,2.5)

$$d1 = \sqrt{(3-2.2)^2+(2.}$$

$$d2 = \sqrt{(3-2.8)^2+(2.}$$

K-means聚类算法实例求解过程

(3) 计算新的簇中心

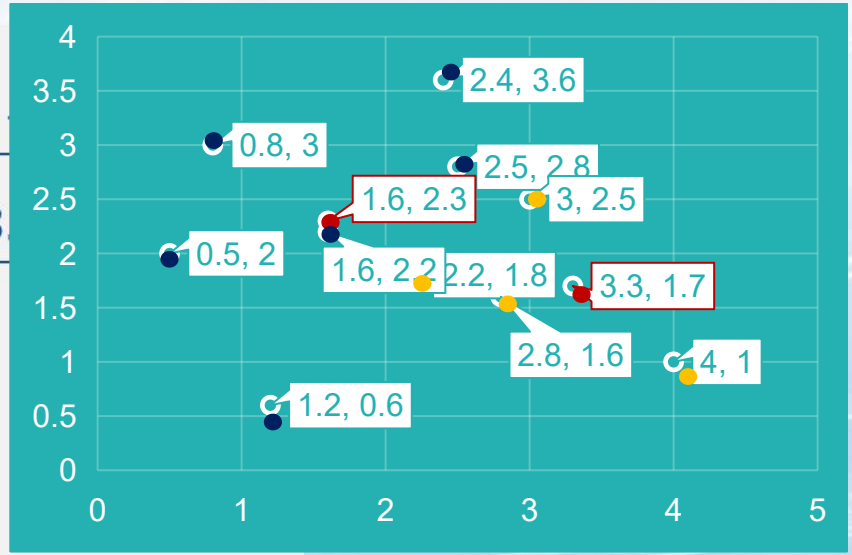
$$c1_x = \frac{0.5 + 0.8 + 1.2 + 1.6 + 2.2 + 2.4}{7}$$

$$c1_y = \frac{2 + 3 + 0.5 + 2.2 + 1.3 + 3.6 + 3}{7}$$

$$c2_x = \frac{2.8 + 3 + 4}{3} = 3.3$$

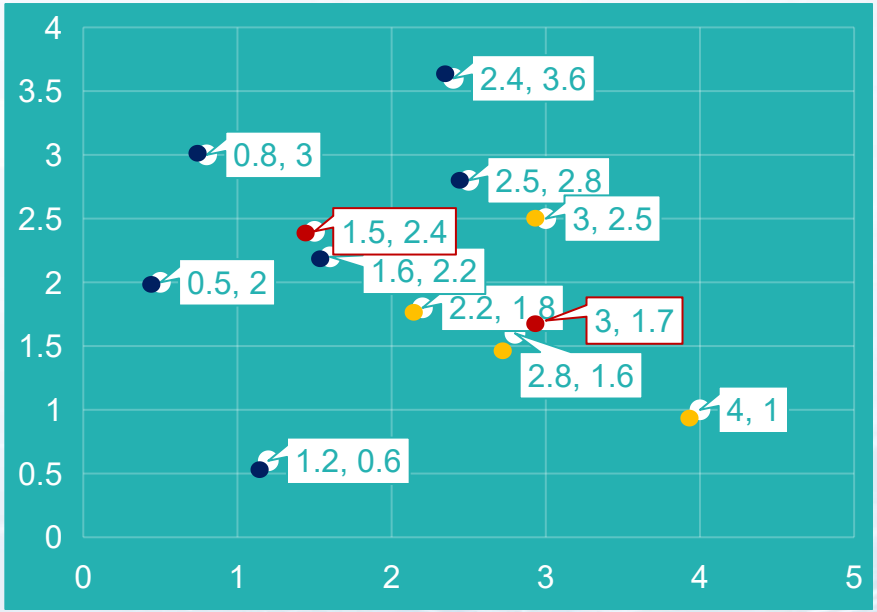
$$c2_y = \frac{1.5 + 2.5 + 1}{3} = 1.7$$

故，C1=(1.6,2.3)，C2=(3.3,1.7)



(4) 重新计算数据集中每个点到两个簇中心的距离，根据其值进行重新分配

K-means聚类算法实例求解过程



(5) 计算新的簇中心

C1=(1.5,2.4)，C2=(3,1.7)

(6) 再次分配结果为：

簇A： (0.5,2),(0.8,3),(1.6,2.2),(1.2,0.6),(2.4,3.6),(2.5,2.8)

