K-平均算法

维基百科，自由的百科全书

***k*-平均算法**源于信号处理中的一种向量量化方法，现在则更多地作为一种聚类分析方法流行于数据挖掘领域。*k*-平均聚类的目的是：把n个点（可以是样本的一次观察或一个实例）划分到k个聚类中，使得每个点都属于离他最近的均值（此即聚类中心）对应的聚类，以之作为聚类的标准。这个问题将归结为一个把数据空间划分为Voronoi cells的问题。

这个问题在计算上是困难的（NP困难），不过存在高效的启发式算法。一般情况下，都使用效率比较高的启发式算法，它们能够快速收敛于一个局部最优解。这些算法通常类似于通过迭代优化方法处理高斯混合分布的最大期望算法（EM算法）。而且，它们都使用聚类中心来为数据建模；然而*k*-平均聚类倾向于在可比较的空间范围内寻找聚类，期望-最大化技术却允许聚类有不同的形状。

k-平均聚类与*k*-近邻之间没有任何关系（后者是另一流行的机器学习技术）。

目录

算法描述

已知观测集$(x_1, x_2, \ldots, x_n)$，其中每个观测都是一个d-维实向量，*k*-平均聚类要把这n个观测划分到k个集合中(k≤n),使得组内平方和（WCSS within-cluster sum of squares）最小。换句话说，它的目标是找到使得下式满足的聚类S_i，

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

其中$\boldsymbol{\mu}_i$是S_i中所有点的均值。

历史源流

虽然其思想能够追溯到1957年的Hugo Steinhaus [1] ，术语“*k*-均值”于1967年才被James MacQueen [2] 首次使用。标准算法则是在1957年被Stuart Lloyd作为一种脉冲码调制的技术所提出，但直到1982年才被贝尔实验室公开出版 [3] 。在1965年，E.W.Forgy发表了本质上相同的方法，所以这一算法有时被称为Lloyd-Forgy方法。更高效的版本则被Hartigan and Wong提出（1975/1979） [4][5][6]。

算法

标准算法

最常用的算法使用了迭代优化的技术。它被称为*k*-平均算法而广为使用，有时也被称为Lloyd算法（尤其在计算机科学领域）。已知初始的*k*个均值点$\boldsymbol{m}_1^{(1)}, \ldots, \boldsymbol{m}_k^{(1)}$,算法的按照下面两个步骤交替进行[7] ：

- 分配(Assignment)：将每个观测分配到聚类中，使得组内平方和（WCSS）达到最小。

因为这一平方和就是平方后的欧氏距离，所以很直观地把观测分配到离它最近得均值点即可 [8] 。（数学上，这意味依照由这些均值点生成的Voronoi图来划分上述观测）。

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - m_i^{(t)} \right\|^2 \leq \left\| x_p - m_j^{(t)} \right\|^2 \forall j, 1 \leq j \leq k \right\}$$

其中每个$\boldsymbol{x}_p$都只被分配到一个确定的聚类S^t中，尽管在理论上它可能被分配到2个或者更多的聚类。

- 更新(Update)：对于上一步得到的每一个聚类，以聚类中观测值的图心，作为新的均值点。

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

因为算术平均是最小平方估计，所以这一步同样减小了目标函数组内平方和（WCSS）的值。

这一算法将在对于观测的分配不再变化时收敛。由于交替进行的两个步骤都会减小目标函数WCSS的值，并且分配方案只有有限种，所以算法一定会收敛于某一（局部）最优解。注意：使用这一算法无法保证得到全局最优解。

这一算法经常被描述为“把观测按照距离分配到最近的聚类”。标准算法的目标函数是组内平方和（WCSS），而且按照“最小平方和”来分配观测，确实是等价于按照最小欧氏距离来分配观测的。如果使用不同的距离函数来代替（平方）欧氏距离，可能使得算法无法收敛。然而，使用不同的距离函数，也能得到*k*-均值聚类的其他变体，如球体*k*-均值算法和*k*-中心点算法。

初始化方法

通常使用的初始化方法有Forgy和随机划分(Random Partition)方法 [9] 。Forgy方法随机地从数据集中选择*k*个观测作为初始的均值点；而随机划分方法则随机地为每一观测指定聚类，然后运行“更新(Update)”步骤,即计算随机分配的各聚类的图心，作为初始的均值点。Forgy方法易于使得初始均值点散开，随机划分方法则把均值点都放到靠近数据集中心的地方。参考Hamerly et al的文章 [9] ，可知随机划分方法一般更适用于*k*-调和均值和模糊*k*-均值算法。对于期望-最大化(EM)算法和标准*k*-均值算法，Forgy方法作为初始化方法的表现会更好一些。

