



WIKIPEDIA  
The Free Encyclopedia

Main page  
Contents  
Featured content  
Current events  
Random article  
Donate to Wikipedia  
Wikipedia store

Interaction

Help  
About Wikipedia  
Community portal  
Recent changes  
Contact page

Tools

What links here  
Related changes  
Upload file  
Special pages  
Permanent link  
Page information  
Wikidata item  
Cite this page

Print/export

Create a book  
Download as PDF  
Printable version

Languages

Català  
Deutsch  
Español  
Français  
한국어  
Bahasa Indonesia  
Italiano  
עברית  
日本語  
Norsk bokmål  
Polski  
Português  
Русский  
Српски / srpski  
עברית  
Українська  
اردو  
中文

Edit links

Create account Log in

Article **Talk**

Read **Edit** View history

Search

# *k*-means clustering

From Wikipedia, the free encyclopedia

***k*-means clustering** is a method of **vector quantization**, originally from signal processing, that is popular for **cluster analysis** in **data mining**. *k*-means clustering aims to **partition** *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest **mean**, serving as a **prototype** of the cluster. This results in a partitioning of the data space into **Voronoi cells**.

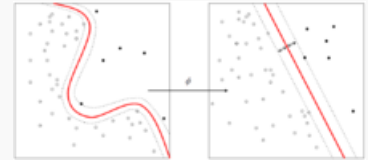
The problem is computationally difficult (**NP-hard**); however, there are efficient **heuristic algorithms** that are commonly employed and converge quickly to a **local optimum**. These are usually similar to the **expectation-maximization algorithm** for **mixtures of Gaussian distributions** via 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.

The algorithm has nothing to do with and should not be confused with *k*-nearest neighbor, another popular **machine learning** technique.

## Contents [hide]

- Description
- History
- Algorithms
  - Standard algorithm
    - Initialization methods
  - Complexity
  - Variations
- Discussion
- Applications
  - Vector quantization
  - Cluster analysis
  - Feature learning
- Relation to other statistical machine learning algorithms
  - Mean shift clustering
  - Principal component analysis (PCA)
  - Independent component analysis (ICA)
  - Bilateral filtering
- Similar problems
- Software implementations
  - Free
  - Commercial
- See also
- References

## Machine learning and data mining



### Problems

Classification · Clustering · Regression · Anomaly detection · Association rules · Reinforcement learning · Structured prediction · Feature learning · Online learning · Semi-supervised learning · Unsupervised learning · Learning to rank · Grammar induction

### Supervised learning

(classification · regression)

Decision trees · Ensembles (Bagging, Boosting, Random forest) · *k*-NN · Linear regression · Naive Bayes · Neural networks · Logistic regression · Perceptron · Support vector machine (SVM) · Relevance vector machine (RVM)

### Clustering

BIRCH · Hierarchical · ***k*-means** · Expectation-maximization (EM) · DBSCAN · OPTICS · Mean-shift

### Dimensionality reduction

Factor analysis · CCA · ICA · LDA · NMF · PCA · t-SNE

### Structured prediction

Graphical models (Bayes net, CRF, HMM)

### Anomaly detection

*k*-NN · Local outlier factor

### Neural nets

Autoencoder · Deep learning · Multilayer perceptron · RNN · Restricted Boltzmann machine · SOM · Convolutional neural network

### Theory

Bias-variance dilemma · Computational learning theory · Empirical risk minimization · PAC learning · Statistical learning · VC theory



**Machine learning portal**



**Computer science portal**



**Statistics portal**

v · t · e

## Description [edit]

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

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

where  $\mu_j$  is the mean of points in  $S_j$ .

## History [\[edit\]](#)

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

## Algorithms [\[edit\]](#)

### Standard algorithm [\[edit\]](#)

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  $m_1^{(1)}, \dots, m_k^{(1)}$  (see below), the algorithm proceeds by alternating between two steps:<sup>[7]</sup>

**Assignment step:** Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared [Euclidean distance](#), this is intuitively the "nearest" mean.<sup>[8]</sup> (Mathematically, this means partitioning the observations according to the [Voronoi diagram](#) generated by the means).

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

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

**Update step:** Calculate the new means to be the [centroids](#) of the observations in the new clusters.

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

Since the arithmetic mean is a [least-squares estimator](#), this also minimizes the within-cluster sum of squares (WCSS) objective.

