第53卷第5期 南京航空航天大学学报 Vol. 53 No. 5
2021年10月 Journal of Nanjing University of Aeronautics & Astronautics Oct. 2021

DOI:10.16356/j.1005-2615.2021.05.006

自适应时间平滑的演化谱聚类

何 萍,姜玉麟,徐晓华,林惠惠,葛方毅,方 威,仁 祥
(扬州大学信息工程学院,扬州 225009)

摘要: 传统的聚类算法一般只适用于静态数据的处理,而真实世界的数据往往数据量大且变化多,静态的聚类算法不能为动态数据提供其演化规律的分析学习。演化数据的聚类,一方面要正确反映每一时刻数据的合理簇划分,另一方面又要使动态的聚类结果在演化过程中尽可能平滑。本文提出了一种自适应时间平滑的演化聚类框架,该模型考虑到当前时刻数据与历史时刻数据的未知关联,通过限定时间回溯的范围,自适应地寻找与当前快照最相关的历史快照,并通过有机融合基于Itakura-Saito距离的静态相似度和基于时间序列的动态相似度,计算各个时间片快照上的相似度矩阵。本文进一步提出了两种自适应时间平滑的演化谱聚类算法,从不同的角度定义时间代价,得到不同的演化聚类结果。在真实数据集上的实验表明这两种算法能够有效地利用历史数据,在聚类结果上准确性更高,时间平滑性也更好。

关键词: 演化数据;时间平滑性;Bregman散度;谱聚类
中图分类号: TP181 **文献标志码:** A **文章编号:** 1005-2615(2021)05-0700-08

Adaptive Time-Smoothed Evolutionary Spectral Clustering

HE Ping, *JIANG Yulin*, *XU Xiaohua*, *LIN Huihui*, *GE Fangyi*, *FANG Wei*, *REN Xiang*
(College of Information Engineering, Yangzhou University, Yangzhou 225009, China)

Abstract: Traditional clustering algorithms are generally only suitable for static data processing, while the real world data are often large and changeable, so static clustering algorithms cannot provide the analysis and learning of evolution rules for dynamic data. On one hand, the clustering of evolutionary data needs to reflect the reasonable cluster partition of data at each snapshot; on the other hand, it needs to make sure the dynamic clustering results are as smooth as possible. This paper proposes an adaptive time-smoothed evolutionary clustering framework, which takes into account of the unknown relationship between the current data and the historical data. By imposing a time window for backtracking, it adaptively finds the most relevant historical snapshot to the current snapshot. Meanwhile, it fuses the static similarity based on the Itakura-Saito distance and the dynamic similarity based on the time series to compute, so as so compute the similarity matrix on each snapshot. Under this framework, this paper further proposes two adaptive time-smoothed evolutionary spectral clustering algorithms, which define the time cost from different aspects, and obtain different evolutionary clustering results. Experiments on real datasets show that the two proposed algorithms can effectively utilize historical data, and achieve better clustering performance as well as better temporal smoothness.

Key words: evolutionary data; time smoothness; Bregman divergence; spectral clustering

聚类作为数据挖掘领域中一种非常有效的数据分析方式,其主要是将数据间相似度较高的数据

基金项目: 国家自然科学基金(61402395)资助项目;江苏省自然科学基金(BK20201430,BK20151314,BK20140492)资助项目。
收稿日期: 2020-09-25;**修订日期:** 2020-11-09
通信作者: 徐晓华,男,副教授,E-mail:arterx@gmail.com。

引用格式: 何萍,姜玉麟,徐晓华,等.自适应时间平滑的演化谱聚类[J].南京航空航天大学学报,2021,53(5):700-707.
HE Ping, JIANG Yulin, XU Xiaohua, et al. Adaptive time-smoothed evolutionary spectral clustering[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2021, 53(5):700-707.

样本划分到同一簇中，将相似度相差较大的划分到不同的簇中。传统的聚类算法往往在静态数据的处理上具有良好的效果，但在实际问题的处理中，数据往往是随时间的推移而变化的，通过聚类挖掘数据的演化机制，并且保证聚类结果在时间上尽可能平滑，即当前时间快照上的聚类结果应该与历史快照上的聚类结果要尽可能地相似。

传统的机器学习的基本假设是所有的数据都是独立同分布的，不会随着时间的推移而发生变化，也不会随着时间的推移出现数据的增加或衰减的情况。比如在文本的挖掘、图像的合成、分割等任务中，假设训练数据集和测试数据集的数据都是在一定的时间点，从一个概率分布中独立地抽取得到的。然而在一些实际应用问题的数据分布是随着时间动态变化的，例如，在新闻、博客和BBS等在线媒体中，人们讨论的话题大多数都会随着时间发生变化，即使对于同一个话题，一年前和当前的内容也不完全相同，这被称为概念漂移[1]。

图1演示的是演化数据随着时间的分布移动。图1(a～d)分别是不同时刻的数据分布。从图中可以看出，T_1～T_4时刻有3个簇，分别用红、绿、蓝表示。随着时间推移，图1(b～d)中3个簇的位置发生变化。由此可见，演化数据在时间的推移过程中，数据的位置分布和相互关系往往会发生变化。

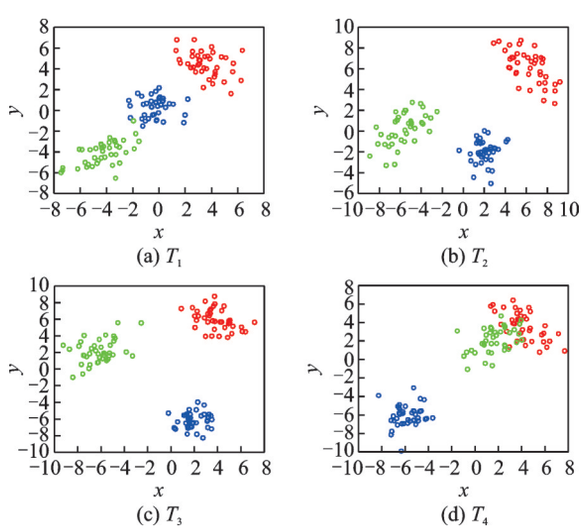


图1　演化数据示例

Fig.1　Illustration of evolutionary data

传统的聚类算法，如K均值和谱聚类，处理的是静态的数据，算法被要求在给定的数据集合上有尽可能好的聚类划分。演化数据上的聚类是这样一个学习任务：数据的分布随时间而变化，在每一时刻，为演化的数据作出新的聚类划分。

