

Chapter 8: Image Segmentation

Jun.-Prof. Dr.-Ing. Thomas Schultz

URL: <http://cg.cs.uni-bonn.de/iaan/>

E-Mail: schultz@cs.uni-bonn.de

Office: Friedrich-Ebert-Allee 144, 53113 Bonn

January 10, 2017

8.1 Image Segmentation: Goals and Definitions

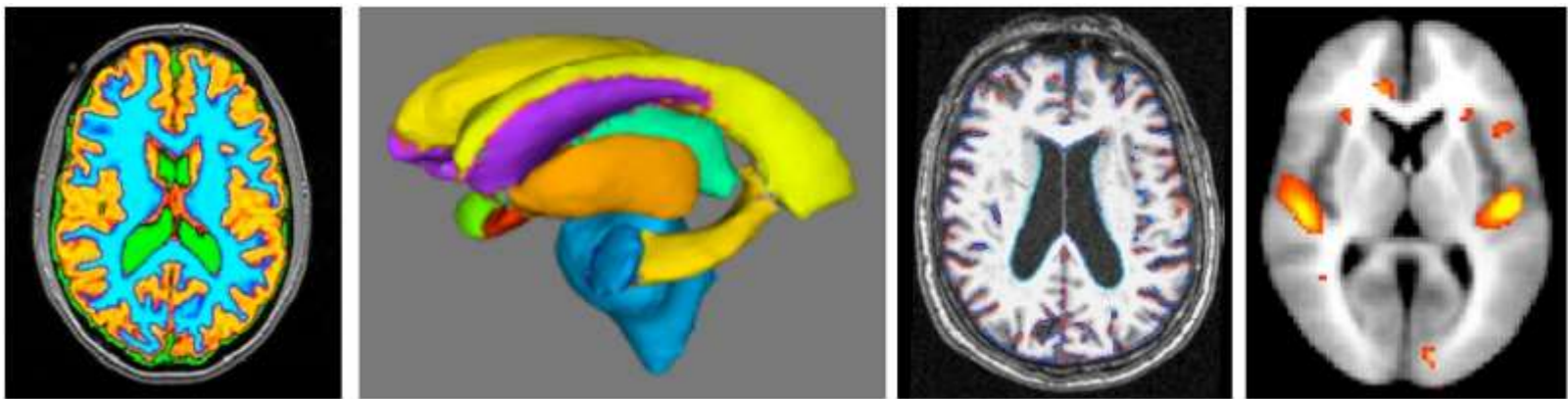
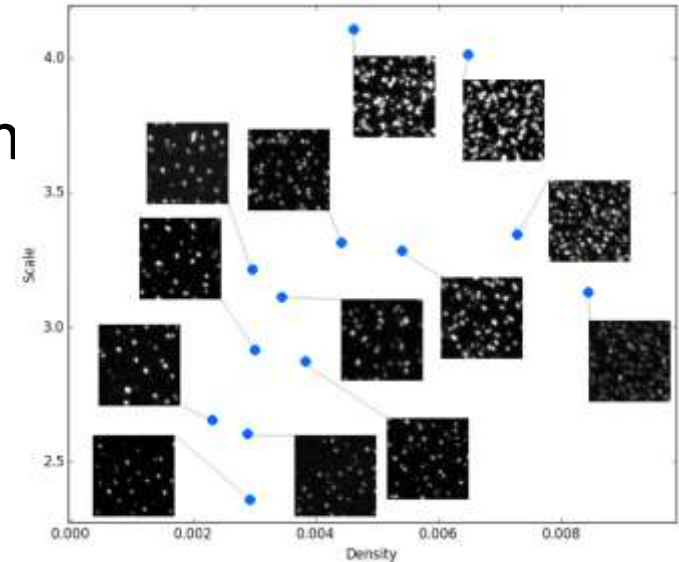
Segmentation, Clustering, Classification

Three closely related problems:

- **Segmentation** partitions images into connected regions that belong to the same object or material
- **Clustering** identifies groups of similar points in a (potentially abstract) data space
 - Often used for segmentation, examples below
- **Classification** gives a name to a voxel or a pre-segmented object
 - *Example:* Detect material at each voxel

Segmentation in Biological Imaging

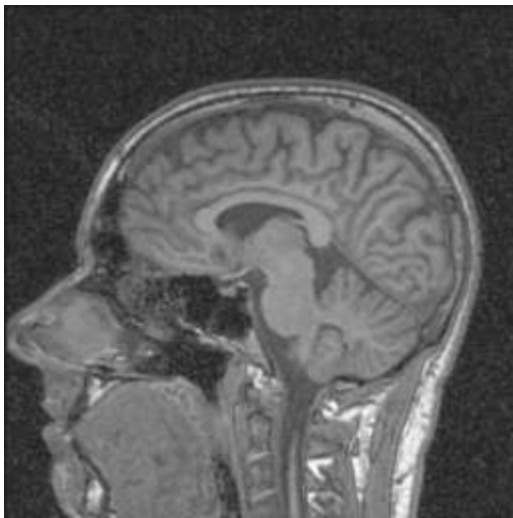
- In biological imaging, segmentation is required for quantification
- In neuroimaging in particular:
 - Extract the brain from MR scans
 - Define regions of interest (ROIs) for analysis
 - Measure the volume of the brain or specific structures



Basic Segmentation Techniques

I expect you already know about:

- Histogram-based Thresholding
- Basic Morphological Operations
- Region Growing



Original Image



Thresholded Image



Opening with
11x11 Circle

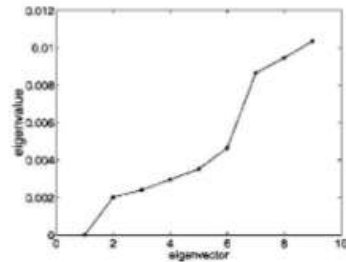
Spectral Clustering for Image Segmentation

- More sophisticated approach that often captures meaningful non-uniform regions [Shi/Malik 2000]:
 - Build a graph in which each pixel becomes a node and is connected to a random subset of pixels within some distance r
 - Subsampling edges speeds up computation
 - Weight edges by a combination of differences in intensity and spatial distance
 - Recursively subdivide the graph using spectral clustering (cf. Section 3.4) while subdivisions have a sufficiently small Normalized Cut value

NCut Image Segmentation: Eigenvectors



Input Image



(a)



(b)



(c)



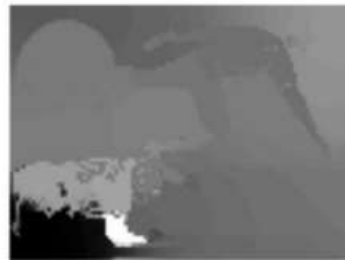
(d)



(e)



(f)



(g)



(h)



(i)

Eigenvectors 2-9

NCut Image Segmentation: Results

- Result of recursive bisection with $\text{NCut} < 0.04$:



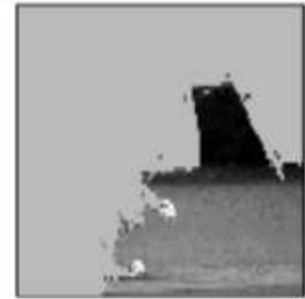
(a)



(b)



(c)



(d)



(e)



(f)



