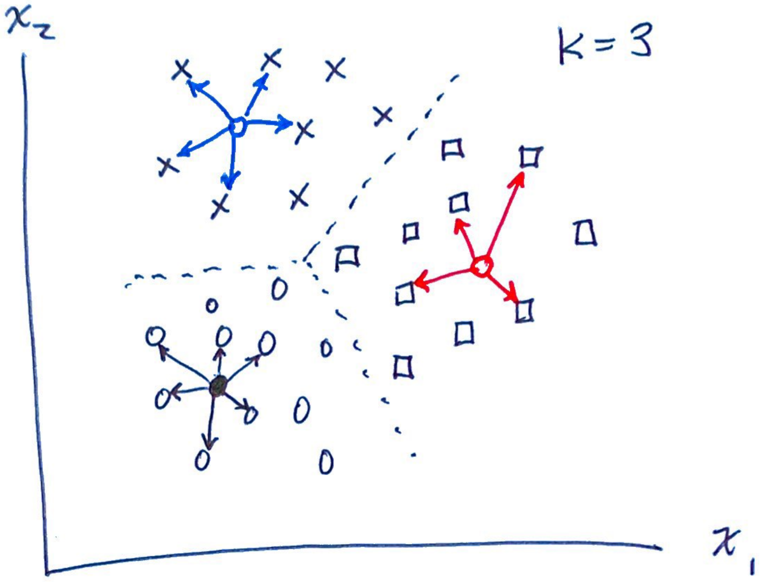
K-means clustering



The kmeans function

>> idx = kmeans(X, k)

where

k	# of clusters
X	data matrix
idx	cluster index column vector

Options

Distance Metric:

By default, the euclidean distance is used to assess the similarity b/t two observations. You can use other metrics, such as correlation.

```
>> g = kmeans(X, 2, "Distance", "correlation")
```

starting Locations of Cluster Centroids:

You can use the "Start" option to specify the starting centroids of the clusters, for example [0 -1] and [6 5]

```
>> g = kmeans(X, 2, "Start", [0 -1; 6 5])
```

Replicates

Another way to optimize clustering is to perform the analysis multiple times w/ different starting positions, and then choose the clustering scheme which minimizes the sum of the distances b/t the centroids and the observations (sumd). This can be done with the "Replicates" option. The following command repeats the clustering five times and returns the clusters with the lowest sumd.

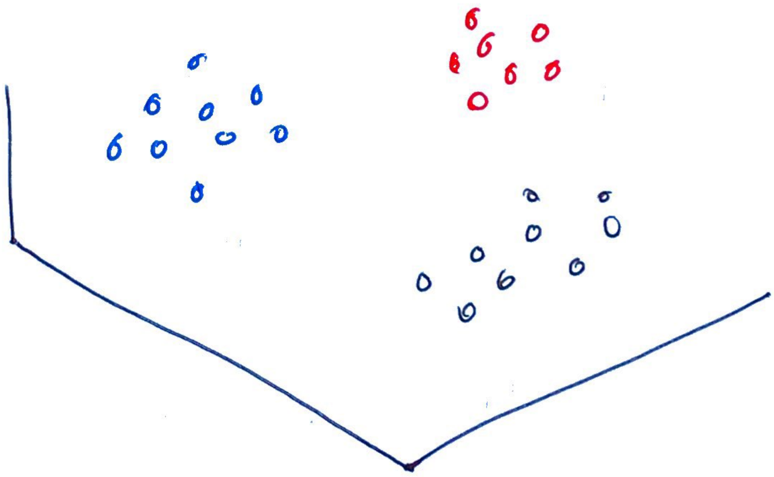
```
>> g = kmeans(X, 2, "Replicates", 5)
>> load data.csv

>> grp = kmeans (data, 3)
```

three groups

```
>> scatter3 (data (:, 1), data (:, 2), data (:, 3),
                 10, grp
```

size of the markers

each group gets a different color according to the vector grp



Basketball Players:

```
>> data = readtable ("bball.txt");
>> data (:, [26 : end]) = [];
>> data. pos = categorical (data. pos);
```

Extract & Normalize columns of interest

```
                             "assists "blocks" "dRebounds"...
>> stats = data {:, [5 6 11 : end ]};
>> stats = table2array (stats)
>> statsNorm = normalize (stats);
```

Use kmeans clustering on statsNorm to group the data into two sets in grp. Set the # of replicates to five.

```
>> grp = kmeans (statsNorm, 3, "Replicates, 5)
```

perform PCA and plot the transformed data by group

```
>> [pcs, scrs] = pca (statsNorm)
>> scatter3 (scrs (:, 1), scrs (:, 2), scrs (:, 3), 10, grp)
>> view (110, 40)
% try "Distance" = " correlation"
```

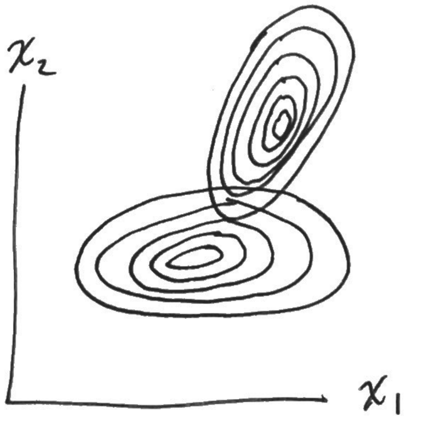
Gaussian Mixture Models (GMM)

Another clustering method is GMM. GMM clustering fits several n-dimensional normal distributions to the data, using those distributions to assign each observation to a cluster.

Step 1 Fit Gaussian Mixture Model

```
>> gm = fitgmdist (X,2)
```

You can use the function fitgmdist to fit several multidimensional gaussian (normal) distributions, e.g. two distributions



Step 2 Identify Clusters

```
>> g = cluster(gm, X)
```

Now the data can be clustered probabilistically, by calculating each observation's posterior probability for each component.

```
>> [g, ~, p] = cluster (gm, X)
```

You can also return the individual probabilities used to determine the clusters. The matrix p has two columns, one for each of the two clusters.

```
>> load data.csv

>> mdl = fitgmdist(data, 3)
% try "CovarianceType" = "diagonal"
>> grp = cluster(mdl, data)


>> scatter3(data(:,1), data(:,2), ...
      data(:,3), 15, grp, "filled")

% to see the individual probabilities
>> [grp, ~, p] = cluster(mdl, data)

>> p
% this model is pretty sure which cluster
  each data point belongs to.
```

Basketball Players

```
                                                  data is
>> data = readtable("bball.txt");               % normalized
                                                  by game
% show data
>> data(1:11, :)

% remove unused data

>> data(:, [26:end]) = [];

% extract columns of interest

stats = data(:, ["assists", "blocks", ...]);

% matrix

stats = table2array(stats);
% normalize to zero mean and standard dev 1
statsNorm = normalize(stats);
                                          # of groups
% use GMM on statsNorm
mdl = fitgmdist(statsNorm, 3, "Replicates", 5 ...

          "RegularizationValue", 0.02)
                             show
% group the data and ~~find~~ the probabilities used
                                to determine the clusters
[grp, ~, gprob] = cluster(mdl, StatsNorm)
% plot the PCA transformed data by group

[pcs, scrs] = pca(statsNorm)

scatter3(scrs(:,1), scrs(:,2), scrs(:,3), 15, grp)
```