WIKIPEDIA

k-means clustering

***k*-means clustering** is a method of vector quantization, originally from signal processing, that aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. *k*-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms 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 *k-means* and *Gaussian mixture modeling*. 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.

The unsupervised k-means algorithm has a loose relationship to the *k*-nearest neighbor classifier, a popular supervised machine learning technique for classification that is often confused with *k*-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by *k*-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

Contents

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 ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \left\| \mathbf{x} - \mathbf{y} \right\|^2$$

The equivalence can be deduced from identity $\sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_i - \mathbf{y})$. Because the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in *different* clusters (between-cluster sum of squares, BCSS),[1] which follows from the law of total variance.

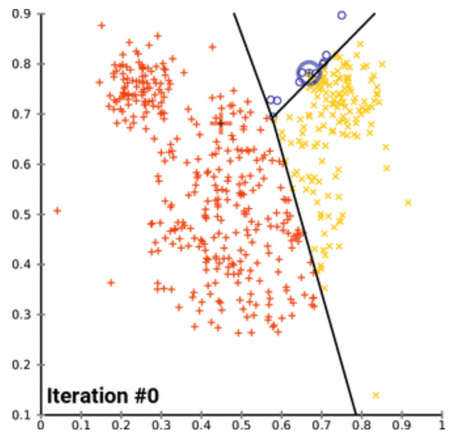
History

The term "k-means" was first used by James MacQueen in 1967,[2] though the idea goes back to Hugo Steinhaus in 1956.[3] The standard algorithm was first proposed by Stuart Lloyd of Bell Labs in 1957 as a technique for pulse-code modulation, although it was not published as a journal article until 1982.[4] In 1965, Edward W. Forgy published essentially the same method, which is why it is sometimes referred to as the Lloyd–Forgy algorithm.[5]

Algorithms

Standard algorithm (naive k-means)

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. It is sometimes also referred to as "naïve *k*-means", because there exist much faster alternatives.[6]



Convergence of *k*-means

Given an initial set of *k* means $m_1^{(1)},\ldots,m_k^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:[7]

Assignment step: Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance.[8] (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means.)

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

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

Update step: Recalculate means (centroids) for observations assigned to each cluster.

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

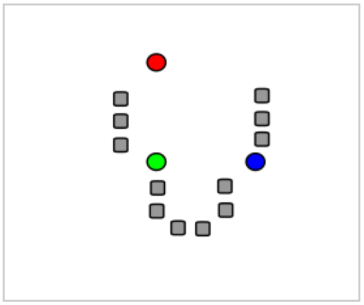
The algorithm has converged when the assignments no longer change. The algorithm is not guaranteed to find the optimum.[9]

The algorithm is often presented as assigning objects to the nearest cluster by distance. Using a different distance function other than (squared) Euclidean distance may prevent the algorithm from converging. Various modifications of *k*-means such as spherical *k*-means and *k*-medoids have been proposed to allow using other distance measures.

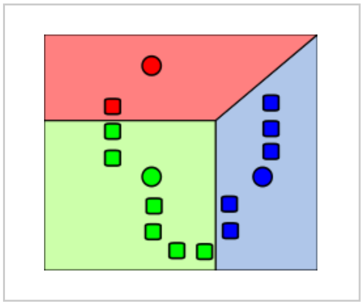
Initialization methods

Commonly used initialization methods are Forgy and Random Partition.[10] The Forgy method randomly chooses *k* observations from the dataset 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.,[10] 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. A comprehensive study by Celebi et al.,[11] however, found that popular initialization methods such as Forgy, Random Partition, and Maximin often perform poorly, whereas Bradley and Fayyad's approach[12] performs "consistently" in "the best group" and *k*-means++ performs "generally well".

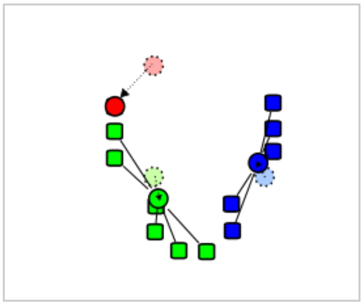
Demonstration of the standard algorithm



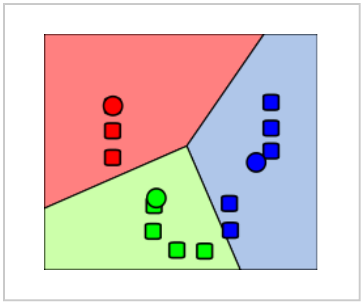
1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).



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.

The algorithm does not guarantee convergence to the global optimum. The result may depend on the initial clusters. As the algorithm is usually fast, it is common to run it multiple times with different starting conditions. However, worst-case performance can be slow: in particular certain point sets, even in two dimensions, converge in exponential time, that is $2^{\Omega(n)}$.[13] 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.[14]

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

Complexity

Finding the optimal solution to the *k*-means clustering problem for observations in *d* dimensions is:

- NP-hard in general Euclidean space (of *d* dimensions) even for two clusters,[15][16][17][18]
- NP-hard for a general number of clusters *k* even in the plane,[19]
- if *k* and *d* (the dimension) are fixed, the problem can be exactly solved in time $O(n^{dk+1})$, where *n* is the number of entities to be clustered.[20]

Thus, a variety of heuristic algorithms such as Lloyd's algorithm given above are generally used.

