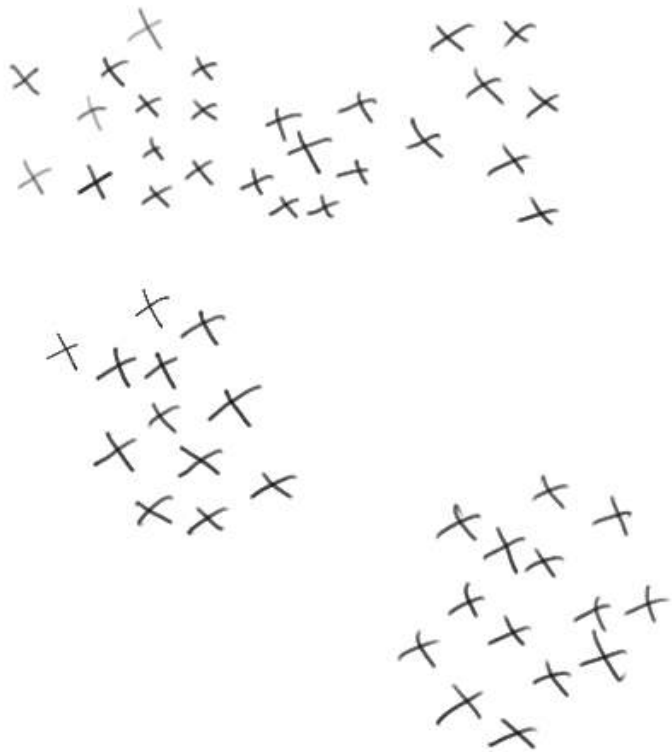
Clustering by proximity to prototypes (k-means clustering)

INTRODUCTION TO INTELLIGENT SYSTEMS 17/18

What is the goal of clustering?

