法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意及内容，我们保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

聚类(上)

邹博

本次目标

- 理解相似度度量的各种方法与相互联系
- 掌握K-means聚类的思路和使用条件
- 层次聚类的思路和方法
- 聚类指标及其意义

聚类的定义

- 聚类就是对大量未知标注的数据集，按数据的内在相似性将数据集划分为多个类别，使类别内的数据相似度较大而类别间的数据相似度较小
  - 无监督

相似度/距离计算方法总结

- 闵可夫斯基距离Minkowski/欧式距离 $dist(X,Y)=\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}$
- 杰卡德相似系数(Jaccard) $J(A,B)=\frac{|A\cap B|}{|A\cup B|}$
- 余弦相似度(cosine similarity) $\cos(\theta)=\frac{a^Tb}{|a|\cdot|b|}$
- Pearson相似系数 $\rho_{XY}=\frac{\operatorname{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}=\frac{\sum_{i=1}^{n}(X_i-\mu_X)(Y_i-\mu_Y)}{\sqrt{\sum_{i=1}^{n}(X_i-\mu_X)^2}\sqrt{\sum_{i=1}^{n}(Y_i-\mu_Y)^2}}$
- 相对熵(K-L距离) $D(p\|q)=\sum_x p(x)\log\frac{p(x)}{q(x)}=E_{p(x)}\log\frac{p(x)}{q(x)}$
- Hellinger距离 $D_\alpha(p\|q)=\frac{2}{1-\alpha^2}\left(1-\int p(x)^{\frac{1+\alpha}{2}}q(x)^{\frac{1-\alpha}{2}}dx\right)$

Hellinger distance

$$D_\alpha(p\|q)=\frac{2}{1-\alpha^2}\left(1-\int p(x)^{\frac{1+\alpha}{2}}q(x)^{\frac{1-\alpha}{2}}dx\right)$$

$$\Rightarrow D_H(p\|q)=2\left(1-\int\sqrt{p(x)q(x)}dx\right)\qquad s.t.\quad \alpha=0$$

$$=2-2\int\sqrt{p(x)q(x)}dx$$

$$=\int p(x)dx+\int q(x)dx-\int 2\sqrt{p(x)q(x)}dx$$

$$=\int\left(p(x)-2\sqrt{p(x)q(x)}+q(x)\right)dx$$

$$=\int\left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2dx$$

- 该距离满足三角不等式，是对称、非负距离

余弦相似度与Pearson相似系数

- n维向量x和y的夹角记做θ，根据余弦定理，其余弦值为：

$$\cos(\theta)=\frac{x^T y}{|x|\cdot|y|}=\frac{\sum_{i=1}^{n}x_i y_i}{\sqrt{\sum_{i=1}^{n}x_i^2}\sqrt{\sum_{i=1}^{n}y_i^2}}$$

- 这两个向量的相关系数是：

$$\rho_{XY}=\frac{\mathrm{cov}(X,Y)}{\sigma_X\sigma_Y}=\frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X\sigma_Y}=\frac{\sum_{i=1}^{n}(x_i-\mu_X)(y_i-\mu_Y)}{\sqrt{\sum_{i=1}^{n}(x_i-\mu_X)^2}\sqrt{\sum_{i=1}^{n}(y_i-\mu_Y)^2}}$$

- 相关系数即将x、y坐标向量各自**平移到原点后的夹角余弦**！
  - 这即解释了为何文档间求距离使用**夹角余弦**——因为这一物理量表征了文档**去均值化**后的随机向量间**相关系数**。

聚类的基本思想

- 给定一个有N个对象的数据集，构造数据的k个簇，k≤n。满足下列条件：
  - 每一个簇至少包含一个对象
  - 每一个对象属于且仅属于一个簇
  - 将满足上述条件的k个簇称作一个合理划分
- 基本思想：对于给定的类别数目k，首先给出初始划分，通过迭代改变样本和簇的隶属关系，使得每一次改进之后的划分方案都较前一次好。

k-Means算法

- k-Means算法，也被称为k-平均或k-均值，是一种广泛使用的聚类算法，或者成为其他聚类算法的基础。
- 假定输入样本为S=$x_1,x_2,...,x_m$，则算法步骤为：
  - 选择初始的k个类别中心$\mu_1\mu_2...\mu_k$
  - 对于每个样本x_i，将其标记为距离类别中心最近的类别，即：

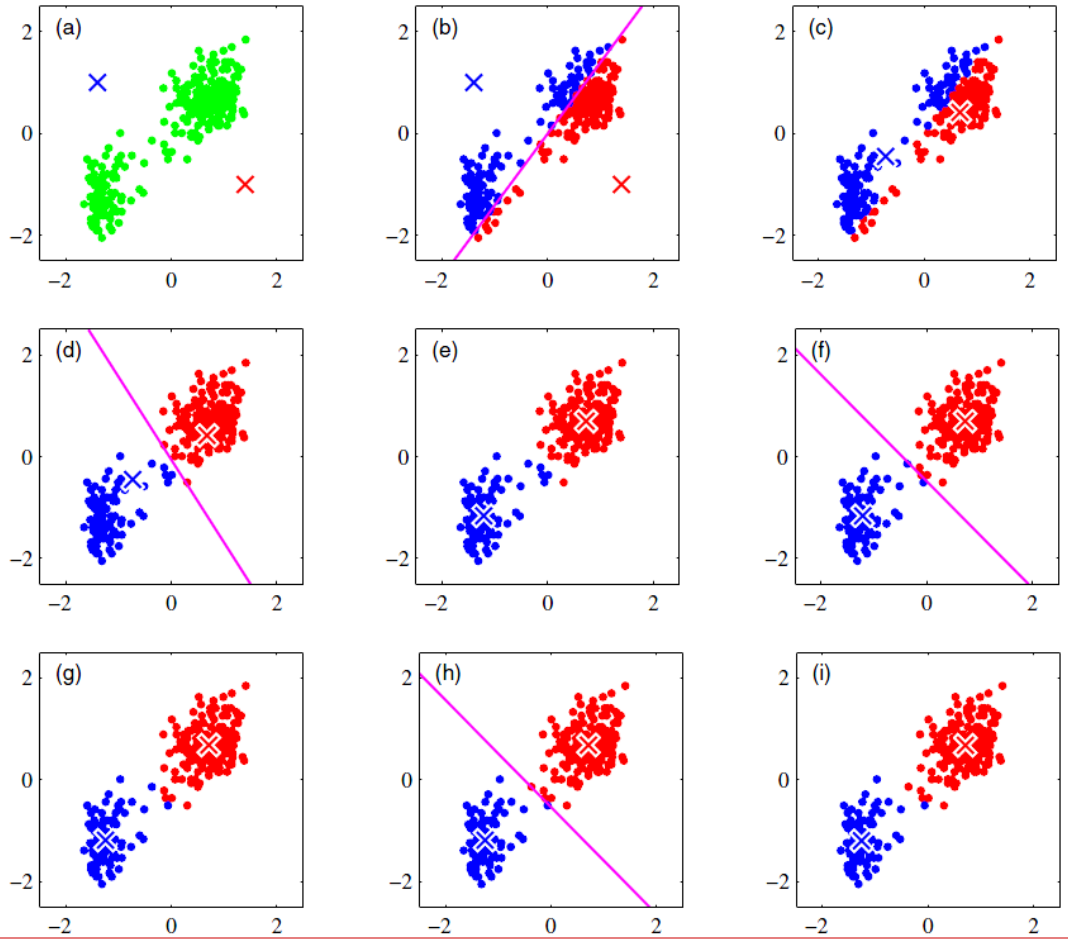
$$label_i = \arg\min_{1 \le j \le k} \left\| x_i - \mu_j \right\|$$

