

# 关于粗糙集和邻域粗糙集的基本理论和程序算例（第二版）

索子

Email: [buaasuozi@126.com](mailto:buaasuozi@126.com)

QQ:379786867

第一版本发布时间：2013-07-06

第二版本发布时间：2014-04-30

## 第二版本声明：

在此首先感谢大家对我的支持和理解，第一版本发布之后，收到了不少网友的反馈意见和建议，在此谢谢大家！

其次，需要声明一下，在第一版本的算例中，我算错了一些地方，可能给大家带来了误导，在此向大家郑重道歉！版本一中的错误我在版本二中做了修正，希望大家在版本一的基础上，以版本二为主来学习和研究邻域粗糙集理论。

版本二区别版本一的另外一个重要改进：鉴于很多网友需求邻域粗糙集的计算，但又苦于不擅长编程，所以我重新编写了邻域粗糙集的相关计算程序，供大家使用，大家也可以根据实际需求，在我给出的程序上做进一步的修改和扩展。代码注释率应该超过了 60%，估计大家阅读起来不会存在大问题。在版本二中，对我编写的邻域粗糙集计算程序做了调用的说明，大家按照我的说明调用使用就可以了。

祝大家五一节快乐！

索子

2014-04-30

## 前言：

本人在做毕业论文的时候用到了粗糙集和邻域粗糙集的相关知识，当初在网上找了一些资料，但是还是感觉很模糊，最后通过各种渠道，包括搜集资料，翻阅书刊，最终才把粗糙集和邻域粗糙集研究透彻了一些。可能我在学习粗糙集之前询问过很多人，留下了 qq 号，所以后来就有好多网友加我好友，向我询问

一些关于粗糙集的东西。其实吧，我感觉粗糙集的知识没那么难，只是那些理论说的高深了一些，如果通俗点理解，是没那么难的。为了能一并地帮助大家理解，省的我一个个针对性地讲解，在这里写下这篇文章。

现在毕业了，我在这里发布一些关于邻域粗糙集和粗糙集的基础性知识，供大家交流学习。说明一下，我研究的也不是很透彻，肯定会存在不足的地方，希望大家提出，我会进一步整改。欢迎交流学习！本人 qq: 379786867。关于加 qq 的那些提问，我想你能理解怎么去填写。

**关于参考资料：**

[Pawlak Z. Rough sets: theoretical aspects of reasoning about data. Dordrecht: Kluwer Academic Publisher, 1991.](#)（这个是鼻祖）

[苗夺谦, 李道国. 粗糙集理论、算法与应用\[M\]. 北京: 清华大学出版社, 2008.](#)（这个是牛人，国内粗糙集的大牛）

[胡清华, 于达仁, 谢宗霞. 基于邻域粒化和粗糙逼近的数值属性约简\[J\]. 软件学报. 2008.](#)（这个是邻域粗糙集研究的大牛，我这里用到了很多相关知识和程序，在此向哈工大胡清华老师表示感谢！）

建议大家多看看苗夺谦的相关书籍，比较有帮助。还有张文修的，也不错，各有不同吧！邻域粗糙集的看看胡清华的，但是在看胡清华的文章之前，最好已经具备了粗糙集的一些基础知识，否则比较难理解。

**关于软件和程序：**

软件知名的就是 rosetta 了，比较强大，网上有相关教程，我这里不多说这个。

程序有经典粗糙集的和邻域粗糙集的，经典的就不说了，网上也有。邻域的就是胡清华编写的了，很不错的 matlab 程序。

## 1 粗糙集和邻域粗糙集

邻域粗糙集是建立在经典粗糙集的基础上的，经典粗糙集是波兰华沙理工大学教授帕拉克（Z.Pawlak）在 1982 年，研究不完整数据和不精确知识的表达运用中提出的。粗糙集具体来说能做什么？有哪些优点？

1、属于数据挖掘领域的一个基础性理论，能客观地挖掘出数据内部的信息。

(就是找到数据的本质信息)

2、将数据通过一定的原则进行提炼，得出数据的内涵信息（说白了，就是减肥）

上述的作用，也就是粗糙集的规则提取和数据约简。

## 1.1 粗糙集的基本概念

粗糙集的一些基本概念就是上近似、下近似、正域、重要度和依赖度。

先列举一些看似很神秘实际上什么都不是的概念：

**定义 1.1** (关系)  $A \times B$  的任一子集为从集合  $A$  到集合  $B$  的一个二元关系，记为  $R$ ，即  $R \subseteq A \times B$ 。

**通俗理解：**这里说的就是数据集合的关系，就是说一个集合的一堆数对应地和另一个集合的一堆数有点关系。

**定义 1.2** (论域，知识库) 将我们感兴趣的讨论对象组成的非空有限集合，称为一个论域，记作  $U$ 。若给出论域  $U$  和  $U$  上的一簇等价关系  $R$ ，称二元组  $K = (U, R)$  是关于论域  $U$  上的一个知识库。

**通俗理解：**给一个集合，集合内的数据包含了很多上述对应的关系，例如某一系列数据和另一列数据存在关系（就是决策系统中条件属性和决策属性的对应关系）。然后就把这样的数据集合叫做了知识库，关系就是一种知识。

**定义 1.3** (不可分辨关系) 给定一个知识库  $K = (U, R)$ ，若  $P \subseteq R$ ，且  $P \neq \emptyset$ ，则  $\cap P$  仍然是论域  $U$  上的一个等价关系，称为  $P$  上的不可分辨关系，记为  $IND(P)$ ，简记为  $P$ 。

根据子集所确定的不可分辨关系，可以将论域  $U$  分为多个划分，记作  $U / IND(P)$ ，简记为  $U / P$ 。

**通俗理解：**不可分辨关系就是一种规则，也就是上表说的那些关系。这个需要通过实例才能更好地理解。

**定义 1.4** (集合的上近似和下近似) 若给定一个知识库  $K = (U, R)$ ，且有任意子集  $X \subseteq U$ ，等价关系  $R \in IND(K)$ ，则有

$$\underline{R}(X) = \cup \{Y \in U / R : Y \subseteq X\} \quad (1.1)$$

$$\bar{R}(X) = \cup \{Y \in U / R : Y \cap X \neq \emptyset\} \quad (1.2)$$

称  $\underline{R}(X)$  为  $X$  的下近似， $\bar{R}(X)$  为  $X$  的上近似。

于是可得子集  $X$  的边界域为

$$BN_R(X) = \bar{R}(X) - \underline{R}(X) \quad (1.3)$$

$Pos_R(X) = \underline{R}(X)$  称为子集  $X$  的  $R$  正域， $Neg_R(X) = U - \bar{R}(X)$  称为子集  $X$  的  $R$  负域。

**通俗理解：**下近似就是求解真子集（粗糙集的研究主要用到这个了）。上近似就是求解交集不为空。另外，正域这个概念比较重要，在约简计算中主要依靠这个。

**定义 1.5**（粗糙集和精确集）给定一个知识库  $K = (U, R)$ ， $\forall X \subseteq U$ ，若  $\underline{R}(X) = \bar{R}(X)$ ，则称集合  $X$  是关于论域  $U$  的相对于知识（等价关系） $R$  的精确集；若  $\underline{R}(X) \neq \bar{R}(X)$ ，则称集合  $X$  是关于论域  $U$  的相对于知识（等价关系） $R$  的粗糙集。

**通俗理解：**上近似等于下近似就是精确集，不等就是粗糙集。

下面用图形来粗略地表示一下粗糙集的相关概念。

若给定某一离散空间（集合），则经典粗糙集的相关定义可以通过图 1.1 表示出来。

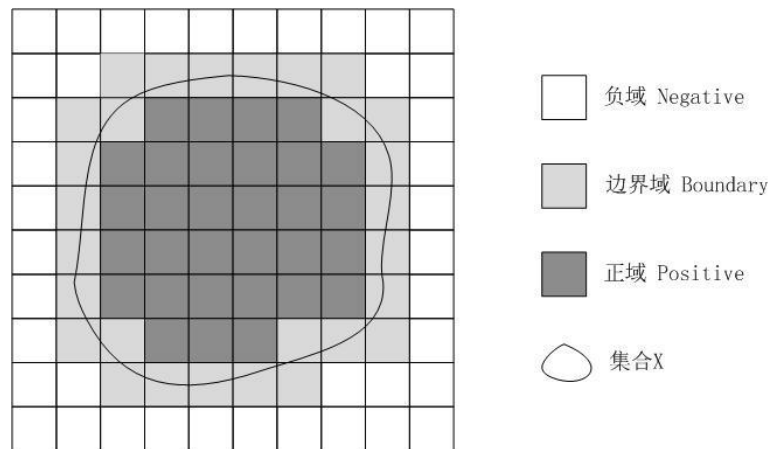


图 1.1 离散空间表示的经典粗糙集

Fig.1.1 Classical rough sets in discrete spaces

### 依赖度定义：

对于一个决策系统（一般情况下，在粗糙集的研究中，主要是研究这种称之为决策系统的集合） $DS = (U, C \cup D, V, f)$ ，其内部函数分别为： $U$  为论域， $C$  为条件属性， $D$  为决策属性，且  $C \cap D = \emptyset$ ， $D \neq \emptyset$ ， $V$  为各属性值  $V_a$  的集合， $V = \bigcup_{a \in C \cup D} V_a$ ， $f = U \times (C \cup D) \rightarrow V$  为信息函数（这个信息函数的概念完全可以不去理会，因为它说白了就是上边粗糙集定义中的那堆关系，就是各列属性对应得到的集合中的那些数值，也就是  $V$ ），表示样本、属性和属性值之间的映射关系。则给定一个决策系统  $DS$ ， $\forall B \subseteq C$ ，决策属性  $D$  对条件属性子集  $B$  的依赖度定义为：

$$\gamma_B(D) = \frac{|Pos_B(D)|}{|U|} \quad (1.4)$$

其中， $|\cdot|$  表示集合的基数，即集合内元素的个数。由公式定义可知，决策属性集对条件属性子集的依赖度即为条件属性子集  $B$  所确定的正域集合在论域  $U$  中所占的比例。

### 以下这个重要的推论是粗糙集约简的依据：

由依赖度公式 (1.4) 和正域定义可知，给定决策系统  $DS = (U, C \cup D, V, f)$ ，若有  $B_1 \subseteq B_2 \subseteq C$ ，则有  $|Pos_{B_1}(D)| \leq |Pos_{B_2}(D)| \leq |Pos_C(D)|$ ，从而由依赖度公式可得  $|\gamma_{B_1}(D)| \leq |\gamma_{B_2}(D)| \leq |\gamma_C(D)|$ ，即依赖度随着属性集合的增加而单调递增。

