



“华为杯”第十五届中国研究生 数学建模竞赛

学 校 中国科学院大学

参赛队号 18144300063

队员姓名 1. 邢丽超
2. 张中一
3. 陈惠琴



“华为杯”第十五届中国研究生 数学建模竞赛

题 目 基于数据挖掘的恐怖袭击事件记录量化分析

摘 要:

恐怖袭击事件给人民生命、健康和财产带来极大危害,扰乱了正常社会生产和生活秩序,如何快速识别和划分恐怖袭击事件的级别与类型,从而采取相应的应急措施,并进行合理规划与有效管理,将事件的危害性降到最低是亟待解决的问题。本文主要根据全球恐怖主义数据库(GTD)中 1998-2017 年世界上发生恐怖袭击事件的记录,通过对数据进行量化处理,分别从恐怖袭击事件的危害性等级划分问题、无责任恐袭事件的嫌疑人排序问题、未来恐袭事件的预测问题以及恐袭事件的相关性分析四个方面建立相应的数学模型对问题进行分析与求解。

针对问题一,为了对恐怖袭击事件的危害性进行等级划分,进而保证反恐预案的合理执行和应急资源的优化配置,本文采用层次分析法,通过对人员伤亡、经济损失、攻击类型、攻击目标、武器类型和犯罪者数量六项指标进行量化处理与无量纲化处理,建立了基于层次分析法的危害性量化分级模型,根据危害系数将事件划分为五个等级,并列出近二十年危害程度最大的十大事件,其中“911 恐怖袭击”事件位列榜首。最后本文通过对典型事件的危害系数进行分析,列出了每个事件的级别。

针对问题二,为了提高作案者尚未确定事件的破案效率,尽早发现新生或者隐藏的恐怖分子,本文采用 K 均值聚类算法,首先通过数据过滤提取 29 个表征恐怖事件的显著特征,然后对具有相关性的事件基于 related 特征进行强聚类,并基于特征采用 K 均值聚类算法对无责任恐怖事件分别以硬聚类为种子,将 related 特征为空的事件聚合到种子类上的聚类方法和以第一次聚类的结果为单位并基于各类质心进行深度聚类的聚类方法进行两次聚类,建立了基于多深度聚类算法的嫌疑人排序模型,最后通过类的危害系数确定恐怖分子对于具体事件的

嫌疑程度，并对典型事件的五个嫌疑人进行嫌疑程度的定量排序。

针对问题三，为了研究 2015-2017 年恐怖袭击事件发生的主要原因、时空特性、蔓延特性、级别分布等规律，对 2018 年全球或某些重点地区的反恐态势进行预测分析，本文采用时间序列分析方法，建立了基于时间序列分析的恐怖袭击风险预测模型，通过构造 ARIMA(8,1,8)模型，从事件发生的数量、地区以及所使用的武器类型进行预测，并对伊拉克、阿富汗等五个恐袭事件发生频率较高的地区从恐怖组织与武器使用类型方面进行详细地预测分析，由此提出相应的治理措施。在预测过程中，本文融入了时间序列预测中的递推计算思想，将每一步预测的不确定性作为下一次预测迭代的输入要素进行充分考虑，提升了预测精确度。

针对问题四，本文通过对 GTD 数据库中数据的进一步分析，提取事件之间的相关性信息，按照一定规则构造恐怖组织联系网络，绘制其网络拓扑结构图并计算其网络特征参数，得出其度分布符合无标度特征，并由此建立了基于无标度网络的恐怖袭击事件相关性分析模型，以重要事件节点为中心，对事件的潜在关联性和蔓延趋势进行分析，得出了“富者愈富”的蔓延趋势，从复杂网络的角度对事件的危害程度和发展态势进行了分析研究。

关键词：层次分析法 K 均值聚类算法 时间序列分析 无标度网络

目录

1.问题重述	4
1.1 问题的背景	4
1.2 要解决的问题	4
2.模型假设	5
3.符号说明	5
4.问题的分析	5
4.1 基于层次分析法的危害性量化分级模型	5
4.1.1 问题描述及分析	5
4.1.2 模型准备	6
4.1.3 模型建立与求解	7
4.1.4 结果分析	10
4.2 基于多深度聚类算法的嫌疑人排序模型	11
4.2.1 问题描述及分析	11
4.2.2 模型准备	11
4.2.3 模型建立与求解	12
4.2.4 结果分析	15
4.3 基于时间序列分析的恐怖袭击风险预测模型	16
4.3.1 问题描述及分析	16
4.3.2 模型准备	16
4.3.3 模型建立与求解	17
4.3.4 结果分析	22
4.4 基于无标度网络的恐怖袭击事件相关性研究	25
4.4.1 问题描述及分析	25
4.4.2 模型准备	26
4.4.3 模型建立与求解	27
4.4.5 结果分析	30
5.模型评价	31
5.1 模型的优点	31
5.2 模型的缺点	31
6.参考文献	32
7.附录	32

1.问题重述

1.1 问题的背景

恐怖袭击是指极端分子或组织人为制造的、针对但不仅限于平民及民用设施的、不符合国际道义的攻击行为，它不仅具有极大的杀伤性与破坏力，能直接造成巨大的人员伤亡和财产损失，而且还给人们带来巨大的心理压力，造成社会一定程度的动荡不安，妨碍正常的工作与生活秩序，进而极大地阻碍经济的发展。

恐怖主义是人类共同威胁，打击恐怖主义是每个国家应该承担的责任。对恐怖袭击事件相关数据的深入分析有助于加深人们对恐怖主义的认识，为反恐防恐提供有价值的信息支持。

附件 1 选取了某组织搜集整理的全球恐怖主义数据库（GTD）中 1998-2017 年世界上发生的恐怖袭击事件的记录，本文通过建立相应的数学模型，对以下几个问题进行分析和求解。

1.2 要解决的问题

附件 1 选取了某组织搜集整理的全球恐怖主义数据库（GTD）中 1998-2017 年世界上发生的恐怖袭击事件的记录。本文将通过建立相应的数学模型，对以下几个问题进行分析和求解。

问题一：依据危害性对恐怖袭击事件分级

对灾难性事件比如地震、交通事故、气象灾害等等进行分级是社会管理中的重要工作。通常的分级一般采用主观方法，由权威组织或部门选择若干个主要指标，强制规定分级标准，如我国《道路交通事故处理办法》第六条规定的交通事故等级划分标准，主要按照人员伤亡和经济损失程度划分。

但恐怖袭击事件的危害性不仅取决于人员伤亡和经济损失这两个方面，还与发生的时机、地域、针对的对象等等诸多因素有关，因而采用上述分级方法难以形成统一标准。所以，问题要求本文依据附件 1 以及其它有关信息，结合现代信息处理技术，借助数学建模方法建立基于数据分析的量化分级模型，将附件 1 给出的事件按危害程度从高到低分为一至五级，列出近二十年来危害程度最高的十大恐怖袭击事件，并对典型时间的危害级别进行定义。

问题二：依据事件特征发现恐怖袭击事件制造者

附件 1 中有多起恐怖袭击事件尚未确定作案者。如果将可能是同一个恐怖组织或个人在不同时间、不同地点多次作案的若干案件串联起来统一组织侦查，有助于提高破案效率，有利于尽早发现新生或者隐藏的恐怖分子。问题要求本文针对在 2015、2016 年度发生的、尚未有组织或个人宣称负责的恐怖袭击事件，运用数学建模方法寻找上述可能性，即将可能是同一个恐怖组织或个人在不同时间、不同地点多次作案的若干案件归为一类，对应的未知作案组织或个人标记不同的代号，并按该组织或个人的危害性从大到小选出其中的前 5 个，记为 1 号-5 号。

问题三：对未来反恐态势的分析

对未来反恐态势的分析评估有助于提高反恐斗争的针对性和效率。问题要求本文依据附件 1 并结合因特网上的有关信息，建立适当的数学模型，研究近三年

来恐怖袭击事件发生的主要原因、时空特性、蔓延特性、级别分布等规律，进而分析研判下一年全球或某些重点地区的反恐态势，用图/表给出本文的研究结果，提出本文对反恐斗争的见解和建议。

问题四：数据的进一步利用

利用数学建模的思想，对附件 1 中的数据进行进一步的分析，给出相应的模型和方法。

2.模型假设

为了便于问题的研究与模型的建立，本文对题目中的某些条件进行合理化的假设。

- 假设 1：附件 1 中的所有指标都可通过适当的量化准则进行量化；
- 假设 2：用于统计分析的数据均为有效数据，忽略缺失值的影响；
- 假设 3：经济损失包括政府的公共安全财政支出，财物损坏等费用和支出；
- 假设 4：在选取评价指标时，对危害性影响相对较小的因素可以忽略。

3.符号说明

符号	符号说明
a_{ij}	要素 i 与要素 j 重要性比较的结果
CI	一致性指标
c_i	第 i 个指标
w_i	第 i 决策指标的权重系数
E_i	第 i 单元的本项系数的评价得分值
X_i	第 i 单元本项参数的实际值
D	危害系数
ϕ_i	自回归参数
θ_j	滑动平均参数
$\{a_t\}$	白噪声序列

4.问题的分析

4.1 基于层次分析法的危害性量化分级模型

4.1.1 问题描述及分析

该问题要求本文模仿《道路交通事故处理方法》第六条规定的交通事故等级划分标准，对恐怖袭击事件的等级进行划分。但恐怖袭击事件的危害性不仅取决于人员伤亡和经济损失程度这两个方面，还与发生的时机、地域、针对的对象等因素有关。所以本文结合附件 1 所给的数据，通过对比分析，可以提取出影响恐怖袭击事件危害程度的几个关键因素，基于此，利用层次分析法，对恐怖袭击事件的危害程度进行量化分级。

4.1.2 模型准备

层次分析法是指将一个复杂的多目标决策问题作为一个系统,将目标分解为多个目标或准则,进而分解为多指标(或准则、约束)的若干层次,通过定性指标模糊量化方法算出层次单排序(权数)和总排序,以作为目标(多指标)、多方案优化决策的系统方法。

基本原理:层次分析法根据问题的性质和要达到的总目标,将问题分解为不同的组成因素,并按照因素间的相互关联影响以及隶属关系将因素按不同层次聚集组合,形成一个多层次的层次结构模型,从而最终使问题归结为最低层(供决策的方案、措施等)相对于最高层(总目标)的相对重要权值的确定或相对优劣次序的排定。

基本步骤: 以下为层次分析法实现的基本步骤

Step 1: 建立层次结构模型

将决策的目标、考虑的因素(决策准则)和决策对象按它们之间的相互关系分为最高层、中间层和最低层,绘出层次结构图。最高层是指决策的目的、要解决的问题。最低层是指决策时的备选方案。中间层是指考虑的因素、决策的准则。对于相邻的两层,称高层为目标层,低层为因素层。

Step 2: 构造判断(成对比较)矩阵

在确定各层次各因素之间的权重时,如果只是定性的结果,则常常不容易被别人接受,因而 **Santy** 等人提出一致矩阵法,即不把所有因素放在一起比较,而是两两相互比较,对此时采用相对尺度,以尽可能减少性质不同的诸因素相互比较的困难,以提高准确度。如对某一准则,对其下的各方案进行两两对比,并按其重要性程度评定等级。

a_{ij} 为要素 i 与要素 j 重要性比较的结果。表 1 中列出了 **Santy** 给出的 9 个重要性等级及其赋值。按两两比较结果构成的矩阵称作判断矩阵。判断矩阵具有如下性质:

$$a_{ij} = \frac{1}{a_{ji}}$$

判断矩阵元素 a_{ij} 的标度方法如下:

表 1 比例标度表

因素 i 比因素 j	量化值
同等重要	1
稍微重要	3
较强重要	5
强烈重要	7
极端重要	9
两相邻判断的中间值	2, 4, 6, 8

Step 3: 层次单排序及其一致性检验

对应于判断矩阵最大特征根 λ_{max} 的特征向量,经归一化(使向量中各元素之和等于 1)后记为 W 。 W 的元素为同一层次因素对于上一层次因素某因素相对重要性的排序权值,这一过程称为层次单排序。能否确认层次单排序,则需要进行一致性检验,所谓一致性检验是指对 A 确定不一致的允许范围。其中, n 阶一致阵的唯一非零特征根为 n ; n 阶正互反阵 A 的最大特征根 $\lambda \geq n$,当且仅当 $\lambda = n$ 时, A 为一致矩阵。

由于 λ 连续的依赖于 a_{ij} ,则 λ 比 n 大的越多, A 的不一致性越严重,一致性指标用 CI 计算, CI 越小,说明一致性越大。用最大特征值对应的特征向量作为被比较因素对上层某因素影响程度的权向量,其不一致程度越大,引起的判断误差越大。因而可以用 $\lambda - n$ 数值的大小来衡量 A 的不一致程度。定义一致性指标为:

$$CI = \frac{\lambda - n}{n - 1}$$

$CI = 0$, 有完全的一致性; CI 接近于 0, 有满意的一致性; CI 越大, 不一致越严重。

为衡量 CI 的大小, 引入随机一致性指标 RI :

$$RI = \frac{CI_1 + CI_2 + \dots + CI_n}{n}$$

其中, 随机一致性指标 RI 和判断矩阵的阶数有关, 一般情况下, 矩阵阶数越大, 则出现一致性随机偏离的可能性也越大, 其对应关系如表 2:

表 2 平均随机一致性指标 RI 标准值(不同的标准不同, RI 的值也会有微小的差异)