  - 将每个类别中心更新为隶属该类别的所有样本的均值

$$\mu_j = \frac{1}{|c_j|} \sum_{i \in c_j} x_i$$

  - 重复最后两步，直到类别中心的变化小于某阈值。
- 中止条件：
  - 迭代次数/簇中心变化率/最小平方误差MSE(Minimum Squared Error)

k-Means过程



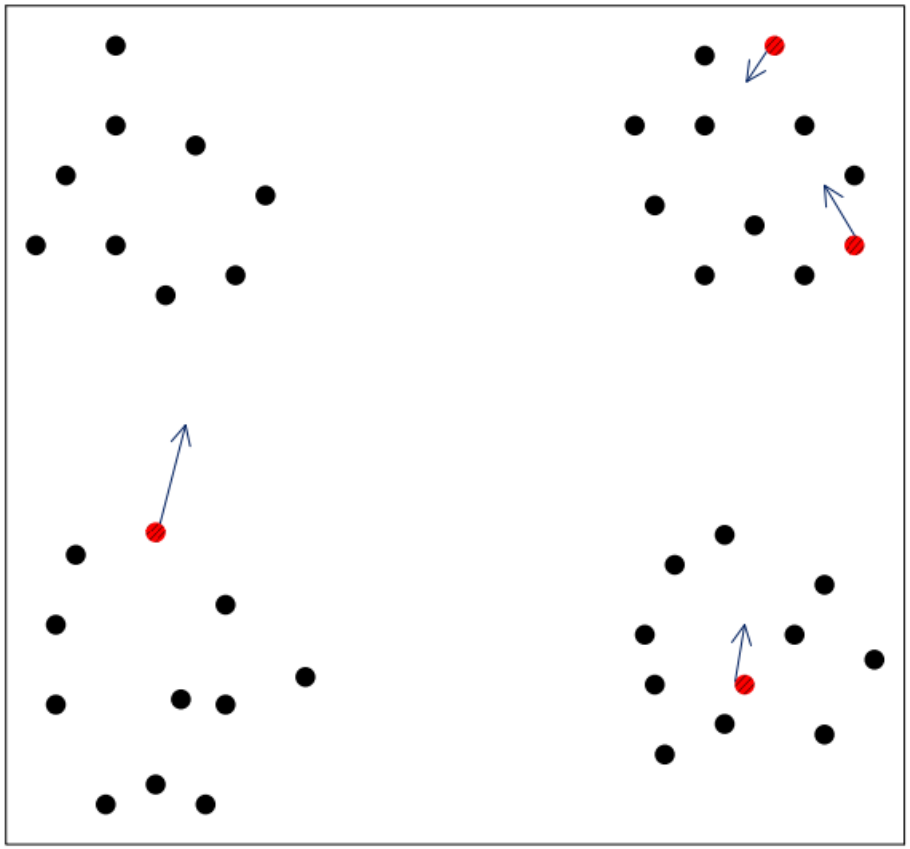
Code

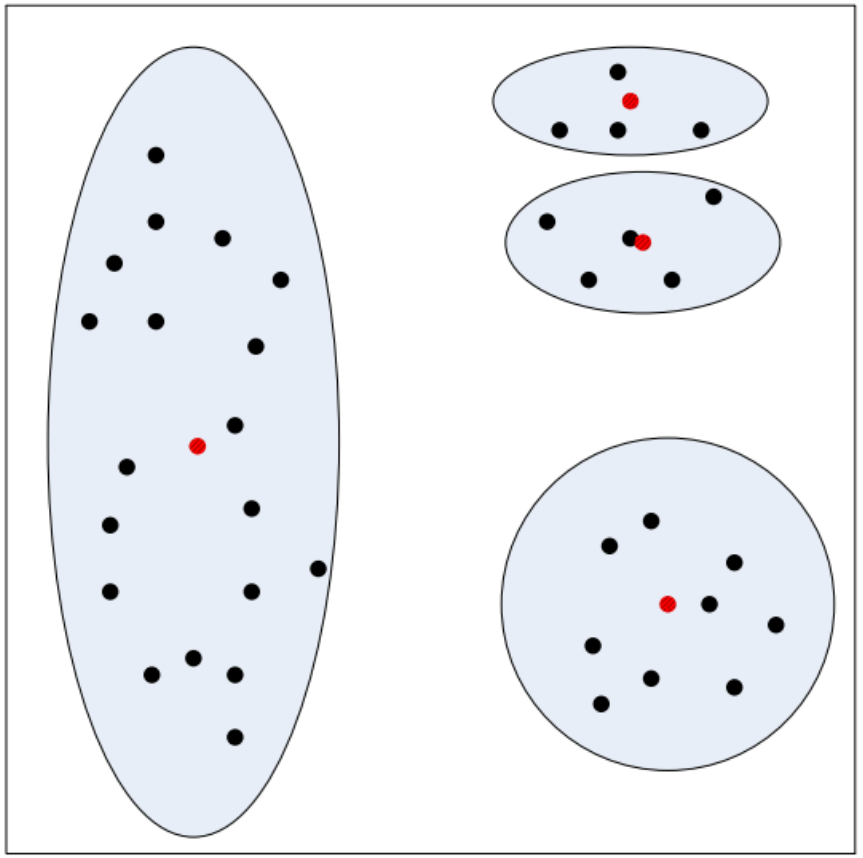
```
def k_means(data):
    m = len(data)
    n = len(data[0])
    cluster = [-1 for x in range(m)]      # 所有样本尚未聚类
    cluster_center = [[] for x in range(k)]  # 聚类中心
    cc = [[] for x in range(k)]           # 下一轮的聚类中心
    c_number = [0 for x in range(k)]      # 每个簇中样本的数目

    # 随机选择簇中心
    i = 0
    while i < k:
        j = random.randint(0, m-1)
        if is_similar(data[j], cluster_center):
            continue
        cluster_center[i] = data[j][:]
        cc[i] = [0 for x in range(n)]
        i += 1
    for times in range(40):
        for i in range(m):
            c = nearest(data[i], cluster_center)
            cluster[i] = c       # 第i个样本归于第c簇
            c_number[c] += 1
            add(cc[c], data[i])
        for i in range(k):
            divide(cc[i], c_number[i])
            c_number[i] = 0
            cluster_center[i] = cc[i][:]
            zero_list(cc[i])
        print times, cluster_center
    return cluster
```

对k-Means的思考

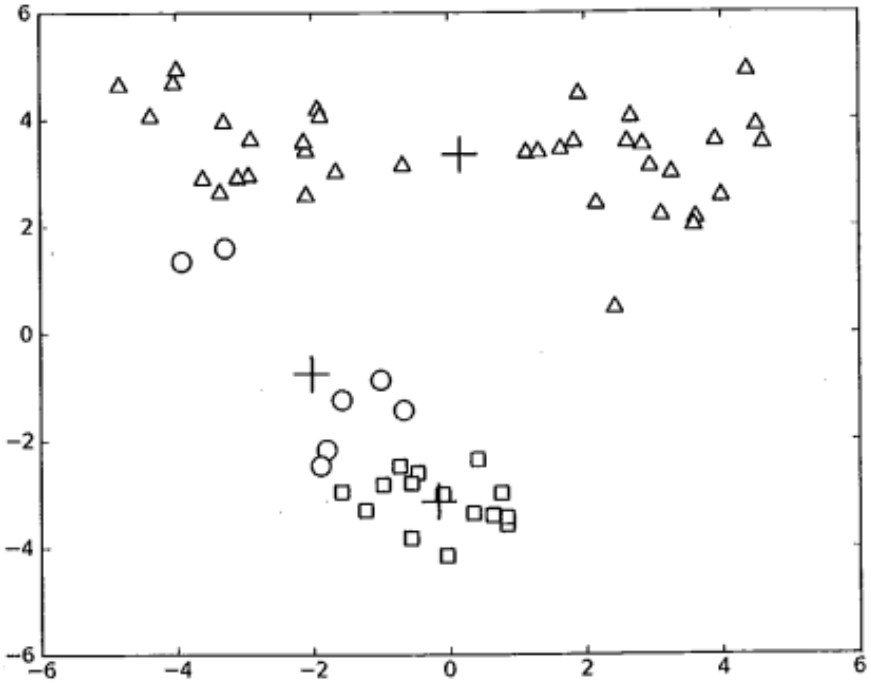
- k-Means将簇中所有点的均值作为新质心，若簇中含有异常点，将导致均值偏离严重。以一维数据为例：
  - 数组1、2、3、4、100的均值为22，显然距离“大多数”数据1、2、3、4比较远
  - 改成求数组的中位数3，在该实例中更为稳妥。
  - 这种聚类方式即k-Mediods聚类(K中值距离)
- 初值的选择，对聚类结果有影响吗？
  - 如何避免？