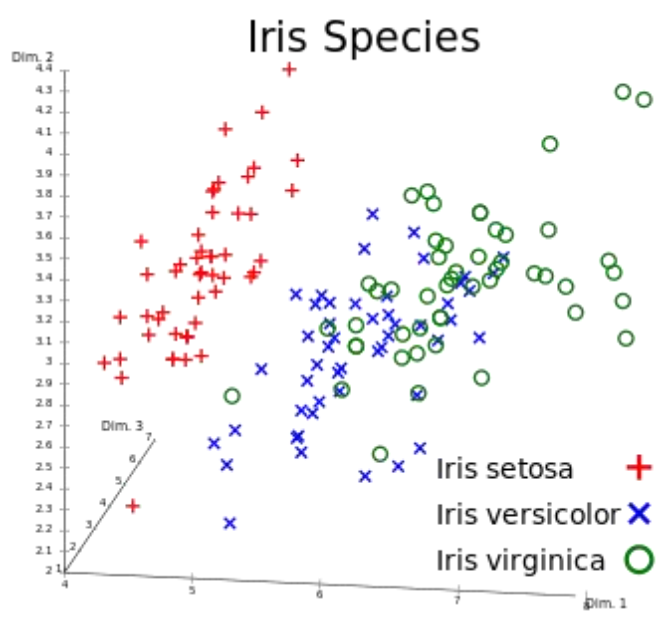
簇B： (2.2,1.8),(3,2.5),(2.8,1.6),(4,1)

在三次迭代之后，此时簇中心不变，所以停止迭代过程，算法停止

Iris 数据集的聚类结果

Iris数据集

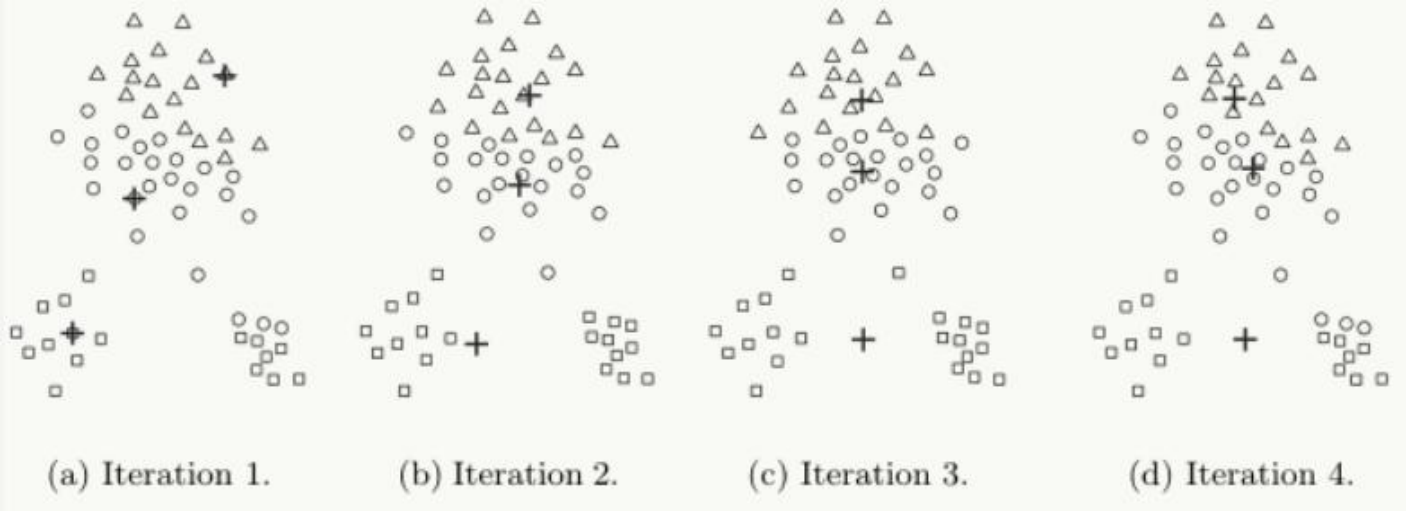
sepal length in cm （花萼长度）	Sepal width in cm （花萼宽度）	petal length in cm （花瓣长度）	petal width in cm（花瓣宽度）	l wid m（
5.1	3.5	1.4	0.2	
4.9	3	1.4	0.2	
6	2.2	4	1	
6.9	3.2	5.7	2.3	
……				