矩阵阶数	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

考虑到一致性的偏离可能是由于随机原因造成的,因此在检验判断矩阵是否具有满意的一致性时,还需将 CI 和随机一致性指标 RI 进行比较,得出检验系数 CR , 公式如下:

$$CR = \frac{CI}{RI}$$

一般地, 如果 $CR < 0.1$, 则认为该判断矩阵通过一致性检验, 否则就不具有满意一致性。

Step 4: 层次总排序及其一致性检验

计算某一层次所有因素对于最高层(总目标)相对重要性的权值,称为层次总排序。这一过程是从最高层次到最低层次依次进行的。

4.1.3 模型建立与求解

Step 1: 运用层次分析法, 将各个指标进行权重大小比较, 构造比较矩阵: 目标层与准则层的关系图如下图所示:

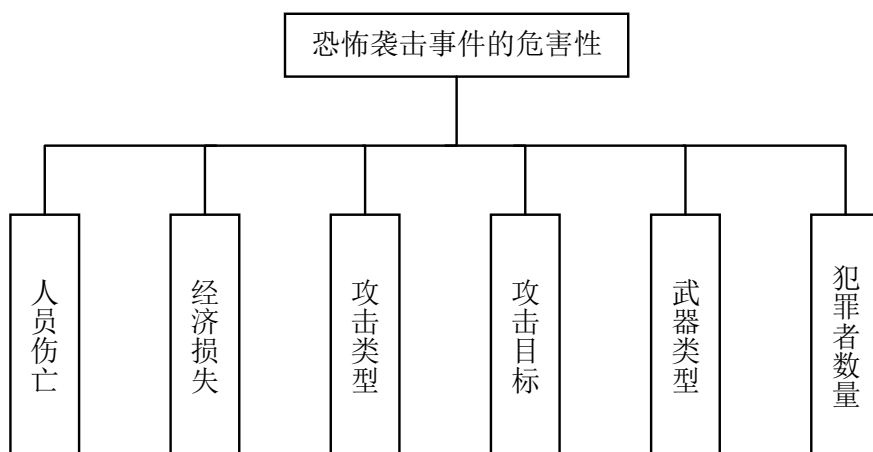


图 1 恐怖袭击事件的危害性准则决策图

c_1 : 人员伤亡, c_2 : 经济损失, c_3 : 攻击类型,
 c_4 : 攻击目标, c_5 : 武器类型, c_6 : 犯罪者数量

根据调查和查阅资料, 对这六个指标的权重进行大小比较:

$$c_i : c_j = a_{ij}; i, j = 1, 2, 3, 4, 5, 6;$$

$$a_{ij} > 0, a_{ij} = \frac{1}{a_{ji}} \quad (1)$$

由此可得表 3:

表 3 权重比较表

目标层	人员伤亡 c_1	经济损失 c_2	攻击类型 c_3	攻击目标 c_4	武器类型 c_5	犯罪者数量 c_6
人员伤亡 c_1	1	5	7	7	5	7
经济损失 c_2	$\frac{1}{5}$	1	3	5	3	3
攻击类型 c_3	$\frac{1}{7}$	$\frac{1}{3}$	1	3	5	3
攻击目标 c_4	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{3}$	1	7	3
武器类型 c_5	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{7}$	1	3
犯罪者数量 c_6	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1

令 $A = (a_{ij})_{6 \times 6}$, 从而可得比较矩阵 A 为

$$A = \begin{pmatrix} 1 & 5 & 7 & 7 & 5 & 7 \\ 1/5 & 1 & 3 & 5 & 3 & 3 \\ 1/7 & 1/3 & 1 & 3 & 5 & 3 \\ 1/7 & 1/5 & 1/3 & 1 & 7 & 3 \\ 1/5 & 1/3 & 1/5 & 1/7 & 1 & 3 \\ 1/7 & 1/3 & 1/3 & 1/3 & 1/3 & 1 \end{pmatrix}_{6 \times 6} \quad (2)$$

Step 2: 计算准则层各个指标所对应的权向量:

$$\begin{aligned} w &= (w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ w_6)^T, \\ \sum_{i=1}^6 w_i &= 1, \\ Aw &= \lambda w; \end{aligned} \quad (3)$$

Step 3: 一致性检验:

一致性指标 CI 的定义公式为:

$$CI = \frac{\lambda - n}{n - 1} \quad (4)$$

$CI = 0$ 时 A 为一致阵, CI 越大 A 的不一致性程度越严重。

随机一致性指标 RI 的数值如表 2 所示。表中 $n=1,2$ 时, $RI=0$, 是因为 1,2 阶的正反矩阵总是一致阵。对于 $n \geq 3$ 的成对比矩阵 A , 将它的一致性指标 CI 与 $n=4$ 的随机一致性指标 RI 相比, 得到一致性比率 CR , 即:

$$CR = \frac{CI}{RI} \quad (5)$$

当 $CR < 0.1$ 时认为 A 的不一致程度在容许的范围内, 可用其特征向量作为权向量。

Step 4: 数据处理:

在该部分, 本文首先从附件 1 中提取出 1998-2017 年能够反映全球恐怖袭击事件的人员伤亡、经济损失、攻击类型、攻击目标、武器类型、犯罪者数量的信息, 对这些信息进行量化处理。

给出评价指标之后, 需对各参数进行无量纲化, 以剔除不同量纲之间的影响。此处选择权重法对各参数进行无量纲化处理, 处理过程如式(6)所示。

$$E_i = \frac{X_i}{\sum X_i} \quad (6)$$

其中: E_i : 第 i 单元的本项系数的评价得分值;

X_i : 第 i 单元本项参数的实际值;

据此可得, 对于人员伤亡、经济损失、攻击类型、攻击目标、武器类型、犯罪者数量这六项指标进行量化处理与无量纲化处理后的数据如表 4 所示。

表 4 量化数据与无量纲化数据

决策指标	量化类型	量化数据	无量纲数据
人员伤亡	轻度人员伤亡	3	0.167
	中度人员伤亡	6	0.333

经济损失	重度人员伤亡	9	0.500
	轻度经济损失	3	0.167
	中度经济损失	6	0.333
	重度经济损失	9	0.500
攻击类型	轰炸/爆炸	7	0.206
	武装袭击	5	0.147
	暗杀	10	0.294
	劫持	9	0.265
	徒手	3	0.088
攻击目标	政府	10	0.278
	军事	8	0.222
	商业	6	0.167
	教育机构	7	0.194
	其他	5	0.139
武器类型	爆炸物/炸药	7	0.250
	轻武器	5	0.179
	化学武器	9	0.321
	燃烧武器	7	0.250
犯罪者数量	集团	9	0.750
	个体	3	0.250

Step 5: 对恐怖袭击事件的危害性进行分级

针对量化评定恐怖袭击事件的留个指标, 本文最终将事件的危害性分为五个等级, 设 D 为恐怖袭击事件的危害系数, 则其等级划分如表 5 所示。

表 5 恐怖袭击事件危害系数等级划分表

危害系 D	$[0,0.2)$	$[0.2,0.4)$	$[0.4,0.6)$	$[0.6,0.8)$	$[0.8,1.0)$
等级	五级	四级	三级	二级	一级

4.1.4 结果分析

根据上述的等级划分规则, 可以得到近二十年来危害程度最高的十大恐怖袭击事件如表 6 所示和典型事件的危害级别如表 7 所示。

表 6 近二十年危害程度最高的十大恐怖袭击事件

事件编号	事件摘要	危害系数
199808070002	美国大使馆爆炸案	0.854
200109110004	911 恐怖袭击	0.889
200210120003	巴厘岛爆炸案	0.851
200210230004	莫斯科剧院人质	0.855
200403020001	伊拉克清真寺连环爆炸	0.857

200403110003	马德里爆炸案	0.852
200409010002	俄罗斯别斯兰人质	0.873
200507070001	伦敦地铁爆炸	0.858
200811260001	印度孟买恐怖袭击	0.855
201304150001	美国波士顿马拉松赛爆炸案	0.856

表 7 典型事件危害级别

事件编号	危害级别
200108110012	一级
200511180002	二级
200901170021	一级
201402110015	三级
201405010071	四级
201411070002	五级
201412160041	二级
201508010015	五级
201705080012	二级

4.2 基于多深度聚类算法的嫌疑人排序模型

4.2.1 问题描述及分析

问题二是要对附件 1 中尚未确定作案人的恐怖袭击事件进行侦查，为案件确定嫌疑人。该题的建议是，如果将可能是同一恐怖组织或个人在不同时间、不同地点多次作案的若干事件串联起来统一组织侦查，有助于提高破案效率，有利于尽早发现新生或者隐藏的恐怖分子。该问题要求本文针对在 2015、2016 年度发生、尚未有组织或者个人宣称负责的恐怖袭击事件，运用数学建模方法寻找上述可能性，即将可能是同一恐怖组织或个人在不同时间、不同地点多次作案的若干事件归为一类，对应的未知作案组织或个人标记不同的代号，并将该组织或个人的危害性从大到小选取其中的前 5 个，记为 1 号-5 号。

基于该问题，本文首先可以对附件中的数据进行过滤，提取对解决该问题有用的信息，然后通过对此聚类，找出类与类之间的相似度，最后可以通过定义此类的危害程度，为判断嫌疑人的可疑程度提供依据。

4.2.2 模型准备

聚类分析是一种定量方法，从数据挖掘的角度看，聚类分析的主体思想就是希望族内的相似度尽可能高，族间的相似度尽可能低。要用数量化的方法对事物进行分类，就要用数量化的方法来定义每个样本的相似程度，这个相似程度在数学上可以称之为距离，最常用的距离有如下几种。

(1) 样本相似性度量：最常用的是欧式距离，设 x 和 y 是两个候选计算相似度的变量，则欧式距离如(7)式所示。

$$d(\mathbf{x}, \mathbf{y}) = \left[\sum_{k=1}^p |x_k - y_k|^2 \right]^{\frac{1}{2}} \quad (7)$$

基于欧式距离没有考虑样本的各指标的数量级水平。当样本的各指标数量级相差悬殊时，该距离不再适合，于是又出现了改进后的马式距离^[8]。

(2) 类与类间的相似性度量：计算两个类之间的距离常用的方法是重心法，设有两个样本类 G_1 和 G_2 ，则这两个类的距离可以表示为：

$$D(G_1, G_2) = d(\bar{g}_1, \bar{g}_2) \quad (8)$$

其中， \bar{g}_1 ， \bar{g}_2 分别为两个类的重心。同时，还有类平均法来衡量类与类间的相似性度量^[9]。

k 均值聚类 (K-Means 算法)

K-Means 算法也称 K-均值聚类算法，是一种广泛使用的聚类算法，也是其他聚类算法的基础。

假定输入样本为 $S = X_1, X_2, \dots, X_m$ ，则算法步骤为：

Step 1: 选择初始的 k 个类别中心 $\mu_1, \mu_2, \dots, \mu_k$ ；

Step 2: 对于每个样本 X_i ，将其标记为距离类别中心最近的类别（距离计算一般采用欧式距离）；

Step 3: 将每个类别中心更新为隶属该类别的所有样本的均值；

Step 4: 重复最后两步，直到类别中心的变化小于某阈值。

终止条件一般有迭代次数，簇中心变化率，最小平方误差 MSE (Minimum Squared Error) 等。

如图 2 所示展现了 k 均值聚类过程。

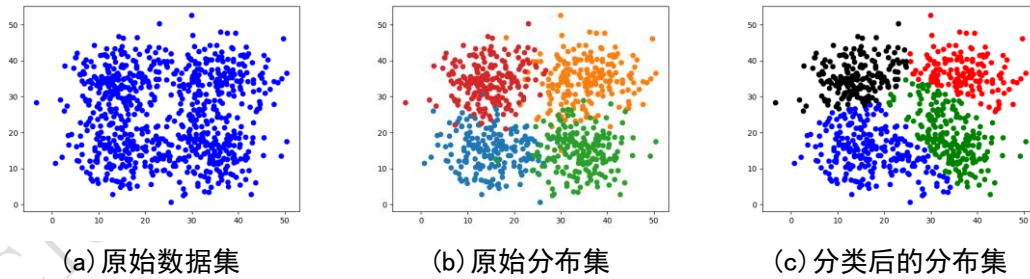


图 2 k 均值聚类过程

4.2.3 模型建立与求解

在该部分，本文通过对附件 1 所给的数据进行过滤，得到 2015 年与 2016 年中没有一个组织或个人声称对攻击事件负责的所有恐怖袭击事件（特征 $claimed=0$ ）数据集。由于可能是同一个恐怖组织或个人在不同时间、不同地点多次作案的恐怖事件的作案规模以及方式等会有所相同，所以本文选取了如死亡总人数 ($nkill$) 等 29 个特征用于表征恐怖袭击事件。特征的选取及空值的处理由表 8 给出。再采用 K-Means 算法基于选取的特征对过滤后的数据进行多次聚类，

最后将可能是同一个恐怖组织或个人在不同时间、不同地点多次作案的若干案件归为一类。

表 8 特征的选取及空值的处理由表

序号	特征	序号	特征	序号	特征
1	是否为持续事件 (extened)	11	目标/受害者类型 (targtype1)	21	人质/绑架受害者总数 (nhostkid) (-1 填充空值)
2	入选标准 (crit1, crit2, crit3)	12	目标/受害者子类型 (targsubtype1) (用 -1 值填充空值)	22	索要赎金 (ransom) (-1 填充空值)
3	疑似恐怖主义 (doubtterr)	13	目标/受害者的国籍 (natlty1) (-1 填充空值)	23	总索取赎金 (ransomamt) (-1 填充空值)
4	其它 (alternative) (-1 填充空值)	14	凶手数量 (nperps) (-99 填充空值)	24	绑架/人质结果 (hostkidoutcome) (-1 填充空值)
5	国家 (country)	15	声称负责 (claimed)	25	释放/逃脱/获救的数量 (nreleased) (-1 填充空值)
6	地区 (region)	16	声称负责的模式 (claimmode) (10 填充空值)	26	国际后勤 (INT_LOG)
7	攻击类型 (attacktype1)	17	死亡总数 (nkill) (0 填充空值)	27	国际的意识形态 (INT_IDEO)
8	自杀式袭击 (suicide)	18	财产损失 (property)	28	国际杂类 (INT_MISC)
9	武器类型 (weaptype1)	19	财产损害程度 (propextent) (-1 填充空值)	29	国际 - 以上任意一类 (INT_ANY)
10	武器子类型 (weapsubtype1) (27 填充空值)	20	人质或绑架的受害者 (ishostkid)		

