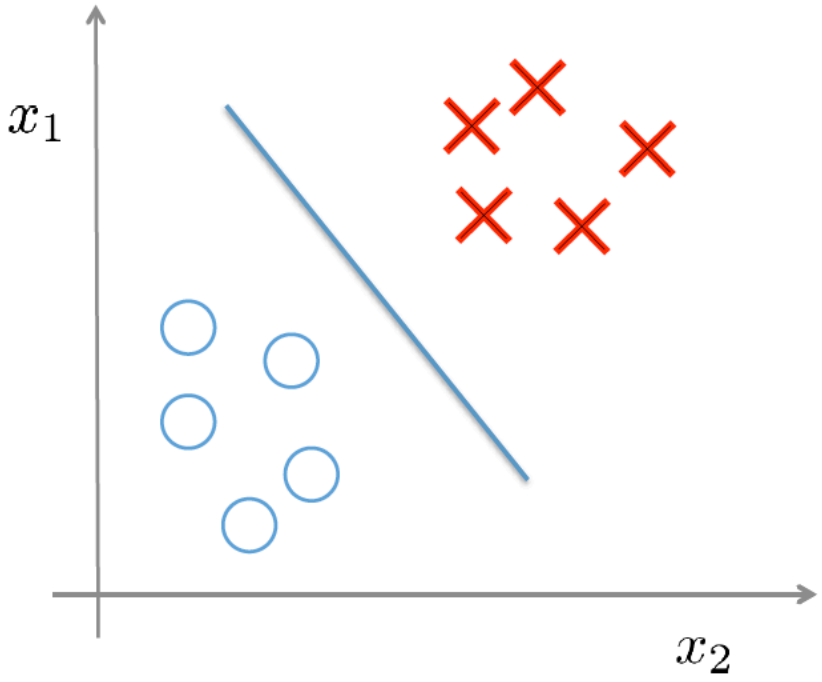
Clustering by proximity to prototypes

Supervised vs Unsupervised learning

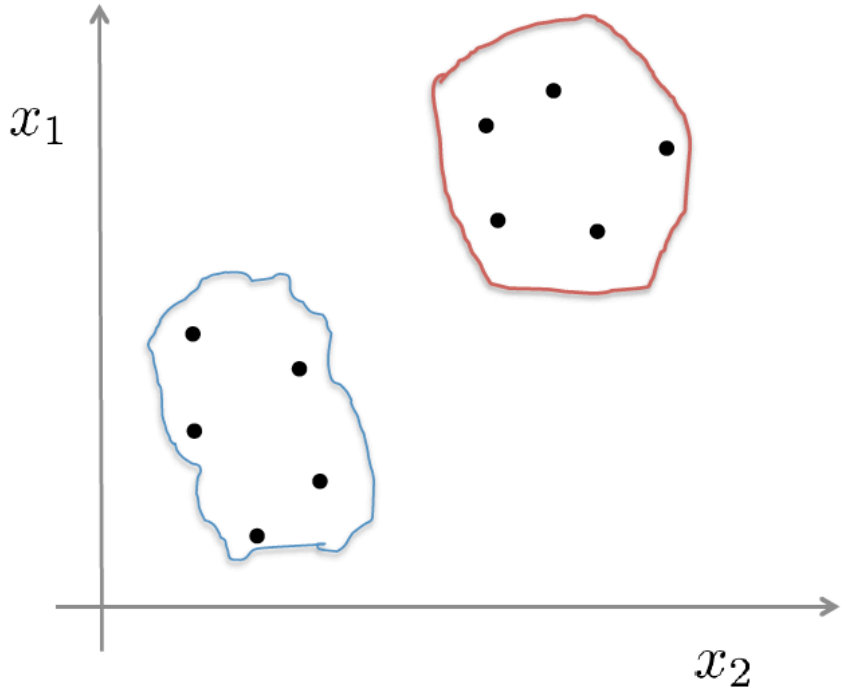
Supervised Learning



$Training\ set$: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

Supervised vs Unsupervised learning

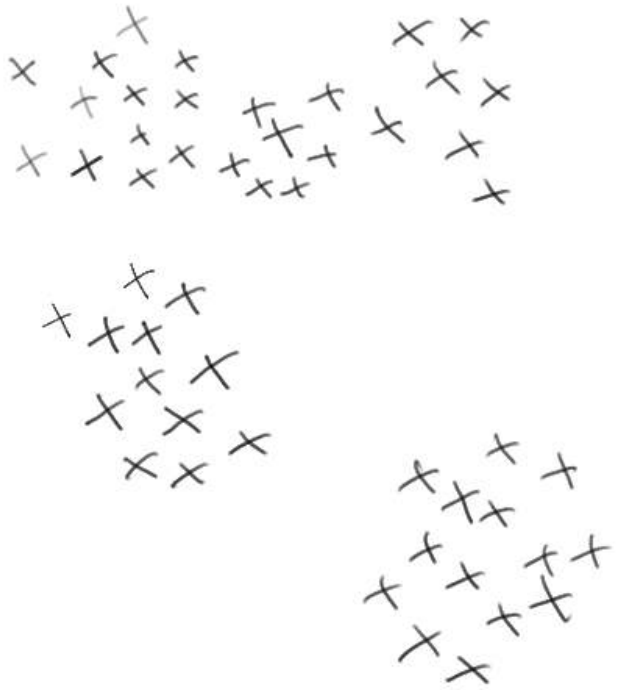
Unsupervised Learning (Clustering)



$Training\ set$: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

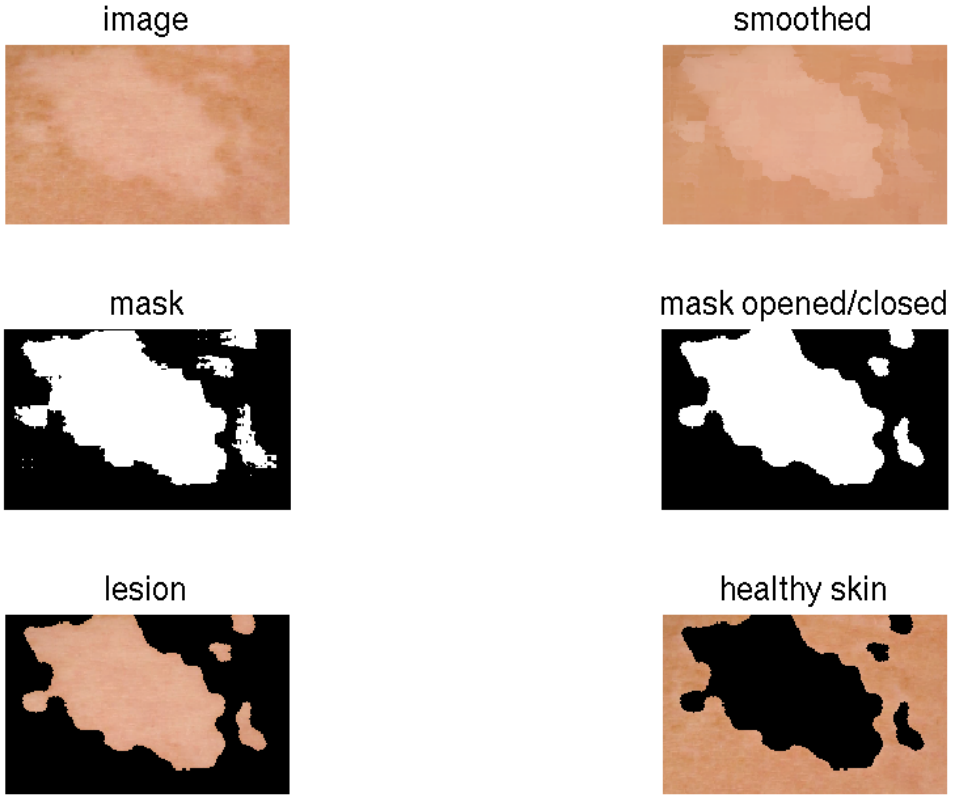
What is the goal of clustering?

Division of a data set X into k disjoint subsets C_1 … C_k such that objects within each subset are similar and objects in different subsets are dissimilar



Similarity

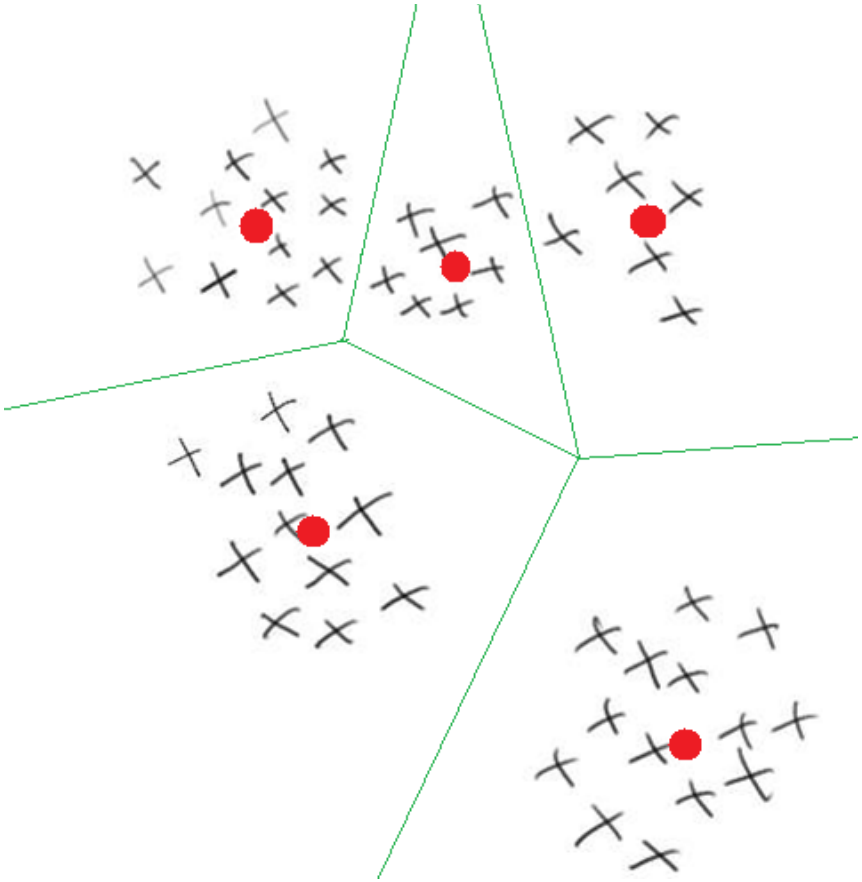
Example



Example of 2-means clustering: a skin image is segmented in two regions of lesion and healthy skin by grouping pixels in two clusters according to their color (result shown in image 'mask')

k-means clustering

Euclidean-distance, prototype-based clustering



k-means clustering

Given: elements x^j in R^n, number of clusters k

Goal: find k prototypes $\vec{\mu}^1, \ldots, \vec{\mu}^k \in R^n$

that minimize the quantization error

$$J_e = \frac{1}{2} \sum_{\vec{\mu}^i} \sum_{\vec{x}^j \in C(\vec{\mu}^i)} \left\| \vec{x}^j - \vec{\mu}^i \right\|^2$$

$C(\mu^i)$ – cluster (subset of X) associated with μ^i (also called receptive field of μ^i)

Lloyd's algorithm for k-means clustering

Idea : J_e is hard to minimize directly, but it is easy

- to find optimum prototypes for given clusters

$$minimize \frac{1}{2} \sum_{\vec{\mu}^i} \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} \left|\left|\vec{x}^j - \vec{\mu}^i\right|\right|^2 \; for\ known\ C(\vec{\mu}^i)$$

$$i.e\ \partial \frac{1}{2} \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} ||\vec{x}^j - \vec{\mu}^i||^2 / \partial \vec{\mu}^i = 0$$

$$i.e\ \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} (\vec{\mu}^i - \vec{x}^j) = 0$$

$$i.e\ \vec{\mu}^i = \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} \vec{x}^j \; / \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} 1$$

- to find optimum clusters for given prototypes

$$minimize \frac{1}{2} \sum_{\vec{\mu}^i} \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} \left|\left|\vec{x}^j - \vec{\mu}^i\right|\right|^2 \; for\ known\ \vec{\mu}^i$$

$$i.e\ C(\vec{\mu}^i)\ collects\ the\ points\ \vec{x}^j\ with\ smallest\ \left|\left|\vec{x}^j - \vec{\mu}^i\right|\right|^2$$

Lloyd's algorithm for k-means clustering

1. begin **initialize** $\mu^1, \mu^2, \ldots, \mu^k$ (e.g. take randomly k samples from the data set)
2. do **assign** data points to nearest μ^i (compute C_i)
3. **re-compute** μ^i as the mean of points in C_i
4. **until no change** in $\mu^1, \mu^2, \ldots, \mu^k$
5. **return** $C_1, C_2, \ldots, C_k$ and $\mu^1, \mu^2, \ldots, \mu^k$
6. end

$init\ \vec{\mu}^i$

repeat

$$C(\vec{\mu}^i) := \{\vec{x}^j \,|\, ||\vec{x}^j - \vec{\mu}^i|| < ||\vec{x}^j - \vec{\mu}^m|| \forall \mathrm{m} \neq \mathrm{i}\}$$

$$\vec{\mu}^i := \sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} \vec{x}^j / |C(\vec{\mu}^i)|$$

Does Lloyd's algorithm converge?

