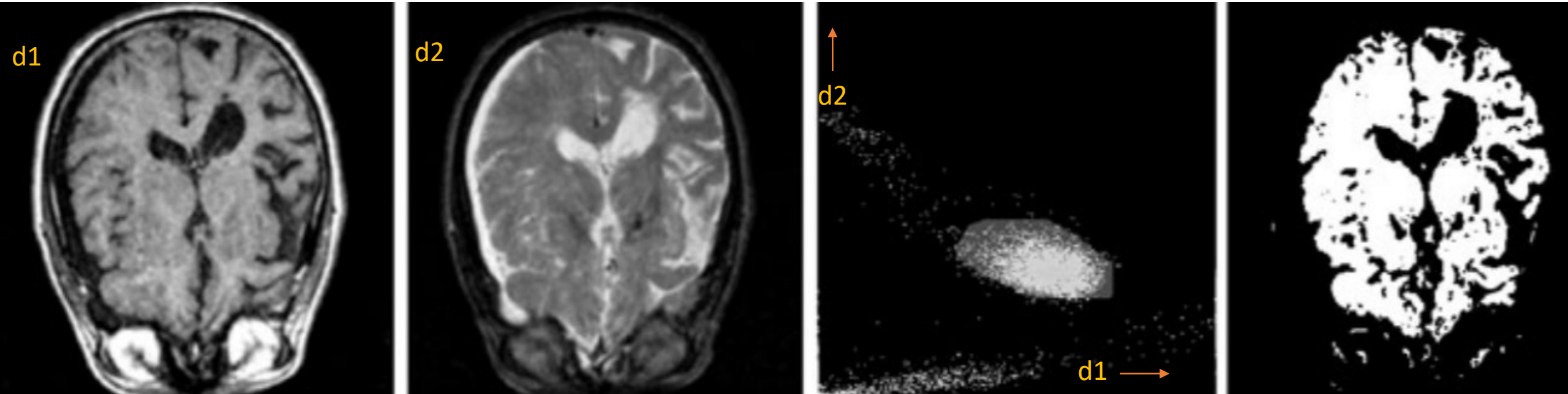
CS463/516
Medical Imaging

Lecture 15

Segmentation in feature space continued..

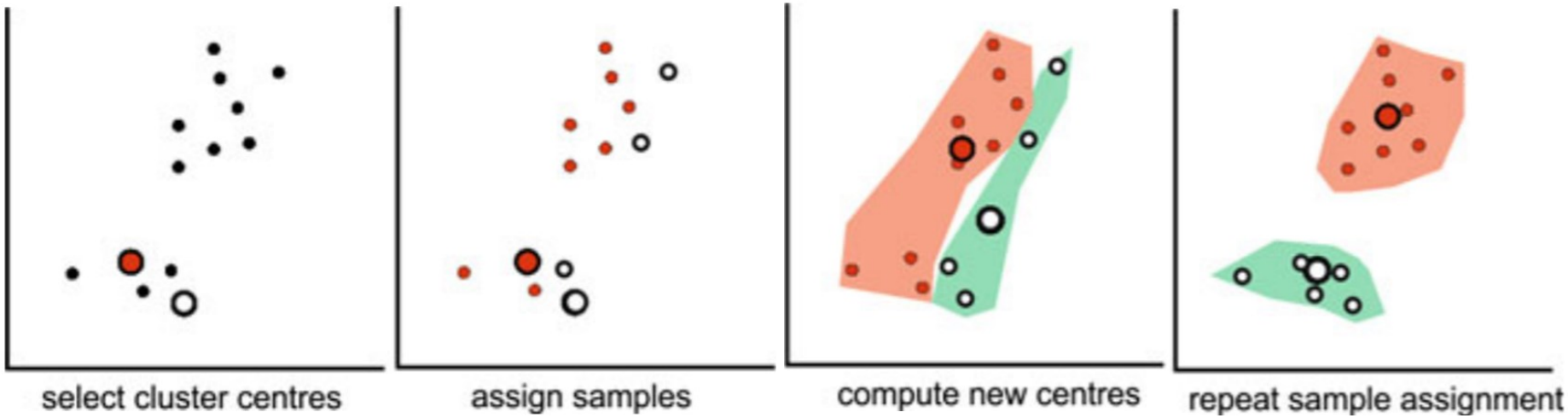
Clustering in feature space

- Clustering assumes that scene elements from same object have more similar features than those that belong to other objects
- Clustering for segmentation works in low-dimensional feature space
  - Simplest way is interactively – plot the distribution and user delineates clusters
- Example: low dimensional feature space (d=2)
  - Can simply plot one feature as function of other feature, and delineate point cloud
  - Highly subjective, and more for just getting a ‘feel’ for the data



Partitional clustering

- Objective: find cluster centers $CC = \{c_1, \dots c_K\}$ in feature space such that distance of all samples to their center is minimal:
- $CC_{min} = argmin_{CC} d_{CC}(\boldsymbol{f}) = argmin_{CC} \sum_{i=1}^{M} \|\boldsymbol{f}_i - \boldsymbol{c}(\boldsymbol{f}_i)\|$
  - Where $\boldsymbol{c}(\boldsymbol{f}_i)$ delivers cluster center $\boldsymbol{c}_k$ that is closest to $\boldsymbol{f}_i$
- Heuristic strategy for finding optimal cluster centers:
  - 1) cluster centers chosen randomly,
  - 2) samples assigned to cluster that minimizes d_{CC}
  - 3) new center $\boldsymbol{c}_c$ defined for each cluster c that minimizes $\sum_{f_i \in c_c} \|\boldsymbol{f}_i - \boldsymbol{c}_k\|$
  - 4) repeat until cluster centers no longer change