4.4 K-均值聚类算法（K-means算法）

- k-means存在缺点

k-means是局部最优的，容易受到初始质心的影响；比如在下图中，因选择初始质心不恰当而造成次优的聚类结果（SSE较大）：



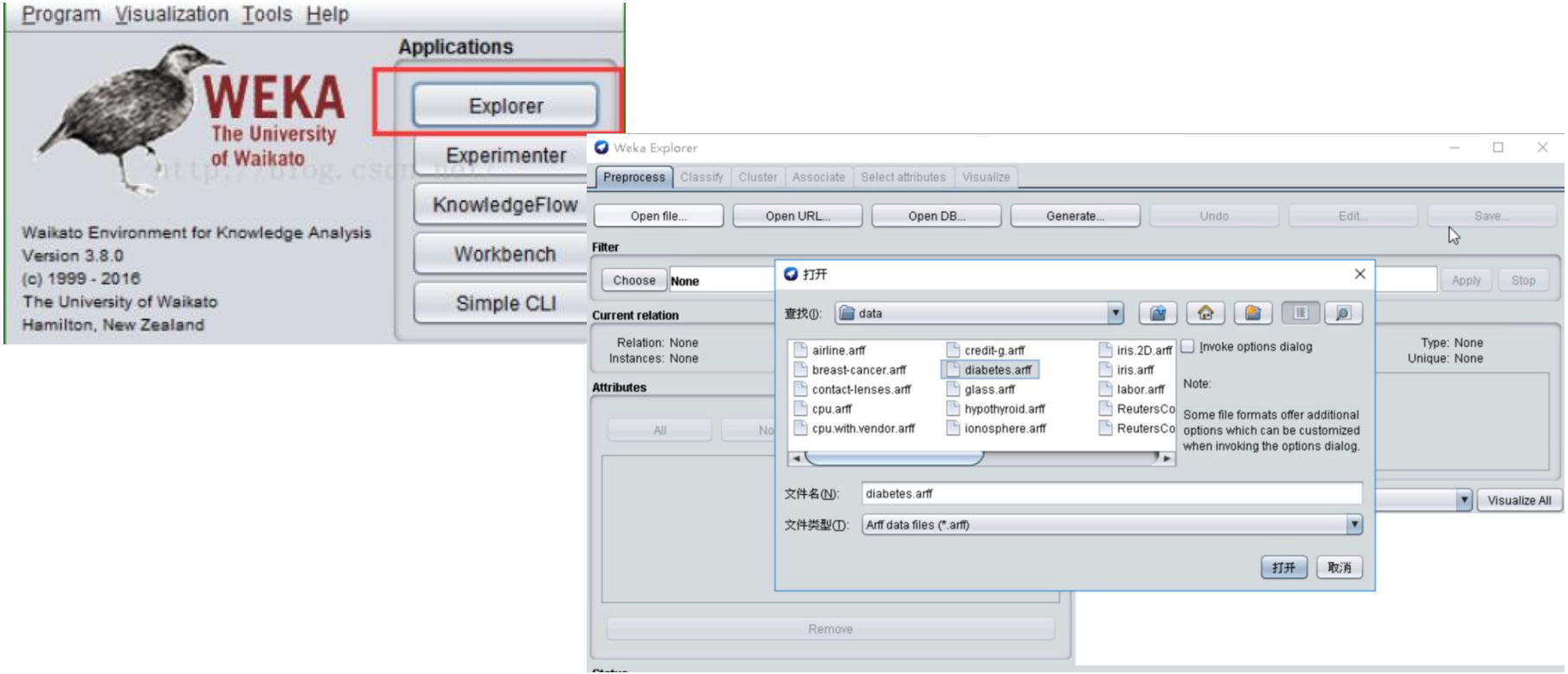
使用Weka实现k-means聚类的算法示例

- **实验目的：**熟悉Weka平台，学习掌握k-means算法，利用Weka和不同参数设置进行聚类分析，对比结果，得出结论，对问题进行总结

- **实验过程**
  - （1）打开Weka，并导入数据
  - （2） SimpleKMeans算法聚类
  - （3）运行观察结果
    - 观察聚类输出结果
    - 修改参数值重新运行并观察结果
    - 可视化聚类结果