The running time of Lloyd's algorithm (and most variants) is $O(nkdi)$,[9][21] where:

- *n* is the number of *d*-dimensional vectors (to be clustered)
- *k* the number of clusters
- *i* the number of iterations needed until convergence.

On data that does have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Lloyd's algorithm is therefore often considered to be of "linear" complexity in practice, although it is in the worst case superpolynomial when performed until convergence.[22]

- In the worst-case, Lloyd's algorithm needs $i = 2^{\Omega(\sqrt{n})}$ iterations, so that the worst-case complexity of Lloyd's algorithm is superpolynomial.[22]
- Lloyd's *k*-means algorithm has polynomial smoothed running time. It is shown that[14] 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 proven for simple cases. For example, it is shown that the running time of *k*-means algorithm is bounded by $O(dn^4M^2)$ for n points in an integer lattice $\{1,\dots,M\}^d$.[23]

Lloyd's algorithm is the standard approach for this problem. However, it spends a lot of processing time computing the distances between each of the k cluster centers and the n data points. Since points usually stay in the same clusters after a few iterations, much of this work is unnecessary, making the naïve implementation very inefficient. Some implementations use caching and the triangle inequality in order to create bounds and accelerate Lloyd's algorithm.[9][24][25][26][27]

Variations

- Jenks natural breaks optimization: *k*-means applied to univariate data
- *k*-medians clustering uses the median in each dimension instead of the mean, and this way minimizes L_1 norm (Taxicab geometry).
- *k*-medoids (also: Partitioning Around Medoids, PAM) uses the medoid instead of the mean, and this way minimizes the sum of distances for *arbitrary* distance functions.
- 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.
- *k*-means++ chooses initial centers in a way that gives a provable upper bound on the WCSS objective.
- The filtering algorithm uses kd-trees to speed up each *k*-means step.[28]

- Some methods attempt to speed up each *k*-means step using the triangle inequality.[24][25][26][29][27]
- Escape local optima by swapping points between clusters.[9]
- The Spherical *k*-means clustering algorithm is suitable for textual data.[30]
- Hierarchical variants such as Bisecting *k*-means,[31] X-means clustering[32] and G-means clustering[33] repeatedly split clusters to build a hierarchy, and can also try to automatically determine the optimal number of clusters in a dataset.
- Internal cluster evaluation measures such as cluster silhouette can be helpful at determining the number of clusters.
- Minkowski weighted *k*-means automatically calculates cluster specific feature weights, supporting the intuitive idea that a feature may have different degrees of relevance at different features.[34] These weights can also be used to re-scale a given data set, increasing the likelihood of a cluster validity index to be optimized at the expected number of clusters.[35]
- Mini-batch *k*-means: *k*-means variation using "mini batch" samples for data sets that do not fit into memory.[36]

Hartigan–Wong method

Hartigan and Wong's method[9] provides a variation of *k*-means algorithm which progresses towards a local minimum of the minimum sum-of-squares problem with different solution updates. The method is a local search that iteratively attempts to relocate a sample into a different cluster as long as this process improves the objective function. When no sample can be relocated into a different cluster with an improvement of the objective, the method stops (in a local minimum). In a similar way as the classical *k*-means, the approach remains a heuristic since it does not necessarily guarantee that the final solution is globally optimum.

Let $\varphi(S_j)$ be the individual cost of S_j defined by $\sum_{x\in S_j}(x-\mu_j)^2$, with μ_j the center of the cluster.

Assignment step: Hartigan and Wong's method starts by partitioning the points into random clusters $\{S_j\}_{j\in\{1,\cdots k\}}$.

Update step: Next it determines the $n, m \in \{1,\ldots,k\}$ and $x \in S_n$ for which the following function reaches a maximum

$$\Delta(m,n,x)=\varphi(S_n)+\varphi(S_m)-\varphi(S_n\setminus\{x\})-\varphi(S_m\cup\{x\}).$$

For the x, n, m that reach this maximum, x moves from the cluster S_n to the cluster S_m.

Termination: The algorithm terminates once $\Delta(m,n,x)$ is less than zero for all x, n, m.

Different move acceptance strategies can be used. In a *first-improvement* strategy, any improving relocation can be applied, whereas in a *best-improvement* strategy, all possible relocations are iteratively tested and only the best is applied at each iteration. The former approach favors speed, whether the latter approach generally favors solution quality at the expense of additional computational time. The function Δ used to calculate the result of a relocation can also be efficiently evaluated by using equality[37]

$$\Delta(x,n,m) = \frac{|S_n|}{|S_n|-1} \cdot \|\mu_n - x\|^2 - \frac{|S_m|}{|S_m|+1} \cdot \|\mu_m - x\|^2.$$

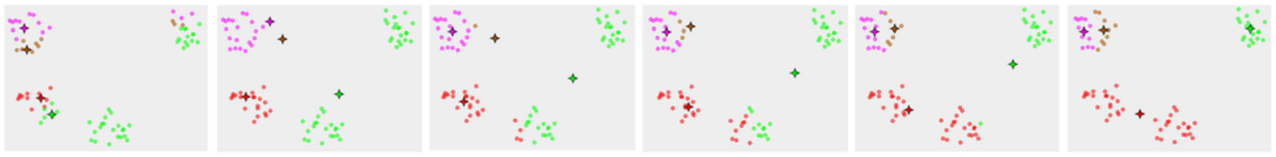
Global optimization and metaheuristics

The classical k-means algorithm and its variations are known to only converge to local minima of the minimum-sum-of-squares clustering problem defined as