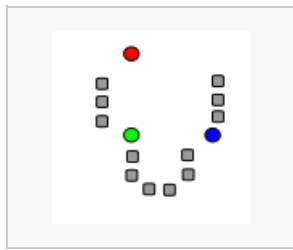
The algorithm has converged when the assignments no longer change. Since both steps optimize the WCSS objective, and there only exists a finite number of such partitionings, the algorithm must converge to a (local) optimum. There is no guarantee that the global optimum is found using this algorithm.

The algorithm is often presented as assigning objects to the nearest cluster by distance. The standard algorithm aims at minimizing the WCSS objective, and thus assigns by "least sum of squares", which is exactly equivalent to assigning by the smallest Euclidean distance. Using a different distance function other than (squared) Euclidean distance may stop the algorithm from converging.<sup>[citation needed]</sup> Various modifications of *k*-means such as spherical *k*-means and [k-medoids](#) have been proposed to allow using other distance measures.

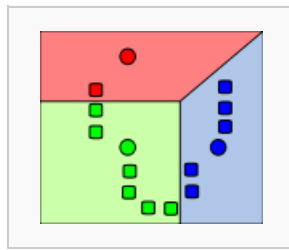
### Initialization methods [\[edit\]](#)

Commonly used initialization methods are Forgy and Random Partition.<sup>[9]</sup> 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.,<sup>[9]</sup> 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.

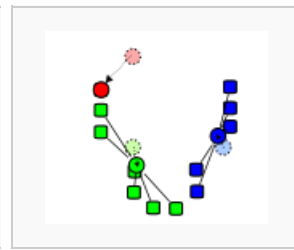
### Demonstration of the standard algorithm



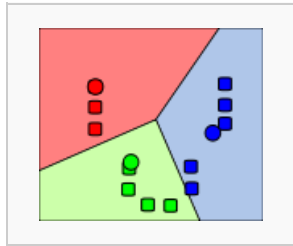
1.  $k$  initial "means" (in this case  $k=3$ ) are randomly generated within the data domain (shown in color).



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.<sup>[10]</sup> 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.<sup>[11]</sup>

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 [\[edit\]](#)

Regarding computational complexity, finding the optimal solution to the  $k$ -means clustering problem for observations in  $d$  dimensions is:

- **NP-hard** in general Euclidean space  $d$  even for 2 clusters<sup>[12][13]</sup>
- **NP-hard** for a general number of clusters  $k$  even in the plane<sup>[14]</sup>
- If  $k$  and  $d$  (the dimension) 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<sup>[15]</sup>

Thus, a variety of [heuristic algorithms](#) such as Lloyd's algorithm given above are generally used.

The running time of Lloyd's algorithm is often given as  $O(nkdi)$ , where  $n$  is the number of  $d$ -dimensional vectors,  $k$  the number of clusters and  $i$  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.

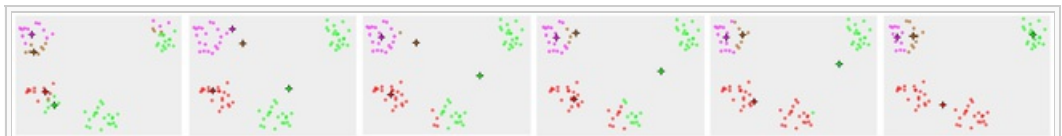
Following are some recent insights into this algorithm complexity behaviour.

- Lloyd's  $k$ -means algorithm has polynomial smoothed running time. It is shown that<sup>[11]</sup> for arbitrary set of  $n$  points in  $[0, 1]^d$ , if each point is independently perturbed by a normal distribution with mean  $\mathbf{0}$  and variance  $\sigma^2$ , then the expected running time of  $k$ -means algorithm is bounded by  $O(n^{34} k^{34} d^{18} \log^4(n) / \sigma^6)$ , which is a polynomial in  $n$ ,  $k$ ,  $d$  and  $1/\sigma$ .
- Better bounds are proved for simple cases. For example,<sup>[16]</sup> showed that the running time of  $k$ -means algorithm is bounded by  $O(dn^4 M^2)$  for  $n$  points in an [integer lattice](#)  $\{1, \dots, M\}^d$ .