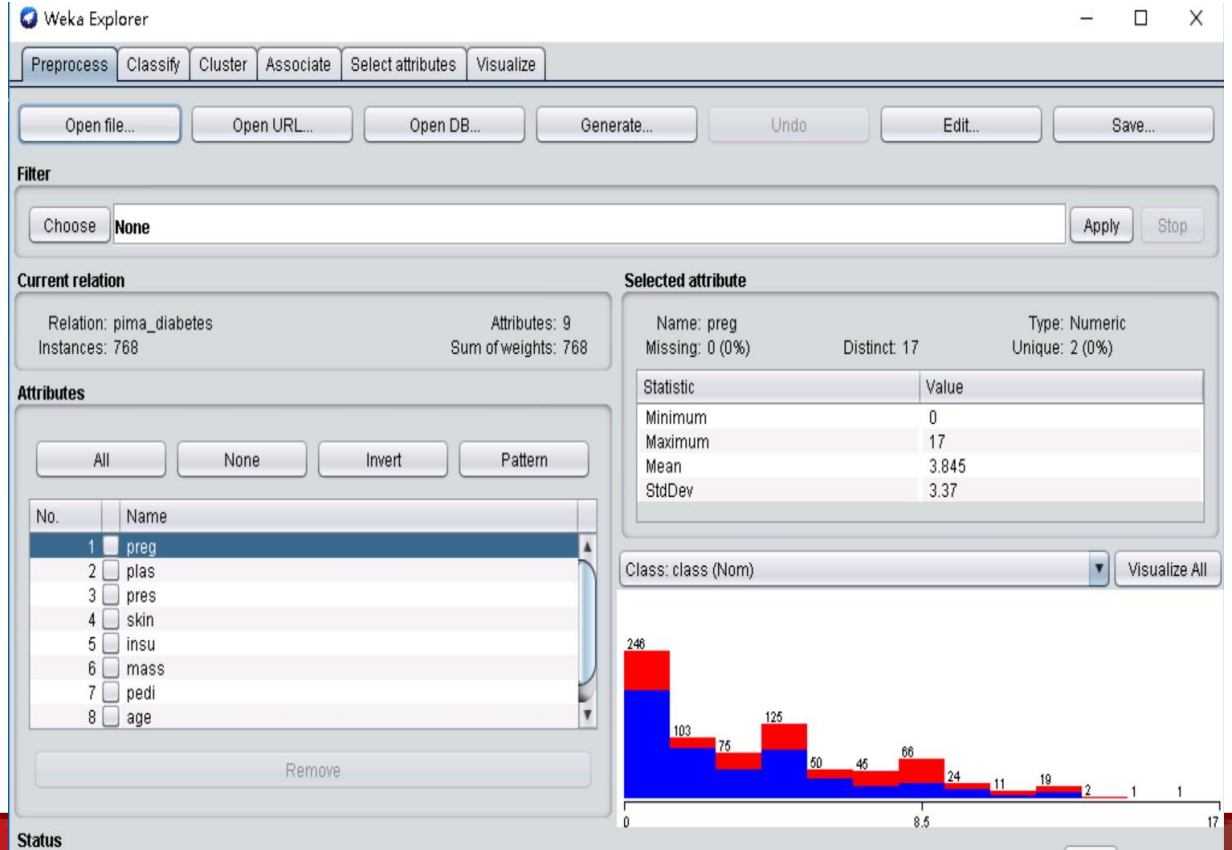
使用Weka实现k-means聚类的算法示例

- 打开Weka自带的数据集“diabetes.arff”（768条实例数据）：



使用Weka实现k-means聚类的算法示例

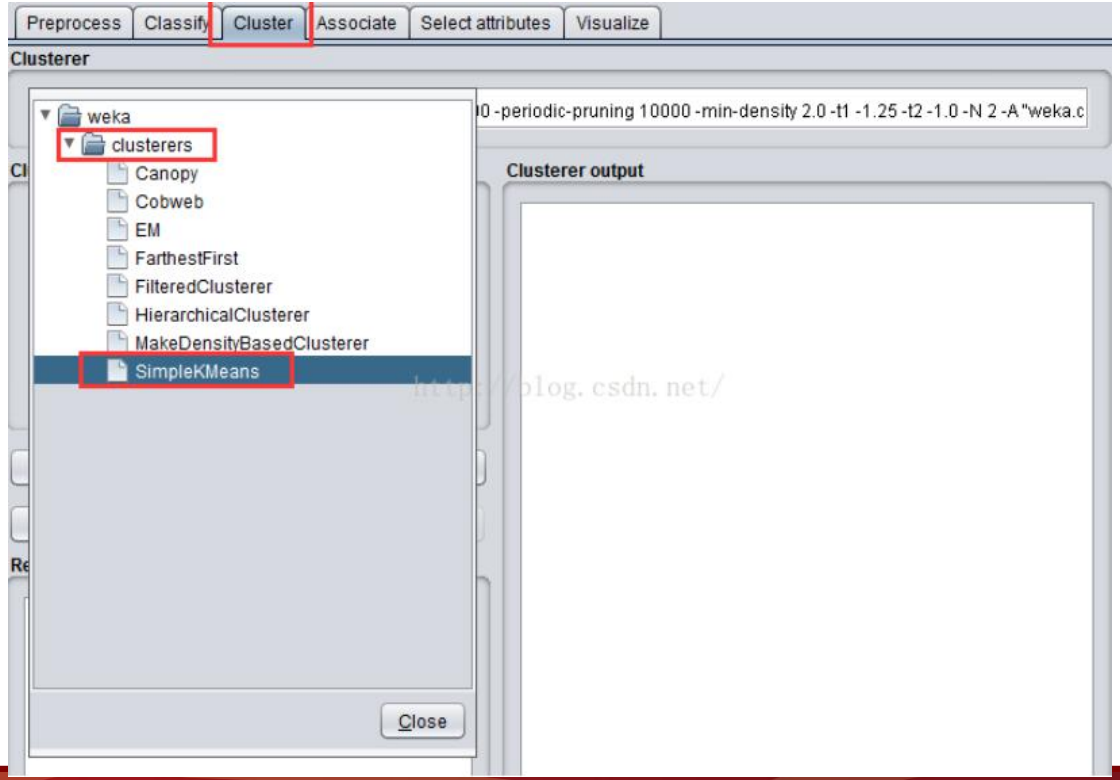
- 打开数据集后，界面出现该数据集的相关描述，



可以观察得到共有9个不同的属性，包括“preg”、“plas”、“skin”、“insu”、“mass”、“pedi”、“age”和“class”，

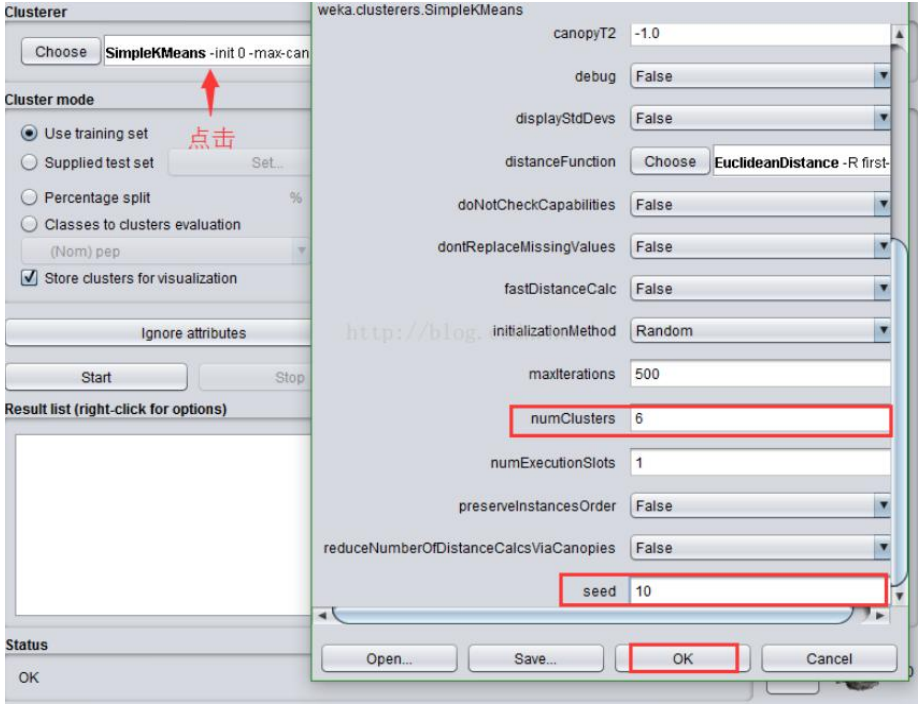
使用Weka实现k-means聚类的算法示例

- 点击“Choose”按钮，切换到“Cluster”，选择目录下的“SimpleKMeans”这是WEKA中实现的K均值聚类的算法

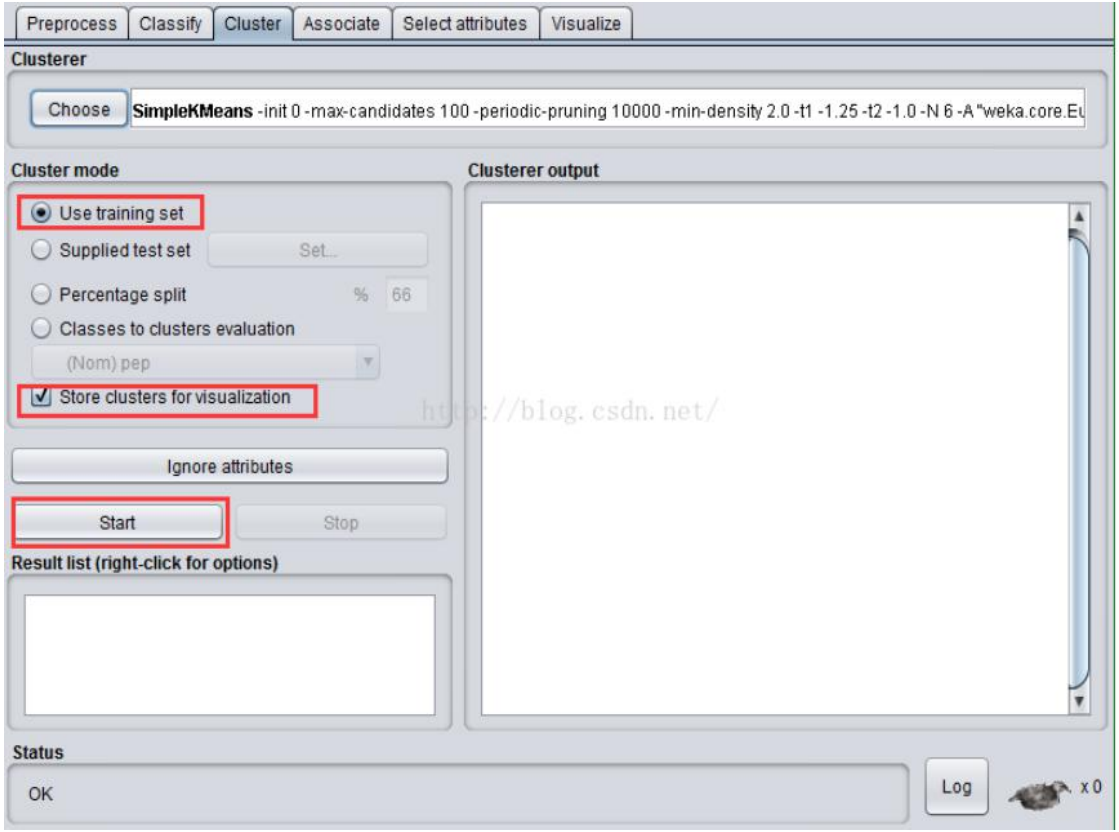


使用Weka实现k-means聚类的算法示例

- 点击“Choose”旁边的文本框，修改“numClusters”为6，说明我们希望把这768条实例聚成6类，即K=6



使用Weka实现k-means聚类的算法示例



使用Weka实现k-means聚类的算法示例

- 观察右边窗口“Clusterer output”给出的聚类结果，
- 结果中的“Cluster centroids”之后列出了各个簇中心的位置。

Clusterer output

```
mass                        31.9926         32.3651         30.6822         26.9807
pedi                         0.4719           0.875          0.4095          0.3647
age                         33.2409         27.6349         48.6262         25.6959
class               tested_negative tested_negative tested_negative tested_negative tested_n




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