### 重要度的概念：

在决策系统中，属性重要度定义为条件属性对决策属性的影响程度。

常见的计算属性重要度的方法有基于属性依赖度的方法、基于信息熵的方法和基于互信息的方法。其中，基于属性依赖度的计算方法如下：

(1) 删除属性情况下的计算方法 给定决策系统  $DS = (U, C \cup D, V, f)$ ，

$\forall B \subseteq C$ ，若属性  $a \in B$ ，那么条件属性  $a$  对于决策属性  $D$  的重要性公式为：

$$Sig(a, B, D) = \gamma_B(D) - \gamma_{B-\{a\}}(D) \quad (1.5)$$

由公式可知，属性  $a$  对于决策属性  $D$  的重要度即为从条件属性集  $B$  中删除属

性  $a$  后，决策属性  $D$  对条件属性  $B$  依赖度减小的程度。

(2) 增加属性情况下的计算方法 同样在决策系统  $DS$  中，若属性  $a \in C$ ，但  $a \notin B$ ，那么条件属性  $a$  相对于条件属性集  $B$  对于决策属性  $D$  的重要度公式为：

$$Sig(a, B, D) = \gamma_{B \cup \{a\}}(D) - \gamma_B(D) \quad (1.6)$$

同样从公式中可以得出，属性  $a$  对于决策属性  $D$  的重要度即为在条件属性集  $B$  中增加属性  $a$  后，决策属性  $D$  对条件属性  $B$  依赖度增加的程度。

通过以上两个属性重要度的定义可知，属性重要度也是单调变化的。同样运用在属性的约简中。

**约简：**

决策系统中的约简即为将不必要的、冗余的属性删除，但又不影响决策系统本身的分类能力（分类精度会增加，才是好的约简）。

**定义 1.6**（知识的约简）给定一个知识库  $K = (U, R)$  和其上的一簇等价关系  $P \subseteq R$ ，对任意的  $G \subseteq P$ ，若  $G$  满足以下条件：

- (1)  $G$  是独立的，即  $G$  中每一个元素都是必不可少的；
- (2)  $IND(G) = IND(P)$ ，即不影响知识库的划分。

则称  $G$  是  $P$  的一个约简，记作  $G \in Red(P)$ 。其中， $Red(P)$  表示  $P$  的全体约简组成的集合。

**定义 1.7**（知识的核）给定一个知识库  $K = (U, R)$  和其上的一簇等价关系  $P \subseteq R$ ，对任意的  $Q \in P$ ，若  $Q$  满足

$$IND(P - \{Q\}) \neq IND(P) \quad (1.7)$$

则称  $Q$  为  $P$  中必要的， $P$  中所有必要的知识所组成的集合即为  $P$  的核，记作  $Core(P)$ 。

通常情况下，属性的约简并不唯一，可以有多个约简的集合。但知识系统的核只有一个，是众多约简集合的交集，是所有约简计算的基础。在约简过程中，属性的核是不能被删除的，否则将减弱系统的分类能力。

**定义 1.8**（决策系统的相对约简） 给定一个决策系统  $DS = (U, C \cup D, V, f)$ ，

$B \subseteq C$ ，若条件属性子集  $B$  满足以下条件：

(1)  $\gamma_B(D) = \gamma_C(D)$ ，即  $Pos_B(D) = Pos_C(D)$ ，条件属性子集  $B$  和  $C$  分类能力相同；

(2)  $\forall a \in B, \gamma_B(D) > \gamma_{B-\{a\}}(D)$ ，即条件属性子集  $B$  中没有冗余。

则称条件属性子集  $B$  是条件属性集  $C$  的一个相对约简。

以上这些基本概念都比较好理解，只要仔细读一遍就差不多了。

## 1.2 邻域粗糙集的基本概念

邻域粗糙集是为了解决经典粗糙集不便于处理数值型属性的数据集合而提出的。经典粗糙集在处理连续型数据时需要将数据进行离散化处理，但是离散化处理后改变数据原始的属性性质。

**定义 1.9** (度量) 在一给定的  $N$  维实数空间  $\Omega$  中， $\Delta = R^N \times R^N \rightarrow R$ ，则称  $\Delta$  为  $R^N$  上的一个度量 (距离)，若  $\Delta$  满足以下条件：

(1)  $\Delta(x_1, x_2) \geq 0$ ，其中当且仅当  $x_1 = x_2$  时等号成立， $\forall x_1, x_2 \in R^N$ ；

(2)  $\Delta(x_1, x_2) = \Delta(x_2, x_1)$ ， $\forall x_1, x_2 \in R^N$ ；

(3)  $\Delta(x_1, x_3) \leq \Delta(x_1, x_2) + \Delta(x_2, x_3)$ ， $\forall x_1, x_2, x_3 \in R^N$ ，

则称  $(\Omega, \Delta)$  为度量空间。 $\Delta(x_i, x_j)$  为距离函数，表示元素  $x_i$  和元素  $x_j$  之间的距离。常见的距离计算函数有以下几种：

$$\text{曼哈顿距离函数: } \Delta(x_i, x_j) = \sum_{k=1}^N |f(x_i, a_k) - f(x_j, a_k)| \quad (1.8)$$

$$\text{欧几里德距离函数: } \Delta(x_i, x_j) = \left( \sum_{k=1}^N (f(x_i, a_k) - f(x_j, a_k))^2 \right)^{1/2} \quad (1.9)$$

$$P \text{ 范数距离函数: } \Delta(x_i, x_j) = \left( \sum_{k=1}^N |f(x_i, a_k) - f(x_j, a_k)|^p \right)^{1/p} \quad (1.10)$$

**通俗理解：**这个度量就是距离，空间中的任意两点之间的距离。

**定义 1.10** ( $\delta$ -邻域) 在给定实数空间  $\Omega$  上的非空有限集合  $U = \{x_1, x_2, \dots, x_n\}$ ，

对  $\forall x_i$  的邻域  $\delta$  - 定义为

$$\delta(x_i) = \{x | x \in U, \Delta(x, x_i) \leq \delta\} \quad (1.11)$$

其中,  $\delta \geq 0$ 。

**定义 1.11** (邻域的上近似和下近似) 给定实数空间  $\Omega$  上的非空有限集合  $U = \{x_1, x_2, \dots, x_n\}$  及其上的邻域关系  $N$ , 即二元组  $NS = (U, N)$ ,  $\forall X \subseteq U$ , 则  $X$  在邻域近似空间  $NS = (U, N)$  中的上近似和下近似分别为

$$\bar{N}X = \{x_i | \delta(x_i) \cap X \neq \emptyset, x_i \in U\} \quad (1.12)$$

$$\underline{N}X = \{x_i | \delta(x_i) \subseteq X, x_i \in U\} \quad (1.13)$$

则可以得出  $X$  的近似边界为

$$BN(X) = \bar{N}X - \underline{N}X \quad (1.14)$$

同经典粗糙集理论中的定义相似, 同样定义  $X$  的下近似  $\underline{N}X$  为正域, 与  $X$  完全无关的区域为负域, 即

$$Pos(X) = \underline{N}X \quad (1.15)$$

$$Neg(X) = U - \bar{N}X \quad (1.16)$$

以上这些东西和经典粗糙集中的一样, 在这里不做过多解释。

**定义 1.12** (邻域决策系统的上近似和下近似) 给定一邻域决策系统  $NDS = (U, A \cup D)$ , 决策属性  $D$  将论域  $U$  划分为  $N$  个等价类  $(X_1, X_2, \dots, X_N)$ ,  $\forall B \subseteq A$ , 则决策属性  $D$  关于子集  $B$  的上、下近似分别为

$$\bar{N}_B D = \bigcup_{i=1}^N \bar{N}_B X_i \quad (1.18)$$

$$\underline{N}_B D = \bigcup_{i=1}^N \underline{N}_B X_i \quad (1.19)$$

其中,

$$\bar{N}_B X = \{x_i | \delta_B(x_i) \cap X \neq \emptyset, x_i \in U\} \quad (1.20)$$

$$\underline{N}_B X = \{x_i | \delta_B(x_i) \subseteq X, x_i \in U\} \quad (1.21)$$



同样可得决策系统的边界为

$$BN(D) = \bar{N}_B D - \underline{N}_B D \quad (1.22)$$

邻域决策系统的正域和负域分别为

$$Pos_B(D) = \underline{N}_B D \quad (1.23)$$

$$Neg_B(D) = U - \bar{N}_B D \quad (1.24)$$

决策属性  $D$  对条件属性  $B$  的依赖度为

$$k_D = \gamma_B(D) = \frac{|Pos_B(D)|}{|U|} \quad (1.25)$$

由上式可得依赖度  $k_D$  是单调的，若  $B_1 \subseteq B_2 \subseteq \dots \subseteq A$ ，则

$$\gamma_{B_1}(D) \leq \gamma_{B_2}(D) \leq \dots \leq \gamma_A(D)。$$

对比来说，经典粗糙集和邻域粗糙集相关定义的情况如下表所示。

	粗糙集	邻域粗糙集
邻域	——	$\delta_B(x_i) = \{x_j   x_j \in U, \Delta_B(x_i, x_j) \leq \delta\}$
上近似	$\bar{R}(X) = \cup \{Y \in U / R : Y \subseteq X\}$	$\bar{N}_B D = \bigcup_{i=1}^N \bar{N}_B X_i \quad \bar{N}_B X = \{x_i   \delta_B(x_i) \cap X \neq \emptyset, x_i \in U\}$
下近似	$\underline{R}(X) = \cup \{Y \in U / R : Y \cap X \neq \emptyset\}$	$\underline{N}_B D = \bigcup_{i=1}^N \underline{N}_B X_i \quad \underline{N}_B X = \{x_i   \delta_B(x_i) \subseteq X, x_i \in U\}$
正域	$Pos_R(X) = \underline{R}(X)$	$Pos_B(D) = \underline{N}_B D$
负域	$Neg_R(X) = U - \bar{R}(X)$	$Neg_B(D) = U - \bar{N}_B D$
依赖度	$\gamma_B(D) = \frac{ Pos_B(D) }{ U }$	$k_D = \gamma_B(D) = \frac{ Pos_B(D) }{ U }$
重要度	$Sig(a, B, D) = \gamma_B(D) - \gamma_{B-\{a\}}(D)$	$Sig(a, B, D) = \gamma_B(D) - \gamma_{B-\{a\}}(D)$