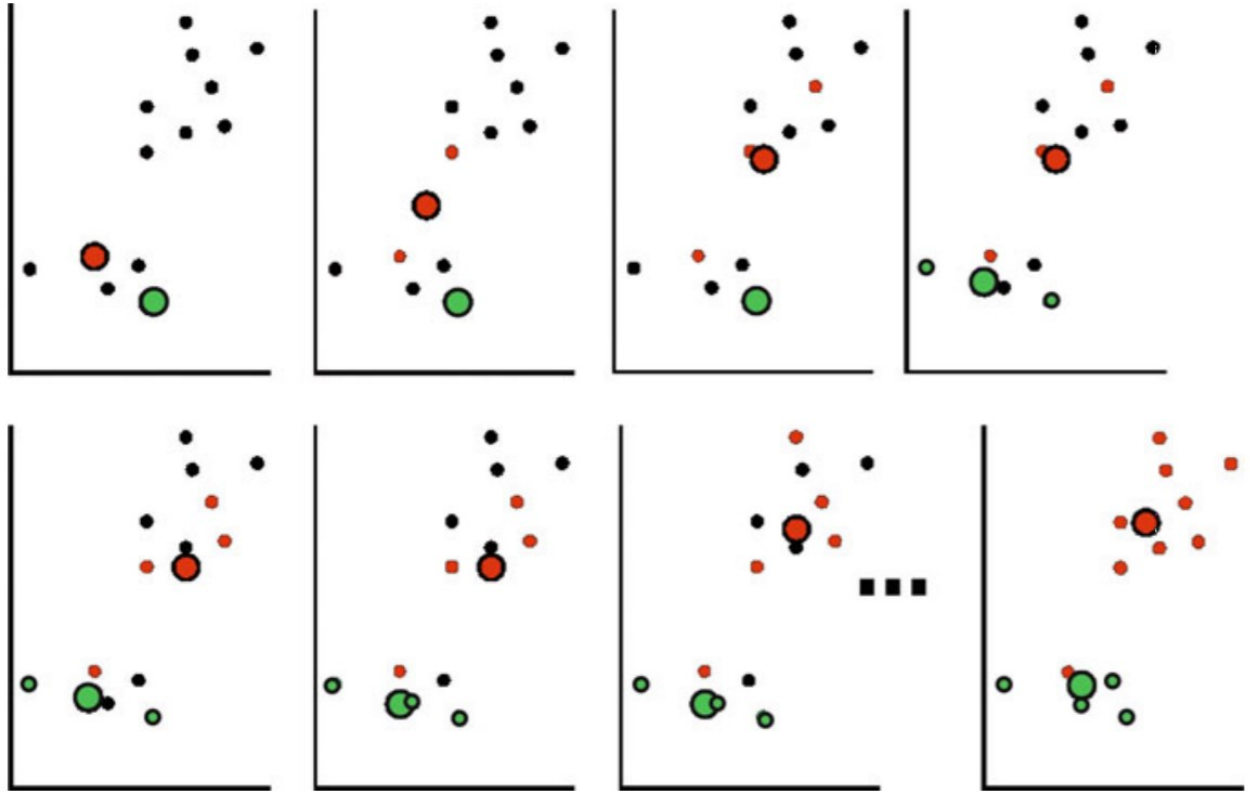


- Process is heuristic, since assignment to clusters (step 2) and determination of new cluster centers (step 3) are optimized separately.
- May not produce optimal result and is not guaranteed to converge

K-means clustering

- Same objective as partitional clustering, but always terminates, requires just 2 iterations.
  - Instead of assigning all samples to the current estimate for cluster centers before computing a new cluster center, the two objectives are interleaved by computing new cluster centers after each assignment of a feature to a cluster

First pass of k-means clustering algorithm. Samples are selected at random and assigned to nearest cluster center. After each assignment, cluster center locations are updated.

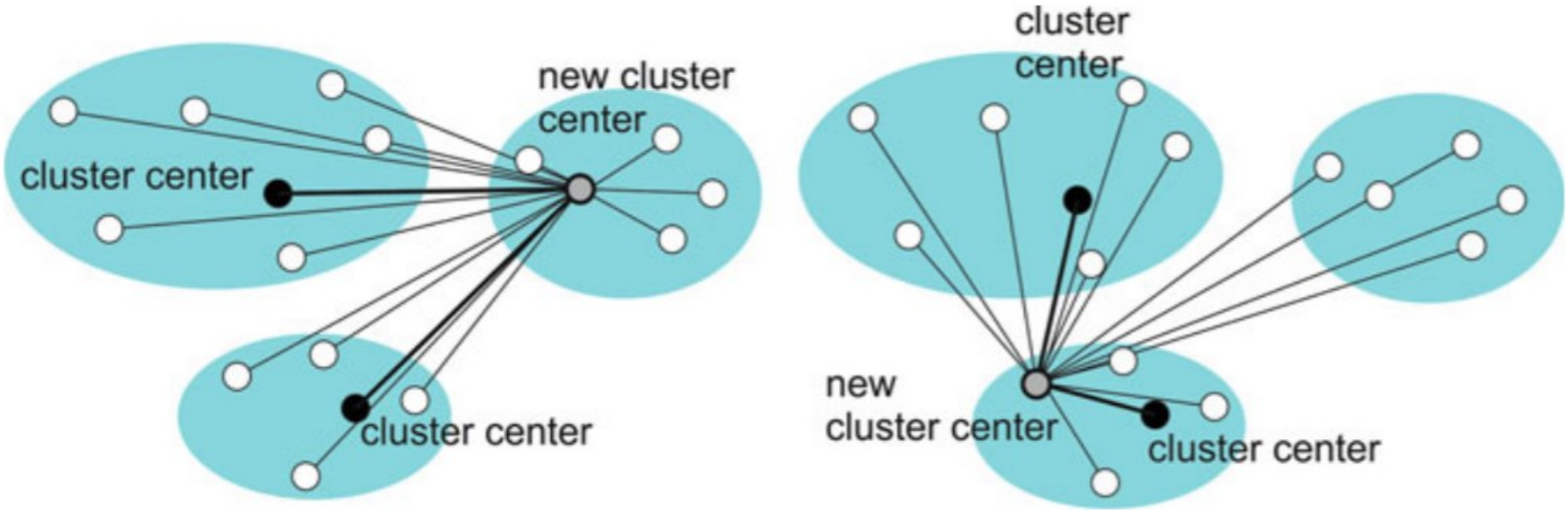


Second phase of k-means clustering. Given estimates for cluster centers from first step, all samples are analyzed according to their distance to nearest cluster center, and potentially re-assigned.

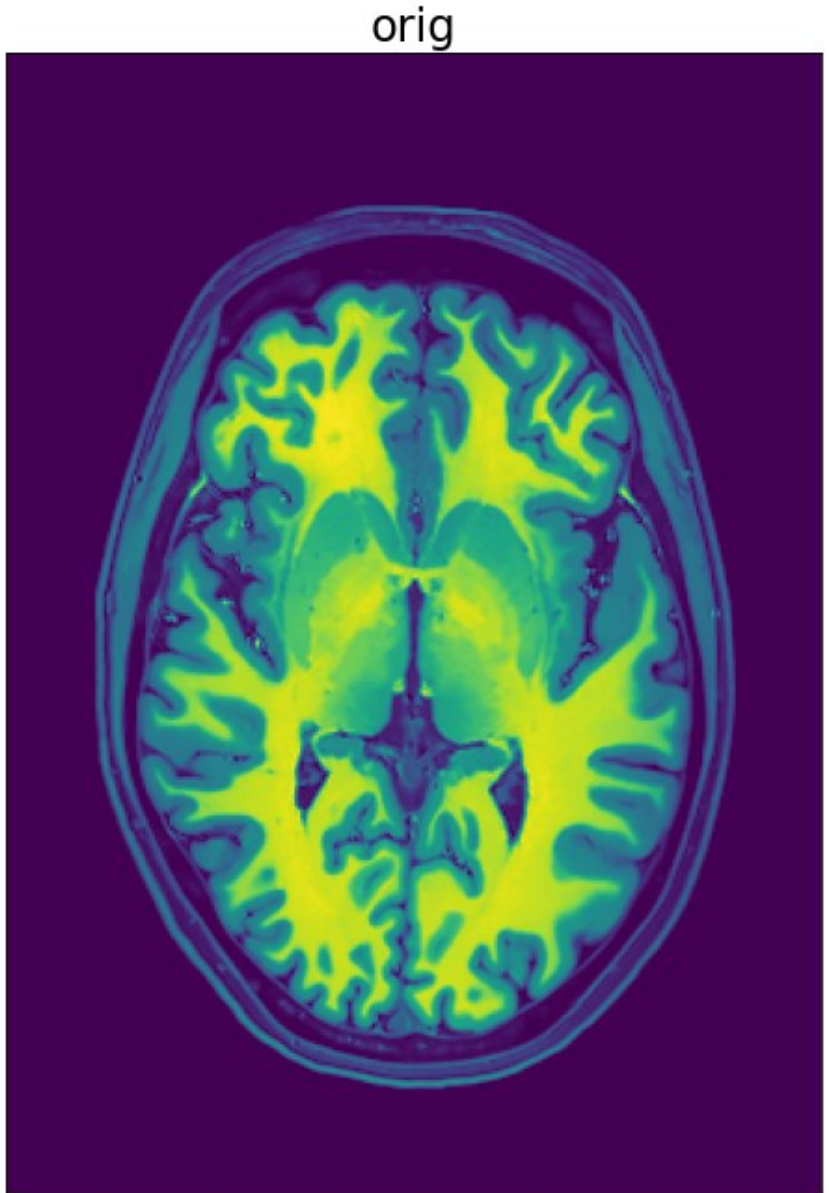
need to know initial number of clusters and good initialization for cluster centers to employ k-means successfully for segmentation. Usually, use more clusters than required segments, merge later.