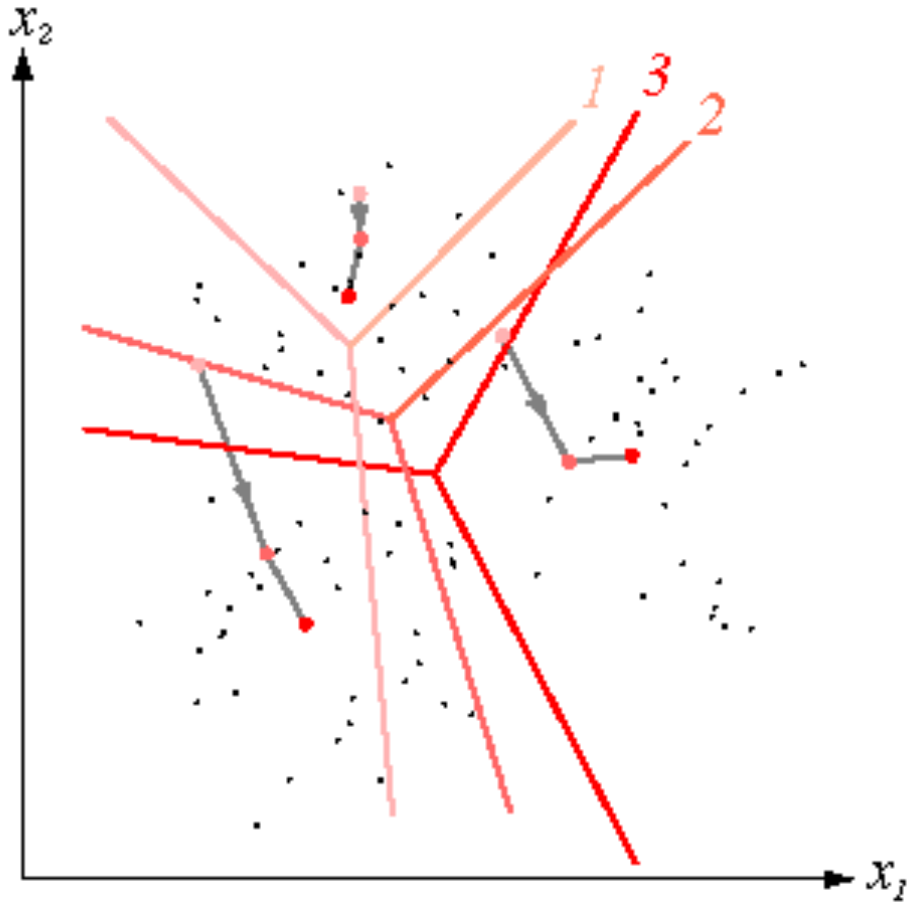
Yes, in a finite number of steps, because a non-negative cost function (the quantization error) decreases (or remains constant) with each step:

$$J_e = \frac{1}{2}\sum_{\vec{\mu}^i}\sum_{\vec{x}^j \in C(\vec{\mu}^i)} ||\vec{x}^j - \vec{\mu}^i||^2$$

$$\vec{\mu}^i = \frac{1}{n}\sum_{\vec{x}^j \in C(\vec{\mu}^i)} \vec{x}$$

However, there is **no guarantee** that a global minimum is reached

Example of k-means clustering



Evolution of the (3) computed means (and Voronoi cells) during 3-means clustering

(from Duda, Hart, Stork (2001) Pattern classification)

Iris Data

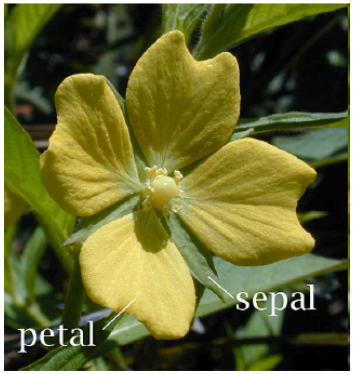
Setosa

Versicolor

Virginica

- 50 samples from each of 3 species
- 4 features per sample:

  length & width of sepal and petal

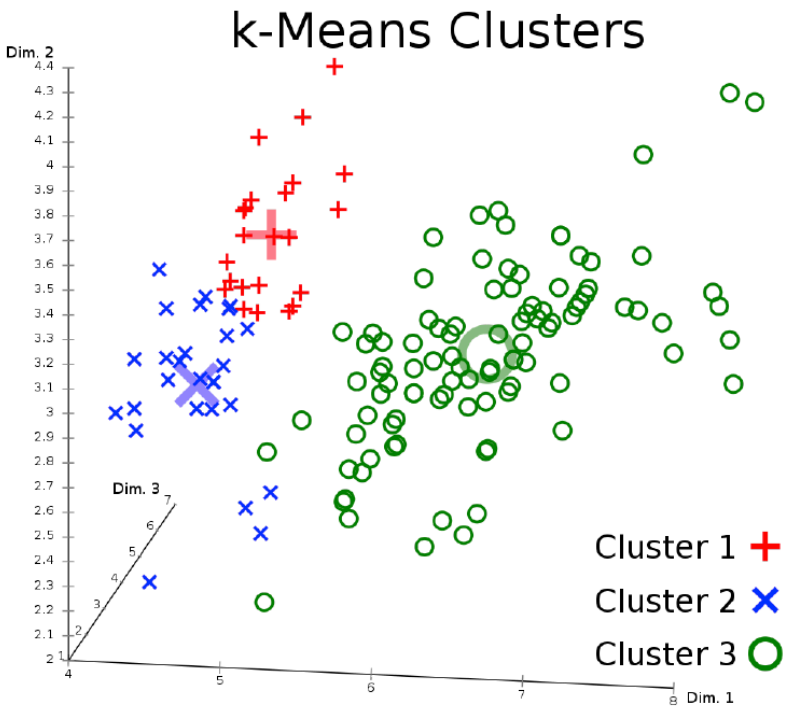


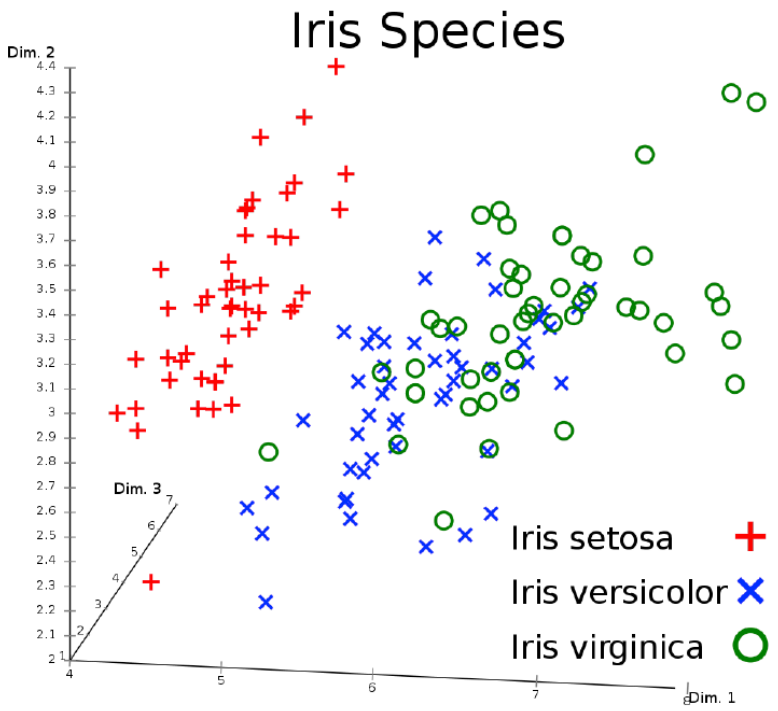
Enrique Alegre based on Nicolai Petkov's slides

Iris data

- see UCI repository: http://archive.ics.uci.edu/ml/

  150 points; 4 dim, 3 classes



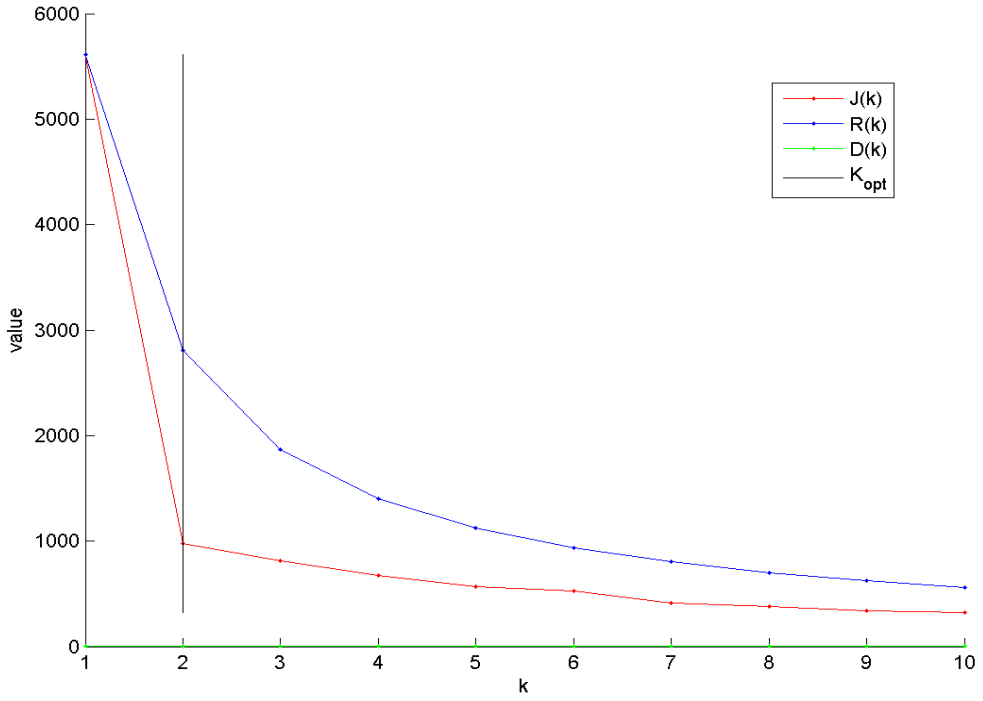


K-means applet

http://www.cs.rug.nl/~petkov/teaching/PatternRecognition/supplements/k-means/

How to choose k?