Step 1: 数据过滤

首先基于年份 2015、2016 过滤得到 2015-2016 年的数据。要对于 2015、2016 年度发生的、尚未有组织或个人宣称负责的恐怖袭击事件进行分类，由于数据中的 `related` 特征表示与此事件相关的其他事件，所以有必要判断过滤后的用于聚类的目标数据中每一个事件的 `related` 事件是否已经有一个组织或个人声称对攻击事件负责；如果某一事件它的 `claimed` 特征为 0，但是其 `related` 中相关的事件的 `claimed` 为 1，说明此事件一定有一个组织或个人声称对攻击事件负责，此时需要将此 `claimed=0` 的事件更新到 `claimed` 为 1 的备用数据中，其他的数据是满

足要求的，将其存储到目标数据中，以保证用于聚类的所有事件全部是没有组织或个人声称对攻击事件负责的。具体过程如图 3 所示。

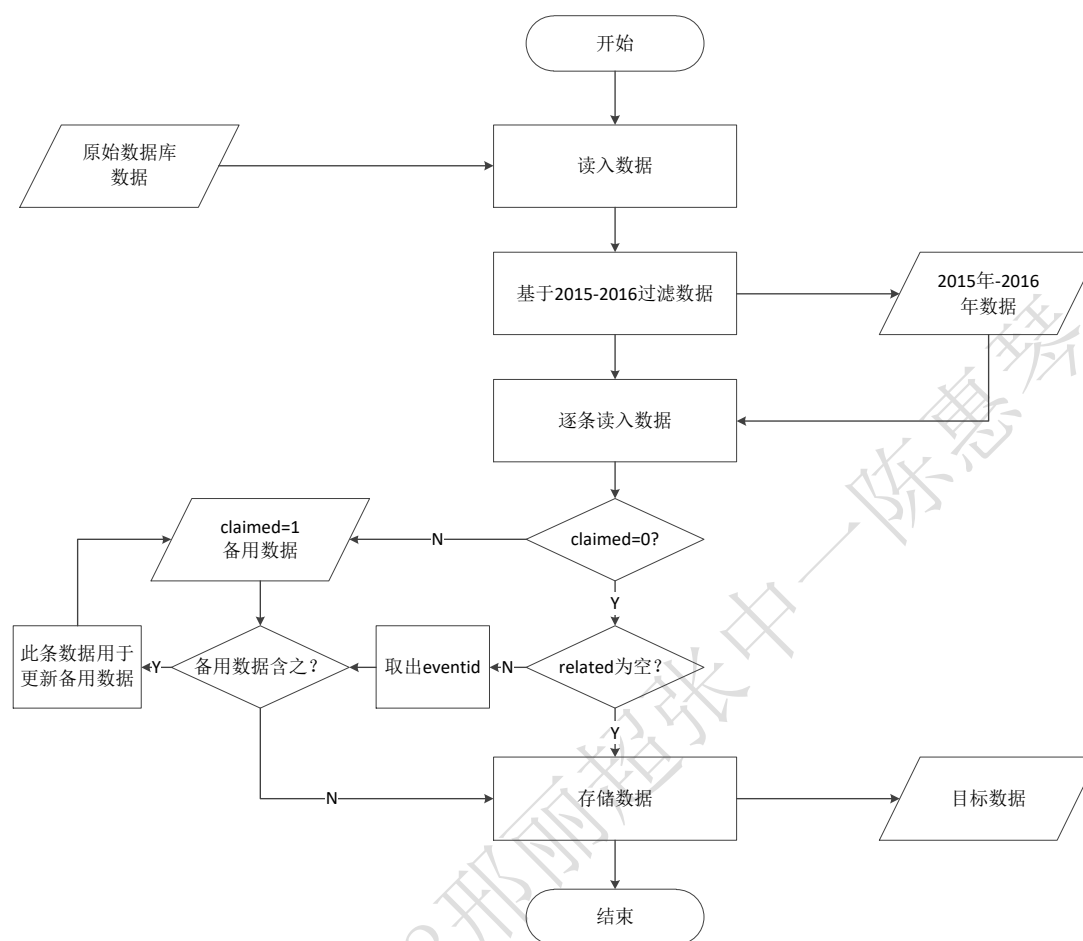


图 3 数据过滤流程图

Step 2: 多次聚类

由于 related 表示两个事件是关联事件，肯定是同一类，所以，首先根据 related 进行硬聚类，此时会得到众含有少数事件的初始种子类，此时再将这些种子类的质心作为聚类的中心，将 related 的事件以此为基础进行聚类并得到中间类；最后，计算此中间类的质心，根据第四题的挖掘值指定最终聚类个数为 10，再次进行聚类，得到最终的聚类结果。演示示例由图 4 给出。

Step 3: 计算此类的危害系数

由于每一类中的事件都是根据样本间的距离定量分类获得，所以，有理由相信每一类中事件具有很大的内在关联，可以认为危害性最大的恐怖袭击是此类的“头目”，所以比较每一类各个事件的危害系数，并以最大的危害系数作为本类的危害系数。最后获得危害最大的 TOP5 并编号为 1 号-5 号，如表 9 给出了前五名的具体情况，包含此类中哪一事件危害系数最高，以及 5 个嫌疑人的具体的危害系数。

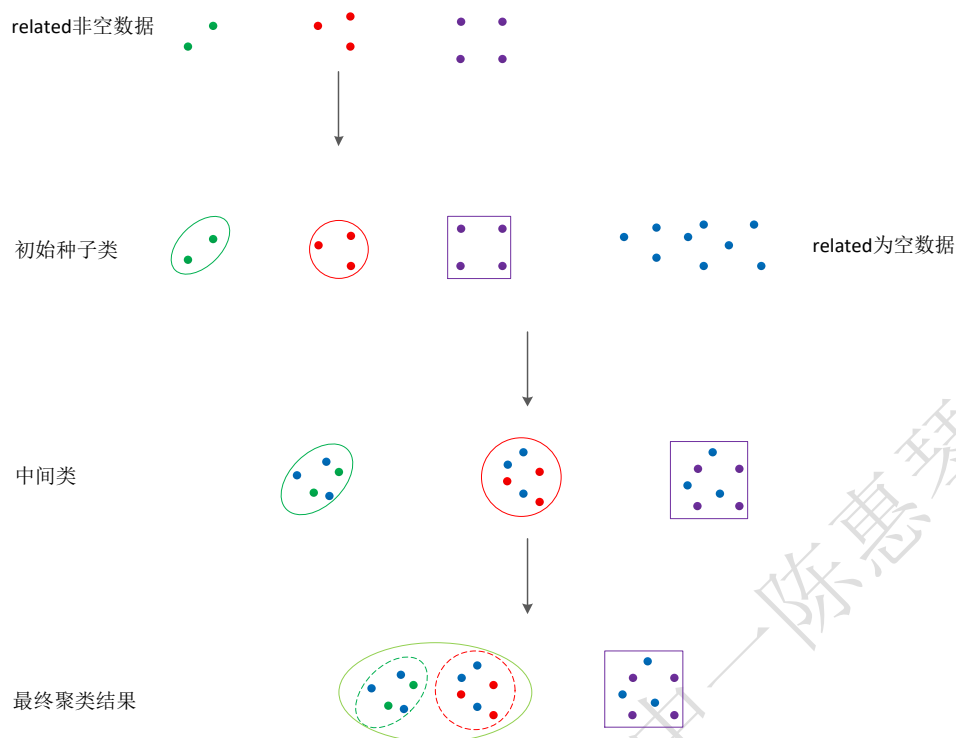


图 4 聚类结果展示图

表 9 前五名的危害系数表

嫌疑人编号	类中危害系数最大的事件 eventid	危害系数
1 号嫌疑人	201504090006	0.804
2 号嫌疑人	201608190039	0.618
3 号嫌疑人	201505210084	0.572
4 号嫌疑人	201501120018	0.397
5 号嫌疑人	201506280116	0.264

Step 4: 确定恐怖分子关于典型事件的嫌疑度

基于特征向量，将典型事件与 Step3 中的 5 个嫌疑人（类）中的各个事件进行求欧式距离，最后通过除此类的事件总数获得典型事件与嫌疑人的“亲密度”，“亲密度”值越小表示嫌疑人的嫌疑越大。

4.2.4 结果分析

在上述的模型建立与求解过程中，本文通过选取具有代表性的特征对 2015 年与 2016 年的无负责恐怖袭击事件进行聚类。本文采用 K 均值多次深度聚类的方法，首先对于具有相关性的事件基于 related 特征进行强聚类，其次，由于同一恐怖组织在实施不同恐怖行动时的武器以及事件规模会有相同性，所以选取 29 个显著特征来表征恐怖事件方案，并基于特征采用 K 均值聚类算法对无责任恐怖事件进行两次不同深度的聚类，最终得到基于特征向量距离的嫌疑人排序。表 10 是基于本文模型的基础上，对恐怖分子关于典型事件的嫌疑度的分析与预测。

表 10 恐怖分子关于典型事件的嫌疑度

	1 号嫌疑人	2 号嫌疑人	3 号嫌疑人	4 号嫌疑人	5 号嫌疑人
201701090031					
201702210037					
201703120023					
201705050009					
201705050010					
201707010028					
201707020006					
201708110018					
201711010006					
201712010003					

4.3 基于时间序列分析的恐怖袭击风险预测模型

4.3.1 问题描述及分析

问题三要求建立适当的数学模型，研究近三年来恐怖袭击事件发生的主要原因、时空特性、蔓延特性、级别分布等规律，进而分析判断下一年全球或某些重点地区的反恐态势，对未来反恐态势的分析评估提供有效的依据，从而提高反恐斗争的针对性和效率。

对具有多种决定因素事件，通过已经发生的事件记录，去预测未来的发展趋势，例如，天气情况的预测，股票的波动情况等，一般都采用时间序列模型进行分析。由附件 1 中的数据可知，该问题的目的是根据 2015-2017 年影响恐怖袭击事件的主要原因、时空特性、蔓延特性、级别分布等各项指标的变化对未来一年中恐怖袭击事件的发生情况进行定量预测。综合考虑各种背景因素，受股票价格预测的启发，本文考虑使用时间序列分析模型解决此问题。

时间序列分析是进行定量预测的方法之一，该方法根据系统有限长度的运行记录(观察数据)，建立能够比较精确地反映序列中所包含的动态数据依存关系的数学模型，并借以对系统的未来进行预报。而且在该题中，恐怖袭击事件发生的各种因素都具有不确定性和无规律性，这在时间序列中属于不规则变动，是一种无规律可循的变动。基于此，本文可以建立基于时间序列的恐怖袭击风险预测模型，根据附件所给数据，对下一年的反恐态势进行预测分析。

4.3.2 模型准备

无论是按时间序列排列的观测数据还是按空间位置顺序排列的观测数据，数据之间都或多或少的存在统计自相关现象。时间序列分析是 20 世纪 20 年代后期开始出现的一种数据处理方法，是系统辨识与系统分析的重要方法之一，是一种动态的数据处理方法。时间序列分析的特点在于：逐次的观测值通常是不独立的，且分析必须考虑到观测资料的时间顺序，当逐次观测值相关时，未来数值可以由过去观测资料来预测，可以利用观测数据之间的自相关性建立相应的数学模型来描述客观现象的动态特征^{[3],[4]}。

时间序列分析^[7]的基本思想是：对于平稳、正态、零均值的时间序列 $\{x_t\}$ ，

$\{x_t\}$ 的取值不仅与其前 n 步的各个取值 $\{x_{t-1}\}, \{x_{t-2}\}, \dots \{x_{t-p}\}$ 有关, 而且还与前 m 步的干扰 $\{a_{t-1}\}, \{a_{t-2}\}, \dots \{a_{t-q}\}$ 有关, 则按多元线性回归的思想, 可得到一般的 ARMA 模型为:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \dots + \phi_p x_{t-p} - \theta_1 a_{t-1} - \theta_2 a_{t-2} \dots - \theta_q a_{t-q} + a_t \quad (9)$$

式中, $\phi_i (i = 1, 2, \dots, p)$ 称为自回归参数, $\theta_j (j = 1, 2, \dots, q)$ 称为滑动平均参数, $\{a_t\}$ 为白噪声序列, 该式称为 x_t 的自回归滑动平均模型, 记为 ARMA(p, q)模型。

特殊地, 当 $\theta_j = 0$ 时, 式(9)变为:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \dots + \phi_p x_{t-p} + a_t \quad (10)$$

该式称为 p 阶自回归模型, 记为 AR(p)模型。

当 $\phi_j = 0$ 时, 式(9)变为:

$$x_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} \dots - \theta_q a_{t-q} \quad (11)$$