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

Many studies have attempted to improve the convergence behavior of the algorithm and maximize the chances of attaining the global optimum (or at least, local minima of better quality). Initialization and restart techniques discussed in the previous sections are one alternative to find better solutions. More recently, mathematical programming algorithms based on branch-and-bound and column generation have produced ‘‘provenly optimal’’ solutions for datasets with up to 2,300 entities.[38] As expected, due to the NP-hardness of the subjacent optimization problem, the computational time of optimal algorithms for K-means quickly increases beyond this size. Optimal solutions for small- and medium-scale still remain valuable as a benchmark tool, to evaluate the quality of other heuristics. To find high-quality local minima within a controlled computational time but without optimality guarantees, other works have explored metaheuristics and other global optimization techniques, e.g., based on incremental approaches and convex optimization,[39] random swaps[40] (i.e., iterated local search), variable neighborhood search[41]and genetic algorithms.[42][43] It is indeed known that finding better local minima of the minimum sum-of-squares clustering problem can make the difference between failure and success to recover cluster structures in feature spaces of high dimension.[43]

Discussion

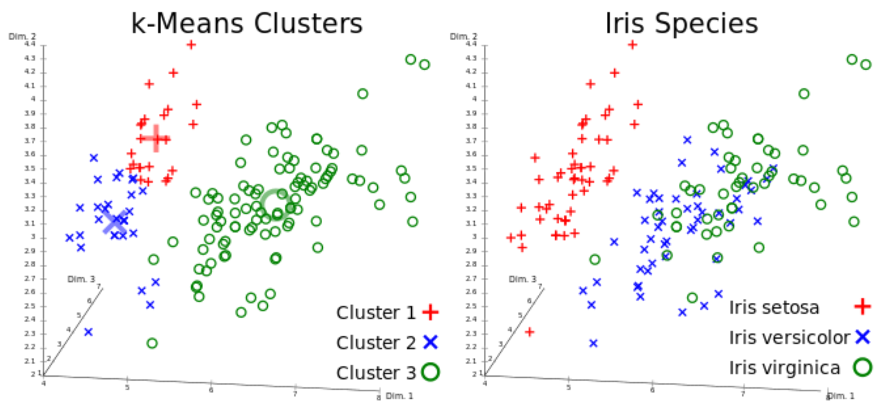


A typical example of the *k*-means convergence to a local minimum. In this example, the result of *k*-means clustering (the right figure) contradicts the obvious cluster structure of the data set. The small circles are the data points, the four ray stars are the centroids (means). The initial configuration 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.[44]

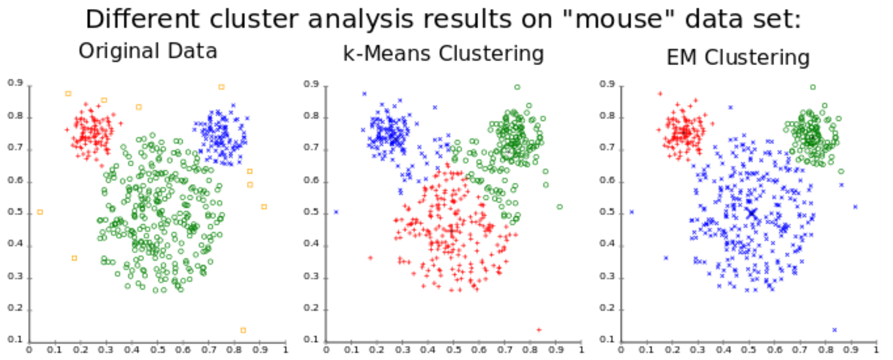
Three key features of *k*-means that make it efficient are often regarded as its biggest drawbacks:

- 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.).

A key limitation of k-means is its cluster model. The concept is based on spherical clusters that are separable so that the mean 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's containing 3 *classes*. As with any other clustering algorithm, the k-means result makes assumptions that the data satisfy certain criteria. It works well on some data sets, and fails on others.



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.



k-means clustering vs. EM clustering on an artificial dataset ("mouse"). The tendency of *k*-means to produce equal-sized clusters leads to bad results here, while EM benefits from the Gaussian distributions with different radius present in the data set.

The result of k-means can 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 (arguably a generalization of k-means) are more flexible 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). In counterpart, EM requires the optimization of a larger number of free parameters and poses some methodological issues due to vanishing clusters or badly-conditioned covariance matrices. K-means is closely related to nonparametric Bayesian modeling.[45]

Applications

k-means clustering is rather easy to apply to even large data sets, particularly when using heuristics such as Lloyd's algorithm. It has been successfully used in market segmentation, computer vision, and astronomy among many other domains. It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

Vector quantization

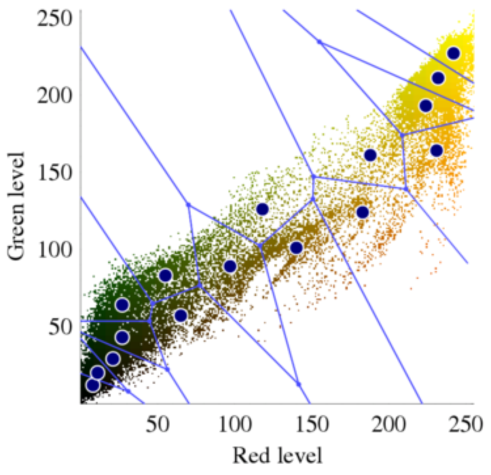
k-means originates from signal processing, and still finds use in this domain. For example, in computer graphics, color quantization is the task of reducing the color palette of an image to a fixed number of colors *k*. The *k*-means algorithm can easily be used for this task and produces competitive results. A use case for this approach is image segmentation. Other uses of vector quantization include non-random sampling, as *k*-means can easily be used to choose *k* different but prototypical objects from a large data set for further analysis.

