k-means clustering

In data mining, ***k*-means clustering** is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard), however there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data, however *k*-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

Description

Given a set of observations ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* sets ($k \leq n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

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

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

History

The term "*k*-means" was first used by James MacQueen in 1967,[1] though the idea goes back to Hugo Steinhaus in 1957.[2] The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published outside Bell labs until 1982.[3] In 1965, E.W.Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy, too.[4] A more efficient version was proposed and published in Fortran by Hartigan and Wong in 1975/1979.[5][6]

Algorithms

Standard algorithm

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the ***k*-means algorithm**; it is also referred to as **Lloyd's algorithm**, particularly in the computer science community.

Given an initial set of *k* means $\mathbf{m}_1^{(1)},\ldots,\mathbf{m}_k^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:[7]

Assignment step: Assign each observation to the cluster whose mean is closest to it (i.e. partition the observations according to the Voronoi diagram generated by the means).

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

where each x_p is assigned to exactly one $S^{(t)}$, even if it could be is assigned to two or more of them.

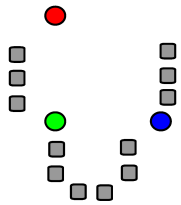
Update step: Calculate the new means to be the centroids of the observations in the new clusters.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

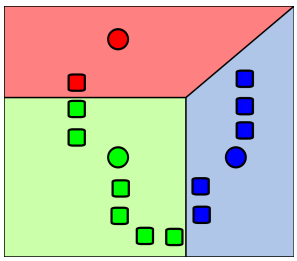
The algorithm has converged when the assignments no longer change.

Commonly used initialization methods are Forgy and Random Partition.[8] The Forgy method randomly chooses *k* observations from the data set and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to Hamerly et al.,[8] the Random Partition method is generally preferable for algorithms such as the *k*-harmonic means and fuzzy *k*-means. For expectation maximization and standard *k*-means algorithms, the Forgy method of initialization is preferable.

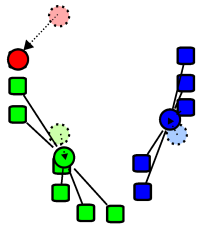
Demonstration of the standard algorithm



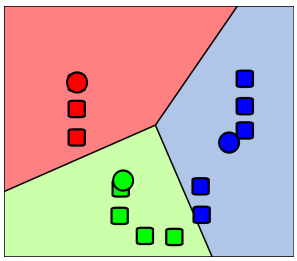
1) *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).



2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



3) The centroid of each of the *k* clusters becomes the new mean.



4) Steps 2 and 3 are repeated until convergence has been reached.

As it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum, and the result may depend on the initial clusters. As the algorithm is usually very fast, it is common to run it multiple times with different starting conditions. However, in the worst case, *k*-means can be very slow to converge: in particular it has been shown that there exist certain point sets, even in 2 dimensions, on which *k*-means takes exponential time, that is $2^{\Omega(n)}$, to converge.[9] These point sets do not seem to arise in practice: this is corroborated by the fact that the smoothed running time of *k*-means is polynomial.[10]

The "assignment" step is also referred to as **expectation step**, the "update step" as **maximization step**, making this algorithm a variant of the *generalized* expectation-maximization algorithm.

Complexity

Regarding computational complexity, the *k*-means clustering problem for observations in *d* dimensions is:

- NP-hard in general Euclidean space *d* even for 2 clusters [11][12]
- NP-hard for a general number of clusters *k* even in the plane [13]
- If *k* and *d* are fixed, the problem can be exactly solved in time $O(n^{dk+1} \log n)$, where *n* is the number of entities to be clustered [14]

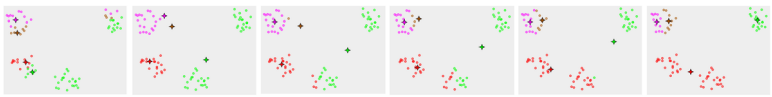
Thus, a variety of heuristic algorithms are generally used.

- The k-means algorithm discussed below has polynomial smoothed running time. It is shown that [10] for arbitrary set of n points in $[0, 1]^d$, if each point is independently perturbed by a normal distribution with mean 0 and variance σ^2, then the expected running time of k-means algorithm is bounded by $O(n^{34}k^{34}d^8 log^4(n)/\sigma^6)$, which is a polynomial in n, k, d and $1/\sigma$.
- Better bounds are proved for simple cases. For example,[15] showed that the running time of k-means algorithm is bounded by $O(dn^4M^2)$ for n points in an integer lattice $\{1, \ldots, M\}^d$.

