Clustering and K-means

Root Mean Square Error (RMS)

Data: $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N \in R^d$

Approximations: $\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_N \in R^d$

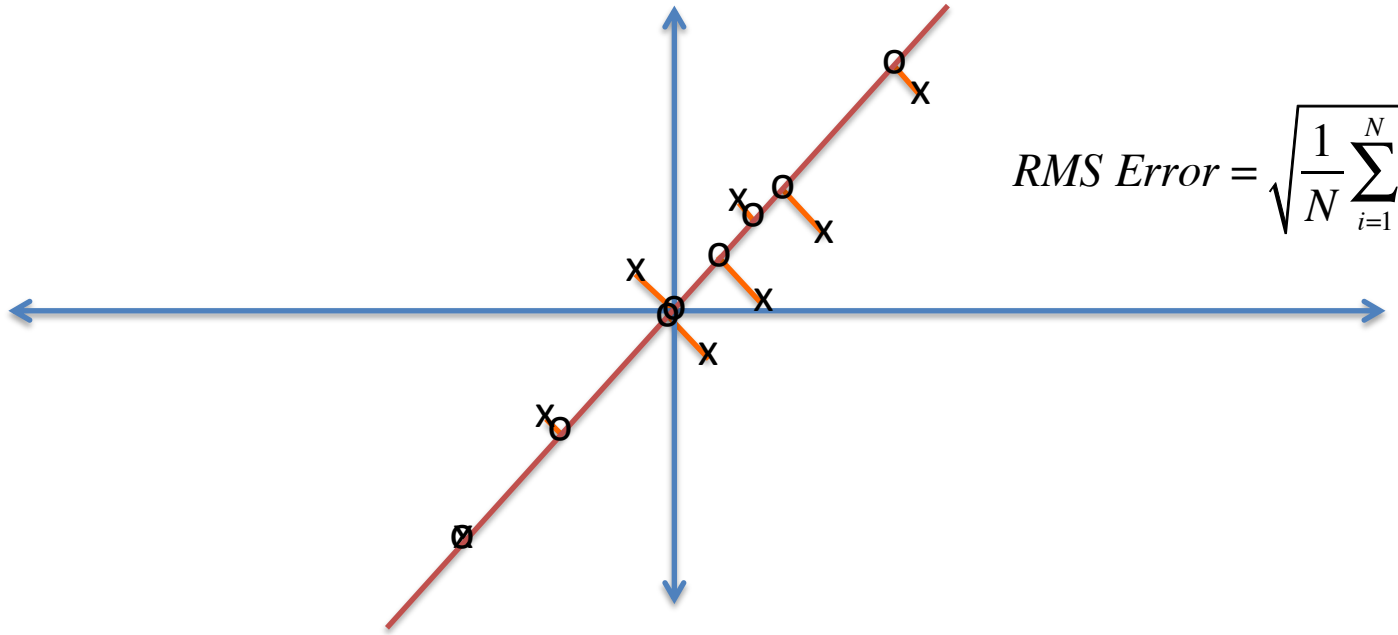
Root Mean Square error = $\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left\| \vec{x}_i - \vec{y}_i \right\|_2^2}$

PCA based prediction

Data: $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N \in R^d$

Mean vector: $\vec{\mu}$ Top k eigenvectors: $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k$

Approximation of $\vec{x}_j$: $\vec{o}_j = \vec{\mu} + \sum_{i=1}^{k} \left(\vec{v}_i \cdot \vec{x}_j \right) \vec{v}_i$