主要区别在于邻域，其他基本相同。

## 2 邻域粗糙集算例

建议大家最好按照我这里的计算思路和计算过程，严格地走一遍，这样能加深你对邻域粗糙集的理解，否则后面的程序也会比较迷糊。

在第一个版本中，我手动计算了下面这个算例，但是很抱歉，在计算过程中出现了失误，算错了一些集合，可能会给大家带来了误导，在此向大家道歉！其

中错误的部分就是我用黄色标记的部分，希望大家区别对待。

下面选取国际标准数据集（UCI）中 Iris 部分数据对邻域粗糙集相关理论进行验证计算。为了适应实际决策系统中属性的真实数据类型，本例子在 Iris 数据集的基础上增加了符号型数据，符号型数据是根据属性  $a_1$  和属性  $a_2$  数值的和确定的，根据两个属性数值和的大小，将属性  $a_3$  分为 1 和 2 两类，以保证不影响数据集的整体分类趋势。具体数据内容如表 1.1 所示，其中  $x_i (i=1,2,\dots,8)$  为选取的样本， $a_1$  和  $a_2$  为数值型数值属性， $a_3$  为符号型属性， $D$  为决策属性。

表 1.1 邻域决策系统

Table 1.1 Neighborhood decision system

样本	a1	a2	a3	D
$x_1$	5.1	3.5	1	Setosa
$x_2$	4.9	3.0	1	Setosa
$x_3$	5.8	2.7	1	Virginica
$x_4$	6.3	3.3	1	Virginica
$x_5$	7.1	3.0	2	Virginica
$x_6$	6.9	3.1	2	Versicolor
$x_7$	7.0	3.2	2	Versicolor
$x_8$	6.4	3.2	1	Versicolor

在邻域粗糙集数据处理中，数据通常会存在数量级和量纲的差异，为了得到精确的数据处理结果，在数据处理之前需要对原始数据进行归一化处理。常用的归一化处理方法多采用最大最小值法，其公式为

$$f(x_i) = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \quad (i = 1, 2, \dots, n) \quad (1.26)$$

其中， $x_{\max}$  和  $x_{\min}$  分别为样本数组的最大和最小值。经过归一化处理之后，数据均落在  $[0,1]$  区间。

对表 1.1 邻域决策系统中数据做归一化处理后得到表 1.2 的数据结果。

表 1.2 归一化后的邻域决策系统

Table 1.2 Neighborhood decision system after normalized

样本	a1	a2	a3	D
$x_1$	0.0909	1.0000	1	Setosa
$x_2$	0	0.3750	1	Setosa
$x_3$	0.4091	0	1	Virginica
$x_4$	0.6364	0.7500	1	Virginica
$x_5$	1.0000	0.3750	2	Virginica
$x_6$	0.9091	0.5000	2	Versicolor
$x_7$	0.9545	0.6250	2	Versicolor
$x_8$	0.6818	0.6250	1	Versicolor

对归一化处理后的属性  $a_1$ 、 $a_2$  数据求标准差可得，属性  $a_1$  的标准差  $Stdq=0.38602$ ，属性  $a_2$  的标准差  $Std a_2=0.296934$ 。此处应用公式（1.27）进行计算。

$$\delta(a_i) = Std a_i / \lambda \quad (1.27)$$

其中  $Std a_i$  为属性  $a_i$  数据的标准差， $\lambda$  是一个设定的参数，用于根据数据分类精度调整邻域大小，通常取值在 2~4 之间。本文算例中暂取  $\lambda=2$ ，可得两个属性的邻域半径分别为  $\delta(a_1)=0.1930$ ， $\delta(a_2)=0.1485$ 。

以上这样做的好处就是为了避免原始邻域粗糙集邻域半径不能确定的问题。

对于属性子集  $B_1=\{a_1\}$ ， $B_2=\{a_2\}$ ， $B_{12}=\{a_1,a_2\}$  而言，根据邻域计算公式（1.17）计算可得在属性子集  $B_1$ ， $B_2$  和  $B_{12}$  下的各个样本的邻域，即以某一样本具体属性值为中心，邻域半径为半径做空间内图形（二维下为圆，三维下为球），则包含在该图形内的所有样本属性值的集合即为该属性值的邻域。计算结果如表

1.3 所示。

表 1.3 样本邻域

Table 1.3 Sample neighborhood

样本邻域	$B_1$	$B_2$	$B_{12}$
$\delta_{B_1}(x_1)$	$\{x_1, x_2\}$	$\{x_1\}$	$\{x_1\}$
$\delta_{B_1}(x_2)$	$\{x_1, x_2\}$	$\{x_2, x_5, x_6\}$	$\{x_2\}$
$\delta_{B_1}(x_3)$	$\{x_3\}$	$\{x_3\}$	$\{x_3\}$
$\delta_{B_1}(x_4)$	$\{x_4, x_8\}$	$\{x_4, x_7, x_8\}$	$\{x_4, x_8\}$
$\delta_{B_1}(x_5)$	$\{x_5, x_6, x_7\}$	$\{x_2, x_5, x_6\}$	$\{x_5, x_6\}$
$\delta_{B_1}(x_6)$	$\{x_5, x_6, x_7\}$	$\{x_2, x_5, x_6, x_7, x_8\}$	$\{x_5, x_6, x_7\}$
$\delta_{B_1}(x_7)$	$\{x_5, x_6, x_7\}$	$\{x_4, x_6, x_7, x_8\}$	$\{x_6, x_7\}$
$\delta_{B_1}(x_8)$	$\{x_4, x_8\}$	$\{x_4, x_7, x_8\}$	$\{x_4, x_8\}$

决策  $D$  对论域  $U$  的划分为  $U/D = \{\{x_1, x_2\}, \{x_3, x_4, x_5\}, \{x_6, x_7, x_8\}\}$ ，令  $X_1 = \{x_1, x_2\}$ ， $X_2 = \{x_3, x_4, x_5\}$ ， $X_3 = \{x_6, x_7, x_8\}$ ，则根据式 1.20 和式 1.21 计算可得：

(1) 决策子集  $X_1$ ， $X_2$ ， $X_3$  关于属性子集  $B_1$  的上、下近似：

$$\bar{N}_{B_1} X_1 = \{x_1, x_2\}, \quad \bar{N}_{B_1} X_2 = \{x_3, x_4, x_5, x_6, x_7, x_8\}, \quad \bar{N}_{B_1} X_3 = \{x_4, x_5, x_6, x_7, x_8\};$$

$$\underline{N}_{B_1} X_1 = \{x_1, x_2\}, \quad \underline{N}_{B_1} X_2 = \{x_3\}, \quad \underline{N}_{B_1} X_3 = \emptyset.$$

(2) 决策子集  $X_1$ ， $X_2$ ， $X_3$  关于属性子集  $B_2$  的上、下近似：

$$\bar{N}_{B_2} X_1 = \{x_1, x_2, x_5, x_6\}, \quad \bar{N}_{B_2} X_2 = \{x_2, x_3, x_4, x_5, x_6, x_7, x_8\}, \quad \bar{N}_{B_2} X_3 = \{x_2, x_4, x_5, x_6, x_7, x_8\};$$

$$\underline{N}_{B_2} X_1 = \{x_1\}, \quad \underline{N}_{B_2} X_2 = \{x_3\}, \quad \underline{N}_{B_2} X_3 = \emptyset.$$

(3) 决策子集  $X_1$ ， $X_2$ ， $X_3$  关于属性子集  $B_{12}$  的上、下近似：

$$\bar{N}_{B_{12}} X_1 = \{x_1, x_2\}, \quad \bar{N}_{B_{12}} X_2 = \{x_3, x_4, x_5, x_6, x_8\}, \quad \bar{N}_{B_{12}} X_3 = \{x_4, x_5, x_6, x_7, x_8\};$$

$$\underline{N}_{B_{12}} X_1 = \{x_1, x_2\}, \quad \underline{N}_{B_{12}} X_2 = \{x_3\}, \quad \underline{N}_{B_{12}} X_3 = \{x_6, x_7\}。$$

整理计算可得：

(1) 决策属性  $D$  关于属性子集  $B_1$  的上、下近似为：

$$\bar{N}_{B_1} D = \bar{N}_{B_1} X_1 \cup \bar{N}_{B_1} X_2 \cup \bar{N}_{B_1} X_3 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} = U；$$

$$\underline{N}_{B_1} D = \underline{N}_{B_1} X_1 \cup \underline{N}_{B_1} X_2 \cup \underline{N}_{B_1} X_3 = \{x_1, x_2, x_3\}。$$

(2) 决策属性  $D$  关于属性子集  $B_2$  的上、下近似为：

$$\bar{N}_{B_2} D = \bar{N}_{B_2} X_1 \cup \bar{N}_{B_2} X_2 \cup \bar{N}_{B_2} X_3 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} = U；$$

$$\underline{N}_{B_2} D = \underline{N}_{B_2} X_1 \cup \underline{N}_{B_2} X_2 \cup \underline{N}_{B_2} X_3 = \{x_1, x_3\}。$$

(3) 决策属性  $D$  关于属性子集  $B_{12}$  的上、下近似为：

$$\bar{N}_{B_{12}} D = \bar{N}_{B_{12}} X_1 \cup \bar{N}_{B_{12}} X_2 \cup \bar{N}_{B_{12}} X_3 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} = U；$$

$$\underline{N}_{B_{12}} D = \underline{N}_{B_{12}} X_1 \cup \underline{N}_{B_{12}} X_2 \cup \underline{N}_{B_{12}} X_3 = \{x_1, x_2, x_3, x_6, x_7\} = Pos_{B_{12}}(D)。$$