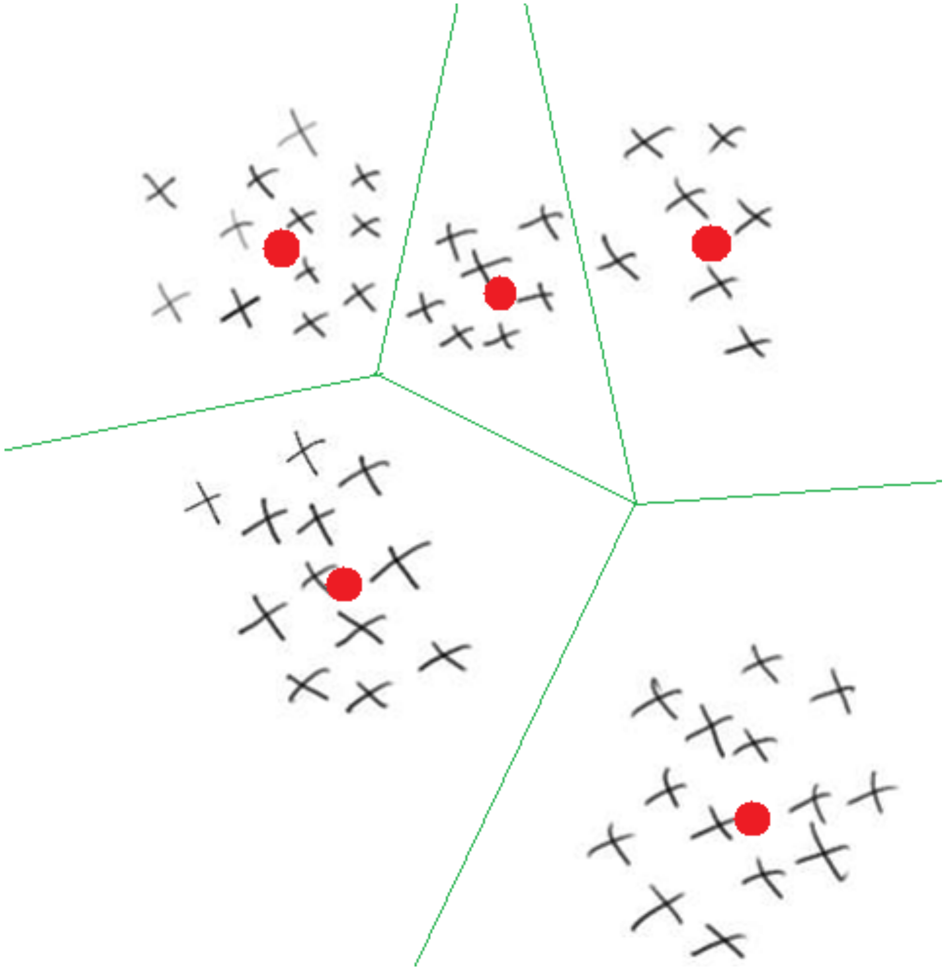


Division of a data set X into k disjoint subsets $C_1 \ldots C_k$ such that objects within each subset are similar and objects in different subsets are dissimilar

k-means clustering

Euclidean distance, prototype-based clustering:

assign a data point to the nearest prototype



k-means clustering

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

goal: find k prototypes μ^i

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(μ^i) – cluster (subset of X) associated with μ^i

(also called receptive field of μ^i)

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, … , C^k and $\mu^1, \mu^2, \ldots, \mu^k$
6. end

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:

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

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

- No guarantee that a global minimum is reached

Does Lloyd’s algorithm converge?

- The quantization error as a function of the iteration number MUST decrease monotonously
- If the quantization error shows oscillations (up and down) THERE MUST be a bug in your code

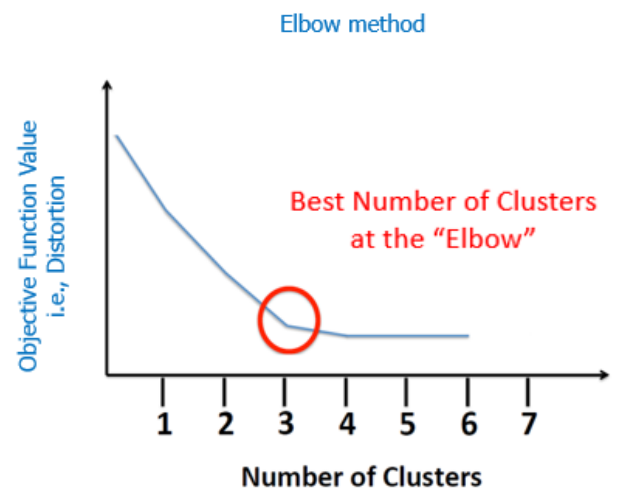
Initialization of k means

- MULTIPLE INITIALIZATIONS, e.g. take data points randomly
  - Run the k-means algorithm for different initializations and take the result for which the quantization error is minimum
- Run the k-means with a subset (randomly sampled) of the original data set. Use the means as initialization seeds for the k-means run on the complete data set

How to choose k?

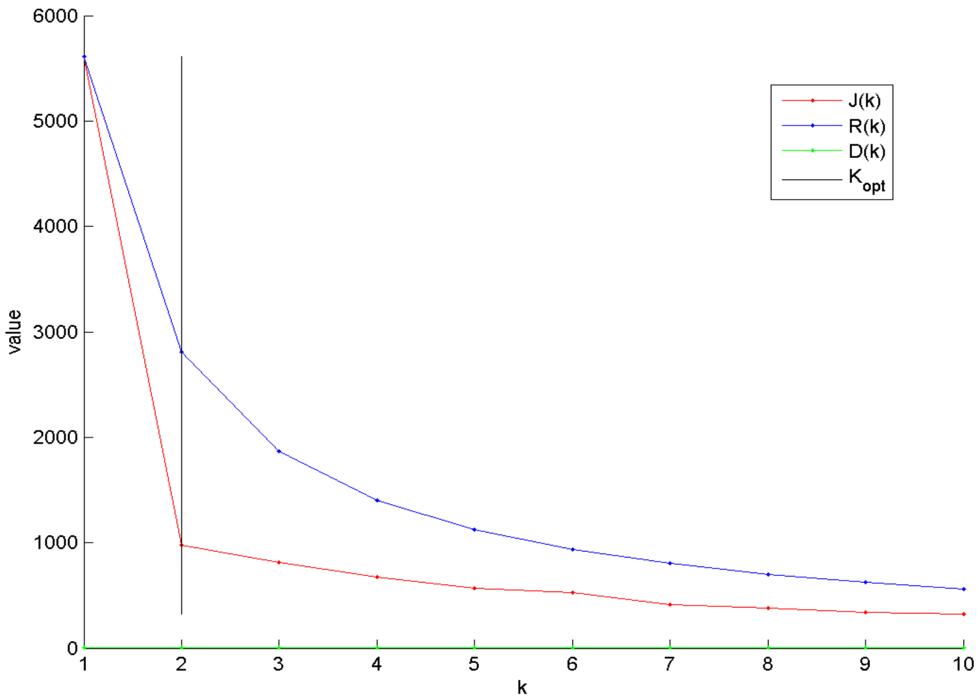
'ELBOW' METHOD (simplified):

