机器学习学习报告

周珊琳

上海电力大学
上海, 2019-08-16, 中国

第一节　算法概述

算法名称：约简算法

算法原理：设 U 为所讨论对象的非空有限集论域，R 为非空的属性有限集，则称二元有序组 $K=(U,R)$ 为一个知识库，亦称近似空间。在知识库中可能含有冗余的知识，知识约简是研究知识库中哪些知识是必要的，以及在保持分类能力不变的前提下，删除冗余的知识。特别是，当信息系统中的数据是随机采集的其冗余性更为普遍。知识约简是粗糙集理论的核心内容之一，在信息系统分析与数据挖掘等领域具有重要的应用意义。

在一个知识系统中，不同的属性具有的重要程度是不同的。在传统的数据分析中，这种重要性需要事先假设，一般有领域专家给出的权重表示，具有一定的主观色彩。在粗糙集方法中，不需要事先假定的信息（先验知识)，利用决策表中的数据可以计算其属性的重要性。

判断属性重要性的方法: 从决策表中去掉一些属性，再来考虑没有该属性后分类会怎样变化：若去掉该属性会相应地改变分类，则说明该属性的强度大，而重要性高: 反之说明该属性的强度小，即重要性低。

对于属性的重要性可以利用依赖度 $r_p(Q)$ 来描述。对于属性集 D 导出的分类属性集 $B^{'} \subseteq B$ 的重要性，采用两者的依赖度的差来度量，即 $r_B(D)-r_{B-B'}(D)$。这表示从集合 B 中去掉某些属性子集进行分类时，分类 U/D 的正域将会受到怎样的影响。

第二节　相关理论

粗糙集理论：在粗糙集理论中，知识被认为是一种分类能力. 人们的行为基本是分辨现实的或抽象的对象的能力. 假定我们起初对论域内的对象（或称元素、样木、个体) 已具有必要的信息或知识，通过这些知识能够将其划分到不同的类别。若我们对两个对象具有相同的信息，则它们是不可区分的，即根据已有的信息不能将其划分开。粗糙集理论的核心是等价关系，通常用等价关系替代分类，根据这个等价关系划分样本集合为等价类。从知识库的观点看，每个等价类被称为一个概念，即一条知识 (规则)。即，每个等价类唯一地表示了一个概念，属于一个等价类的不同对象对该概念是不可区分的。

基本集合：由论域中相互不可分辨的对象组成的集合称之为基本集合，它是组成论域知识的颗粒。

下近似集：下近似集: 根据现有知识 R，判断 U 中所有肯定属于集合 X 的对象所组成的集合，即 $R\_(X) = \{x \in U, [x]_R \subseteq X\}$ 其中，$[x]_R$ 表示等价关系 R 下包含元素 x 的等价类。