Variations

- Fuzzy C-Means Clustering is a soft version of K-means, where each data point has a fuzzy degree of belonging to each cluster.
- Gaussian mixture models trained with expectation-maximization algorithm (EM algorithm) maintains probabilistic assignments to clusters, instead of deterministic assignments, and multivariate Gaussian distributions instead of means.
- Several methods have been proposed to choose better starting clusters. One recent proposal is k-means++.
- The filtering algorithm uses kd-trees to speed up each k-means step.[16]
- Some methods attempt to speed up each k-means step using coresets[17] or the triangle inequality.[18]
- Escape local optima by swapping points between clusters.[6]
- The Spherical k-means clustering algorithm is suitable for directional data.[19]
- The Minkowski metric weighted k-means deals with the problem of noise features by assigning weights to each feature per cluster[20]

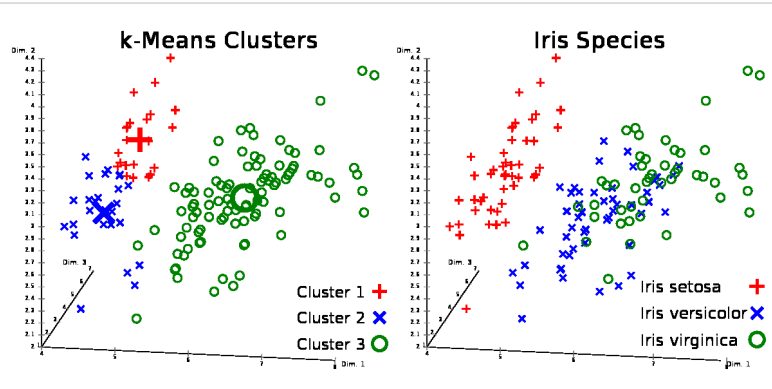
Discussion



A typical example of the k-means convergence to a local minimum. In this example, the result of k-means clustering (the right figure) contradicts the obvious cluster structure of the data set. The small circles are the data points, the four ray stars are the centroids (means). The initial configuration is on the left figure. The algorithm converges after five iterations presented on the figures, from the left to the right. The illustration was prepared with the Mirkes Java applet.[21]

The two key features of *k*-means which make it efficient are often regarded as its biggest drawbacks:

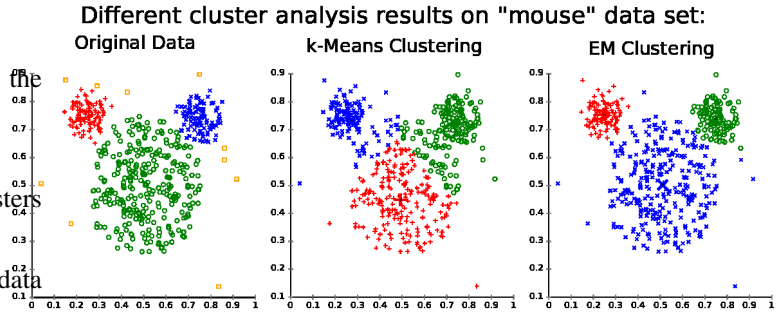
- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter.
- The number of clusters *k* is an input parameter: an inappropriate choice of *k* may yield poor results. That is why, when performing k-means, it is important to run diagnostic checks for determining the number of clusters in the data set.
- Convergence to a local minimum may produce counterintuitive ("wrong") results (see example in Fig.).



k-means clustering result for the Iris flower data set and actual species visualized using ELKI. Cluster means are marked using larger, semi-transparent symbols.