1. Run the k-means algorithm for multiple values of k and for each value of k record the value of the quantization error upon convergence
2. Plot the reached quantization error as a function of k
3. If the plot shows an 'elbow' for a certain k, take that k



How to choose k? (2) (*gap statistics*)

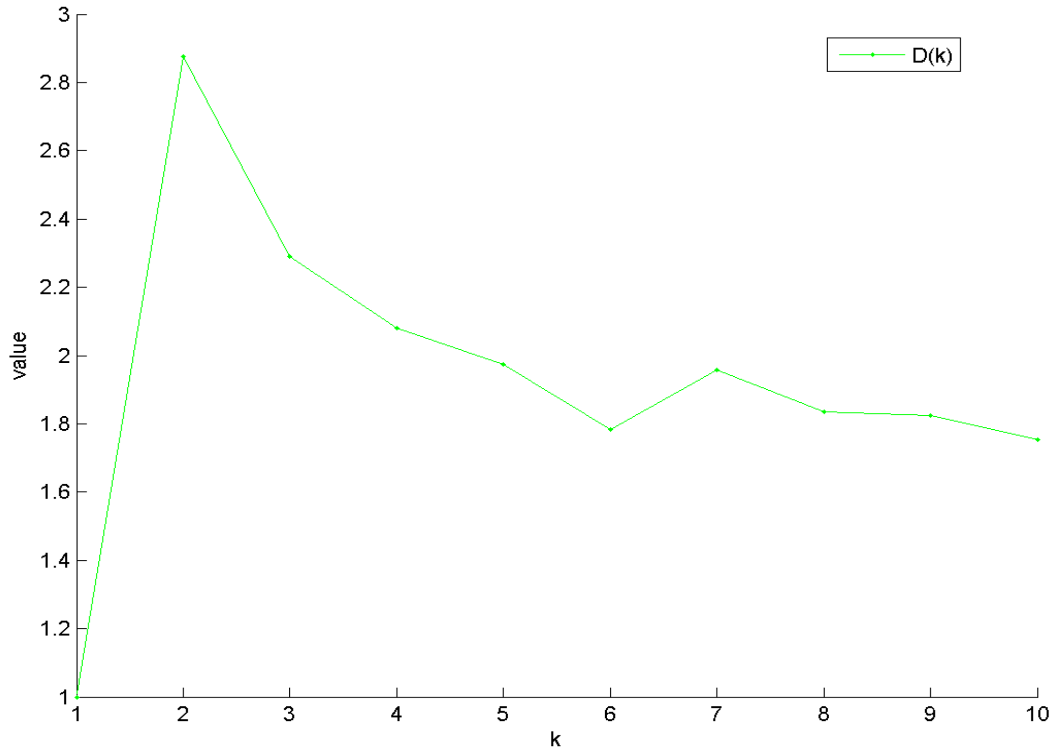
1. Compute the quantization error as a function of k: J(k)
2. Compute the quantization error R(k) for a uniformly distributed reference data set. ($R(k) \sim k^{-2/d}$, d – dimensionality)



How to choose k? (3)