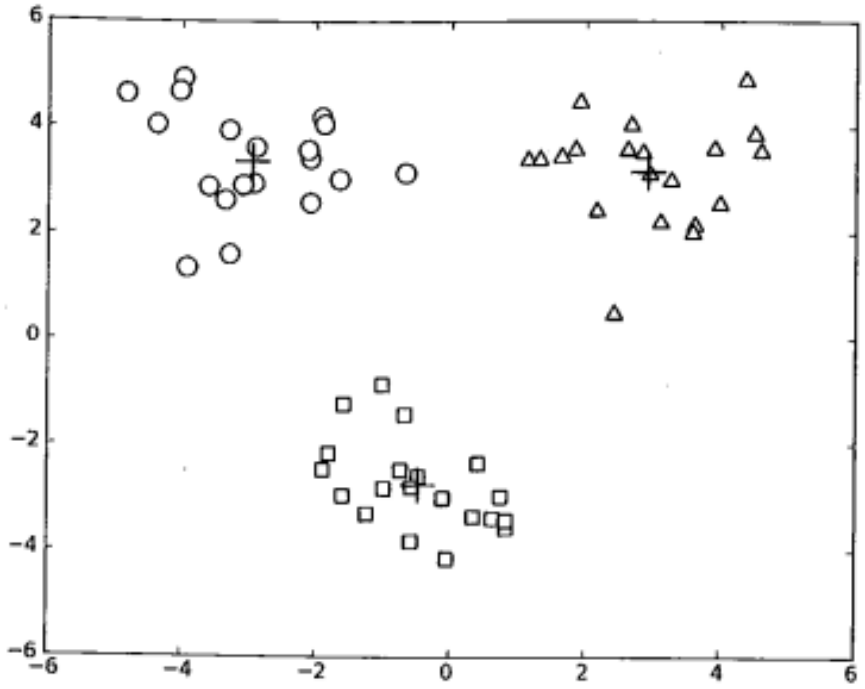
k-Means是初值敏感的





二分k-Means





Code2

```
def k_means(data):
    m = len(data)       # 样本个数
    n = len(data[0])    # 维度
    cluster_center = np.zeros((k, n))  # 聚类中心

    # 选择合适的初始聚类中心
    j = np.random.randint(m)    # [0,m)
    cluster_center[0] = data[j][:]
    dis = np.zeros(m)-1           # 样本到当前所有聚类中心的最近距离
    i = 0
    while i < k-1:
        for j in range(m):        # j:样本
            d = (cluster_center[i]-data[j]) ** 2  # data[j]与最新聚类中心cluster_cen
            d = np.sum(d)
            if (dis[j] < 0) or (dis[j] > d):
                dis[j] = d
        j = random_select(dis)  # 按照dis加权选择样本j
        i += 1
        cluster_center[i] = data[j][:]

    # 聚类
    cluster = np.zeros(m, dtype=np.int) - 1     # 所有样本尚未聚类
    cc = np.zeros((k, n))          # 下一轮的聚类中心
    c_number = np.zeros(k)     # 每个簇中样本的数目
    for times in range(40):
        for i in range(m):
            c = nearest(data[i], cluster_center)
            cluster[i] = c      # 第i个样本归于第c簇
            c_number[c] += 1
            cc[c] += data[i]
        for i in range(k):
            cluster_center[i] = cc[i] / c_number[i]
        cc.flat = 0
        c_number.flat = 0
    return cluster
```

k-Means的公式化解释

- 记K个簇中心为$\mu_1, \mu_2, \cdots, \mu_k$，每个簇的样本数目为$N_1, N_2, \cdots, N_k$
- 使用平方误差作为目标函数：

$$J(\mu_1, \mu_2, \cdots \mu_k) = \frac{1}{2}\sum_{j=1}^{K}\sum_{i=1}^{N_j}(x_i - \mu_j)^2$$

- 对关于$\mu_1, \mu_2, \cdots, \mu_k$的函数求偏导，其驻点为：

$$\frac{\partial J}{\partial \mu_j} = -\sum_{i=1}^{N_j}(x_i - \mu_j) \xrightarrow{令} 0 \Rightarrow \mu_j = \frac{1}{N_j}\sum_{i=1}^{N_j} x_i$$

如果使用其他相似度/距离度量

- 如：余弦相似度：$\cos\left(x_i, x_j\right)=\frac{x_i^T \cdot x_j}{\left|x_i\right| \cdot \left|x_j\right|}$

- 使用余弦相似度平方作为目标函数：

$$J\left(\mu_1, \mu_2, \cdots \mu_k\right)=\frac{1}{2}\sum_{j=1}^{K}\sum_{i=1}^{N_j}\cos^2\left(x_i, \mu_j\right)$$

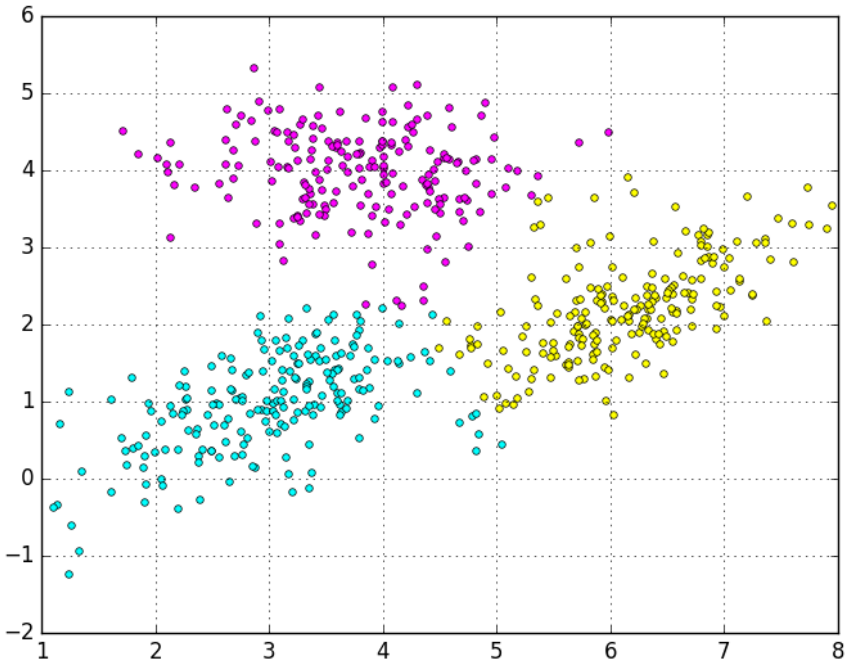
- 对关于$\mu_1, \mu_2, \cdots, \mu_k$的函数求偏导，其驻点为：

$$\frac{\partial J}{\partial \mu_j}=-\sum_{i=1}^{N_j}\cos\left(x_i, \mu_j\right)\sin\left(x_i, \mu_j\right)\cdot x_i \xrightarrow{令} 0 \Rightarrow \mu_j = ?$$