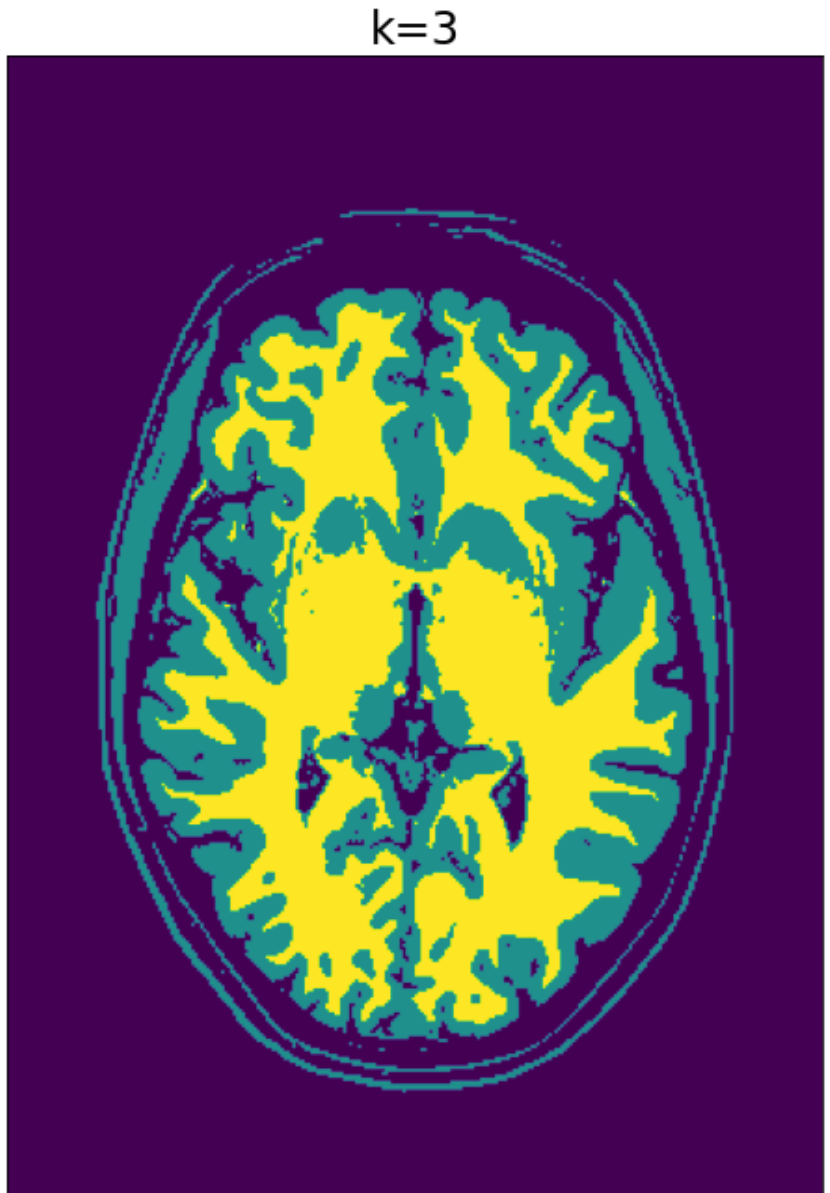
Cluster center initialization for k-means

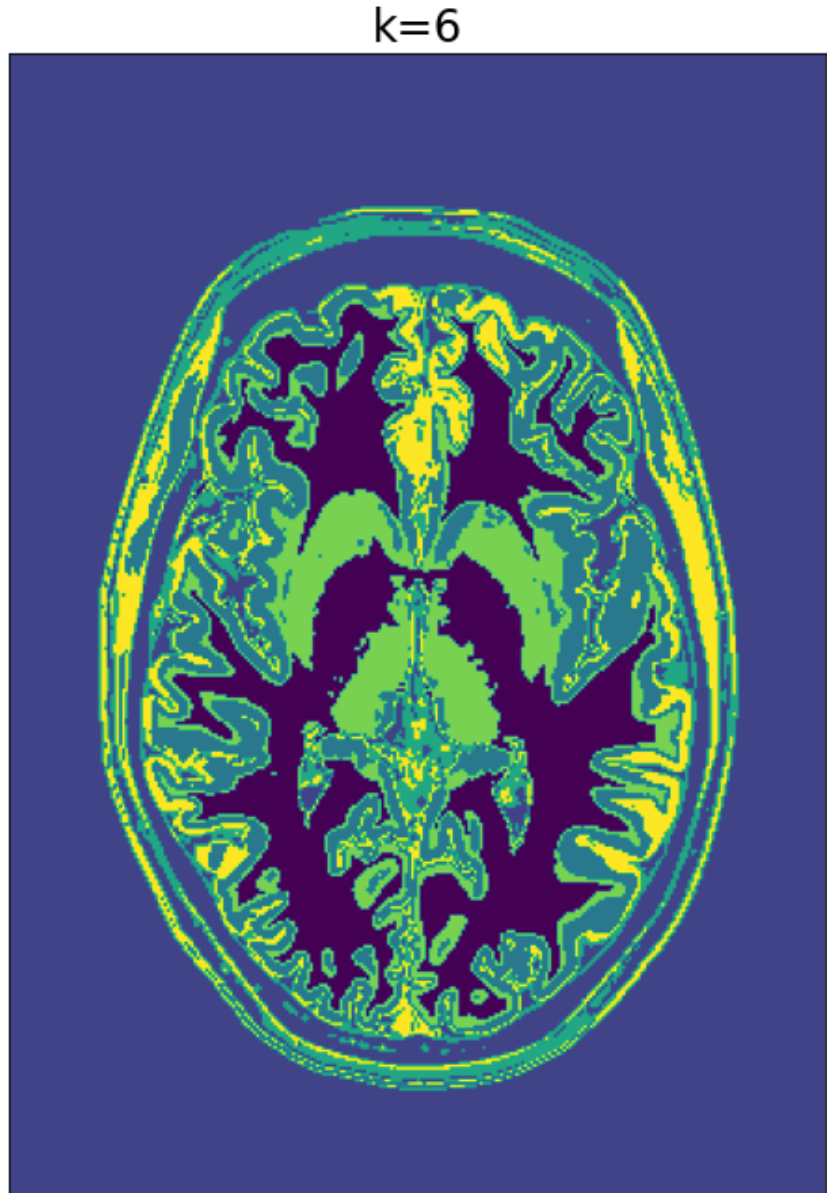
- Simplest way: separate the feature space into K compartments at random, and compute centers of gravity from samples in each compartment
- Kaufman-Rousseeuw method take *distribution characteristics* into account:
  - Starts with sample locations, but selects samples according to diversity measure
    - Diversity measure D rates distance of newly selected cluster center $\boldsymbol{C}_{new}$ to all already-selected cluster centers $\boldsymbol{C}_1 \dots, \boldsymbol{C}_k, k < K$ with respect to distances of $\boldsymbol{C}_{new}$ to all non-selected samples $\boldsymbol{s}_i$ as follows:
    - $d_{min}^{c} = \min_k(\|\boldsymbol{C}_{new} - \boldsymbol{C}_k\|)$, $D = \max_i(0, d_{min}^{c} - \|\boldsymbol{C}_{new} - \boldsymbol{s}_i\|)$
    - Iteratively, a sample is selected as a new cluster center that maximizes the diversity measure until K clusters have been selected