Two-channel (for illustration purposes -- red and green channels only) color image.

Cluster analysis

In cluster analysis, the *k*-means algorithm can be used to partition the input data set into *k* partitions (clusters).

However, the pure *k*-means algorithm is not very flexible, and as such is of limited use (except for when vector quantization as above is actually the desired use case). In particular, the parameter *k* is known to be hard to choose (as discussed above) when not given by external constraints. Another limitation is that it cannot be used with arbitrary distance functions or on non-numerical data. For these use cases, many other algorithms are superior.



Vector quantization of colors present in the image above into Voronoi cells using *k*-means.

Feature learning

k-means clustering has been used as a feature learning (or dictionary learning) step, in either (semi-)supervised learning or unsupervised learning.[46] The basic approach is first to train a *k*-means clustering representation, using the input training data (which need not be labelled). Then, to project any input datum into the new feature space, an "encoding" function, such as the thresholded matrix-product of the datum with the centroid locations, computes the distance from the datum to each centroid, or simply an indicator function for the nearest centroid,[46][47] or some smooth transformation of the distance.[48] Alternatively, transforming the sample-cluster distance through a Gaussian RBF, obtains the hidden layer of a radial basis function network.[49]

This use of *k*-means has been successfully combined with simple, linear classifiers for semi-supervised learning in NLP (specifically for named entity recognition)[50] and in computer vision. On an object recognition task, it was found to exhibit comparable performance with more sophisticated feature learning approaches such as autoencoders and restricted Boltzmann machines.[48] However, it generally requires more data, for equivalent performance, because each data point only contributes to one "feature".[46]

Relation to other algorithms

Gaussian mixture model

The slow "standard algorithm" for *k*-means clustering, and its associated expectation-maximization algorithm, is a special case of a Gaussian mixture model, specifically, the limiting case when fixing all covariances to be diagonal, equal and have infinitesimal small variance.[51]:850 Instead of small variances, a hard cluster assignment can also be used to show another equivalence of *k*-means clustering to a special case of "hard" Gaussian mixture modelling.[52](11.4.2.5) This does not mean that it is efficient to use Gaussian mixture modelling to compute *k*-means, but just that there is a theoretical relationship, and that Gaussian mixture modelling can be interpreted as a generalization of *k*-means; on the contrary, it has been suggested to use k-means clustering to find starting points for Gaussian mixture modelling on difficult data.[51]:849

K-SVD

Another generalization of the *k*-means algorithm is the K-SVD algorithm, which estimates data points as a sparse linear combination of "codebook vectors". *k*-means corresponds to the special case of using a single codebook vector, with a weight of 1.[53]

Principal component analysis

The relaxed solution of *k*-means clustering, specified by the cluster indicators, is given by principal component analysis (PCA).[54][55] The intuition is that *k*-means describe spherically shaped (ball-like) clusters. If the data has 2 clusters, the line connecting the two centroids is the best 1-dimensional projection direction, which is also the first PCA direction. Cutting the line at the center of mass separates the clusters (this is the continuous relaxation of the discrete cluster indicator). If the data have three clusters, the 2-dimensional plane spanned by three cluster centroids is the best 2-D projection. This plane is also defined by the first two PCA dimensions. Well-separated clusters are effectively modelled by ball-shaped clusters and thus discovered by *k*-means. Non-ball-shaped clusters are hard to separate when they are close. For example, two half-moon shaped clusters intertwined in space do not separate well when projected onto PCA subspace. *k*-means should not be expected to do well on this data.[56] It is straightforward to produce counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.[57]

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.[58] One of the advantages of mean shift over *k*-means is that the number of clusters is not pre-specified, because mean shift is likely to find only a few clusters if 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.

Independent component analysis

Under sparsity assumptions and when input data is pre-processed with the whitening transformation, *k*-means produces the solution to the linear independent component analysis (ICA) task. This aids in explaining the successful application of *k*-means to feature learning.[59]

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.[58] 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 implementations

Different implementations of the algorithm exhibit performance differences, with the fastest on a test data set finishing in 10 seconds, the slowest taking 25,988 seconds (~7 hours).[1] The differences can be attributed to implementation quality, language and compiler differences, different termination criteria and precision levels, and the use of indexes for acceleration.

Free Software/Open Source

The following implementations are available under Free/Open Source Software licenses, with publicly available source code.