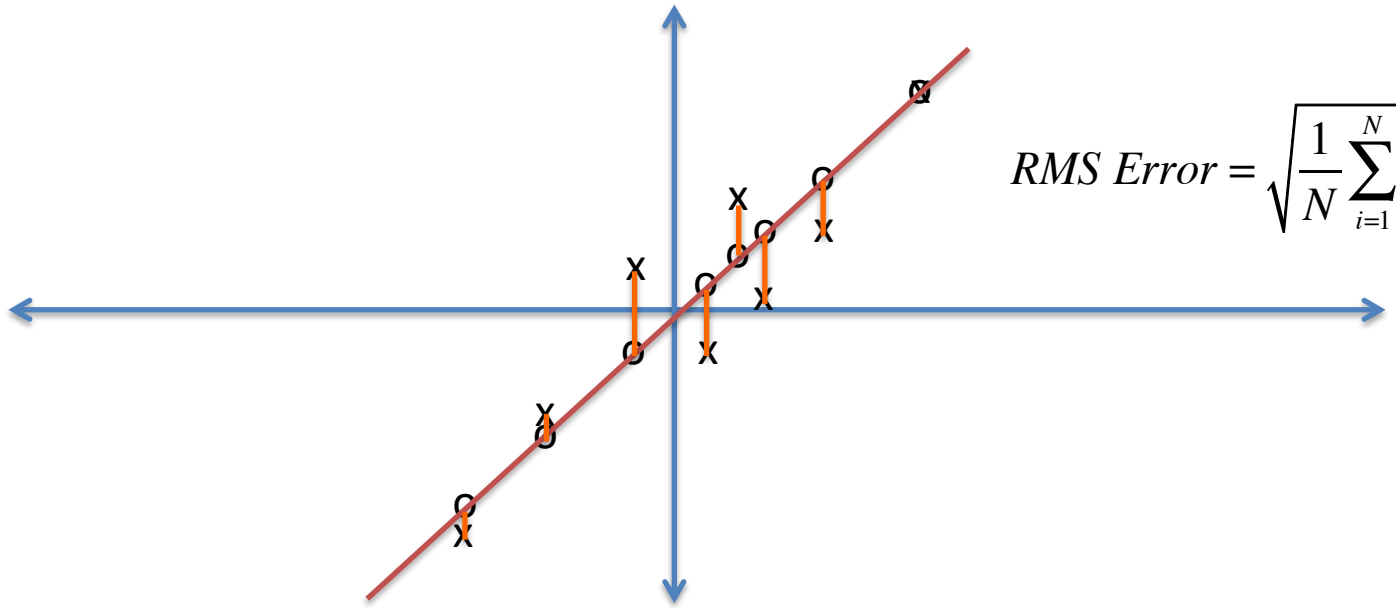
$$RMS\ Error = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left\| \vec{x}_i - \vec{o}_i \right\|_2^2}$$

Regression based Prediction

Data: $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_N, y_N) \in R^d$

Input: $\vec{x} \in R^d$ Output: $y \in R$

Approximation of y given $\vec{x}$: $\hat{y} = a_0 + \sum_{i=1}^{d} a_i x_i$



$$RMS\ Error = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2}$$

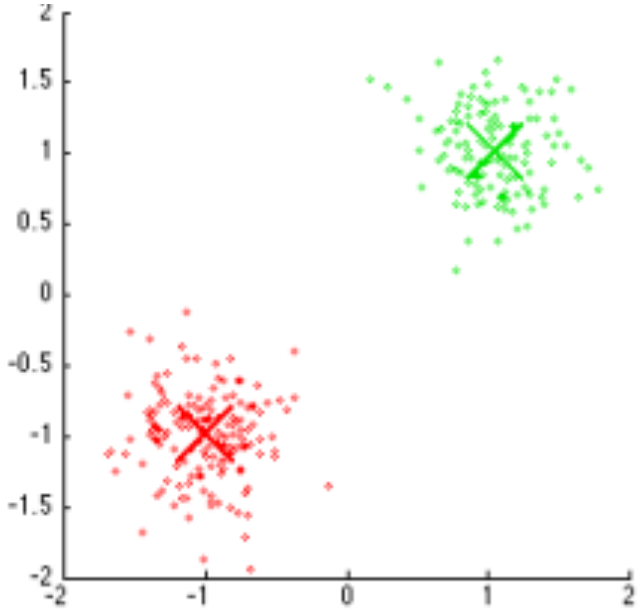
K-means clustering

Data: $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N \in R^d$

Model: k representatives: $\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_k \in R^d$

Approximation of $\vec{x}_j$: $\vec{o}_j = \text{argmin}_{\vec{r}_i} \left\| \vec{x}_j - \vec{r}_i \right\|_2^2$