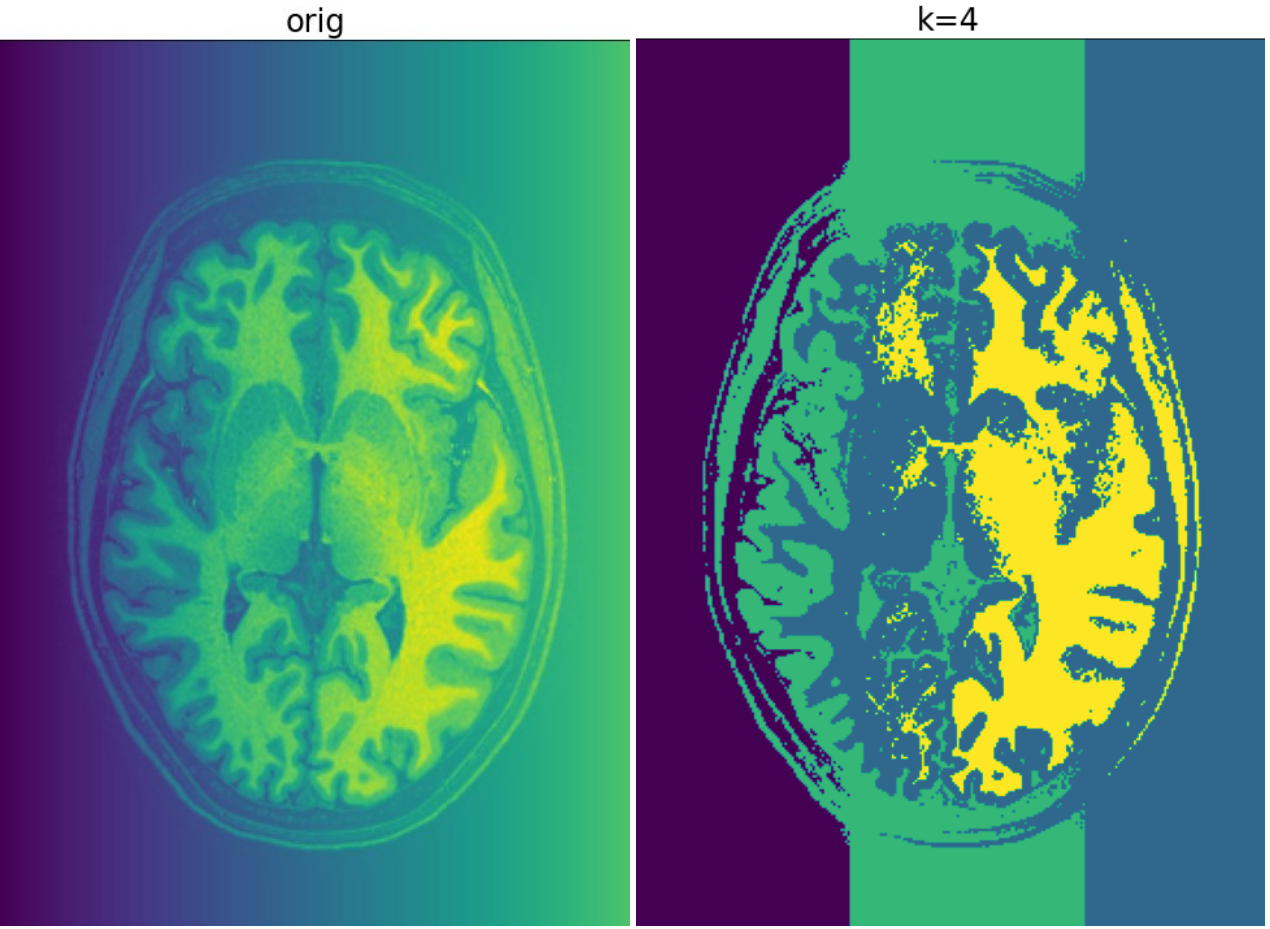
K-means clustering on denoised T1



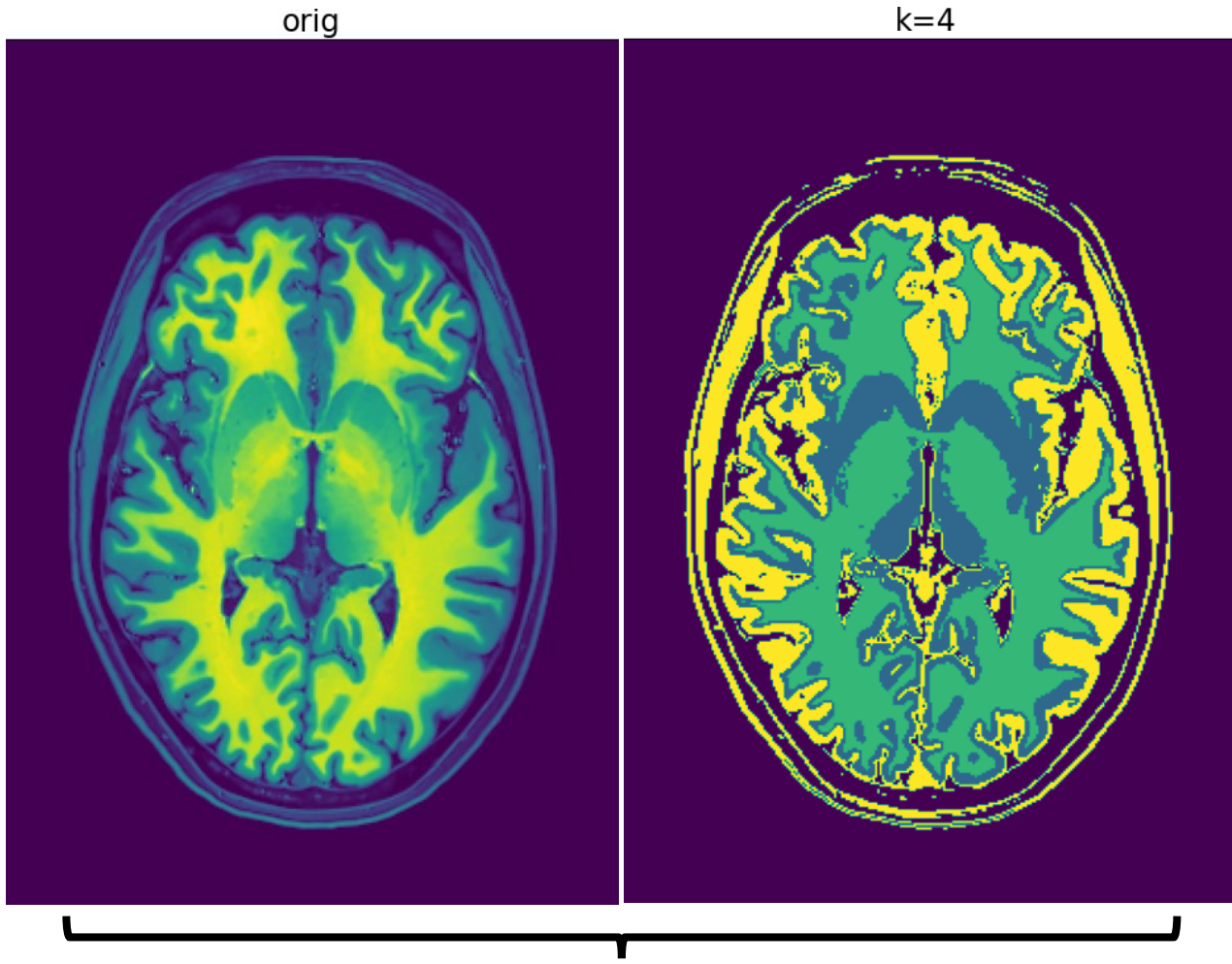




The importance of pre-processing



orig has some random noise + bias in x-direction
Yields poor segmentation using k-means