这是一个启发式算法，无法保证收敛到全局最优解，并且聚类的结果会依赖于初始的聚类。又因为算法的运行速度通常很快，所以一般都以不同的起始状态运行多次来得到更好的结果。不过，在最差的情况下，*k*-均值算法会收敛地特别慢：尤其是已经证明了存在这一的点集（甚至在2维空间中），使得*k*-均值算法收敛的时间达到指数级（$2^{\Omega(n)}$）[10]。好在在现实中，这样的点集几乎不会出现：因为*k*-均值算法的平滑运行时间是多项式时间的[11]。

注：把“分配”步骤视为“期望”步骤，把“更新”步骤视为“最大化步骤”，可以看到，这一算法实际上是广义期望-最大化算法（GEM）的一个变体。

复杂度

在d维空间中找到k-均值聚类问题的最优解的计算复杂度：

- NP-hard：一般欧式空间中，即使目标聚类数仅为2[12][13]
- NP困难：平面中，不对聚类数目k作限制[14]
- 如果k和d都是固定的，时间复杂度为$O(n^{dk+1}logn)$,其中n为待聚类的观测点数目[15]

相比之下，Lloyds算法的运行时间通常为$O(nkdi)$,k和d定义如上，i为直到收敛时的迭代次数。如果数据本身就有一定的聚类结构，那么收敛所需的迭代数目通常是很少的，并且进行少数迭代之后，再进行迭代的话，对于结果的改善效果很小。鉴于上述原因，Lloyds算法在实践中通常被认为几乎是线性复杂度的。

下面有几个关于这一算法复杂度的近期研究：

- Lloyd's *k*-均值算法具有多项式平滑运行时间。对于落在空间$[0,1]^d$任意的n点集合，如果每一个点都独立地受一个均值为0，标准差为σ的正态分布所影响，那么*k*-均值算法的期望运行时间上界为$O(n^{34}k^{34}d^8log^4(n)/\sigma^6)$，即对于$n,k,i,d$和$1/\sigma$都是多项式时间的[11]。
- 在更简单的情况下，有更好的上界。例如[16]，在整数网格$\{1,\dots,M\}^d$中，*k*-均值算法运行时间的上界为$O(dn^4M^2)$。

算法的变体

更多的讨论

使得*k*-均值算法效率很高的两个关键特征同时也被经常被视为它最大的缺陷：

- 聚类数目k是一个输入参数。选择不恰当的k值可能会导致糟糕的聚类结果。这也是为什么要进行特征检查来决定数据集的聚类数目了。
- 收敛到局部最优解，可能导致反直观”的错误结果。

k-均值算法的一个重要的局限性即在于它的聚类模型。这一模型的基本思想在于：得到相互分离的球状聚类，在这些聚类中，均值点趋向收敛于聚类中心。 一般会希望得到的聚类大小大致相当，这样把每个观测都分配到离它最近的聚类中心（即均值点）就是比较正确的分配方案。

k-均值聚类的结果也能理解为由均值点生成的Voronoi cells。

相关应用

k-均值聚类（尤其是使用如Lloyd's算法的启发式方法的聚类）即使是在巨大的数据集上也非常容易部署实施。正因为如此，它在很多领域都得到的成功的应用，如市场划分、机器视觉、 地质统计学[17]、天文学和农业等。它经常作为其他算法的预处理步骤，比如要找到一个初始设置。

向量的量化

k-均值起源于信号处理领域，并且现在也能在这一领域找到应用。例如在计算机图形学中，色彩量化的任务，就是要把一张图像的色彩范围减少到一个固定的数目*k*上来。*k*-均值算法就能很容易地被用来处理这一任务，并得到不错的结果。其它得向量量化的例子有非随机抽样，在这里，为了进一步的分析，使用*k*-均值算法能很容易的从大规模数据集中选出*k*个合适的不同观测。

聚类分析

在聚类分析中，*k*-均值算法被用来将输入数据划分到*k*个部分(聚类)中。 然而，纯粹的*k*-均值算法并不是非常灵活，同样地，在使用上有一定局限（不过上面说到得向量量化，确实是一个理想的应用场景）。特别是，当没有额外的限制条件时，参数*k*是很难选择的（真如上面讨论过的一样）。算法的另一个限制就是它不能和任意的距离函数一起使用、不能处理非数值数据。而正是为了满足这些使用条件，许多其他的算法才被发展起来。