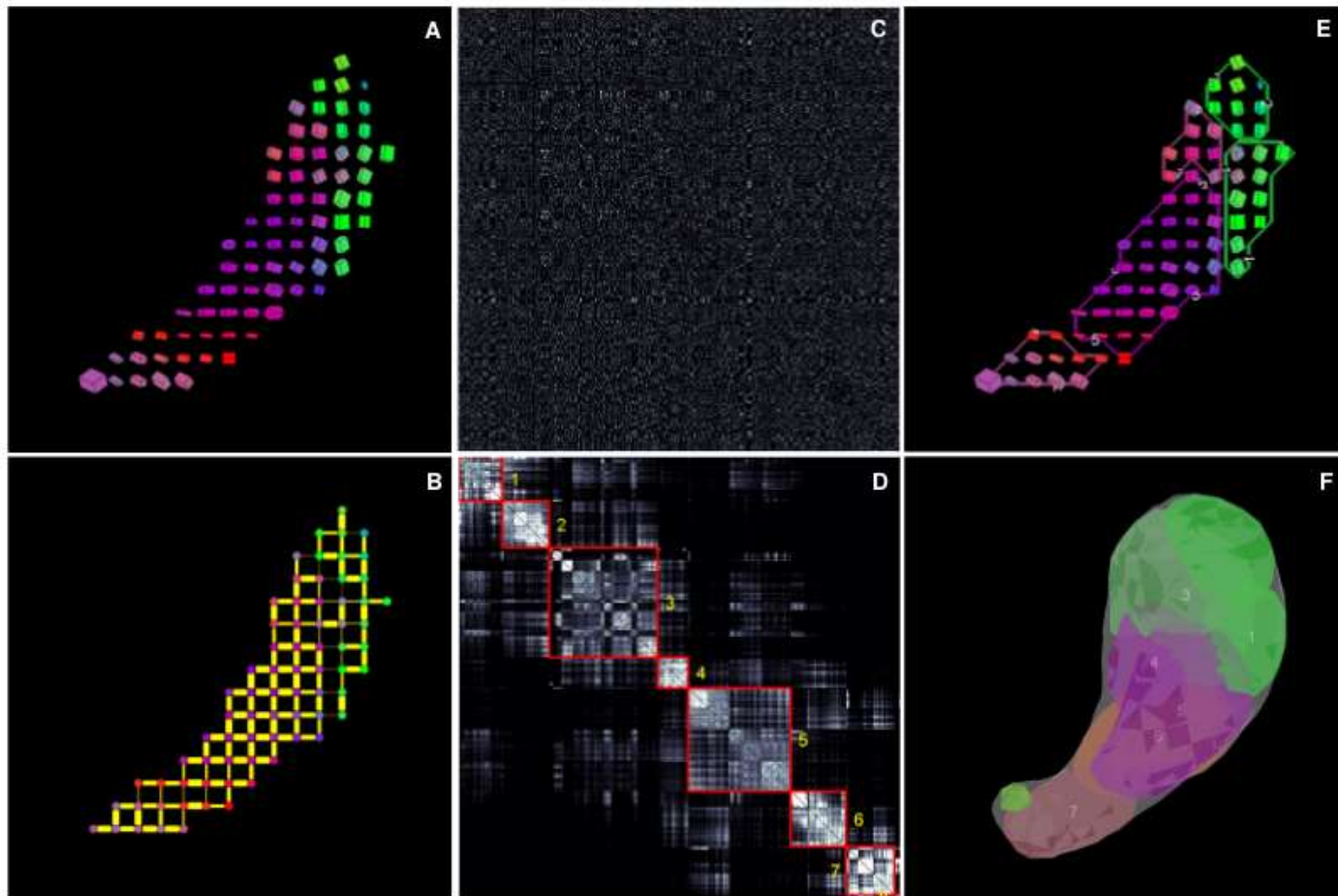
(g)



(h)

NCut Segmentation in Neuroimaging

- *Example:* Segmentation of the thalamus based on diffusion MRI data



Other Segmentation Techniques

- **Deformable Models**
 - Active Contours
 - Level Set Segmentation
- **Markov Random Fields**
 - Main topic of Today's Lecture
 - Gaussian Mixture Models
 - EM Algorithm

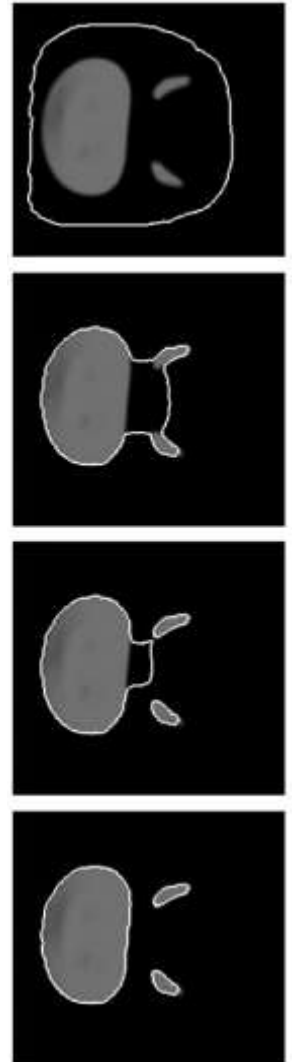


Image: [McInerney/Terzopoulos 1999]

8.2 K-Means Clustering

K-means Distortion Term

- **Basic idea:** Given a fixed number of clusters K , find the optimal cluster centers $\boldsymbol{\mu}_j$ and assignment γ from data index i to cluster index j that jointly minimize the distortion D :

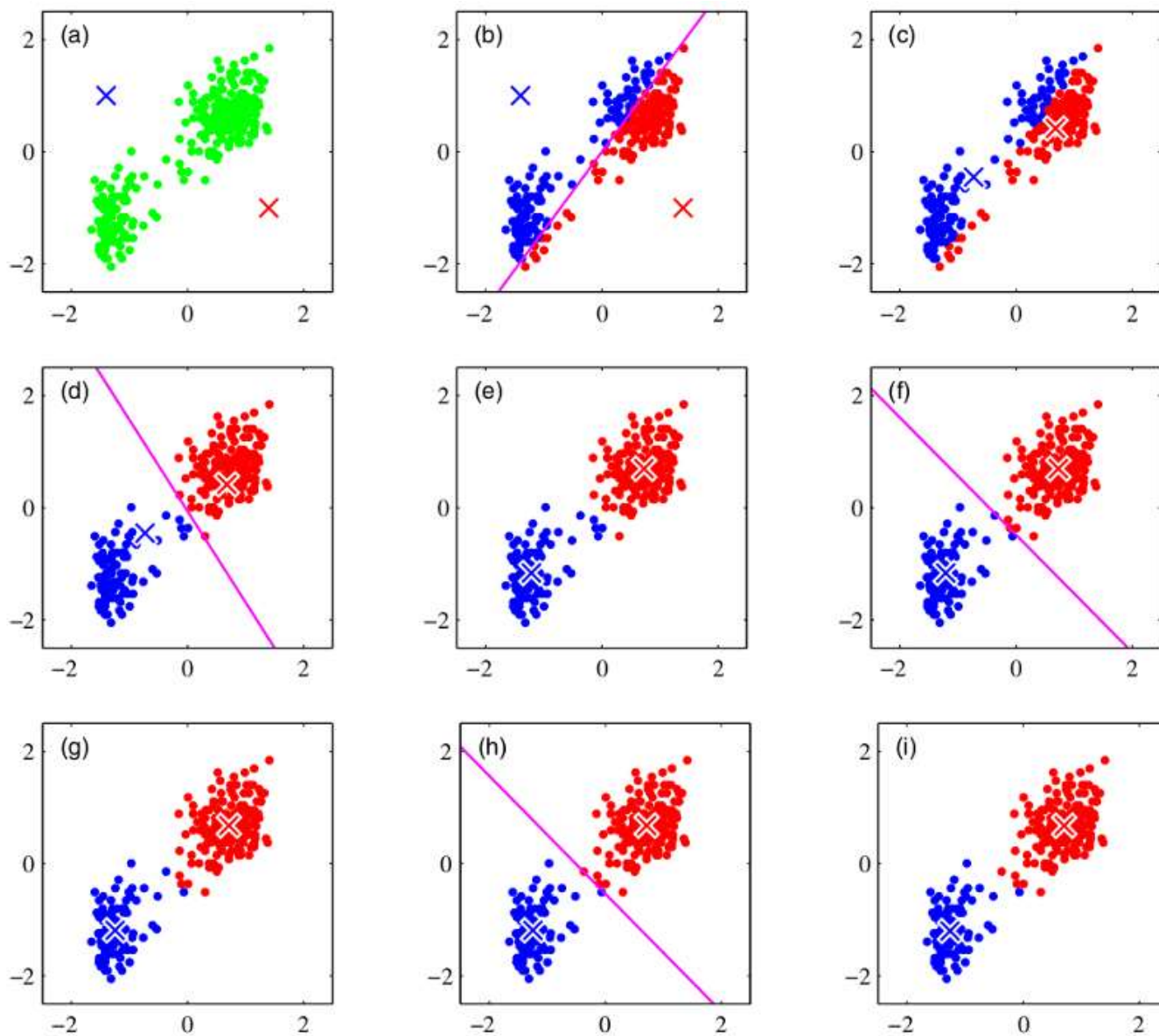
$$D = \sum_i \|\mathbf{v}_i - \boldsymbol{\mu}_{\gamma(i)}\|^2$$

- **Justification:** D measures the overall squared error when approximating original data \mathbf{v}_i with their cluster centers $\boldsymbol{\mu}_j$
- Optimizing D is NP-hard already for $k=2$
[Dasgupta 2008] [Kanade et al. 2008] [Aloise et al. 2009]

K-means Algorithm

- The K -means algorithm optimizes D iteratively by alternating updates to assignments and cluster centers:
 1. Initialize cluster centers μ_1, \dots, μ_k
 - *Methods:* pick or generate K random points; set random assignment and compute means
 2. Update $\gamma(i)$: For each data point, compute the cluster center it is closest to and assign the data point to this cluster
 3. Update μ_j : set cluster centers to mean of data points in cluster
 4. Continue at step 2, until there are no more re-assignments

Example: K -means



Formal Justification of K -means

- K -means alternates between optimizing D with respect to **assignments** or **cluster positions**, which can be done in closed form:

$$D = \sum_i \|\mathbf{v}_i - \boldsymbol{\mu}_{\gamma(i)}\|^2$$

- When $\boldsymbol{\mu}_j$ are fixed, all $\gamma(i)$ are independent
- When cluster members C_j are fixed, setting the derivative of D w.r.t. $\boldsymbol{\mu}_j$ to zero gives:

$$2 \sum_{i \in C_j} (\mathbf{v}_i - \boldsymbol{\mu}_j) = 0 \quad \Leftrightarrow \quad \boldsymbol{\mu}_j = \frac{\sum_{i \in C_j} \mathbf{v}_i}{|C_j|}$$

- **Note:** K -means does *not* guarantee convergence to the global optimum!