= the representative closest to $\vec{x}_j$



$$RMS\ Error = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left\| \vec{x}_i - \vec{o}_i \right\|_2^2}$$

K-means Algorithm

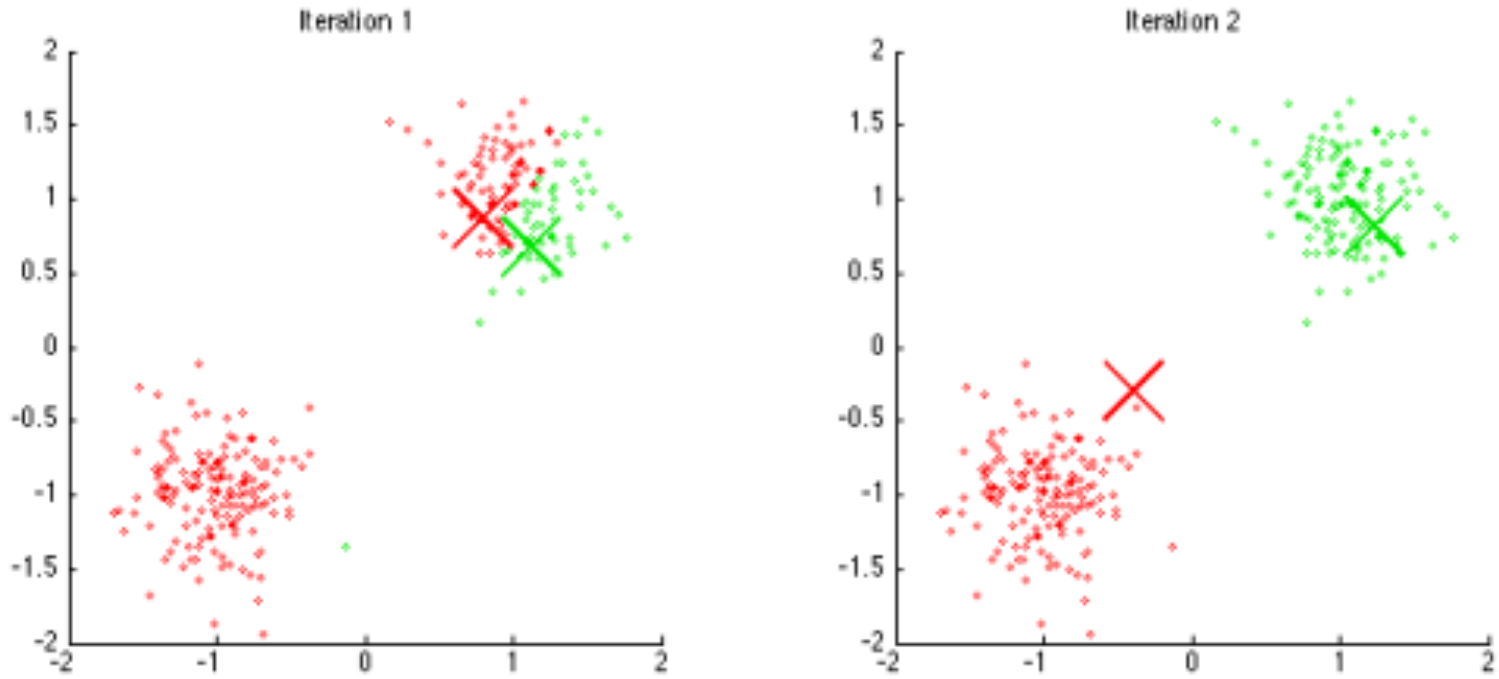
Initialize k representatives $\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_k \in R^d$