由于演化聚类处理的是随时间动态变化的数据，且在每个时刻都要产生一个聚类结果，因此任何时刻的聚类结果不仅需要反映数据的分布，并且前后时刻的聚类结果要尽可能相似，即保持相邻时刻聚类结果的平滑，不能出现较大的抖动。总体来说，现有的演化聚类算法主要有3个目标：首先，每一时刻上的数据聚类性能要尽可能好；其次，希望通过聚类发掘数据的演化机制，例如聚类的出现、分裂、消失等；最后，要充分利用历史信息帮助提高聚类性能、保持聚类结果的时间平滑性能，从而挖掘数据的演化规律[1]。

演化聚类的研究收到了广泛的关注，虽然已经提出了许多方法，但大多数算法并没有很好地考虑到数据在时序上的信息融合。Chakrabarti等[2-4]通过修改标准K-means的目标函数，提出了演化K-means。Chi等[5]将时序平滑思想引入谱聚类中，提出了PCQ(Preserving cluster quality)和PCM(Preserving cluster membership)两种演化谱聚类框架。Lin等[6]则是提出了一种分析动态社交网络及其演化的概率生成模型，该模型从贝叶斯的角度解决了演化聚类的问题，其假设在一段时间内社区的数目固定，采用迭代期望最大化算法的随机分块模型以及基于Dirchlet分布的概率模型来捕捉演化规律。Tang等[7]提出了一种新的演化聚类算法来分析动态网络，通过不同类型的节点组成网络，交替优化进行求解，但其计算成本较大。Folino等[8-9]将演化谱聚类的问题定义为多目标优化问题，认为其解是通过遗传算法求解得到，但在研究演化数据的过程中，数据点的多样性以及复杂性往往没有得到充分考虑。因此，Rana等[10]和Giulio等[11]主要通过利用邻域的方式，针对演化数据的动态性，对其演化的过程进行预测。Xu等[12-13]以及Yu等[14]则是提出了进化聚类框架，其能够精确地跟踪数据之间随时间变化之间的相似性，并静态聚类。Li等[15]则是提出了一种演化协同聚类算法，其所涉及的优化问题是非凸、非光滑和不可分离的。在上述的一些演化算法中，大多数不能很好地利用先验信息，只考虑相邻时间节点或者节点的邻域信息，认为相邻时刻的数据与当前时刻数据存在一定的线性关系，并不能很好地反映数据的演化机制。

为了克服上述问题，本文提出一种基于时间平滑性的演化聚类框架，通过设定滑动窗口的大小，自适应地寻找与当前数据最为近似的历史数据，而不是单纯地考虑前一时刻的数据，从而对历史数据进行更为合理的利用；有机融合了基于静态和基于时间序列的两种相似度计算方法，其能够较好地反映样本点之间的关系；最后将框架应用到谱聚类算法中，提出了两种自适应平滑的演化谱聚类算法

ESC-PCQ（Evolutionary spectral clustering-preserving cluster quality）和ESC-PCM（Evolutionary spectral clustering-preserving cluster membership）。

1 相关工作

在演化数据的聚类问题上，最常见的方法就是采用传统聚类的方式设计函数从而对每个时刻的数据进行处理，随后通过在代价函数上加入光滑正则项的方式来显示数据的动态演化。该类方法侧重于如何提高当前时刻数据的聚类性能以及如何保持在聚类过程中的平滑性。

1.1 在线式框架

Chakrabarti等[2]首先提出了一种泛化的在线式框架，并且将其应用到了K-means算法上得到演化K-means算法。此框架包括两个部分：快照聚类质量和时间损失。在其所提出的演化聚类框架中，每一时刻t上的聚类任务可以视为以下目标函数的最大化

$$sq(\boldsymbol{C}^t,\boldsymbol{M}^t)-cp\cdot hc(\boldsymbol{C}^{t-1},\boldsymbol{C}^t) \tag{1}$$

式中：$sq(\boldsymbol{C}^t,\boldsymbol{M}^t)$衡量了算法在当前时刻$t$数据上的聚类质量；$hc(\boldsymbol{C}^{t-1},\boldsymbol{C}^t)$则表示算法的时间损失；参数$cp$则为用户自定义的，主要用于约束时间损失所占大小。

1.2 时间平滑的演化谱聚类

Chi等[5]在进化聚类的框架思想上，提出两种结合时间平滑的演化谱聚类算法。这两种算法将时间平滑度与演化聚类相结合。在“平滑假设”的基础上进一步扩展，结合了图分割的标准割思想与平均关联。算法中代价函数的定义包括两部分，传统的聚类质量代价和引入规范的时间平滑代价。其中，快照聚类质量（CS），主要针对当前数据特征度量聚类结果的聚类质量，其中较高的快照代价意味着较差的聚类结果；时间损失（CT），根据聚类结果的拟合度，针对历史数据特征或者历史聚类结果，测量时间平滑度，其中，较高的时间代价意味着较差的时间平滑度。总代价函数定义为两个代价的线性组合

$$\mathrm{Cost}=\alpha\cdot\mathrm{CS}+(1-\alpha)\cdot\mathrm{CT} \tag{2}$$

式中：$0\leqslant\alpha\leqslant1$为用户分配的参数，反映了用户对聚类代价和时间代价的权衡。

两种演化谱聚类算法的区别在于如何定义时间代价CT。第1个算法为PCQ，它将当前分区应用于历史数据，产生的群集质量决定了时间代价。

它对式(2)中的CS定义为多路标准割函数NC，然而在实际问题求解中属于NP-hard[16]，因此采取宽松的方式进行处理，其表达式为

$$\mathrm{CS}=p-\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{W}}^t\boldsymbol{U}^t] \tag{3}$$

