第 12 章　聚类算法 I：顺序算法

12.1　引言

在前一章，我们主要介绍了一些近邻测度。结合聚类过程得到的聚类类型，给出了每种相似性和不相似性的不同解释。这一章和接下来的三章，重点介绍各种聚类算法和准则。就像前面已经阐述过的，专家能给出不同的近邻测度和聚类方案组合会产生不同的结果。

这一章首先简单介绍几种聚类算法，然后主要讲解顺序算法。

12.1.1　可能聚类的数量

给定时间和资源，将集合 X 中的特征向量 $\boldsymbol{x}_i$, $i = 1,\cdots,N$ 分到聚类中的最好方法是识别所有可能的划分，并根据事先确定的准则选择最可判断的聚类。然而，对于中等 N 值这都是不可能的。令 $S(N,\ m)$表示将 N 个向量聚类到 m 组的所有可能结果。由定义可知，聚类不能是空的，很明显需满足下列条件[Spat 80, Jain 88]：

- $S(N, 1) = 1$
- $S(N, N) = 1$
- $S(N, m) = 0$，对于 $m > N$

令 L_{N-1}^{k} 表示 $N-1$ 个向量分到 k 类的所有可能，其中 $k = m, m-1$。第 N 个向量

- 或者添加到 L_{N-1}^{m} 的任一个成员的聚类中
- 或者对 L_{N-1}^{m-1} 的每个成员形成一个新聚类

因此，可以得出

$$S(N,m) = mS(N-1,m) + S(N-1,m-1) \tag{12.1}$$

式(12.1)的解就是所谓的第二类的 Stirling 数量（参见［Liu 68］）[①]：

$$S(N,m) = \frac{1}{m!}\sum_{i=0}^{m}(-1)^{m-i}\binom{m}{i}i^N \tag{12.2}$$

例 12.1　设 $X=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$，要找到将 X 的所有元素分为两类的所有可能的聚类。很容易推导出

$$L_2^1 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}\}$$

和

$$L_2^2 = \{\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\}\}$$

考虑式(12.1)，很容易得到 $S(3, 2) = 2 \times 1 + 1 = 3$。实际上，$L_3^2$ 是

① 与 Cover 理论的二分法数量相比较。

$$L_3^2 = \{\{\boldsymbol{x}_1, \boldsymbol{x}_3\}, \{\boldsymbol{x}_2\}\}, \{\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2, \boldsymbol{x}_3\}\}, \{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3\}\}$$

特殊地，对于 $m = 2$，式(12.2)变成

$$S(N, 2) = 2^{N-1} - 1 \tag{12.3}$$

参见习题 12.1。式(12.2)的一些值是 [Spat 80]

- $S(15, 3) = 2375101$
- $S(20, 4) = 45232115901$
- $S(25, 8) = 690223721118368580$
- $S(100, 5) \approx 10^{68}$

很清楚，如果聚类数固定，则这种计算是有效的。如果不是这种情况，则需要对所有可能的 m 值计算所有可能的聚类。由上面的分析可知，即使对于中等值 N，评估所有的聚类来寻找最可判断的一个也是不现实的。例如，如果要评估 100 个对象分到 5 类的所有可能聚类，用计算机计算每个聚类要用 10^{-12} 秒，大约需 10^{48} 年之后才会得到最可判断的聚类。

12.2　聚类算法的种类

聚类算法可以视为：通过考虑包含在 X 中的所有可能划分集合的一小部分，就可以得到可判断聚类的方案，这个结果依赖于使用的算法和准则。因此，聚类算法是一个试图识别数据集合聚类的特殊性质的学习过程。聚类算法主要包括以下几种。

- **顺序算法**（Sequential algorithm）：这些算法产生一个独立的聚类，它们是非常直接和快速的算法。这种算法的大多数都至少将所有特征向量使用一次或几次（一般不超过五六次），最后的结果依赖于向量参与算法的顺序。这种方法会产生致密和超球面或超椭圆面形状的聚类（取决于使用的距离度量）。这一章的结尾会学到这种方法。
- **层次聚类算法**（Hierarchical clustering algorithm）：这种方法被进一步分为
  - — 合并算法（Agglomerative algorithm）。这些算法会在每一步产生减少聚类数量 m 的聚类序列，聚类生成的结果都来自于前一步的两个聚类的合并。合并算法的典型代表是单一和完全链接算法。合并算法被进一步分为下面这两种：
    - * 由矩阵理论得到的算法
    - * 由图形理论得到的算法

    这些算法适用于长轴聚类（用单一链接算法）和致密聚类（用完全链接算法）。
  - — 分裂算法（Divisive algorithm）。这种算法的原理与合并算法的原理相反，在每一步产生增加聚类数量 m 的聚类序列。在每一步聚类产生的结果都是将前一步的一个聚类分裂成两个得到的。
- **基于代价函数最优的聚类算法**（Clustering algorithms based on cost function optimization）：这种方法用代价函数 J 来量化可判断性，通常聚类数量 m 是固定的。这种算法用微分学概念，通过最优 J 产生连续的聚类，当 J 的局部最优确定时，算法才结束。这种类型的算法也称为迭代函数最优方法，这类算法又可细分为
  - — 硬或脆聚类算法（Hard or crisp clustering algorithm）。其中一个向量绝对属于特定聚类。根据选择的最优准则，以最优分类将向量分到各个聚类中。这种类型中最著名的算法是 Isodata 或者 Lloyd 算法 [Lloy 82, Duda 01]。

— 概率聚类算法（Probabilistic clustering algorithm）。它是硬聚类算法的特例，采用贝叶斯分类方法，并且每个向量 $\boldsymbol{x}$ 被分到使 $P(C_i|\boldsymbol{x})$最大的聚类 C_i 中，通过适当地定义优化任务完成概率估计。

— 模糊聚类算法（Fuzzy clustering algorithm）。在这种算法中，向量属于超过特定阈值的聚类。

— 可能聚类算法（Possibilistic clustering algorithm）。在这种情况下，我们测量向量 $\boldsymbol{x}$ 属于聚类 C_i 的可能性。

— 边界检测算法（Boundary detection algorithm）。不同于用向量本身来确定聚类，这些算法迭代调整聚类的边界。这些算法虽然包括了代价函数优化原理，但它们与以上算法有本质的区别。前述所有算法使用聚类表达，目的是用最优方法来确定局部空间。相反地，边界检测算法寻找聚类间边界最优放置的方法，这使我们决定用独立的章节来阐述这些算法，同时包含下面讨论的算法。

● **其他算法**：最后一类包括一些特殊的聚类技术，这些技术不能归结到上述任何一类中。它们包括：

— 分支和约束聚类算法（Branch and bound clustering algorithm）。对于给定的聚类数 m，这些算法使用预定的准则，不需要考虑到所有可能的聚类就可以提供全局最优聚类。然而，它们的计算量大。

— 遗传聚类算法（Genetic clustering algorithm）。这些算法用一个可能的聚类作为初始种群，迭代生成新的种群，按照相关的准则，新种群一般比原来的种群具有更优的聚类。

— 随机松弛算法（Stochastic relaxation method）。在指定准则情况下，这种方法保证在一定的条件下概率收敛于全局最优聚类，但是计算量大。

需要指出的是随机松弛算法（以及遗传算法、分支和约束算法）是代价函数最优方法。与前面归类的其他算法相比较，下面的每种算法在处理问题上是概念不同的方法，因此我们将分别处理。

— 谷点搜索聚类算法（Valley-seeking clustering algorithm）。这种方法把特征向量当做一个（多维）任意变量 $\boldsymbol{x}$ 的实例，这些算法都基于一个普遍接受的假设，即很多向量驻留的 $\boldsymbol{x}$ 区域对应着 $\boldsymbol{x}$ 的概率密度函数值增加的区域。所以，对概率密度函数的估计可能使得聚类形成的区域更加显著。