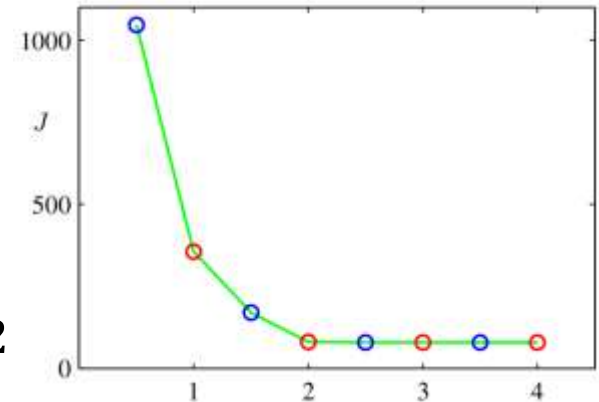


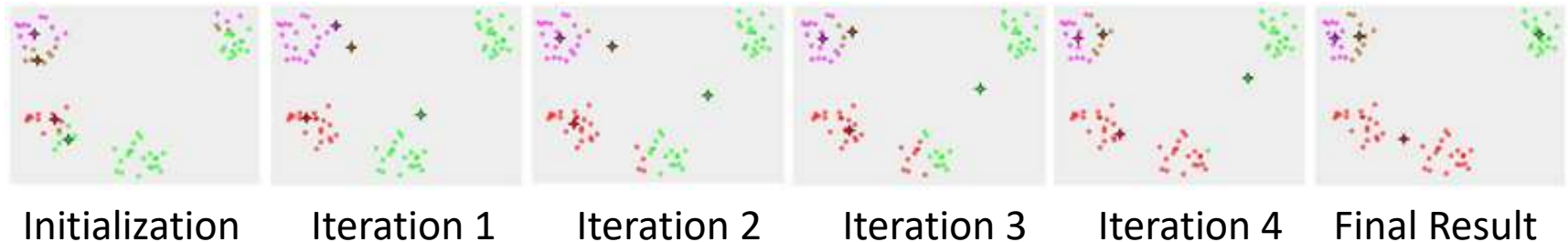
Illustration from [Bishop 2006]

***K*-medoids Clustering**

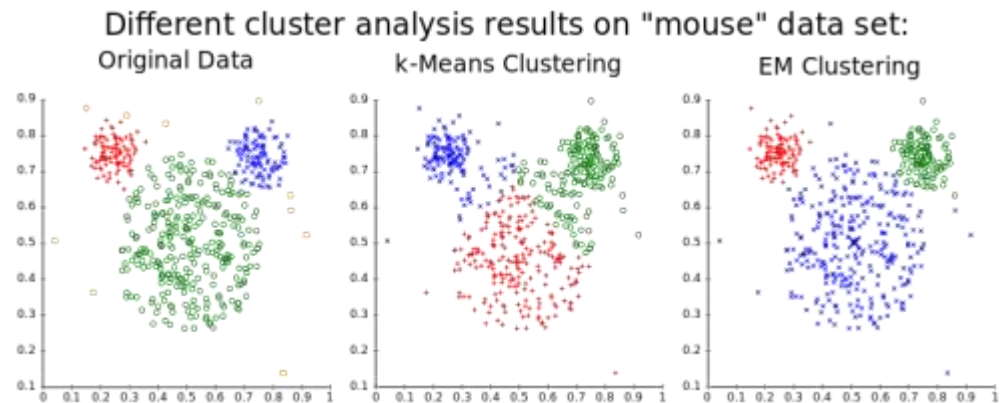
- Same algorithm as *K*-means, but always uses an existing data point as the cluster center
 - Definition of medoid: Minimizes the sum of squared distances to all other members of the cluster
- Problem: “Collapsing” clusters
 - In *K*-means, clusters might be left empty after re-assignment
 - In *K*-medoids, there may be only the medoid itself left after re-assignment
 - Heuristic solution: Detect and re-initialize collapsing clusters

Problems with *K*-means Clustering

- Random initialization means that you may get different clusters each time
 - Can try different initializations and keep the result that minimizes the distortion measure



- Implicit assumption: Clusters are spherical and have same size



K-means for Image Segmentation

Strategy: Represent each pixel as a point in a feature space, apply *K*-means to the result

- Simple example: RGB color channels
- Can be combined with XY(Z) image position

Original image



$K = 2$



$K = 3$

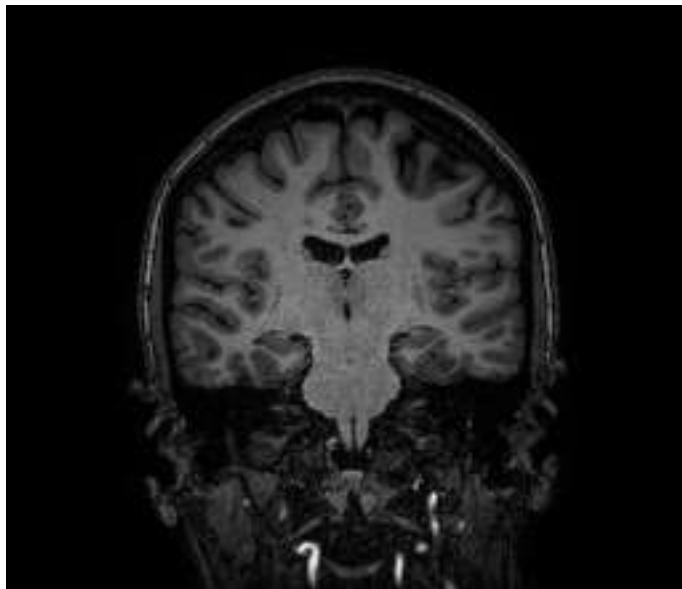



$K = 10$

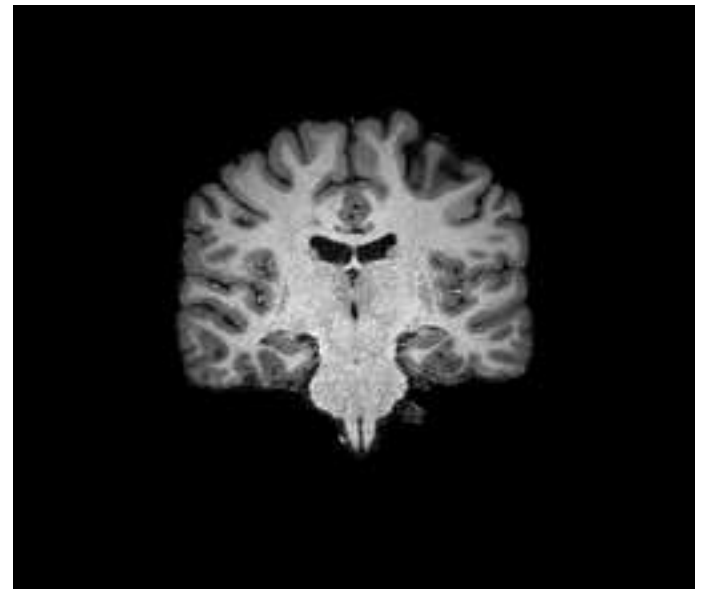


K-means for Tissue Segmentation

- After brain extraction, we would like to classify gray matter vs. white matter vs. CSF:



Brain
Extraction




- Could try *k*-means, but:
 - Would like to model partial voluming / uncertainty
 - Would like to allow for different variances per class