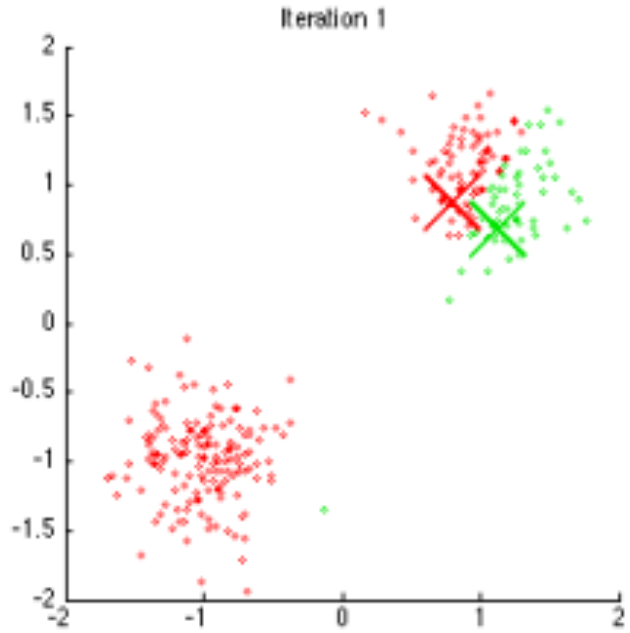
Iterate until convergence:

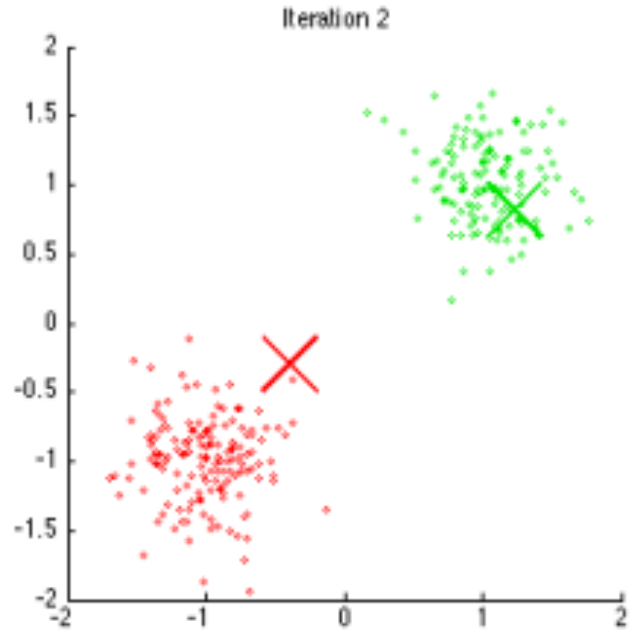
a. Associate each $\vec{x}_i$ with it's closest representative $\vec{x_i} \rightarrow \vec{r_j}$

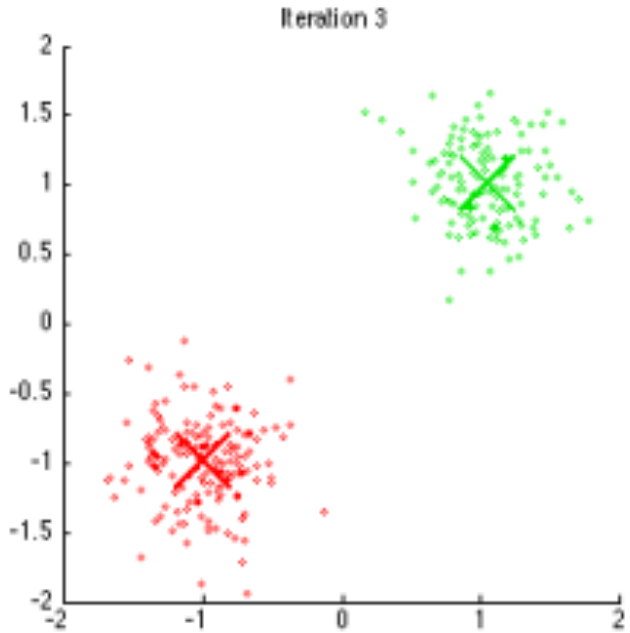
b. Replace each representative $\vec{r}_j$ with the mean of the points assigned to $\vec{r}_j$