### Variations [\[edit\]](#)

- **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.<sup>[17]</sup>
- Some methods attempt to speed up each  $k$ -means step using **coresets**<sup>[18]</sup> or the **triangle inequality**.<sup>[19]</sup>
- Escape local optima by swapping points between clusters.<sup>[6]</sup>
- The **Spherical k-means** clustering algorithm is suitable for directional data.<sup>[20]</sup>
- The **Minkowski metric weighted k-means** deals with irrelevant features by assigning cluster specific weights to each feature<sup>[21]</sup>

## Discussion [\[edit\]](#)



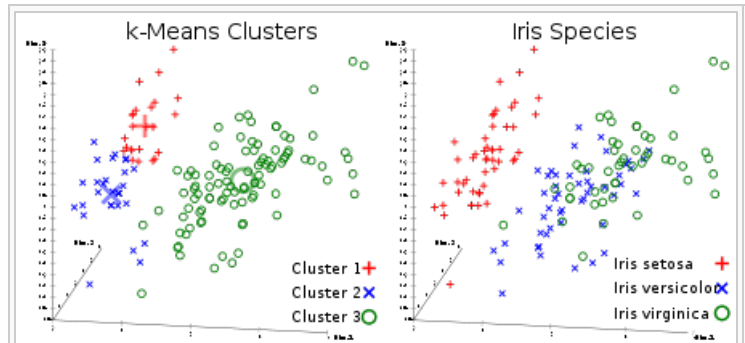
A typical example of the  $k$ -means convergence to a local minimum. In this example, the result of  $k$ -means clustering (the right figure) contradicts the obvious cluster structure of the data set. The small circles are the data points, the four raystars are the centroids (means). The initial configuration 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.<sup>[22]</sup>

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

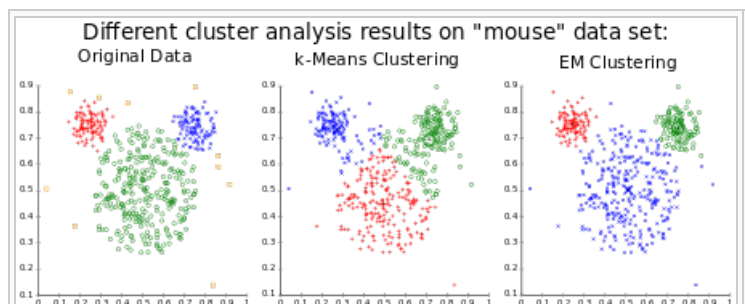
- **Euclidean distance** is used as a **metric** and **variance** is used as a measure of cluster scatter.
- The number of clusters  $k$  is an input parameter: an inappropriate choice of  $k$  may yield poor results. That is why, when performing  $k$ -means, it is important to run diagnostic checks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

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



$k$ -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 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

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.

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 [\[edit\]](#)

$k$ -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, including [market segmentation](#), [computer vision](#), [geostatistics](#),<sup>[23]</sup> [astronomy](#) and [agriculture](#). It often is used as a preprocessing step for other algorithms, for example to find a starting configuration.

## Vector quantization [\[edit\]](#)

*Main article: [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. 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.

## Cluster analysis [\[edit\]](#)

*Main article: [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 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 of the algorithm is that it cannot be used with arbitrary distance functions or on non-numerical data. For these use cases, many other algorithms have been developed since.

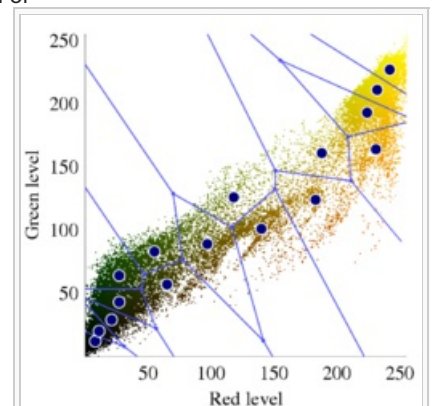
## Feature learning [\[edit\]](#)

$k$ -means clustering has been used as a [feature learning](#) (or [dictionary learning](#)) step, in either (semi-)supervised learning or [unsupervised learning](#).<sup>[24]</sup> 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, we have a choice of "encoding" functions, but we can use for example the thresholded matrix-product of the datum with the centroid locations, the distance from the datum to each centroid, or simply an indicator function for the nearest centroid,<sup>[24][25]</sup> or some smooth transformation of the distance.<sup>[26]</sup> Alternatively, by transforming the sample-cluster distance through a [Gaussian RBF](#), one effectively obtains the hidden layer of a [radial basis function network](#).<sup>[27]</sup>

This use of  $k$ -means has been successfully combined with simple, [linear classifiers](#) for semi-supervised learning in [NLP](#) (specifically for [named entity recognition](#))<sup>[28]</sup> 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](#).<sup>[26]</sup> However, it generally requires more data than the sophisticated methods, for equivalent performance, because each data point only contributes to one "feature" rather than multiple.<sup>[24]</sup>



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



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



## Relation to other statistical machine learning algorithms [\[edit\]](#)

$k$ -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.<sup>[29]</sup> 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.<sup>[30]</sup>

### Mean shift clustering [\[edit\]](#)

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.<sup>[31]</sup> 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) [\[edit\]](#)

It was asserted in<sup>[32][33]</sup> 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,<sup>[34]</sup>), and it is straightforward to uncover counterexamples to the statement that the cluster centroid subspace is spanned by the principal directions.<sup>[35]</sup>

### Independent component analysis (ICA) [\[edit\]](#)

It has been shown in <sup>[36]</sup> that 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](#) task. This aids in explaining the successful application of  $k$ -means to [feature learning](#).