— 竞争学习算法（Competitive learning algorithm）。这是迭代方法，不使用代价函数。根据某种距离度量，它们产生一些聚类，而且收敛于最可判断的一个。其典型代表是基本的竞争学习方法和漏洞学习算法。

— 基于形态学变换技术的算法（Algorithms based on morphological transformation technique）。这些算法使用形态学的变换，以获得更好的聚类划分。

— 基于密度的算法（Density-based algorithm）。该算法把聚类视为 l 维空间中数据较为密集区域。从这个角度看，该类算法与谷点搜索算法相似。然而，现在解决问题的方式是二选一。来自不同途径的该系列的算法，将其中的变量量化成密度项。由于数据集 X 上的大部分只有很少可通过（它们中的一部分只考察数据点一次），它们是要处理的大数据集的一系列候选者。

— 子空间聚类算法（Subspace clustering algorithm）。该类算法非常适合处理高维数据集。在某些应用中，特征空间的维数甚至可以达到几千。必须面对的一个重要问题是“维

数灾”，以及必须为这样苛刻的任务配置合适的工具库。

— 基于核的方法（Kernel-based method）。该类思想的基础是在非线性支持向量机中，采用“核方法”（在第 4 章中讨论）的方式实现原始空间 X 的映射，把 X 转化为高维空间，并且探讨该工具的非线性能力。

在过去的时间里，数据库和网络技术的发展已经使得数据的采集更容易和快速，从而产生了带有多模式与/或多维数的大而复杂的数据集[Pars 04]。遇到这样巨大的数据集，例如在 Web 挖掘中，其目的是从 Web 中获取知识[Pier 03]。该领域的两个重要分支是 Web 内容挖掘（目的在于从网页内容中获取有用的知识）和 Web 使用挖掘（目的在于通过分析 Web 使用数据来发现感兴趣的使用模式）。一般而言，这种 Web 数据的规模比通常遇到的聚类应用大得多。因此，按照网页内容（Web 内容挖掘）分类网页或按照用户经常访问的页面（Web 使用挖掘）分类用户的聚类网页的任务变成了非常具有挑战性的问题。此外，在 Web 内容挖掘中，如果按照每个页面内容中的有意义词的数量来表示，那么数据空间的维数将变得非常大。

另一类需要计算资源的聚类应用例子来自生物信息领域，尤其是 DNA 序列分析。这是一个很重要，且很引人注目的科学领域，已经吸引了大量的研究工作和投资。在这类应用中，遇到的数据集的维数高达 4000[Pars 04]。

有效地处理大规模与/或高维数的数据集的需要，产生用于这类复杂任务的聚类算法研究。在这类算法中，有些被归类到前面提到的分类中；但是我们仍然选择在相关的章节中单独讨论它们，以便特别强调特点和特性。

有几本书中，如［Ande 73，Dura 74，Ever 01，Gord 99，Hart 75，Jain 88，Kauf 90 和 Spat 80］致力于聚类问题的研究。此外，还有很多关于聚类算法的研究论文也已经发表。［Jain 99］是从统计学观点来介绍聚类算法的代表。在［Hans 97］中，使用数学编程框架描述了聚类问题。在［Kola 01］中，讨论了关于空间数据库系统的聚类算法的应用。其他的研究论文有 ［Berk 02, Murt 83, Bara 99］和［Xu 05］。

除了上述所说的以外，一些论文中也给出了不同聚类方法的比较研究。例如，在［Raub 00］中，对 5 种典型聚类算法进行了比较，并讨论了它们的相对优点。在［Wei 00］中，比较了用于大型数据库的有效计算方法。

最后，也有涉及了对应用于上下文的不同聚类技术的评价。例如，来自于 DNA 序列实验的遗传表达数据的聚类应用参见［Jian 04, Made 04］，聚类技术的实验评价文章参见［Stei 00］。

12.3　顺序聚类算法

在这一节中，我们将介绍一种基本顺序算法方案（Basic Sequential Algorithmic Scheme, BSAS，推广讨论参见［Hall 67］），同时给出相应的改进方案。考虑到所有向量只参与算法一次，在这种情况下，聚类数事先未知，而是随着算法的进行生成新的聚类。

令 $d(\boldsymbol{x}, C)$表示从向量 $\boldsymbol{x}$ 到聚类 C 的距离（或不相似性），这种定义既考虑了 C 中所有的向量，也考虑了它的表达向量（参见第 11 章）。这个算法方案需要用户定义参数的不相似性阈值Θ和允许的最大聚类数q。算法的基本思想如下：由于要考虑每个新向量，根据向量到已有聚类的距离，将它分配到一个已有聚类中，或者一个新生成的聚类中。设由算法生成的聚类数为 m，那么这个算法方案可以描述如下。

基本顺序算法方案（BSAS）

● $m = 1$

- $C_m=\{\boldsymbol{x}_1\}$
- For $i=2$ to N

  —FindC_k : $d(\boldsymbol{x}_i, C_k)=\min_{1\leqslant j\leqslant m} d(\boldsymbol{x}_i, C_j)$

  —If ($d(\boldsymbol{x}_i, C_k)>\Theta$)AND($m<q$)then

  * $m=m+1$
  * $C_m=\{\boldsymbol{x}_i\}$

  —Else

  * $C_k=C_k\cup\{\boldsymbol{x}_i\}$
  *如果必要，更新向量表达[①]

  —End{if}
- End{For}

选择不同的 $d(\boldsymbol{x}, C)$会产生不同的算法，并且可以使用第 11 章介绍的任何测度。当 C 由一个单向量表达时，$d(\boldsymbol{x}, C)$变为

$$d(\boldsymbol{x}, C)=d(\boldsymbol{x}, \boldsymbol{m}_C) \tag{12.4}$$

其中 $\boldsymbol{m}_C$ 是 C 的表达。在以均值向量为表达的情况下，更新会以迭代形式出现，即

$$\boldsymbol{m}_{C_k}^{\text{new}}=\frac{(n_{C_k}^{\text{new}}-1)\boldsymbol{m}_{C_k}^{\text{old}}+\boldsymbol{x}}{n_{C_k}^{\text{new}}} \tag{12.5}$$

其中 nC_k^{new} 是将 $\boldsymbol{x}$ 分到该聚类后 C_k 的势，$\boldsymbol{m}_{C_k}^{\text{new}}$ ($\boldsymbol{m}_{C_k}^{\text{old}}$) 是将 $\boldsymbol{x}$ 分到该聚类后 C_k 的表达(参见习题 12.2)。

不难看出，这些向量在 BSAS 中的顺序非常重要。无论是聚类的数量还是聚类本身，不同的顺序会导致完全不同的聚类结果（ 参见习题 12.3)。

另一个影响聚类算法结果的重要因素是阈值Θ的选择，这个值直接影响由 BSAS 产生的聚类数量。如果Θ选得太小，会生成不必要的聚类；另一方面，如果Θ选得太大，则聚类的数量又会不够。在这两种情况下都不会生成最适合数据集的聚类数量。

如果最大允许聚类数量 q 没有限制，则让算法去决定适合的聚类数量。考虑图 12.1 的例子，由 X 点形成致密的，且很好分离的 3 个聚类。如果允许的最大聚类数量是 2，则 BSAS 算法不可能生成 3 个聚类。在这种情况下，也许最右边的两类可合并成一个聚类。另一方面，如果 q 没有限制，BSAS 算法很有可能形成 3 个聚类（ 选择合适的阈值Θ)，至少对于这个以均值向量为表达的例子是这样的。然而，对于处理的问题可用计算资源有限时，限制 q 变得很必要。在下一节中将介绍一种简单的确定聚类数的技术[②]。