A key limitation of *k*-means is its cluster model. The concept is based on spherical clusters that are separable in a way so that the mean value converges towards the cluster center. The clusters are expected to be of similar size, so that the assignment to the nearest cluster center is the correct assignment. When for example applying *k*-means with a value of $k = 3$ onto the well-known Iris

flower data set, the result often fails to separate the three Iris species contained in the data set. With $k = 2$, the two visible clusters (one containing two species) will be discovered, whereas with $k = 3$ one of the two clusters will be split into two even parts. In fact, $k = 2$ is more appropriate for this data set, despite the data set containing 3 *classes*. As with any other clustering algorithm, the *k*-means result relies on the data set to satisfy the assumptions made by the clustering algorithms. It works well on some data sets, while failing on others.



k-means clustering and EM clustering on an artificial dataset ("mouse"). The tendency of *k*-means to produce equi-sized clusters leads to bad results, while EM benefits from the Gaussian distribution present in the data set

The result of *k*-means can also be seen as the Voronoi cells of the cluster means. Since data is split halfway between cluster means, this can lead to suboptimal splits as can be seen in the "mouse" example. The Gaussian models used by the Expectation-maximization algorithm (which can be seen as a generalization of *k*-means) are more flexible here by having both variances and covariances. The EM result is thus able to accommodate clusters of variable size much better than *k*-means as well as correlated clusters (not in this example).

Applications of the algorithm

k-means clustering in particular when using heuristics such as Lloyd's algorithm is rather easy to implement and apply even on large data sets. As such, it has been successfully used in various topics, ranging from market segmentation, computer vision, geostatistics,[22] and astronomy to agriculture. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

Relation to other statistical machine learning algorithms

k-means clustering, and its associated expectation-maximization algorithm, is a special case of a Gaussian mixture model, specifically, the limit of taking all covariances as diagonal, equal, and small. It is often easy to generalize a *k*-means problem into a Gaussian mixture model.[23]

Mean shift clustering

Basic mean shift clustering algorithms maintain a set of data points the same size as the input data set. Initially, this set is copied from the input set. Then this set is iteratively replaced by the mean of those points in the set that are within a given distance of that point. By contrast, *k*-means restricts this updated set to *k* points usually much less than the number of points in the input data set, and replaces each point in this set by the mean of all points in the *input set* that are closer to that point than any other (e.g. within the Voronoi partition of each updating point). A mean shift algorithm that is similar then to *k*-means, called *likelihood mean shift*, replaces the set of points undergoing replacement by the mean of all points in the input set that are within a given distance of the changing set.[24] One of the advantages of mean shift over *k*-means is that there is no need to choose the number of clusters, because mean shift is likely to find only a few clusters if indeed only a small number exist. However, mean shift can be much slower than *k*-means, and still requires selection of a bandwidth parameter. Mean shift has soft variants much as *k*-means does.

Principal component analysis (PCA)

It was asserted in [25][26] that the relaxed solution of k-means clustering, specified by the cluster indicators, is given by the PCA (principal component analysis) principal components, and the PCA subspace spanned by the principal directions is identical to the cluster centroid subspace. However, that PCA is a useful relaxation of k-means clustering was not a new result (see, for example,[27]), and it is straightforward to uncover counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.

Bilateral filtering

k-means implicitly assumes that the ordering of the input data set does not matter. The bilateral filter is similar to K-means and mean shift in that it maintains a set of data points that are iteratively replaced by means. However, the bilateral filter restricts the calculation of the (kernel weighted) mean to include only points that are close in the ordering of the input data.[24] This makes it applicable to problems such as image denoising, where the spatial arrangement of pixels in an image is of critical importance.

Similar problems

The set of squared error minimizing cluster functions also includes the k-medoids algorithm, an approach which forces the center point of each cluster to be one of the actual points, i.e., it uses medoids in place of centroids.

Software

Free

- Apache Mahout k-Means [28]
- CrimeStat implements two spatial K-means algorithms, one of which allows the user to define the starting locations.
- ELKI contains k-means (with Lloyd and MacQueen iteration, along with different initializations such as k-means++ initialization) and various more advanced clustering algorithms
- MLPACK [29] contains a K-Means implementation
- R kmeans [30] implements a variety of algorithms[1][3][6]
- SciPy vector-quantization [31]
- Silverlight widget demonstrating k-means algorithm [32]
- PostgreSQL extension for k-means [33]
- CMU's GraphLab Clustering library [34] Efficient multicore implementation for large scale data.
- Weka contains k-means and a few variants of it, including k-means++ and x-means.
- Spectral Python [35] contains methods for unsupervised classification including a K-means clustering method.
- scikit learn [36] machine learning in Python contains a K-Means implementation

Commercial

- IDL Cluster, Clust_Wts
- *Mathematica* ClusteringComponents function [37]
- MATLAB kmeans [38]
- SAS FASTCLUS [39]
- VisuMap kMeans Clustering [40]