Mini-batch k-Means算法描述

Mini-batch k-Means

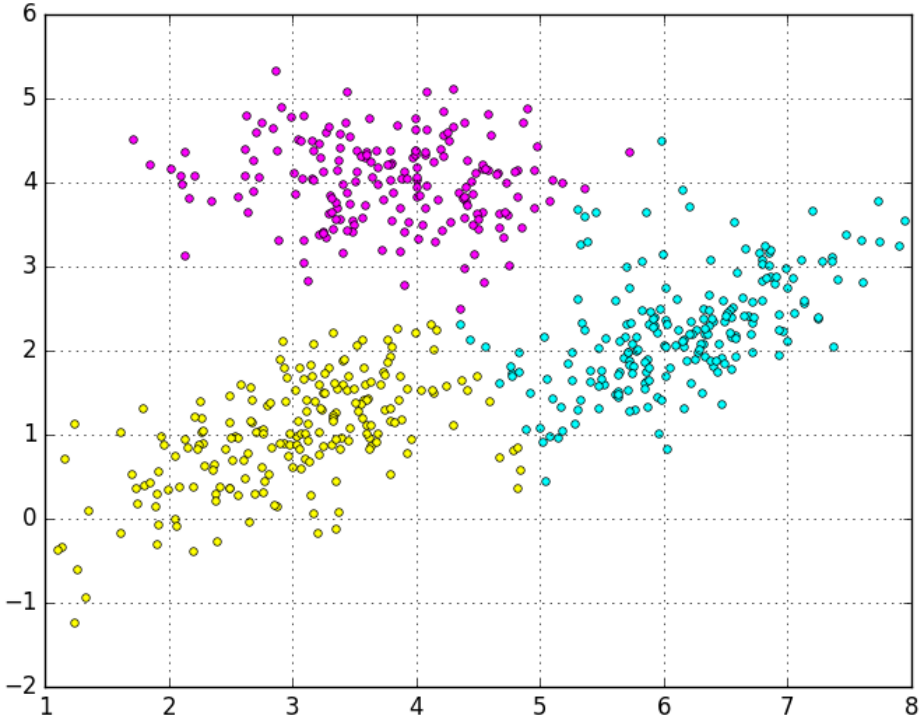
Given: k, mini-batch size b, iterations t, data set X
Initialize each $\mathbf{c} \in C$ with an $\mathbf{x}$ picked randomly from X
$\mathbf{v} \leftarrow 0$
for $i = 1$ to t **do**
  $M \leftarrow b$ examples picked randomly fi
  for $\mathbf{x} \in M$ **do**
    $\mathbf{d}[\mathbf{x}] \leftarrow f(C, \mathbf{x})$ // Cache the ce
  end for
  for $\mathbf{x} \in M$ **do**
    $\mathbf{c} \leftarrow \mathbf{d}[\mathbf{x}]$ // Get cached ce
    $\mathbf{v}[\mathbf{c}] \leftarrow \mathbf{v}[\mathbf{c}] + 1$ // Update per-ce
    $\eta \leftarrow \frac{1}{\mathbf{v}[\mathbf{c}]}$ // Get per-cente
    $\mathbf{c} \leftarrow (1 - \eta)\mathbf{c} + \eta\mathbf{x}$ // Take
  end for
end for

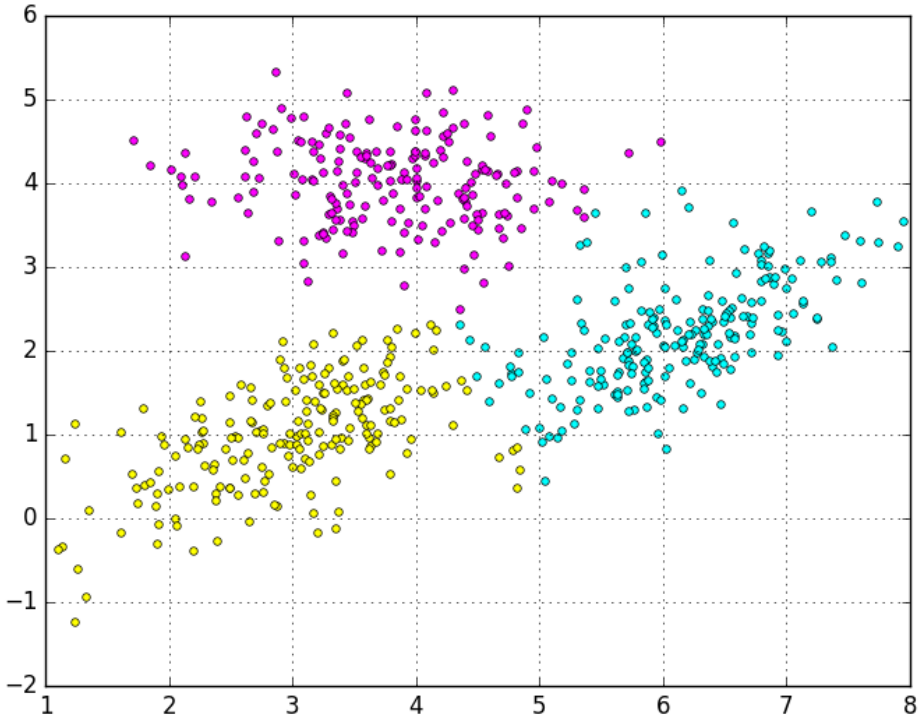


Code3

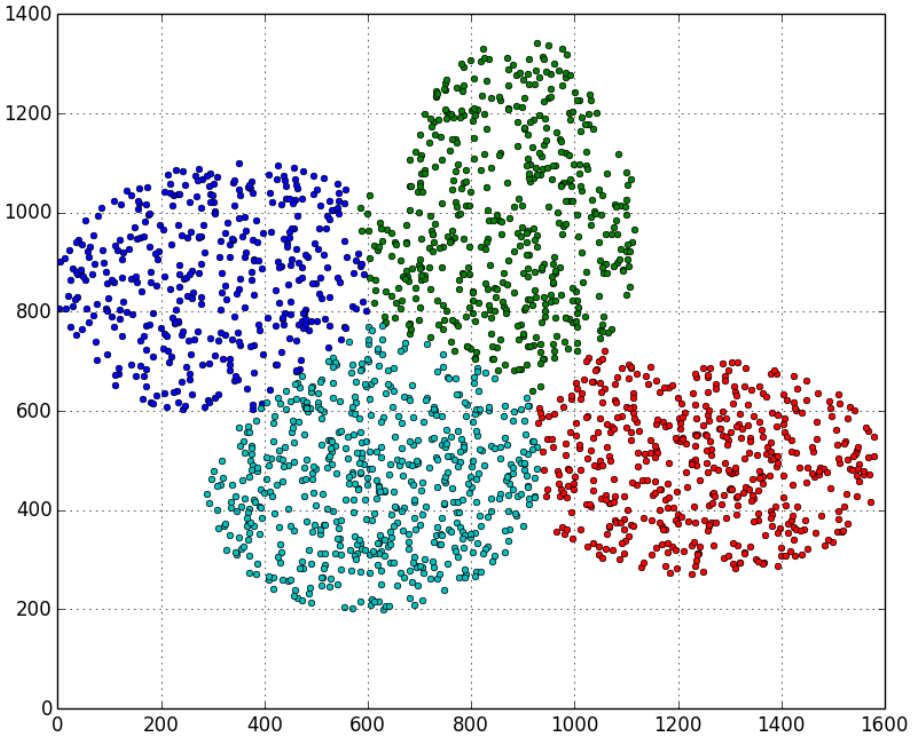
```
def k_means(data, k):
    m = len(data)       # 样本个数
    n = len(data[0])    # 维度
    cluster_center = np.zeros((k, n))  # 聚类中心

    # 选择合适的初始聚类中心
    j = np.random.randint(m)    # [0,m)
    cluster_center[0] = data[j][:]
    dis = np.zeros(m)-1           # 样本到当前所有聚类中心的最近距离
    i = 0
    while i < k-1:
        for j in range(m):        # j:样本
            d = (cluster_center[i]-data[j]) ** 2  # data[j]与最新聚类中心cluster_cent
            d = np.sum(d)
            if (dis[j] < 0) or (dis[j] > d):
                dis[j] = d
        j = random_select(dis)  # 按照dis加权选择样本j
        i += 1
        cluster_center[i] = data[j][:]
        print "Find Cluster Center:", i

    # 聚类
    cluster = np.zeros(m, dtype=np.int) - 1     # 所有样本尚未聚类
    cc = np.zeros((k, n))      # 下一轮的聚类中心
    c_number = np.zeros(k)     # 每个簇的样本数目
    times = 50
    for t in range(times):
        for i in range(m):
            if np.random.random() * m > 100:
                continue
            c = nearest(data[i], cluster_center)
            cluster[i] = c      # 第i个样本归于第c簇
            cc[c] += data[i]
            c_number[c] += 1
        for i in range(k):
            cluster_center[i] = cc[i] / c_number[i]
        cc.flat = 0
        c_number.flat = 0
        print t, '%.2f%%' % (100 *float(t) / times)
        print cluster_center
    return cluster
```