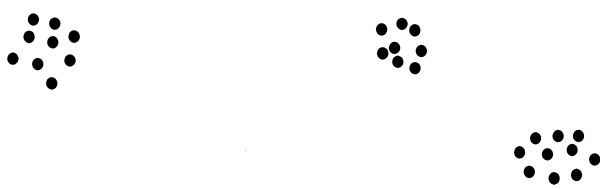

图 12.1　由特征向量生成的 3 个聚类。当 q 值小于 3 时，BSAS 算法不能生成这 3 个聚类

① 该声明在每一个聚类由单个向量表达时有效。例如，若每一个聚类由平均向量表达，每次都要更新，则新向量成为聚类的一员。

② 这个问题将在第 16 章中讨论。

注释

- BSAS 可以做适当调整：用相似性替代不相似性，即用最小运算替换最大运算。
- 这表明使用聚类表达的 BSAS 更适合于致密聚类。因此，如果能明显看出其他类型的聚类，就不要使用 BSAS。

  BSAS 算法对整个数据集 X 进行一次扫描。在每次迭代中，计算当前向量与聚类的距离。因为最后的聚类数 m 被认为是远小于 N，BSAS 的时间复杂度是 $O(N)$。
- 上述算法与 ART2（Adaptive resonance theory，自适应谐振理论）神经元结构密切相关[Carp 87, Burk 91]。

12.3.1　聚类数的估计

这一节介绍确定聚类数的简单方法（也可参见第 16 章）。该方法适用于 BSAS，也适用于其他算法，其聚类数不是输入参数。下面，BSAS(Θ)指具有给定不相似阈值Θ的 BSAS 算法。

- For $\Theta = a$ to b step c

  —算法 BSAS(Θ)执行 s 次，每一次都用不同的顺序表示数据。

  —估计聚类数，m_Θ 作为从 s 次 BSAS(Θ)算法得来的最常出现的聚类数。
- NextΘ

a 值和 b 值是 X 的所有向量对的最小和最大不相似级别，即 $a = \min_{i,j=1,\cdots,N} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$，$b = \max_{i,j=1,\cdots,N} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$。$c$ 的选择直接受 $d(\boldsymbol{x}, C)$的影响。s 值越大，统计采样就越大，因此，结果的精度就越高。接下来，画出聚类数 m_Θ 与Θ的关系图。该图有一定数量的平坦区域。我们估计聚类数将对应最宽的平坦区域。至少对于分离性很好的致密聚类情况，这个聚类数是理想的。下面直观地解释这个参数。假设数据形成两个分离性很好的致密聚类 C_1 和 C_2。C_1 (C_2) 的两个向量间的最大距离是 $r_1(r_2)$，并且假设 $r_1 < r_2$。令 $r(> r_2)$是所有距离 $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$中的最小值，这里 $\boldsymbol{x}_i \in C_1$，且 $\boldsymbol{x}_j \in C_2$。很明显，对于$\Theta \in [r_2, r - r_2]$，由 BSAS 得到的聚类数是 2。另外，如果 $r \gg r_2$，区间很大，因此在Θ与 m_Θ 关系图中对应着一个很宽的范围。例 12.2 证明了这个观点。

例 12.2　考虑均值分别是$[0,0]^{\mathrm{T}}$和$[20,20]^{\mathrm{T}}$的二维高斯分布。两个分布的协方差矩阵是$\boldsymbol{\Sigma} = 0.5\boldsymbol{I}$，其中 $\boldsymbol{I}$ 是 2×2 的单位矩阵。从每个分布生成 50 个点［参见图 12.2(a)］，隐含聚类的数量是 2。应用前面描述的方法得到图 12.2(b)，其中 $a = \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in X} d_2(\boldsymbol{x}_i, \boldsymbol{x}_j), b = \max_{\boldsymbol{x}_i, \boldsymbol{x}_j \in X} d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$，且 $c \approx 0.3$。可以看出对应聚类数 2 的区域是最宽的平坦区域，这正是隐含的聚类数量。

在上述过程中，隐含假设了特征向量确实能够组成聚类。如果不是这种情况，这种方法就没有用了。关于“是否存在聚类”的处理方法将在第 16 章中讨论。如果向量是致密聚类，但是分离得不好，这种方法将给出的结果不可靠，因为Θ 与 m_Θ 的图中不可能包含宽的平坦区域。

在有些情况下，应该考虑在Θ与 m_Θ 图中比较大的平坦区域对应的所有聚类数 m_Θ。例如，如果有三个聚类，前两个很近，而远离第三个，最平坦区域的聚类数可能是 $m_\Theta = 2$，次平坦区域的聚类数可能是 $m_\Theta = 3$。如果放弃次平坦的区域，就会丢掉三个聚类的解决方法（参见习题 12.6）。

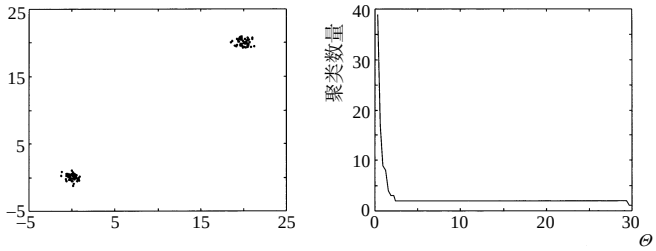

图 12.2 (a) 数据集；(b) 聚类数与Θ的对应图。可以看出对于Θ值的宽区域，聚类数 $m=2$

12.4 BSAS 的改进

已经讲过，BSAS 的基本思想是每个输入向量 $\boldsymbol{x}$ 被分配到已有聚类或新形成的聚类中。因此，对于 $\boldsymbol{x}$ 的分配是在最后的聚类形成之前决定的，而最后的聚类是在所有的向量都处理后才形成的。下面介绍克服了这个缺点的 BSAS，称为改进的 BSAS（MBSAS），这种改进的代价是 X 的向量必须参与算法两次。算法包括两个阶段，第一阶段是将 X 的一些向量分配到聚类中来形成类；在第二阶段中，没有被分配的向量第二次参与算法，并分配到合适的类中。MBSAS 具体形式如下。

改进的基本顺序算法方案（Modified Basic Sequential Algorithmic Scheme, MBSAS）

聚类的确定

- $m=1$
- $C_m=\{\boldsymbol{x}_1\}$
  —For $i=2$ to N
  —Find $C_k: d(\boldsymbol{x}_i, C_k)=\min_{1\leqslant j\leqslant m} d(\boldsymbol{x}_i, C_j)$
  —If $(d(\boldsymbol{x}_i, C_k)>\Theta)AND(m<q)$then
  * $m=m+1$
  * $C_m=\{\boldsymbol{x}_i\}$
  —End{if}
- End{For}

模式分类

- For $i=1$ to N
  —如果 $\boldsymbol{x}_i$ 没有分配到一个聚类中，那么
  * Find $C_k: d(\boldsymbol{x}_i, C_k)=\min_{1\leqslant j\leqslant m} d(\boldsymbol{x}_i, C_j)$
  * $C_k=C_k\cup\{\boldsymbol{x}_i\}$
  * 如果必要，更新向量表达
  —End{if}
- End{For}

聚类数在第一阶段决定，然后不许改变。因此，在第二个阶段，对每个向量做决定时，都要考虑所有的聚类。

当聚类的均值向量作为表达时，在每一个向量被分配到一个聚类后，用式(12.5)来调整合适的聚类表达值。

同 BSAS 情况一样，MBSAS 也对向量参与算法的顺序敏感。另外，因为 MBSAS 在数据集 X 上执行两遍（每个阶段一遍），按预期，它的速度应该慢于 BSAS。但是，它的时间复杂度应该与 BSAS 处在同一数量级上，用 $O(N)$表示。

最后必须指出，在改进之后，当使用相似性测度时，也可以使用 MBSAS（参见习题 12.7）。

另一种被归入到 MBSAS 的算法是所谓的最大最小算法[Kats 94, Juan 00]。在 MBSAS 方法中，在第一遍处理时形成一个聚类，每次已经形成的聚类到一个向量的距离比阈值大。与此相反的是，最大最小算法在第一阶段采用不同的策略。到当前的迭代步骤时，令 W 作为所有已经被选择用来形成聚类的点集合。为了形成一个新聚类，计算从 W 中的每个点到 $X-W$ 中的每个点的距离。如果 $\boldsymbol{x}\in X-W$，令 d_x 作为从 W 中的所有点到 $\boldsymbol{x}$ 的最小距离。对 $X-W$ 中的每个点都执行这个操作。然后选择最大的点（记为 $\boldsymbol{y}$），d_y 是最小距离（从 W 中的向量）；即

$$d_{\boldsymbol{y}} = \max_{\boldsymbol{x}} d_{\boldsymbol{x}},\ \boldsymbol{x} \in X - W$$