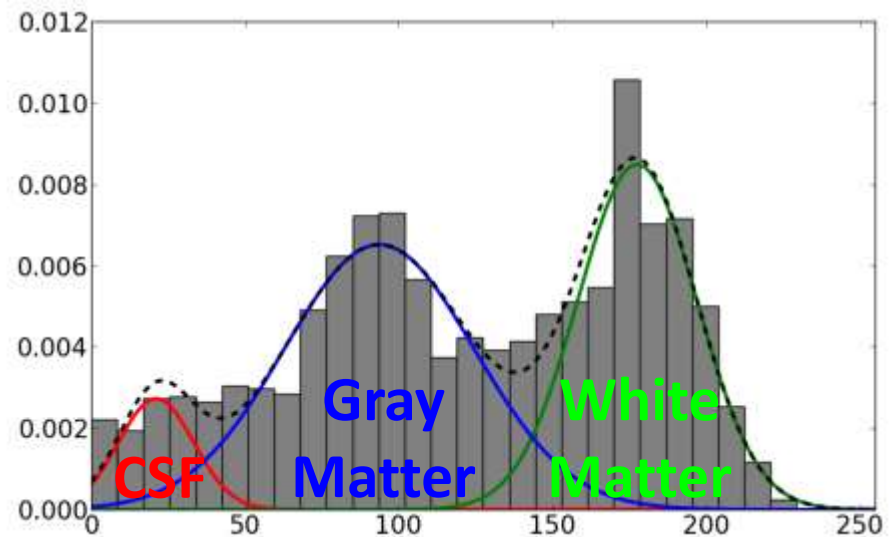
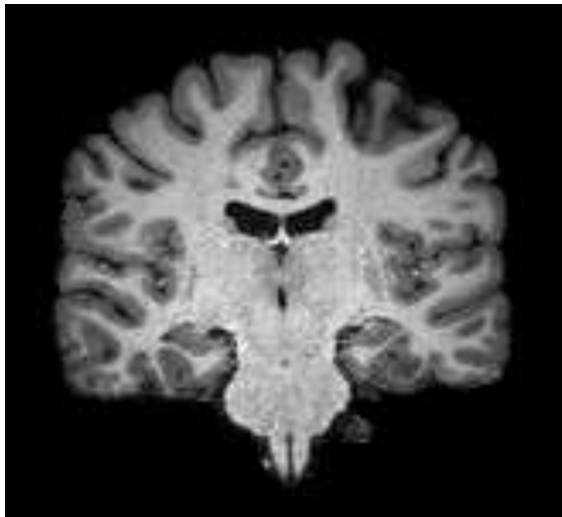
Summary: *K*-means Clustering

- ***K*-means** is one of the simplest and most widely used algorithms for clustering
 - Alternates between optimizing cluster assignments and the positions of cluster centers
 - Optimizes a formal distortion measure
- Can be used for **basic tissue segmentation**
 - Drawbacks will be addressed by Gaussian Mixture Models (next section)

8.3 Gaussian Mixture Models

Assumptions in Gaussian Mixture Models

- CSF, gray matter, and white matter can be distinguished in the histogram:



- Gaussian Mixture Models assume:
 - Each material has normally distributed MR intensities
 - We are observing a mixture of them

Gaussian Mixture Models

- Formally, a Gaussian Mixture Model is given by mixing K Gaussian distributions:

$$p(x) = \sum_{k=1}^K \pi_k N(x|\mu_k, \sigma_k^2)$$

- where $N(x|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$
- π_k are the mixing coefficients
 - Normalization constraint: $\sum_{k=1}^K \pi_k = 1$

Joint and Marginal Probabilities

- The **joint probability distribution** of two random variables provides the probability that both variables simultaneously take on specific values
 - *Example:* Joint distribution of the place and type of consumed beer

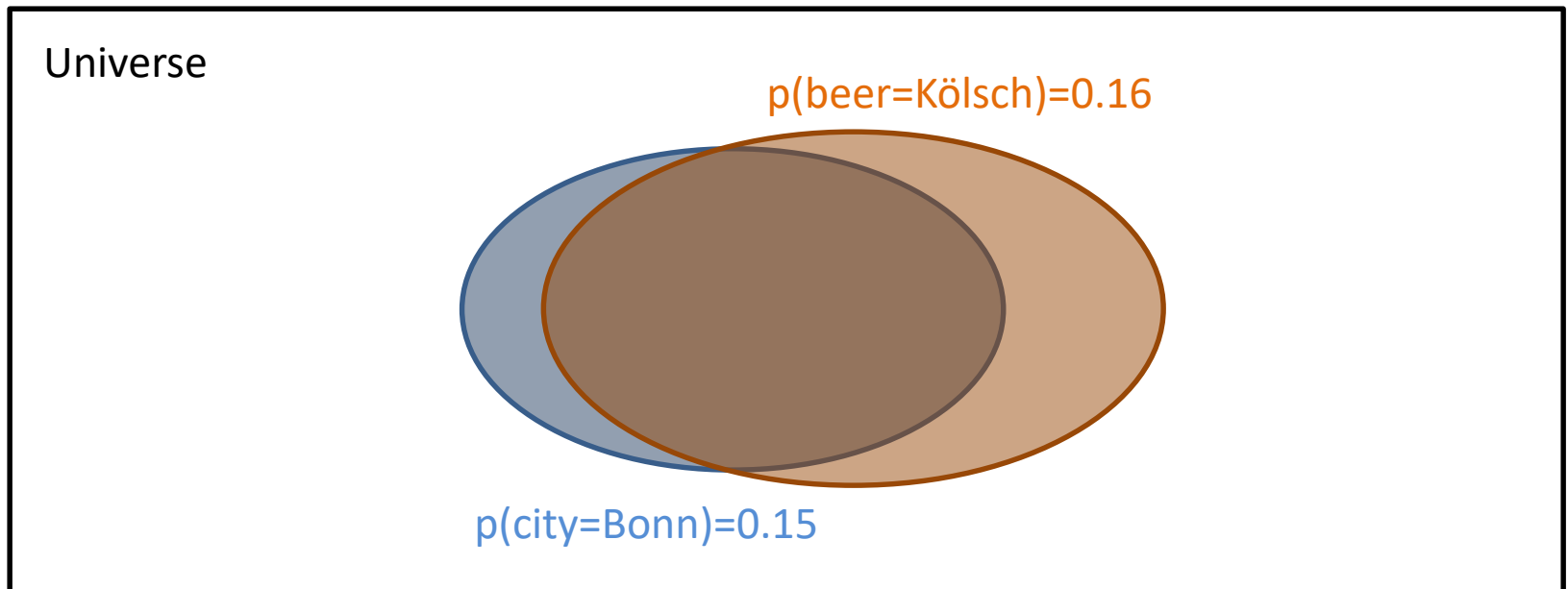
	Bonn	Hamburg	
Kölsch	0.12	0.04	0.16
Pils	0.03	0.81	0.84
	0.15	0.85	

- The **marginal probability** is the probability of one variable taking on a specific value regardless of the other one
 - Computed by summing over all values of the other variable („marginalization“)

Conditional Probabilities

- The **conditional probability** $p(B|A)$ of B (e.g., beer=Kölsch) given A (e.g., city=Bonn) is obtained by re-normalizing their joint probability $p(A \cap B)$ w.r.t. $p(A)$:

$$p(B|A) = \frac{p(A \cap B)}{p(A)}$$



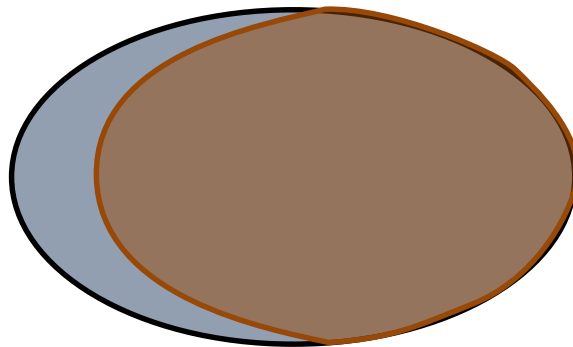
Conditional Probabilities

- The **conditional probability** $p(B|A)$ of B (e.g., beer=Kölsch) given A (e.g., city=Bonn) is obtained by re-normalizing their joint probability $p(A \cap B)$ w.r.t. $p(A)$:

$$p(B|A) = \frac{p(A \cap B)}{p(A)}$$

- „Restrict the universe“ to A

$p(\text{beer}=\text{Kölsch} \mid \text{city}=\text{Bonn})=0.8$



“Local Universe”: city=Bonn

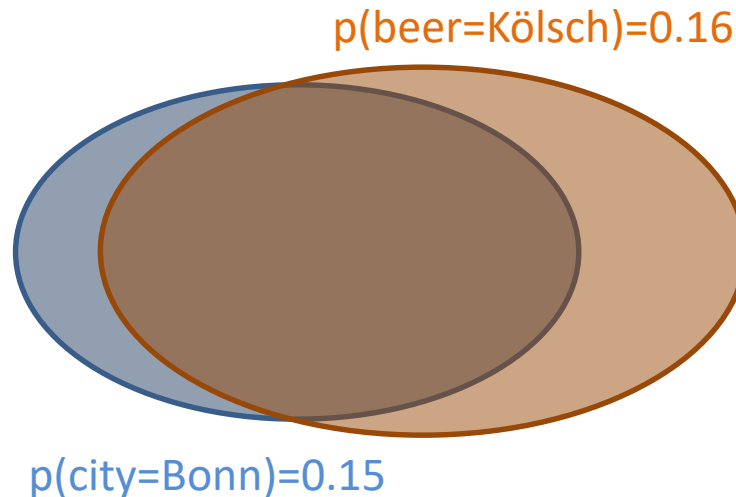
Bayes' Rule

- **Bayes' rule** computes the conditional probability $p(A|B)$ from $p(B|A)$, $p(A)$, and $p(B)$:

$$p(A|B) = \frac{p(B|A) p(A)}{p(B)}$$

- Follows from: $p(A \cap B) = p(B|A)p(A) = p(A|B)p(B)$

Universe



Membership Vectors and Mixing Coefficients

- For each data point $i \in \{1, 2, \dots, n\}$, introduce a hidden **membership vector** $\mathbf{z}_i \in \{0, 1\}^K$ that specifies the cluster k to which i belongs.
 - For each i , $\sum_{k=1}^K z_{ik} = 1$
- Results in **mixing coefficient** π_k of cluster k

$$\pi_k = \frac{1}{n} \sum_{i=1}^n z_{ik}$$

- Naturally fulfill the normalization constraint $\sum_{k=1}^K \pi_k = 1$

Responsibilities

- Our task in clustering is to estimate the membership vectors. For this, we define a **conditional probability of point i belonging to cluster k** given the data

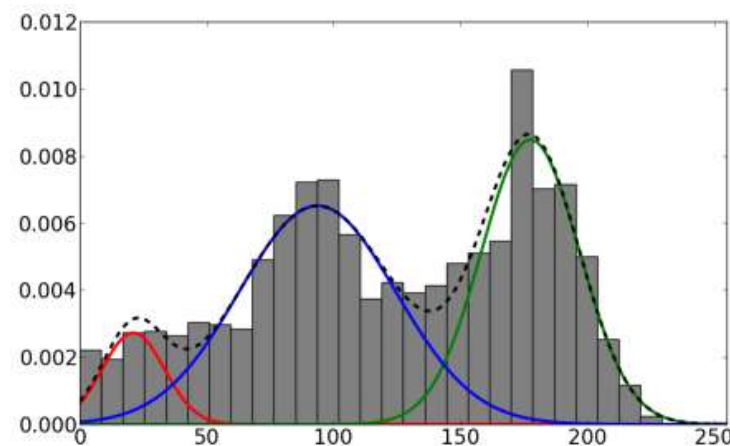
$$\rho_{ik} := p(z_{ik} = 1 | x_i)$$

- “Responsibility” of cluster k for data i
- **Expectation-Maximization** algorithm alternates two steps:
 - **E-Step:** Given π, μ, σ , estimate ρ
 - **M-Step:** Given ρ , optimize π, μ, σ to fit data

E-Step: Computing Responsibilities

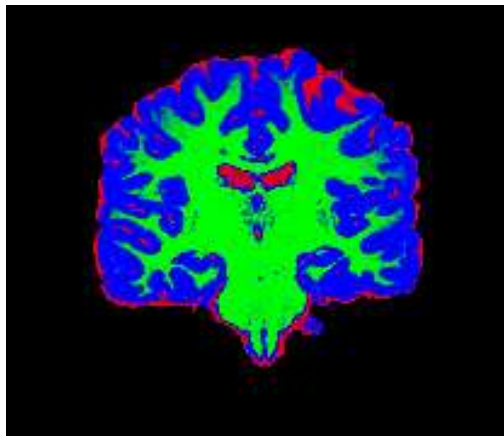
- Given the parameters of a GMM, ρ_{ik} can be computed using Bayes' rule:

$$\begin{aligned}\rho_{ik} &= p(z_{ik} = 1 | x_i) \\ &= \frac{p(x_i | z_{ik} = 1) p(z_{ik} = 1)}{p(x_i)} \\ &= \frac{N(x_i | \mu_k, \sigma_k^2) \pi_k}{\sum_{l=1}^K \pi_l N(x_i | \mu_l, \sigma_l^2)}\end{aligned}$$

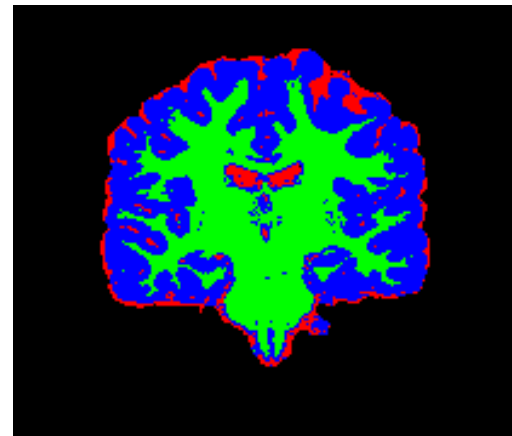


Segmentation Using GMMs

- Given a GMM, the ρ_{ik} provide the desired probability of a voxel containing **CSF**/**GM**/**WM**
 - If hard classification is desired, we can assign voxel i to material k with maximum ρ_{ik}



Probabilities



Hard Assignments

- **But:** Still need to estimate parameters π, μ, σ

M-Step: Fitting GMM to Data

- We would like the GMM to explain our data as well as possible. Therefore, we will **maximize the likelihood** of our data given the GMM. If we assume independent samples x_i :

$$p(X|\pi, \mu, \sigma) = \prod_{i=1}^n \sum_{k=1}^K \pi_k N(x_i|\mu_k, \sigma_k^2) \rightarrow \max$$

- For ease of computation, we take the log:

$$\ln p(X|\pi, \mu, \sigma) = \sum_{i=1}^n \ln \sum_{k=1}^K \pi_k N(x_i|\mu_k, \sigma_k^2)$$

Optimizing μ_k

Reminder: $\ln p(X|\pi, \mu, \sigma) = \sum_{i=1}^n \ln \sum_{l=1}^K \pi_l N(x_i | \mu_l, \sigma_l^2)$

- In order to optimize μ_k while keeping all other parameters fixed, set derivative w.r.t. μ_k to 0:

$$\frac{\partial \ln p}{\partial \mu_k} = \sum_{i=1}^n \frac{\pi_k N(x_i | \mu_k, \sigma_k^2)}{\underbrace{\sum_{l=1}^K \pi_l N(x_i | \mu_l, \sigma_l^2)}_{\rho_{ik}}} \frac{2(x_i - \mu_k)}{2\sigma_k^2} = 0$$

- Results in the following update rule:

$$\mu_k = \frac{\sum_{i=1}^n \rho_{ik} x_i}{N_k} \quad \text{with} \quad N_k := \sum_{i=1}^n \rho_{ik} \quad \text{“effective number of voxels in cluster } k\text{”}$$

Optimizing σ_k

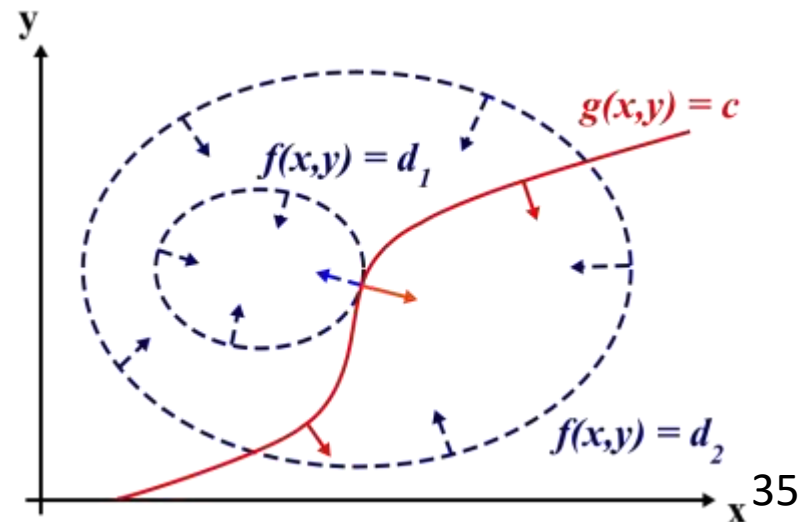
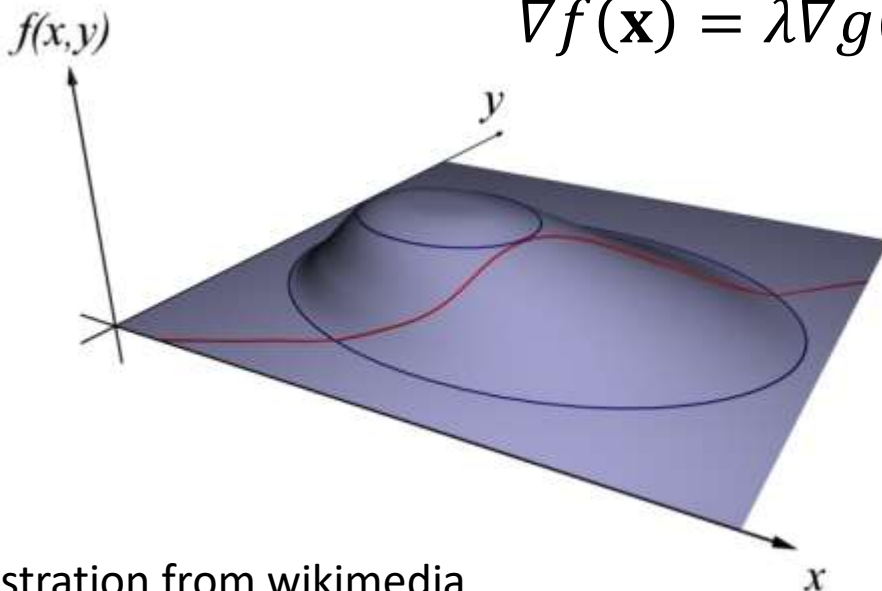
- σ_k can be optimized in a very similar way (left as an exercise)
- Results in the following update rule:

$$\sigma_k^2 = \frac{\sum_{i=1}^n \rho_{ik} (x_i - \mu_k)^2}{N_k}$$

Method of Lagrange Multipliers

- While optimizing π_k , we need to maintain the normalization constraint: $\sum_{l=1}^K \pi_l = 1$
- This can be achieved using a Lagrange multiplier:
 - A necessary condition for a local maximum of $f(\mathbf{x})$ subject to an equality constraint $g(\mathbf{x})=c$ is given by solving the system:

$$\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x}) \text{ and } g(\mathbf{x}) = c$$



Optimizing π_k

Reminder: $\ln p(X|\pi, \mu, \sigma) =$
 $\sum_{i=1}^n \ln \sum_{l=1}^K \pi_l N(x_i|\mu_l, \sigma_l^2)$

- Computing $\frac{\partial \ln p}{\partial \pi_k} = \lambda \frac{\partial \sum_{l=1}^K \pi_l}{\partial \pi_k}$ gives us

$$\sum_{i=1}^n \frac{N(x_i|\mu_k, \sigma_k^2)}{\sum_{l=1}^K \pi_l N(x_i|\mu_l, \sigma_l^2)} = \lambda \quad (*)$$

- Multiplying both sides of (*) by π_k and summing over k gives $\lambda = n$ (since $\sum_{k=1}^K \pi_k = 1$)
- Multiplying (*) by π_k and substituting $\lambda = n$ gives us the update rule $\pi_k = \frac{N_k}{n}$

Overview of the EM Algorithm

1. Initialize K means μ_k , variances σ_k^2 , and mixing coefficients π_k

2. E-step: Update responsibilities

$$\rho_{ik} = \frac{\pi_k N(x_i | \mu_k, \sigma_k^2)}{\sum_{l=1}^K \pi_l N(x_i | \mu_l, \sigma_l^2)}$$

3. M-step: Update GMM parameters

$$\mu_k = \frac{\sum_{i=1}^n \rho_{ik} x_i}{N_k}; \sigma_k^2 = \frac{\sum_{i=1}^n \rho_{ik} (x_i - \mu_k)^2}{N_k}; \pi_k = \frac{N_k}{n}$$

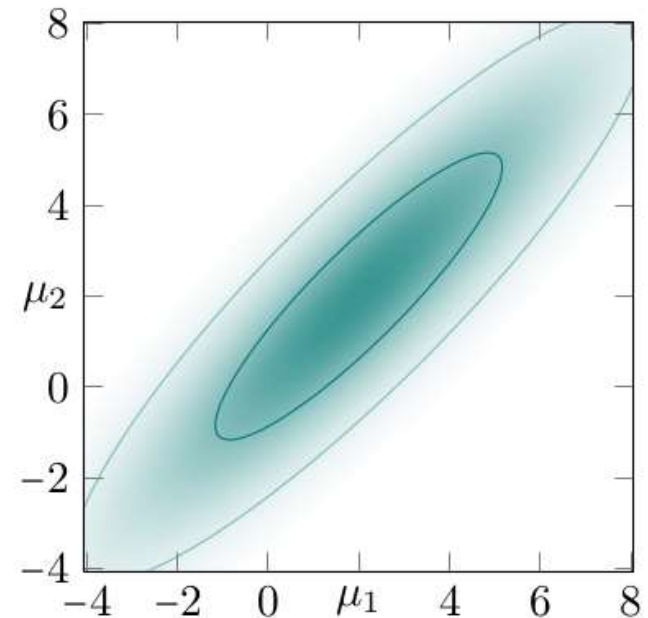
4. Iterate E-step and M-step until log likelihood or model parameters have stabilized

Multivariate Normal Distributions

- The 1D Gaussian distribution that we've worked with so far has a natural generalization to M -D vector spaces:

$$N(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{M}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

- Mean vector $\boldsymbol{\mu}$
- Covariance matrix $\boldsymbol{\Sigma}$
 - symmetric positive definite



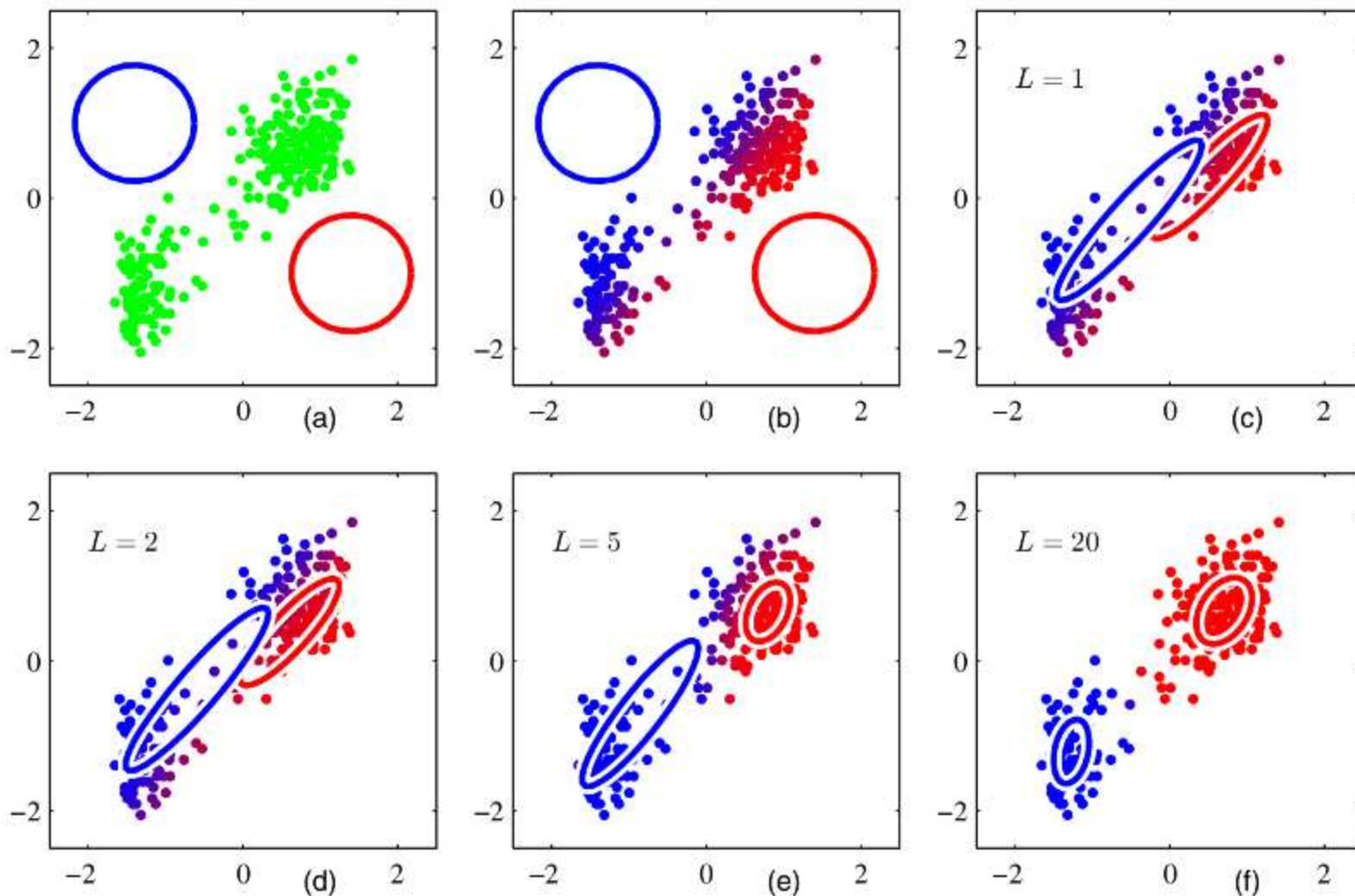
EM With Vector Input

- The basic EM algorithm remains exactly the same with vector input. The update rules for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ in the M-step are:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^n \rho_{ik} \mathbf{x}_i}{N_k}$$
$$\boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^n \rho_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{N_k}$$