Source code

- ELKI and Weka are written in Java and include k-means and variations
- K-means application in PHP,[41] using VB,[42] using Perl,[43] using C++,[44] using Matlab,[45] using Ruby,[46][47] using Python with scipy,[48] using X10[49]
- A parallel out-of-core implementation in C[50]
- An open-source collection of clustering algorithms, including k-means, implemented in Javascript.[51] Online demo.[52]

Visualization, animation and examples

- ELKI can visualize k-means using Voronoi cells and Delaunay triangulation for 2D data. In higher dimensionality, only cluster assignments and cluster centers are visualized
- Demos of the K-means-algorithm[53][54][55][56][57]
- K-means and K-medoids (Applet), University of Leicester[21]
- Clustergram - cluster diagnostic plot - for visual diagnostics of choosing the number of (k) clusters (R code)[58]

References

[1] MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations" (http://projecteuclid.org/euclid.bsmsp/1200512992). **1**. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. MR0214227. Zbl 0214.46201. . Retrieved 2009-04-07.

[2] Steinhaus, H. (1957). "Sur la division des corps matériels en parties" (in French). *Bull. Acad. Polon. Sci.* **4** (12): 801–804. MR0090073. Zbl 0079.16403.

[3] Lloyd, S. P. (1957). "Least square quantization in PCM". *Bell Telephone Laboratories Paper*. Published in journal much later: Lloyd., S. P. (1982). "Least squares quantization in PCM" (http://www.cs.toronto.edu/~roweis/csc2515-2006/readings/lloyd57.pdf). *IEEE Transactions on Information Theory* **28** (2): 129–137. doi:10.1109/TIT.1982.1056489. . Retrieved 2009-04-15.

[4] E.W. Forgy (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". *Biometrics* **21**: 768–769.

[5] J.A. Hartigan (1975). *Clustering algorithms*. John Wiley & Sons, Inc..

[6] Hartigan, J. A.; Wong, M. A. (1979). "Algorithm AS 136: A K-Means Clustering Algorithm". *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **28** (1): 100–108. JSTOR 2346830.

[7] MacKay, David (2003). "Chapter 20. An Example Inference Task: Clustering" (http://www.inference.phy.cam.ac.uk/mackay/itprnn/ps/284.292.pdf). *Information Theory, Inference and Learning Algorithms* (http://www.inference.phy.cam.ac.uk/mackay/itila/book.html). Cambridge University Press. pp. 284–292. ISBN 0-521-64298-1. MR2012999. .

[8] Hamerly, G. and Elkan, C. (2002). "Alternatives to the k-means algorithm that find better clusterings" (http://charlotte.ucsd.edu/users/elkan/cikm02.pdf). *Proceedings of the eleventh international conference on Information and knowledge management (CIKM)*. .

[9] Vattani., A. (2011). "k-means requires exponentially many iterations even in the plane" (http://cseweb.ucsd.edu/users/avattani/papers/kmeans-journal.pdf). *Discrete and Computational Geometry* **45** (4): 596–616. doi:10.1007/s00454-011-9340-1. .

[10] Arthur, D.; Manthey, B.; Roeglin, H. (2009). "k-means has polynomial smoothed complexity". *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*.

[11] Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). "NP-hardness of Euclidean sum-of-squares clustering". *Machine Learning* **75**: 245–249. doi:10.1007/s10994-009-5103-0.

[12] Dasgupta, S. and Freund, Y. (July 2009). "Random Projection Trees for Vector Quantization". *Information Theory, IEEE Transactions on* **55**: 3229–3242. arXiv:0805.1390. doi:10.1109/TIT.2009.2021326.

[13] Mahajan, M.; Nimbhorkar, P.; Varadarajan, K. (2009). "The Planar k-Means Problem is NP-Hard". *Lecture Notes in Computer Science* **5431**: 274–285. doi:10.1007/978-3-642-00202-1_24.

[14] Inaba, M.; Katoh, N.; Imai, H. (1994). "Applications of weighted Voronoi diagrams and randomization to variance-based *k*-clustering". Proceedings of 10th ACM Symposium on Computational Geometry. pp. 332–339. doi:10.1145/177424.178042.

[15] Arthur; Abhishek Bhowmick (2009). "A theoretical analysis of Lloyd's algorithm for k-means clustering".