特征学习

在（半）监督学习或无监督学习中，*k*-均值聚类被用来进行特征学习（或字典学习）步骤[18]。基本方法是，首先使用输入数据训练出一个*k*-均值聚类表示，然后把任意的输入数据投射到这一新的特征空间。 *k*-均值的这一应用能成功地与自然语言处理和计算机视觉中半监督学习的简单线性分类器结合起来。在对象识别任务中，它能展现出与其他复杂特征学习方法（如自动编码器、受限Boltzmann机等）相当的效果。然而，相比复杂方法，它需要更多的数据来达到相同的效果，因为每个数据点都只贡献了一个特征（而不是多重特征）。

与其他统计机器学习方法的关系

k-均值聚类，以及它与EM算法的联系，是高斯混合模型的一个特例。很容易能把*k*-均值问题一般化为高斯混合模型[19]。另一个*k*-均值算法的推广则是*k*-SVD算法，后者把数据点视为“编码本向量”的稀疏线性组合。而*k*-均值对应于使用单编码本向量的特殊情形（其权重为1）[20]。

Mean Shift 聚类

基本的Mean Shift聚类要维护一个与输入数据集规模大小相同的数据点集。初始时，这一集合就是输入集的副本。然后对于每一个点，用一定距离范围内的所有点的均值来迭代地替换它。与之对比，*k*-均值把这样的迭代更新限制在（通常比输入数据集小得多的）K个点上，而更新这些点时，则利用了输入集中与之相近的所有点的均值（亦即，在每个点的Voronoi划分内）。还有一种与*k*-均值类似的Mean shift算法，即 似然Mean shift，对于迭代变化的集合，用一定距离内在输入集中所有点的均值来更新集合里的点[21]。Mean Shift聚类与*k*-均值聚类相比，有一个优点就是不用指定聚类数目，因为Mean shift倾向于找到尽可能少的聚类数目。然而，Mean shift会比*k*-均值慢得多，并且同样需要选择一个“宽度”参数。和*k*-均值一样，Mean shift算法有许多变体。

主成分分析（PCA）

有一些研究[22][23]表明，*k*-均值的放松形式解（由聚类指示向量表示），可由主成分分析中的主成分给出，并且主成分分析由主方向张成的子空间与聚类图心空间是等价的。不过，主成分分析是*k*-均值聚类的有效放松形式并不是一个新的结果(如，见[24])，并且还有的研究结果直接揭示了关于聚类图心子空间是由主成分方向张成的这一论述的反例[25]。

独立成分分析(ICA)

有研究表明[26]，在稀疏假设以及输入数据经过白化的预处理后，*k*-均值得到的解就是独立成分分析的解。这一结果对于解释*k*-均值在特征学习方面的成功应用很有帮助。

双向过滤

k-均值算法隐含地假设输入数据的顺序不影响结果。双向过滤与*k*-均值算法和Mean shift算法类似之处在于它同样维护着一个迭代更新的数据集（亦是被均值更新）。然而，双向过滤限制了均值的计算只包含了在输入数据中顺序相近的点[21]，这使得双向过滤能够被应用在图像去噪等数据点的空间安排是非常重要的问题中。