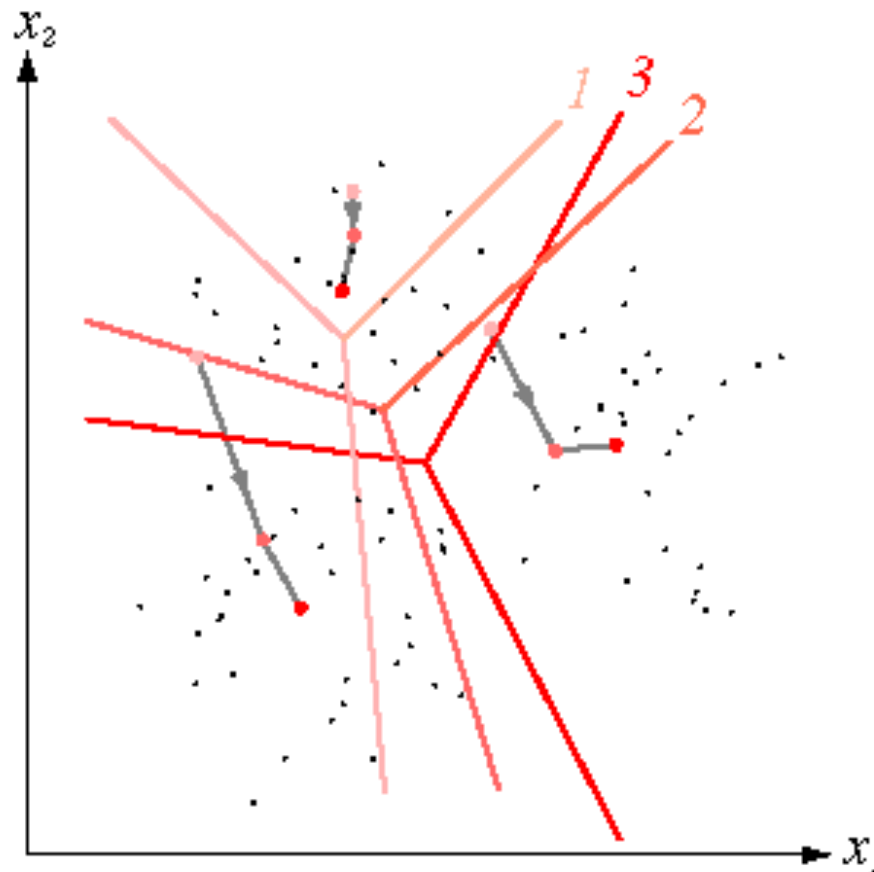
Find the maximum of the ratio D(k)=R(k)/J(k).