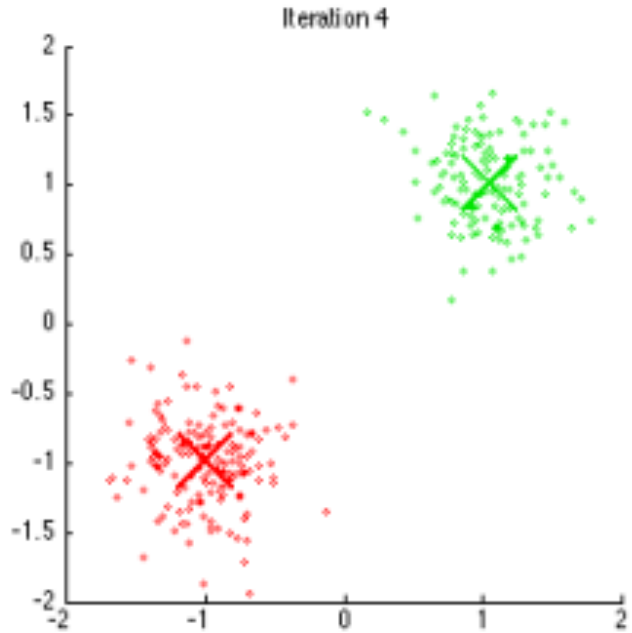
Both a step and b step reduce RMSE




Iteration 1
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2


Iteration 2
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2


Iteration 3
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2


Iteration 4
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2

Summary

- PCA and Regression find a line (or hyperplane) that is close to the data points.
- Kmeans finds a set of representatives such that every data point has a close-by representative.
- The Kmeans algorithm finds a local minimum.
  it might not find the global minimum.

Initialization of Kmeans

Simple Initialization

Simplest Initialization: choose representative from data points independently at random.

- Problem: some representatives are close to each other and some parts of the data have no representatives.
- Kmeans is a local search method – can get stuck in local minima.

Kmeans++

- A different method for initializing representatives.
- Spreads out initial representatives
- Add representatives one by one
  - Before adding representative, define distribution over unselected data points.

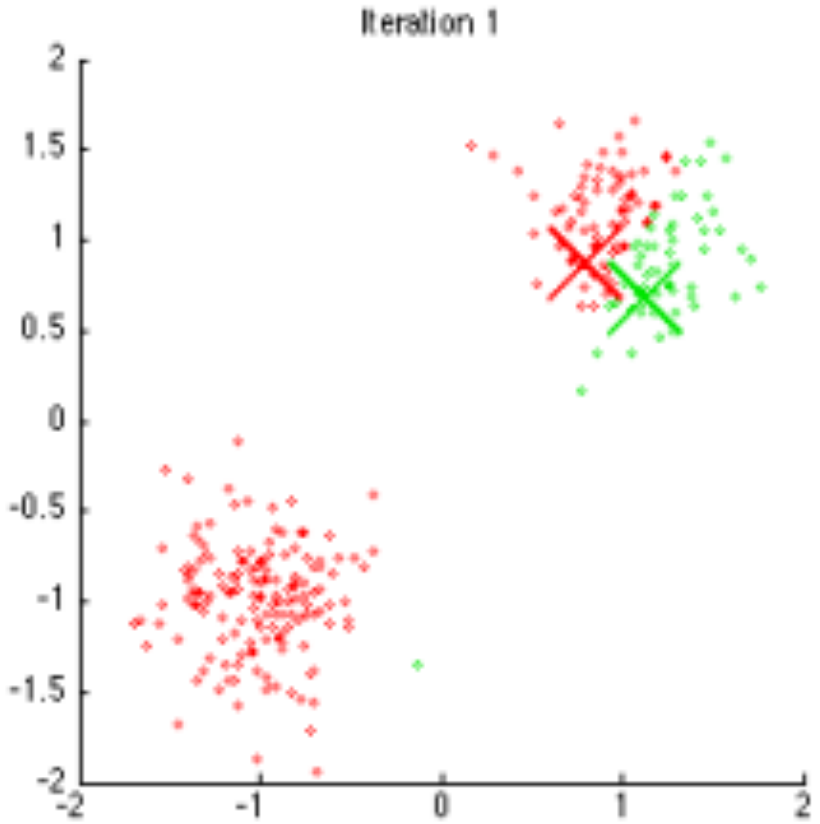
Data: $\vec{x}_1,\ldots,\vec{x}_N$ Current Reps: $\vec{r}_1,\ldots,\vec{r}_j$