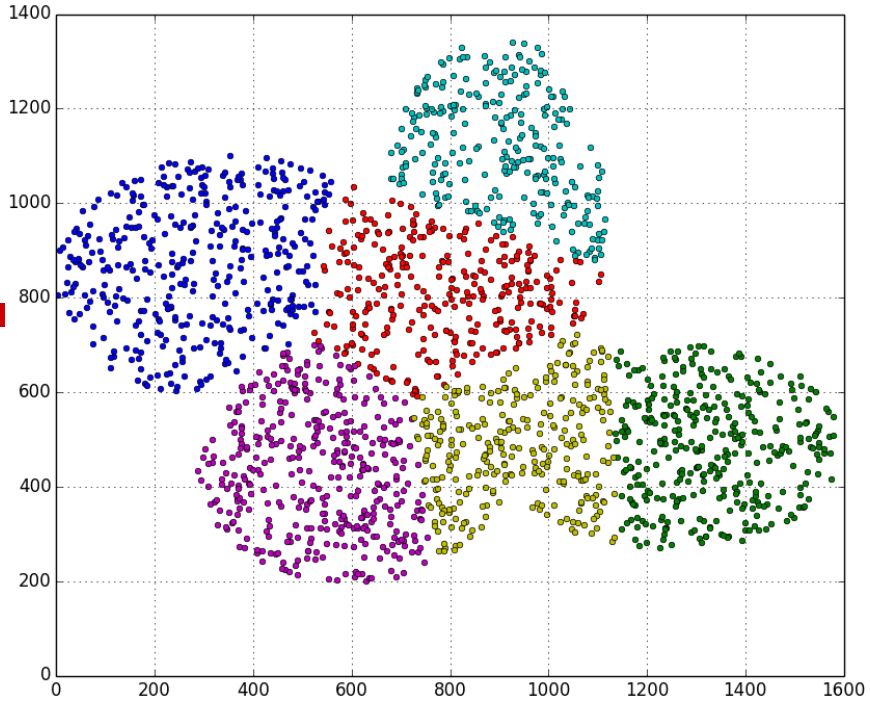
Mini-batch k-Means效果

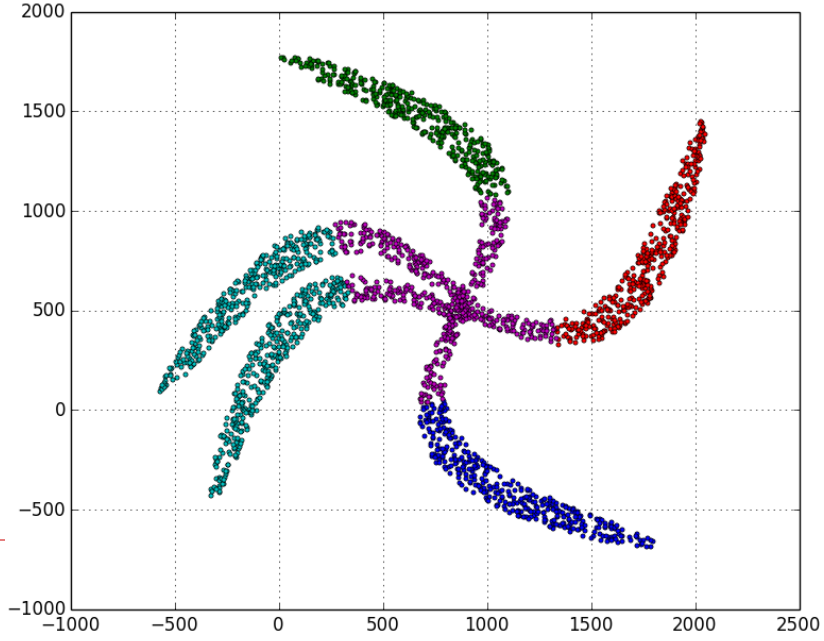




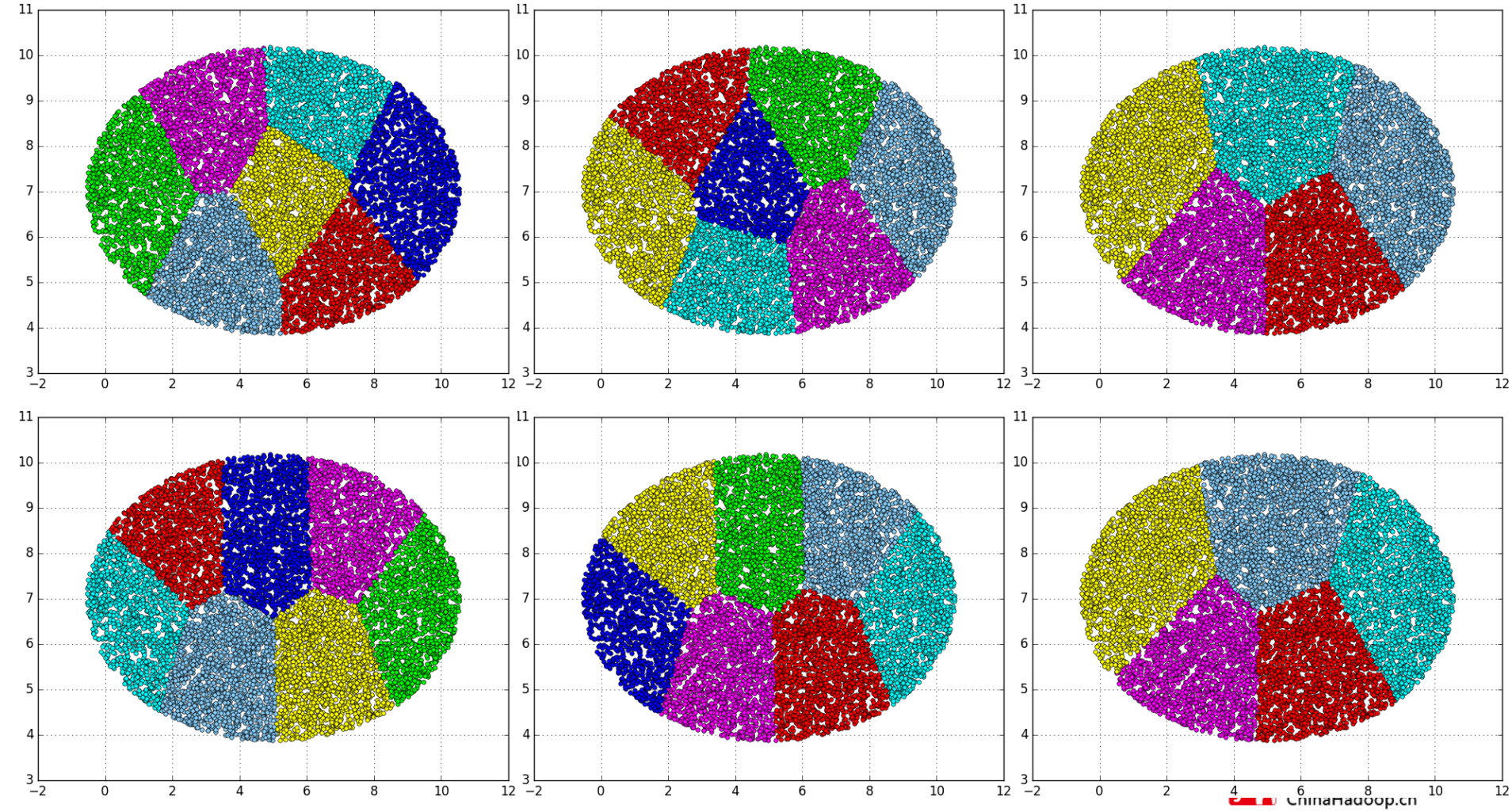
k-Means适用范围


1400
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600


1400
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600


2000
1500
1000
500
0
−500
−1000
−1000
−500
0
500
1000
1500
2000
2500

k-Means++算法测试



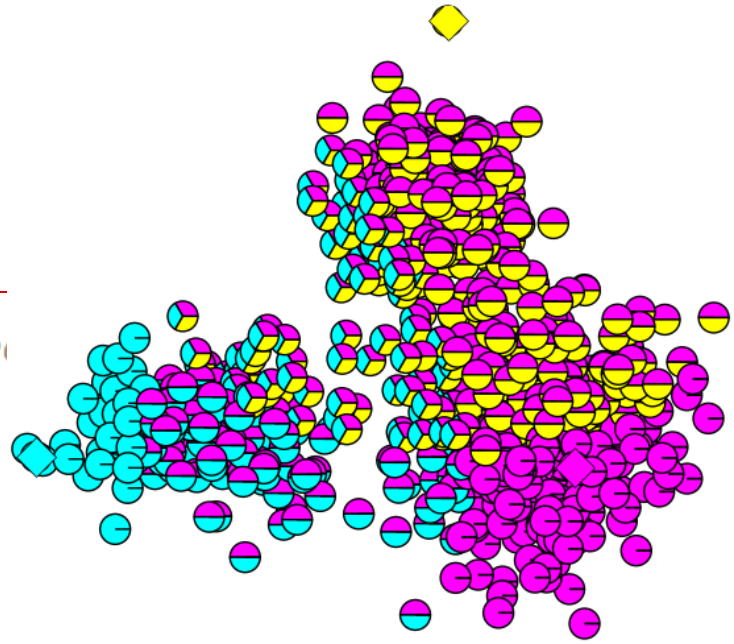
k-Means聚类方法总结

- 优点：
  - 是解决聚类问题的一种经典算法，简单、快速
  - 对处理大数据集，该算法保持可伸缩性和高效率
  - 当簇近似为高斯分布时，它的效果较好
- 缺点
  - 在簇的平均值可被定义的情况下才能使用，可能不适用于某些应用
  - 必须事先给出k(要生成的簇的数目)，而且对初值敏感，对于不同的初始值，可能会导致不同结果。
  - 不适合于发现非凸形状的簇或者大小差别很大的簇
  - 对躁声和孤立点数据敏感
- 可作为其他聚类方法的基础算法，如谱聚类

Canopy算法

- 虽然Canopy算法可以划归为聚类算法，但更多的可以使用Canopy算法做空间索引，其时空复杂度都很出色，算法描述如下：
- 对于给定样本$x_1, x_2 \cdots x_m$，给定先验值r_1, r_2，$(r_1 < r_2)$
  - $x_1, x_2 \cdots x_m$形成列表L；构造$x_j (1 \le j \le m)$的空列表C_j；