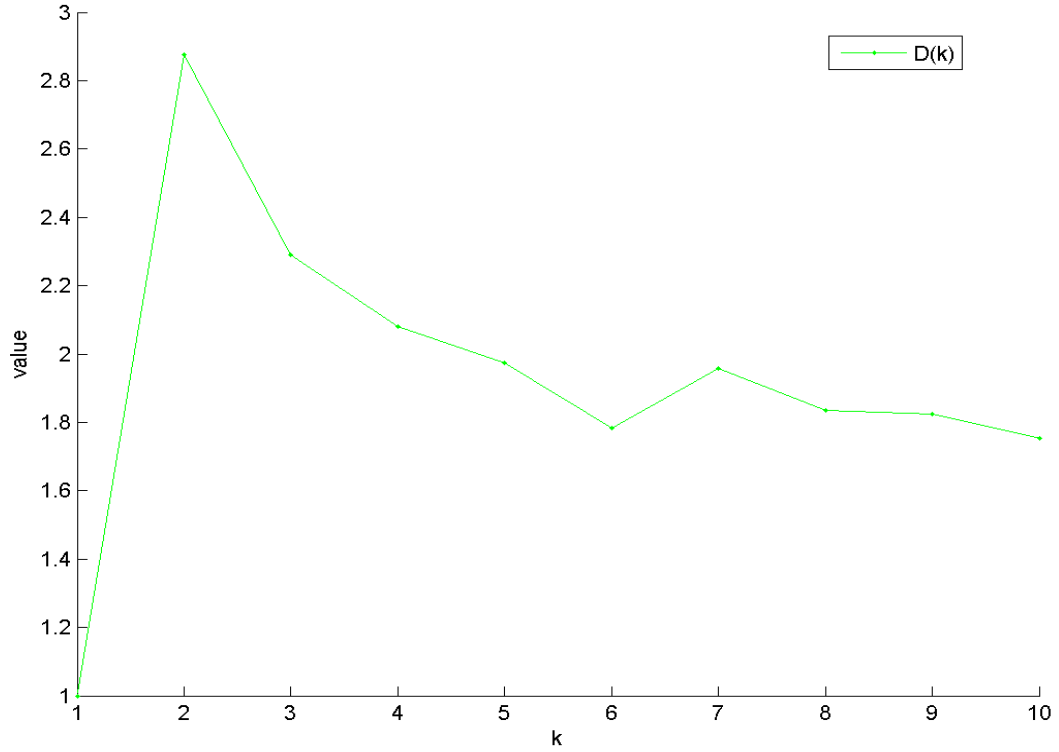
1. Compute the quantization error as a function of k: J(k)
2. Compute the quantization error R(k) for a uniformly distributed reference data set. ($R(k) \sim k^{-2/d}$, d – dimensionality)



How to choose k?

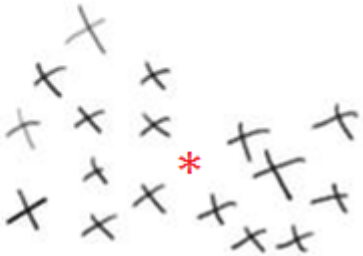
Find the maximum of the ratio D(k)=R(k)/J(k).

$$k_{opt} = \arg(\max_k(D(k)))$$

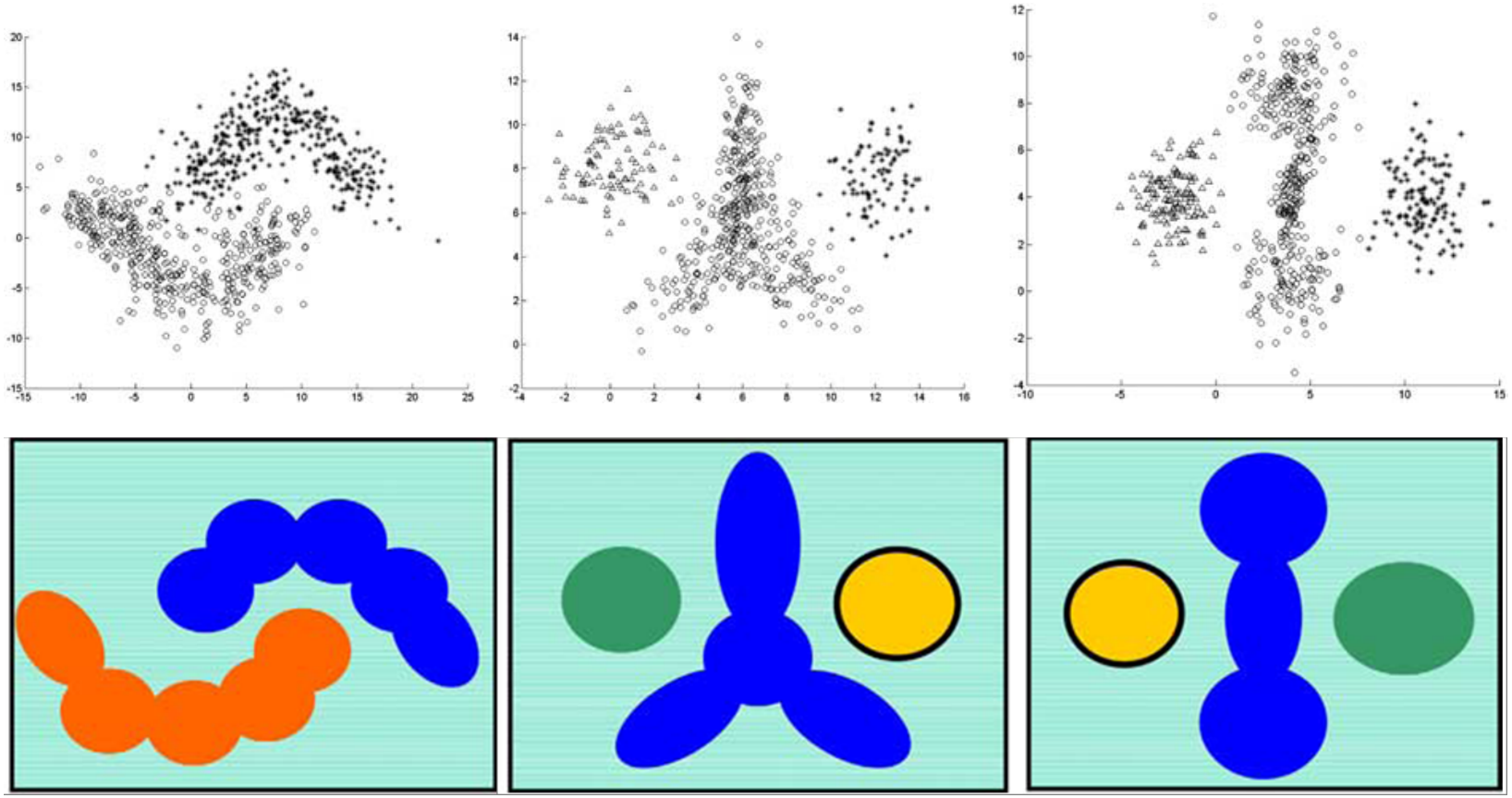


Problems with k-means clustering: dead prototypes

If some prototypes are initialized far away from the input data set, no data points are assigned to them and they are never used



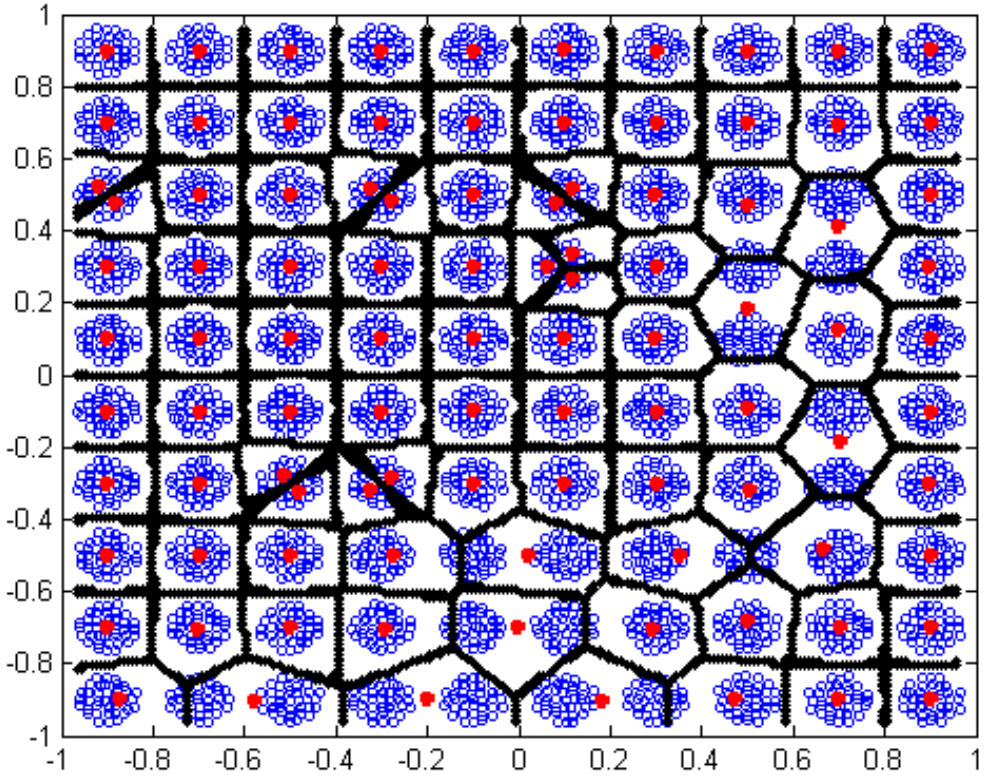
Problems with k-means clustering: non-spherical clusters



Examples of non-spherical clusters: (a) Teaeguk, (b) Triangle, (c) Xours (Cho et al.,2006)

Problems: local optima

Checkerboard data with 100 data clusters and their cluster centers

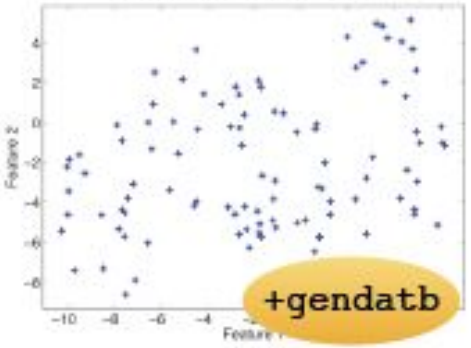


k-means and k-centers

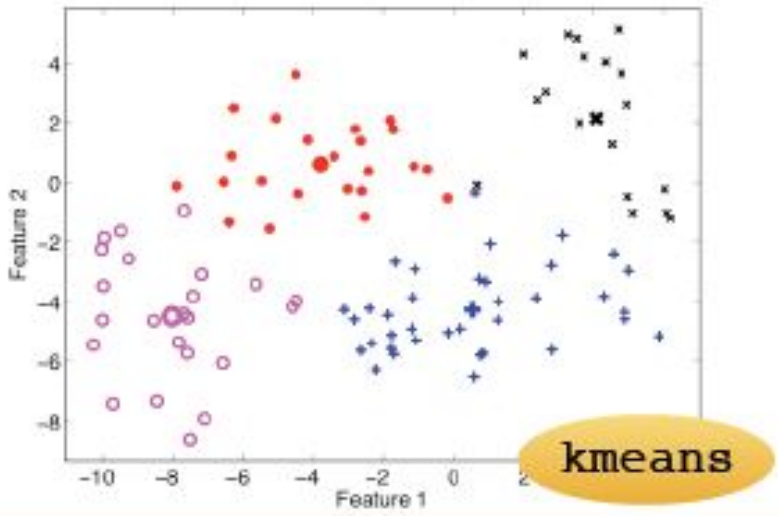
- *k*-centres minimizes the maximum distance
- within objects in the cluster

- *k*-centres selects existing objects as prototypes
- *k*-means constructs prototypes not existing
- in the training set

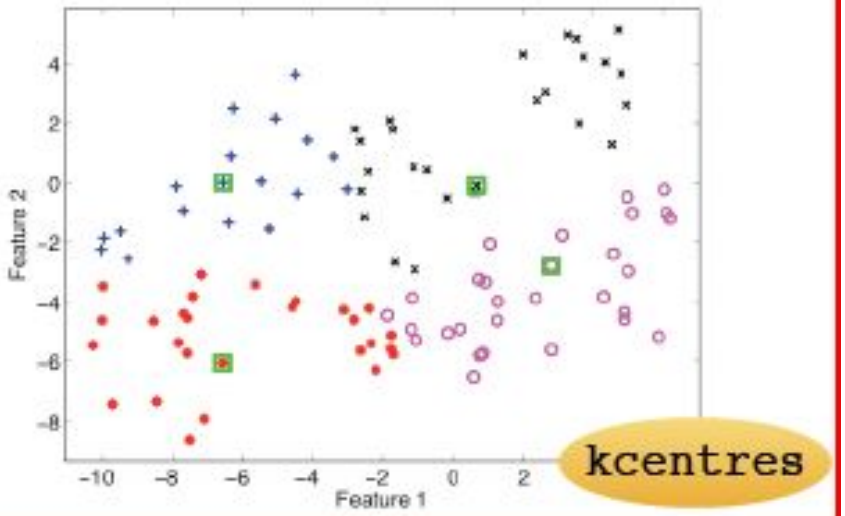
banana dataset



k-means clustering



k-centres clustering



From Advanced Pattern Recognition, TUD2009

Summary of k-means algorithm

Advantages

- Generally fast
- A (local) optimum is usually found

Drawbacks

- Difficult to find the global optimum
- Inizialization (position and number K of clusters)
- Sensitive to outliers
- Works for Gaussian-shaped clusters
- Sensitive to variation in size and density of the clusters

Fuzzy k-means clustering algorithm

In k-means, a data point is assumed to belong to one single cluster.