式中：p为簇数；$\widetilde{\boldsymbol{W}}^t=(\boldsymbol{D}^t)^{-1/2}\boldsymbol{W}^t(\boldsymbol{D}^t)^{-1/2}$，$\boldsymbol{D}^t$为$n\times n$对角矩阵；$\boldsymbol{D}^t(i,j)=\sum_1^n w^t(i,j)$，$\boldsymbol{U}^t$为满足$(\boldsymbol{U}^t)^{\mathrm{T}}\boldsymbol{U}^t=\boldsymbol{I}_k$正交条件的$n\times c$维矩阵，表示$t$时刻的簇指示矩阵。

因此，PCQ算法的具体代价函数为

$$\begin{aligned}\mathrm{Cost}_{\mathrm{PCQ}}&=\alpha\cdot\mathrm{CS}+(1-\alpha)\cdot\mathrm{CT}=\alpha\cdot p-\\&\alpha\cdot\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{W}}^t\boldsymbol{U}^t]+(1-\alpha)\cdot p-(1-\\&\alpha)\cdot\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{W}}^{t-1}\boldsymbol{U}^t]=p-\\&\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}(\alpha\widetilde{\boldsymbol{W}}^t+(1-\alpha)\widetilde{\boldsymbol{W}}^{t-1})\boldsymbol{U}^t]\end{aligned} \tag{4}$$

第2个算法PCM，其快照聚类质量与PCQ保持一致，区别在于时间损失的构造，主要通过比较当前时刻的簇划分与上一个时刻簇划分的差异来决定时间代价，其代价函数主要为

$$\begin{aligned}\mathrm{Cost}_{\mathrm{PCM}}&=\alpha\cdot\mathrm{CS}+(1-\alpha)\cdot\mathrm{CT}=\alpha\cdot p-\\&\alpha\cdot\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{W}}^t\boldsymbol{U}^t]+\frac{(1-\alpha)}{2}\cdot\\&\left\|\boldsymbol{U}^t(\boldsymbol{U}^t)^{\mathrm{T}}-\boldsymbol{U}^{t-1}(\boldsymbol{U}^{t-1})^{\mathrm{T}}\right\|^2=p-\\&\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}(\alpha\widetilde{\boldsymbol{W}}^t+(1-\alpha)\boldsymbol{U}^{t-1}(\boldsymbol{U}^{t-1})^{\mathrm{T}})\boldsymbol{U}^t]\end{aligned} \tag{5}$$

由此可见，PCM和PCQ的差异主要体现在历史代价的定义上，PCQ将当前时刻数据的簇划分应用到前一时刻的数据上来衡量聚类效果，而PCM则是衡量前后两个时刻簇划分之间的差异度来作为演化聚类的时间代价。

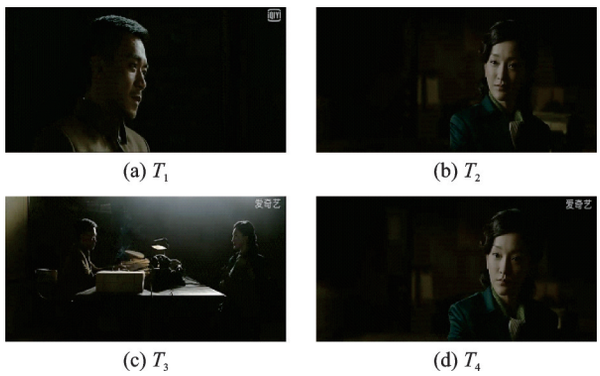
2 时间平滑的演化谱聚类

2.1 问题引入

在Chakrabarti等[2]所提出的在线式框架中，在时间损失上存在一定的局限性，它只考虑的是前一时刻的数据点对当前时刻数据点的影响，假设数据的演化结构是线性的，但在实际应用中一些演化数据在t时刻受到影响最大的有可能不是它的前一时刻，而是更为久远的历史数据。

如图2所示，截取电影中的4个时间戳的视频图片，图2(a)是主人公A出现，图2(b)为主人公B出现，图2(c)是主人公A和B同时出现，图2(d)则是主人公B出现，可以发现图2(d)聚类（或图像分割）与前一时刻图2(c)并没有太大的关联，反而与图2(b)关联性较强，即T_4时刻与T_2时刻的数据关联性较强。

因此在Chakrabarti等人所提出的演化聚类框架中，并没有有效地利用历史时刻的聚类信息，且前一时刻的历史时间代价也不尽可能小。



(a) T_1　(b) T_2　(c) T_3　(d) T_4

图 2　非线性演化数据示例

Fig.2　Illustration of non-linear evolutionary data

2.2　基本框架

给定数据集 $X=\{X^1,X^2,\cdots,X^T\}$，其中 $X^t(X^t\subseteq X)$ 表示 t 时刻所有的数据对象。在每个时刻 $t(1\leqslant t\leqslant T)$，都有一个新的数据集 X^t 到达。假设这个数据集可以用一个 $n\times n$ 的矩阵 $\boldsymbol{M}^t$ 来表示每对数据间的相互关系。在每个时刻 t，根据新的矩阵 $\boldsymbol{M}^t$ 和历史信息得到时刻 t 的聚类结果 $\boldsymbol{C}^t$。

考虑到 t 时刻到达的数据，对其造成影响的可能不是 $t-1$ 时刻的数据，而是与之前的某一时刻 $t-k(k<t)$ 有关，因此在针对时间损失部分对目标函数进行改写，其修改后的目标函数为

$$Q=sq(\boldsymbol{C}^t,\boldsymbol{M}^t)-\eta\cdot\min_{k=1,\cdots,r-1}\{hc(\boldsymbol{C}^{t-k},\boldsymbol{C}^t)\} \tag{6}$$