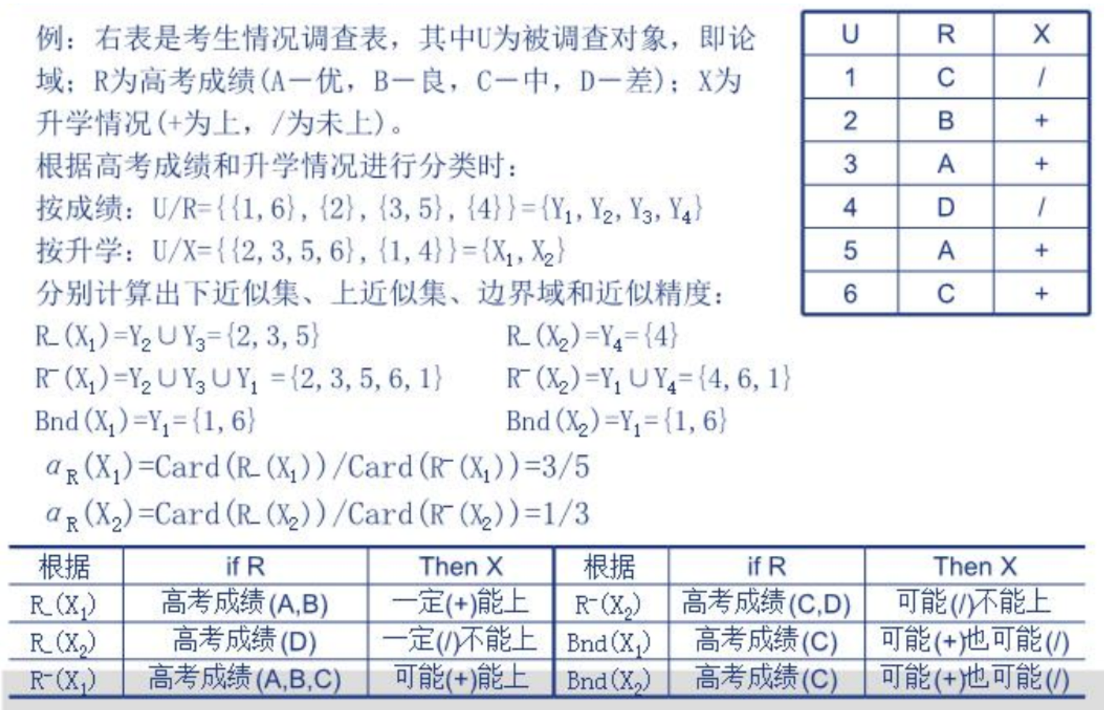
上近似集：根据现有知识 R，判断 U 中一定属于和可能属于集合 X 的对象所组成的集合，即 $R^-(X) = \{x \in U, [x]_R \cap X \neq \phi\}$ 其中，$[x]_R$ 表示等价关系 R 下包含元素 x 的等价类。

正域：$Pos(X) = R\_(X)$，即根据知识 R，U 中能完全确定地归入集合 X 的元素的集合。

负域:$Neg(X) = U - R\_(X)$, 即根据知识 R，U 中能不能确定一定属于集合 X 的元素的集合，它们是 X 的补集。

边界域:$Bnd(X) = R^-(X) - R\_(X)$，边界域是某种意义上论域的不确定域。如果 $Bnd(X)$ 是空集，则称集合 X 关于 R 是清晰的: 反之，如果 $Bnd(X)$ 不是空集，则称集合 X 为关于 R 的粗糙集。因此，粗糙中的“粗糙”主要体现在边界域的存在。集合 X 的边界域越大，其确定性程度就越小。

粗糙度: 对于知识 R(即属性子集)，样本子集 X 的不确定程度可以用粗糙度 $\alpha_R(X)$ 来表示为 $\alpha_R(X) = \frac{Card(R\_(X))}{Card(R^-(X))}$



例：右表是考生情况调查表，其中U为被调查对象，即论域；R为高考成绩(A—优，B—良，C—中，D—差)；X为升学情况(+为上，/为未上)。

U	R	X
1	C	/
2	B	+
3	A	+
4	D	/
5	A	+
6	C	+

根据高考成绩和升学情况进行分类时：

按成绩：$U/R=\{\{1,6\},\{2\},\{3,5\},\{4\}\}=\{Y_1,Y_2,Y_3,Y_4\}$

按升学：$U/X=\{\{2,3,5,6\},\{1,4\}\}=\{X_1,X_2\}$

分别计算出下近似集、上近似集、边界域和近似精度：

$R\_(X_1)=Y_2\cup Y_3=\{2,3,5\}$　　$R\_(X_2)=Y_4=\{4\}$

$R^-(X_1)=Y_2\cup Y_3\cup Y_1=\{2,3,5,6,1\}$　　$R^-(X_2)=Y_1\cup Y_4=\{4,6,1\}$

$Bnd(X_1)=Y_1=\{1,6\}$　　$Bnd(X_2)=Y_1=\{1,6\}$

$\alpha_R(X_1)=Card(R\_(X_1))/Card(R^-(X_1))=3/5$

$\alpha_R(X_2)=Card(R\_(X_2))/Card(R^-(X_2))=1/3$

根据	if R	Then X	根据	if R	Then X
$R\_(X_1)$	高考成绩(A,B)	一定(+)能上	$R^-(X_2)$	高考成绩(C,D)	可能(/)不能上
$R\_(X_2)$	高考成绩(D)	一定(/)不能上	$Bnd(X_1)$	高考成绩(C)	可能(+)也可能(/)
$R^-(X_1)$	高考成绩(A,B,C)	可能(+)能上	$Bnd(X_2)$	高考成绩(C)	可能(+)也可能(/)