orig is denoised with no bias
Yields somewhat useful segmentation using k-means

Mean-shift clustering

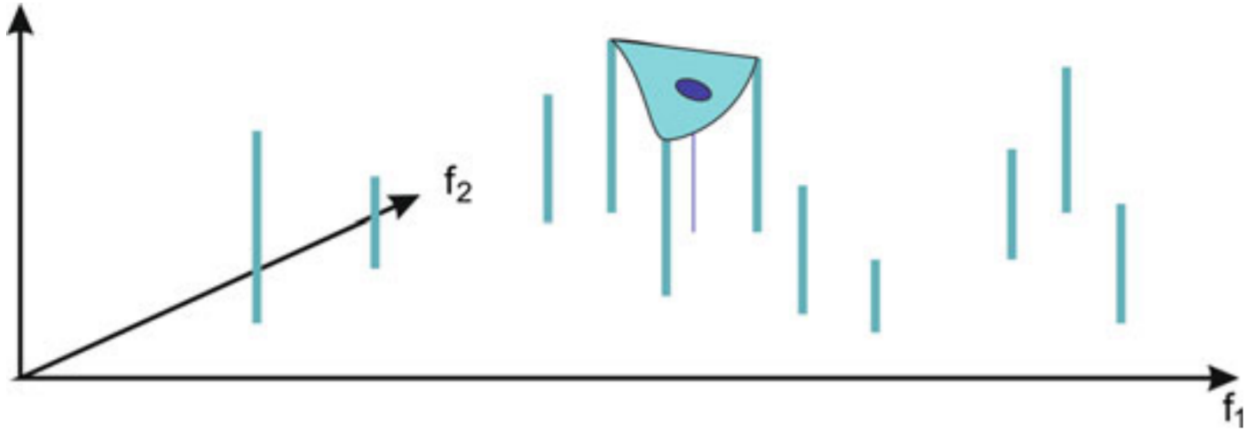
- Mean shift clustering does not require the number of clusters to be known
- Mean shift clustering attempts to find all possible cluster centers in feature space
- Samples are assumed to stem from a density function, which is a mixture of an unknown number of probability density functions (pdfs)
- Each probability function describes probability of a sample belonging to a cluster
- ‘ideal clustering’ would identify the probability functions of the mixture model, and then assign cluster membership based on this probability
- Determining parameters of an unknown number of probability functions of an unknown type difficult if not impossible

Mean-shift clustering

- Mean shift clustering uses heuristics to arrive at feasible solution, assumes:
  - 1) pdfs of mixture model have only one maximum, which represents mean of that function
  - 2) combining pdfs in mixture model preserves each pdf's local maxima
    - Local maxima of mixture function represents means of underlying pdfs
  - 3) local minima of mixture model segment feature space, such that each segment contains only a single local maxima
- Finding local maxima under these assumptions produces clusters in feature space
- **Mean shift algorithm:**
  - `1) for each location in feature space, shift marker towards next local maximum using gradient ascent`
  - `2) if local maximum is found with no cluster label, label cluster`
  - `3) apply cluster label to location from where local maximum was found`

Gradient ascent

- Gradient ascent involves computing gradient of density function
- Approximate gradient of sampled pdf using kernel window estimator (a):
- $k(\boldsymbol{x}) = k(\|\boldsymbol{x}\|^2)$
  - Where k is a one-dimensional function on distance such as Gaussian $k(\boldsymbol{x}) = \exp(-\frac{\boldsymbol{x}^2}{2})$
- Kernel density estimation for location $\boldsymbol{x}$ in d-dimensional feature space with samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is then $f_{h,K}(\boldsymbol{x}) = \frac{c_{K,d}}{Nh^d} \sum_{i=1}^{N} k(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h})$
  - Where h determines kernel width and $c_{K,d}$ is a normalizing constant



a) The gradient at some location in feature space is approximated by interpolation over a predefined neighborhood using a suitable kernel function

- Kernel density estimation for location $\boldsymbol{x}$: $f_{h,K}(\boldsymbol{x}) = \frac{c_{K,d}}{Nh^d}\sum_{i=1}^{N} k(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h})$
- If k is differentiable, and g is derivative of $-k$, gradient is:

$$\nabla f_{h,K}(\boldsymbol{x}) = \frac{c_{K,d}}{Nh^{d+2}}\sum_{i=1}^{N} g(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h})(\frac{\sum_{i=1}^{N} x_i g\left(\frac{\|x-x_i\|^2}{h}\right)}{\sum_{i=1}^{N} g\left(\frac{\|x-x_i\|^2}{h}\right)} - \boldsymbol{x})$$

  - first part is a kernel estimator using derivative g instead of original function k
  - second part represents error between actual position $\boldsymbol{x}$ and its estimate by kernel
  - **This is called the *mean shift* $m_{h,G(x)}$ *of* $\boldsymbol{x}$**
- Separate two parts by rearranging, introducing new normalization constant $c_{G,d}$ for g:

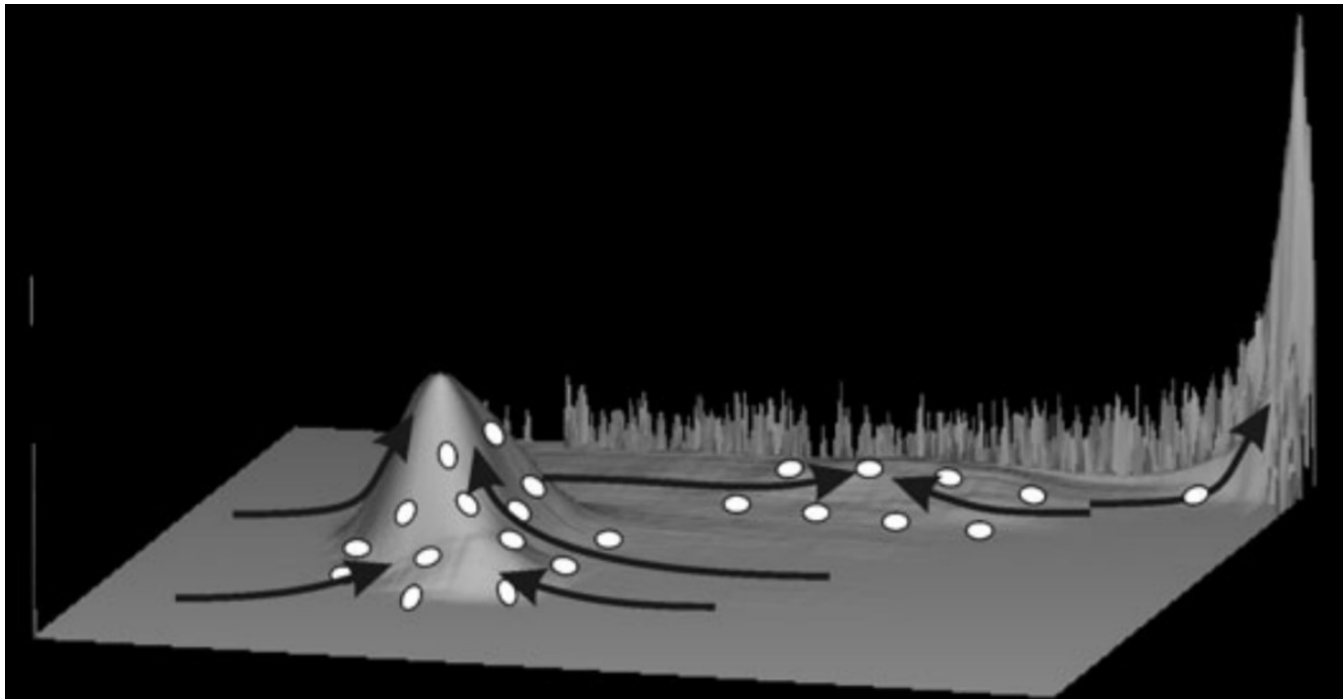
$$\nabla f_{h,K}(\boldsymbol{x}) = \frac{c_{G,d}}{Nh^d}\sum_{i=1}^{N} g\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h}\right)\frac{c_{K,d}}{c_{G,d}h^2}(\frac{\sum_{i=1}^{N} x_i g\left(\frac{\|x-x_i\|^2}{h}\right)}{\sum_{i=1}^{N} g\left(\frac{\|x-x_i\|^2}{h}\right)} - \boldsymbol{x})$$

