Purpose

K-means clustering

Syntax

```
IDX = kmeans(X,k)
[IDX,C] = kmeans(X,k)
[IDX,C,sumd] = kmeans(X,k)
[IDX,C,sumd,D] = kmeans(X,k)
[...] = kmeans(...,param1,val1,param2,val2,...)
```

Description

`IDX = kmeans(X,k)` partitions the points in the n-by-p data matrix `X` into `k` clusters. This iterative partitioning minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. Rows of `X` correspond to points, columns correspond to variables. `kmeans` returns an n-by-1 vector `IDX` containing the cluster indices of each point. By default, `kmeans` uses squared Euclidean distances.

`[IDX,C] = kmeans(X,k)` returns the `k` cluster centroid locations in the k-by-p matrix `C`.

`[IDX,C,sumd] = kmeans(X,k)` returns the within-cluster sums of point-to-centroid distances in the 1-by-k vector `sumd`.

`[IDX,C,sumd,D] = kmeans(X,k)` returns distances from each point to every centroid in the n-by-k matrix `D`.

`[...] = kmeans(...,param1,val1,param2,val2,...)` enables you to specify optional parameter/value pairs to control the iterative algorithm used by `kmeans`. Valid parameter strings are listed in the following table.

Parameter	Value
`'distance'`	Distance measure, in p-dimensional space. `kmeans` minimizes with respect to this parameter. `kmeans` computes centroid clusters differently for the different supported distance measures.

Parameter	Value	
	`'sqEuclidean'`	Squared Euclidean distance (default). Each centroid is the mean of the points in that cluster.
	`'cityblock'`	Sum of absolute differences, i.e., the L1 distance. Each centroid is the component-wise median of the points in that cluster.
	`'cosine'`	One minus the cosine of the included angle between points (treated as vectors). Each centroid is the mean of the points in that cluster, after normalizing those points to unit Euclidean length.
	`'correlation'`	One minus the sample correlation between points (treated as sequences of values). Each centroid is the component-wise mean of the points in that cluster, after centering and normalizing those points to zero mean and unit standard deviation.
	`'Hamming'`	Percentage of bits that differ (only suitable for binary data). Each centroid is the component-wise median of points in that cluster.

Parameter	Value	
'emptyaction'	Action to take if a cluster loses all its member observations.	
	'error'	Treat an empty cluster as an error (default).
	'drop'	Remove any clusters that become empty. kmeans sets the corresponding return values in C and D to NaN.
	'singleton'	Create a new cluster consisting of the one point furthest from its centroid.
'onlinephase'	Flag indicating whether kmeans should perform an online update phase in addition to a batch update phase. The online phase can be time consuming for large data sets, but guarantees a solution that is a local minimum of the distance criterion, that is, a partition of the data where moving any single point to a different cluster increases the total sum of distances.	
	'on'	Perform online update (default).
	'off'	Do not perform online update.
'options'	Options for the iterative algorithm used to minimize the fitting criterion, as created by statset.	
'replicates'	Number of times to repeat the clustering, each with a new set of initial cluster centroid positions. kmeans returns the solution with the lowest value for sumd. You can supply 'replicates' implicitly by supplying a 3D array as the value for the 'start' parameter.	

Parameter	Value	
'start'	Method used to choose the initial cluster centroid positions, sometimes known as *seeds*.	
	'sample'	Select k observations from X at random (default).
	'uniform'	Select k points uniformly at random from the range of X. Not valid with Hamming distance.
	'cluster'	Perform a preliminary clustering phase on a random 10% subsample of X. This preliminary phase is itself initialized using 'sample'.
	Matrix	k-by-p matrix of centroid starting locations. In this case, you can pass in [] for k, and kmeans infers k from the first dimension of the matrix. You can also supply a 3-D array, implying a value for the 'replicates' parameter from the array's third dimension.

Algorithm

kmeans uses a two-phase iterative algorithm to minimize the sum of point-to-centroid distances, summed over all k clusters:

1. The first phase uses *batch updates*, where each iteration consists of reassigning points to their nearest cluster centroid, all at once, followed by recalculation of cluster centroids. This phase occasionally does not converge to solution that is a local minimum, that is, a partition of the data where moving any single point to a different cluster increases the total sum of distances. This is more likely for small data sets. The batch phase is fast, but potentially only approximates a solution as a starting point for the second phase.