式中：$sq(\boldsymbol{C}^t,\boldsymbol{M}^t)$ 主要指在当前时间快照 t 上的聚类质量，而 $hc(\boldsymbol{C}^{t-k},\boldsymbol{C}^t)$ 则表示当前时间快照与回溯窗口范围内历史快照上的数据之间的时间损失；参数 $\eta>0$ 则是由用户自己定义的，其主要控制时间损失在目标函数中的权重，而 r 则表示时间回溯范围。

2.3　M^t 构造

在所提出的在线式框架中，$\boldsymbol{M}^t$ 相似度矩阵的计算主要包含两个部分：基于当前时刻的数据点间静态相似度和基于时间序列的动态相似度。对于当前时刻数据点的静态相似度，计算过程只使用当前时刻的信息而忽略其他的所有时刻信息；而后者则针对的是各个数据在随时间演化规律上的相似性。

在每对数据之间的静态相似度计算上，使用基于 Bregman 散度的 Itakura-Saito(IS)距离进行计算。Bregman 散度[17]是一种类似于距离度量的方式，主要用于衡量两者之间的差异大小。Bregman 散度可以认为是损失或者失真函数，考虑如下情况：假设点 x 是点 y 的失真或近似点，即可能通过对点 y 添加一些噪声形成点 x，损失函数的目的是度量点 x 近似点 y 导致的失真或者损失，其形式化定义如下。

给定一个严格凸函数 Φ，由该函数生成的 Bregman 散度为

$$d_\Phi(\boldsymbol{x},\boldsymbol{y})=\Phi(\boldsymbol{x})-\Phi(\boldsymbol{y})-\left\langle\nabla\Phi(\boldsymbol{y}),(\boldsymbol{x}-\boldsymbol{y})\right\rangle \tag{7}$$

式中：$\nabla\Phi(\boldsymbol{y})$ 为函数 Φ 在 $\boldsymbol{y}$ 点处的梯度，$(\boldsymbol{x}-\boldsymbol{y})$ 为向量 $\boldsymbol{x}$ 和向量 $\boldsymbol{y}$ 的差；$\left\langle\nabla\Phi(\boldsymbol{y}),(\boldsymbol{x}-\boldsymbol{y})\right\rangle$ 为 $\nabla\Phi(\boldsymbol{y})$ 与 $(\boldsymbol{x}-\boldsymbol{y})$ 的内积。

本文在衡量当前时间快照与历史快照之间的相似度 S^t 时，使用的是基于 Bregman 散度的 Itakura-Saito(IS)距离测度。

$$\mathrm{dist}_{\mathrm{IS}}(i,j)=\sum_{d=1}^{D}\left(\frac{x_i(d)}{x_j(d)}-\log\left(\frac{x_i(d)}{x_j(d)}\right)-1\right)$$

$$S^t(i,j)=\mathrm{e}^{-\frac{(\mathrm{dist}_{\mathrm{IS}}(i,j))^2}{2\sigma^2}} \tag{8}$$

时间序列上的动态相似度计算定义了一个改进的时间序列相关系数，具体为

$$\mathrm{Corr}^t(i,j,t)=\frac{\sum_{\tau=1}^{t}\alpha(\tau)(x_i^\tau-\mu_i^t)(x_j^\tau-\mu_j^t)}{\sqrt{\frac{\sum_{\tau=1}^{t}\alpha(\tau)(x_i^\tau-\mu_i^t)^2}{t}\cdot\frac{\sum_{\tau=1}^{t}\alpha(\tau)(x_j^\tau-\mu_j^t)^2}{t}}} \tag{9}$$

式中：$\alpha(t)=2^{1-t}$，μ_i^t 为 x_i 在 0～t 上的平均值，t 为当前时刻。最后合并两种相似度构成最终的相似度矩阵，则在 t 时刻的相似度为

$$M^t(i,j)=\theta\cdot S^t(i,j)+(1-\theta)\cdot\mathrm{Corr}^t(i,j,t) \tag{10}$$

式中 θ 表示时间序列之间的相似度在目标函数中所占比重。

2.4　ESC 算法

将上述框架结合谱聚类，得到两种演化谱聚类(Evolutionary spectral clustering，ESC)算法，分别是 ESC-PCQ 和 ESC-PCM。在 ESC-PCQ 中通过衡量当前时刻的簇划分应用到前 r 个时刻的历史数据上的最佳聚类效果来确定时间代价，而 ESC-PCM 则是通过衡量当前时刻的簇划分与用前 r 个时刻的历史数据的最小差异来确定时间代价。

2.4.1　基于 PCQ 的演化谱聚类

基于 PCQ 的算法框架中，通过设定滑动窗口大小，寻找利用当前数据的簇划分能够得到最好聚类质量的历史时刻，构造自适应的时间损失项。结合上述思想，令 $\widetilde{\boldsymbol{M}}^t=(\boldsymbol{D}_M^t)^{-1/2}\boldsymbol{M}^t(\boldsymbol{D}_M^t)^{-1/2}$ 表示 t 时刻的归一化相似度矩阵，$\widetilde{\boldsymbol{M}}^{t-k}=(\boldsymbol{D}_M^{t-k})^{-1/2}\boldsymbol{M}^{t-k}(\boldsymbol{D}_M^{t-k})^{-1/2}$ 表示 $t-k$ 时刻的归一化相似度矩阵，其中 $\boldsymbol{D}_M^t=\mathrm{diag}(\boldsymbol{M}^t1)$ 表示 t 时刻的度数矩阵。在此基础上，定义 ESC-PCQ 算法的总代

价函数为

$$\boldsymbol{Q}_{\mathrm{PCQ}}=sq(\boldsymbol{C}^t,\boldsymbol{M}^t)-\eta\cdot\min_{k=1,\cdots,r-1}\{hc(\boldsymbol{C}^{t-k},\boldsymbol{C}^t)\}=$$
$$p-\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{M}}^t\boldsymbol{U}^t]-\eta\cdot\min_{k=1,\cdots,r-1}\{p-$$
$$\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{M}}^{t-k}\boldsymbol{U}^t]\} \tag{11}$$