### Bilateral filtering [\[edit\]](#)

$k$ -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.<sup>[31]</sup> 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 [\[edit\]](#)

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 [\[edit\]](#)

### Free [\[edit\]](#)

- [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.
- [Julia](#) contains a  $k$ -means implementation in the Clustering package.<sup>[37]</sup>
- [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.
- [R](#) contains three  $k$ -means variations.<sup>[1][3][6]</sup>

- [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.

## Commercial [\[edit\]](#)


























- [Grapheme](#)
- [MATLAB](#)
- [Mathematica](#)
- [SAS](#)
- [Stata](#)
- [Teradata Aster](#)

## See also [\[edit\]](#)

- [Canopy clustering algorithm](#)
- [Centroidal Voronoi tessellation](#)
- [k q-flats](#)
- [Linde–Buzo–Gray algorithm](#)
- [Nearest centroid classifier](#)
- [Self-organizing map](#)
- [Silhouette clustering](#)
- [Head/tail Breaks](#)

## References [\[edit\]](#)

- <sup>^</sup> <sup>a</sup> <sup>b</sup> MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations* . Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability **1**. University of California Press. pp. 281–297. [MR 0214227](#) . [Zbl 0214.46201](#) . Retrieved 2009-04-07.
- <sup>^</sup> Steinhaus, H. (1957). "Sur la division des corps matériels en parties". *Bull. Acad. Polon. Sci.* (in French) **4** (12): 801–804. [MR 0090073](#) . [Zbl 0079.16403](#) .
- <sup>^</sup> <sup>a</sup> <sup>b</sup> 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"  (PDF). *IEEE Transactions on Information Theory* **28** (2): 129–137. doi:10.1109/TIT.1982.1056489 . Retrieved 2009-04-15.
- <sup>^</sup> E.W. Forgy (1965). "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". *Biometrics* **21**: 768–769. [JSTOR 2528559](#) .
- <sup>^</sup> J.A. Hartigan (1975). *Clustering algorithms*. John Wiley & Sons, Inc.
- <sup>^</sup> <sup>a</sup> <sup>b</sup> <sup>c</sup> 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](#) .
- <sup>^</sup> MacKay, David (2003). "Chapter 20. An Example Inference Task: Clustering"  (PDF). *Information Theory, Inference and Learning Algorithms* . Cambridge University Press. pp. 284–292. ISBN 0-521-64298-1. [MR 2012999](#) .
- <sup>^</sup> Since the square root is a monotone function, this also is the minimum Euclidean distance assignment.
- <sup>^</sup> <sup>a</sup> <sup>b</sup> Hamerly, G. and Elkan, C. (2002). "Alternatives to the k-means algorithm that find better clusterings"  (PDF). *Proceedings of the eleventh international conference on Information and knowledge management (CIKM)*.
- <sup>^</sup> Vattani, A. (2011). "k-means requires exponentially many iterations even in the plane"  (PDF). *Discrete and Computational Geometry* **45** (4): 596–616. doi:10.1007/s00454-011-9340-1 .
- <sup>^</sup> <sup>a</sup> <sup>b</sup> Arthur, D.; Manthey, B.; Roeglin, H. (2009). "k-means has polynomial smoothed complexity". *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*.
- <sup>^</sup> 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 .
- <sup>^</sup> Dasgupta, S. and Freund, Y. (July 2009). "Random Projection Trees for Vector Quantization". *Information Theory, IEEE Transactions on* **55**: 3229–3242. [arXv:0805.1390](#) . doi:10.1109/TIT.2009.2021326 .
- <sup>^</sup> 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 .
- <sup>^</sup> 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 .
- <sup>^</sup> Arthur, Abhishek Bhowmick (2009). *A theoretical analysis of Lloyd's algorithm for k-means clustering* (Thesis).<sup>[1]</sup> <sup>[dead link]</sup>