- Accord.NET contains C# implementations for *k*-means, *k*-means++ and *k*-modes.
- ALGLIB contains parallelized C++ and C# implementations for *k*-means and *k*-means++.
- AOSP contains a Java implementation for *k*-means.
- 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.
- Smile contains *k*-means and various more other algorithms and results visualization (for java, kotlin and scala).
- Julia contains a *k*-means implementation in the JuliaStats Clustering package.
- KNIME contains nodes for *k*-means and *k*-medoids.
- Mahout contains a MapReduce based *k*-means.
- mlpack contains a C++ implementation of *k*-means.
- Octave contains *k*-means.
- OpenCV contains a *k*-means implementation.
- Orange includes a component for *k*-means clustering with automatic selection of *k* and cluster silhouette scoring.

- PSPP contains *k*-means, The QUICK CLUSTER command performs *k*-means clustering on the dataset.
- R contains three *k*-means variations.
- SciPy and scikit-learn contain multiple *k*-means implementations.
- Spark MLlib implements a distributed *k*-means algorithm.
- Torch contains an *unsup* package that provides *k*-means clustering.
- Weka contains *k*-means and *x*-means.

Proprietary

The following implementations are available under proprietary license terms, and may not have publicly available source code.

- Ayasdi
- Mathematica
- MATLAB
- OriginPro
- RapidMiner
- SAP HANA
- SAS
- SPSS
- Stata

See also

- BFR algorithm
- Centroidal Voronoi tessellation
- Head/tail Breaks
- k q-flats
- K-means++
- Linde–Buzo–Gray algorithm
- Self-organizing map

References

1. Kriegel, Hans-Peter; Schubert, Erich; Zimek, Arthur (2016). "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?". *Knowledge and Information Systems*. **52** (2): 341–378. doi:10.1007/s10115-016-1004-2 (https://doi.org/10.1007%2Fs10115-016-1004-2). ISSN 0219-1377 (https://www.worldcat.org/issn/0219-1377). S2CID 40772241 (https://api.semanticscholar.org/CorpusID:40772241).
2. MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations* (http://projecteuclid.org/euclid.bsmsp/1200512992). Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. **1**. University of California Press. pp. 281–297. MR 0214227 (https://www.ams.org/mathscinet-getitem?mr=0214227). Zbl 0214.46201 (https://zbmath.org/?format=complete&q=an:0214.46201). Retrieved 2009-04-07.
3. Steinhaus, Hugo (1957). "Sur la division des corps matériels en parties". *Bull. Acad. Polon. Sci.* (in French). **4** (12): 801–804. MR 0090073 (https://www.ams.org/mathscinet-getitem?mr=0090073). Zbl 0079.16403 (https://zbmath.org/?format=complete&q=an:0079.16403).
4. Lloyd, Stuart P. (1957). "Least square quantization in PCM". *Bell Telephone Laboratories Paper*. Published in journal much later: Lloyd, Stuart P. (1982). "Least squares quantization in PCM" (http://www.cs.toronto.edu/~roweis/csc2515-2006/readings/lloyd57.pdf) (PDF). *IEEE Transactions on Information Theory*. **28** (2): 129–137. CiteSeerX 10.1.1.131.1338 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.131.1338). doi:10.1109/TIT.1982.1056489 (https://doi.org/10.1109%2FTIT.1982.1056489). Retrieved 2009-04-15.

5. Forgy, Edward W. (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". *Biometrics*. **21** (3): 768–769. JSTOR 2528559 (https://www.jstor.org/stable/2528559).
6. Pelleg, Dan; Moore, Andrew (1999). "Accelerating exact k -means algorithms with geometric reasoning" (http://portal.acm.org/citation.cfm?doid=312129.312248). *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '99*. San Diego, California, United States: ACM Press: 277–281. doi:10.1145/312129.312248 (https://doi.org/10.1145%2F312129.312248). ISBN 9781581131437. S2CID 13907420 (https://api.semanticscholar.org/CorpusID:13907420).
7. MacKay, David (2003). "Chapter 20. An Example Inference Task: Clustering" (http://www.inference.phy.cam.ac.uk/mackay/itprnn/ps/284.292.pdf) (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 978-0-521-64298-9. MR 2012999 (https://www.ams.org/mathscinet-getitem?mr=2012999).
8. Since the square root is a monotone function, this also is the minimum Euclidean distance assignment.
9. Hartigan, J. A.; Wong, M. A. (1979). "Algorithm AS 136: A *k*-Means Clustering Algorithm". *Journal of the Royal Statistical Society, Series C*. **28** (1): 100–108. JSTOR 2346830 (https://www.jstor.org/stable/2346830).
10. Hamerly, Greg; Elkan, Charles (2002). "Alternatives to the *k*-means algorithm that find better clusterings" (http://people.csail.mit.edu/tieu/notebook/kmeans/15_p600-hamerly.pdf) (PDF). *Proceedings of the eleventh international conference on Information and knowledge management (CIKM)*.
11. Celebi, M. E.; Kingravi, H. A.; Vela, P. A. (2013). "A comparative study of efficient initialization methods for the *k*-means clustering algorithm". *Expert Systems with Applications*. **40** (1): 200–210. arXiv:1209.1960 (https://arxiv.org/abs/1209.1960). doi:10.1016/j.eswa.2012.07.021 (https://doi.org/10.1016%2Fj.eswa.2012.07.021). S2CID 6954668 (https://api.semanticscholar.org/CorpusID:6954668).
12. Bradley, Paul S.; Fayyad, Usama M. (1998). "Refining Initial Points for *k*-Means Clustering". *Proceedings of the Fifteenth International Conference on Machine Learning*.
13. Vattani, A. (2011). "k-means requires exponentially many iterations even in the plane" (http://cseweb.ucsd.edu/users/avattani/papers/kmeans-journal.pdf) (PDF). *Discrete and Computational Geometry*. **45** (4): 596–616. doi:10.1007/s00454-011-9340-1 (https://doi.org/10.1007%2Fs00454-011-9340-1). S2CID 42683406 (https://api.semanticscholar.org/CorpusID:42683406).
14. Arthur, David; Manthey, B.; Roeglin, H. (2009). "k-means has polynomial smoothed complexity". *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*. arXiv:0904.1113 (https://arxiv.org/abs/0904.1113).
15. Garey, M.; Johnson, D.; Witsenhausen, H. (1982-03-01). "The complexity of the generalized Lloyd - Max problem (Corresp.)". *IEEE Transactions on Information Theory*. **28** (2): 255–256. doi:10.1109/TIT.1982.1056488 (https://doi.org/10.1109%2FTIT.1982.1056488). ISSN 0018-9448 (https://www.worldcat.org/issn/0018-9448).
16. Kleinberg, Jon; Papadimitriou, Christos; Raghavan, Prabhakar (1998-12-01). "A Microeconomic View of Data Mining". *Data Mining and Knowledge Discovery*. **2** (4): 311–324. doi:10.1023/A:1009726428407 (https://doi.org/10.1023%2FA%3A1009726428407). ISSN 1384-5810 (https://www.worldcat.org/issn/1384-5810). S2CID 15252504 (https://api.semanticscholar.org/CorpusID:15252504).
17. Aloise, D.; Deshpande, A.; Hansen, P.; Popat, P. (2009). "NP-hardness of Euclidean sum-of-squares clustering" (https://doi.org/10.1007%2Fs10994-009-5103-0). *Machine Learning*. **75** (2): 245–249. doi:10.1007/s10994-009-5103-0 (https://doi.org/10.1007%2Fs10994-009-5103-0).
18. Dasgupta, S.; Freund, Y. (July 2009). "Random Projection Trees for Vector Quantization". *IEEE Transactions on Information Theory*. **55** (7): 3229–3242. arXiv:0805.1390 (https://arxiv.org/abs/0805.1390). doi:10.1109/TIT.2009.2021326 (https://doi.org/10.1109%2FTIT.2009.2021326). S2CID 666114 (https://api.semanticscholar.org/CorpusID:666114).