In fuzzy k-means, the point can belong to multiple clusters, the membership to a cluster has some "graded" value

The cluster membership values are normalized:

$$\hat{P}(\omega^i \mid \mathrm{x}^{\mathrm{j}})$$

$$\sum_{i=1}^{k} \hat{P}(\omega^i \mid \mathrm{x}^{\mathrm{j}}) = 1, \quad j = 1,\ldots,n$$

Fuzzy k-means clustering algorithm

We seek to minimize a heuristic global cost function:

$$J_{fuz} = \sum_{i=1}^{k} \sum_{j=1}^{n} [\hat{P}(\omega^i \mid \mathrm{x}^{\mathrm{j}})]^b \, \| \mathrm{x}^{\mathrm{j}} - \mu^i \|^2$$

Solution can be found from:

$$\partial J_{fuz} / \partial \mu^i = 0$$

$$\partial J_{fuz} / \hat{P}(\omega^i \mid \mathrm{x}^{\mathrm{j}}) = 0$$

Fuzzy k-means clustering algorithm

Solution:

$$\mu_i = \frac{\sum_{j=1}^{n} [\hat{P}(\omega^i \mid \mathrm{x}^j)]^b \ \mathrm{x}_j}{\sum_{j=1}^{n} [\hat{P}(\omega^i \mid \mathrm{x}^j)]^b}$$

$$\hat{P}(\omega^i \mid \mathrm{x}^j) = \frac{1}{\sum_{r=1}^{k} (d_{ij} / d_{rj})^{2/(b-1)}} \quad and \quad d_{ij} = \| \mathrm{x}^j - \mu^i \|^2$$

Fuzzy k-means clustering algorithm

1. begin **initialize** $n, k, \mu^1, \mu^2, \ldots, \mu^k$
2. **normalize** $\hat{P}(\omega^i \mid \mathrm{x}^j)$
3. **do** re-compute μ^i
4. re-compute $\hat{P}(\omega^i \mid \mathrm{x}^j)$
5. **until** small change in μ^i and $\hat{P}(\omega^i \mid \mathrm{x}^j)$
6. **return** $\mu^1, \mu^2, \ldots, \mu^k$ and $\hat{P}(\omega^i \mid \mathrm{x}^j)$
7. end

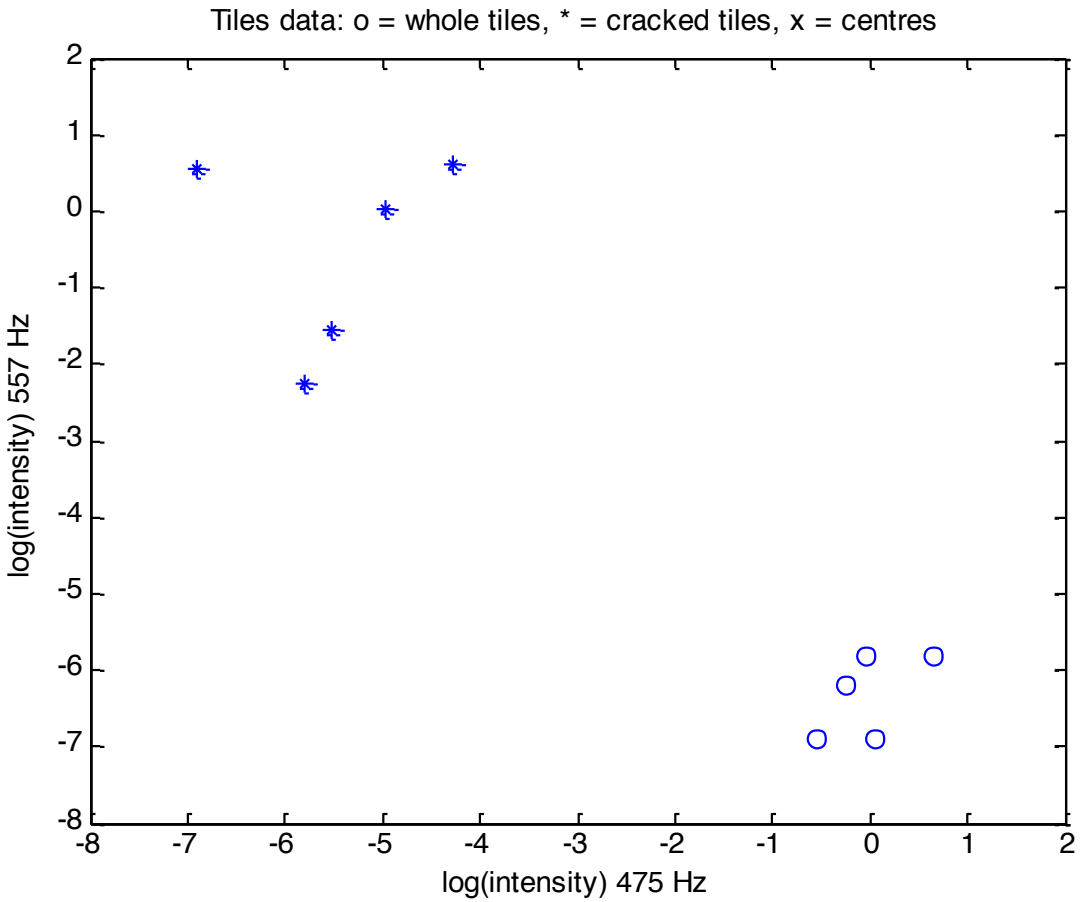
Example: Broken Tiles

475Hz	557Hz	Ok (no cracks)
0.958	0.003	Yes
1.043	0.001	Yes
1.907	0.003	Yes
0.780	0.002	Yes
0.579	0.001	Yes
0.003	0.105	No
0.001	1.748	No
0.014	1.839	No
0.007	1.021	No
0.004	0.241	No

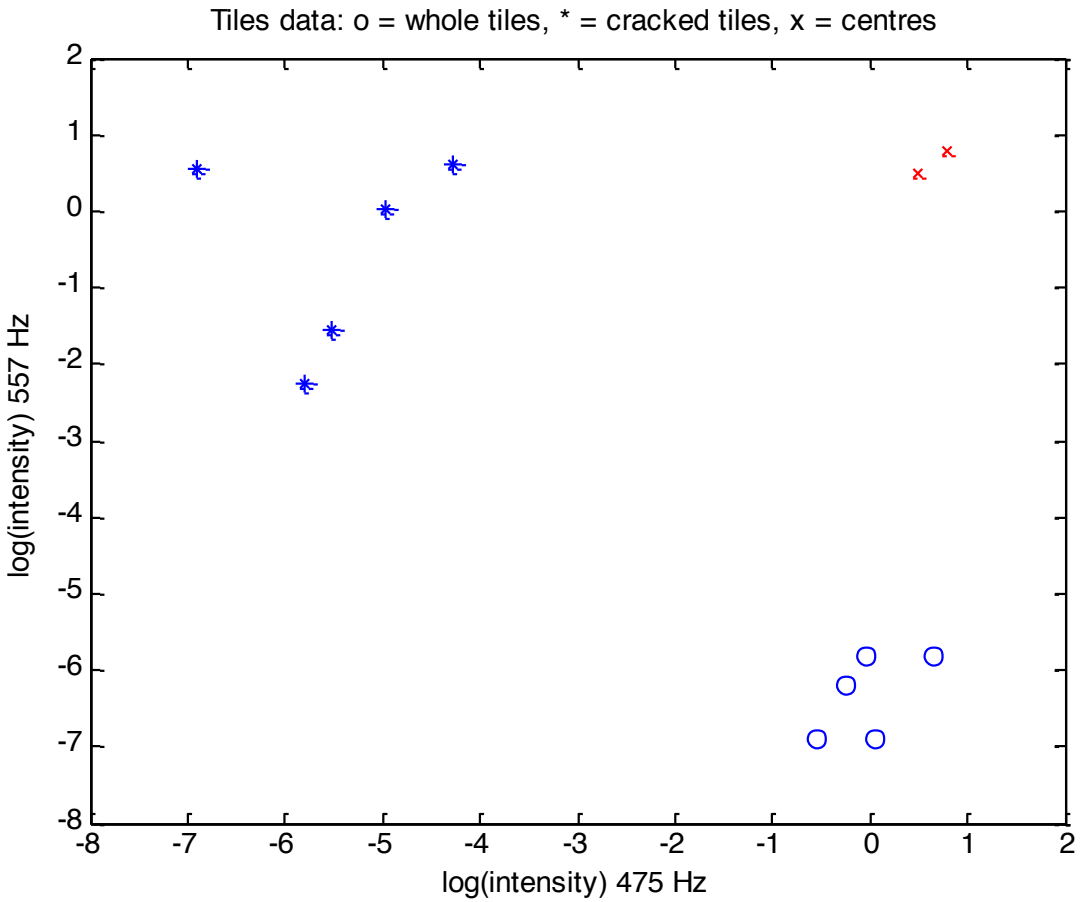
Tiles from clay are shaped, glazed and then baked. Possible inner cracks may remain invisible.

Cracks can be detected by hitting a tile with a hammer and recording the signal response in certain frequencies.