  - 随机选择L中的样本c，要求c的列表C_c为空：
    - 计算L中样本x_j与c的距离d_j
      - 若$d_j < r_1$，则在L中删除x_j，将C_j赋值为{c}
      - 否则，若$d_j < r_2$，则将C_j增加{c}
  - 若L中没有不满足条件的样本c，算法结束。

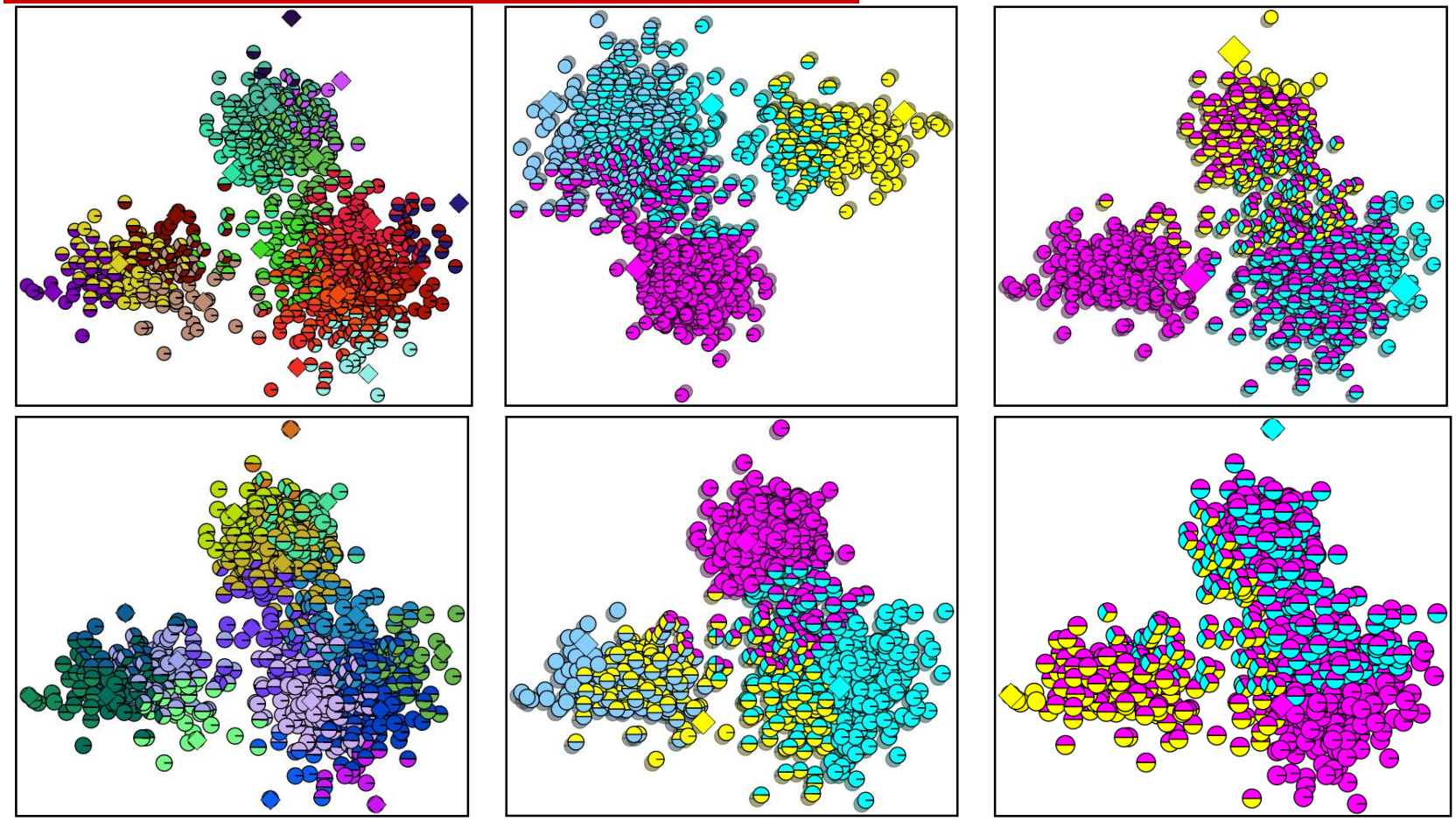
Code



```
def canopy(data, d_near, d_far):          # d_ne
    m = len(data)          # 样本个数
    cluster = [[] for i in range(m)]
    center = []
    has = m
    while has > 0:
        i = np.random.randint(has)
        i = find_new_canopy(cluster, i)      # 查找cluster中第i个为空的值
        center.append(i)                     # canopy中心
        if i == -1:
            break
        for j in range(m):    # 针对新canopy中心data[i]，计算所有样本与之距离
            d = distance(data[i], data[j])
            if d < d_near:
                cluster[j] = [i]
            elif d < d_far:
                cluster[j].append(i)
        has = calc_candidate(cluster)
    return cluster, center
```