不难发现,式(11)等价于

$$-\min_{k=1,\cdots,r-1}\{\mathrm{Tr}[(U^t)^{\mathrm{T}}(\widetilde{\boldsymbol{M}}^t-\eta\cdot\widetilde{\boldsymbol{M}}^{t-k})\boldsymbol{U}^t]\}\Leftrightarrow$$
$$\max_{k=1,\cdots,r-1}\{\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}(\widetilde{\boldsymbol{M}}^t-\eta\cdot\widetilde{\boldsymbol{M}}^{t-k})\boldsymbol{U}^t]\} \tag{12}$$

为了求解式(12),对于$k=1,\cdots,r-1$,令

$$\boldsymbol{\Omega}^t_{\mathrm{PCQ}}(k)=\widetilde{\boldsymbol{M}}^t-\eta\cdot\widetilde{\boldsymbol{M}}^{t-k} \tag{13}$$

依次计算出$\boldsymbol{\Omega}^t_{\mathrm{PCQ}}(k)$,并对其进行特征值分解

$$[\boldsymbol{\Lambda}^t(k),\boldsymbol{U}^t(k)]=\mathrm{eigs}(\boldsymbol{\Omega}^t_{\mathrm{PCQ}}(k),c) \tag{14}$$

式中:$\boldsymbol{\Lambda}^t(k)=\{\lambda^t_1(k),\lambda^t_2(k),\cdots,\lambda^t_c(k)\}$,$\boldsymbol{U}^t(k)=\{u^t_1(k),u^t_2(k),\cdots,u^t_c(k)\}$。

对$k=1,\cdots,r-1$中所分解得到的特征值进行加和,即$\xi^t(k)=\sum_{i=1}^{c}\lambda^t_i(k)$,找到其中最大值所对应的$k$值记为$k^*$,即

$$k^*=\underset{k}{\mathrm{argmax}}\,(\xi^t(k)) \tag{15}$$

然后,将找到k^*所对应的前c个最大特征值所对应的特征向量记为$\boldsymbol{U}^t=\boldsymbol{U}^t(k^*)$,通过对$\boldsymbol{U}^t$进行行归一化,能够将顶点的簇指示矩阵的行向量投影到一个单位超球面上

$$\boldsymbol{Y}^t=(\boldsymbol{D}^t_U)^{-1/2}\boldsymbol{U}^t \tag{16}$$

式中:$\boldsymbol{D}^t_U=\mathrm{diag}(\mathrm{diag}(\boldsymbol{U}^t(\boldsymbol{U}^t)^{\mathrm{T}}))$,最后使用K-means计算$t$时刻的聚类结果。

ESC-PCQ的具体算法框架如算法1所示。

算法1 ESC-PCQ算法

输入:时间步$t(1\leqslant t\leqslant T)$,演化数据对象$X=\{X^1,X^2,\cdots,X^T\}$,权重参数$\eta$,聚类数目$c$

输出:每个时刻聚类结果$\boldsymbol{C}^1,\boldsymbol{C}^2,\cdots,\boldsymbol{C}^{\mathrm{T}}$

(1) 初始化:对X^1进行谱聚类,得到初始聚类结果C^1并输出;

(2) For t=2 to T

(3) 根据式(8)计算IS距离的相似度;

(4) 根据式(9)计算时间序列上的相似度;

(5) 根据式(10)进行相似度融合得$\boldsymbol{M}^t$;

(6) For $k=1$ to $r-1$

(7) 计算$\boldsymbol{\Omega}^t_{\mathrm{PCQ}}(k)=\widetilde{\boldsymbol{M}}^t-\eta\cdot\widetilde{\boldsymbol{M}}^{t-k}$;

(8) $[\boldsymbol{\Lambda}^t(k),\boldsymbol{U}^t(k)]=\mathrm{eigs}(\boldsymbol{\Omega}^t_{\mathrm{PCQ}}(k),c)$;

(9) End for

(10) $\xi^t(k)=\sum_{i=1}^{c}\lambda^t_i(k)$;

(11) $k^*=\underset{k}{\mathrm{argmax}}\,(\xi^t(k))$;

(12) $\boldsymbol{U}^t=\boldsymbol{U}^t(k^*)$;

(13) 归一化$\boldsymbol{U}^t$,并根据式(16)计算$\boldsymbol{Y}^t$;

(14) 使用K-means对$\boldsymbol{Y}^t$聚类,得到聚类$\boldsymbol{C}^t$;

(15) 输出$\boldsymbol{C}^t$;

(16) End for

2.4.2 基于PCM的演化谱聚类

在基于PCM的算法框架中,主要考虑到当前时刻的数据与历史时刻数据的差异,需要计算t时刻的簇划分$\boldsymbol{U}^t$与$t-k$时刻的簇划分$\boldsymbol{U}^{t-k}$之间的距离,进而构造自适应的时间损失,因此构造距离$\mathrm{dist}(\boldsymbol{U}^t,\boldsymbol{U}^{t-k})$函数,其具体形式为

$$\mathrm{dist}(\boldsymbol{U}^t,\boldsymbol{U}^{t-k})=\frac{1}{2}\left\|\boldsymbol{U}^t(\boldsymbol{U}^t)^{\mathrm{T}}-\boldsymbol{U}^{t-k}(\boldsymbol{U}^{t-k})^{\mathrm{T}}\right\|^2 \tag{17}$$

因此基于PCM的演化谱聚类算法的目标函数为

$$Q_{\mathrm{PCM}}=sq(\boldsymbol{C}^t,\boldsymbol{M}^t)-\eta\cdot\min_{k=1,\cdots,r-1}\left\{hc(\boldsymbol{C}^{t-k},\boldsymbol{C}^t)\right\}=$$
$$p-\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}\widetilde{\boldsymbol{M}}^t\boldsymbol{U}^t]-$$
$$\frac{\eta}{2}\cdot\min_{k=1,\cdots,r-1}\left\|\boldsymbol{U}^t(\boldsymbol{U}^t)^{\mathrm{T}}-\boldsymbol{U}^{t-k}(\boldsymbol{U}^{t-k})^{\mathrm{T}}\right\|^2 \tag{18}$$