如果比阈值大，则这个向量形成一个新聚类。否则，算法的第一阶段结束。必须强调的是，与 BSAS，MSAS 相反，最大最小算法使用了一个数据相关的阈值。在第二遍处理中，还没有分配给聚类的点被分配给使用 MBSAS 方法创建的聚类。最大最小算法尽管比 MBSAS 的计算量大，但可以生成高质量的聚类。

12.5　两个阈值的顺序方法

前面已经指出，BSAS 和 MBSAS 都依赖于向量参与算法的顺序和Θ值。选择不合适的Θ可能导致无意义的聚类结果，因此解决这一困难的办法是定义一个“灰度”区域，参见[Trah 89]，通过使用两个阈值Θ_1和$\Theta_2(>\Theta_1)$来达到这个目的。如果向量 $\boldsymbol{x}$ 和最近的聚类 C 的不相似性 $d(\boldsymbol{x}, C)$ 比Θ_1 小，则 $\boldsymbol{x}$ 被分配到 C；如果 $d(\boldsymbol{x},C)>\Theta_2$，生成一个新聚类，$\boldsymbol{x}$ 分配到其中。否则，如果$\Theta_1\leqslant d(\boldsymbol{x}, C)\leqslant\Theta_2$，则存在不确定性，$\boldsymbol{x}$ 的分配在后面的阶段决定。令 clas($\boldsymbol{x}$)是一个标志，标志 $\boldsymbol{x}$ 已经分类(1)，还是没有分类(0)。用 m 表示现在已经形成的聚类数。接着，我们假设聚类数（如 $q=N$）没有限制。这个算法如下：

两个阈值的顺序算法方法（Two-Threshold Sequential Algorithmic Scheme, TTSAS）

$m = 0$
clas($\boldsymbol{x}$) = 0，$\forall \boldsymbol{x}\in X$
prev_change = 0
cur_change = 0
exists_change = 0

While（至少存在一个 clas($\boldsymbol{x}$) = 0 的特征向量 $\boldsymbol{x}$）do

- For $i = 1$ to N
  —if clas($\boldsymbol{x}_i$) = 0 AND it is the first in the new while loop AND
  exists_change = 0 then
    * $m = m + 1$
    * $C_m = \{\boldsymbol{x}_i\}$
    * clas($\boldsymbol{x}_i$) = 1

∗ cur_change = cur_change + 1

—Else if clas($\boldsymbol{x}_i$) = 0 then

∗ Find $d(\boldsymbol{x}_i, C_k) = \min_{1 \leqslant j \leqslant m} d(\boldsymbol{x}_i, C_j)$

∗ if $d(\boldsymbol{x}_i, C_k) < \Theta_1$ then

• $C_k = C_k \cup \{\boldsymbol{x}_i\}$

• clas($\boldsymbol{x}_i$) = 1

• cur_change = cur_change + 1

∗ else if $d(\boldsymbol{x}_i, C_k) > \Theta_2$ then

• $m = m + 1$

• $C_m = \{\boldsymbol{x}_i\}$

• clas($\boldsymbol{x}_i$) = 1

• cur_change = cur_change + 1

∗ End{If}

—Else if clas($\boldsymbol{x}_i$) = 1 then

∗ cur_change = cur_change + 1

—End{If}

- End{For}
- exists_change = |cur_change - prev_change|
- prev_change = cur_change
- cur_change = 0

End{While}

exists_change 核对在当前检查 X 过程中是否至少存在一个向量已经分类（如整个循环的当前迭代）。通过比较当前检查 X 已经分类的向量数 cur_change 和上次检查 X 已经分类的向量数 prev_change 来得到这个值。如果 exists_change = 0，即在最后一次检查 X 时没有向量分类，第一个没有分类的向量形成一个新聚类。

在 For 循环的 if 条件能保证在 N 次检查 X（N 为 While 循环的执行结果）以后算法结束，则第一次是最重要的。确实，当最后一次检查 X 没有向量分类时，这种条件强迫第一个没分类的向量形成一个新聚类。这给在指定循环中没有向量分配的情况找到了一个出路。

但是，在实践中，要求检查的次数少于 N。必须指出：这个算法几乎和前两个方案一样，计算代价很高，因为在通常情况下至少要求两次检查 X。此外，由于一个向量的分配要延迟到有足够的信息，所以这个算法对数据参与算法的顺序不是非常敏感。

在前面的例子中，向量和聚类之间不相似性选择的不同会导致不同的结果。当使用点聚类做表达时，这个算法也适用于致密聚类。

注释

- 注意到对于所有的算法都没有死锁状态。也就是说，没有一个算法会进入到这样一个状态，即无法将向量分配到已经存在的聚类中，也不能生成新聚类。BSAS 和 MBSAS 方案都保证：在分别对 X 经过一次和两次检查后结束。在整个 X 检查中没有向量分类发生时，强行将第一个没有分类的向量指定到新聚类中，这样在 TTSAS 中也避免了死锁情况。

例 12.3 考虑向量 $\boldsymbol{x}_1 = [2,5]^{\mathrm{T}}$, $\boldsymbol{x}_2 = [6,4]^{\mathrm{T}}$, $\boldsymbol{x}_3 = [5,3]^{\mathrm{T}}$, $\boldsymbol{x}_4 = [2,2]^{\mathrm{T}}$, $\boldsymbol{x}_5 = [1,4]^{\mathrm{T}}$, $\boldsymbol{x}_6 = [5,2]^{\mathrm{T}}$, $\boldsymbol{x}_7 = [3,3]^{\mathrm{T}}$,

$\boldsymbol{x}_8=[2,3]^{\mathrm{T}}$。从 $\boldsymbol{x}$ 到聚类 C 的距离是 $\boldsymbol{x}$ 和 C 平均向量之间的欧几里得距离。如果按照上面的顺序使用 MBSAS 算法，并令 $\Theta=2.5$，就会得到三个聚类 $C_1=\{\boldsymbol{x}_1, \boldsymbol{x}_5, \boldsymbol{x}_7, \boldsymbol{x}_8\}$, $C_2=\{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_6\}$ 和 $C_3=\{\boldsymbol{x}_4\}$，参见图 12.3(a)。

另一方面，如果把向量按照上面的顺序使用 TTSAS 算法，令 $\Theta_1=2.2$，$\Theta_2=4$，可以得到图 12.3(b) 所示的两个聚类 $C_1=\{\boldsymbol{x}_1, \boldsymbol{x}_5, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_4\}$，$C_2=\{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_6\}$。这种情况下，除了 $\boldsymbol{x}_4$ 外，所有的向量都在第一次检查 X 时就都分配给了聚类。在第二次检查 X 时，$\boldsymbol{x}_4$ 被赋给 C_1。每次检查 X，至少会有一个向量被赋给聚类。因此，没有向量被强行赋给一个新聚类。

很显然，最后的算法给出了比 MBSAS 更合理的结果。然而，需要注意的是，MBSAS 也可能得到相同的聚类，例如，如果向量按照顺序 $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_5, \boldsymbol{x}_3, \boldsymbol{x}_8, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_4$ 排列。