图 1: 具体案例 (引自华北电力大学智能信息处理技术 PPT)

依赖度:$k = r_P(Q) = Card(Pos_P(Q))/Card(U)$ 记作 P=>kQ：当 k=1 时，称知识 Q

完全依赖于知识 P; 当 0<k<1 时，称知识 Q 部分依赖于知识 P；当 k=0 时，称知识 Q 完全独立于知识 P。依赖度反映了根据知识 P 将对象分类到 Q 的基本概念中去的能力。确切的说，当 P=kQ 时，论域中共有 k*Card(U) 个属于 Q 的 P 正域的对象，这些对象可以依据知识 P 分类到知识 Q 的基本概念中去。

例 $U=\{x_1, x_2, \cdots, x_8\}$，$U/P=\{\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_5, x_6\}, \{x_7, x_8\}\}$，$U/Q=\{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5, x_6\}, \{x_7\}, \{x_8\}\}$，求依赖度k。

解：$Pos_P(Q)=\{x_1\} \cup \{x_2\} \cup \{x_3, x_4\} \cup \{x_5, x_6\}=\{x_1, x_2, x_3, x_4, x_5, x_6\}$

$k=6/8=0.75$

即知识Q相对于知识P的依赖度为0.75

图 2: 具体案例 (引自华北电力大学智能信息处理技术 PPT)

相对约简：在实际应用中，一个分类相对于另一个分类的关系非常重要。在粗糙集中相对约简的概念，即条件属性相对决策属性的约简。设 P 和 Q 为论域 U 上的等价关系，Q 的 P 正域记为 $Pos_p(Q)$，即 $Pos_p(Q) = \bigcup P\_(X)$，Q 的 P 正域是论域 U 中的所有那些使用分类 U/P 所表达的知识，能够正确地划入到 U/Q 的等价类之中的对象给出的集合。

例 设$K=\{U,P\}$为知识库，$U=\{x_1, x_2, \cdots, x_8\}$，$P=\{R_1, R_2, R_3\}$。等价关系$R_1, R_2, R_3$类集合如下：

$U/R_1=\{(x_1, x_2, x_3, x_4\}, \{x_5, x_6, x_7, x_8\}\}$

$U/R_2=\{(x_1, x_3, x_4, x_7\}, \{x_2, x_6\}, \{x_5, x_8\}\}$

$U/R_3=\{(x_1, x_5, x_8\}, \{x_2, x_3, x_4\}, \{x_6, x_7\}\}$

由等价关系族Q导出的不可分辨关系的等价类集合为

$U/Q=U/Ind(Q)=\{\{x_1, x_3, x_4\}, \{x_2, x_5, x_6\}, \{x_7, x_8\}\}$

求P的Q约简及P的Q核。

解：等价关系族P导出的不可分辨关系Ind(P)的等价类为

$U/P=U/Ind(P)=\{\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_5, x_8\}, \{x_6\}, \{x_7\}\}$

Q的P正域为

$Pos_P(Q)=\{x_1\} \cup \{x_2\} \cup \{x_3, x_4\} \cup \{x_6\} \cup \{x_7\}=\{x_1, x_2, x_3, x_4, x_6, x_7\}$

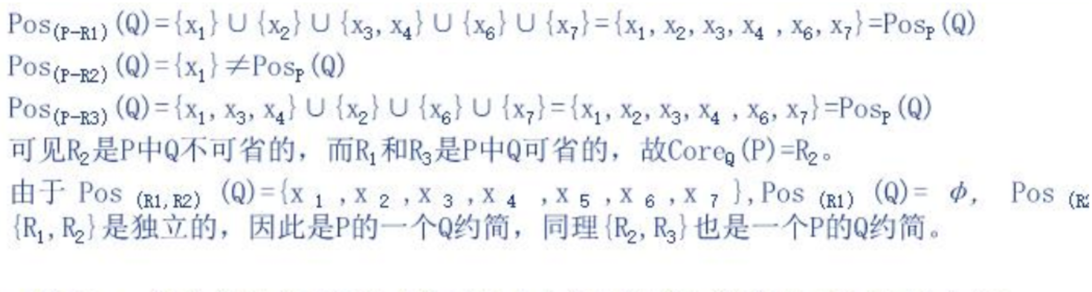
P中不可省的关系为

$U/(P-R_1)=U/(R_2, R_3)=\{\{x_1\}, \{x_2\}, \{x_3, x_4\}, \{x_5, x_8\}, \{x_6\}, \{x_7\}\}$