2 The second phase uses *online updates*, where points are individually reassigned if doing so will reduce the sum of distances, and cluster centroids are recomputed after each reassignment. Each iteration during the second phase consists of one pass though all the points. The second phase will converge to a local minimum, although there may be other local minima with lower total sum of distances. The problem of finding the global minimum can only be solved in general by an exhaustive (or clever, or lucky) choice of starting points, but using several replicates with random starting points typically results in a solution that is a global minimum.

References

[1] Seber, G. A. F. *Multivariate Observations*. Hoboken, NJ: John Wiley & Sons, Inc., 1984.

[2] Spath, H. *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples*. Translated by J. Goldschmidt. New York: Halsted Press, 1985.

Examples

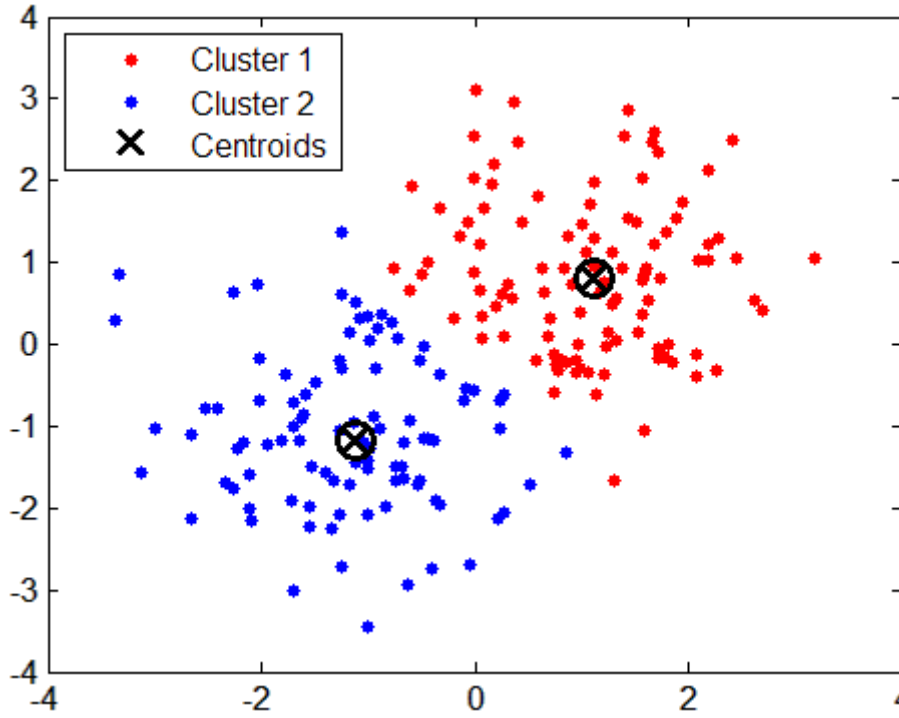
The following creates two clusters from separated random data:

```
X = [randn(100,2)+ones(100,2);...
     randn(100,2)-ones(100,2)];
opts = statset('Display','final');

[idx,ctrs] = kmeans(X,2,...
                    'Distance','city',...
                    'Replicates',5,...
                    'Options',opts);
5 iterations, total sum of distances = 284.671
4 iterations, total sum of distances = 284.671
4 iterations, total sum of distances = 284.671
3 iterations, total sum of distances = 284.671
3 iterations, total sum of distances = 284.671

plot(X(idx==1,1),X(idx==1,2),'r.','MarkerSize',12)
hold on
plot(X(idx==2,1),X(idx==2,2),'b.','MarkerSize',12)
```

```
plot(ctrs(:,1),ctrs(:,2),'kx',...
     'MarkerSize',12,'LineWidth',2)
plot(ctrs(:,1),ctrs(:,2),'ko',...
     'MarkerSize',12,'LineWidth',2)
legend('Cluster 1','Cluster 2','Centroids',...
       'Location','NW')
```



See Also

linkage, clusterdata, silhouette