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

Clustered Instances

0       63 (  8%)
1      107 ( 14%)
2      171 ( 22%)
3      139 ( 18%)
4      268 ( 35%)
5       20 (  3%)
```

使用Weka实现k-means聚类的算法示例

- 对于数值型的属性，簇中心就是它的均值（Mean）。
- 对于分类型的属性，簇中心就是它的众数（Mode）

```
Final cluster centroids:
                                        Cluster#
Attribute               Full Data              0              1              2              3              4              5
                          (768.0)         (63.0)        (107.0)        (171.0)        (139.0)        (268.0)        (20.0)
==================================================================================================================================
preg                        3.8451         2.0794         7.4766         1.9883         2.2518         4.8657           3.25
plas                      120.8945        127.4603       116.6449       100.0702       110.6259       141.2575           99.5
pres                       69.1055        69.6508        76.5794        68.2632        70.5971        70.8246            1.2
skin                       20.5365        27.8571        15.1682        10.8596        32.7698        22.1642            2.1
insu                       79.7995       202.2063        39.5701        29.5965        88.7554       100.3358           1.25
mass                       31.9926        32.3651        30.6822        26.9807         34.777        35.1425          19.12
pedi                        0.4719          0.875         0.4095         0.3647          0.333         0.5505         0.3632
age                        33.2409        27.6349        48.6262        25.6959        26.3669        37.0672           29.6
class              tested_negative tested_negative tested_negative tested_negative tested_negative tested_positive tested_negative



Time taken to build model (full training data) : 0.03 seconds

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

Clustered Instances

0       63 (  8%)
1      107 ( 14%)
2      171 ( 22%)
3      139 ( 18%)
4      268 ( 35%)
5       20 (  3%)
```