该式称为 q 阶滑动平均模型, 记为 MA(q)模型。

具有时变均值的任何时间序列都是非平稳的。差分是得到平稳序列的常用数据变换方法。通过 d 阶差分, 可以将 ARMA(p, q)模型转化为一个自回归滑动平均自动求和模型 ARIMA(p, d, q)模型, 对问题进行分析和预测。

在建立时间序列模型时, 需要进行模型识别, 确定适当的模型, 一般通过以下三个步骤对模型进行识别:

Step 1: 模型的选择: 对于给定的时间序列, 如何选取恰当的 p, d, q 值, 以此确定模型的一般表达式;

Step 2: 参数的估计: 估计一个 ARIMA(p, d, q)模型的参数;

Step 3: 模型的检验: 通过绘制样本模型的自相关函数(ACF)图和偏自相关函数(PACF)图像, 检验模型是否平稳, 从而判断模型的合理性。

总的策略即为: 首先确定合适的但为尝试性的 p, d, q 的值, 然后用估计模型中的参数 ϕ, θ , 最后检验模型是否合适。若模型拟合不充分, 则需要选择其他模型, 并检验其充分性。该方法称为“Box-Jenkins”方法。

表 11 中给出了 ARMA 模型的自相关系数和偏自相关函数的特征:

表 11 ARMA 模型 ACF 和 PACF 的一般特征

	AR(p)	MA(p)	ARMA(p,q)
ACF	拖尾	滞后 q 阶后截尾	拖尾
PACF	滞后 p 阶后截尾	拖尾	拖尾

4.3.3 模型建立与求解

在该部分, 本文通过对附件 1 所给数据进行分析与整理, 提取出 2015-2017 年每个月全球恐怖袭击事件发生的数量, 建立相应的时间序列模型, 对 2018 年

全球恐怖袭击事件的发生情况进行定量预测。具体如下所述：

Step 1: 数据处理

根据附件 1，本文利用 hadoop 的 MapReduce 对数据进行处理，并从中提取出 2015-2017 年每个月全球恐怖袭击时间发生的数量，共得到如表 12 所示的 36 组数据。

表 12 2015-2017 年每个月全球恐怖袭击事件发生的频数

年份	2015 年	2016 年	2017 年
1 月	1534	1162	879
2 月	1295	1153	879
3 月	1183	1145	961
4 月	1277	1120	865
5 月	1316	1353	1081
6 月	1168	1156	1077
7 月	1263	1114	994
8 月	1290	1162	968
9 月	1107	1045	838
10 月	1269	1140	805
11 月	1172	1114	804
12 月	1091	923	749
合计	14965	13587	10900

Step 2: 平稳性检验

序列的平稳性是进行时间序列分析的前提条件，这是因为在大数定理和中心定理中要求样本同分布（这里的同分布等价于时间序列中的平稳性），而大多数建模过程都是建立在大数定理和中心极限定理的前提条件下，如果该条件不满足，则得到的许多结论都是不可靠的。以虚假回归为例，当响应变量和输入变量不平稳时，其标准化系数不再满足 T 分布，这时如果再使用 T 检验进行显著性分析，会导致拒绝原假设的概率增加，即容易犯第一类错误，从而得到错误的结论。

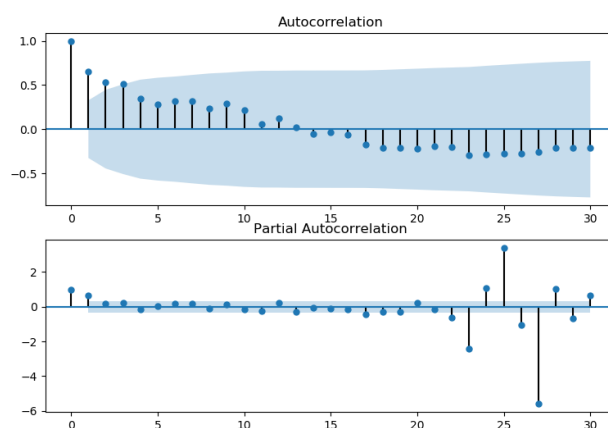


图 5 样本数据的 ACF 与 PACF 拟合图

根据表 11 可知，ARMA 模型的 ACF 和 PACF 应该具有拖尾的特征，而在对样本数据的拟合中，发现样本的 ACF 并没有出现明显的拖尾现象，而样本的 PACF 也出现了大幅波动的现象。由此，可以大概判断该序列不满足 ARMA 模型。下面对样本数据进行单位根检验，判断该序列是否平稳。

表 13 样本序列的单位根检验

Test Statistic	-2.4184816772861106
p-value	0.13658789037509417
#Lags Used	0
Number of Observations Used	35
Critical Value (5%)	-2.9485102040816327
Critical Value (1%)	-3.6327426647230316
Critical Value (10%)	-2.6130173469387756

通过对样本上数据进行单位根检验，得到 p 值为 0.137，所以在置信水平为 95% 的区间下并不显著，不能拒绝原假设，即该样本序列是不平稳的。

Step 3: 差分&模型识别

差分是时间序列分析中最常用来剔除周期性因素的方法，它主要是对等周期间隔的数据进行线性求减。差分也是将非平稳序列转化为平稳序列的常用方法之一。在本题中，样本序列经过平稳性检验是非平稳的，所以下面通过对样本序列进行差分，进一步判断差分后序列的平稳性。

首先对一阶差分后的序列进行单位根检验，如表 14 所示。根据表中数据可知， P 值为 0.01，观察其统计量发现该序列在置信水平为 99% 的区间下是显著地，所以有理由拒绝原假设，认为该序列是平稳的。

表 14 一阶差分后的单位根检验

Test Statistic	-3.4221845974000797
p-value	0.010225451656768048
#Lags Used	10
Number of Observations Used	24
Critical Value (5%)	-2.9922162731481485
Critical Value (1%)	3.7377092158564813
Critical Value (10%)	-2.635746736111111

由于一阶差分后的序列是平稳序列，满足时间序列分析的前提条件，所以对差分后的数据，通过其 ACF 和 PACF 图像，便可进行模型识别，确定模型的参数。如图 7 所示为一阶差分数据的 ACF 和 PACF 图像。

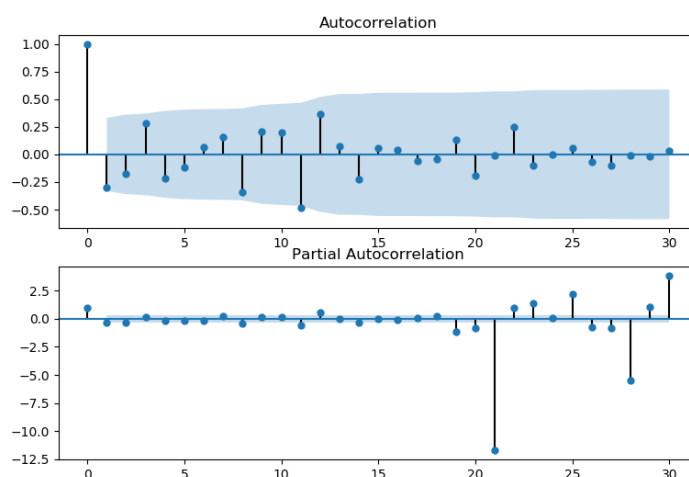


图 6 一阶差分后的 ACF 与 PACF 图

以 2015-2016 年作为训练数据，通过一阶差分，发现样本的 ACF 与 PACF 具有明显的拖尾现象，根据图像分析可得模型的参数为 $p = 8, q = 8$, 差分阶数 $d = 1$, 步长为 1，由此确定样本的时间序列模型为 $ARIMA(8,1,8)$ 。

Step 4: 模型的可行性分析

基于 $ARIMA(8,1,8)$ 模型，本文通过 2015-2016 年恐怖袭击事件的样本值对 2017 年恐怖袭击事件进行预测，并将预测值与样本值进行对比分析，如果样本值与预测值的误差足够小，则认为该预测模型是合理的。预测与对比结果如图 8 所示。

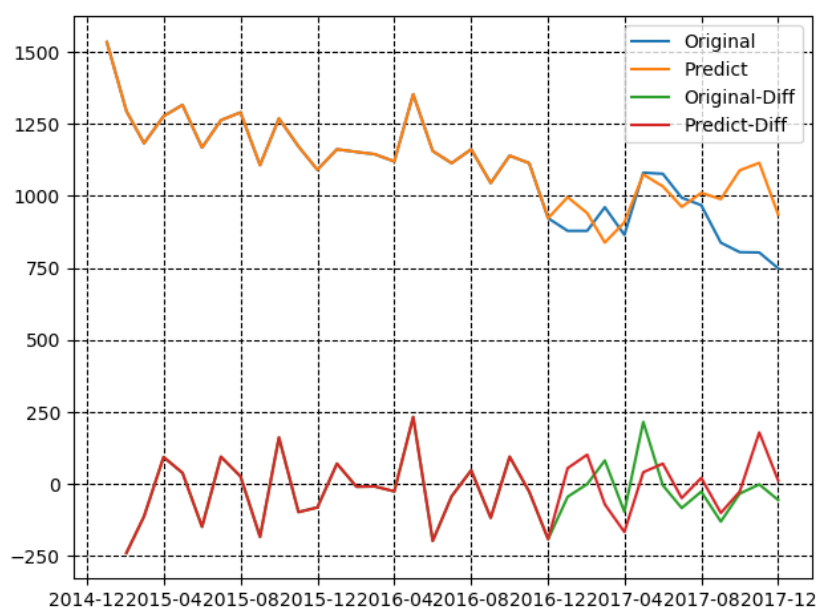


图 7 2017 年的样本值与预测值的对比分析

根据 $ARIMA(8,1,8)$ 拟合得到的 2017 年恐怖袭击事件的预测值与样本值在相

对误差为 0.0906 的情况下基本吻合，具体的预测对比结果如表 15 所示。由此证明了本文模型的可行性。

表 15 2017 年恐怖袭击时间样本值与预测值对比分析

时间	2017-01	2017-02	2017-03	2017-04	2017-05	2017-06
样本值	879	879	961	865	1081	1077
预测值	995	940	838	908	1074	1033
时间	2017-07	2017-08	2017-09	2017-10	2017-11	2017-12
样本值	994	968	838	805	804	749
预测值	962	1010	989	1089	1114	935
相对误差	0.0906					

Step 5: 模型预测分析

由 Step 4 可知，ARIMA(8,1,8)可以在一定的误差范围内对未来事件的发生情况进行预测，由此，接下来，本文将通过 2015-2017 年恐怖袭击事件发生的情况对 2018 年恐怖袭击事件的发生情况进行定量预测。预测结果如图 9 所示。

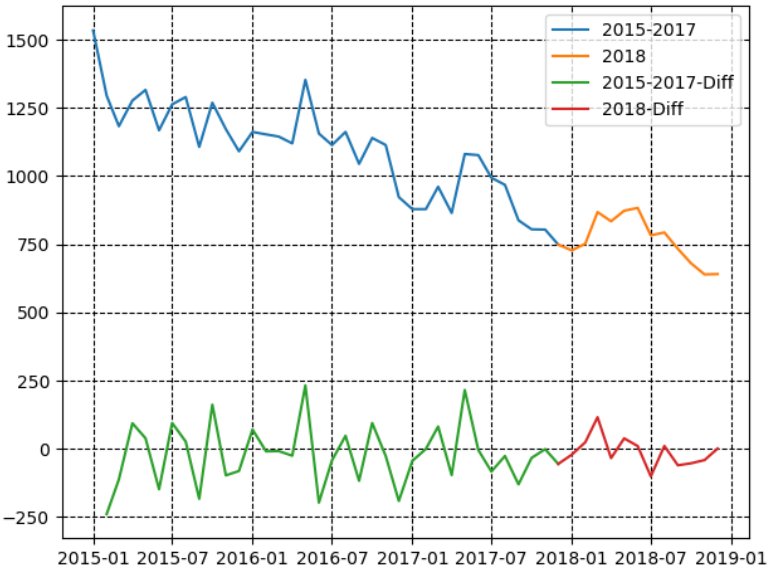


图 8 2018 年恐怖袭击事件的定量预测

根据图 9 的预测曲线变化可得，2018 年每个月全球恐怖袭击事件的发生的数量如表 16 所示。

表 16 2018 年全球恐怖袭击事件数量预测值

2018 年全球恐怖袭击事件每个月发生的数量												
月份	01	02	03	04	05	06	07	08	09	10	11	12
事件数量	727	752	868	834	873	883	783	793	733	680	639	640
总计	9204											

4.4.4 结果分析

在上述的模型建立与求解过程中, 本文对 2018 年全球恐怖袭击事件的发生情况进行了定量的预测, 在该部分, 本文将结合上述的预测结果, 从不同国家恐怖袭击事件的发生情况以及武器使用情况方面进行进一步的分析, 并针对预测结果提出相应的建议。

结合 2015-2016 年全球恐怖袭击时间的数量, 对全球恐怖袭击事件发生数量排行前 20 的国家统计数据如表 17 所示。

表 17 2015-2016 年全球恐怖袭击事件发生数量的国家排行榜

序号	国家	恐袭事件数	序号	国家	恐袭事件数
1	Iraq	8576	11	Libya	1155
2	Afghanistan	4960	12	Turkey	1145
3	India	2875	13	Thailand	786
4	Pakistan	2827	14	Ukraine	759
5	Philippines	2046	15	Bangladesh	598
6	Nigeria	1655	16	West Bank and Gaza Strip	486
7	Somalia	1634	17	Democratic Republic of the Congo	455
8	Yemen	1415	18	Sudan	437
9	Egypt	1247	19	Colombia	362
10	Syria	1207	20	Mali	361