19. Mahajan, Meena; Nimbhorkar, Prajakta; Varadarajan, Kasturi (2009). *The Planar* k*-Means Problem is NP-Hard*. Lecture Notes in Computer Science. **5431**. pp. 274–285. CiteSeerX 10.1.1.331.1306 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.331.1306). doi:10.1007/978-3-642-00202-1_24 (https://doi.org/10.1007%2F978-3-642-00202-1_24). ISBN 978-3-642-00201-4.
20. 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 (https://doi.org/10.1145%2F177424.178042).
21. Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich (2008). *Introduction to information retrieval*. New York: Cambridge University Press. ISBN 978-0521865715. OCLC 190786122 (https://www.worldcat.org/oclc/190786122).
22. Arthur, David; Vassilvitskii, Sergei (2006-01-01). *How Slow is the* k*-means Method?*. *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*. SCG '06. New York, NY, USA: ACM. pp. 144–153. doi:10.1145/1137856.1137880 (https://doi.org/10.1145%2F1137856.1137880). ISBN 978-1595933409. S2CID 3084311 (https://api.semanticscholar.org/CorpusID:3084311).
23. Bhowmick, Abhishek (2009). "A theoretical analysis of Lloyd's algorithm for *k*-means clustering" (https://web.archive.org/web/20151208140946/https://gautam5.cse.iitk.ac.in/opencs/sites/default/files/final.pdf) (PDF). Archived from the original (https://gautam5.cse.iitk.ac.in/opencs/sites/default/files/final.pdf) (PDF) on 2015-12-08. See also here (https://www.researchgate.net/publication/267854906_A_theoretical_analysis_of_Lloyd's_algorithm_for_k-means_clustering).
24. Phillips, Steven J. (2002-01-04). "Acceleration of K-Means and Related Clustering Algorithms". In Mount, David M.; Stein, Clifford (eds.). *Acceleration of* k*-Means and Related Clustering Algorithms*. Lecture Notes in Computer Science. **2409**. Springer Berlin Heidelberg. pp. 166–177. doi:10.1007/3-540-45643-0_13 (https://doi.org/10.1007%2F3-540-45643-0_13). ISBN 978-3-540-43977-6.
25. Elkan, Charles (2003). "Using the triangle inequality to accelerate *k*-means" (http://www-cse.ucsd.edu/~elkan/kmeansicml03.pdf) (PDF). *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*.
26. Hamerly, Greg. "Making *k*-means even faster". CiteSeerX 10.1.1.187.3017 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.187.3017).
27. Hamerly, Greg; Drake, Jonathan (2015). *Accelerating Lloyd's algorithm for* k*-means clustering*. *Partitional Clustering Algorithms*. pp. 41–78. doi:10.1007/978-3-319-09259-1_2 (https://doi.org/10.1007%2F978-3-319-09259-1_2). ISBN 978-3-319-09258-4.
28. Kanungo, Tapas; Mount, David M.; Netanyahu, Nathan S.; Piatko, Christine D.; Silverman, Ruth; Wu, Angela Y. (2002). "An efficient *k*-means clustering algorithm: Analysis and implementation" (http://www.cs.umd.edu/~mount/Papers/pami02.pdf) (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **24** (7): 881–892. doi:10.1109/TPAMI.2002.1017616 (https://doi.org/10.1109%2FTPAMI.2002.1017616). Retrieved 2009-04-24.
29. Drake, Jonathan (2012). "Accelerated *k*-means with adaptive distance bounds" (http://opt.kyb.tuebingen.mpg.de/papers/opt2012_paper_13.pdf) (PDF). *The 5th NIPS Workshop on Optimization for Machine Learning, OPT2012*.
30. Dhillon, I. S.; Modha, D. M. (2001). "Concept decompositions for large sparse text data using clustering" (https://doi.org/10.1023%2Fa%3A1007612920971). *Machine Learning*. **42** (1): 143–175. doi:10.1023/a:1007612920971 (https://doi.org/10.1023%2Fa%3A1007612920971).
31. Steinbach, M.; Karypis, G.; Kumar, V. (2000). " "A comparison of document clustering techniques". In". *KDD Workshop on Text Mining*. **400** (1): 525–526.
32. Pelleg, D.; & Moore, A. W. (2000, June). "X-means: Extending *k*-means with Efficient Estimation of the Number of Clusters (http://cs.uef.fi/~zhao/Courses/Clustering2012/Xmeans.pdf)". In *ICML*, Vol. 1
33. Hamerly, Greg; Elkan, Charles (2004). "Learning the k in k-means" (http://papers.nips.cc/paper/2526-learning-the-k-in-k-means.pdf) (PDF). *Advances in Neural Information Processing Systems*. **16**: 281.
34. Amorim, R. C.; Mirkin, B. (2012). "Minkowski Metric, Feature Weighting and Anomalous Cluster Initialisation in *k*-Means Clustering". *Pattern Recognition*. **45** (3): 1061–1075. doi:10.1016/j.patcog.2011.08.012 (https://doi.org/10.1016%2Fj.patcog.2011.08.012).