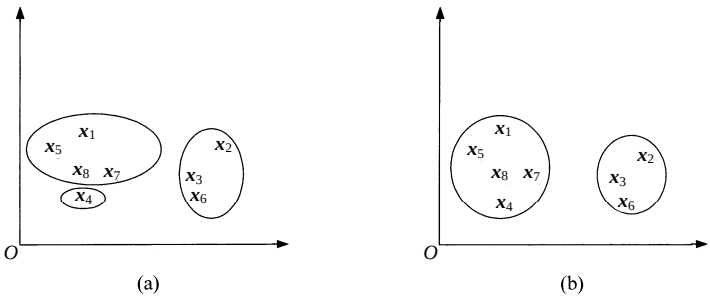

图 12.3　(a)MBSAS 生成的聚类；(b)TTSAS 生成的聚类

12.6　改进阶段

在所有上述算法中，可能出现两个聚类的位置离得很近的情况，把它们合并成一类是最理想的。对这种情况，上述所有算法都无能为力。解决这种问题的方法是在上述过程结束后，执行下面的合并过程（参见［Fu 93］）。

合并过程

- (A)找到 $C_i, C_j (i<j)$，使 $d(C_i, C_j)=\min_{k,r=1,\cdots,m,k\neq r} d(C_k, C_r)$
- If $d(C_i, C_j)\leqslant M_1$ then
  —把 C_i, C_j 合并为 C_i，并删除 C_j
  —更新 C_i 的聚类表达（如果使用聚类表达）
  —分别将聚类 $C_{j+1},\cdots,C_m$ 命名为 $C_j,\cdots,C_{m-1}$
  —$m=m-1$
  —Go to(A)
- Else
  —Stop
- End {If}

M_1 是用户定义的参数，用来量化 C_i 和 C_j 两个聚类之间的接近程度。聚类之间的不相似性 $d(C_i, C_j)$ 可以用第 11 章中给出的定义。

顺序算法的另一个缺点是对向量表达顺序的敏感性。例如，使用 BSAS 时，$\boldsymbol{x}_2$ 被赋给第一个聚类 C_1，算法结束后，形成 4 个聚类。可是 $\boldsymbol{x}_2$ 有可能更接近不同于 C_1 的另外一个聚类，然而，对于 $\boldsymbol{x}_2$，一旦赋给一个聚类，就没有办法将它移到另一个离它更近的聚类。解决这个问题的最简单的办法是用下面的重分配过程：

重分配过程

- For $i = 1$ to N
  —找到 C_j 使得 $d(\boldsymbol{x}_i, C_j) = \min_{k=1,\cdots,m} d(\boldsymbol{x}_i, C_k)$
  —令 $b(i) = j$
- End {For}
- For $j = 1$ to m
  —令 $C_j = \{\boldsymbol{x}_i \in X : b(i) = j\}$
  —更新表达（如果使用）
- End {For}

在这个程序中，$b(i)$指示离 $\boldsymbol{x}_i$ 最近的聚类。这个过程可在算法结束后应用，如果需要合并过程，就要在合并过程结束之后使用。

在［MacQ 67］中介绍了包含两个改进过程的 BSAS 算法，但它只考虑了使用点表达的情况。根据这个算法，我们从 $m > 1$ 聚类开始，而不是从一个聚类开始，每一个聚类包含 X 中前 m 个向量中的一个。应用合并过程，并将剩下的向量用于算法中，将当前的向量赋给一个聚类，并更新它的表达之后，再执行合并过程。如果 $\boldsymbol{x}_i$ 和离它最近的聚类之间的距离大于一个预先确定的阈值，就会产生一个仅包含 $\boldsymbol{x}_i$ 的新聚类。最后，所有的向量都用于算法之后，再进行一次重分配过程。合并程序被执行 $N - m + 1$ 次。最后，在［Ande 73］中给出了改进算法。

在［Mant 85］中讨论了只需要检查 X 一次的不同顺序算法。更特殊的是，它假设向量是由 k 个混合高斯概率密度 $p(\boldsymbol{x}|C_i)$生成的，即

$$p(\boldsymbol{x}) = \sum_{j=1}^{k} P(C_j) p(\boldsymbol{x}|C_j; \mu_j, \Sigma_j) \tag{12.6}$$

其中μ_j 和Σ_j 是第 j 个高斯分布的均值和协方差矩阵，$P(C_j)$是 C_j 的先验概率。为了方便，假设所有的 $P(C_j)$ 都相等，由算法形成的聚类假设服从高斯分布。开始，第一个向量形成一个单独的聚类。然后，对于每一个新分配来的向量 $\boldsymbol{x}_i$，第 m 个聚类的均值向量和协方差矩阵进行适当的更新，并且估计条件概率 $P(C_j|\boldsymbol{x}_i)$。如果 $P(C_q|\boldsymbol{x}_i) = \max_{j=1,\cdots,m} P(C_j|\boldsymbol{x}_i)$比预先确定的阈值 a 大，则 $\boldsymbol{x}_i$ 被分配给 C_q。否则，就要创建一个包括 $\boldsymbol{x}_i$ 的新聚类。在［Amad 05］中描述了另一种顺序聚类方法所使用的统计工具。

12.7 神经网络的实现

在这一节中将介绍神经网络的体系结构，并将其应用到 BSAS 中。

12.7.1 结构描述

结构如图 12.4(a)所示。它包括了两个模块，即匹配记分生成器（MSG）和 MaxNet 网络（MN）[①]。

第一个模块包括 q 个 $l \times 1$ 维的参数向量 $\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_q$[②]和一个实现函数 $f(\boldsymbol{x}, \boldsymbol{w})$，这个函数表示了 $\boldsymbol{x}$ 和 $\boldsymbol{w}$ 之间的相似性。$f(\boldsymbol{x}, \boldsymbol{w})$的值越大，$\boldsymbol{x}$ 和 $\boldsymbol{w}$ 就越相似。

① 这是［Lipp 87］介绍的 Hamming 网络的推广。
② 也称为样本模式。

当向量 $\boldsymbol{x}$ 输入网络时，MSG 模块输出一个 $q \times 1$ 的向量 $\boldsymbol{\upsilon}$，它的第 i 个坐标等于 $f(\boldsymbol{x},\boldsymbol{w}_i), i = 1,\cdots,q$。

第二个模块把向量 $\boldsymbol{\upsilon}$ 作为输入，并确认它的最大坐标。它的输出是一个 $q \times 1$ 的向量 $\boldsymbol{s}$，$\boldsymbol{s}$ 的所有元素除了对应 $\boldsymbol{\upsilon}$ 的最大坐标处为 1，其他都为 0。这种类型的模块大多数都需要 $\boldsymbol{\upsilon}$ 至少有一个坐标为正。

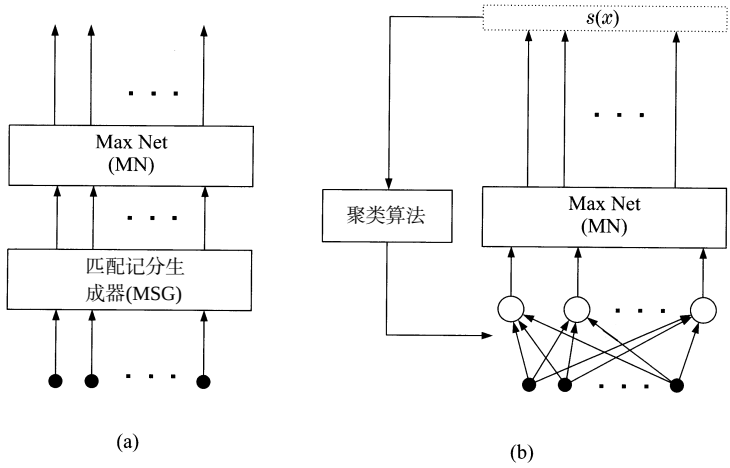

图 12.4　(a)神经结构；(b)当每一个聚类用均值向量表达并且使用两个向量间的欧几里得距离时，BSAS 算法的实现

使用不同的近邻测度，MSG 可以有不同的实现方法。例如，函数 f 是内积，则 MSG 包含 q 个线性节点，并且阈值是 0。每一个点都和一个参数向量 i 对应，它的输出是 $\boldsymbol{x}$ 和 $\boldsymbol{w}_i$ 的内积。