- Kernel estimator is then: $\nabla f_{h,G}(\boldsymbol{x}) = \frac{c_{G,d}}{Nh^d}\sum_{i=1}^{N} g\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h}\right)$
- And mean shift is: $m_{h,G}(\boldsymbol{x}) = \frac{c_{K,d}}{c_{G,d}h^2}(\frac{\sum_{i=1}^{N} g\left(\frac{\|x-x_i\|^2}{h}\right)}{\sum_{i=1}^{N} x_i g\left(\frac{\|x-x_i\|^2}{h}\right)} - \boldsymbol{x})$

Mean shift clustering

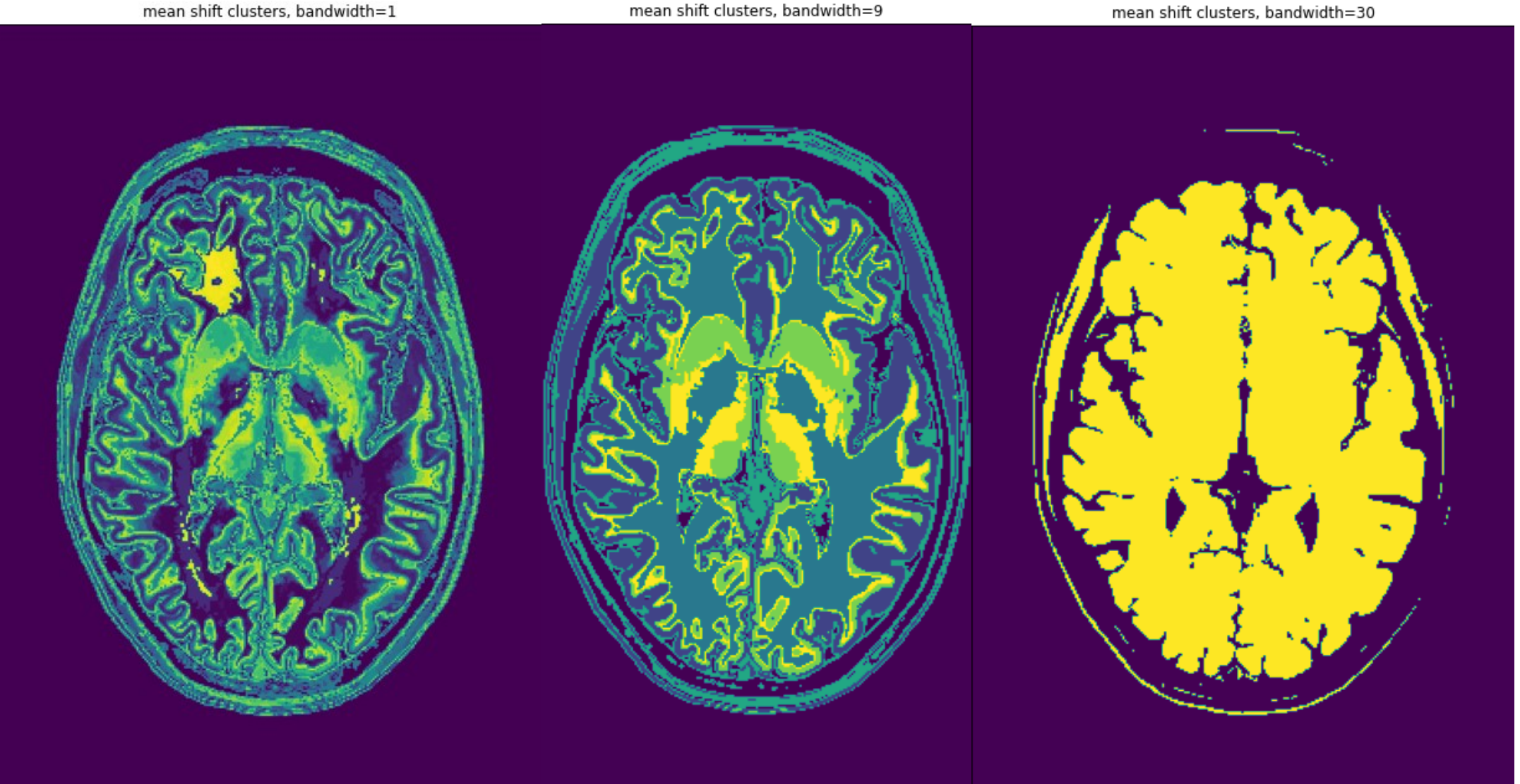
- Starting at some initial position $\boldsymbol{x}$ the algorithm proceeds by repeatedly changing position by the mean shift until the gradient length is zero and a local maximum is reached
- If this is done for all sample points, and cluster labels are created and assigned as described previously, results in assigning each location in feature space to its corresponding mode
- **Advantage** to mean shift: parameter free – determination of number of clusters does not depend on user
- **Disadvantage**: often results in oversegmented images, since every local maximum in feature space forms its own cluster

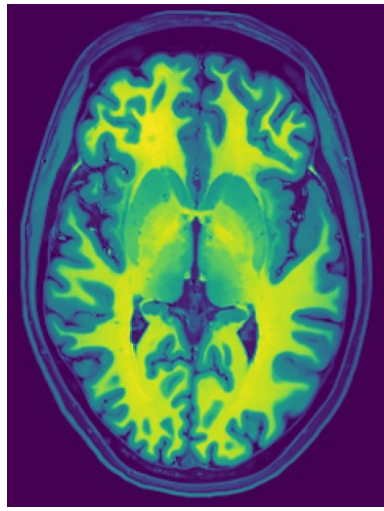
$$m_{h,G}(\boldsymbol{x}) = \frac{c_{K,d}}{c_{G,d}h^2}\left(\frac{\sum_{i=1}^{N} g\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h}\right)}{\sum_{i=1}^{N} x_i g\left(\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{h}\right)} - \boldsymbol{x}\right)$$



Mean shift clustering

Bandwidth used in the RBF kernel determines (roughly) how many clusters
Higher bandwidth ⇒ fewer clusters