根据该统计结果, 对恐怖袭击事件数量排名 TOP5 的国家进行具有针对性的分析与预测, 为降低此地区的恐袭事件数提供理论依据与帮助。

表 18 恐怖袭击事件 gname 和 weaptype1 的统计结果

序号	国家	gname/数量(所占百分比)		weaptype1/数量(所占百分比)	
1	Iraq	Islamic State of Iraq and the Levant (ISIL)	3356(39.1%)	Explosives	6905(80.5%)
		Asa'ib Ahl al-Haqq	55(0.641%)	Firearms	695(8.10%)
		Al-Naqshabandiya Army	11(0.128%)	Incendiary	82(0.956%)
		Kurdistan Workers' Party (PKK)	11(0.128%)	Melee	36(0.420%)
		Kata'ib Hezbollah	4(0.047%)	Chemical	35(0.408%)
2	Afghanistan	Taliban	3204(64.6%)	Explosives	2161(43.6%)
		Khorasan Chapter of the Islamic State	242(4.88%)	Firearms	1643(33.1%)
		Haqqani Network	15(0.302%)	Incendiary	124(2.50%)
		Hizb-I-Islami	4(0.081%)	Melee	46(0.927%)
		Tehrik-i-Taliban Pakistan (TTP)	4(0.081%)	Chemical	23(0.463%)

3	India	Maoists	746(25.9%)	Explosives	1261(43.9%)
		Communist Party of India - Maoist (CPI-Maoist)	290(10.1%)	Firearms	872(30.3%)
		Hizbul Mujahideen (HM)	69(2.40%)	Incendiary	329(11.4%)
		Gorkha Janmukti Morcha (GJM)	67(2.33%)	Melee	165(5.74%)
		Lashkar-e-Taiba (LeT)	62(2.16%)	Sabotage Equipment	9(0.313%)
4	Pakistan	Tehrik-i-Taliban Pakistan (TTP)	298(10.5%)	Explosives	1577(55.8%)
		Baloch Republican Army (BRA)	116(4.10%)	Firearms	1130(40.0%)
		Baloch Liberation Front (BLF)	109(3.86%)	Incendiary	25(0.884%)
		Khorasan Chapter of the Islamic State	94(3.33%)	Melee	4(0.141%)
		Baloch Liberation Army (BLA)	62(2.19%)	Chemical	1(0.0353%)
5	Philippines	New People's Army (NPA)	852(41.6%)	Firearms	1150(56.2%)
		Bangsamoro Islamic Freedom Movement (BIFM)	214(10.5%)	Explosives	678(33.1%)
		Abu Sayyaf Group (ASG)	203(9.92%)	Incendiary	96(4.69%)
		Maute Group	31(1.52%)	Melee	12(0.587%)
		Islamic State of Iraq and the Levant (ISIL)	26(1.27%)	-	-

表 18 是对恐怖袭击事件发生排行 TOP5 的事件进一步从犯罪集团和所使用的武器类型方面进行统计分析。根据统计结果可以看出，在每个恐怖袭击频繁发生的国家，都有一个强有力的恐怖组织，引发了该国家的大部分恐袭事件。如 Iraq 的 Islamic State of Iraq and the Levant (ISIL)，Afghanistan 的 Taliban，India 的 Maoists，Pakistan 的 Tehrik-i-Taliban Pakistan (TTP)，Philippines 的 New People's Army (NPA)，如果这些国家的相关政府部门可以对这些恐怖组织进行实时监控与控制，将在一定程度上可以降低该国家的恐怖袭击事件数量，而且如果以“杀鸡儆猴，以一儆百”的策略对其他恐怖组织进行警告，那么反恐效果将更加明显。

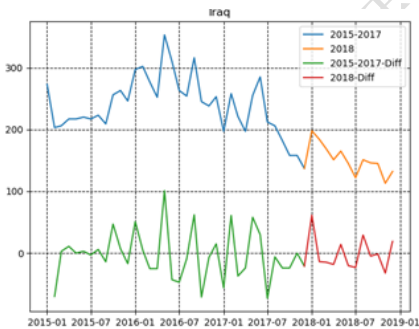
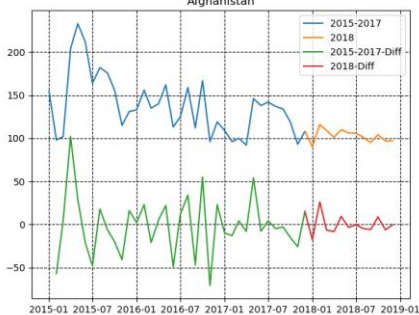
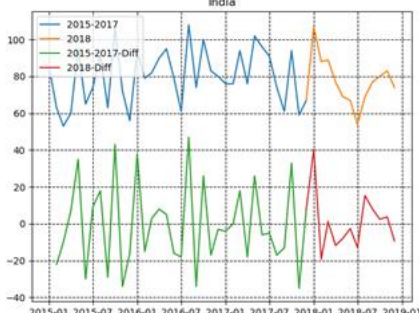
在使用的武器类型这部分，观察统计数据可以发现，Explosives, Firearms, Incendiary, Melee 在这些国家的恐袭事件中全部出现，Chemical 出现在 Iraq, Afghanistan, Pakistan 中，Sabotage Equipment 仅出现在 Pakistan 中，所以我们有理由相信，控制这些武器的传播会很大程度上减少恐袭事件的发生。为了对这五个国家 2018 年的恐怖袭击事件的发生提出更具有针对性的意见，本文分别对这五个国家进行预测，如表 19 所示。

从预测结果可以看出，Iraq 预测会发生由炸药(Explosives)引起的恐袭事件 1466 起，Afghanistan 预计会发生由炸药(Explosives)引起的恐袭事件 537 起，India 预测会发生由炸药(Explosives)引起的恐袭事件 410 起，Pakistan 预测会发生由炸

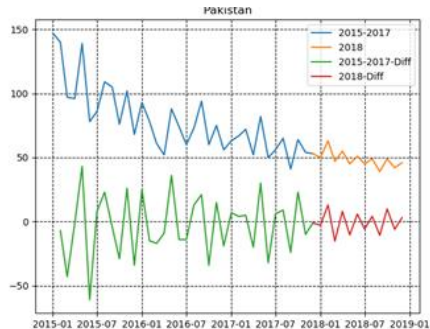
药(Explosives)引起的恐袭事件 324 起, Philippines 预测会发生由枪支(Firearms)引起的恐袭事件 364 起。

根据这个预测结果, 从武器使用类型的角度, 可以提出以下建议, 用于在一定程度上减少恐怖袭击事件的数量。

表 19 2015-2017 恐袭事件数 TOP5 的国家 2018 年发生恐袭事件数预测结果

2018 年 Iraq 发生恐怖袭击事件数的预测结果						
时间	2018 01	2018 02	2018 03	2018 04	2018 05	2018 06
数量	198	184	169	151	165	145
时间	2018 07	2018 08	2018 09	2018 10	2018 11	2018 12
数量	122	151	146	145	113	132
总计	1821					
						
2018 年 Afghanistan 发生恐怖袭击事件数的预测结果						
时间	2018 01	2018 02	2018 03	2018 04	2018 05	2018 06
数量	90	116	109	101	110	106
时间	2018 07	2018 08	2018 09	2018 10	2018 11	2018 12
数量	106	101	95	104	97	97
总计	1232					
						
2018 年 India 发生恐怖袭击事件数的预测结果						
时间	2018 01	2018 02	2018 03	2018 04	2018 05	2018 06
数量	107	88	89	77	69	67
时间	2018 07	2018 08	2018 09	2018 10	2018 11	2018 12
数量	54	69	77	80	83	74
总计	934					
						

2018 年 Pakistan 发生恐怖袭击事件数的预测结果						
时间	2018 01	2018 02	2018 03	2018 04	2018 05	2018 06
数量	50	63	47	55	45	51
时间	2018 07	2018 08	2018 09	2018 10	2018 11	2018 12
数量	45	49	39	49	42	46
总计	581					

						
--	--	--	--	--	--	--

2018 年 Philippines 发生恐怖袭击事件数的预测结果						
时间	2018 01	2018 02	2018 03	2018 04	2018 05	2018 06
数量	59	56	52	56	61	56
时间	2018 07	2018 08	2018 09	2018 10	2018 11	2018 12
数量	51	47	50	55	55	50
总计	648					

Tip 1: 相关政府部门应该严格控制自己国家火药的生产量, 严禁各种小作坊私自生产火药;

Tip 2: 对于合法的火药生产部门生产的火药, 每一批火药的去向与用途都要进行严格的登记造册, 并与使用地政府联合实行监察机制;

Tip 3: 严格清查各个海关, 严格禁止各种军火武器的非法入境;

Tip 4: 严禁私人持有枪支弹药, 并制定相应的明文法律条例予以约束, 打击各种黑市非法私售枪支弹药, 从源头控制非法武器的流动;

Tip 5: 国家应该重点推进本国经济的发展, 制定适合本国国情的治国方针和经济发展策略, 关注民生, 缩小贫富差距, 为人民提供和谐稳定的生活环境, 从精神和物质方面逐渐安抚民生, 从而实现国家的安定和平。

以上为本文根据 1998-2017 年全球恐怖袭击事件的发生情况进行分析, 对于恐怖事件频繁发生的国家给予的一些建议, 希望能够对降低恐怖袭击事件的发生有效。

4.4 基于无标度网络的恐怖袭击事件相关性研究

4.4.1 问题描述及分析

问题四要求本文运用数学建模的思想, 对附件 1 中的数据进行进一步分析。随着经济全球化以及互联网技术的普及, 恐怖组织的跨国活动日益成为恐怖主义

的一种基本活动模式，策划、组织、指挥、联络、实施、协助、支持等行动分别在不同的国家完成或同时进行，不同国家的不同恐怖组织之间的相互勾结、相互联系日益增多。各国恐怖组织的跨国合作日益紧密，跨国活动日趋频繁。各恐怖组织已经建立起密切的网络，这张网络不是静态的，而是动态的，并逐步向其他地区扩散，呈现构建全球恐怖网络之势。

基于此，本文将根据每个恐怖事件的相关性，构造恐怖事件的相关性网络，从宏观上考虑网络的特性和演化模式，不考虑恐怖事件及恐怖组织的内部组成，利用复杂网络理论对其网络结构进行分析研究，得出其度分布符合无标度特性的结论，进一步对不同时间、不同地点的恐怖事件发生的相关性进行分析与讨论。

4.4.2 模型准备

网络作为一门科学始于 18 世纪欧拉开创图论，第二个重要发展阶段始于 1960 年，Erdos 和 Renyi 基于随机机制提出著名的随机图模型。在该模型中，节点以等概率成对连接形成网络，随机图模型揭示了现实世界中具有随机特性的各种系统的随机原理。最近的网络研究发现，大量的真实网络既不是规则网络，也不是随机网络，而是具有独特特征的复杂网络，这种网络在 20 世纪末成为新的研究热点。复杂网络目前研究重点集中在网络特征的描述，而小世界效应(Small-World Effect)和无标度^[6]特性(Scale-Free Property)是目前最受关注的两类复杂网络特征。随机网络图与无标度网络图的一般模型如图 10 所示。

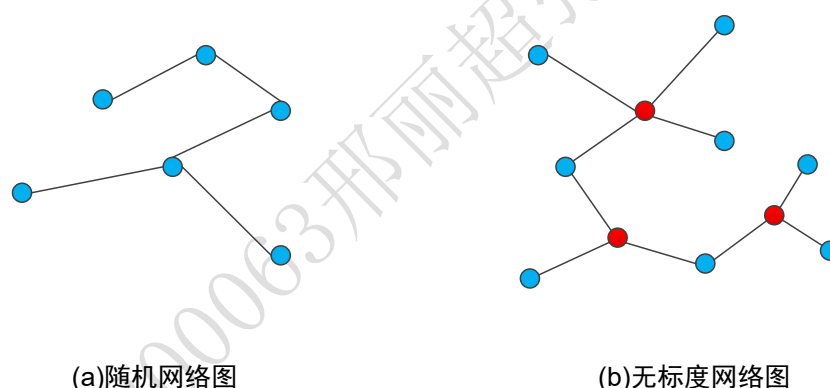
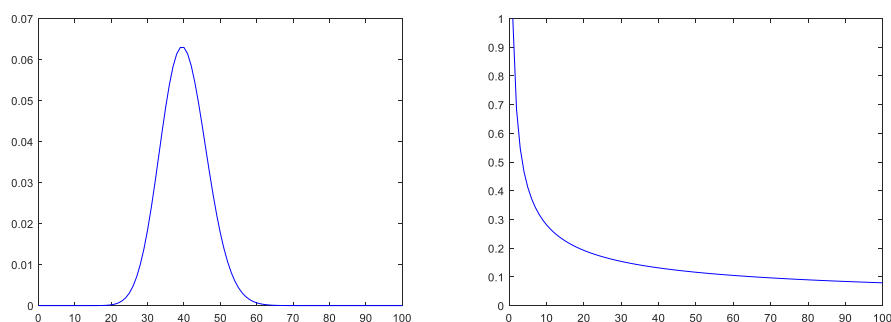


图 9 随机网络图与无标度网络图的基本模型

为了解释幂律分布的产生机理，Barabasi 和 Albert 基于增长与优先连接(preferential attachment)机制提出了著名的无标度模型，这两条机理缺少任何一条都不能生成无标度网络。增长特性是指网络的规模是不断扩大的，优先连接特性是指新加入的节点更倾向于与那些具有较高连接度的“大”节点相连接，这种现象也可称为“富者愈富(richer get richer)”或“马太效应(Matthew effect)”。由该模型演化生成的网络，其度分布 $p(k)$ 服从递减幂律分布，即 $p(k) \propto k^{-\gamma}$ ，其中标度指数 $\gamma > 0$ ， k 表示结点的度。无标度网络的特性在于其度分布没有一个特定的平均指标，即大多数节点的度在某一区间范围内。在一般的随机网络中，度下降的速度是指数型的，而在无标度网络中，度只以多项式的速度下降，如图 11 所示。由此可知，无标度图中度分布在高端的衰减明显更慢。