$U/(P-R_2)=U/(R_1, R_3)=\{\{x_1\}, \{x_2, x_3, x_4\}, \{x_5, x_8\}, \{x_6, x_7\}\}$

$U/(P-R_3)=U/(R_1, R_2)=\{\{x_1, x_3, x_4\}, \{x_2\}, \{x_5, x_8\}, \{x_6\}, \{x_7\}\}$

图 3: 具体案例 (引自华北电力大学智能信息处理技术 PPT)



$Pos_{(P-R1)}(Q)=\{x_1\}\cup\{x_2\}\cup\{x_3, x_4\}\cup\{x_6\}\cup\{x_7\}=\{x_1, x_2, x_3, x_4, x_6, x_7\}=Pos_P(Q)$

$Pos_{(P-R2)}(Q)=\{x_1\}\neq Pos_P(Q)$

$Pos_{(P-R3)}(Q)=\{x_1, x_3, x_4\}\cup\{x_2\}\cup\{x_6\}\cup\{x_7\}=\{x_1, x_2, x_3, x_4, x_6, x_7\}=Pos_P(Q)$

可见R_2是P中Q不可省的，而R_1和R_3是P中Q可省的，故$Core_Q(P)=R_2$。

由于 $Pos_{(R1,R2)}(Q)=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$，$Pos_{(R1)}(Q)=\phi$， Pos (R

$\{R_1, R_2\}$是独立的，因此是P的一个Q约简，同理$\{R_2, R_3\}$也是一个P的Q约简。

图 4: 具体案例 (引自华北电力大学智能信息处理技术 PPT)

第三节　算法设计

3.1　算法流程

Algorithm 1 约简算法

1: 数据预处理，离散归一化。运用粗糙处理决策表时，要求决策表中的值用离散数据表达。因此在智能信息处理中，对定性的属性或属性的值域是连续的数据要进行预先处理，将其离散化，转换为粗糙集理论所识别的数据，从而提取有用信息，从中发现知识。

2: 数据约简：计算各类的基本集合，计算各类正域，计算依赖度，根据依赖度差值判断重要性

例 某一知识表达系统如表所示。计算表中属性a, b, c相对属性d, e的重要性。

U	a	b	c	d	e
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

解：定义C={a, b, c}, D={d, e}，则可以构成各种分类：

U/(b, c)={{1, 5}, {2, 7, 8}, {3}, {4}, {6}}

U/(a, c)={{1, 5}, {2, 8}, {3, 6}, {4}, {7}}

U/(a, b)={{1, 5}, {2, 8}, {3}, {4}, {6}, {7}}

U/(a, b, c)={{1, 5}, {2, 8}, {3}, {4}, {6}, {7}}

U/(d, e)={{1}, {2, 7}, {3, 6}, {4}, {5, 8}}

图 5: 具体案例 (引自华北电力大学智能信息处理技术 PPT)

$Pos_C(D)=\{3, 4, 6, 7\}$

$Pos_{C-a}(D)=\{3, 4, 6\}$

$Pos_{C-b}(D)=\{3, 4, 6, 7\}$

$Pos_{C-c}(D)=\{3, 4, 6, 7\}$

故

$r_C(D)=Card(Pos_C(D))/Card(U)=4/8=0.5$

$r_{C-a}(D)=Card(Pos_{C-a}(D))/Card(U)=3/8=0.375$

$r_{C-b}(D)=Card(Pos_{C-b}(D))/Card(U)=4/8=0.5$

$r_{C-c}(D)=Card(Pos_{C-c}(D))/Card(U)=4/8=0.5$

因此

$r_C(D) - r_{C-a}(D) = 0.125$

$r_C(D) - r_{C-b}(D) = 0$

$r_C(D) - r_{C-c}(D) = 0$

可知，属性a是最重要的，其将U/D的正域改变的最多；属性b和c无关紧要，去掉它们后，分类依赖度未产生变化。

图 6: 具体案例 (引自华北电力大学智能信息处理技术 PPT)