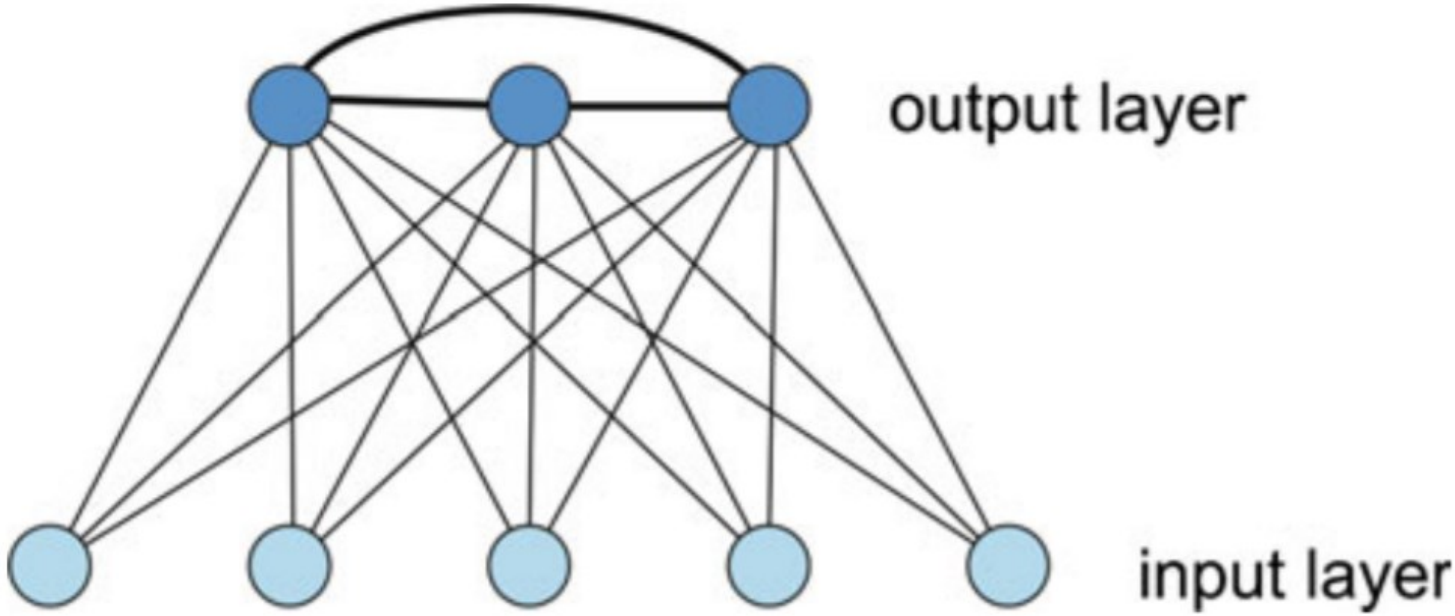




Kohonen's self organizing maps (SOM)

- Neural networks may be used for clustering, by association of a feature vector to a model cluster vector using a similarity measure
- Kohonen's SOM clusters data based on similarity between feature vectors
- Also, SOM attempts to find underlying structure between different clusters
  - Structure information can be used to guide clustering procedure

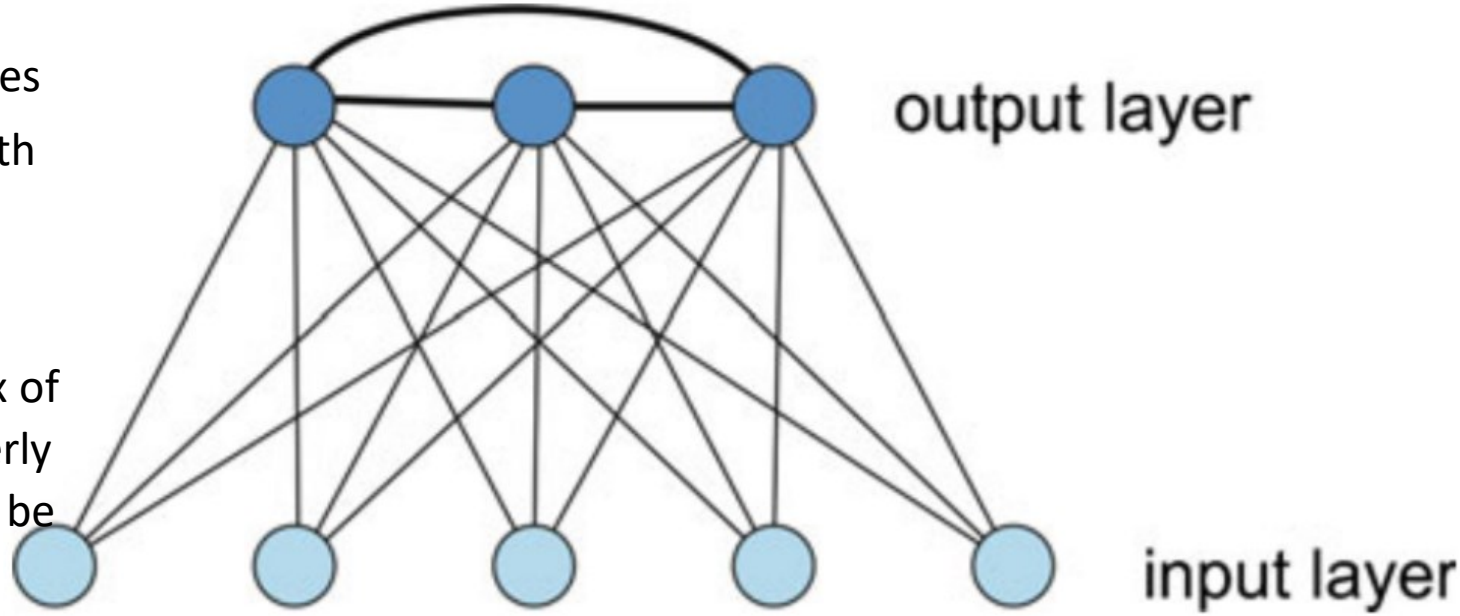
- Consists of single output layer fully connected to all nodes in input layer
- Each node in input layer represents a feature in feature space
- Each node in output layer corresponds to a potential cluster
- Called 'self-organizing' because learns feature patterns and organization of feature distribution without supervision



Kohonen's self organizing maps (SOM)

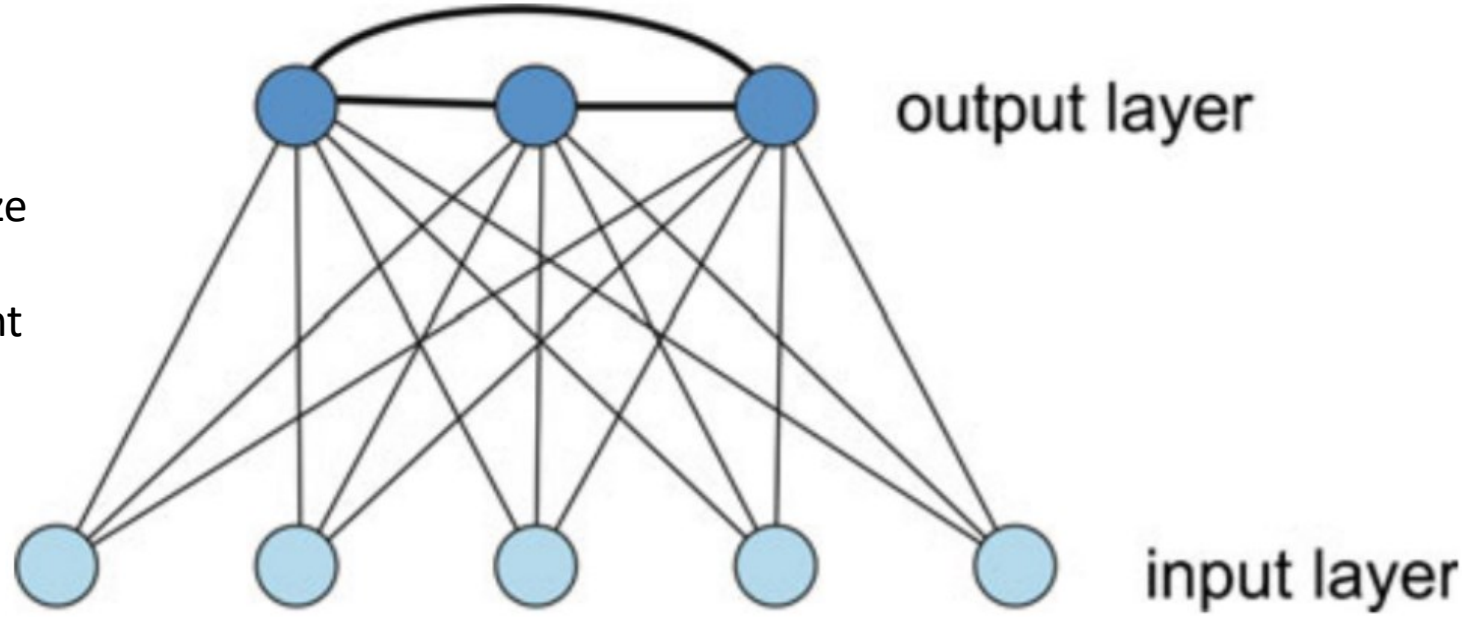
- Feature patterns represented by weight vectors $\boldsymbol{w}_j = \left(w_{1j}, w_{2j} \ \ldots w_{Nj}\right)^T$ leading to output nodes j
- Organization of feature distribution represented by connecting output nodes to a 1d or 2d map, and letting output node j influence adjacent nodes in neighborhood $N_\delta(j)$
- In simplest form, neighborhood between output nodes is $N_\delta(j) = 0$ (no influence between output nodes), in this case network is a simple *association network*