(a) 随机网络的度分布

(b) 无标度网络的度分布

图 10 随机网络和无标度网络的度分布

无标度网络的提出，极大地激发了科学界的研究热情，人们对来自不同领域的大量实际网络拓扑特征进行了广泛的实证性研究，发现许多现实网络，包括信息网络、社会网络和生物网络甚至大规模软件系统如 JDK 都具有无标度特性。他们探索无标度网络背后的形成机理和组织原则，建立恰当的模型去拟合复杂网络自组织过程。通过简单的、理想化模型的分析，去拟合或预测现实网络的小世界效应和无标度特性。

4.4.3 模型建立与求解

本文所研究的数据来源于全球恐怖主义数据库 GTD(The Global Terrorism Database)，它包含了 1998-2017 年世界上发生的恐怖袭击事件的详细记录，字段

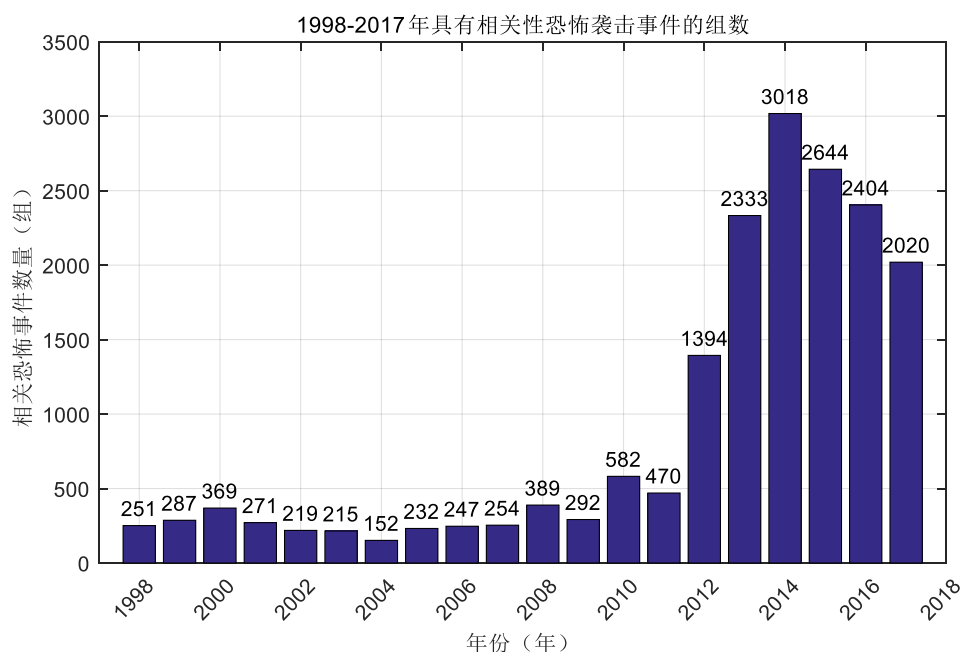


图 11 1998-2017 年具有相关性恐怖袭击事件的组数

包含受袭国家、所属地域、袭击时间、恐怖组织、受害者类型、袭击方式、死亡

人数、使用武器、事件概要等等。编写程序清除 GTD 数据库中的重复事件和信息不完整的数据，提取每年相关恐怖袭击事件的组数，如图 12 所示。

本文通过对 1998-2017 年这 20 年恐怖袭击事件进行上述统计分析，共获得 18043 组具有相关性的恐怖袭击事件。由图 12 可知，从 1998 年-2011 年，相关恐怖袭击事件的组数波动相对稳定。然而，随着信息化时代的到来，自 2012 起，恐怖袭击事件之间相互关联的组数陡然上升。基于以上的数据分析，本文对每一年相关性恐怖袭击事件的情况进行整合，表 20 通过从 2017 年的处理数据集中随机抽取的几组数据，对本文的数据处理模式进行简单说明。根据表 1 中显示的信息可知，对于编号为 201701030046 的事件而言，事件 201701030047 和事件 201701030048 的发生与事件 201701030046 的发生具有一定的相关性。

基于以上数据，假设每一个事件对应一个节点，如果两个事件的发生具有相关性，则可以认为引发这两个事件的恐怖组织或凶手之间存在资金、人员、军火武器等方面的沟通交流，这两个节点之间就有一条边相连。按照这个原则，编写程序对所有节点进行分析，并根据 DL 协议对 GTD 数据库中 18043 个事件的编号生成相应的网络描述文件。下面以 1998 年和 2007 年的数据为例，进行分析。

表 20 2017 年相关恐怖袭击事件数据处理表

恐怖袭击事件编号	相关恐怖袭击事件集
201701030046	201701030047、201701030048
201701080030	201701080031、201701080032
201701100032	201701100033、201701100034、201701100035、201701100036、 201701100037、201701100038、201701100039
201701140053	201701140054、201701140055、201701140056、201701140057、 201701140058、201701140059、201701140061、201701170034、 201701170035、201701170036、201701170037
201701150011	201701150012、201701150013、201701150014、201701150015
201701170007	201701170008、201701170009、201701170010、201701170011、 201701170012、201701170013、201701170014
201701180014	201701180015
201701180024	201701180025、201701180026、201701180027、201701180028、 201701180029、201701180030
201702040005	201702040006、201702040007、201702040008、201702040009、 201702040010、201702040011、201702040012、201702040013、 201702040014、201702040015、201702040016、201702040017、 201702040018、201702040019、201702040020、201702040021、 201702040022、201702040023

对 1998 年的 251 组数据根据 DL 协议生成如下的网络描述文件：

IMPORT FROM EXCEL

Input Excel file X:\out1998.csv
Output UCINET dataset: X:\out1998

out1998

199801040002

199801110004	199801110528	
199801110005	199801110528	
199801120001	199801126912	199801126912
199801120002	199801126912	199801126912
199801120003	199801126912	199801126912
199801190001	199801192448	
199801190003	199801192448	
199801210002	199801208832	199801208832
199801210003	199801208832	199801208832
199801210004	199801208832	199801208832

利用 NetDraw 工具读取上述描述文件进行画图, 如图 13 所示为 1998 年全球恐怖事件发生的网络拓扑结构图, 图 14 为 2007 年全球恐怖事件发生的网络拓扑结构图 (由于 2017 年发生的恐怖事件包含的数据量过大, 所以该年的网络拓扑结构图将在附录中给出)。

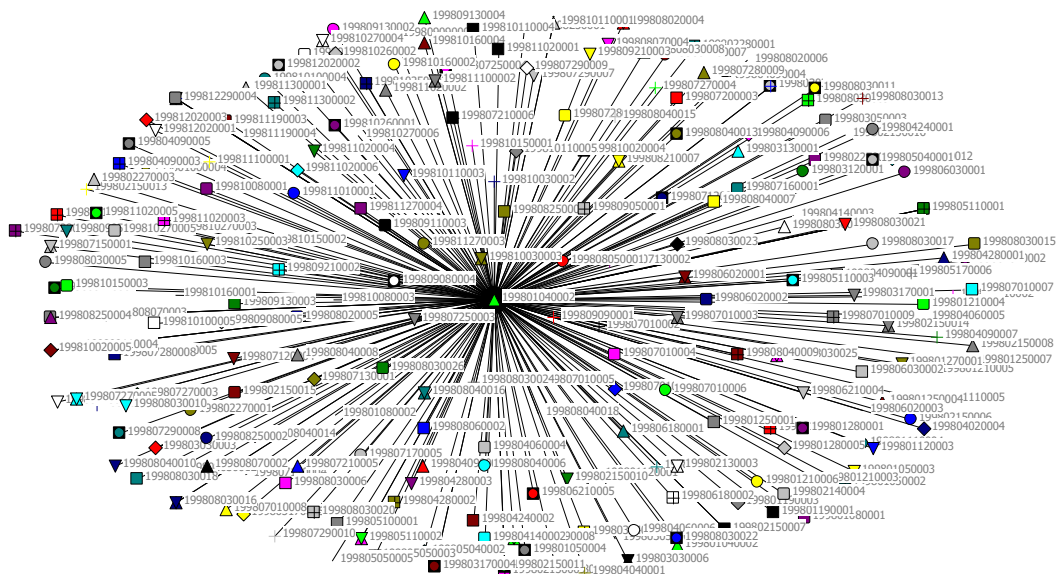


图 12 1998 年全球恐怖事件发生的网络拓扑结构图

在图中, 蓝色的节点作为转折节点存在, 属于辅助构图节点, 而非事件节点。对于一个网络拓扑结构图而言, 转折节点存在的越多, 事件之间的相关性概率就越大。从这些图中可以直观的看出, 相互关联的恐怖事件之间的网络规模是持续扩大的, 每一年都会有新的节点出现, 而且每个新节点的出现都倾向于连接那些已经具有较大“度”的节点, 致使这些节点的度更大, 由此导致“富者愈富”的现象出现。而且转折节点的数量也在持续增加, 说明事件之间的关联性持续增大。

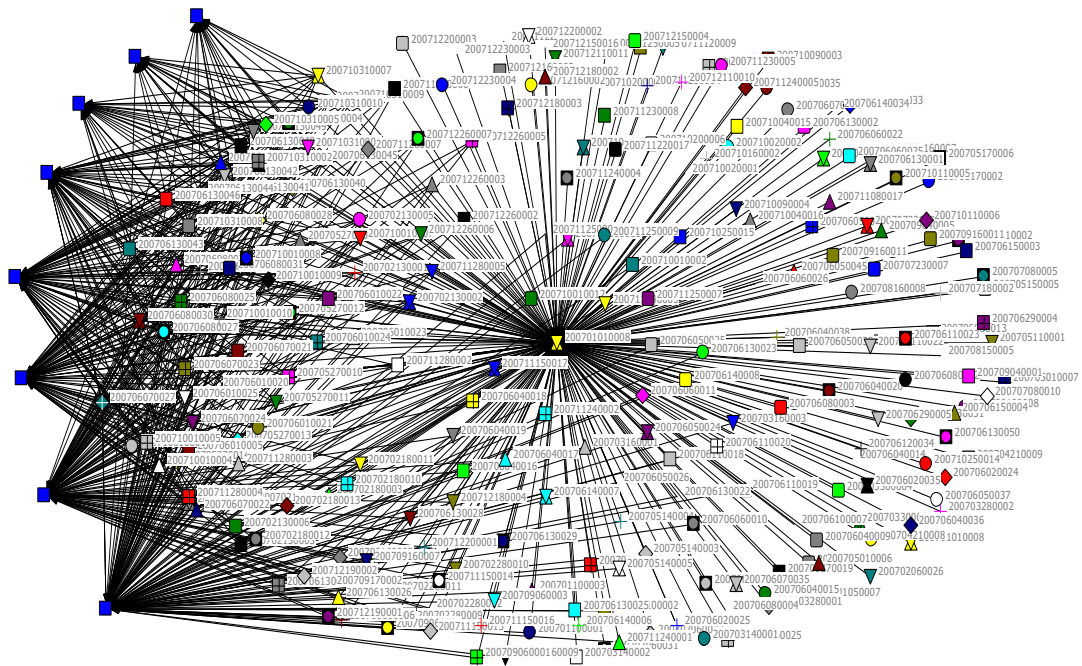


图 13 2007 年全球恐怖事件发生的网络拓扑结构图

4.4.5 结果分析

通过利用无标度网络模型对恐怖袭击事件之间的相关性分析，从而亦可获取一些潜在的信息。在网络拓扑结构图中，蓝色的点可以算为一个簇，可以以蓝色的点为中心对事件进行分组，如表 21 种所示的事件分组数。平均每组事件可以反应为这一组的社会危害性，组中包含的事件数越多，说明其社会危害性越大。

表 21 全球恐怖袭击事件的相关性分组与危害性

年份	具有关联最大的事件 eventid	事件分组数	平均每组事件数	年份	具有关联最大的事件 eventid	事件分组数	平均每组事件数
1998	199801040002	25	39	2008	200801010005 200801010006	15	320
1999	199901050006	10	140	2009	200901010013 200901010014	7	674
2000	200001010002	38	38	2010	201001010006 201001010007	23	210
2001	200101070009	8	209	2011	201102060019	15	338
2002	200201010005	17	78	2012	201201010004	42	202

2003	200301010003	10	128	2013	201301010003	35	343
	200301010004						
	200301010005						
	200301010006						
	200301010007						
	200301010008						
	200301010009						
	200301010010						
	200301010011						
	200301010012						
2004	200402010009	7	167	2014	201401010055 201401010056 201401010057	47	369
2005	200501020005	7	288	2015	201501020062	23	651
2006	200601020005 200601020006	5	552	2016	201601010009	79	171
2007	200701010008	9	360	2017	201701020004	41	266

5.模型评价

5.1 模型的优点

本文所建立的模型具有以下几个优点：

(1) 针对问题一建立的基于层次分析法的恐怖袭击事件危害性等级划分模型，该模型把研究对象作为一个系统，按照分解、比较判断、综合的思维方式进行决策，成为继理性分析之后发展起来的重要工具。在该模型中，在每个层次中的每个因素对结果的影响程度都可以量化，非常清晰明确。所以该模型具有系统性、简洁实用、所需定向数据信息较少的特点，对于处理许多用传统的最优化技术无法着手的实际问题具有很大优势。