35. Amorim, R. C.; Hennig, C. (2015). "Recovering the number of clusters in data sets with noise features using feature rescaling factors". *Information Sciences*. **324**: 126–145. arXiv:1602.06989 (https://arxiv.org/abs/1602.06989). doi:10.1016/j.ins.2015.06.039 (https://doi.org/10.1016%2Fj.ins.2015.06.039). S2CID 315803 (https://api.semanticscholar.org/CorpusID:315803).
36. Sculley, David (2010). "Web-scale *k*-means clustering" (http://dl.acm.org/citation.cfm?id=1772862). *Proceedings of the 19th international conference on World Wide Web*. ACM. pp. 1177–1178. Retrieved 2016-12-21.
37. Telgarsky, Matus. "Hartigan's Method: *k*-means Clustering without Voronoi" (http://proceedings.mlr.press/v9/telgarsky10a/telgarsky10a.pdf) (PDF).
38. Aloise, Daniel; Hansen, Pierre; Liberti, Leo (2012). "An improved column generation algorithm for minimum sum-of-squares clustering". *Mathematical Programming*. **131** (1–2): 195–220. doi:10.1007/s10107-010-0349-7 (https://doi.org/10.1007%2Fs10107-010-0349-7). S2CID 17550257 (https://api.semanticscholar.org/CorpusID:17550257).
39. Bagirov, A. M.; Taheri, S.; Ugon, J. (2016). "Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems". *Pattern Recognition*. **53**: 12–24. doi:10.1016/j.patcog.2015.11.011 (https://doi.org/10.1016%2Fj.patcog.2015.11.011).
40. Fränti, Pasi (2018). "Efficiency of random swap clustering" (https://doi.org/10.1186%2Fs40537-018-0122-y). *Journal of Big Data*. **5** (1): 1–21. doi:10.1186/s40537-018-0122-y (https://doi.org/10.1186%2Fs40537-018-0122-y).
41. Hansen, P.; Mladenovic, N. (2001). "J-Means: A new local search heuristic for minimum sum of squares clustering". *Pattern Recognition*. **34** (2): 405–413. doi:10.1016/S0031-3203(99)00216-2 (https://doi.org/10.1016%2FS0031-3203%2899%2900216-2).
42. Krishna, K.; Murty, M. N. (1999). "Genetic k-means algorithm" (https://www.researchgate.net/publication/5600582). *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. **29** (3): 433–439. doi:10.1109/3477.764879 (https://doi.org/10.1109%2F3477.764879). PMID 18252317 (https://pubmed.ncbi.nlm.nih.gov/18252317).
43. Gribel, Daniel; Vidal, Thibaut (2019). "HG-means: A scalable hybrid metaheuristic for minimum sum-of-squares clustering". *Pattern Recognition*. **88**: 569–583. arXiv:1804.09813 (https://arxiv.org/abs/1804.09813). doi:10.1016/j.patcog.2018.12.022 (https://doi.org/10.1016%2Fj.patcog.2018.12.022). S2CID 13746584 (https://api.semanticscholar.org/CorpusID:13746584).
44. Mirkes, E. M. "K-means and *k*-medoids applet" (http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html). Retrieved 2 January 2016.
45. Kulis, Brian; Jordan, Michael I. (2012-06-26). *Revisiting* k*-means: new algorithms via Bayesian nonparametrics* (https://icml.cc/2012/papers/291.pdf) (PDF). *ICML*. pp. 1131–1138. ISBN 9781450312851.
46. Coates, Adam; Ng, Andrew Y. (2012). "Learning feature representations with *k*-means" (https://cs.stanford.edu/~acoates/papers/coatesng_nntot2012.pdf) (PDF). In Montavon, G.; Orr, G. B.; Müller, K.-R. (eds.). *Neural Networks: Tricks of the Trade*. Springer.
47. Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cédric (2004). *Visual categorization with bags of keypoints* (https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/csurka-eccv-04.pdf) (PDF). ECCV Workshop on Statistical Learning in Computer Vision.
48. Coates, Adam; Lee, Honglak; Ng, Andrew Y. (2011). *An analysis of single-layer networks in unsupervised feature learning* (https://web.archive.org/web/20130510120705/http://www.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf) (PDF). International Conference on Artificial Intelligence and Statistics (AISTATS). Archived from the original (http://www.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf) (PDF) on 2013-05-10.
49. Schwenker, Friedhelm; Kestler, Hans A.; Palm, Günther (2001). "Three learning phases for radial-basis-function networks". *Neural Networks*. **14** (4–5): 439–458. CiteSeerX 10.1.1.109.312 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.312). doi:10.1016/s0893-6080(01)00027-2 (https://doi.org/10.1016%2Fs0893-6080%2801%2900027-2). PMID 11411631 (https://pubmed.ncbi.nlm.nih.gov/11411631).