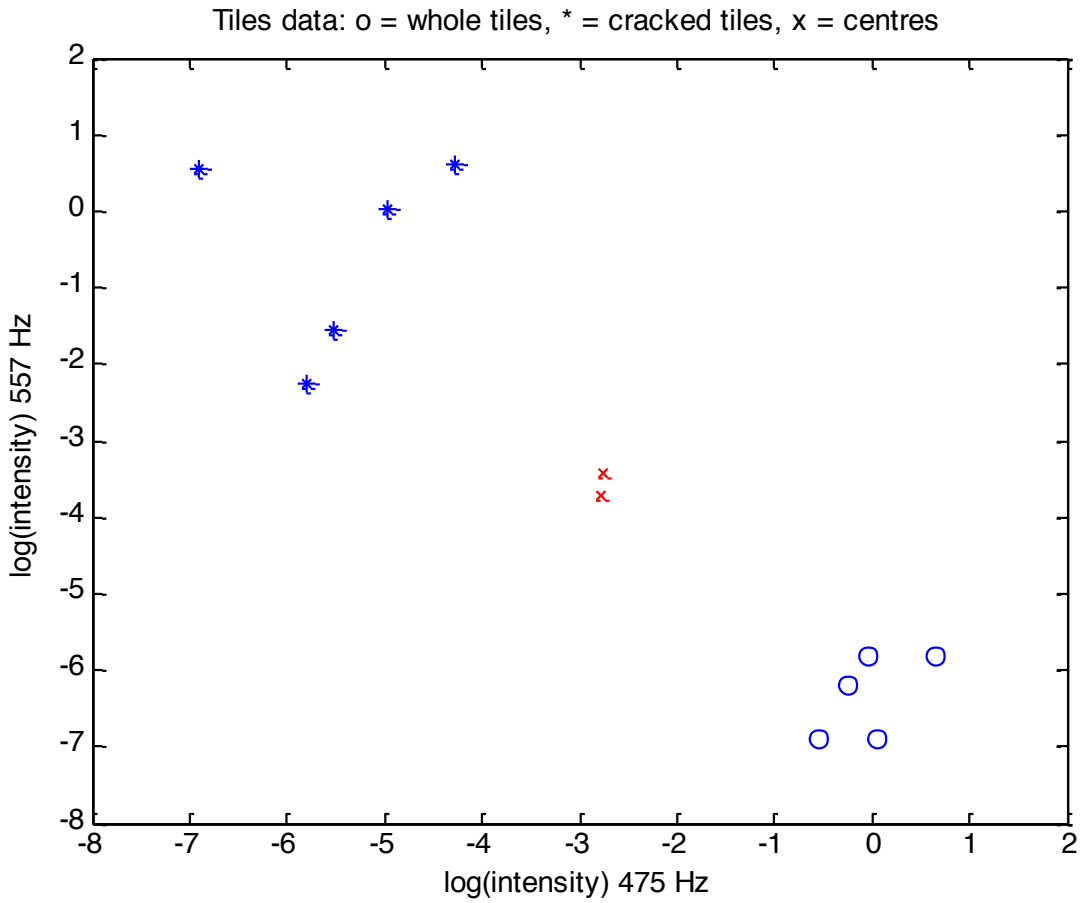
Dataset


Tiles data: o = whole tiles, * = cracked tiles, x = centres
log(intensity) 557 Hz
log(intensity) 475 Hz
2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-8
-7
-6
-5
-4
-3
-2
-1
0
1
2

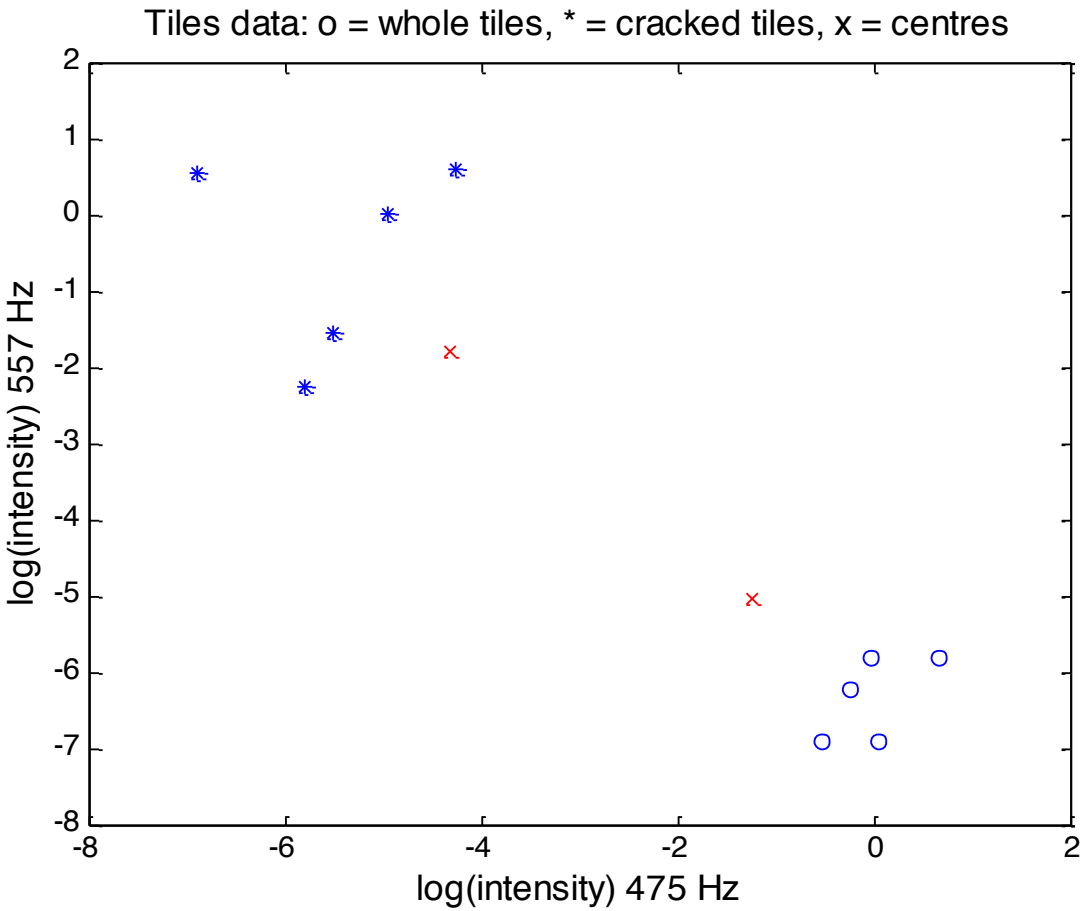
2-means (initialization)



2-means (1st step)

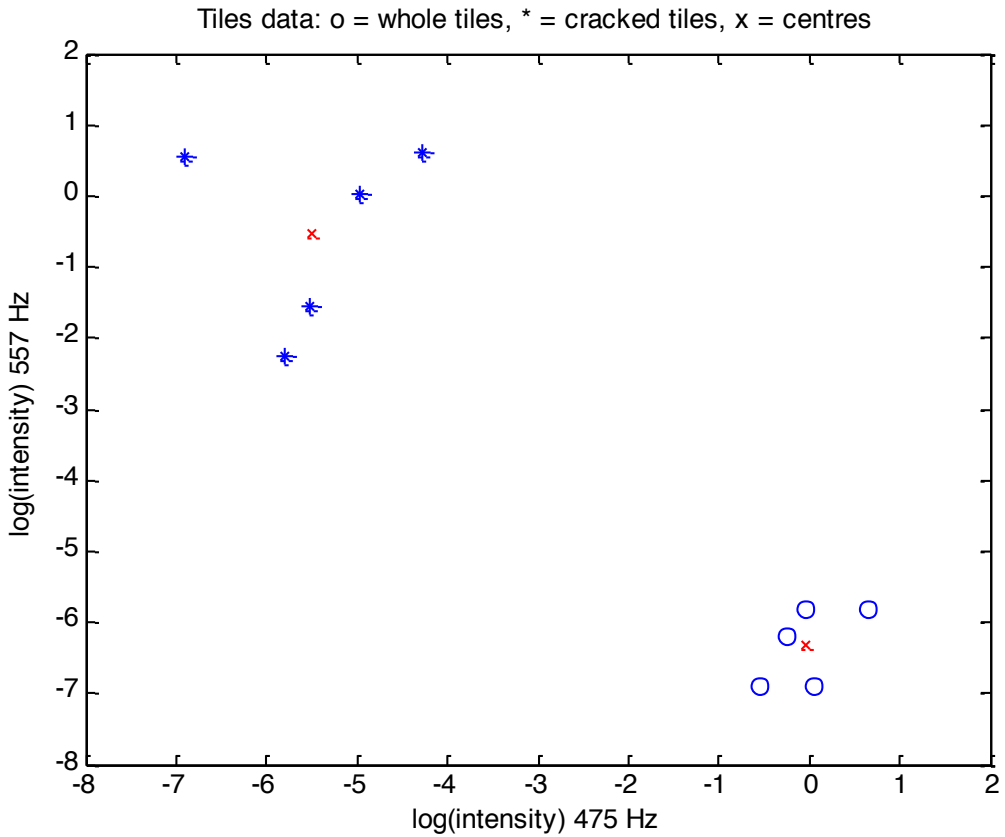


2-means (2nd step)