如果使用欧几里得距离，MSG 模块也包括 q 个线性节点。然而，需要不同的设置过程。和第 i 个节点相关的权向量是 $\boldsymbol{w}_i$，它的阈值等于 $T_i = \frac{1}{2}(Q - \|\boldsymbol{w}_i\|^2)$，其中 Q 是正常数，它保证了第一层节点中至少有一个能输出正的匹配分数，并且 $\|\boldsymbol{w}_i\|$ 是 $\boldsymbol{w}_i$ 的欧几里得范数。因此，节点输出是

$$f(\boldsymbol{x}, \boldsymbol{w}_i) = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}_i + \frac{1}{2}(Q - \|\boldsymbol{w}_i\|^2) \qquad (12.7)$$

很容易证明 $d_2(\boldsymbol{x},\boldsymbol{w}_i) < d_2(\boldsymbol{x},\boldsymbol{w}_j)$ 等价于 $f(\boldsymbol{x},\boldsymbol{w}_i) > f(\boldsymbol{x},\boldsymbol{w}_j)$，因此，MSG 的输出对应着与 $\boldsymbol{x}$ 具有最小欧几里得距离的 $\boldsymbol{w}_i$（参见习题 12.8）。

MN 模块可以通过其他的方法实现，如 Hamming MaxNet 神经网络比较器、其他前馈结构[Lipp 87, Kout 95, Kout 05, Kout 98]或传统的比较器[Mano 79]。

12.7.2　BSAS 算法的实现

在这一节中，当(a)每个聚类都是由它的均值向量表达的；(b)使用两个向量间的欧几里得距离时，我们将说明 BSAS 算法是如何映射到神经网络结构中的，参见图 12.4(b)。Hamming 网络的结构必须稍稍地修改一下，使得第一层的每一个节点都有一个额外的输入项 $\frac{1}{2}\|\boldsymbol{x}\|^2$。令 $\boldsymbol{w}_i$ 和 T_i 分别是 MSG 模型中的第 i 个节点的权向量和阈值。令 $\boldsymbol{a}$ 是 $q \times 1$ 的向量，它的第 i 个成分表示第 i 个聚类中向量的个数。当网络的输入是 $\boldsymbol{x}$ 时，令 $\boldsymbol{s}(\boldsymbol{x})$ 是 MN 模块的输出。另外，令 t_i 是 MSG 的第 i

个节点和 MN 中对应的节点之间的链接。最后，令 sgn(z)是阶跃函数，当 $z>0$ 时为 1，否则为 0。

q 个 $\boldsymbol{w}_i$ 的前 m 个对应聚类的表达由算法定义。在每一次迭代过程中，无论什么时候生成新聚类，前 m 个 $\boldsymbol{w}_i$ 或者有一个进行更新，或者使用一个新参数向量 $\boldsymbol{w}_{m+1}$（如果 $m<q$）。算法描述如下：

- 初始化
  - —$a=0$
  - —$\boldsymbol{w}_i=0, i=1,\cdots,q$
  - —$t_i=0, i=1,\cdots,q$
  - —$m=1$
  - —对于第一个向量 $\boldsymbol{x}_1$
    - *$\boldsymbol{w}_i=\boldsymbol{x}_1$
    - * $a_1=1$
    - * $t_1=1$
- 主程序
  - —重复
    - * 将下一个向量 $\boldsymbol{x}$ 输入到网络
    - * 计算输出向量 $\boldsymbol{s}(\boldsymbol{x})$
    - * $\mathrm{GATE}(\boldsymbol{x})=\mathrm{AND}((1-\sum_{j=1}^{q}(s_j(\boldsymbol{x}))),\mathrm{sgn}(q-m))$
    - * $m=m+\mathrm{GATE}(\boldsymbol{x})$
    - * $a_m=a_m+\mathrm{GATE}(\boldsymbol{x})$
    - * $\boldsymbol{w}_m=\boldsymbol{w}_m+\mathrm{GATE}(\boldsymbol{x})\,\boldsymbol{x}$
    - * $T_m=\Theta-\frac{1}{2}\|\boldsymbol{w}_m\|^2$
    - * $t_m=1$
    - * For $j=1$ to m
      - • $a_j=a_j+(1-\mathrm{GATE}(\boldsymbol{x}))\,s_j(\boldsymbol{x})$
      - • $\boldsymbol{w}_j=\boldsymbol{w}_j-(1-\mathrm{GATE}(\boldsymbol{x}))\,s_j(\boldsymbol{x})(\frac{1}{a_j}(\boldsymbol{w}_j-\boldsymbol{x}))$
      - • $T_j=\Theta-\frac{1}{2}\|\boldsymbol{w}_j\|^2$
    - * Next j
  - —直到所有的向量都输入到网络中

注意只考虑了 MSG 模型的前 m 个节点的输出，因为只有它们才和聚类相对应。因为 $t_k=0, k=m+1,\cdots,q$，所以剩余的节点输出都不考虑。假设一个新向量输入网络，使 $\min_{1\leqslant j\leqslant m}d(\boldsymbol{x},\boldsymbol{w}_j)>\Theta$ 且 $m<q$，则 $\mathrm{GATE}(\boldsymbol{x})=1$。因此，创建一个新的聚类，并且激活下一个节点。既然 $1-\mathrm{GATE}(\boldsymbol{x})=0$，For 循环内的指令不影响网络的任何参数。

假设下一步 $\mathrm{GATE}(\boldsymbol{x})=0$。这等价于 $\min_{1\leqslant j\leqslant m}d(\boldsymbol{x},\boldsymbol{w}_j)\leqslant\Theta$，或者没有更多的节点可以用来表达其他的聚类。然后在 For 循环内的指令执行使权向量和满足 $\min_{1\leqslant j\leqslant m}d(\boldsymbol{x},\boldsymbol{w}_j)$ 的节点 k 的阈值更新。这是因为 $s_k(\boldsymbol{x})=1$，并且 $s_j(\boldsymbol{x})=0$，$j=1,\cdots,q, j\neq k$。

习题

12.1　用归纳法证明式(12.3)。

12.2　证明式(12.5)。

12.3　这个问题主要是说明在BSAS和MBSAS算法中的向量分配顺序的影响。考虑下列二维向量：$\boldsymbol{x}_1=[1,1]^{\mathrm{T}}$，$\boldsymbol{x}_2=[1,2]^{\mathrm{T}}$，$\boldsymbol{x}_3=[2,2]^{\mathrm{T}}$，$\boldsymbol{x}_4=[2,3]^{\mathrm{T}}$，$\boldsymbol{x}_5=[3,3]^{\mathrm{T}}$，$\boldsymbol{x}_6=[3,4]^{\mathrm{T}}$，$\boldsymbol{x}_7=[4,4]^{\mathrm{T}}$，$\boldsymbol{x}_8=[4,5]^{\mathrm{T}}$，$\boldsymbol{x}_9=[5,5]^{\mathrm{T}}$，$\boldsymbol{x}_{10}=[5,6]^{\mathrm{T}}$，$\boldsymbol{x}_{11}=[-4,5]^{\mathrm{T}}$，$\boldsymbol{x}_{12}=[-3,5]^{\mathrm{T}}$，$\boldsymbol{x}_{13}=[-4,4]^{\mathrm{T}}$，$\boldsymbol{x}_{14}=[-3,4]^{\mathrm{T}}$。并考虑每个聚类都由其均值向量表达的情形。

(a) 当向量以给定的顺序给出时，运行 BSAS 和 MBSAS 算法。用两个向量间的欧几里得距离，并令 $\Theta=\sqrt{2}$ 。

(b) 改变分配顺序为 $\boldsymbol{x}_1, \boldsymbol{x}_{10}, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_{13}, \boldsymbol{x}_8, \boldsymbol{x}_{14}, \boldsymbol{x}_9$，重新运行算法。

(c) 按照下列分配顺序运行算法：$\boldsymbol{x}_1, \boldsymbol{x}_{10}, \boldsymbol{x}_5, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, \boldsymbol{x}_4, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_{13}, \boldsymbol{x}_{14}, \boldsymbol{x}_8, \boldsymbol{x}_9$。

(d) 画出给定的向量，并讨论结果。

(e) 完成数据的聚类。给定的向量可以产生多少聚类?