使用Weka实现k-means聚类的算法示例

- 结果中的 Within cluster sum of squared errors，表示误差平方和。这是评价聚类好坏的标准，数值越小说明同一簇实例之间的距离越小

```
kMeans
======

Number of iterations: 9
Within cluster sum of squared errors: 110.23525958855677

Initial starting points (random):

Cluster 0: 1,126,56,29,152,28.7,0.801,21,tested_negative
Cluster 1: 8,95,72,0,0,36.8,0.485,57,tested_negative
Cluster 2: 1,97,66,15,140,23.2,0.487,22,tested_negative
Cluster 3: 2,112,68,22,94,34.1,0.315,26,tested_negative
Cluster 4: 14,100,78,25,184,36.6,0.412,46,tested_positive
Cluster 5: 0,94,0,0,0,0,0.256,25,tested_negative

Missing values globally replaced with mean/mode
```

使用Weka实现k-means聚类的算法示例

- 将“seed”值修改为“20”，再次运行。观察结果。

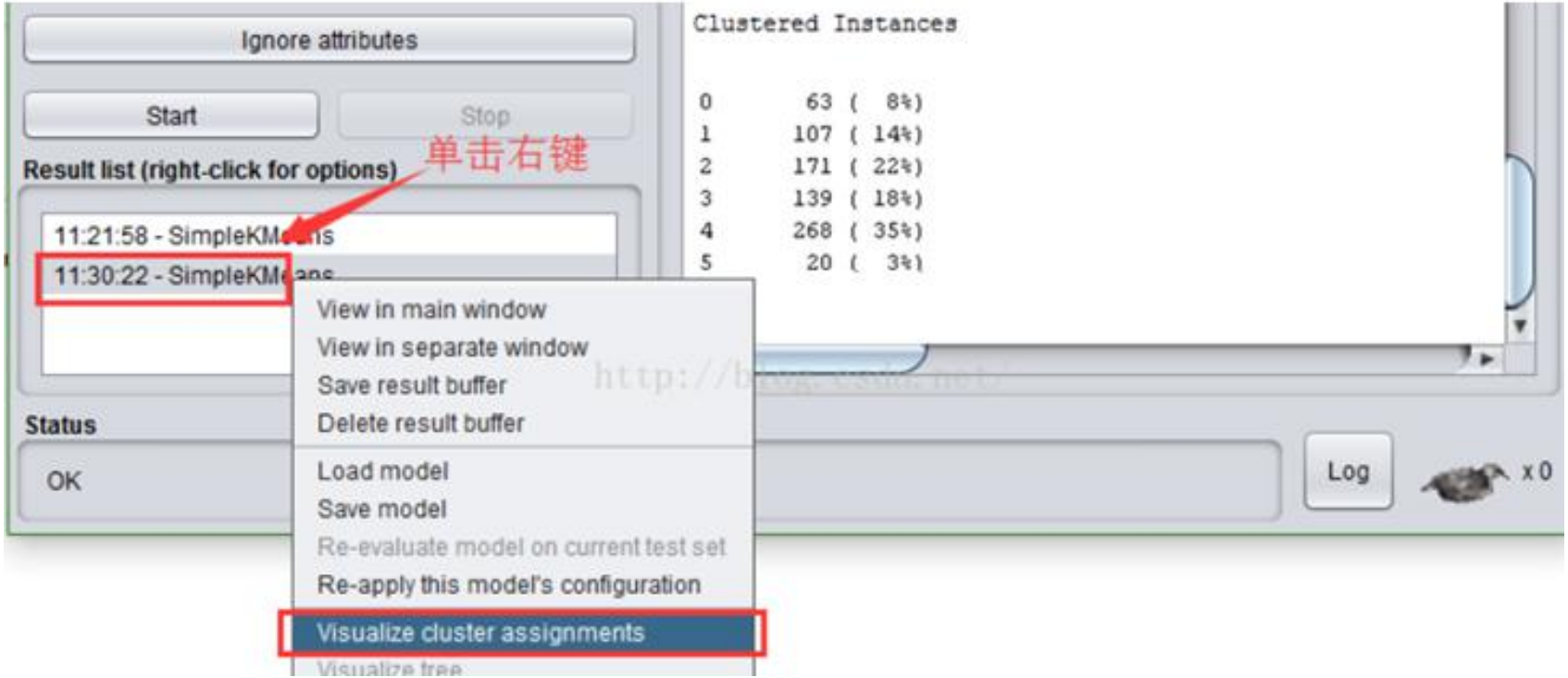
说明同一簇实例之间的距离变小了，聚类的结果也就越好。

多次试验，找到该值趋于最小的值，即得到了本次实验最好的方案结果

```
kMeans
======

Number of iterations: 16
Within cluster sum of squared errors: 99.80911137649608

Initial starting points (random):

Cluster 0: 6,125,78,31,0,27.6,0.565,49,tested_positive
Cluster 1: 6,117,96,0,0,28.7,0.157,30,tested_negative
Cluster 2: 1,128,88,39,110,36.5,1.057,37,tested_positive
Cluster 3: 5,106,82,30,0,39.5,0.286,38,tested_negative
Cluster 4: 8,110,76,0,0,27.8,0.237,58,tested_negative
Cluster 5: 2,158,90,0,0,31.6,0.805,66,tested_positive

Missing values globally replaced with mean/mode
```