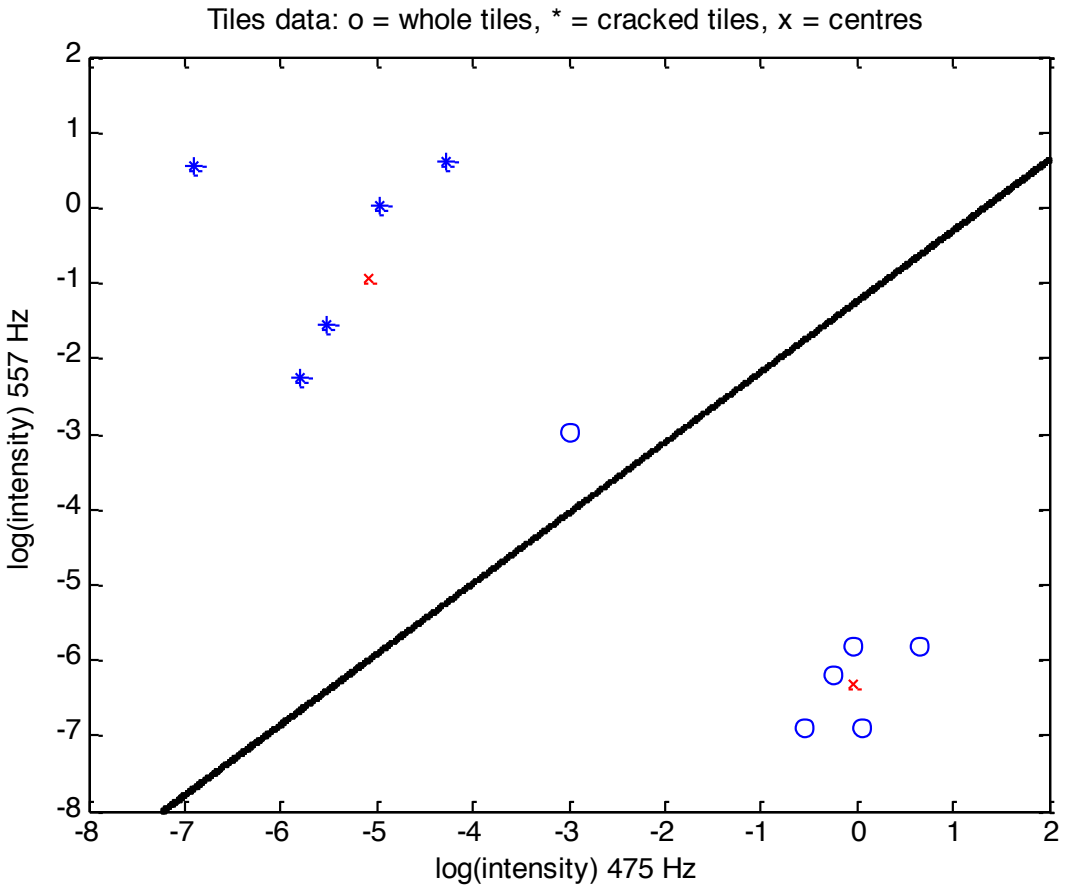
Results k-means

Stability after 3 iterations

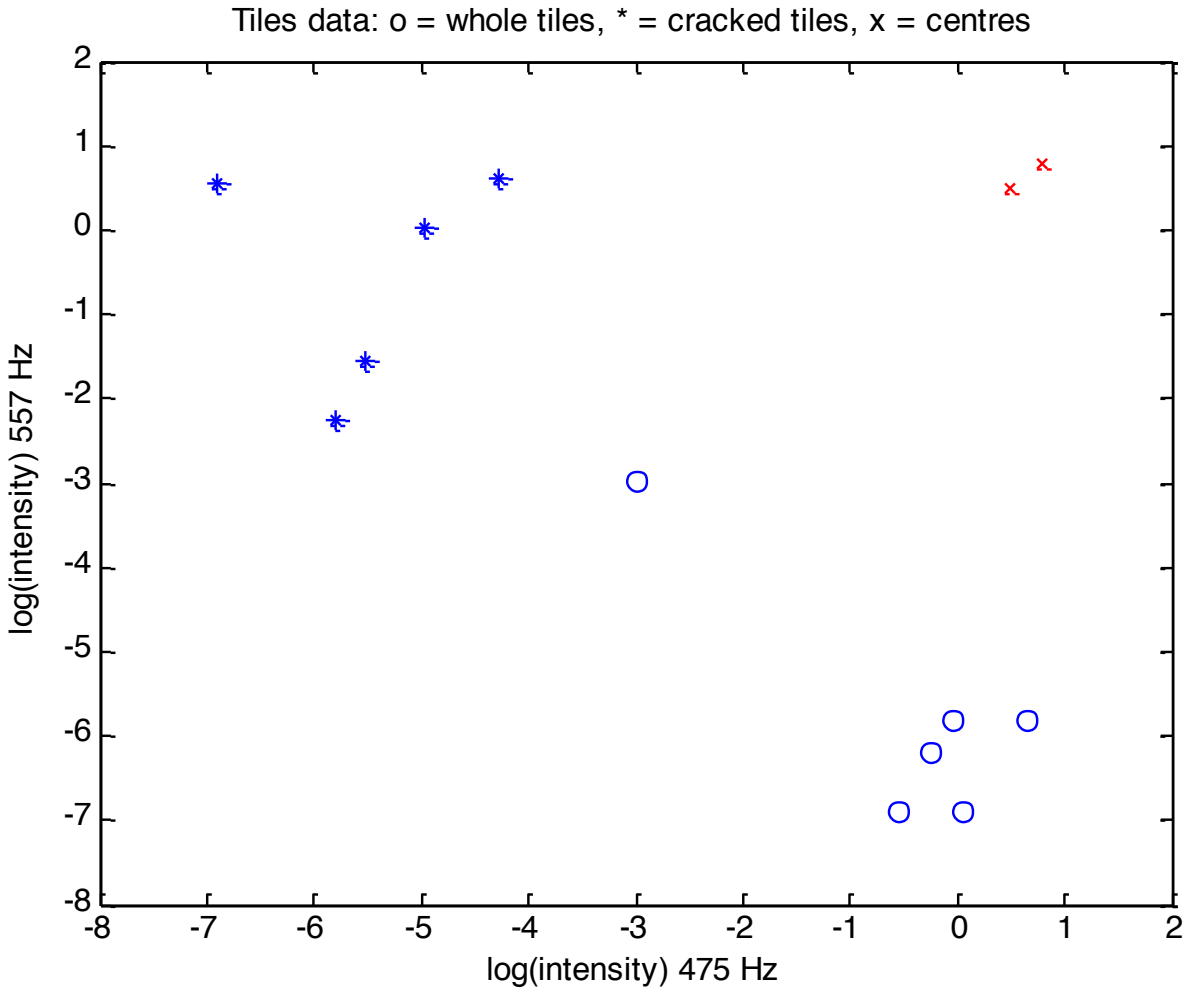


K-means can misclassify!

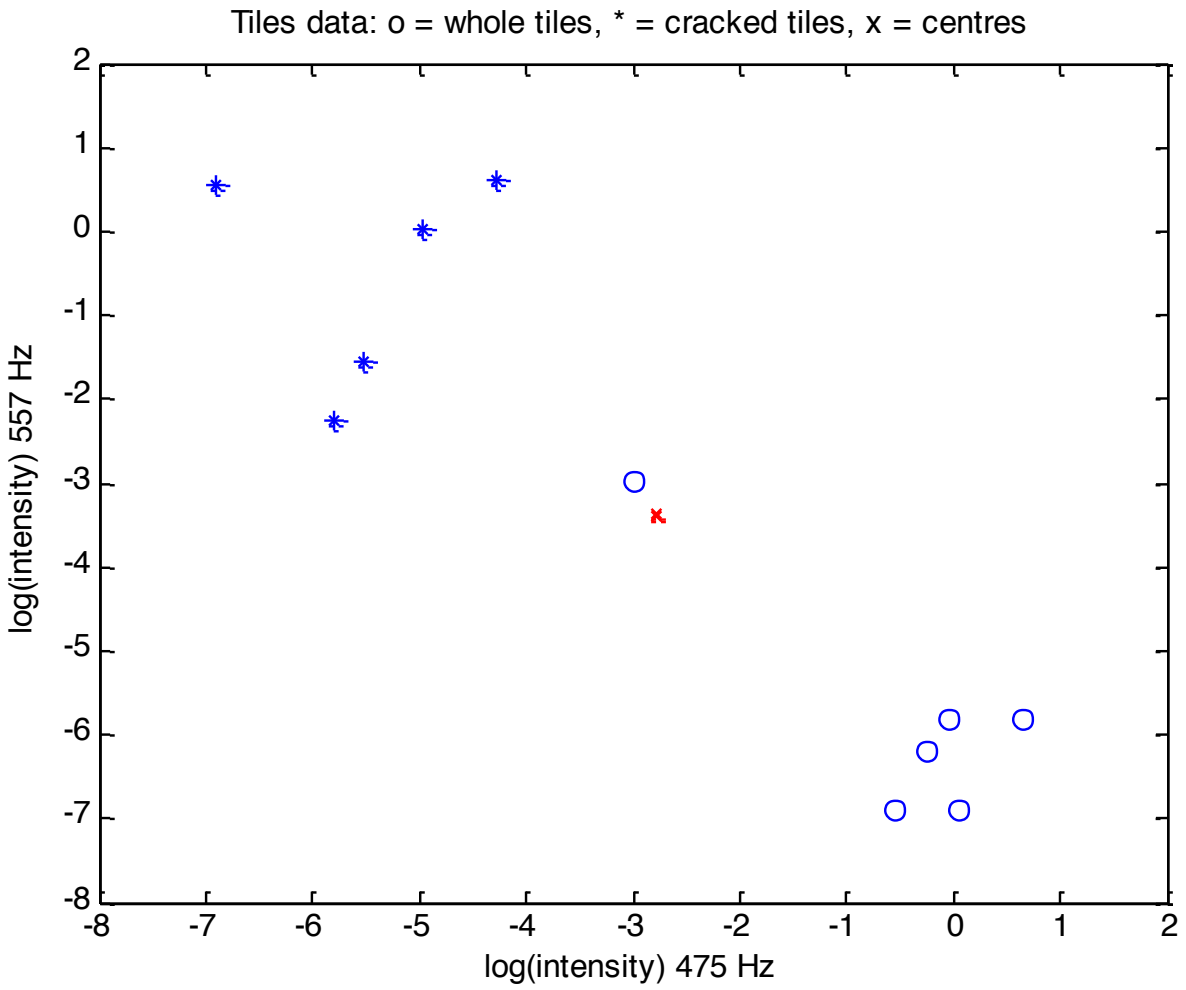
Data points near class boundaries can be misclassified



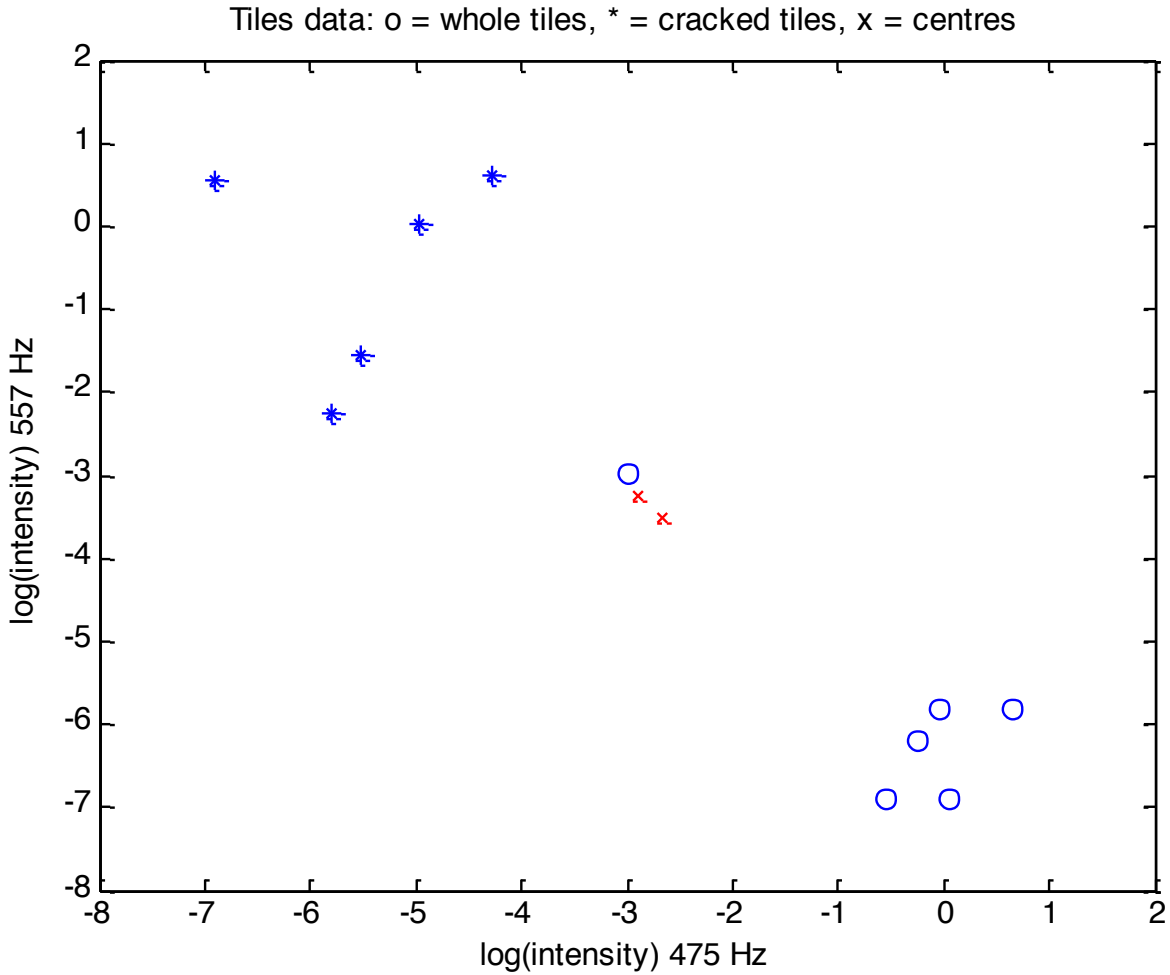
Fuzzy k-means, b=2 (initialization)



Fuzzy k-means (1st step)