12.4　考虑例 12.2 的数据。当$\Theta=5$ 时，运行 BSAS 和 MBSAS 算法，每个聚类都是以均值向量表达的。讨论结果。

12.5　考虑图 12.5，内部正方形的边 $S_1=0.3$，外部正方形框的内边和外边分别是 $S_2=1, S_3=1.3$。内部正方形中包括 50 个点，它们是从正方形的均匀分布中得来的。类似地，外部框也包括 50 个均匀分布的点。

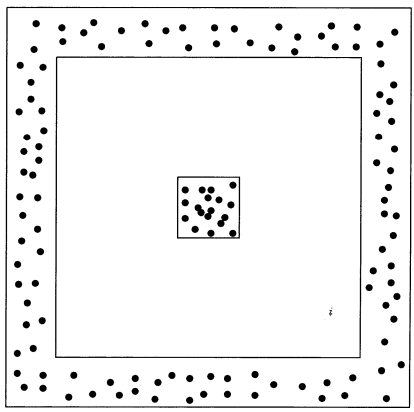

图 12.5　习题 12.5 的数据

(a) 完成数据的聚类。给定的点可以产生多少聚类?

(b) 当每个聚类都是以均值向量和两个向量间的欧几里得距离表达时，运行BSAS和MBSAS算法，这时

$$\Theta=\min_{i,j=1,\cdots,100} d(\boldsymbol{x}_i,\boldsymbol{x}_j),\quad \text{至}\quad \max_{i,j=1,\cdots,100} d(\boldsymbol{x}_i,\boldsymbol{x}_j)\text{，步长为 }0.2$$

并且用随机的数据顺序。给出结果的量化解释。与前一个问题的结果进行比较。

(c) 在 $d_{\min}^{ps}$ 被当做向量和聚类之间的非相似性时，重复步骤(b)（参见 11 章）。

12.6　考虑三个均值为$[0,0]^{\mathrm{T}}$,$[6,0]^{\mathrm{T}}$ 和$[12,6]^{\mathrm{T}}$ 的二维高斯分布，每个分布的协方差矩阵等于单位矩阵 $\boldsymbol{I}$。为每一个分布产生 30 个点，并且令 X 为结果数据集。当 $a=\min_{i,\,j=1,\cdots,100} d(\boldsymbol{x}_i,\boldsymbol{x}_j)$, $b=\max_{i,}$

$_{j=1,\cdots,100}d(\boldsymbol{x}_i, \boldsymbol{x}_j)$并且 $c = 0.3$ 时，应用欧几里得距离和在 12.3.1 节中对 X 聚类数估计的步骤，画出Θ与 m 的关系图，并得出你的结论。

12.7 令 s 是向量和聚类之间的相似性测度，根据 s 表示 BSAS，MBSAS 和 TTSAS 算法。

12.8 当用向量之间的欧几里得距离，并且 MSG 模块的输出函数由式(12.7)给出时，证明 $d_2(\boldsymbol{x},\boldsymbol{w}_1) < d_2(\boldsymbol{x},\boldsymbol{w}_2)$等价于 $f(\boldsymbol{x},\boldsymbol{w}_1) > f(\boldsymbol{x},\boldsymbol{w}_2)$。

12.9 当每个聚类都由分配给它的第一个向量表示时，对于 BSAS 算法，用类似于 12.7 节中给定的一种方法描述它的神经网络实现过程。

12.10 如果使用均值向量，MBSAS 算法实现的神经网络结构和图 12.4(b)中使用的欧几里得距离相类似。如果使用均值向量，编写一个算法，它类似于 12.7 节中给定的 MBSAS 算法，指出两种实现方法的明显不同之处。

MATLAB 编程和练习

上机编程

12.1 MBSAS 算法。编写出一个名为 MBSAS 的 MATLAB 函数，用它来实现 MBSAS 算法。函数的输入有：(a)一个 $l \times N$ 维的矩阵，它的第 i 列是第 i 个数据向量；(b)参数 theta（它对应文中的Θ）；(c)允许的聚类 q 的最大数量；(d)一个 N 维的行数组，叫做排序（order），它定义了 X 的向量对算法的显示顺序。比如说，如果有一个排序 =[3 4 1 2]，第三个向量会第一个显示，第四个向量会第二个显示，等等。如果排序 =[ ]，那么就不会发生重排序。函数的输出是：(a)一个 N 维的行向量 bel，它的第 i 个元素包括聚类的标识符，这个聚类里数据向量的显示顺序“i”已经分配好了（一个聚类的标识符是{1,2,⋯,n_clust}中的一个整数，其中 n_clust 是聚类的数量）；(b)一个 $l \times n$_clust 矩阵 $\boldsymbol{m}$，它的第 i 行代表第 i 个聚类。使用欧几里得距离来测量两个向量之间的距离。

解：

在下面代码中，不要键入星号（*）。稍后它们将被用做引用目的。

```
function[bel,m]=MBSAS(X,theta,q,order)
  %Ordering the data
  [l,N]=size(X);
  if(length(order)==N)
    X1=[];
    for i=1:N
      X1=[X1 X(:,order(i))];
    end
    X=X1;
    clear X1
  end
  %Cluster determination phase
  n_clust=1;%no.of clusters
  [l,N]=size(X);
  bel=zeros(1,N);
  bel(1)=n_clust;
  m=X(:,1);
  for i=2:N
```

```
  [m1,m2]=size(m);
  %Determining the closest cluster representative
  [s1,s2]=min(sqrt(sum((m-X(:,i)*ones(1,m2)),^2)));
  if(s1>theta)&&(n_clust<q)
    n_clust=n_clust+1;
    bel(i)=n_clust;
    m=[mX(:,i)];
  end(*1)
end(*2)
[m1,m2]=size(m);(*3)
%Pattern classification phase(*4)
for i=1:N(*5)
  if(bel(i)==0)(*6)
    %Determining the closest cluster representative(*7)
    [s1,s2]=min(sqrt(sum((m-X(:,i)*ones(1,m2)).^2)));(*8)
    bel(i)=s2;
    m(:,s2)=((sum[bel==s2]-1)*m(:,s2)+X(:,i))/sum(bel==s2);
  end
end
```

12.2 BSAS 算法。编写一个名为 BSAS 的 MATLAB 函数，它可以实现 BSAS 算法。它的输入输出与 MBSAS 函数中定义一样。

解：

在给出的 MBSAS 代码中，用命令 else 代替带有(*1)的行，并且删除所有其他带有*的行。

上机实验

12.1 考虑数据集 $X=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4,\boldsymbol{x}_5,\boldsymbol{x}_6,\boldsymbol{x}_7,\boldsymbol{x}_8\}$，其中 $\boldsymbol{x}_1=[2,5]^{\mathrm{T}}$，$\boldsymbol{x}_2=[8,4]^{\mathrm{T}}$，$\boldsymbol{x}_3=[7,3]^{\mathrm{T}}$，$\boldsymbol{x}_4=[2,2]^{\mathrm{T}}$，$\boldsymbol{x}_5=[1,4]^{\mathrm{T}}$，$\boldsymbol{x}_6=[7,2]^{\mathrm{T}}$，$\boldsymbol{x}_7=[3,3]^{\mathrm{T}}$，$\boldsymbol{x}_8=[2,3]^{\mathrm{T}}$，画出数据向量。

12.2 运行 MBSAS，令 $q=5$，在以上数据集上有

(a)order =[1,5,8,4,7,3,6,2],theta $=\sqrt{2}+0.001$

(b)order =[5,8,1,4,7,2,3,6],theta $=\sqrt{2}+0.001$

(c)order =[1,4,5,7,8,2,3,6],theta = 2.5

(d)order =[1,8,4,7,5,2,3,6],theta = 2.5

(e)与(c)排序相同，并且 theta=3

(f)与(d)排序相同，并且 theta=3.