3.2 核心代码

源代码 1:

```

from csv import reader
import numpy as np
from itertools import combinations

def load_csv(filename):
	dataset =list()
	with open(filename, 'r') as file:
		csv_reader =reader(file)
		for row in csv_reader:
			if not row:
				continue
			dataset.append(row)
	return dataset

# Convert string column to float
def str_column_to_float(dataset, column):
	for row in dataset:
		try:
			row[column] =float(row[column].strip())
		except ValueError:
			print("Error with row",column,":",row[column])
			pass

# Convert string column to integer
```

```python
def str_column_to_int(dataset, column):
    for row in dataset:
        row[column] =int(row[column])


def dependency(dataset,num):
    total=0
    dependency=0
    for j in range(3):
        fold=list()
        for i in range(len(dataset)):
            data=dataset[i]
            #print(i)
            if data[-1]==j:
                fold.append(dataset[i])
        #print("Fold {}".format(fold))
        count=len(fold)
        #print("count {}".format(count))
        for k in range(len(fold)):
            list1=fold[k]
            for l in range(len(dataset)):
                #print("len{}".format(len(fold)))
                list2=dataset[l]
                if list1[:num]==list2[:num] and list1[-1]!=list2[-1]:
                    count =count-1
                    #print("Count inside {}".format(count))
                    break
        total=total+count
        #print("total {}",format(total))
    dependency=total/len(dataset)
    #print("{}".format(dependency))
    return dependency

def generate_new_dataset(row,l):
    #print(row)
    X =np.empty((8, 0))
    #print(X)
    for i in range(len(row)):
        col=row[i]
        x=[row_new[col] for row_new in dataset]
        x=np.array([x])
        #print(np.transpose(x))
        #print(x)
        x=x.T
        X=np.append(X, x, axis=1)
    x=[row[-1] for row in dataset]
    X=np.append(X, [[x[0]],[x[1]],[x[2]],[x[3]],[x[4]],[x[5]],[x[6]],[x[7]]], axis=1)
    X=np.array(X).tolist()
    #print("X {}".format(X))
    return X


filename ='TestData.csv'
dataset =load_csv(filename)
```

```
for i in range(len(dataset[0])-1):
	str_column_to_float(dataset, i)
# convert class column to integers
str_column_to_int(dataset, len(dataset[0])-1)
#print(dataset)
#this is fuzzify input based on class belongin granulation
dp=dependency(dataset,4)

n=4
initial_val=[0,1,2,3]
comb=combinations([0,1,2,3],3)

co=[i for i in combinations([0,1,2,3],3)]
c=len(co)

while n>1 :
	for row in comb:

		data_X=generate_new_dataset(row,len(row))
		#print(data_X)
		dp_data_X=dependency(data_X,len(row))

		if dp >dp_data_X:
			c=c-1
			continue
		else:
			n=n-1
			prev_row=row

			break
	if c ==0:
		break
	comb=combinations(row,n)
	comb_temp=[i for i in combinations(row,n)]
	c=len(comb_temp)


print("final reduct {}".format(prev_row))
```

第四节　选用数据

TestData 行数：8. 列数：5

	A	B	C	D	E
1	1	0	2	2	0
2	0	1	1	1	2
3	2	0	0	1	1
4	1	1	0	2	2
5	1	0	2	0	1
6	2	2	0	1	1
7	2	1	1	1	2
8	0	1	1	0	1

图 7: 数据展示

第五节　实验结果展示

```
final reduct (1, 3)

In [21]:
```

图 8: 约简结果展示

第六节　遇到的问题及解决方法，实践心得

问题：关于粗糙理论的小知识很多，容易搞混；关于分辨函数具体如何执行的不是很懂。

心得：这次的算法，主要时间花在了基础知识的理解学习上，对于知识点，我是先通读概念然后根据具体实例用笔演算再回头看概念解释进行理解，因此也花了很长的时间，经常

会在具体应用的时候搞混，回头从看概念，这也是我没能实现代码的原因之一吧，对基本概念掌握的还不是很熟。