- $N \cdot C$ edges connect input nodes with output layer, with weights w_{ij} (N features, C clusters)
- Activation signal $f_j(\boldsymbol{f})$ at output node j is norm of difference between feature vector and vector of edges $\boldsymbol{w}_j = \left(w_{1j}, w_{2j}, \ldots w_{NJ}\right)^T$ connecting input nodes with output node:
- $f_j(\boldsymbol{f}) = \|\boldsymbol{f} - \boldsymbol{w}_j\| = \sqrt{\sum_{i=1}^{N}\left(f_i - w_{ij}\right)^2}$
- Output of association network is not f_j but the index of node with highest value, if weights w_{ij} are set properly so that $\boldsymbol{w}_j$ is the center of a cluster $\boldsymbol{c}_j$, the result will be the index of that cluster



Kohonen's self organizing maps (SOM)

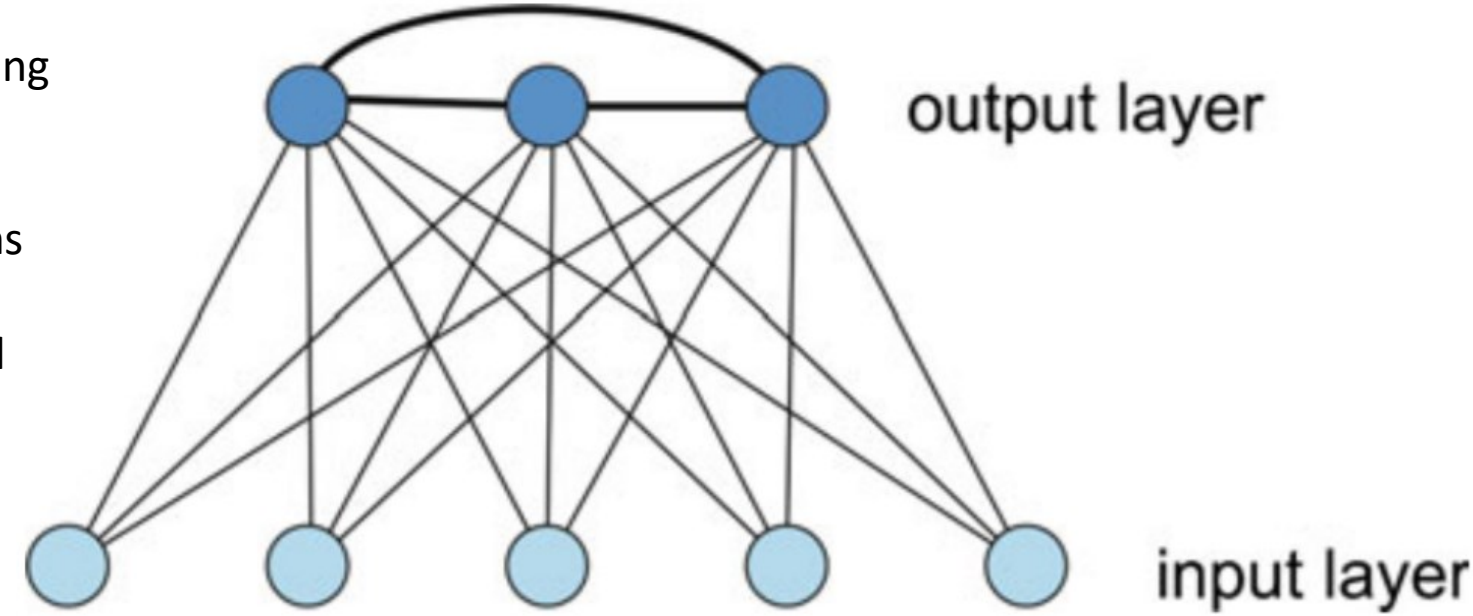
- Training seeks to move cluster centers $\boldsymbol{w}_j$ closer to the feature vectors
- Training is unsupervised, so called *reinforcement* or *competitive* learning
- Weights initially set to random values, and sample feature vectors fed to network
- Winning neuron in output layer is determined
- Weights leading to winning neuron adapted so it becomes more similar to feature vector: $\boldsymbol{w}_j^{n+1} = \boldsymbol{w}_j^n + \alpha(\boldsymbol{f} - \boldsymbol{w}_j)$
  - Where α is the learning rate, $\alpha < 1$
  - α must be < 1 in order to let network memorize previous activations
  - $\alpha = 1$ would cause perfect adaptation of weight vector $\boldsymbol{w}_j$ to the pattern presented by $\boldsymbol{f}$, and network "forgets" enforcements due to other vectors presented earlier
  - $\alpha = 0.25$ or $\alpha = 0.5$ are common



Kohonen's self organizing maps (SOM)

- Original variant introduced by Kohonen requires normalization of weights and gradually reduces angle between $\boldsymbol{w}_j$ and $\boldsymbol{f}$: $\boldsymbol{w}_j^{n+1} = \frac{\boldsymbol{w}_j^n + \alpha \boldsymbol{f}}{\left\| \boldsymbol{w}_j^n + \alpha \boldsymbol{f} \right\|}$
- Training is done on all samples. Since neuron j wins for which $\boldsymbol{f}$ is most similar to $\boldsymbol{w}_j$, correction is done only for those weights that need minimum change

- If samples are clustered in feature space, different output neurons will win for different samples, causing a gradual separation of weight vectors leading to different output neurons
- Hence, the weight vectors will learn feature patterns that are present in the data
- The process is repeated several times (epochs) until weight change falls below some threshold
- Learning rate decreases with time



Kohonen's self organizing maps (SOM)

- Several factors influence the convergence of the system:
  - 1) first few samples has larger influence than later samples on weights and clusters
  - 2) if number of cluster centers (output neurons) does not match true number in data, some weight vectors $\boldsymbol{w}_j$ will receive only weak reinforcement
  - 3) initial distribution of random weights may misguide pattern search
  - convergence is failing? restart system with new random weights, different sample order
- Training Kohonen network produces feature map that doesn't associate samples to cluster centers, rather, maps out regions in the grid of output neurons
  - Hence, number of output neurons in Kohonen network must be larger than number of expected clusters
  - Essentially, Kohonen network detects clusters in high-dimensional feature space by mapping it to low-dimensional space of node connectivity in the network
- Separation into clusters is a separate step after training network

Conclusion: segmentation

- Segmentation as classification of scene elements solves two problems at once:
- 1) detects an object
- 2) delineates its boundary
- Requires discriminating features of scene elements such as intensity, or vector of intensities from different imaging channels (color, or multiple MRI modalities)
- If discriminating features exist, classification is simple and leads to automatic segmentation.
  - All necessary parameters can be learned from training data
- Compared to classification in general, dimension of feature space is low and sample density of training data is high
  - This allows for estimation of likelihood functions from training data, and consequently, classification by computing conditional posterior probabilities