[16] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silverman, R.; Wu, A. Y. (2002). "An efficient k-means clustering algorithm: Analysis and implementation" (http://www.cs.umd.edu/~mount/Papers/pami02.pdf). *IEEE Trans. Pattern Analysis and Machine Intelligence* **24**: 881–892. doi:10.1109/TPAMI.2002.1017616. . Retrieved 2009-04-24.

[17] Frahling, G.; Sohler, C. (2006). "A fast k-means implementation using coresets" (http://www.frahling.de/Gereon_Frahling/Publications_files/A fast k-means implementation using Coresets (Frahling, Sohler).pdf). *Proceedings of the twenty-second annual symposium on Computational geometry (SoCG)*. .

[18] Elkan, C. (2003). "Using the triangle inequality to accelerate k-means" (http://www-cse.ucsd.edu/~elkan/kmeansicml03.pdf). *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*. .

[19] Dhillon, I. S.; Modha, D. M. (2001). "Concept decompositions for large sparse text data using clustering". *Machine Learning* **42** (1): 143–175.

[20] Amorim, R. C.; Mirkin, B (2012). "Minkowski metric, feature weighting and anomalous cluster initializing in K-Means clustering". *Pattern Recognition* **45** (3): 1061–1075. doi:10.1016/j.patcog.2011.08.012.

[21] E.M. Mirkes, K-means and K-medoids applet (http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html). University of Leicester, 2011.

[22] 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

[23] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 16.1. Gaussian Mixture Models and k-Means Clustering" (http://apps.nrbook.com/empanel/index.html#pg=842). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8. .

[24] Little, M.A.; Jones, N.S. (2011). "Generalized Methods and Solvers for Piecewise Constant Signals: Part I" (http://www.maxlittle.net/publications/pwc_filtering_arxiv.pdf). *Proc. Roy. Soc. A.*. .

[25] H. Zha, C. Ding, M. Gu, X. He and H.D. Simon (Dec. 2001). "Spectral Relaxation for K-means Clustering" (http://ranger.uta.edu/~chqding/papers/Zha-Kmeans.pdf). *Neural Information Processing Systems vol.14 (NIPS 2001)* (Vancouver, Canada): 1057–1064. .

[26] Chris Ding and Xiaofeng He (July 2004). "K-means Clustering via Principal Component Analysis" (http://ranger.uta.edu/~chqding/papers/KmeansPCA1.pdf). *Proc. of Int'l Conf. Machine Learning (ICML 2004)*: 225–232. .

[27] Drineas, P.; A. Frieze, R. Kannan, S. Vempala, V. Vinay (2004). "Clustering large graphs via the singular value decomposition" (http://www.cc.gatech.edu/~vempala/papers/dfkvv.pdf). *Machine learning* **56**: 9–33. . Retrieved 2012-08-02.

[28] http://cwiki.apache.org/MAHOUT/k-means-clustering.html

[29] http://www.mlpack.org

[30] http://stat.ethz.ch/R-manual/R-patched/library/stats/html/kmeans.html

[31] http://docs.scipy.org/doc/scipy/reference/cluster.vq.html

[32] http://www.codeding.com/?article=14

[33] http://pgxn.org/dist/kmeans/

[34] http://graphlab.org/toolkits/clustering/

[35] http://spectralpython.sourceforge.net/algorithms.html#k-means-clustering

[36] http://scikit-learn.org/dev/modules/generated/sklearn.cluster.KMeans.html

[37] http://reference.wolfram.com/mathematica/ref/ClusteringComponents.html

[38] http://www.mathworks.com/access/helpdesk/help/toolbox/stats/kmeans.html

[39] http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/fastclus_toc.htm

[40] http://www.visumap.com/index.aspx?p=Products

[41] http://www25.brinkster.com/denshade/kmeans.php.htm

[42] K-Means Clustering Tutorial: Download (http://people.revoledu.com/kardi/tutorial/kMean/download.htm)

[43] Perl script for Kmeans clustering (http://www.lwebzem.com/cgi-bin/k_means/test3.cgi)

[44] Antonio Gulli's coding playground: K-means in C (http://codingplayground.blogspot.com/2009/03/k-means-in-c.html)

[45] K-Means Clustering Tutorial: Matlab Code (http://people.revoledu.com/kardi/tutorial/kMean/matlab_kMeans.htm)

[46] AI4R :: Artificial Intelligence for Ruby (http://ai4r.org/index.html)