不难发现,式(18)等价于

$$-\min_{k=1,\cdots,r-1}\{\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}(\widetilde{\boldsymbol{M}}^t-\eta\boldsymbol{U}^{t-k}(\boldsymbol{U}^{t-k})^{\mathrm{T}})\boldsymbol{U}^t]\}\Leftrightarrow$$
$$\max_{k=1,\cdots,r-1}\{\mathrm{Tr}[(\boldsymbol{U}^t)^{\mathrm{T}}(\widetilde{\boldsymbol{M}}^t-\eta\boldsymbol{U}^{t-k}(\boldsymbol{U}^{t-k})^{\mathrm{T}})\boldsymbol{U}^t]\} \tag{19}$$

为了求解式(19),对于$k=1,\cdots,r-1$,令

$$\boldsymbol{\Omega}^t_{\mathrm{PCM}}(k)=\widetilde{\boldsymbol{M}}^t-\eta\boldsymbol{U}^{t-k}(\boldsymbol{U}^{t-k})^{\mathrm{T}} \tag{20}$$

依次计算出$\boldsymbol{\Omega}^t_{\mathrm{PCM}}(k)$,并对其进行特征值分解

$$[\boldsymbol{\Lambda}^t(k),\boldsymbol{U}^t(k)]=\mathrm{eigs}(\boldsymbol{\Omega}^t_{\mathrm{PCM}}(k),c) \tag{21}$$

式中:$\boldsymbol{\Lambda}^t(k)=\{\lambda^t_1(k),\lambda^t_2(k),\cdots,\lambda^t_c(k)\}$;$\boldsymbol{U}^t(k)=\{u^t_1(k),u^t_2(k),\cdots,u^t_c(k)\}$。

对$k=1,\cdots,r-1$中所分解得到的特征值进行加和,即$\xi^t(k)=\sum_{i=1}^{c}\lambda^t_i(k)$,找到其中最大值所对应的$k$值记为$k_*$,然后将$k_*$所对应的前$c$个最大特征值所对应的特征向量记为$\boldsymbol{U}^t=\boldsymbol{U}^t(k_*)$,根据式(14)对$\boldsymbol{U}^t$进行归一化得到$\boldsymbol{Y}^t$,最后使用K-means计算$t$时刻的聚类结果。

ESC-PCM的具体算法如算法2所示。

算法2 ESC-PCM算法

输入:时间步$t(1\leqslant t\leqslant T)$,演化数据对象$X=\{X^1,X^2,\cdots,X^T\}$,权重参数$\eta$,聚类数目$c$

输出:每个时刻聚类结果$\boldsymbol{C}^1,\boldsymbol{C}^2,\cdots,\boldsymbol{C}^T$

(1) 初始化:对$\boldsymbol{X}^1$进行谱聚类,得到初始聚类结果$\boldsymbol{C}^1$并输出;

(2) For t=2 to T

(3) 根据式(8)计算IS距离的相似度；

(4) 根据式(9)计算时间序列上的相似度；

(5) 根据式(10)进行相似度融合得M^t；

(6) For $k=1$ to $r-1$

(7) 计算$\boldsymbol{\Omega}_{\mathrm{PCM}}^t(k)=\widetilde{\boldsymbol{M}}^t-\eta\boldsymbol{U}^{t-k}(\boldsymbol{U}^{t-k})^{\mathrm{T}}$；

(8) $[\boldsymbol{\Lambda}^t(k),\boldsymbol{U}^t(k)]=\mathrm{eigs}(\boldsymbol{\Omega}_{\mathrm{PCM}}^t(k),c)$；

(9) End for

(10) $\xi^t(k)=\sum_{i=1}^{c}\lambda_i^t(k)$；

(11) $k_*=\underset{k}{\mathrm{argmax}}\,(\xi^t(k))$；

(12) $\boldsymbol{U}^t=\boldsymbol{U}^t(k_*)$；

(13) 归一化$\boldsymbol{U}^t$，并根据式(16)计算$\boldsymbol{Y}^t$；

(14) 使用K-means对$\boldsymbol{Y}^t$聚类，得到聚类$\boldsymbol{C}^t$；

(15) 输出$\boldsymbol{C}^t$；

(16) End for

3　实验与分析

为了验证两种ESC算法的聚类性能，在真实数据集上，通过比对标准谱聚类、演化K-means以及演化谱聚类算法，验证算法的聚类性能。在参数的设置上，令$\eta=0.125$，$r=6$。

3.1　数据集

第1个验证数据集是Iris数据集。由于Iris数据集样本数较少，因此在初始时刻后的数据是通过对原始数据添加随机噪声所产生的二维数据集，并分配到10个不同的时刻进行模拟演化，噪声分布满足$N(0,0.5)$。Iris数据集是鸢尾花的特征数据集，一共有150个样本数据，每个样本的属性分别为鸢尾花萼片的长与宽以及花瓣的长与宽。

第2个验证数据集是MNIST数据集，是由数字0～9的手写数字构成的字符数据集(图3)。MNIST数据集有784维，共分为10类。从MNIST数据集的每个类中随机选取600个样本，并把数据随机分配成10等份，分配到10个不同的时刻进行模拟演化。因此每个时刻的样本有600个，属于10个类别。

图3　MNIST部分数据样本示例

Fig.3　Illustration of data samples in MNIST dataset

3.2　评价指标

采用了3种评价标准来衡量算法性能。快照聚类质量衡量当前时间片的聚类质量。

$$sq(\boldsymbol{C}^t)=\sum_{\boldsymbol{x}^t\in\boldsymbol{U}^t}\left(1-\min_{\boldsymbol{c}^t\in\boldsymbol{C}^t}\left\|\boldsymbol{c}^t-\boldsymbol{x}^t\right\|\right)\tag{22}$$

时间损失衡量当前时间片的聚类结果与历史时间的聚类结果的差距，时间损失越小越好。

$$hc(\boldsymbol{C}^t,\boldsymbol{C}^{t-k})=\min_{f:[k]\to[k]}\left\|\boldsymbol{c}_i^t-\boldsymbol{c}_{f(i)}^{t-k}\right\|\tag{23}$$