Distance of example to Reps: $d(\vec{x},\{\vec{r}_1,\ldots,\vec{r}_j\}) = \min_{1\le i\le j} \| \vec{x} - \vec{r}_i \|$

Prob. of selecting example $\vec{x}$ as next representative: $P(\vec{x}) = \frac{1}{Z} d(\vec{x},\{\vec{r}_1,\ldots,\vec{r}_j\})^2$

Example for Kmeans++

This is an unlikely initialization for kmeans++



Parallelized Kmeans

- Suppose the data points are partitioned randomly across several machines.
- We want to perform the a,b steps with minimal communication btwn machines.

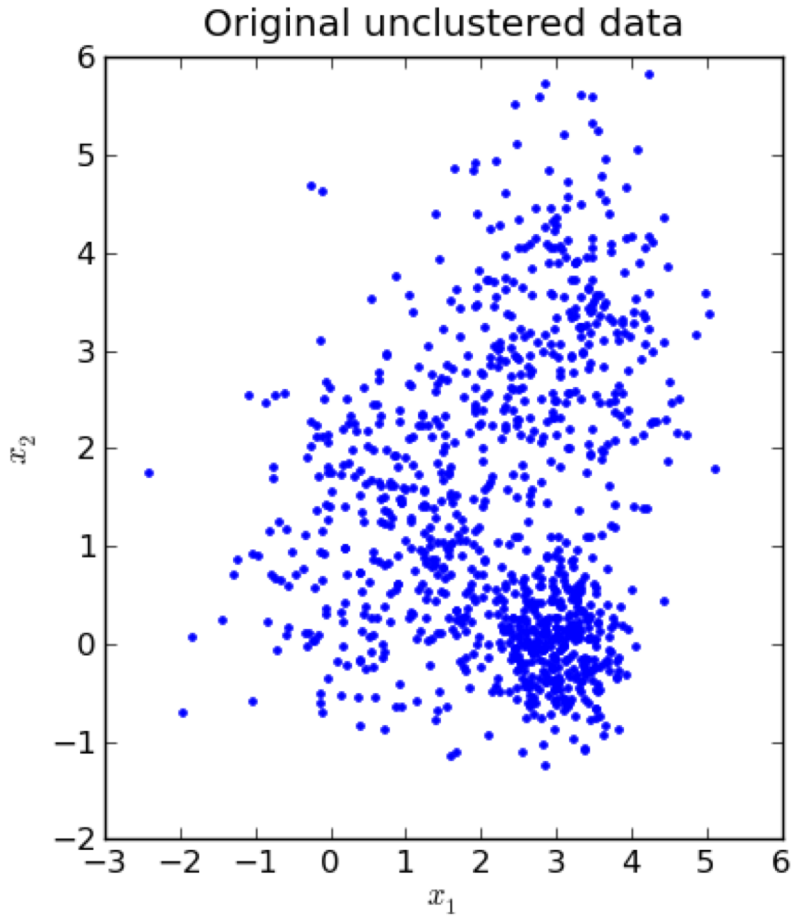
1. Choose initial representatives and broadcast to all machines.
2. Each machine partitions its own data points according to closest representative. Defines (key,value) pairs where key=index of closest representative. Value=example.
3. Compute the mean for each set by performing reduceByKey. (most of the summing done locally on each machine).
4. Broadcast new reps to all machines.