Canopy的调参

0.5	1	1.5
0.5	1	2



聚类的衡量指标

- 均一性
  - Homogeneity

$$h=\begin{cases}1 & if\ H(C)=0\\ 1-\dfrac{H(C|K)}{H(C)} & otherwise\end{cases}$$

  - 一个簇只包含一个类别的样本，则满足均一性
- 完整性
  - Completeness

$$c=\begin{cases}1 & if\ H(K)=0\\ 1-\dfrac{H(K|C)}{H(K)} & otherwise\end{cases}$$

  - 同类别样本被归类到相同簇中，则满足完整性

- V-measure $v_\beta=\dfrac{(1+\beta)\cdot h\cdot c}{\beta\cdot h+c}$
  - 均一性和完整性的加权平均

ARI

C	Y_1	Y_2	$\cdots$	Y_s	sum
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sum	b_1	b_2	$\cdots$	b_s	N

- 数据集S共有N个元素，两个聚类结果分别是：

$$X=\{X_1,X_2,\cdots X_r\} \quad Y=\{Y_1,Y_2,\cdots Y_s\}$$

- X和Y的元素个数为：$a=\{a_1,a_2,\cdots a_r\}$ $b=\{b_1,b_2,\cdots b_s\}$
- 记：$n_{ij}=|X_i \cap Y_j|$
- 则：

$$ARI=\frac{Index-EIndex}{MaxIndex-EIndex}=\frac{\sum_{i,j}C_{n_{ij}}^2-\left[\left(\sum_i C_{a_i}^2\right)\cdot\left(\sum_j C_{b_j}^2\right)\right]\Big/C_n^2}{\frac{1}{2}\left[\left(\sum_i C_{a_i}^2\right)+\left(\sum_j C_{b_j}^2\right)\right]-\left[\left(\sum_i C_{a_i}^2\right)\cdot\left(\sum_j C_{b_j}^2\right)\right]\Big/C_n^2}$$

AMI

- 使用与ARI相同的记号，根据信息熵，得：
- 互信息/正则化互信息：

$$MI(X,Y)=\sum_{i=1}^{r}\sum_{j=1}^{s}P(i,j)\log\frac{P(i,j)}{P(i)P(j)} \qquad NMI(X,Y)=\frac{MI(X,Y)}{\sqrt{H(X)H(Y)}}$$

- X服从超几何分布，求互信息期望：

$$E[MI]=\sum_{x}P(X=x)MI(X,Y)=\sum_{x=\max(1,a_i+b_i-N)}^{\min(a_i,b_i)}\left[MI\cdot\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!x!(a_i-x)!(b_j-x)!(N-a_i-b_j+x)!}\right]$$

- 借鉴ARI，有：$AMI(X,Y)=\dfrac{MI(X,Y)-E[MI(X,Y)]}{\max\{H(X),H(Y)\}-E[MI(X,Y)]}$

轮廓系数(Silhouette)

- Silhouette系数是对聚类结果有效性的解释和验证，由Peter J. Rousseeuw于1986提出。
- 计算样本i到同簇其他样本的平均距离a_i。a_i越小，说明样本i越应该被聚类到该簇。将a_i称为样本i的**簇内不相似度**。
  - 簇C中所有样本的a_i均值称为簇C的**簇不相似度**。
- 计算样本i到其他某簇C_j的所有样本的平均距离b_{ij}，称为样本i与簇C_j的不相似度。定义为样本i的**簇间不相似度**：$b_i = \min\{b_{i1}, b_{i2}, \cdots\cdots b_{i,K}\}$
  - b_i越大，说明样本i越不属于其他簇。

轮廓系数(Silhouette)

- 根据样本i的簇内不相似度a_i和簇间不相似度b_i，定义样本i的轮廓系数：

$$s(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$$

$$s(i)=\begin{cases}1-\dfrac{a(i)}{b(i)}, & a(i)<b(i)\\ 0, & a(i)=b(i)\\ \dfrac{b(i)}{a(i)}-1, & a(i)>b(i)\end{cases}$$

- s_i接近1，则说明样本i聚类合理；s_i接近-1，则说明样本i更应该分类到另外的簇；若s_i近似为0，则说明样本i在两个簇的边界上。
- 所有样本的s_i的均值称为聚类结果的轮廓系数，是该聚类是否合理、有效的度量。