引入符号属性子集  $B_3 = \{a_3\}$ ，其对论域  $U$  的划分为  $U / \mathcal{B} = \{\{x_1, x_2, x_3, x_4, x_8\}, \{x_5, x_6, x_7\}\}$ ，则基于属性集合  $A = \{a_1, a_2, a_3\}$  的邻域表示为： $\delta_A(x_1) = \{x_1\}$ ， $\delta_A(x_2) = \{x_2\}$ ， $\delta_A(x_3) = \{x_3\}$ ， $\delta_A(x_4) = \{x_4, x_8\}$ ， $\delta_A(x_5) = \{x_5, x_6\}$ ， $\delta_A(x_6) = \{x_5, x_6, x_7\}$ ， $\delta_A(x_7) = \{x_7\}$ ， $\delta_A(x_8) = \{x_4, x_8\}$ 。按照以上算例步骤，计算得出的决策属性  $D$  关于属性集合  $A$  的上、下近似为：

$$\bar{N}_A D = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\} = U；$$

$$\underline{N}_A D = \{x_1, x_2, x_3, x_6, x_7\} = Pos_A(D)。$$

为了后续计算各属性重要度方便，在此处直接给出决策属性  $D$  关于属性子集  $B_{13} = \{a_1, a_3\}$  和  $B_{23} = \{a_2, a_3\}$  的下近似计算结果，中间计算过程同上述相似。

$$\underline{N}_{B_{13}} D = \{x_1, x_2, x_3\} = Pos_{B_{13}}(D)， \quad \underline{N}_{B_{23}} D = \{x_1, x_2, x_3, x_6, x_7\} = Pos_{B_{23}}(D)。$$

### 3 邻域粗糙集约简

#### 3.1 邻域粗糙集约简基本思想

邻域决策系统条件属性的重要度公式同经典粗糙集理论的（公式 1.4、1.5、1.6）相同，在此不再赘述。根据条件属性的重要度，可以在邻域粗糙集约简计算中通过判断属性重要度是否大于零而得出约简集合。因此，在研究邻域粗糙集约简之前，本文先对邻域粗糙集下邻域决策系统有关的依赖度和属性重要度做一个算例。

本算例采用的数据同前一小节中表 1.1 相同，在依赖度和重要度计算中，引用的计算结果即为前一小节计算结果。具体计算如下：

决策  $D$  对属性子集  $B_1$ 、 $B_2$ 、 $B_{12}$ 、 $B_{13}$ 、 $B_{23}$  和  $A$  的依赖度分别为：

$$\gamma_{B_1}(D) = \frac{|Pos_{B_1}(D)|}{|U|} = \frac{|N_{B_1}D|}{|U|} = \frac{|\{x_1, x_2, x_3\}|}{|\{x_1, x_2, \dots, x_8\}|} = \frac{3}{8} = 0.375;$$

$$\gamma_{B_2}(D) = \frac{|Pos_{B_2}(D)|}{|U|} = \frac{|N_{B_2}D|}{|U|} = \frac{|\{x_1, x_3\}|}{|\{x_1, x_2, \dots, x_8\}|} = \frac{2}{8} = 0.25;$$

$$\gamma_{B_{12}}(D) = \frac{|Pos_{B_{12}}(D)|}{|U|} = \frac{|N_{B_{12}}D|}{|U|} = \frac{|\{x_1, x_2, x_3, x_6, x_7\}|}{|\{x_1, x_2, \dots, x_8\}|} = \frac{5}{8} = 0.625;$$

$$\gamma_{B_{13}}(D) = \frac{|Pos_{B_{13}}(D)|}{|U|} = \frac{|N_{B_{13}}D|}{|U|} = \frac{|\{x_1, x_2, x_3\}|}{|\{x_1, x_2, \dots, x_8\}|} = \frac{3}{8} = 0.375;$$

$$\gamma_{B_{23}}(D) = \frac{|Pos_{B_{23}}(D)|}{|U|} = \frac{|N_{B_{23}}D|}{|U|} = \frac{|\{x_1, x_2, x_3, x_6, x_7\}|}{|\{x_1, x_2, \dots, x_8\}|} = \frac{5}{8} = 0.625;$$

$$\gamma_A(D) = \frac{|Pos_A(D)|}{|U|} = \frac{|N_AD|}{|U|} = \frac{|\{x_1, x_2, x_3, x_6, x_7\}|}{|\{x_1, x_2, \dots, x_8\}|} = \frac{5}{8} = 0.625。$$

属性  $a_1, a_2, a_3$  在属性集合  $A$  中相对于决策  $D$  的属性重要度分别为：

$$Sig(a_1, A, D) = \gamma_A(D) - \gamma_{A-\{a_1\}}(D) = \gamma_A(D) - \gamma_{B_{23}}(D) = 0.625 - 0.625 = 0;$$

$$Sig(a_2, A, D) = \gamma_A(D) - \gamma_{A-\{a_2\}}(D) = \gamma_A(D) - \gamma_{B_{13}}(D) = 0.625 - 0.375 = 0.25;$$

$$Sig(a_3, A, D) = \gamma_A(D) - \gamma_{A-\{a_3\}}(D) = \gamma_A(D) - \gamma_{B_{12}}(D) = 0.625 - 0.625 = 0。$$

通过以上计算结果可以看出属性  $a_1 a_3$  相对于决策  $D$  的重要度为 0, 属于冗余属性, 在约简计算中应该予以删除。

## 3.2 上述计算程序说明

包含: reduceSet.m、getPosSet.m、mainAccess.m。

其中:

reduceSet.m: 约简程序

getPosSet.m: 获取正域程序

mainAccess.m: 主程序入口

以上程序是近阶段我自己编写的, 初步调试没有问题, 但是肯定会存在一些不如意的地方, 希望大家给出修改意见, 帮忙找出 bug, 共同修复程序, 完善邻域粗糙集的计算程序。

按照上面给出的手动计算过程和结果, 下面给出程序的调用说明和计算结果。

这里说明一下, 在第一个版本发布之后, 有不少网友提出了宝贵的意见和建议, 对我关于邻域粗糙集和粗糙集的研究帮助很大, 也有网友对 matlab 程序不是很了解, 不知道从哪入门, 所以我在这次编写程序时候, 特意增加了程序代码的注释, 对变量的声明, 方法的运用都做了相应说明。另外, 大家如果还不是很理解我的程序, 那就手动设置断点, 然后根据我上面给出的例子, 自己单步执行操作, 就可以知道具体的计算过程和程序的具体运行情况了。再退一步说, 如果不想仔细研究程序的具体运行, 那就直接按照我下面的步骤执行就行了, 也不用管程序内部是怎么运行的了。不过, 我还是建议大家在一边理解邻域粗糙集理论的同时, 一遍按照程序进行, 这样对掌握邻域粗糙集理论、掌握 matlab 程序都很有帮助。

程序说明如下:

### 3.2.1 mainAccess.m

该程序很简单, 主要是清空变量, 加载数据, 调用其他的 m 文件, 执行对应的操作。大家也可以不用这个入口程序, 直接单独使用每个函数文件自身, 按

照后面的说明操作也是可以的。

主要程序如下：

```
clc;
clear all
load('test.mat') % 加载的 test.mat 数据为邻域粗糙集使用的数值型决策系统数据
% 最后一列为决策属性
lammda=0.6; % 邻域半径计算参数 delta=std (dataArray) /lammda
% 这里 lammda 取值尽量在 0.5~1.5 之间，如果太大，则不能输出正常结果，如果太小，则程序报错
% 如果数据内包含的样本数比较多（几十以上），则调大 lammda=2~4
sig_ctrl=0.01; % 重要度下限的控制参数，取接近 0 的数
reduceSet = reduceSet(test,lammda,sig_ctrl) % 计算约简集合
weightD = weightD(test,lammda) % 计算权重
```

注意上面说明的 lammda 参数和重要度下限控制参数。这里的 lammda 参数我还没有做太多的试验，究竟选取什么样的数值，我也不太清楚。但是有以下几点建议：

（1）这里的 lammda 和胡的约简程序中的 lammda 有些区别，在我试验验证时，胡的程序 lammda 取 2-4 比较合适。但是我这里的程序，该参数建议取 0.5~1.5。

（2）如果进行约简的数据内条件属性比较多，10 个以上，样本数比较少 15 个以内，则尽量降低 lammda 数值。如果条件属性较少，样本数较多，则 lammda 取值适当增大一些。当为诸如 UCI 数据集那样的数据，样本数达到成百上千，则还是取 2~4 为好。

（3）如果上述两条试验结果不好，那么就自己手动调整或者编个程序按照一定步长取一下 lammda，再做试验进行有针对性的选取吧！（注：如果哪位把这个问题解决了，希望能反馈给我一下，多谢！）

至于重要度下限控制参数，这个就根据自己心情选取吧！不过建议选取非常贴近 0 的正数，如 0.001 之类的，因为这里计算出的重要度，是按照粗糙集重要



度概念计算出来的，同胡的程序内的有些区别。当然，也可以根据自己的实际数据情况自己选取。

上面的程序内包含了这样一个语句：`weight = weightD(test,lammda) % 计算权重`

我写了这个程序的目的是因为不少网友利用粗糙集计算权重，所以这里也写了这个程序，同样在主入口程序内调用了。

主入口程序没有调用计算正域的程序，是因为这个结果是单步执行都会有不同结果的，如果大家需要单独计算正域，那就单独执行那个 `m` 文件吧！详细参看后面的说明。

加载的 `test.mat` 文件，大家可以通过数据拷贝的方式，或者其他加载方式，把数据按照指定格式加载到 `matlab/workspace` 中。注意，每个属性是一列，前边是条件属性，最后一列是决策属性，每行是一个样本。数据类型可以是数值型的（如 0.342 3.567 12934.56），也可以是字符型的（如 0 1 2）

### 3.2.2 getPosSet.m

该程序为生成正域的程序。一般是被其他程序调用的，当然也可以单独执行。调用方法为：`PosSet = getPosSet(dataArray,lammda)`

```
% 输入 dataArray 为包含决策属性的数据样本，最后一列为决策属性
% lammda 为邻域半径集合计算时候的参数 delta=std(data)/lammda
% lammda 注意！在这里计算的 lammda 和胡清华程序的 lammda 有区别
% 这里 lammda 取值尽量在 0.5~1.5 之间，如果太大，则不能输出正常结果，
如果太小，则程序报错
% 如果数据内包含的样本数比较多（几十以上），则调大 lammda=2~4
% 输出的 PosSet 为正域集合
```