50. Lin, Dekang; Wu, Xiaoyun (2009). *Phrase clustering for discriminative learning* (http://www.aclweb.org/anthology/P/P09/P09-1116.pdf) (PDF). Annual Meeting of the ACL and IJCNLP. pp. 1030–1038.
51. Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (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 (NY): Cambridge University Press. ISBN 978-0-521-88068-8.
52. Kevin P. Murphy (2012). *Machine learning : a probabilistic perspective*. Cambridge, Mass.: MIT Press. ISBN 978-0-262-30524-2. OCLC 810414751 (https://www.worldcat.org/oclc/810414751).
53. Aharon, Michal; Elad, Michael; Bruckstein, Alfred (2006). "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation" (http://www.cs.technion.ac.il/FREDDY/papers/120.pdf) (PDF). *IEEE Transactions on Signal Processing*. **54** (11): 4311. Bibcode:2006ITSP...54.4311A (https://ui.adsabs.harvard.edu/abs/2006ITSP...54.4311A). doi:10.1109/TSP.2006.881199 (https://doi.org/10.1109%2FTSP.2006.881199). S2CID 7477309 (https://api.semanticscholar.org/CorpusID:7477309).
54. Zha, Hongyuan; Ding, Chris; Gu, Ming; He, Xiaofeng; Simon, Horst D. (December 2001). "Spectral Relaxation for *k*-means Clustering" (http://ranger.uta.edu/~chqding/papers/Zha-Kmeans.pdf) (PDF). *Neural Information Processing Systems Vol.14 (NIPS 2001)*: 1057–1064.
55. Ding, Chris; He, Xiaofeng (July 2004). "K-means Clustering via Principal Component Analysis" (http://ranger.uta.edu/~chqding/papers/KmeansPCA1.pdf) (PDF). *Proceedings of International Conference on Machine Learning (ICML 2004)*: 225–232.
56. Drineas, Petros; Frieze, Alan M.; Kannan, Ravi; Vempala, Santosh; Vinay, Vishwanathan (2004). "Clustering large graphs via the singular value decomposition" (http://www.cc.gatech.edu/~vempala/papers/dfkvv.pdf) (PDF). *Machine Learning*. **56** (1–3): 9–33. doi:10.1023/b:mach.0000033113.59016.96 (https://doi.org/10.1023%2Fb%3Amach.0000033113.59016.96). S2CID 5892850 (https://api.semanticscholar.org/CorpusID:5892850). Retrieved 2012-08-02.
57. Cohen, Michael B.; Elder, Sam; Musco, Cameron; Musco, Christopher; Persu, Madalina (2014). "Dimensionality reduction for *k*-means clustering and low rank approximation (Appendix B)". arXiv:1410.6801 (https://arxiv.org/abs/1410.6801) [cs.DS (https://arxiv.org/archive/cs.DS)].
58. Little, Max A.; Jones, Nick S. (2011). "Generalized Methods and Solvers for Piecewise Constant Signals: Part I" (http://www.maxlittle.net/publications/pwc_filtering_arxiv.pdf) (PDF). *Proceedings of the Royal Society A*. **467** (2135): 3088–3114. Bibcode:2011RSPSA.467.3088L (https://ui.adsabs.harvard.edu/abs/2011RSPSA.467.3088L). doi:10.1098/rspa.2010.0671 (https://doi.org/10.1098%2Frspa.2010.0671). PMC 3191861 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3191861). PMID 22003312 (https://pubmed.ncbi.nlm.nih.gov/22003312).
59. Vinnikov, Alon; Shalev-Shwartz, Shai (2014). "K-means Recovers ICA Filters when Independent Components are Sparse" (http://www.cs.huji.ac.il/~shais/papers/KmeansICA_ICML2014.pdf) (PDF). *Proceedings of the International Conference on Machine Learning (ICML 2014)*.