(2) 在问题二中，本文采用 K 均值多次深度聚类的方法，首先对于具有相关性的事件基于 related 特征进行强聚类，其次，由于同一恐怖组织在实施不同恐怖行动时的武器以及事件规模会有相同性，所以选取 29 个显著特征来表征恐怖事件方案，并基于特征采用 K 均值聚类算法对无责任恐怖事件进行两次不同深度的聚类，最终得到基于特征向量距离的嫌疑人排序。本文将数据挖掘与深度聚类的思想应用在嫌疑犯追踪的问题中，也是本文的创新点之一。

(2) 本文使用时间序列分析法对未来一年恐怖袭击事件的数量进行预测，利用事物发展的统计特性，推测出事物的发展趋势。通过建立恰当的时间序列分析模型，本文在很小的误差范围内实现了对未来数据的预测，用时间序列分析该问题，也是本文的新颖之处。

(3) 本文通过使用无标度网络拓扑结构图，反应了恐怖袭击事件之间的相互关联性，为有效控制恐怖袭击事件发生提供了理论依据。

5.2 模型的缺点

本文模型的缺点在于，在问题一的模型中使用了层次分析法，定量数据较少，

定性成分多,不易令人信服。而且在处理数据的过程中,指标过多时,数据量较大,且权重难以确定,这是本文所建模型的缺点以及有待改进的部分。

6.参考文献

- [1] 王雷,王欣,赵秋红.基于和声搜索算法优化支持向量机的突发暴恐事件分级研究[J].管理评论,2016,28(08):125-132.
- [2] 龚伟志,刘增良,王烨,徐建宏.基于大数据分析恐怖袭击风险预测研究与仿真[J].计算机仿真,2015,32(04):30-33+398.
- [3] <https://www.cnblogs.com/xuanlvshu/p/5410721.html>;
- [4] <http://www.cnblogs.com/foley/p/5582358.html>;
- [5] <https://blog.csdn.net/shine19930820/article/details/72667656>;
- [6] 许晴. 1998~2004 年间世界恐怖活动的无标度特性分析[A]. 中国仪器仪表学会 (CIS)、中国系统仿真学会 (CSSS)、中国仪器仪表学会微型计算机应用学会 (CACIS)、中国系统仿真学会复杂系统建模与仿真计算专业委员会筹备处 (CSSC). 全国第 20 届计算机技术与应用学术会议 (CACIS·2009) 暨全国第 1 届安全关键技术与应用学术会议论文集 (上册) [C]. 中国仪器仪表学会 (CIS)、中国系统仿真学会 (CSSS)、中国仪器仪表学会微型计算机应用学会 (CACIS)、中国系统仿真学会复杂系统建模与仿真计算专业委员会筹备处 (CSSC) :,2009:6.
- [7] 谭贇.时间序列分析模型构建与 MATLAB 实现[J].科技资讯,2009(26):253-254.
- [8] <https://blog.csdn.net/panglinzhuo/article/details/77801869>.
- [9] https://blog.csdn.net/qq_39422642/article/details/78821812.

7.附录

#用于统计的 hadoop 的 MapReduce 代码 (JAVA)

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.log4j.BasicConfigurator;

public class jianmo {

    public static class CarLocusMapper extends Mapper<Object, Text, Text,
IntWritable> {

        private Text StrKey = new Text();
```

```

        private final static IntWritable one = new IntWritable(1);
        public void map(Object key, Text value, Context context) throws IOException,
        InterruptedException {
            String[] SplitValue = value.toString().split("\n");
            for (String str: SplitValue) {
                String[] SplitStr = str.split("[+");
                StrKey.set(SplitStr[1] + "-" + SplitStr[2]);
                context.write(StrKey, one);
            }
        }
    }
}

```

```

public static class CarLocusReducer extends Reducer<Text, IntWritable, Text,
IntWritable> {

```

```

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

```

```

public static void main(String[] args) throws Exception {
    BasicConfigurator.configure();
    Configuration conf = new Configuration();
    if (args.length != 2) {
        System.err.println("Usage: <in> <out>");
        System.exit(2);
    }
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(jianmo.class);
    job.setMapperClass(CarLocusMapper.class);
    job.setCombinerClass(CarLocusReducer.class);
    job.setReducerClass(CarLocusReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
}

```

```

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

#统计 related 的事件数据 (JAVA)

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.InputStreamReader;

public class related {
    public static BufferedWriter makefile(String str) throws IOException {
        BufferedWriter bw = new BufferedWriter(
            new OutputStreamWriter(
                new FileOutputStream(str)));
        return bw;
    }
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(
            new InputStreamReader(
                new FileInputStream("data.csv")));

        String str = "";
        BufferedWriter bw1998 = makefile("out1998.csv");
        BufferedWriter bw1999 = makefile("out1999.csv");
        BufferedWriter bw2000 = makefile("out2000.csv");
        BufferedWriter bw2001 = makefile("out2001.csv");
        BufferedWriter bw2002 = makefile("out2002.csv");
        BufferedWriter bw2003 = makefile("out2003.csv");
        BufferedWriter bw2004 = makefile("out2004.csv");
        BufferedWriter bw2005 = makefile("out2005.csv");
        BufferedWriter bw2006 = makefile("out2006.csv");
        BufferedWriter bw2007 = makefile("out2007.csv");
        BufferedWriter bw2008 = makefile("out2008.csv");
        BufferedWriter bw2009 = makefile("out2009.csv");
        BufferedWriter bw2010 = makefile("out2010.csv");
        BufferedWriter bw2011 = makefile("out2011.csv");
        BufferedWriter bw2012 = makefile("out2012.csv");
        BufferedWriter bw2013 = makefile("out2013.csv");
        BufferedWriter bw2014 = makefile("out2014.csv");
    }
}

```

```

BufferedWriter bw2015 = makefile("out2015.csv");
BufferedWriter bw2016 = makefile("out2016.csv");
BufferedWriter bw2017 = makefile("out2017.csv");
while((str = br.readLine()) != null) {
    String[] SplitStr = str.split("[+");
    if(SplitStr[SplitStr.length - 1].length() > 11) {
        String str_a = SplitStr[SplitStr.length - 1].replace(" ", "").replace("\\",
        "").replaceAll("and", ",").replaceAll(",", "+");
        String key_str = str_a.substring(0, 4);

        switch(key_str) {
            case "1998":
                bw1998.write(str_a);
                bw1998.newLine();
                break;
            case "1999":
                bw1999.write(str_a);
                bw1999.newLine();
                break;
            case "2000":
                bw2000.write(str_a);
                bw2000.newLine();
                break;
            case "2001":
                bw2001.write(str_a);
                bw2001.newLine();
                break;
            case "2002":
                bw2002.write(str_a);
                bw2002.newLine();
                break;
            case "2003":
                bw2003.write(str_a);
                bw2003.newLine();
                break;
            case "2004":
                bw2004.write(str_a);
                bw2004.newLine();
                break;
            case "2005":
                bw2005.write(str_a);
                bw2005.newLine();
                break;
            case "2006":

```

```
        bw2006.write(str_a);
        bw2006.newLine();
        break;
    case "2007":
        bw2007.write(str_a);
        bw2007.newLine();
        break;
    case "2008":
        bw2008.write(str_a);
        bw2008.newLine();
        break;
    case "2009":
        bw2009.write(str_a);
        bw2009.newLine();
        break;
    case "2010":
        bw2010.write(str_a);
        bw2010.newLine();
        break;
    case "2011":
        bw2011.write(str_a);
        bw2011.newLine();
        break;
    case "2012":
        bw2012.write(str_a);
        bw2012.newLine();
        break;
    case "2013":
        bw2013.write(str_a);
        bw2013.newLine();
        break;
    case "2014":
        bw2014.write(str_a);
        bw2014.newLine();
        break;
    case "2015":
        bw2015.write(str_a);
        bw2015.newLine();
        break;
    case "2016":
        bw2016.write(str_a);
        bw2016.newLine();
        break;
    case "2017":
```

```

        bw2017.write(str_a);
        bw2017.newLine();
        break;
    default:
        break;
    }
}
}
br.close();
bw1998.close();
bw1999.close();
bw2000.close();
bw2001.close();
bw2002.close();
bw2003.close();
bw2004.close();
bw2005.close();
bw2006.close();
bw2007.close();
bw2008.close();
bw2009.close();
bw2010.close();
bw2011.close();
bw2012.close();
bw2013.close();
bw2014.close();
bw2015.close();
bw2016.close();
bw2017.close();
}
}
}

```

#绘制相关性恐怖袭击事件的组数（Matlab）

```

data = xlsread('1998-2017 年具有相关性恐怖袭击事件的组数.xlsx');
x = data(:, 1);
y = data(:, 2);
bar(x, y)
grid on;
set(gca,'XTickLabelRotation',46)
for i = 1:length(y)
text(x(i),y(i)
120,num2str(y(i)),'VerticalAlignment','middle','HorizontalAlignment','center');
end

```

+

```

title('1998-2017 年具有相关性恐怖袭击事件的组数')
xlim([1997 2018])
xlabel('年份（年）');
ylabel('相关恐怖事件数量（组）');

```

#ARIMA 预测模型（Python）

```

import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
import numpy as np
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.stattools import adfuller
from pylab import show
from datetime import datetime
import warnings

warnings.filterwarnings('ignore')
def timestamp_datetime(value):
    dt = datetime.strptime(value, "%Y-%m")
    return dt

def read_data(aim_list_1, aim_list_2, file_name):
    _file = open(file_name, 'r')
    data = _file.readlines()
    dict_tmp = {}
    for data_i in data:
        array_sub_data = data_i.split("\t")
        time_ym = timestamp_datetime(array_sub_data[0])
        dict_tmp[time_ym] = int(array_sub_data[1])

    keys = sorted(dict_tmp.keys())
    for key in keys:
        aim_list_1.append(key)
        aim_list_2.append(dict_tmp[key])
    _file.close()
    return

def proper_model(timeseries, maxLag):
    init_bic = 10000000000
    for p in np.arange(maxLag):
        for q in np.arange(maxLag):
            model = ARMA(timeseries, order=(p, q))
            try:
                results_ARMA = model.fit(dis=0, method='css')

```

```

        except:
            continue
        bic = results_ARMA.bic
        if bic < init_bic:
            model_return = results_ARMA
            init_bic = bic
    print("p:", p, "q:", q)
    return model_return

def time_seq_no_diff(ym, num):
    tidx = pd.DatetimeIndex(ym, freq = None)
    dta = pd.Series(num, index = tidx)

    dta_diff= dta.diff(1)
    dta_diff = dta_diff.truncate(before= ym[1])
    plt.plot(dta,label='Original')
    plt.plot(dta_diff,label='1-Diff')
    plt.grid(color='k', linestyle='--')
    plt.legend(loc=2)
    plt.tight_layout()
    plt.show()
    test_stationarity(dta)
    fig = plt.figure(figsize=(9,6))
    ax1 = fig.add_subplot(211)
    fig = sm.graphics.tsa.plot_acf(dta,lags=30,ax=ax1)
    ax2 = fig.add_subplot(212)
    fig = sm.graphics.tsa.plot_pacf(dta,lags=30,ax=ax2)

    show()
    return

def time_seq_with_diff(ym, num):
    tidx = pd.DatetimeIndex(ym, freq = None)
    dta = pd.Series(num, index = tidx)

    dta= dta.diff(1)
    dta = dta.truncate(before= ym[1])

    plt.plot(dta,label='1-Diff')
    plt.grid(color='k', linestyle='--')
    plt.legend(loc=2)
    plt.tight_layout()
    plt.show()

    test_stationarity(dta)

```



```

fig = plt.figure(figsize=(9,6))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(dta,lags=30,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(dta,lags=30,ax=ax2)
show()
return dta

def test_stationarity(timeseries):
    dfctest = adfuller(timeseries, autolag='AIC')
    print(dfctest)
    return

def train_test(timeseries, ym, num):

    dta_bk = timeseries
    dta_train = timeseries.truncate(after= "2016-12")

    rama_mod = proper_model(dta_train, 9)

    predict_sunspots = rama_mod.predict('2017-01', '2017-12', dynamic=True)

    num_tmp = []
    num_tmp_pr = num.copy()
    for num_i in range(len(ym) - 1):
        num_tmp.append(dta_bk[dta_bk.index[num_i]])

    num_last = num[-13]
    sum_or = 0
    sum_pr = 0
    for i in range(12):
        num_tmp[-1 - i] = predict_sunspots[i]
        num_last = num_last + predict_sunspots[i]
        num_tmp_pr[-1 - i] = int(num_last)
        sum_or += num[-1 - i]
        sum_pr += int(num_last)
    print(num[-12: ])
    print(num_tmp_pr[-12: ])
    print(sum_or, sum_pr)
    print((sum_pr - sum_or) * 1.0 / sum_or)
    tidx = pd.DatetimeIndex(ym[1:], freq = None)
    dta_re = pd.Series(num_tmp, index = tidx)

    tidx_or = pd.DatetimeIndex(ym, freq = None)
    dta_or = pd.Series(num, index = tidx_or)

```

```

tidx_pr = pd.DatetimeIndex(ym, freq = None)
dta_pr = pd.Series(num_tmp_pr, index = tidx_pr)

```

```

plt.plot(dta_or,label='Original')
plt.plot(dta_pr,label='Predict')
plt.plot(dta_bk,label='Original-Diff')
plt.plot(dta_re,label='Predict-Diff')
plt.grid(color='k', linestyle='--')
plt.legend(loc=1)
plt.tight_layout()
plt.show()

```