- Examples:* Color images, multi-contrast MR

Example: Expectation-Maximization



EM vs. K means

- **Structural similarity** to K means:
 - **E-step:** Cluster Assignment
 - **M-step:** Cluster center update
 - Can be made formal:
 - K means obtained by setting $\Sigma_k := \epsilon \mathbf{I}$ and letting $\epsilon \rightarrow 0$
(see Bishop 2006 for details)
- Convergence much slower than K means
 - Initializing cluster centers using K means often leads to faster convergence

Singularities in the EM Algorithm

- EM converges to a local optimum
 - Like K means, it relies on suitable initialization
- Components can collapse to single points
 - Similar singularity exists in K means
 - *Heuristic*: Detect degenerate clusters and re-initialize their center, e.g., by splitting the largest cluster

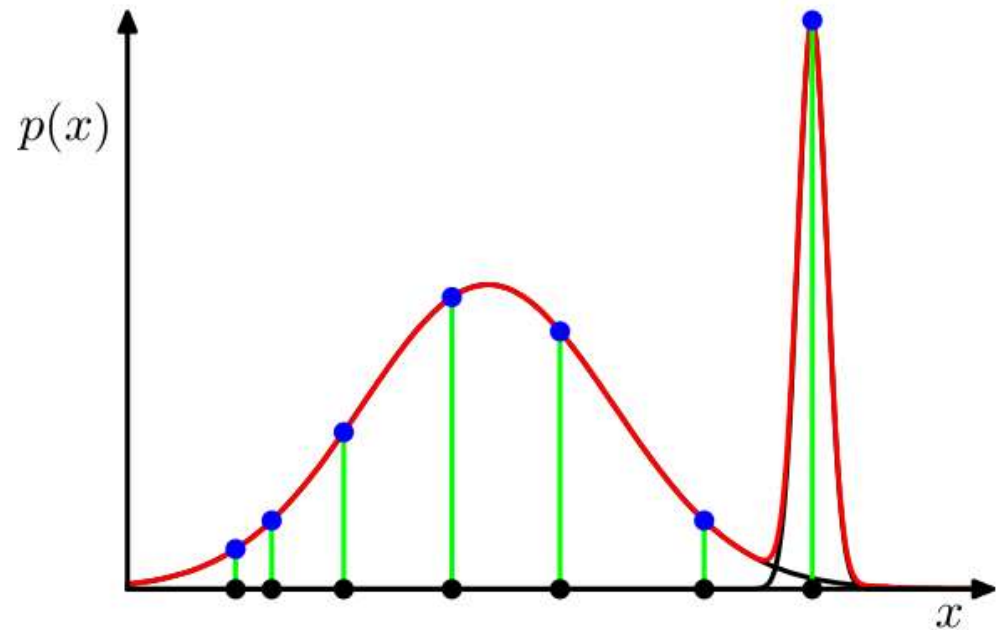


Illustration from [Bishop 2006]

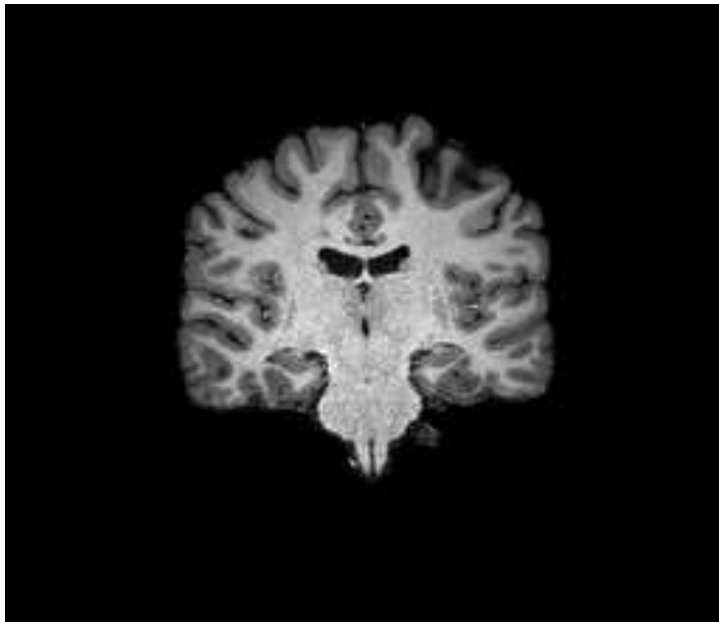
Summary: Gaussian Mixture Models

- **Gaussian Mixture Models (GMMs)** assume that each cluster corresponds to a normal distribution
 - In multivariate case: Ellipsoid cluster shapes
- Probabilistic **cluster membership** (“responsibilities”) computed using Bayes’ rule
- **Expectation-Maximization (EM) Algorithm** used to fit GMMs
 - Convergence to local optima
 - Potential singularities

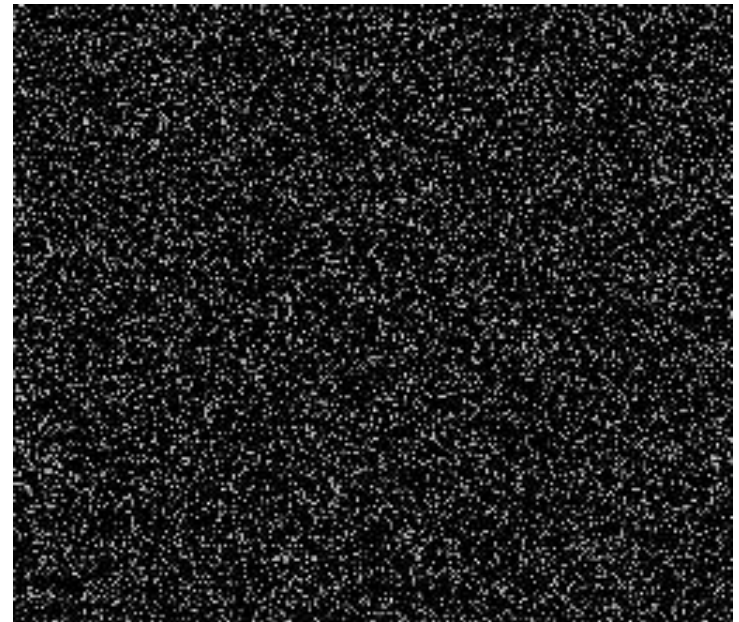
8.4 Markov Random Fields

MRFs: Motivation

- Gaussian Mixture Models are purely histogram-based. Therefore, the following two images look identical to them:



Brain Slice

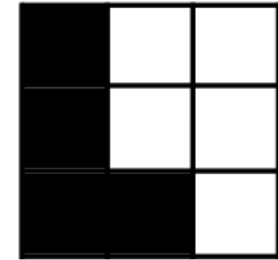


Permuted Brain Slice

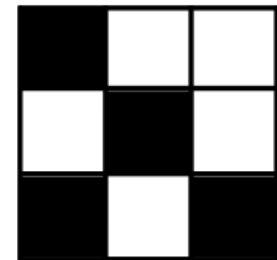
Spatial Structure in Images

- **Segmentation vs. Clustering:**

- Previous section used clustering for image segmentation
- Dedicated segmentation algorithms exploit the **spatial structure of the image**
 - *Example of spatial correlation:* It is much more likely that a voxel contains gray matter if all its neighbors contain gray matter
- Modeling spatial correlations makes the method much more stable for **noisy images**



Likely configuration
High probability



Unlikely configuration
Low probability

Image Noise

- Since Gaussian Mixture Models do not enforce any regularity in their classification results, they are highly susceptible to noise:



(low noise)



(med. noise)



(high noise)

Simulated Noisy Data



GMM Result
(low noise)

Prior Cluster Likelihood Revisited

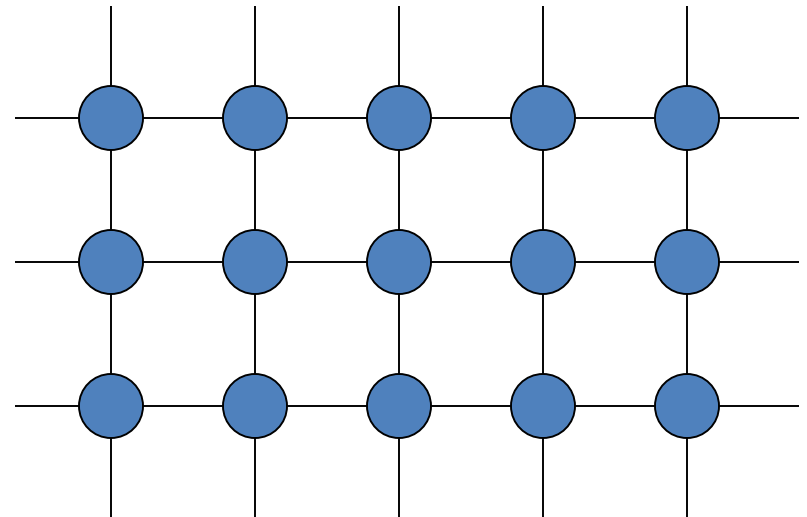
- Remember conditional probability of voxel i belonging to cluster k :

$$p(z_{ik} = 1|x_i) = \frac{p(x_i|z_{ik} = 1) p(z_{ik} = 1)}{p(x_i)}$$

- In the Gaussian Mixture Model, $p(z_{ik} = 1) = \pi_k$, i.e., independent from label of the neighbors.
- Challenge:** We wish to model the spatial dependencies in MR images to better deal with noise. However, introducing long-range dependencies makes the model intractable.