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

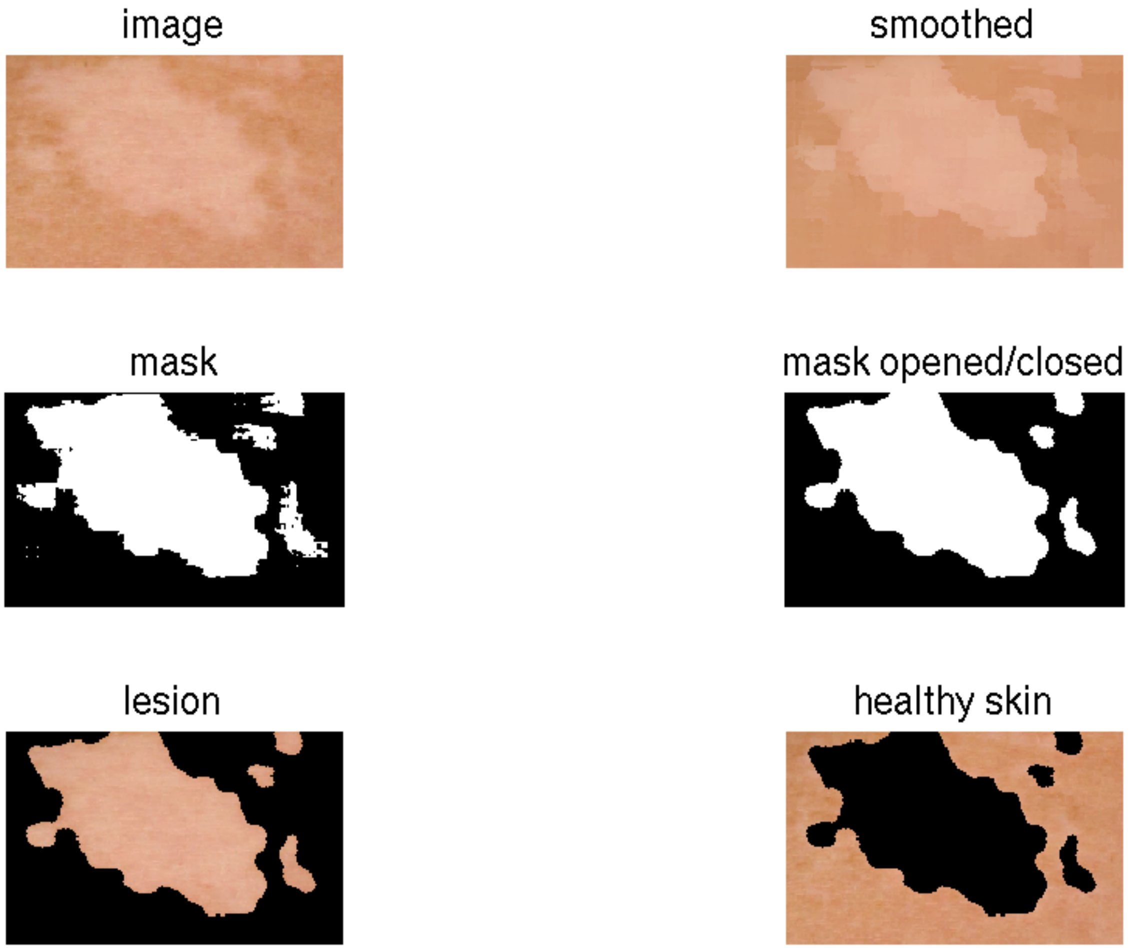


Example of k-means clustering

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



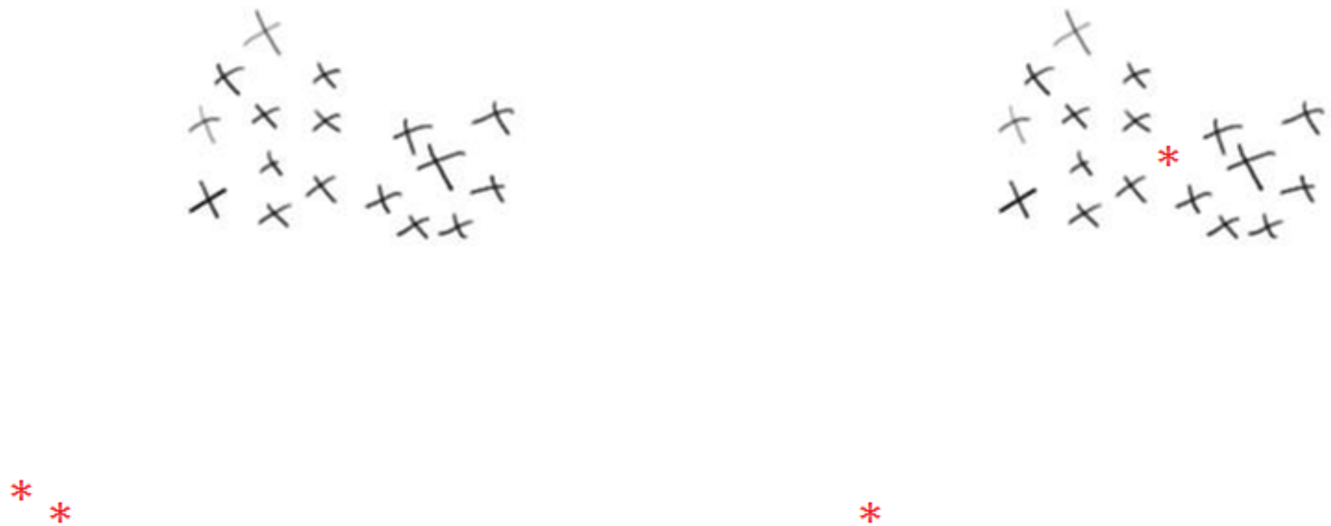
from Duda, Hart, Stork (2001)
Pattern classification