```

def predict(timeseries, ym, num):

```

```

    rama_mod = proper_model(timeseries, 9)
    predict_sunspots = rama_mod.predict('2018-01', '2018-12', dynamic=True)
    list_re = [timeseries[timeseries.index[-1]],]
    list_time = [timestamp_datetime("2017-12"),]
    for pr_i in range(len(predict_sunspots)):
        list_re.append(predict_sunspots[pr_i])
        list_time.append(timestamp_datetime("2018-" + str(pr_i + 1)))

```

```

    num_last = num[-1]
    num_re = [num_last, ]
    sum_num = 0
    for i in range(12):
        num_last = num_last + predict_sunspots[i]
        num_re.append(int(num_last))
        sum_num += int(num_last)
    print(num_re[1: ])
    print(sum_num)
    tidx = pd.DatetimeIndex(list_time, freq = None)
    dta_re = pd.Series(list_re, index = tidx)
    tidx_or = pd.DatetimeIndex(ym, freq = None)
    dta_or = pd.Series(num, index = tidx_or)
    tidx_long = pd.DatetimeIndex(list_time, freq = None)
    dta_long = pd.Series(num_re, index = tidx_long)

```

```

plt.plot(dta_or,label='2015-2017')
plt.plot(dta_long,label='2018')
plt.plot(dta,label='2015-2017-Diff')
plt.plot(dta_re,label='2018-Diff')
plt.grid(color='k', linestyle='--')

```

```

plt.legend(loc = 1)
plt.tight_layout()
# plt.title("Philippines")
plt.title("World")
plt.show()

if __name__ == '__main__':
    ym = []
    num = []
    file_name = 'ym'
    read_data(ym, num, file_name)

    time_seq_no_diff(ym, num)

    dta = time_seq_with_diff(ym, num)
    train_test(dta, ym, num)
    predict(dta, ym, num)

```

#第一层聚类（Python）

```

import imp,sys
import csv
import codecs
from sklearn.cluster import KMeans
import numpy as np
np.set_printoptions(threshold = np.inf)

event_to_vector = {}
vector_to_event = {}
event_dict = {}
event_clusters = set()

list_cluster_center_point = []
list_cluster_center_tmp = []

data_set = []

event_feature_to_event_id = {}

with open('data.csv', 'r') as f:
    reader = csv.DictReader(f, delimiter='+')
    for row in reader:
        select_feature = []

```

```

if (row['eventid'] > '201412310106' and row['eventid'] < '201701010001' and
row['claimed'] == '0' and row['related'] != ''):
    if(row['eventid'] in event_dict.keys()):
        event_dict[row['eventid']].append(row['related'])
    else:
        event_dict[row['eventid']] = row['related']
    select_feature.append(float(row['extended']))
    select_feature.append(float(row['crit1']))
    select_feature.append(float(row['crit2']))
    select_feature.append(float(row['crit3']))
    select_feature.append(float(row['doubtterr']))
    if (row['alternative'] == ''):
        select_feature.append(float('-1'))
    else:
        select_feature.append(float(row['alternative']))
    select_feature.append(float(row['country']))
    select_feature.append(float(row['region']))
    select_feature.append(float(row['attacktype1']))
    select_feature.append(float(row['suicide']))
    select_feature.append(float(row['weaptype1']))
    if (row['weapsubtype1'] == ''):
        select_feature.append(float('27'))
    else:
        select_feature.append(float(row['weapsubtype1']))
    select_feature.append(float(row['targtype1']))

    if (row['targsubtype1'] == ''):
        select_feature.append(float('-1'))
    else:
        select_feature.append(float(row['targsubtype1']))

    if (row['natlty1'] == ''):
        select_feature.append(float('-1'))
    else:
        select_feature.append(float(row['natlty1']))

    if (row['nperps'] == ''):
        select_feature.append(float('-99'))
    else:
        select_feature.append(float(row['nperps']))

    if (row['claimmode'] == ''):
        select_feature.append(float('10'))
    else:

```

```

        select_feature.append(float(row['claimmode']))

if (row['nkill'] == ""):
    select_feature.append(float('0'))
else:
    select_feature.append(float(row['nkill']))

select_feature.append(float(row['property']))

if (row['propextent'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['propextent']))

select_feature.append(float(row['ishostkid']))

if (row['nhostkid'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['nhostkid']))

if (row['ransom'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['ransom']))

if (row['ransomamt'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['ransomamt']))

if (row['hostkidoutcome'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['hostkidoutcome']))

if (row['nreleased'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['nreleased']))

select_feature.append(float(row['INT_LOG']))
select_feature.append(float(row['INT_IDEO']))
select_feature.append(float(row['INT_MISC']))

```

```

select_feature.append(float(row['INT_ANY']))

event_to_vector[row['eventid']] = select_feature
data_set.append(select_feature)

select_feature_tmp = [str(i) for i in select_feature]
select_feature_str = ",".join(select_feature_tmp)
if(select_feature_str in event_feature_to_event_id.keys()):
    event_feature_to_event_id[select_feature_str] =
event_feature_to_event_id[select_feature_str] + ',' + row['eventid']
else:
    event_feature_to_event_id[select_feature_str] = row['eventid']

for(k, v) in event_dict.items():
    event_clusters.add(v)

for event in event_clusters:
    mark = 0
    for event_id in event.split(', '):
        if(not event_id in event_to_vector.keys()):
            mark = 1

    if(mark == 0):
        list_cluster_center_tmp[:] = []
        for event_id in event.split(', '):
            list_cluster_center_tmp.append(event_to_vector[event_id] )

        list_cluster_center_sum = [0 for n in range(0,
len(event_to_vector['201601200023']))]

        for point in list_cluster_center_tmp:
            for i in range(len(event_to_vector['201601200023'])):
                list_cluster_center_sum[i] = list_cluster_center_sum[i] + point[i]

            for i in range(len(event_to_vector['201601200023'])):
                list_cluster_center_sum[i] = list_cluster_center_sum[i] /
len(list_cluster_center_tmp)

        list_cluster_center_point.append(list_cluster_center_sum)

f.close()

X = np.array(data_set)

```

```
kmeans=KMeans(n_clusters=len(list_cluster_center_point),n_init=1,init=np.array(list
_cluster_center_point)).fit(X).predict(X)
```

```
for i in range(len(data_set)):
    print i ,
    print data_set[i] ,
    data_set_tmp = [str(j) for j in data_set[i]]
    print event_feature_to_event_id[".".join(data_set_tmp)] ,
    print kmeans[i] ,
    print
```

#第二层聚类（Python）

```
import imp,sys
import csv
import codecs
from sklearn.cluster import KMeans
import numpy as np
np.set_printoptions(threshold = np.inf)
```

```
event_to_vector = {}
vector_to_event = {}
event_dict = {}
event_clusters = set()
```

```
list_cluster_center_point = []
list_cluster_center_tmp = []
```

```
data_set = []
```

```
event_feature_to_event_id = {}
```

```
with open('data.csv', "r") as f:
    reader = csv.DictReader(f, delimiter='+')
```

```
    for row in reader:
        select_feature = []
```

```
        if (row['eventid'] > '201412310106' and row['eventid'] < '201701010001' and
row['claimed'] == '0' and row['related'] != ''):
            if(row['eventid'] in event_dict.keys()):
                event_dict[row['eventid']].append(row['related'])
            else:
                event_dict[row['eventid']] = row['related']
```

```

select_feature.append(float(row['extended']))
select_feature.append(float(row['crit1']))
select_feature.append(float(row['crit2']))
select_feature.append(float(row['crit3']))
select_feature.append(float(row['doubtterr']))
if (row['alternative'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['alternative']))
select_feature.append(float(row['country']))
select_feature.append(float(row['region']))
select_feature.append(float(row['attacktype1']))
select_feature.append(float(row['suicide']))
select_feature.append(float(row['weaptype1']))
if (row['weapsubtype1'] == ""):
    select_feature.append(float('27'))
else:
    select_feature.append(float(row['weapsubtype1']))
select_feature.append(float(row['targtype1']))

if (row['targsubtype1'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['targsubtype1']))

if (row['natlty1'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['natlty1']))

if (row['nperps'] == ""):
    select_feature.append(float('-99'))
else:
    select_feature.append(float(row['nperps']))

if (row['claimmode'] == ""):
    select_feature.append(float('10'))
else:
    select_feature.append(float(row['claimmode']))

if (row['nkill'] == ""):
    select_feature.append(float('0'))
else:

```



```

        select_feature.append(float(row['nkill']))

select_feature.append(float(row['property']))

if (row['propextent'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['propextent']))

select_feature.append(float(row['ishostkid']))

if (row['nhostkid'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['nhostkid']))

if (row['ransom'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['ransom']))

if (row['ransomamt'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['ransomamt']))

if (row['hostkidoutcome'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['hostkidoutcome']))

if (row['nreleased'] == ""):
    select_feature.append(float('-1'))
else:
    select_feature.append(float(row['nreleased']))

select_feature.append(float(row['INT_LOG']))
select_feature.append(float(row['INT_IDEO']))
select_feature.append(float(row['INT_MISC']))
select_feature.append(float(row['INT_ANY']))

event_to_vector[row['eventid']] = select_feature
data_set.append(select_feature)

```

```

        select_feature_tmp = [str(i) for i in select_feature]
        select_feature_str = ",".join(select_feature_tmp)
        if(select_feature_str in event_feature_to_event_id.keys()):
            event_feature_to_event_id[select_feature_str] =
event_feature_to_event_id[select_feature_str] + ',' + row['eventid']
        else:
            event_feature_to_event_id[select_feature_str] = row['eventid']

    for(k, v) in event_dict.items():
        event_clusters.add(v)

for event in event_clusters:
    mark = 0
    for event_id in event.split(', '):
        if(not event_id in event_to_vector.keys()):
            mark = 1

    if(mark == 0):
        list_cluster_center_tmp[:] = []
        for event_id in event.split(', '):
            list_cluster_center_tmp.append(event_to_vector[event_id] )

        list_cluster_center_sum = [0 for n in range(0,
len(event_to_vector['201601200023']))]

        for point in list_cluster_center_tmp:
            for i in range(len(event_to_vector['201601200023'])):
                list_cluster_center_sum[i] = list_cluster_center_sum[i] + point[i]

            for i in range(len(event_to_vector['201601200023'])):
                list_cluster_center_sum[i] = list_cluster_center_sum[i] /
len(list_cluster_center_tmp)

        list_cluster_center_point.append(list_cluster_center_sum)

f.close()

X = np.array(data_set)
kmeans=KMeans(n_clusters=len(list_cluster_center_point),n_init=1,init=np.array(list
_cluster_center_point)).fit(X).predict(X)

cluster_to_feature = {}

```

```

cluster_to_event_id = {}
for i in range(len(X)):
    if(kmeans[i] in cluster_to_feature.keys()):
        cluster_to_feature[kmeans[i]].append(data_set[i])

    else:
        new_list = []
        new_list.append(data_set[i])
        cluster_to_feature[kmeans[i]] = new_list

    if(kmeans[i] in cluster_to_event_id.keys()):
        data_set_tmp = [str(j) for j in data_set[i]]
        cluster_to_event_id[kmeans[i]] =
list(set(cluster_to_event_id[kmeans[i]].union(set([str(x) for x in
(event_feature_to_event_id[".".join(data_set_tmp)].split(',')))))

    else:
        data_set_tmp = [str(j) for j in data_set[i]]
        cluster_to_event_id[kmeans[i]] = [str(x) for x in
(event_feature_to_event_id[".".join(data_set_tmp)].split(','))

new_cluster_to_feature = {}
for cluster_index in cluster_to_feature.iterkeys():
    list_cluster_new_center_sum = [0 for n in range(0,
len(event_to_vector['201601200023']))]
    for point_new in cluster_to_feature[cluster_index]:
        for i in range(len(event_to_vector['201601200023'])):
            list_cluster_new_center_sum[i] = list_cluster_new_center_sum[i] +
point_new[i]

    for i in range(len(event_to_vector['201601200023'])):
        list_cluster_new_center_sum[i] = list_cluster_new_center_sum[i] /
len(cluster_to_feature[cluster_index])

    new_cluster_to_feature[cluster_index] = list_cluster_new_center_sum

new_cluster_center_to_event_id = {}
new_list_cluster_center_point = []

for cluster_index in new_cluster_to_feature.iterkeys():
    new_cluster_to_feature_tmp = [str(i) for i in
new_cluster_to_feature[cluster_index]]
    new_cluster_to_feature_str = ".".join(new_cluster_to_feature_tmp)

```

```

if(new_cluster_to_feature_str in new_cluster_center_to_event_id.keys()):

    new_cluster_center_to_event_id[new_cluster_to_feature_str].append(cluster_to_
event_id[cluster_index])
    else:
        new_cluster_center_to_event_id[new_cluster_to_feature_str]
        =
        (cluster_to_event_id[cluster_index])

    new_list_cluster_center_point.append(new_cluster_to_feature[cluster_index])

Y = np.array(new_list_cluster_center_point)
kmeans_y=KMeans(n_clusters=30, init='k-means++', n_init=10, max_iter=300,
tol=0.0001,
    precompute_distances='auto',
    verbose=0,
    random_state=None,
    copy_x=True,
    n_jobs=1,
    algorithm='auto').fit(Y).predict(Y)

for i in range(len(new_list_cluster_center_point)):
    print i ,
    print new_list_cluster_center_point[i] ,
    data_set_tmp_y = [str(j) for j in new_list_cluster_center_point[i]]
    print new_cluster_center_to_event_id[".".join(data_set_tmp_y)] ,
    print kmeans_y[i] ,
    print

```