大家可以按照上面给出的调用方法，单独计算个体属性的下近似（正域）。

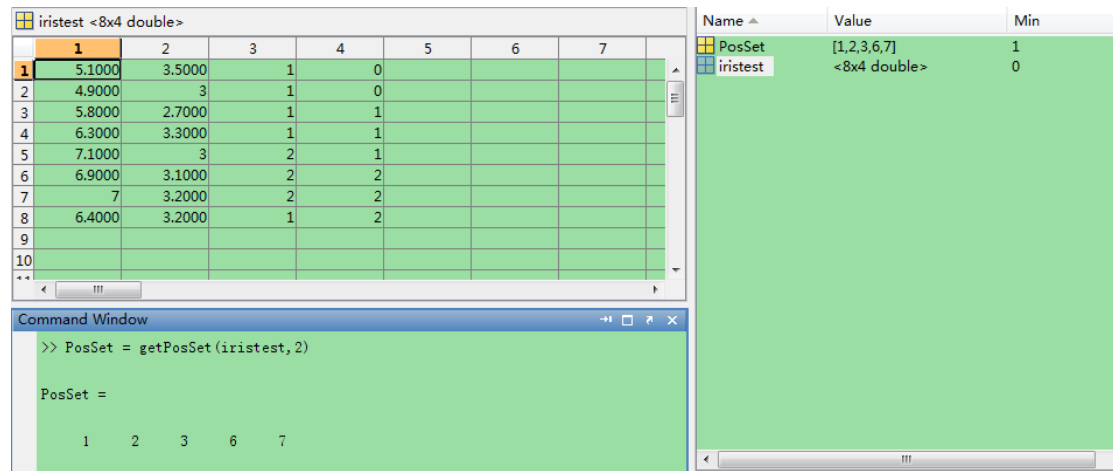
下面给出 2 小节中给出的邻域粗糙集算例的程序调用及结果。

首先，加载 iris 的数据，我命名为 iristest，数据情况及计算的正域如下图所示。

其中，参数 lammda 按照上面例子中的，取 2。

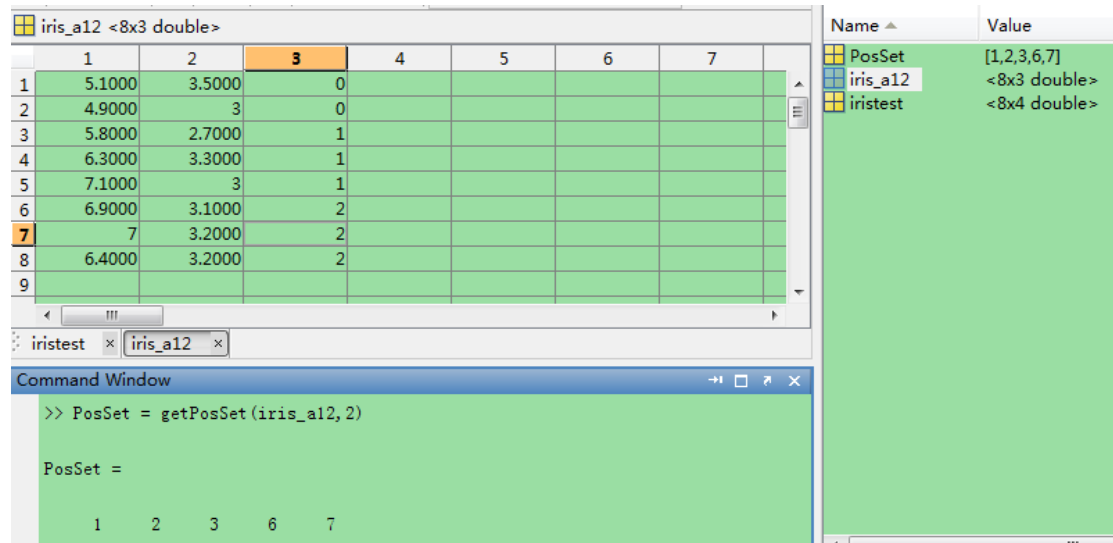
可以看到约简结果为 1 2 3 6 7。

上面 2 中计算的结果同样为： $\underline{N}_A D = \{x_1, x_2, x_3, x_6, x_7\} = Pos_A(D)$ 。



继续调用，计算一下分别去掉单个属性后的其他条件属性的正域。

### (1) 去掉条件属性 3



约简结果为 1 2 3 6 7

2 中算例结果： $\underline{N}_{B_{12}} D = \underline{N}_{B_{12}} X_1 \cup \underline{N}_{B_{12}} X_2 \cup \underline{N}_{B_{12}} X_3 = \{x_1, x_2, x_3, x_6, x_7\} = Pos_{B_{12}}(D)$ 。

## (2) 去掉条件属性 2

iris_a13 <8x3 double>							
	1	2	3	4	5	6	7
1	5.1000	1	0				
2	4.9000	1	0				
3	5.8000	1	1				
4	6.3000	1	1				
5	7.1000	2	1				
6	6.9000	2	2				
7	7	2	2				
8	6.4000	1	2				
9							

Name	Value
PosSet	[1,2,3]
iris_a12	<8x3 double>
iris_a13	<8x3 double>
iristest	<8x4 double>

```

>> PosSet = getPosSet(iris_a13,2)

PosSet =

     1     2     3
  
```

约简结果为 1 2 3

2 中算例结果为:  $\underline{N}_{B_{13}}D = \{x_1, x_2, x_3\} = Pos_{B_{13}}(D)$

## (3) 去掉条件属性 1

iris_a23 <8x3 double>							
	1	2	3	4	5	6	7
1	3.5000	1	0				
2	3	1	0				
3	2.7000	1	1				
4	3.3000	1	1				
5	3	2	1				
6	3.1000	2	2				
7	3.2000	2	2				
8	3.2000	1	2				
9							

Name	Value
PosSet	[1,2,3,6,7]
iris_a12	<8x3 double>
iris_a13	<8x3 double>
iris_a23	<8x3 double>
iristest	<8x4 double>

```

>> PosSet = getPosSet(iris_a23,2)

PosSet =

     1     2     3     6     7
  
```

计算结果为 1 2 3 6 7

2 算例计算结果为:  $\underline{N}_{B_{23}}D = \{x_1, x_2, x_3, x_6, x_7\} = Pos_{B_{23}}(D)$ 。

可以看到，上述程序计算结果和手动计算一致。

## 3.2.3 reduceSet.m

该程序为约简计算程序。根据前边计算出来的正域，在本程序内计算每个条件属性的依赖度（之差），从而得到每个条件属性的重要度，然后根据用书输入

的重要度下限，得到约简结果。

调用方式为：`[redSet,sigSet] = reduceSet(dataArray,lammda,sig_ctrl)`

% 该程序为邻域粗糙集的约简计算

% 输入 `dataArray` 数值型决策系统，最后一列为决策属性

% 输入 `lammda` 计算邻域半径时的参数， $\text{delta}=\text{std}(\text{dataArray})/\text{lammda}$

% `lammda` 注意！在这里计算的 `lammda` 和胡清华程序的 `lammda` 有区别

% 这里 `lammda` 取值尽量在 0.5~1.5 之间，如果太大，则不能输出正常结果，如果太小，则程序报错

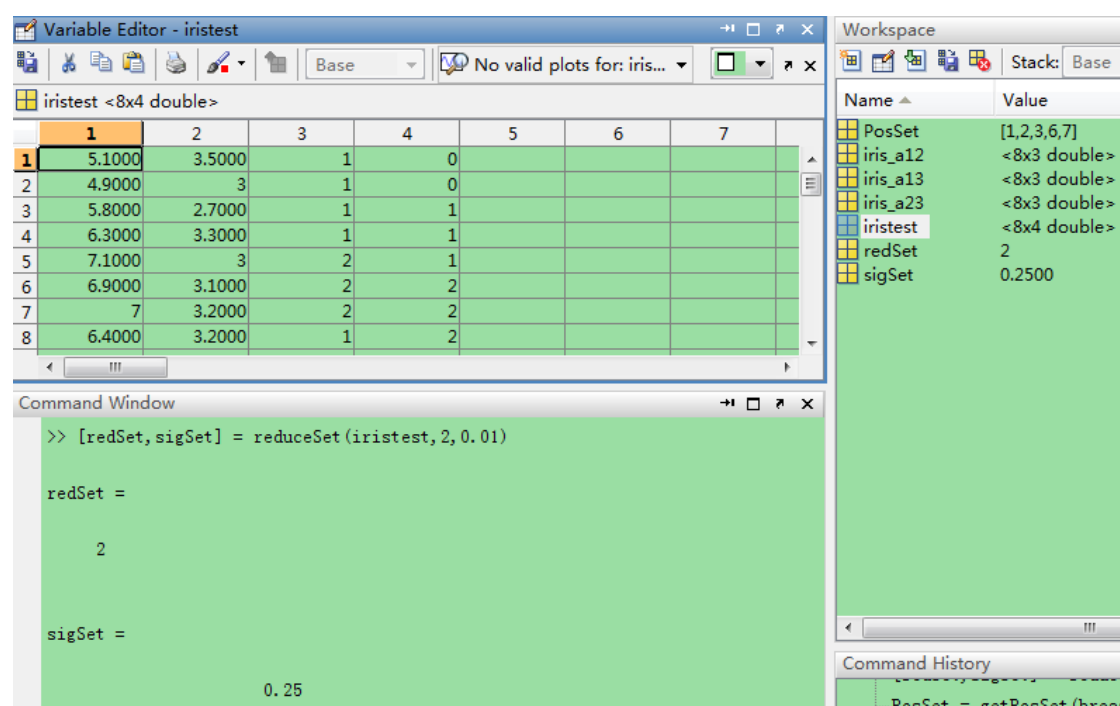
% 如果数据内包含的样本数比较多（几十以上），则调大 `lammda=2~4`

% 输入 `sig_ctrl` 重要度下限的一个控制参数，不能太小，也不能太大。0.001 左右

% 输出 约简后的集合及其重要度

% 此程序调用了计算积极域的程序 `getPosSet`

继续执行 2 算例中的 `iris` 说明程序的调用情况。调用过程及结果如下图所示。



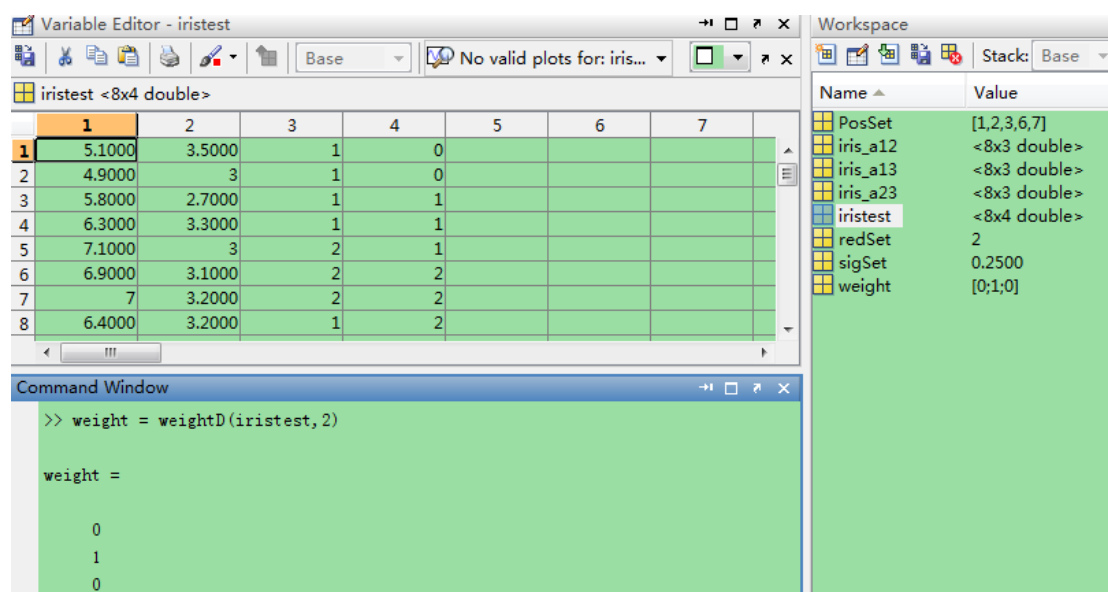
通过 2 中给出的算例，已经可以说明计算结果了，其中条件属性 1 和 3 的重要度为 0（这也是单独使用粗糙集计算重要度的弊端，会出现 0 现象）。所以约简结果为 2（条件属性 2），其重要度为 0.25。