[47] reddavis/K-Means · GitHub (http://github.com/reddavis/K-Means/tree/master)

[48] K-means clustering and vector quantization (scipy.cluster.vq) — SciPy v0.11 Reference Guide (DRAFT) (http://docs.scipy.org/doc/scipy/reference/cluster.vq.html)

[49] http://dist.codehaus.org/x10/applications/samples/KMeansDist.x10

[50] http://www.cs.princeton.edu/~wdong/kmeans/

[51] http://code.google.com/p/figue/FIGUE

[52] http://web.science.mq.edu.au/~jydelort/figue/demo.html

[53] Clustering - K-means demo (http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html)

[54] siebn.de - YAK-Means (http://siebn.de/other/yakmeans/)

[55] k-Means and Voronoi Tesselation: Built with Processing | Information & Visualization (http://informationandvisualization.de/blog/kmeans-and-voronoi-tesselation-built-processing)

[56] Hyper-threaded Java - JavaWorld (http://www.javaworld.com/javaworld/jw-11-2006/jw-1121-thread.html)

[57] Color clustering (http://www.leet.it/home/lale/clustering/)

[58] Clustergram: visualization and diagnostics for cluster analysis (R code) | R-statistics blog (http://www.r-statistics.com/2010/06/clustergram-visualization-and-diagnostics-for-cluster-analysis-r-code/)

Article Sources and Contributors

***k*-means clustering** *Source*: http://en.wikipedia.org/w/index.php?oldid=531016579 *Contributors*: 0sm0sm0, 2A01:E35:2423:A030:2677:3FF:FE23:2958, 3mta3, Agor153, Alai, Amkilpatrick, Andkaha, Annabel, Ashwin, BenFrantzDale, BlueScreenD, CBM, Charibdis, Charles Matthews, Chire, Chrike, ChrisDing, Chrisahn, Cincoutprabu, Corvus cornix, Cronholm144, David Eppstein, Den fjättrade ankan, Denshade, Duncharris, Erniepan, FedeLebron, Fnielsen, Foma84, Foobarhoge, Gazpacho, Gfxguy, Giftlite, Golddan Gin, Greenleaf, Gringer, Hakkinen, Headbomb, Helwr, Hgkamath, Homncruse, Honkkis, Illuminated, Ixfd64, Jcallega, Jim1138, Jnothman, John of Reading, JohnBlackburne, Jonsafari, Jsanchezalmeida, June8th, Killerandy, Kzafer, Leishi, Lessbread, LilHelpa, MEmreCelebi, Mahlon, Manyu aditya, Mark Arsten, MarkPundurs, Materialscientist, Mathiasl26, Mati22081979, Mauls, Maxlittle2007, Mcld, Melcombe, Memming, Michael Hardy, MindAfterMath, Miserlou, NedLevine, Nick Number, Ntvuok, Ostrouchov, PerryTachett, Phillipe Israel, Phoolimin, Pot, Qwertyus, Qwfp, Railwaycat, Ranumao, Ratiocinate, Rcalhoun, Rich Farmbrough, Ricky81682, Robert K S, SamuelRiv, Sanchom, SciCompTeacher, Simeon87, Smartcat, Soren.harward, SpuriousQ, Stimpak, Sundirac, Talgalili, Tbmurphy, Toninowiki, Turketwh, UkPaolo, Utacsecd, Weston.pace, Wonderful597, Woolleynick, WorldsApart, Zanetu, 199 anonymous edits

Image Sources, Licenses and Contributors

Image:K Means Example Step 1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:K_Means_Example_Step_1.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Weston.pace

Image:K Means Example Step 2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:K_Means_Example_Step_2.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Weston.pace

Image:K Means Example Step 3.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:K_Means_Example_Step_3.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Weston.pace

Image:K Means Example Step 4.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:K_Means_Example_Step_4.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Weston.pace

File:K-means convergence to a local minimum.png *Source*: http://en.wikipedia.org/w/index.php?title=File:K-means_convergence_to_a_local_minimum.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Agor153

File:Iris Flowers Clustering kMeans.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Iris_Flowers_Clustering_kMeans.svg *License*: Public Domain *Contributors*: Chire

File:ClusterAnalysis Mouse.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:ClusterAnalysis_Mouse.svg *License*: Public Domain *Contributors*: Chire

License