式中f表示将t时刻的簇心映射到$t-k$时刻簇心的函数。

NMI(Normalized mutual information)是归一化互信息，NMI值越大，聚类质量越高。

$$\mathrm{NMI}(\boldsymbol{X},\boldsymbol{Y})=\frac{I(\boldsymbol{X},\boldsymbol{Y})}{\sqrt{H(\boldsymbol{X})H(\boldsymbol{Y})}}\tag{24}$$

式中：$I(X,Y)$表示随机变量X和Y之间的互信息，而$H(\cdot)$表示随机变量的熵。

3.3　实验结果

实验环境采用的是1台Intel Core i3 M 390 2.67 GHz，RAM 4 GB的计算机，所有实验在Windows 7系统的Matlab R2019a环境下执行。

图4～6给出了在Iris数据集上快照聚类质量、时间损失和NMI的性能比较，其中横轴表示时间戳T_1到T_{10}，从图中可以看出，本文所提出的2种ESC算法的快照聚类质量和NMI高于其他算法，且时间损失也较小。

图5中在T_5时刻之后，两种ESC算法在时间损失上与前一时刻没有较大变化。而演化K-means和演化谱聚类算法则有较大的起伏，这是因为这两种算法在目标函数的构造上只与前一时刻比较，当其分布并不与历史时刻分布一致时容易出现较大的抖动。

图7～9分别为在MNIST数据集上的快照聚类质量、时间损失以及NMI度量指标的实验结果。从图可知，本文提出的两种ESC算法整体而言在对演化数据的聚类问题上聚类的质量较高且时间损失较低，其中ESC-PCM因为更侧重于时间损失比ESC-PCQ更低一些，但快照聚类质量却相当。

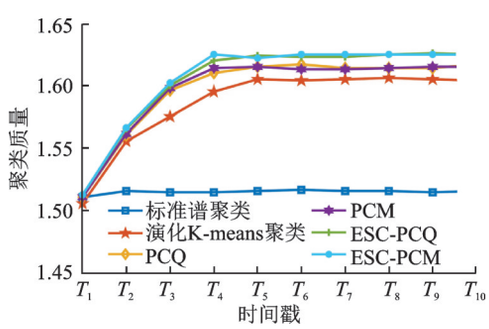


图4　Iris数据集快照聚类质量的比较

Fig.4　Comparison of snapshots quality on Iris dataset

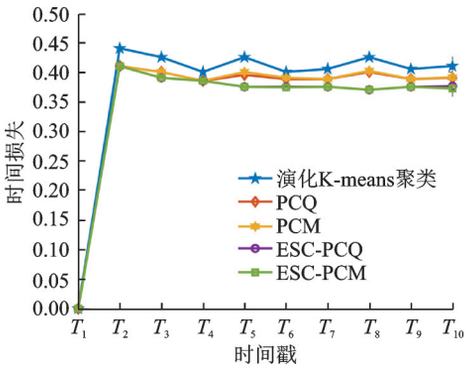


图 5 Iris 数据集时间损失的比较

Fig.5 Comparison of history cost on Iris dataset

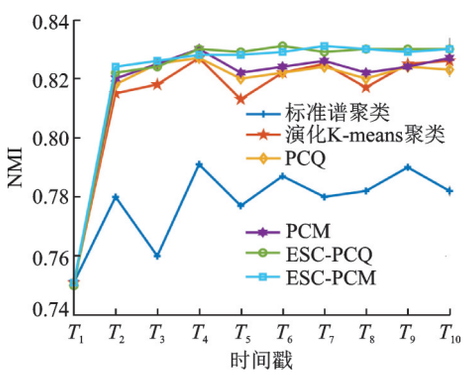


图 6 Iris 数据集 NMI 的比较

Fig.6 Comparison of NMI on Iris dataset

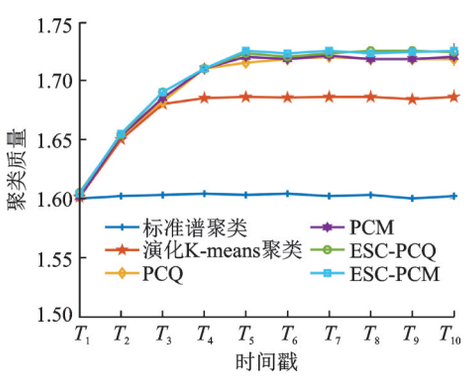


图 7 MNIST 数据集快照聚类质量的比较

Fig.7 Comparison of snapshots quality on MNIST dataset

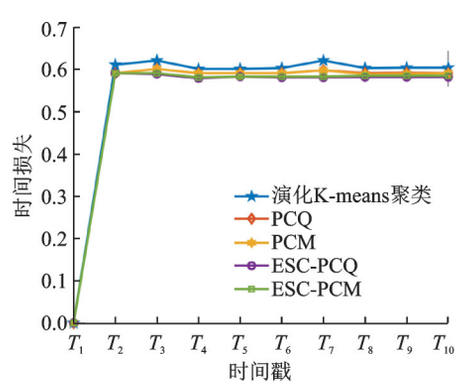


图 8 MNIST 数据集时间损失的比较

Fig.8 Comparison of history cost on MNIST dataset

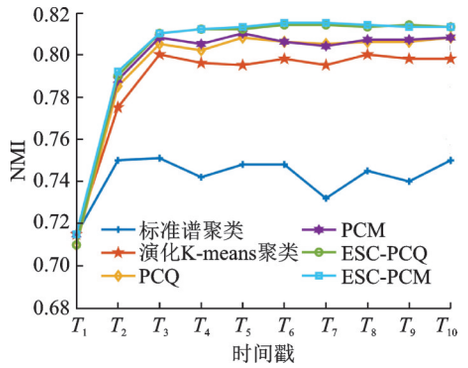


图 9 MNIST 数据集 NMI 的比较

Fig.9 Comparison of NMI on MNIST dataset

为了讨论不同参数对算法性能的影响，以 ESC-PCM 算法为例，在 Iris 数据集上分析回溯范围的参数 r 以及时间损失权重参数 η 对于 NMI 指标的影响。从图 10～11 可以看出，在回溯范围参数 $r=0$ 时，即不设置回溯范围，聚类的性能较低，而随着 r 值的不断增大，聚类的性能不断增长而逐渐持平，但随着 r 值范围逐渐覆盖到数据的全部时间，其计算成本较高，且聚类性能提升不大。另外，在约束时间损失的参数 η 的设置上，如果设置时间损失的权重过大（如 $\eta=0.25, 0.5$），其聚类的性能反而会降低。由此可见，本文采用的 $r=6$，$\eta=0.125$ 是一组性能较好的参数设置。