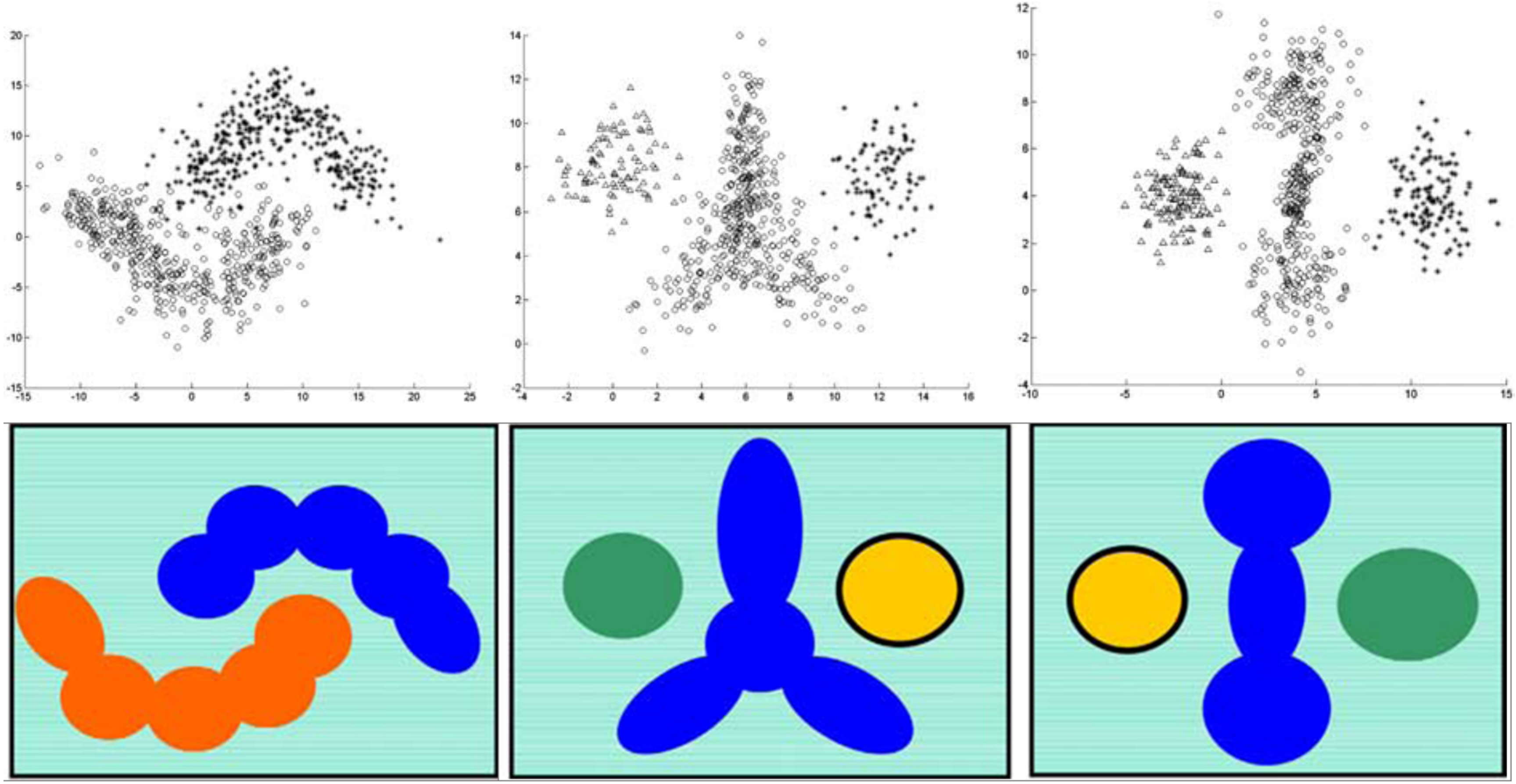
Example of 2-means clustering: a skin image is segmented in two regions of lesion and healthy skin by grouping pixels in two clusters according to their color (result shown in image mask)

Problems with k-means clustering: dead units (poor initialization)

If some prototypes are initialized far away from the input data, no data points are assigned to them and they are never updated (dead prototypes)



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