### 3.2.4 weightD.m

这是权值计算程序，在此就不多说了，主要是利用计算的属性重要度计算每个条件属性的权值。

调用方式为：`weight = weightD(dataArray, lammda)`

计算结果如下图所示。



可以看出，条件属性 2 的权值为 1，其他的为 0.

## 3.3 邻域粗糙集约简等计算程序说明

### 3.3.1 胡程序测试

通过以上算例描述，对于邻域粗糙集的约简大家应该能明确一些了。

目前，邻域粗糙集中应用的算例基本上是胡清华版本的 matlab 程序。

网上有好多这个程序的下载地址，在中国程序员联合开发网或者 CSDN 上都能下载到，如果你实在下载不到，或者懒得下载，那么你可以找我，我发给你。

另外，补充一点，上述算例中，并没有严格按照胡清华的原本程序走的，而是做了一些小小的改进，就是处理了一下邻域半径的问题。把原来的单一邻域半径改变成了一个数组形式的邻域半径，然后分别应用。

邻域粗糙集约简用的是前向贪心算法，主要算法如下图所示：

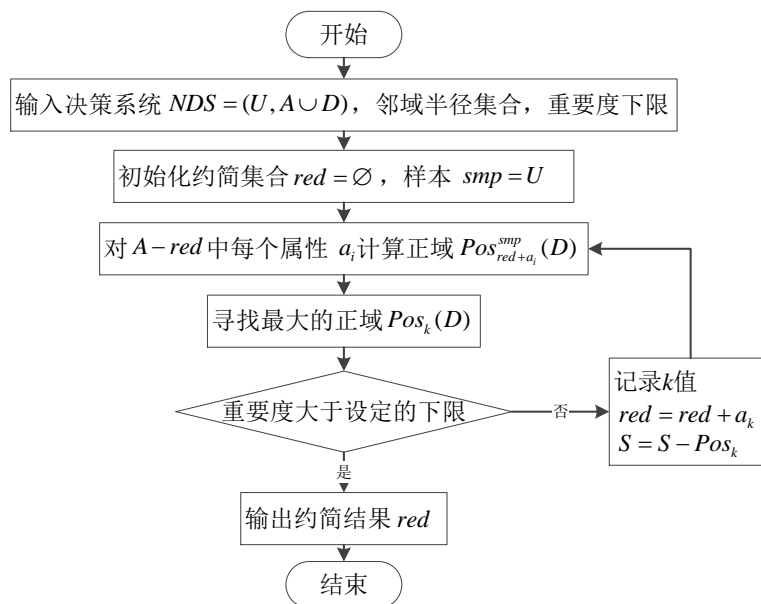


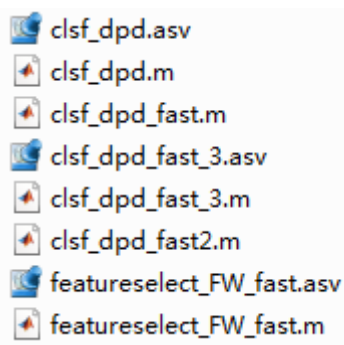
图 3.1 前向贪心约简算法

这个算法还是比较好理解的，算法中，保留了属性重要度最大的那个，也就是保证了核不被约简。如果在参数设置的时候设置不当，也可能造成核被约简，但是这种情况比较少见。

下面来讲解一下这个 matlab 的程序怎么使用：

个人建议，在使用本程序之前，请您针对 matlab 的使用做一些初始化的学习，然后再来应用。

胡清华的程序中包含以下几个文件：



主程序入口是 featureselect\_FW\_fast.m，其他四个 m 文件都是不同的约简选择。

按照胡清华版本的程序使用，需要读懂主程序中的注释说明：

%efficiency: the significance of the selected features。（重要度）

%data\_array: Input data with decision ranked in the last column（约简后的属性）

%delta: the size of neighborhood. generally speaking, delta=[0.1,0.2] (邻域半径)

%efc\_ctrl: The threshold to stop the search. The search is stoped if the increment introduced with every new feature is less than efc\_ctrl. (重要度下限)

%fun\_dpd: This parameter selects the strategies of computing attribute significance. (选择的约简算法)

% fun\_dpd should be cited with single quotation marks, for example, 'clsf\_dpd\_fast'.

在 matlab 命令行中的输入示例:

```
[feature_slct,efficiency]=featureselect_FW_fast(my_array,0.15,0.001,'clsf_dpd_fast')
```

其中 my\_array 就是你所要约简的属性数据,最后一列是决策属性,不需要列些属性名称,直接就是数据就可以,以 mat 文件形式放在工作空间中。

**需要注意的是**,在约简之前,必须要把数据归一化处理,否则处理结果会出错!网上现有的一些归一化程序都是整体归一化,这样等同于没有处理。所以我自己写了一个对每个属性列单独进行归一化的程序:

```
function Normaldata = Data_normalized_suo(Normaldata)
[m,n] = size(Normaldata);
for j=1:n
    amin=min(Normaldata(:,j));
    amax=max(Normaldata(:,j));
    for i=1:m
        Normaldata(i,j)=(Normaldata(i,j)-amin)./(amax-amin);
    end
end
```

如果需要源文件的可以到 CSDN 上下载,我上传到那上了。

下载地址: <http://download.csdn.net/detail/buaasuozi/4905485>

当然,你也可以从这里 down 了,生成.m 文件使用。

然后进入胡清华的程序内部,里面对不同的选择做了不同的算法使用:

%%%%%%%%%%不同的策略,速度有差别

```
switch fun_dpd
    case 'clsf_dpd_fast' %优化算法,距离计算完了,2 范数----较快
        [w,e]=clsf_dpd_fast(array_tmp,delta,sample_lft);
```

```

case 'clsf_dpd_fast2'      %优化算法,距离没计算完,2 范数----最快
    [w,e]=clsf_dpd_fast2(array_tmp,delta,sample_lft);
case 'clsf_dpd'           %优化算法,速度慢, 用于对比,2 范数----最慢
    [w,e]=clsf_dpd(array_tmp,delta,sample_lft);
case 'clsf_dpd_fast_3'    %优化算法,无穷范数 输出的 w 是
dependency, 输出的 e 是 smp_csst
    [w,e]=clsf_dpd_fast_3(array_tmp,delta,sample_lft);

end

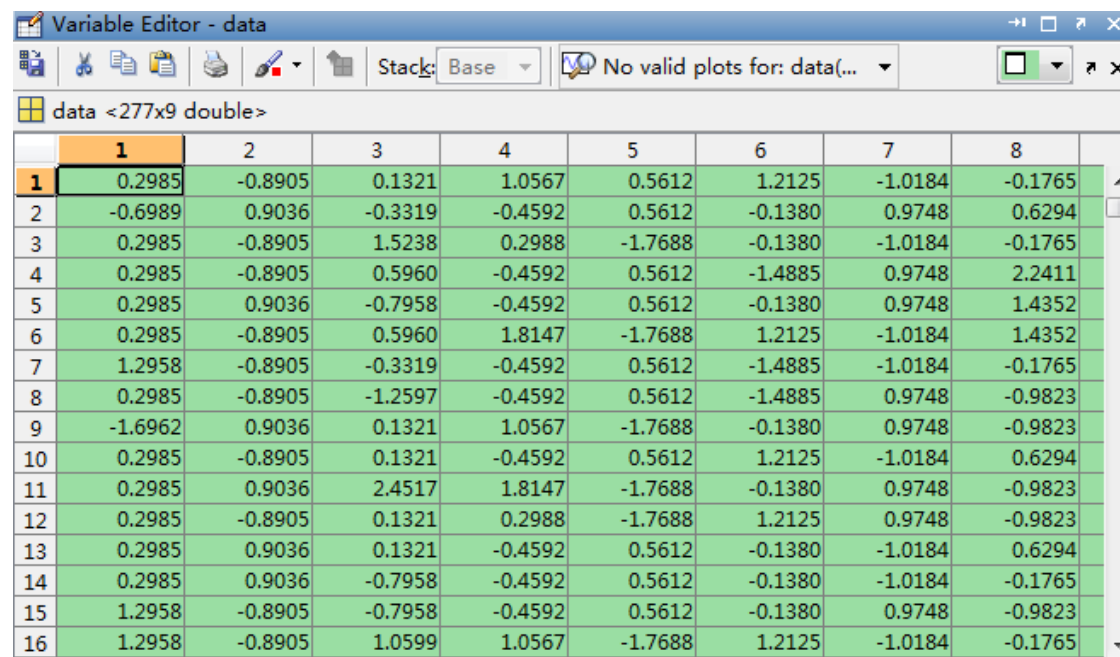
```

这样就可以完成计算了。

下面我主要演示不同邻域半径下的约简计算,也就是我做了改动的约简计算:

这是 breast 原始数据:

9 个条件属性, 决策属性分类为: 2 类。每个属性样本数据的标准差各异, 最大值为 0.5004, 最小值为 0.1422。



	1	2	3	4	5	6	7	8
1	0.2985	-0.8905	0.1321	1.0567	0.5612	1.2125	-1.0184	-0.1765
2	-0.6989	0.9036	-0.3319	-0.4592	0.5612	-0.1380	0.9748	0.6294
3	0.2985	-0.8905	1.5238	0.2988	-1.7688	-0.1380	-1.0184	-0.1765
4	0.2985	-0.8905	0.5960	-0.4592	0.5612	-1.4885	0.9748	2.2411
5	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	0.9748	1.4352
6	0.2985	-0.8905	0.5960	1.8147	-1.7688	1.2125	-1.0184	1.4352
7	1.2958	-0.8905	-0.3319	-0.4592	0.5612	-1.4885	-1.0184	-0.1765
8	0.2985	-0.8905	-1.2597	-0.4592	0.5612	-1.4885	0.9748	-0.9823
9	-1.6962	0.9036	0.1321	1.0567	-1.7688	-0.1380	0.9748	-0.9823
10	0.2985	-0.8905	0.1321	-0.4592	0.5612	1.2125	-1.0184	0.6294
11	0.2985	0.9036	2.4517	1.8147	-1.7688	-0.1380	0.9748	-0.9823
12	0.2985	-0.8905	0.1321	0.2988	-1.7688	1.2125	0.9748	-0.9823
13	0.2985	0.9036	0.1321	-0.4592	0.5612	-0.1380	-1.0184	0.6294
14	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	-1.0184	-0.1765
15	1.2958	-0.8905	-0.7958	-0.4592	0.5612	-0.1380	0.9748	-0.9823
16	1.2958	-0.8905	1.0599	1.0567	-1.7688	1.2125	-1.0184	-0.1765

1、进行归一化:



```

Command Window

>> Normaldata = Data_normalized_suo(data)

Normaldata =

Columns 1 through 3

    0.59999999598928    0.500000001393396    0.500000002155529
    0.39999999598928           1    0.400000004311059
    0.59999999598928    0.500000001393396    0.799999995688941
    0.59999999598928    0.500000001393396           0.6
    0.59999999598928           1    0.300000004311059
    0.59999999598928    0.500000001393396           0.6
           0.8    0.500000001393396    0.400000004311059
    0.59999999598928    0.500000001393396    0.200000004311059
           0.2           1    0.500000002155529
    0.59999999598928    0.500000001393396    0.500000002155529
    0.59999999598928           1           1
    0.59999999598928    0.500000001393396    0.500000002155529
    0.59999999598928           1    0.500000002155529
    0.59999999598928           1    0.300000004311059
           0.8    0.500000001393396    0.300000004311059
           0.8    0.500000001393396    0.700000004311059
    0.59999999598928           1           0.6
    0.39999999598928           1    0.400000004311059
    0.59999999598928    0.500000001393396    0.200000004311059
           0.8    0.500000001393396           1
           0.2           0    0.300000004311059
           0.2           1           0
    0.59999999598928    0.500000001393396    0.400000004311059
           0.8    0.500000001393396    0.100000012933177
           0.8    0.500000001393396    0.200000004311059

```

注意，决策属性可以不进行归一化，归一化也无妨。

## 2、整合

然后整合到一起，条件属性放在前边，决策属性放在后边：

breast <277x10 double>									
1	2	3	4	5	6	7	8	9	
1	0.6000	0.5000	0.5000	0.2500	1	1	0	0.2500	0
2	0.4000	1	0.4000	0	1	0.5000	1	0.5000	1
3	0.6000	0.5000	0.8000	0.1250	0	0.5000	0	0.2500	1
4	0.6000	0.5000	0.6000	0	1	0	1	1	1
5	0.6000	1	0.3000	0	1	0.5000	1	0.7500	1
6	0.6000	0.5000	0.6000	0.3750	0	1	0	0.7500	0
7	0.8000	0.5000	0.4000	0	1	0	0	0.2500	1
8	0.6000	0.5000	0.2000	0	1	0	1	0	1
9	0.2000	1	0.5000	0.2500	0	0.5000	1	0	0
10	0.6000	0.5000	0.5000	0	1	1	0	0.5000	1
11	0.6000	1	1	0.3750	0	0.5000	1	0	1
12	0.6000	0.5000	0.5000	0.1250	0	1	1	0	1
13	0.6000	1	0.5000	0	1	0.5000	0	0.5000	1
14	0.6000	1	0.3000	0	1	0.5000	0	0.2500	1
15	0.8000	0.5000	0.3000	0	1	0.5000	1	0	1
16	0.8000	0.5000	0.7000	0.2500	0	1	0	0.2500	1
17	0.6000	1	0.6000	0.1250	0	0.5000	0	0.2500	0

我这里的决策属性进行了归一化，所以第 10 列就是 0 和 1.

### 3、计算平均差

然后计算平均差：

```
>> std_breast=std(breast)

std_breast =

Columns 1 through 3

    0.20202501262366    0.267741483118812    0.213245161402869

Columns 4 through 6

    0.142213571077581    0.402342328290347    0.364994307016493

Columns 7 through 9

    0.500353038132874    0.30084509385109    0.417561692405915

Column 10

    0.455696947600168

>>
```

这里的计算可以把决策属性一并计算，无所谓的。

### 4、计算邻域半径

然后计算邻域半径，按照公式： $\text{std}/\text{lammda}$

Lammda 经过我的试验（我的试验具体结果就不告诉你们了，谅解），取值建议在 3.0-4.0 之间。不过也可以根据实际情况取值，这个不固定。当然可以随着自己的实际情况调整，然后得出结果，看满意否，再做决定。

下图为邻域半径计算结果：

```
>> delta=std_breast./3

delta =

Columns 1 through 3

    0.0673416708745533    0.0892471610396041    0.0710817204676229

Columns 4 through 6

    0.0474045236925272    0.134114109430116    0.121664769005498

Columns 7 through 9

    0.166784346044291    0.100281697950363    0.139187230801972

Column 10

    0.151898982533389

>> |
```

## 5、约简计算

然后就可以根据我修改过的程序进行计算了：

我修改过的程序直接应用了最后一个约简算法： `clsf_dpd_fast_3`

得到的约简结果：

```
>> [feature_slct,efficiency]=featureselect_FW_fast(breast,delta,0.01)

feature_slct =

    1     3     8     6     7     4     2

efficiency =

Columns 1 through 3

    0.0216606498194946    0.173285198555957    0.458483754512635

Columns 4 through 6

    0.754512635379061    0.848375451263538    0.909747292418773

Column 7

    0.942238267148014

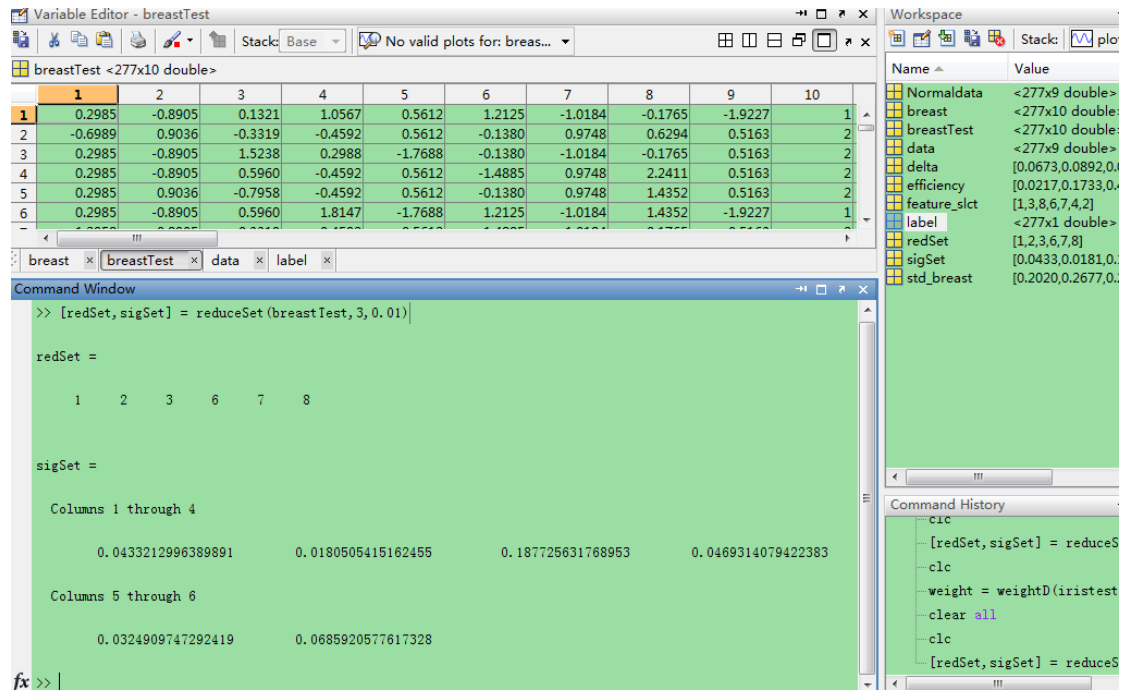
x>> |
```

约简剩余属性为 1、3、8、6、7、4、2.

至此，使用胡程序计算的邻域粗糙集约简计算结束了。

### 3.3.2 索程序测试

下面用我新编写的程序计算一下这个数据的约简，计算结果如下：



约简剩余属性为 1 2 3 6 7 8.同样输出了相应的重要度。

下面尝试调整重要度下限，看看输出的结果。

breastTest <277x10 double>										
	1	2	3	4	5	6	7	8	9	10
1	0.2985	-0.8905	0.1321	1.0567	0.5612	1.2125	-1.0184	-0.1765	-1.9227	1
2	-0.6989	0.9036	-0.3319	-0.4592	0.5612	-0.1380	0.9748	0.6294	0.5163	2
3	0.2985	-0.8905	1.5238	0.2988	-1.7688	-0.1380	-1.0184	-0.1765	0.5163	2
4	0.2985	-0.8905	0.5960	-0.4592	0.5612	-1.4885	0.9748	2.2411	0.5163	2
5	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	0.9748	1.4352	0.5163	2
6	0.2985	-0.8905	0.5960	1.8147	-1.7688	1.2125	-1.0184	1.4352	-1.9227	1
7	1.2958	-0.8905	-0.3319	-0.4592	0.5612	-1.4885	-1.0184	-0.1765	0.5163	2
8	0.2985	-0.8905	-1.2597	-0.4592	0.5612	-1.4885	0.9748	-0.9823	0.5163	2
9	-1.6962	0.9036	0.1321	1.0567	-1.7688	-0.1380	0.9748	-0.9823	-1.9227	2
10	0.2985	-0.8905	0.1321	-0.4592	0.5612	1.2125	-1.0184	0.6294	0.5163	2
11	0.2985	0.9036	2.4517	1.8147	-1.7688	-0.1380	0.9748	-0.9823	0.5163	1
12	0.2985	-0.8905	0.1321	0.2988	-1.7688	1.2125	0.9748	-0.9823	0.5163	2
13	0.2985	0.9036	0.1321	-0.4592	0.5612	-0.1380	-1.0184	0.6294	0.5163	1

```

>> [redSet,sigSet] = reduceSet(breastTest,3,0.1)

redSet =

    3

sigSet =

    0.187725631768953