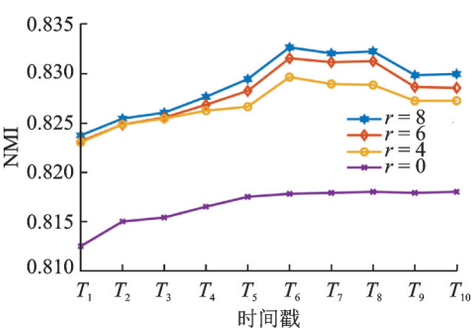


图 10 参数 r 对性能的影响

Fig.10 Influence of parameter r on performance

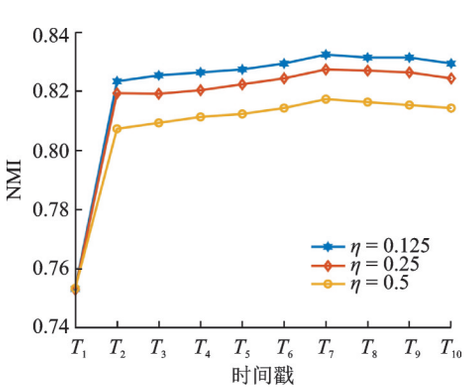


图 11 参数 η 对性能的影响

Fig.11 Influence of parameter η on performance

4 结　论

本文主要针对演化数据的聚类问题,提出了自适应时间平滑的演化谱聚类。考虑当前时刻的簇划分与前r个时刻的历史数据存在一定的关联,对时间回溯窗口进行设定,自适应地寻找与当前快照最为相关的历史快照。在相似度矩阵的构造上有机融合了基于IS的静态相似度以及基于时间序列的动态相似度,并结合谱聚类提出了两种自适应平滑的演化谱聚类算法ESC-PCQ和ESC-PCM。两种算法相较于其他演化聚类算法,在聚类效果上更好且更为平滑。然而,如果设定较大的时间回溯范围,会导致计算代价变高。因此,如何在保持或者降低时间代价的基础上进一步提高演化聚类的性能,将是下一步研究的工作重点。

参考文献:

[1] 张长水,张见闻.演化数据的学习[J].计算机学报,2013,36(2):310-316.
ZHANG Changshui, ZHANG Jianwen. Learning of evolutionary data[J]. Chinese Journal of Computers, 2013, 36(2): 310-316.

[2] CHAKRABARTI D, KUMAR R, TOMKINS A. Evolutionary clustering [C]//Proceedings of 12th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Philadelphia PA USA: ACM, 2006: 554-560.

[3] CHAKRABARTI D, SAMMUT C, WEBB G. Graph mining [C]//Proceedings of Encyclopedia of Machine Learning and Data Mining. Boston, MA: Springer,2017: 581-584.

[4] CHAKRABARTI D, FUNIAK S, CHANG J, et al. Joint label inference in networks[C]//Proceedings of 27th International World Wide Web Conference. Lyon France: ACM,2018: 483-487.

[5] CHI Y, SONG X, ZHOU D, et al. Evolutionary spectral clustering by incorporating temporal smoothness[C]//Proceedings of 13th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. San Jose California USA: ACM, 2007: 153-162.

[6] LIN Y R, CHI Y, ZHU S, et al. Analyzing communities and their evolutions in dynamic social networks [J]. ACM Transactions on Knowledge Discovery from Data, 2009, 3(2): 1-31.

[7] TANG L, WANG X, LIU H. Community detection via heterogeneous interaction analysis[J]. Data Mining and Knowledge Discovery, 2012, 25(1): 1-33.

[8] FOLINO F, PIZZUTI C. An evolutionary multiobjective approach for community discovery in dynamic networks[J]. IEEE Transactions on Knowledge & Data Engineering, 2014, 26(8): 1838-1852.

[9] FOLINO F, PIZZUTI C. Multiobjective evolutionary community detection for dynamic networks[C]//Proceedings of Genetic and Evolutionary Computation Conference. Portland, Oregon, USA: ACM, 2010: 535-536.

[10] RANA C, JAIN S K. An evolutionary clustering algorithm based on temporal features for dynamic recommender systems[J]. Swarm & Evolutionary Computation, 2014, 14: 21-30.

[11] GIULIO R, LUCA P, DINO P, et al. Tiles: An online algorithm for community discovery in dynamic social networks[J]. Machine Learning, 2017, 106(8): 1231-1241.

[12] XU K S, KLIGER M, HERO A O. Adaptive evolutionary clustering[J]. Data Mining & Knowledge Discovery, 2014, 28(2): 304-336.

[13] XU K S, HERO A O. Dynamic stochastic blockmodels for time-evolving social networks[J]. IEEE Journal of Selected Topics in Signal Processing, 2014, 8(4): 552-562.

[14] YU S, LIU M, DOU W, et al. Networking for big data: A survey[J]. IEEE Communications Surveys & Tutorials, 2017, 19(1): 531-549.

[15] LI R, ZHANG W, ZHAO Y, et al. Sparsity learning formulations for mining time-varying data[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 27(5): 1411-1423.

[16] SHI J, MALIK J M. Normalized cuts and image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888-905.

[17] BREGMAN L M. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming[J]. USSR Computational Mathematics & Mathematical Physics, 1967, 7(3): 200-217.

(编辑:夏道家)