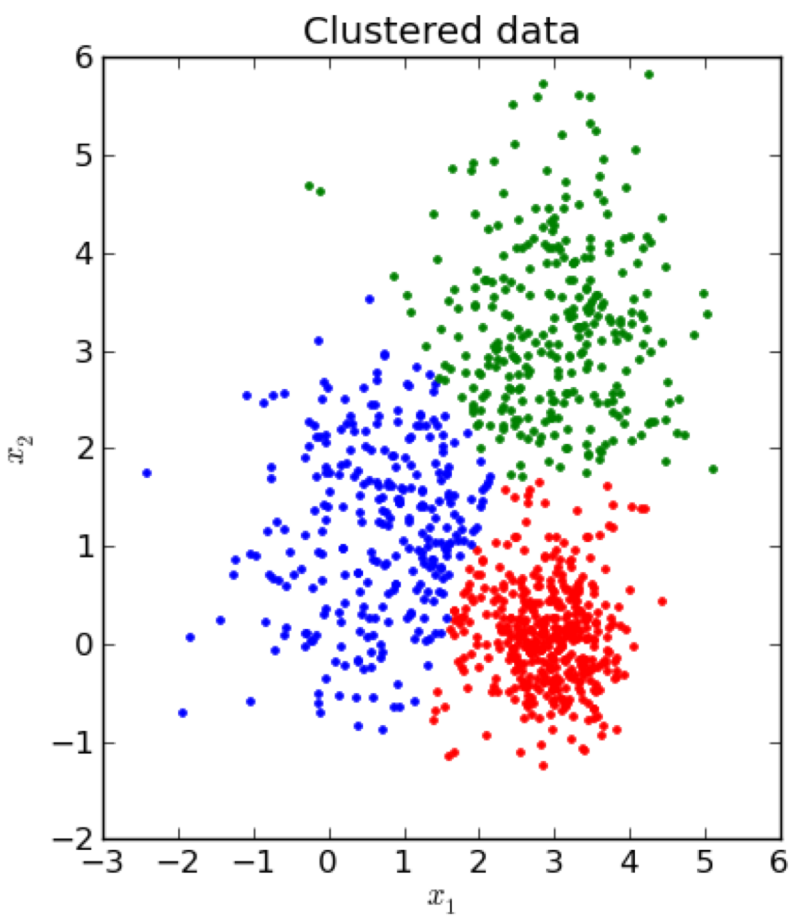
Summary

- Simple initialization: select representatives by choosing examples uniformly at random.
- Kmeans++: Select representatives with higher probability it they are far from existing representative.
- Parallelizing K-means: all computers have the same set of representative, but a different subset of the data.