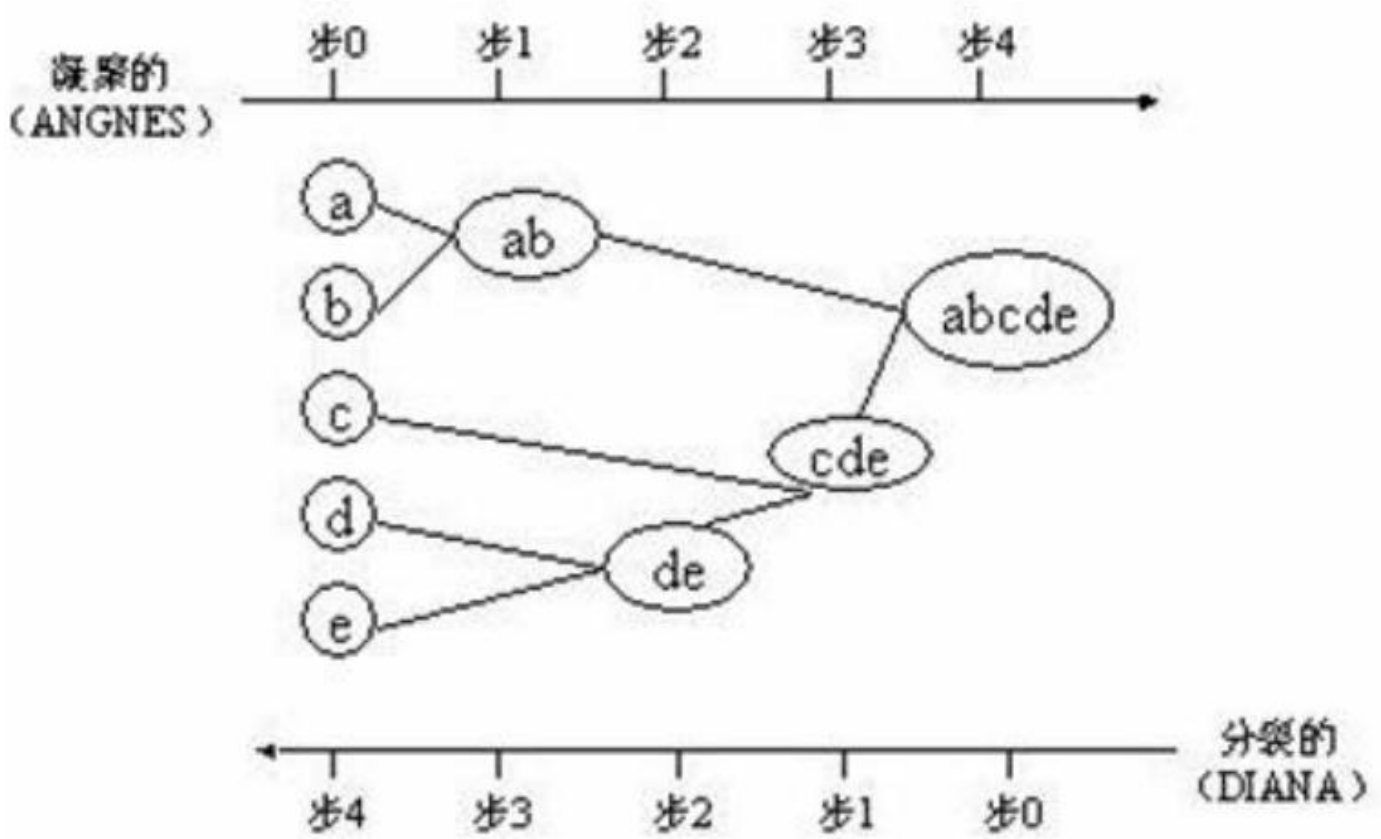
层次聚类方法

- 层次聚类方法对给定的数据集进行层次的分解，直到某种条件满足为止。具体又可分为：
- 凝聚的层次聚类：AGNES算法
  - 一种自底向上的策略，首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到某个终结条件被满足。
- 分裂的层次聚类：DIANA算法
  - 采用自顶向下的策略，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到达到了某个终结条件。

AGNES和DIANA算法

- AGNES (AGglomerative NESting)算法最初将每个对象作为一个簇，然后这些簇根据某些准则被一步步地合并。两个簇间的距离由这两个不同簇中距离最近的数据点对的相似度来确定；聚类的合并过程反复进行直到所有的对象最终满足簇数目。
- DIANA (DIvisive ANAlysis)算法是上述过程的反过程，属于分裂的层次聚类，首先将所有的对象初始化到一个簇中，然后根据一些原则(比如最大的欧式距离)，将该簇分类。直到到达用户指定的簇数目或者两个簇之间的距离超过了某个阈值。

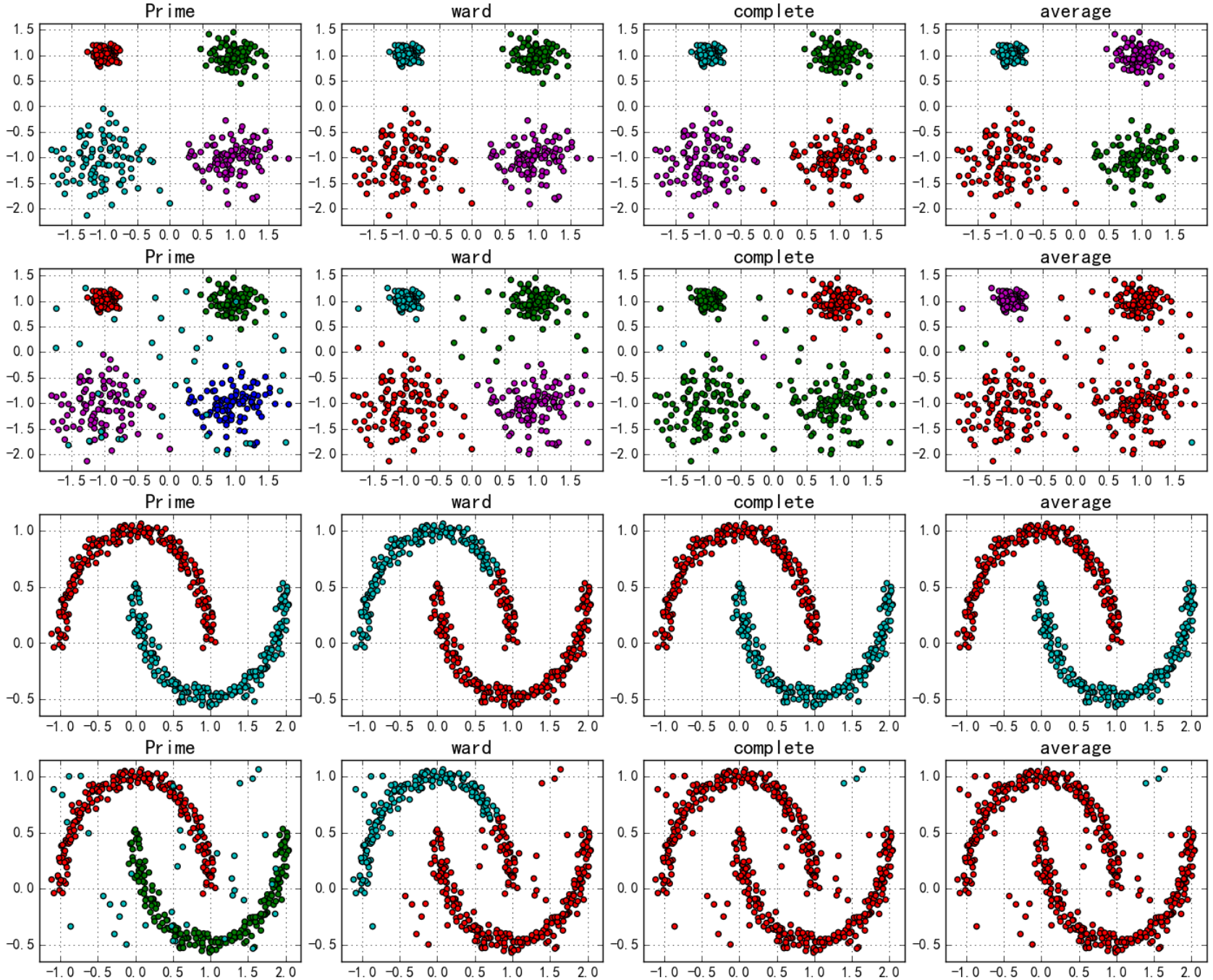
层次聚类


凝聚的
（ANGNES）
步0
步1
步2
步3
步4
a
b
c
d
e
ab
abcde
cde
de
步4
步3
步2
步1
步0
分裂的
（DIANA）

AGNES中簇间距离的不同定义

- 最小距离
  - 两个集合中最近的两个样本的距离
  - 容易形成链状结构
- 最大距离
  - 两个集合中最远的两个样本的距离complete
  - 若存在异常值则不稳定
- 平均距离
  - 1、两个集合中样本间两两距离的平均值average
  - 2、两个集合中样本间两两距离的平方和ward

层次聚类的不同合并策略



参考文献

- Andrew Rosenberg, Julia Hirschberg, *V-Measure: A conditional entropy-based external cluster evaluation measure*, 2007.
- W. M. Rand. *Objective criteria for the evaluation of clustering methods*. Journal of the American Statistical Association. 1971
- Nguyen Xuan Vinh, Julien Epps, James Bailey, *Information theoretic measures for clusterings comparison*, ICML 2009
- Peter J. Rousseeuw, *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*. Computational and Applied Mathematics 20: 53–65, 1987
- https://en.wikipedia.org/wiki/Rand_index
- https://en.wikipedia.org/wiki/Adjusted_mutual_information
- http://hdbscan.readthedocs.io/en/latest/

我们在这里

- http://wenda.ChinaHadoop.cn
  - 视频/课程/社区
- 微博
  - @ChinaHadoop
  - @邹博_机器学习
- 微信公众号
  - 小象学院
  - 大数据分析挖掘

感谢大家！

恳请大家批评指正！