17. <sup>^</sup> 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"  (PDF). *IEEE Trans. Pattern Analysis and Machine Intelligence* **24**: 881–892. doi:10.1109/TPAMI.2002.1017616 . Retrieved 2009-04-24.
18. <sup>^</sup> Frahling, G.; Sohler, C. (2006). *A fast k-means implementation using coresets*  (PDF). *Proceedings of the twenty-second annual symposium on Computational geometry (SoCG)*.
19. <sup>^</sup> Elkan, C. (2003). "Using the triangle inequality to accelerate k-means"  (PDF). *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*.
20. <sup>^</sup> Dhillon, I. S.; Modha, D. M. (2001). "Concept decompositions for large sparse text data using clustering". *Machine Learning* **42** (1): 143–175. doi:10.1023/a:1007612920971 .
21. <sup>^</sup> 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 .
22. <sup>^</sup> Mirkes, E.M. "K-means and K-medoids applet."  <http://www.math.le.ac.uk> . Retrieved 1 May 2015.
23. <sup>^</sup> Honarkhah, M; Caers, J (2010). "Stochastic Simulation of Patterns Using Distance-Based Pattern Modeling". *Mathematical Geosciences* **42**: 487–517. doi:10.1007/s11004-010-9276-7 .
24. <sup>^</sup> <sup>a</sup> <sup>b</sup> <sup>c</sup> Coates, Adam; Ng, Andrew Y. (2012). "Learning feature representations with k-means"  (PDF). In G. Montavon, G. B. Orr, K.-R. Müller. *Neural Networks: Tricks of the Trade*. Springer.
25. <sup>^</sup> Csurka, Gabriella; Dance, Christopher C.; Fan, Lixin; Willamowski, Jutta; Bray, Cédric (2004). *Visual categorization with bags of keypoints*  (PDF). ECCV Workshop on Statistical Learning in Computer Vision.
26. <sup>^</sup> <sup>a</sup> <sup>b</sup> Coates, Adam; Lee, Honglak; Ng, Andrew Y. (2011). *An analysis of single-layer networks in unsupervised feature learning*  (PDF). International Conference on Artificial Intelligence and Statistics (AISTATS).
27. <sup>^</sup> Schwenker, Friedhelm; Kestler, Hans A.; Palm, Günther (2001). "Three learning phases for radial-basis-function networks". *Neural Networks* **14**: 439–458. doi:10.1016/s0893-6080(01)00027-2 . CiteSeerX 10.1.1.109.312 .
28. <sup>^</sup> Lin, Dekang; Wu, Xiaoyun (2009). *Phrase clustering for discriminative learning*  (PDF). Annual Meeting of the ACL and IJCNLP. pp. 1030–1038.
29. <sup>^</sup> Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 16.1. Gaussian Mixture Models and k-Means Clustering" . *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.
30. <sup>^</sup> Aharon, Michal; Elad, Michael; Bruckstein, Alfred (2006). "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation"  (PDF).
31. <sup>^</sup> <sup>a</sup> <sup>b</sup> Little, M.A.; Jones, N.S. (2011). "Generalized Methods and Solvers for Piecewise Constant Signals: Part I"  (PDF). *Proceedings of the Royal Society A* **467**: 3088–3114. doi:10.1098/rspa.2010.0671 .
32. <sup>^</sup> H. Zha, C. Ding, M. Gu, X He and H.D. Simon (Dec 2001). "Spectral Relaxation for K-means Clustering"  (PDF). *Neural Information Processing Systems vol. 14 (NIPS 2001)* (Vancouver, Canada): 1057–1064.
33. <sup>^</sup> Chris Ding and Xiaofeng He (July 2004). "K-means Clustering via Principal Component Analysis"  (PDF). *Proc. of Int'l Conf. Machine Learning (ICML 2004)*: 225–232.
34. <sup>^</sup> Drineas, P.; A. Frieze; R. Kannan; S. Vempala; V. Vinay (2004). "Clustering large graphs via the singular value decomposition"  (PDF). *Machine learning* **56**: 9–33. doi:10.1023/b:mach.0000033113.59016.96 . Retrieved 2012-08-02.
35. <sup>^</sup> Cohen, M.; S. Elder; C. Musco; C. Musco; M. Persu (2014). "Dimensionality reduction for k-means clustering and low rank approximation (Appendix B)" . ArXiv. Retrieved 2014-11-29.
36. <sup>^</sup> Alon Vinnikov and Shai Shalev-Shwartz (2014). "K-means Recovers ICA Filters when Independent Components are Sparse"  (PDF). *Proc. of Int'l Conf. Machine Learning (ICML 2014)*.
37. <sup>^</sup> Clustering.jl  [www.github.com](http://www.github.com)

Categories: [Data clustering algorithms](#) | [Statistical algorithms](#)

This page was last modified on 27 August 2015, at 17:16.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