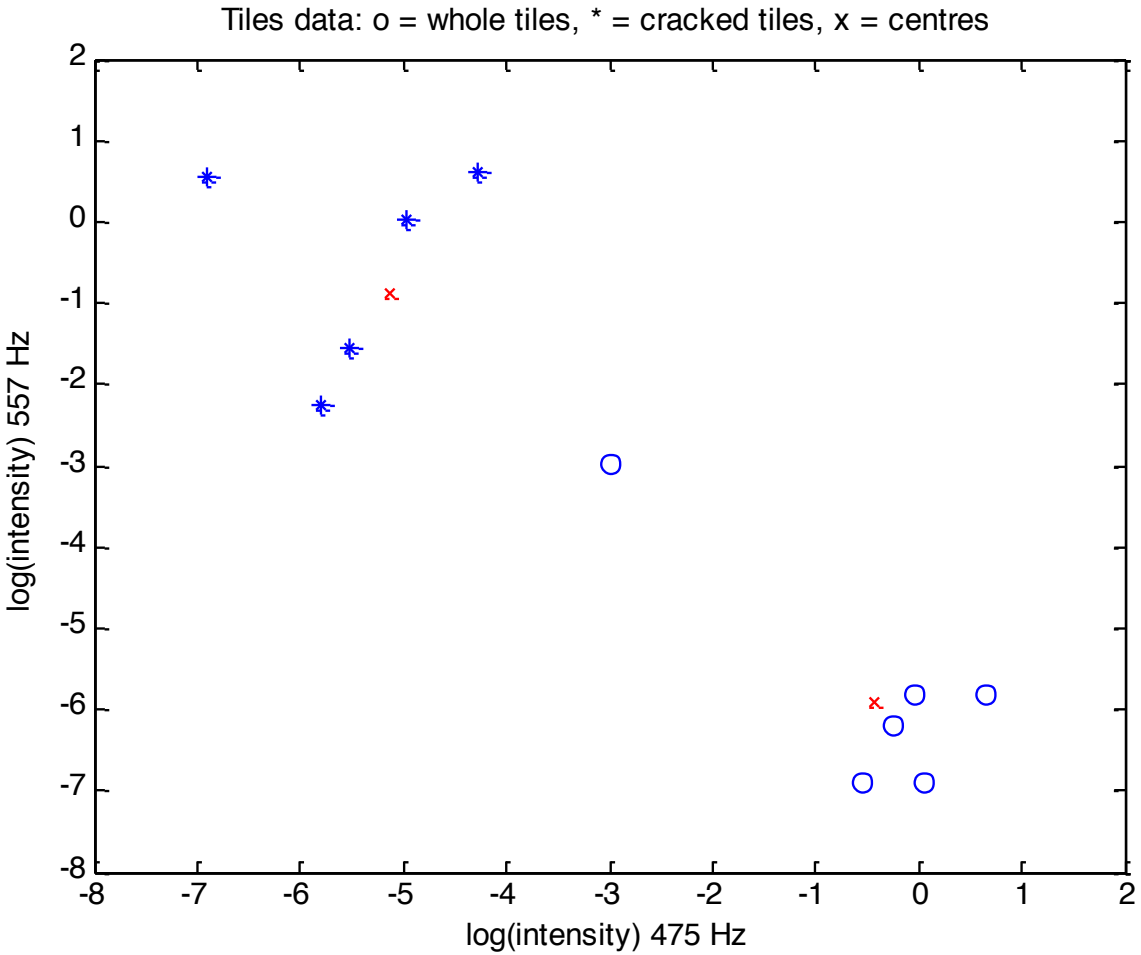


Fuzzy k-means (2nd step)



Result Fuzzy k-means (13th step)

Stability after 13 iterations

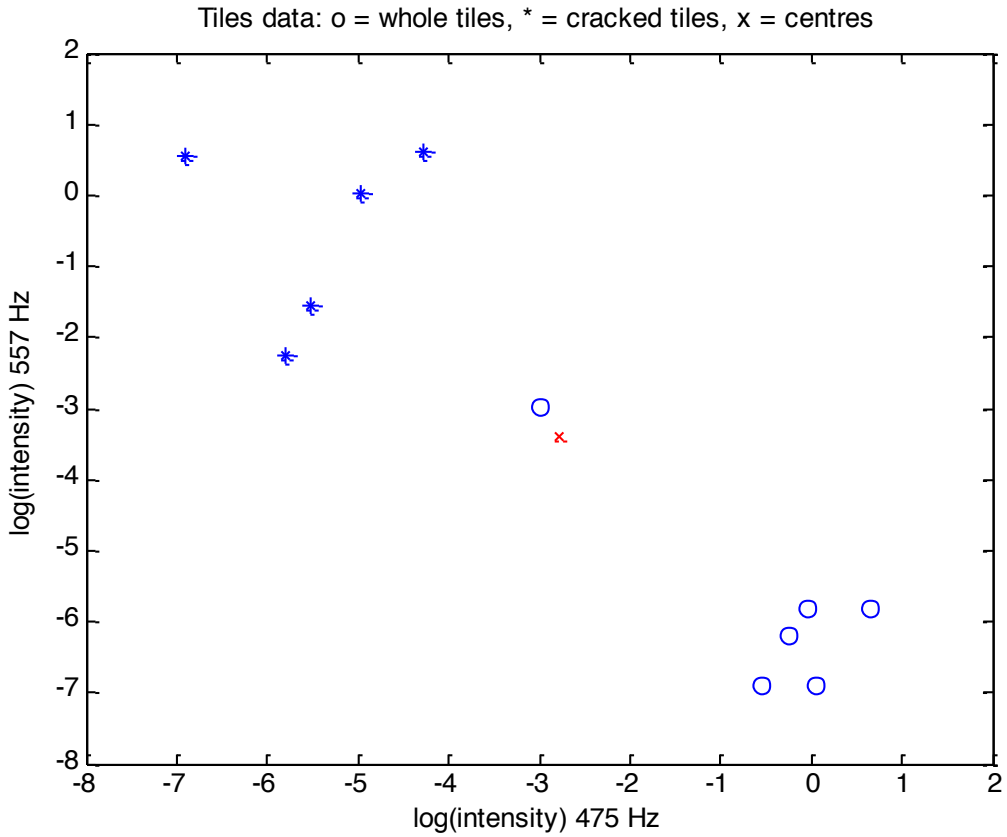


Membership matrix

Xi	Ok (no cracks)	Cracked tiles cluster	Whole tiles cluster
1	Yes	0.0031	0.9969
2	Yes	0.0192	0.9808
3	Yes	0.0196	0.9804
4	Yes	0.0025	0.9975
5	Yes	0.0177	0.9823
6	***Yes***	***0.6223***	***0.3777***
7	No	0.9473	0.0527
8	No	0.9419	0.0581
9	No	0.9507	0.0493
10	No	0.9853	0.0147
11	No	0.9870	0.0130

Convergence of fuzzy k-means

The fuzziness parameter b is important, as wrong b may yield no sensible results! Example with b=3:



Convergence of fuzzy k-means

Resulting membership matrix

Xi	Ok (no cracks)	Cracked tiles cluster	Whole tiles cluster
1	?	0.4994	0.5006
2	?	0.4995	0.5005
3	?	0.4995	0.5005
4	?	0.4994	0.5006
5	?	0.4995	0.5005
6	?	0.5049	0.4950
7	?	0.5006	0.4994
8	?	0.5004	0.4996
9	?	0.5005	0.4995
10	?	0.5005	0.4995
11	?	0.5007	0.4993

Summary of Fuzzy k-means clustering

Advantages

- k-means is a special case of fuzzy k-means (b=1) when each point is a member of only one cluster
- Fuzzy k-means has better convergence properties than k-means

Drawbacks

- The membership depends implicitly on the number of clusters: if this number is incorrectly specified, the final clustering results are wrong.

Optimization by gradient methods

$f : R^n \to R, f(\mu) = \sum_j f(\mu, x^j)$ to be minimized w.r.t μ

gradient vector: $\partial f(\mu, x^j) / \partial \mu^i$, direction of steepest descent for $f(\cdot, x^j)$

stochastic gradient descent:

- iterate: $\mu := \mu - \eta \cdot (\partial f(\mu, x^j) / \partial \mu\ )$
- choose x^j at random

Vector quantization

- Vector quantization = online k-means
- stochastic gradient descent of J_e

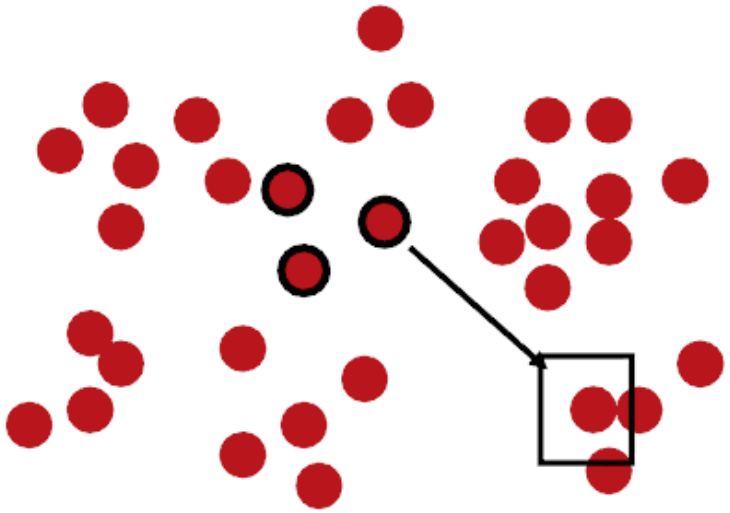
$$minimize\ \frac{1}{2}\sum_{\vec{\mu}^i}\sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)} || \vec{x}^j - \vec{\mu}^i ||^2$$
$$= \frac{1}{2}\sum_{\vec{x}^j \epsilon C(\vec{\mu}^i)}\sum_{\vec{\mu}^i} || \vec{x}^j - \vec{\mu}^i ||^2$$

- stochastic gradient descent
  - init $\vec{\mu}^i$
  - repeat

    choose $\vec{x}^j$

    determine closest $\vec{\mu}^i$, the winner

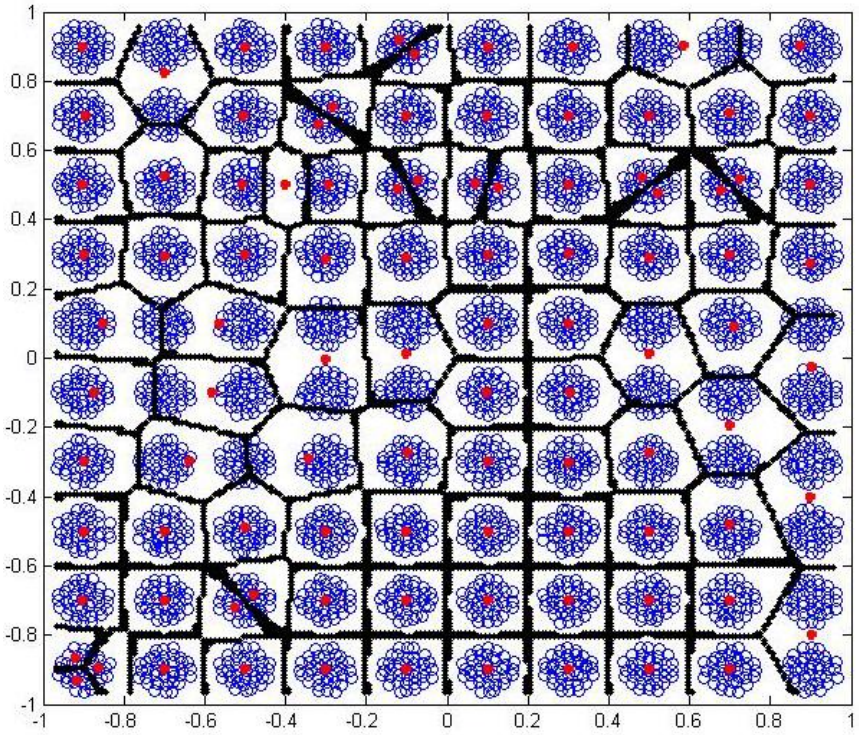
    adapt $\vec{\mu}^i = \vec{\mu}^i + \eta * (\vec{x}^j - \vec{\mu}^i)$



Vector quantization

Vector quantization can be used for streaming data but it has the same problems:

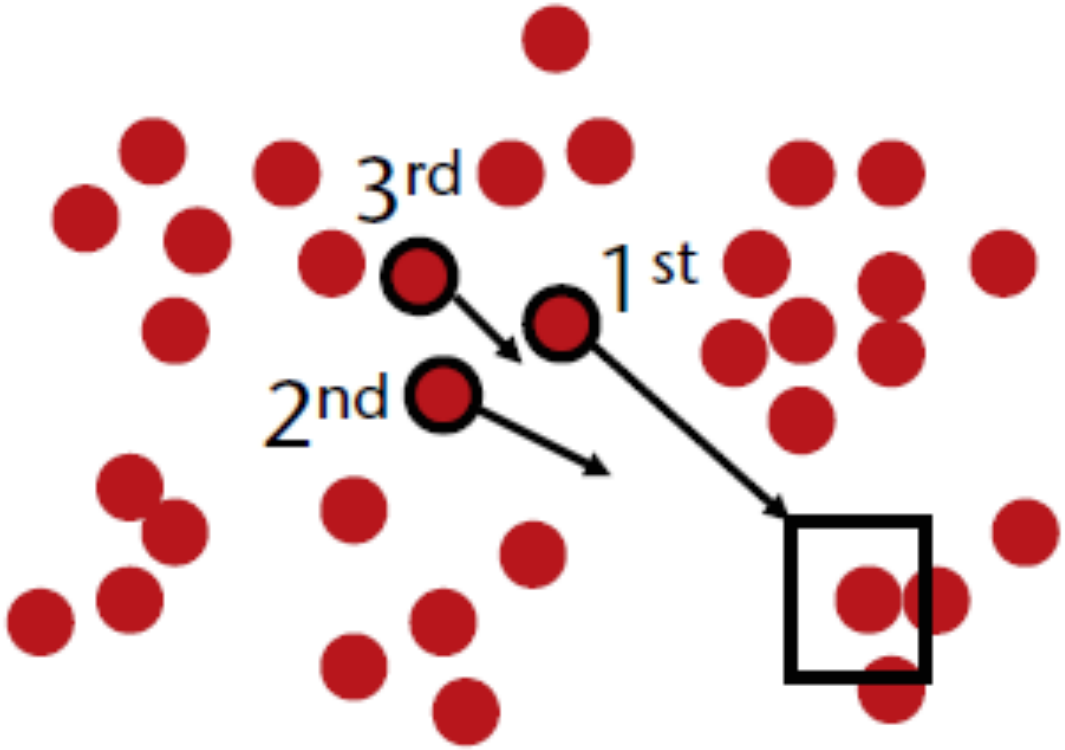
- dead units
- local optima
- multimodal data



VQ and a checkerboard data set

Vector quantization: Neural Gas

adapts all prototypes (in VQ only winner is updated), given a data point:



Neural Gas

init $\vec{\mu}^i$

repeat

choose $\vec{x}^j$

determine the distance $||\vec{\mu}^i - \vec{x}^j||^2$

assign ranks $r(\vec{\mu}^i, \vec{x}^j)$ according to their distances to x^j

adapt $\vec{\mu}^i := \vec{\mu}^i + \eta * e^{\left(-\frac{r(\vec{\mu}^i, \vec{x}^j)}{\sigma^2}\right)}(\vec{x}^j - \vec{\mu}^i)$

σ *is decreased*

Neural Gas

Goal of **k-means**: find prototypes $\vec{\mu}^1, \ldots, \vec{\mu}^k \in R^n$ such that they minimize the quantization error

Goal of **neural gas**: find prototypes $\vec{\mu}^1, \ldots, \vec{\mu}^k \in R^n$ such that they minimize the cost term

$$J_e = \frac{1}{2} \sum_{\vec{\mu}^i} \sum_{\vec{x}^j} \exp\left(-\frac{r\left(\vec{\mu}^i, \vec{x}^j\right)}{\sigma^2}\right) * ||\vec{\mu}^i - \vec{x}^j||^2$$

where the rank is used $r\left(\vec{\mu}^i, \vec{x}^j\right) := |\{\vec{\mu}^k \,|\, ||\vec{\mu}^k - \vec{x}^j|| < ||\vec{\mu}^i - \vec{x}^j||\}|$

Neural Gas

Neural Gas batch adaptation similar to the k-means

$$J_e = \frac{1}{2}\sum_{\vec{\mu}^i}\sum_{\vec{x}^j} \exp\left(-\frac{r\left(\vec{\mu}^i, \vec{x}^j\right)}{\sigma^2}\right) * ||\vec{x}^j - \vec{\mu}^i||^2$$

init $\vec{\mu}^i$

repeat

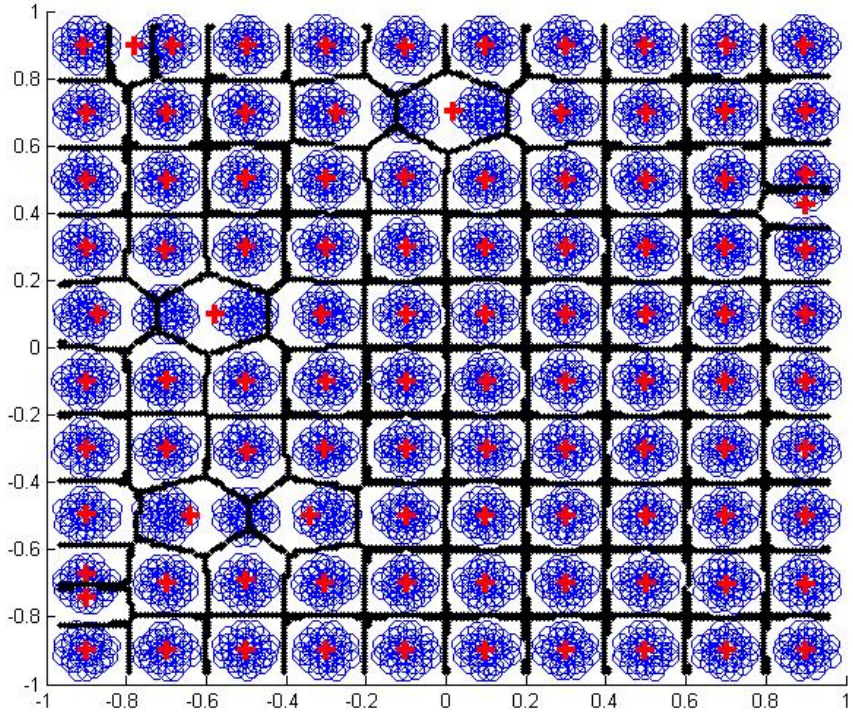
$$r\left(\vec{\mu}^i, \vec{x}^j\right) := |\{\vec{\mu}^k \,|\, ||\vec{x}^j - \vec{\mu}^k|| < ||\vec{x}^j - \vec{\mu}^i||\}|$$

$$\vec{\mu}^i := \left(\sum_{\vec{x}^j} \mathrm{e}^{(-r_{ij}/\sigma^2)} * \vec{x}^j\right) / \sum_{\vec{x}^j} \mathrm{e}^{(-r_{ij}/\sigma^2)}$$

σ *is decreased*

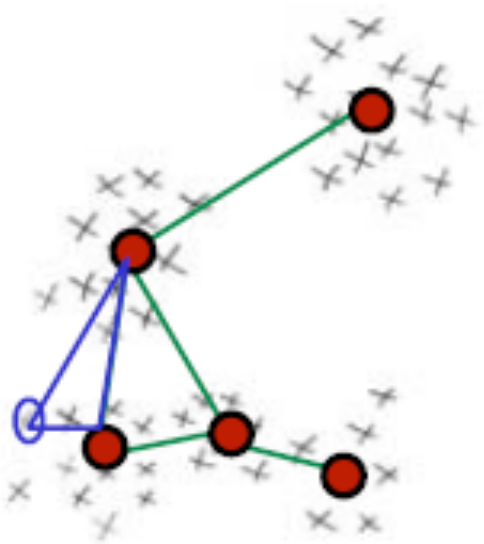
Neural Gas

Where the k-means failed, neural gas performed significantly better (but only after 600 epochs):



Neural Gas: connecting prototypes

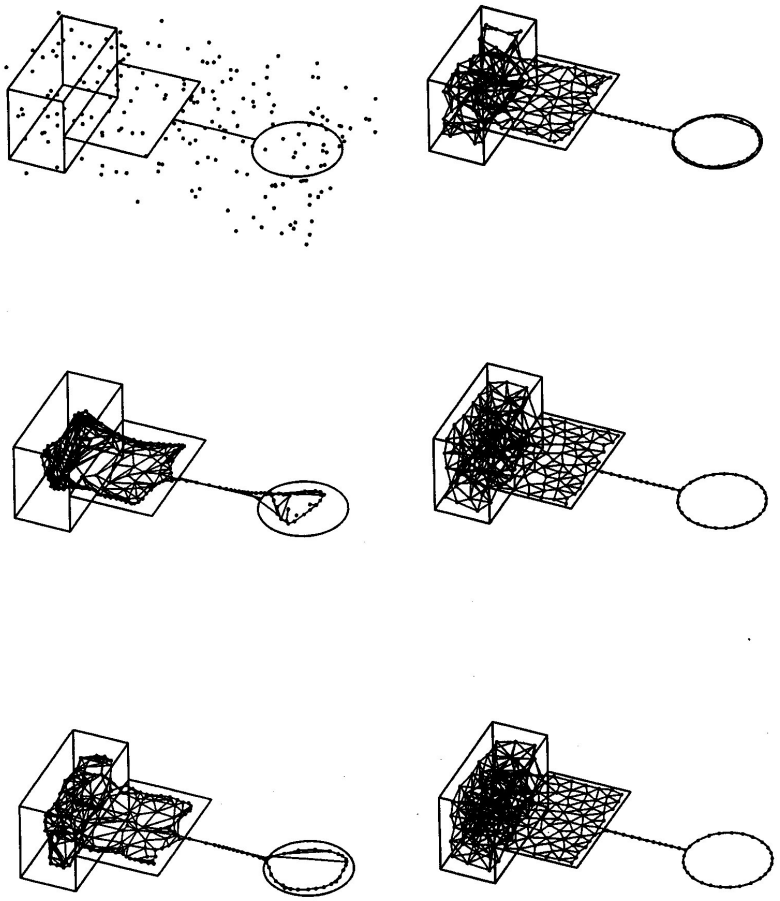
Connect two prototypes if they are are first and second winner for at least one point (for visualisation)



Neural Gas

Evolution of 200 prototypes

The data points fill a three-dimensional (parallelepiped), a two-dimensional (rectangle), and a one-dimensional (circle and connecting line) manifold



(T. Martinetz, K. Schulten,1991)

Neural Gas demo

http://demogng.de/js/demogng.html?model=NG

References

UCI repository of machine learning (http://archive.ics.uci.edu/ml/)

L. Bottou, Y. Bengio: *Convergence Properties of the K-Means Algorithms*. NIPS 1994: 585-592

M.Cottrell, B.Hammer, A.Hasenfuss, T.Villmann, *Batch and Median Neural Gas*, Neural Networks 19:762-771, 2006.

Jim Holmström: *Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG*, Uppsala Master's Thesis in Computer Science Examensarbete DV3, 2002-08-30

Catherine Cho, Sooyoung Kim , Jaewook Lee, Dae-Won Lee: *A tandem clustering process for multimodal datasets*, Elsevier, European Journal of Operational Research 168 (2006) 998–1008

Yiu-Ming Cheung: *k*-Means: A new generalized k-means clustering algorithm,* Elsevier, Pattern Recognition Letters 24 (2003) 2883–2893

T. Martinetz, K. Schulten, *A Neural Gas Network learns topologies,* in: T.Ko-honen et *al. (eds), Artificial Neural Networks, Elsevier* (1991)

Bernd Fritzke: The LBG-U Method for Vector Quantization – an Improvement over LBG Inspired from Neural Networks, *Neural Processing Letters 5: 35–45, 1997, Kluwer Academic Publishers*

Barbara Hammer, Marc Strickert and Thomas Villmann: *Learning Vector Quantization for Multimodal Data*

J. Jantzen: *Neurofuzzy Modelling*. Technical University of Denmark: Oersted-DTU, Tech report no 98-H-874 (nfmod), 1998. URL http://fuzzy.iau.dtu.dk/download/nfmod.pdf

J. Jantzen's presentation. URL: http://fuzzy.iau.dtu.dk/tutor/fcm/cluster.ppt

B. Fritzke, " A growing neural gas network learns topologies," Adv ances in Neural Inform ation Processing Systems 7 (NIPS'94), MIT Press, C ambridge, MA, pp.625-632, 1995.