仔细研究结果，并得出你的结论。

12.3 用 BSAS 重复上机实验 12.2。

参考文献

[Amad 05] Amador J.J. "Sequential clustering by statistical methodology," *Pattern Recognition Letters*, Vol. 26, pp. 2152–2163, 2005.

[Ande 73] Anderberg M.R. *Cluster Analysis for Applications*, Academic Press, 1973.

[Ball 65] Ball G.H. "Data analysis in social sciences," *Proceedings FJCC*, Las Vegas, 1965.

[Bara 99] Baraldi A., Blonda P. "A survey of fuzzy clustering algorithms for pattern recognition, Parts I and II," *IEEE Transactions on Systems, Man and Cybernetics, B. Cybernetics*,

Vol. 29(6), pp. 778-801, 1999.

[Bara 99a] Baraldi A., Schenato L. "Soft-to-hard model transition in clustering: a review," Technical Report TR-99-010, 1999.

[Berk 02] Berkhin P. "Survey of clustering data mining techniques," *Technical Report*, Accrue Software Inc., 2002.

[Burk 91] Burke L.I. "Clustering characterization of adaptive reasonance," *Neural Networks*, Vol. 4, pp. 485-491, 1991.

[Carp 87] Carpenter G.A., Grossberg S. "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, Vol. 26, pp. 4919-4930, 1987.

[Duda 01] Duda R.O., Hart P., Stork D. *Pattern Classification*, 2nd ed., John Wiley & Sons, 2001.

[Dura 74] Duran B., Odell P. *Cluster Analysis: A Survey*, Springer-Verlag, Berlin, 1974.

[Ever 01] Everitt B., Landau S., Leesse M. *Cluster Analysis*, Arnold, London, 2001.

[Flor 91] Floreen P. "The convergence of the Hamming memory networks," *IEEE Transactions on Neural Networks*, Vol. 2(4), pp. 449-459, July 1991.

[Fu 93] Fu L., Yang M., Braylan R., Benson N. "Real-time adaptive clustering of flow cytometric data," *Pattern Recognition*, Vol. 26(2), pp. 365-373, 1993.

[Gord 99] Gordon A. *Classification*, 2nd ed., Chapman & Hall, London, 1999.

[Hall 67] Hall A.V. "Methods for demonstrating resemblance in taxonomy and ecology," *Nature*, Vol. 214, pp. 830-831, 1967.

[Hans 97] Hansen P., Jaumard B. "Cluster analysis and mathematical programming," *Mathematical Programming*, Vol. 79, pp. 191-215, 1997.

[Hart 75] Hartigan J. *Clustering Algorithms*, John Wiley & Sons, 1975.

[Jain 88] Jain A.K., Dubes R.C. *Algorithms for Clustering Data*, Prentice Hall, 1988.

[Jain 99] Jain A., Muthy M., Flynn P. "Data clustering: A review," *ACM Computational Surveys*, Vol. 31(3), pp. 264-323, 1999.

[Jian 04] Jiang D., Tang C., Zhang A. "Cluster analysis for gene expression data: A survey," *IEEE Transactions on Knowledge Data Engineering*, Vol. 16(11), pp. 1370-1386, 2004.

[Juan 00] Juan A., Vidal E. "Comparison of four initialization techniques for the *k*-medians clustering algorithm," *Proceedings of Joint IAPR International Workshops SSPR2000 and SPR2000, Lecture Notes in Computer Science*, Vol. 1876, pp. 842-852, Springer-Verlag, Alacant (Spain), September 2000.

[Kats 94] Katsavounidis I., Jay Kuo C.-C., Zhang Z., "A new initialization technique for generalized Lloyd iteration," *IEEE Signal Processing Letters*, Vol. 1(10), pp. 144-146, 1994.

[Kauf 90] Kaufman L., Roussseeuw P. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[Kola 01] Kolatch E. "Clustering algorithms for spatial databases: A survey," available at *http://citeseer.nj.nec.com/436843.html*.

[Kout 95] Koutroumbas K. "Hamming neural networks, architecture design and applications," Ph.D. thesis, Department of Informatics, University of Athens, 1995.

[Kout 94] Koutroumbas K., Kalouptsidis N. "Qualitative analysis of the parallel and asynchronous modes of the Hamming network," *IEEE Transactions on Neural Networks*, Vol. 5(3), pp. 380-391, May 1994.

[Kout 98] Koutroumbas K., Kalouptsidis N. "Neural network architectures for selecting the maximum input," *International Journal of Computer Mathematics*, Vol. 68(1-2), 1998.

[Kout 05] Koutroumbas K., Kalouptsidis N., "Generalized Hamming Networks and Applications," *Neural Networks*, Vol. 18, pp. 896-913, 2005.

[Lipp 87] Lippmann R.P. "An introduction to computing with neural nets," *IEEE ASSP Magazine*, Vol. 4(2), April 1987.

[Liu 68] Liu C.L. *Introduction to Combinatorial Mathematics*, McGraw-Hill, 1968.

[Lloy 82] Lloyd S.P. "Least squares quantization in PCM," *IEEE Transactions on Information*

Theory, Vol. 28(2), pp. 129-137, March 1982.

[MacQ 67] MacQuenn J.B. "Some methods for classification and analysis of multivariate observations," *Proceedings of the Symposium on Mathematical Statistics and Probability*, 5th ed., Vol. 1, pp. 281-297, AD 669871, University of California Press, Berkeley, 1967.

[Made 04] Madeira S.C., Oliveira A.L. "Biclustering algorithms for biological data analysis: A survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 1(1), pp. 24-45, 2004.

[Mano 79] Mano M. *Digital Logic and Computer Design*, Prentice Hall, 1979.

[Mant 85] Mantaras R.L., Aguilar-Martin J. "Self-learning pattern classification using a sequential clustering technique," *Pattern Recognition*, Vol. 18(3/4), pp. 271-277, 1985.

[Murt 83] Murtagh F. "A survey of recent advanced in hierarchical clustering algorithms," *Journal of Computation*, Vol. 26(4), pp. 354-359, 1983.

[Pars 04] Parsons L., Haque E., Liu H. "Subspace clustering for high dimensional data: A review," *ACM SIGKDD Explorations Newsletter*, Vol. 6(1), pp. 90-105, 2004.

[Pier 03] Pierrakos D., Paliouras G., Papatheodorou C., Spyropoulos C.D. "Web usage mining as a tool for personalization: A survey," *User Modelling and User-Adapted Interaction*, Vol. 13(4), pp. 311-372, 2003.

[Raub 00] Rauber A., Paralic J., Pampalk E. "Empirical evaluation of clustering algorithms," *Journal of Inf. Org. Sci.*, Vol. 24(2), pp. 195-209, 2000.

[Sebe 62] Sebestyen G.S. "Pattern recognition by an adaptive process of sample set construction," *IRE Transactions on Information Theory*, Vol. 8(5), pp. S82-S91, 1962.

[Snea 73] Sneath P.H.A., Sokal R.R. *Numerical Taxonomy*, W.H. Freeman, 1973.

[Spat 80] Spath H. *Cluster Analysis Algorithms*, Ellis Horwood, 1980.

[Stei 00] Steinbach M., Karypis G., Kumar V. "A comparison of document clustering techniques," *Technical Report*, 00-034, University of Minnesota, Minneapolis, 2000.

[Trah 89] Trahanias P., Scordalakis E. "An efficient sequential clustering method," *Pattern Recognition*, Vol. 22(4), pp. 449-453, 1989.

[Wei 00] Wei C., Lee Y., Hsu C. "Empirical comparison of fast clustering algorithms for large data sets," *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pp. 1-10, Maui, HI, 2000.

[Xu 05] Xu R., Wunsch D. "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, Vol. 16(3), pp. 645-678, 2005.