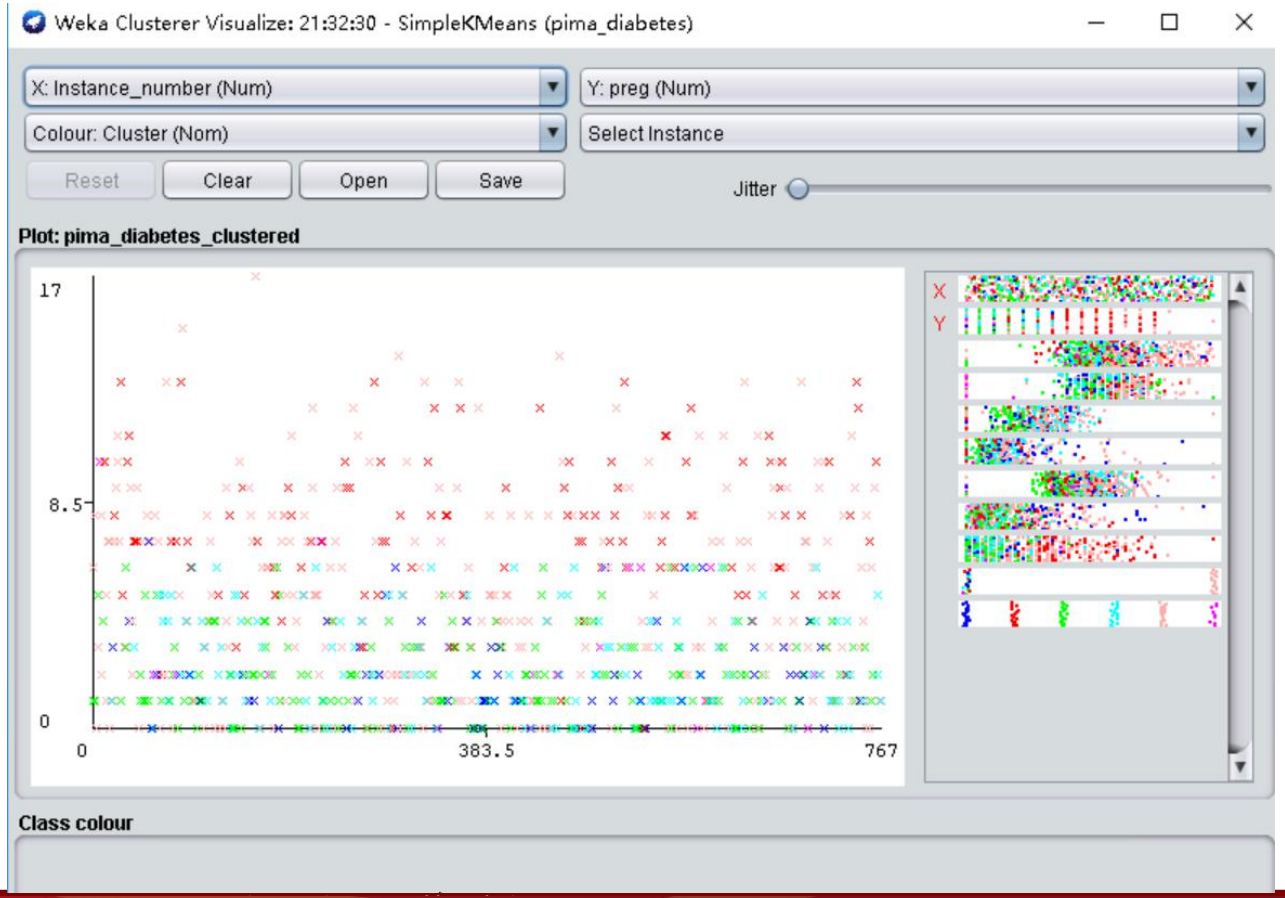
使用Weka实现k-means聚类的算法示例

- 为了观察可视化的聚类结果，在左下方“Result list”列出的结果上右击，点“ Visualize cluster assignments”



使用Weka实现k-means聚类的算法示例

- 弹出的窗口给出了 各实例的散点图。
- 可以点“Save”把聚类结果保存成ARFF文件



谢 谢

Thank you for your attention!