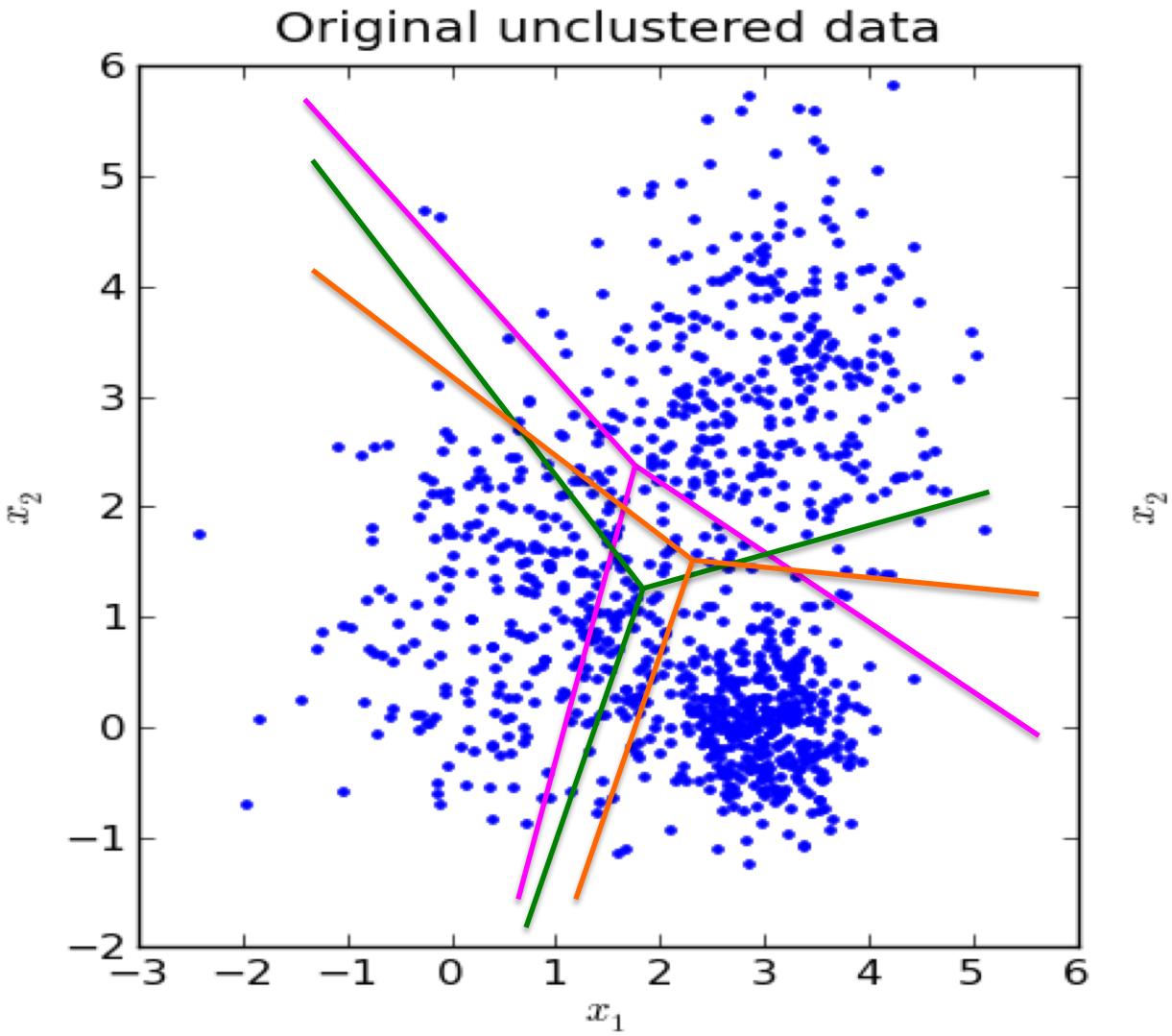
Cluster Stability

Clustering stability





Clustering stability



Clustering using
Starting points I

Clustering using
Starting points 2

Clustering using
Starting points 3

What is clustering stability?

- We say that the clustering is stable if the grouping of the point is similar regardless of starting point.
- Reasons for instability
  - Incorrect number of clusters
  - Clusters are not round
  - Distribution is a ridge, rather than isolated peaks.

Measuring clustering stability

Entry in row “clustering j”, column “xi” contains the index of the closest representative to xi for clustering j

	x1	x2	x3	x4	x5	x6			xn
Clustering 1	1	1	3	1	3	2	2	2	3
Clustering 2	2	2	1	2	1	3	3	3	1
Clustering 3	2	2	3	2	3	1	1	1	3
Clustering 4	1	1	1	1	**3**	3	3	3	1

The first three clusterings are completely consistent with each other
The fourth clustering has a disagreement in x5

How to quantify stability?

- We say that a clustering is **stable** if the examples are always grouped in the same way.
- When we have thousands of examples, we cannot expect all of them to always be grouped the same way.
- We need a way to quantify the stability.
- Basic idea: measure how much groupings differ between clusterings.

Entropy

A partition G of the data defines a distribution over the parts:

$$p_1 + p_2 + \cdots + p_k = 1$$

The information in this partition is measured by the Entropy:

$$H(G) = H(p_1, p_2, \ldots, p_k) = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i}$$

$H(G)$ is a number between

0 (one part with prob. 1)

and

$\log_2 k$ $(p_1 = p_2 = \cdots = p_k = \frac{1}{k})$

Entropy of a combined partition

If clustering1 and clustering 2 partition the data in the exact same way

then $G_1 = G_2$, $\quad H(G_1, G_2) = H(G_1) = H(G_2)$

If clustering1 and clustering 2 are independent

(partition the data independently from each other).

then $H(G_1, G_2) = H(G_1) + H(G_2)$

Suppse we produce many clusterings, using many starting points.

Suppose we plot

$H(G_1), H(G_1, G_2), \ldots, H(G_1, G_2, \ldots, G_i), \ldots$

As a function of i

If the graph increases like $i \log_2 k$ then the clustering is completely unstable

If the graph stops increasing after some i then we reached stability.

Summary

- Clustering stability