```

当重要度下限为 0.1 时，只有 3 一个条件属性输出了。言外之意，3 最重要。

降低重要度下限后：

breastTest <277x10 double>										
	1	2	3	4	5	6	7	8	9	10
1	0.2985	-0.8905	0.1321	1.0567	0.5612	1.2125	-1.0184	-0.1765	-1.9227	1
2	-0.6989	0.9036	-0.3319	-0.4592	0.5612	-0.1380	0.9748	0.6294	0.5163	2
3	0.2985	-0.8905	1.5238	0.2988	-1.7688	-0.1380	-1.0184	-0.1765	0.5163	2
4	0.2985	-0.8905	0.5960	-0.4592	0.5612	-1.4885	0.9748	2.2411	0.5163	2
5	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	0.9748	1.4352	0.5163	2
6	0.2985	-0.8905	0.5960	1.8147	-1.7688	1.2125	-1.0184	1.4352	-1.9227	1
7	1.2958	-0.8905	-0.3319	-0.4592	0.5612	-1.4885	-1.0184	-0.1765	0.5163	2
8	0.2985	-0.8905	-1.2597	-0.4592	0.5612	-1.4885	0.9748	-0.9823	0.5163	2
9	-1.6962	0.9036	0.1321	1.0567	-1.7688	-0.1380	0.9748	-0.9823	-1.9227	2
10	0.2985	-0.8905	0.1321	-0.4592	0.5612	1.2125	-1.0184	0.6294	0.5163	2
11	0.2985	0.9036	2.4517	1.8147	-1.7688	-0.1380	0.9748	-0.9823	0.5163	1
12	0.2985	-0.8905	0.1321	0.2988	-1.7688	1.2125	0.9748	-0.9823	0.5163	2
13	0.2985	0.9036	0.1321	-0.4592	0.5612	-0.1380	-1.0184	0.6294	0.5163	1

```

>> [redSet,sigSet] = reduceSet(breastTest,3,0.001)

redSet =

    1    2    3    5    6    7    8    9

sigSet =

Columns 1 through 4

    0.0433212996389891    0.0180505415162455    0.187725631768953    0.00361010830324915

Columns 5 through 8

    0.0469314079422383    0.0324909747292419    0.0685920577617328    0.00722021660649819

```

可以看出，输出的结果增加了很多。

接下来计算一下权重：

breastTest <277x10 double>

	1	2	3	4	5	6	7	8	9	10
1	0.2985	-0.8905	0.1321	1.0567	0.5612	1.2125	-1.0184	-0.1765	-1.9227	1
2	-0.6989	0.9036	-0.3319	-0.4592	0.5612	-0.1380	0.9748	0.6294	0.5163	2
3	0.2985	-0.8905	1.5238	0.2988	-1.7688	-0.1380	-1.0184	-0.1765	0.5163	2
4	0.2985	-0.8905	0.5960	-0.4592	0.5612	-1.4885	0.9748	2.2411	0.5163	2
5	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	0.9748	1.4352	0.5163	2
6	0.2985	-0.8905	0.5960	1.8147	-1.7688	1.2125	-1.0184	1.4352	-1.9227	1
7	1.2958	-0.8905	-0.3319	-0.4592	0.5612	-1.4885	-1.0184	-0.1765	0.5163	2
8	0.2985	-0.8905	-1.2597	-0.4592	0.5612	-1.4885	0.9748	-0.9823	0.5163	2
9	-1.6962	0.9036	0.1321	1.0567	-1.7688	-0.1380	0.9748	-0.9823	-1.9227	2
10	0.2985	-0.8905	0.1321	-0.4592	0.5612	1.2125	-1.0184	0.6294	0.5163	2

breastTest

Command Window

```
>> weight = weightD(breastTest,3)

weight =

    0.106194690265487
    0.0442477876106196
    0.460176991150442
         0
    0.00884955752212402
    0.115044247787611
    0.0796460176991151
    0.168141592920354
    0.0176991150442478
```

更改 lammda=2

breastTest <277x10 double>

	1	2	3	4	5	6	7	8	9	10
1	0.2985	-0.8905	0.1321	1.0567	0.5612	1.2125	-1.0184	-0.1765	-1.9227	1
2	-0.6989	0.9036	-0.3319	-0.4592	0.5612	-0.1380	0.9748	0.6294	0.5163	2
3	0.2985	-0.8905	1.5238	0.2988	-1.7688	-0.1380	-1.0184	-0.1765	0.5163	2
4	0.2985	-0.8905	0.5960	-0.4592	0.5612	-1.4885	0.9748	2.2411	0.5163	2
5	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	0.9748	1.4352	0.5163	2
6	0.2985	-0.8905	0.5960	1.8147	-1.7688	1.2125	-1.0184	1.4352	-1.9227	1
7	1.2958	-0.8905	-0.3319	-0.4592	0.5612	-1.4885	-1.0184	-0.1765	0.5163	2
8	0.2985	-0.8905	-1.2597	-0.4592	0.5612	-1.4885	0.9748	-0.9823	0.5163	2
9	-1.6962	0.9036	0.1321	1.0567	-1.7688	-0.1380	0.9748	-0.9823	-1.9227	2
10	0.2985	-0.8905	0.1321	-0.4592	0.5612	1.2125	-1.0184	0.6294	0.5163	2

breastTest

Command Window

```
>> weight = weightD(breastTest,2)

weight =

    0.189873417721519
    0.069620253164557
    0.208860759493671
    0.0443037974683545
    0.018987341772152
    0.164556962025316
    0.0822784810126582
    0.183544303797468
    0.0379746835443037
```

breastTest <277x10 double>										
	1	2	3	4	5	6	7	8	9	10
1	0.2985	-0.8905	0.1321	1.0567	0.5612	1.2125	-1.0184	-0.1765	-1.9227	1
2	-0.6989	0.9036	-0.3319	-0.4592	0.5612	-0.1380	0.9748	0.6294	0.5163	2
3	0.2985	-0.8905	1.5238	0.2988	-1.7688	-0.1380	-1.0184	-0.1765	0.5163	2
4	0.2985	-0.8905	0.5960	-0.4592	0.5612	-1.4885	0.9748	2.2411	0.5163	2
5	0.2985	0.9036	-0.7958	-0.4592	0.5612	-0.1380	0.9748	1.4352	0.5163	2
6	0.2985	-0.8905	0.5960	1.8147	-1.7688	1.2125	-1.0184	1.4352	-1.9227	1
7	1.2958	-0.8905	-0.3319	-0.4592	0.5612	-1.4885	-1.0184	-0.1765	0.5163	2
8	0.2985	-0.8905	-1.2597	-0.4592	0.5612	-1.4885	0.9748	-0.9823	0.5163	2
9	-1.6962	0.9036	0.1321	1.0567	-1.7688	-0.1380	0.9748	-0.9823	-1.9227	2
10	0.2985	-0.8905	0.1321	-0.4592	0.5612	1.2125	-1.0184	0.6294	0.5163	2
11	0.2985	0.9036	2.4517	1.8147	-1.7688	-0.1380	0.9748	-0.9823	0.5163	1
12	0.2985	-0.8905	0.1321	0.2988	-1.7688	1.2125	0.9748	-0.9823	0.5163	2
13	0.2985	0.9036	0.1321	0.4592	0.5612	0.1380	1.0184	0.6294	0.5163	1

Command Window										
<pre>&gt;&gt; [redSet, sigSet] = reduceSet(breastTest, 2, 0.05)</pre>										
redSet =										
<div>1    3    6    8</div>										
sigSet =										
<div>0.108303249097473      0.11913357400722      0.0938628158844765      0.104693140794224</div>										

### 3.3.3 对比说明

(1) 这里我先说明一下，胡程序中计算的属性的重要度不是我们所说的那个重要度，这个大家可以研究一下，不过他的计算有他的意义，他的重点不是输出重要度，而是输出约简集合。

(2) 胡的程序在计算之前，需要进行归一化处理。我编写的程序无需这步操作，我直接把归一化写入程序内部了。

(3) 我这里也是通过给定不同的重要度下限，会得到不同的约简结果。所以也就没有验证分类精度的必要了，总归是会通过设置不同的 lammda 参数和重要度下限，得到不同的约简结果，所以分类精度也会随着这些参数的改变而改变。在实际数据应用过程中，大家可以根据自己的实际情况，通过设置不同的 lammda 参数和重要度下限，达到最理想的分类精度要求。

分类精度的验证，大家可以用 weka 软件，利用里面的决策树 C4.5、朴素贝叶斯或者是支持向量机进行十折交叉验证，在此不做重点说明。

至此，所有的邻域粗糙集的正域计算、约简、权重计算都介绍完了。如果

大家想得到其中的个别结果，例如得到依赖度等参数，大家可以根据自己的需求，在 m 文件的 `function` 修改，得到不同的输出。

关于经典粗糙集的约简算法这里不做介绍（东北大学的张雪峰做过这样的工具箱，在网上能下载到相关程序和程序使用说明）。另外，胡清华原版的程序使用这里也不做讲解了，会了这个，那个也就不难了。

## 4 结束语

随后我会把我使用的程序发布到 CSDN 上，供大家使用，也可以到我的空间去找。

CSDN 空间：<http://my.csdn.net/buaasuozi>

如果你没有 CSDN 账号和积分，你也可以加我 qq，我把原始资料传给你。

欢迎大家能够交流学习，知识是无界的，但知识是有版权的。在这样的国度，我还是比较向往那种没有被扭曲的学术界景象……

如果上述内容存在错误，欢迎给我留言，提出意见或建议。

留言方式就随便了，我有新浪微博，不过不怎么上，搜索 buaasuozi 就能搜出来了。也有校内、朋友网等等。邮箱，qq，微信，百度空间，百度知道等等。大家共同学习进步！

最后，如果大家有什么关于邻域粗糙集或者相关知识的创新或成就，希望能与我共享，多谢！

Made by buaa-hitsuozi

第一版本发布时间：2013-07-06

第二版本发布时间：2013-04-30