相似问题

目标函数是使得聚类平方误差最小化的算法还有*k*-中心点算法，该方法保持聚类的中心在一个真实数据点上，亦即使用中心而非图心作为均值点。

参考资料

1. Steinhaus, H. Sur la division des corps matériels en parties. Bull. Acad. Polon. Sci. 1957, **4** (12): 801–804. MR 0090073. Zbl 0079.16403 （法语）.
2. MacQueen, J. B. Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press: 281–297. 1967 [2009-04-07]. MR 0214227. Zbl 0214.46201.
3. Lloyd, S. P. Least square quantization in PCM. Bell Telephone Laboratories Paper. 1957. Published in journal much later: Lloyd., S. P. Least squares quantization in PCM (PDF). IEEE Transactions on Information Theory. 1982, **28** (2): 129–137 [2009-04-15]. doi:10.1109/TIT.1982.1056489.
4. E.W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics. 1965, **21**: 768–769.
5. J.A. Hartigan. Clustering algorithms. John Wiley & Sons, Inc. 1975.
6. Hartigan, J. A.; Wong, M. A. Algorithm AS 136: A *k*-Means Clustering Algorithm. Journal of the Royal Statistical Society, Series C. 1979, **28** (1): 100–108. JSTOR 2346830.
7. MacKay, David. Chapter 20. An Example Inference Task: Clustering (PDF). Information Theory, Inference and Learning Algorithms. Cambridge University Press. 2003: 284–292. ISBN 0-521-64298-1. MR 2012999. （原始内容存档于2016-02-17）.
8. Since the square root is a monotone function, this also is the minimum Euclidean distance assignment.
9. Hamerly, G. and Elkan, C. Alternatives to the *k*-means algorithm that find better clusterings (PDF). Proceedings of the eleventh international conference on Information and knowledge management (CIKM). 2002.
10. Vattani., A. *k*-means requires exponentially many iterations even in the plane (PDF). Discrete and Computational Geometry. 2011, **45** (4): 596–616. doi:10.1007/s00454-011-9340-1.
11. Arthur, D.; Manthey, B.; Roeglin, H. *k*-means has polynomial smoothed complexity. Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS). 2009.
12. Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. NP-hardness of Euclidean sum-of-squares clustering. Machine Learning. 2009, **75**: 245–249. doi:10.1007/s10994-009-5103-0.
13. Dasgupta, S. and Freund, Y. Random Projection Trees for Vector Quantization. Information Theory, IEEE Transactions on. July 2009, **55**: 3229–3242. arXiv:0805.1390. doi:10.1109/TIT.2009.2021326.
14. Mahajan, M.; Nimbhorkar, P.; Varadarajan, K. The Planar *k*-Means Problem is NP-Hard. Lecture Notes in Computer Science. 2009, **5431**: 274–285. doi:10.1007/978-3-642-00202-1_24.
15. Inaba, M.; Katoh, N.; Imai, H. Applications of weighted Voronoi diagrams and randomization to variance-based *k*-clustering. Proceedings of 10th ACM Symposium on Computational Geometry. 332–339. 1994. doi:10.1145/177424.178042.
16. Arthur; Abhishek Bhowmick. A theoretical analysis of Lloyd's algorithm for *k*-means clustering (Thesis). 2009. [1] (http://www.cse.iitk.ac.in/users/bhowmick/lloyd.pdf)
17. Honarkhah, M and Caers, J, 2010, *Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling (http://dx.doi.org/10.1007/s11004-010-9276-7)*, Mathematical Geosciences, 42: 487 - 517
18. Coates, Adam; Ng, Andrew Y. Learning feature representations with *k*-means (PDF). (编) G. Montavon, G. B. Orr, K.-R. Müller. Neural Networks: Tricks of the Trade. Springer. 2012. （原始内容 (PDF)存档于2013-07-06）.
19. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP. Section 16.1. Gaussian Mixture Models and *k*-Means Clustering. Numerical Recipes: The Art of Scientific Computing 3rd. New York: Cambridge University Press. 2007. ISBN 978-0-521-88068-8.
20. Template:Cite Journal
21. Little, M.A.; Jones, N.S. Generalized Methods and Solvers for Piecewise Constant Signals: Part I (PDF). Proceedings of the Royal Society A. 2011.
22. H. Zha, C. Ding, M. Gu, X. He and H.D. Simon. Spectral Relaxation for *k*-means Clustering (PDF). Neural Information Processing Systems vol.14 (NIPS 2001) (Vancouver, Canada). Dec 2001: 1057–1064.
23. Chris Ding and Xiaofeng He. *k*-means Clustering via Principal Component Analysis (PDF). Proc. of Int'l Conf. Machine Learning (ICML 2004). July 2004: 225–232.
24. Drineas, P.; A. Frieze; R. Kannan; S. Vempala; V. Vinay. Clustering large graphs via the singular value decomposition (PDF). Machine learning. 2004, **56**: 9–33 [2012-08-02]. doi:10.1023/b:mach.0000033113.59016.96.
25. Cohen, M.; S. Elder; C. Musco; C. Musco; M. Persu. Dimensionality reduction for *k*-means clustering and low rank approximation (Appendix B). ArXiv. 2014 [2014-11-29].
26. Alon Vinnikov and Shai Shalev-Shwartz. *k*-means Recovers ICA Filters when Independent Components are Sparse (PDF). Proc. of Int'l Conf. Machine Learning (ICML 2014). 2014.

外部链接

- Numerical Example of*k*-means clustering
- Application example which uses*k*-means clustering to reduce the number of colors in images
- Interactive demo of the*k*-means-algorithm(Applet)
- An example of multithreaded application which uses*k*-means in Java
- *k*-means application in php
- *k*-means application in image retrieval
- Another animation of the*k*-means-algorithm

取自“https://zh.wikipedia.org/w/index.php?title=K平均算法&oldid=47632211”

本页面最后修订于2018年1月2日 (星期二) 13:14。

本站的全部文字在知识共享 署名-相同方式共享 3.0协议之条款下提供，附加条款亦可能应用。（请参阅使用条款）
Wikipedia®和维基百科标志是维基媒体基金会的注册商标；维基™是维基媒体基金会的商标。
维基媒体基金会是在美国佛罗里达州登记的501(c)(3)免税、非营利、慈善机构。