Markov Random Fields

- **Common solution:** Define a neighborhood graph from the pixel grid. Each pixel depends on all others only through its immediate neighbors.
 - Generalizes the Markov chain introduced by Andrey Markov (1856-1922)
- z_i are no longer independent; we need to consider the *joint distribution* of z_S (S =set of all voxels)
- **But:** z_i independent from all others given its immediate neighbors $NB(i)$:
$$p(z_i | z_{S-\{i\}}) = p(z_i | z_{NB(i)})$$

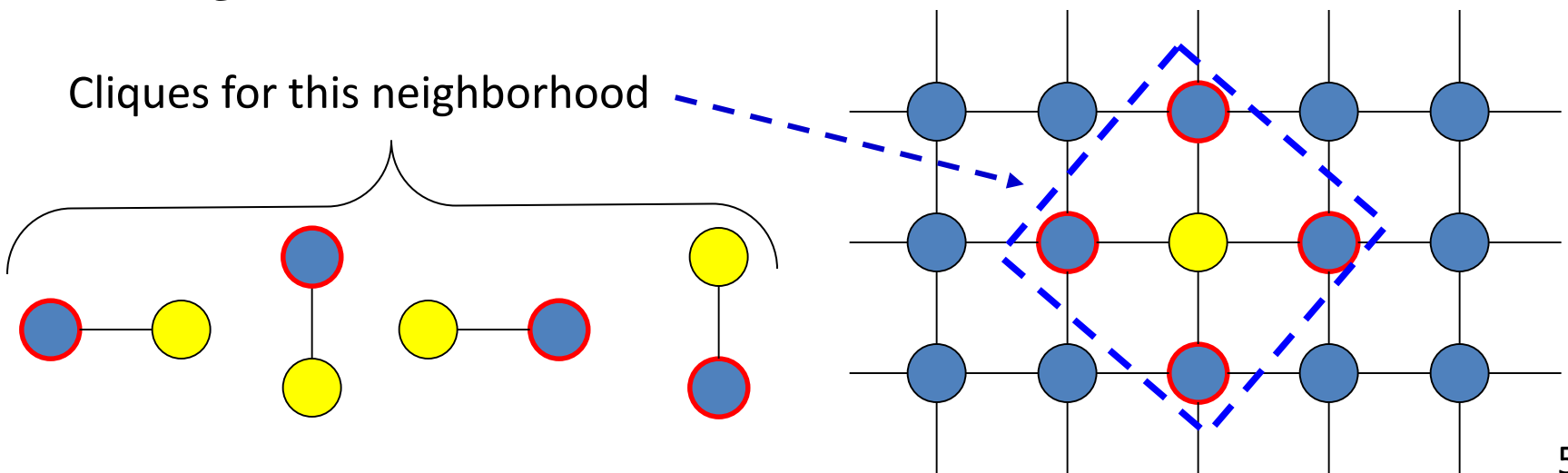


Hammersley-Clifford Theorem

- **Theorem:** If the density of z_S is positive, it factorizes over the cliques C of a Markov Random Field:

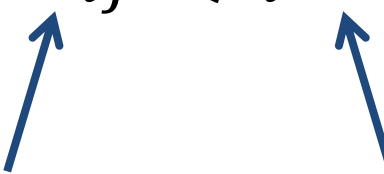
$$p(z_s) \propto \prod_c p(\mathcal{C}) = e^{-\sum_c V_c(z_c)}$$

- V_C are called “clique potentials”



Generalized Potts Model

- The **Generalized Potts Model** is a widely used potential for cliques consisting of pairs of voxels:

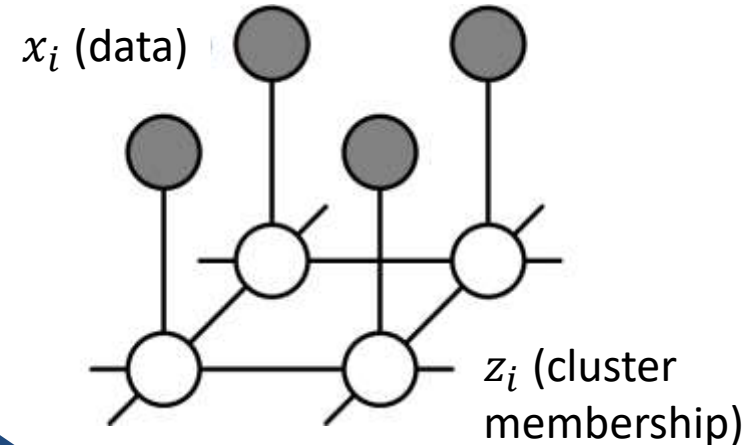
$$V_{ij}(z_i, z_j) = u_{ij} \delta(z_i \neq z_j)$$


u_{ij} = Penalty for discontinuity between voxels i and j ; in the simplest case, the same number β for all pairs

$\delta(z_i \neq z_j) = 1$
if $z_i \neq z_j$, 0 else

Including the Data in the MRF

- So far, we've used MRFs to model the marginal of cluster membership z
- Let's add our data



$$p(z_S|x_S) = \frac{p(x_S|z_S) p(z_S)}{p(x_S)}$$

- Leads to **unary potentials** V_i :

$$p(z_S|x) \propto e^{-\sum_i V_i(x_i, z_i) - \sum_i \sum_{j \in N(i)} V_{ij}(z_i, z_j)}$$

Unary Potentials

- We model the data likelihood with the same Gaussian Model as before:

$$p(x_S|z_S) = \prod_{i=1}^n N(x_i|\mu_{z_i}, \sigma_{z_i}^2)$$

- Re-writing it as unary potentials V_i such that

$$p(x_S|z_S) = e^{-\sum_i V_i(x_i, z_i)}$$

results in

$$V_i(x_i, z_i) = \ln \sqrt{2\pi}\sigma_{z_i} + \frac{(x - \mu_{z_i})^2}{2\sigma_{z_i}^2}$$

MAP Estimation

- Our goal is to find the label set z_S with **maximum a posteriori probability** (MAP)
 - Compute the most likely (hard) label assignment
 - Consider the *joint* distribution of labels for all voxels
 - This is done by minimizing the negative log of the probability, which we call **energy**:

$$\underbrace{\sum_i V_i(x_i, z_i)}_{\substack{\text{Data Term} \\ \text{External Energy}}} + \underbrace{\sum_i \sum_{j \in N(i)} V_{ij}(z_i, z_j)}_{\substack{\text{Smoothness Term} \\ \text{Internal Energy}}}$$

Iterated Conditional Modes (ICMs)

- Finding the MAP estimate of the joint label distribution is a difficult problem
- **Classic approach:** Iterated Conditional Modes
 1. Start with a reasonable initialization (e.g., by ignoring binary potentials)
 2. Iterate over all voxels:
 - Condition on all the neighbors
 - Pick the label that minimizes the energy
 3. Repeat step 2 until convergence
- Converges to a local minimum

Other Methods for Finding the MAP Estimate

- **Simulated annealing**
 - Usually gets much closer to the global optimum, but very time consuming
- **Graph cuts** [Boykov et al. 2001]
 - Formulates binary label assignment as a min-cut/max-flow graph cut problem, for which efficient algorithms exist
 - Solution only approximate for $K > 2$ labels
- **Belief Propagation** [Weiss/Freeman 2001]
 - Nodes exchange messages expressing their belief about cluster assignments
 - Exact for trees, may not converge for loopy graphs

The HMRF-EM Algorithm

- Modification of the EM Algorithm for Hidden Markov Random Fields:

- **E-step:**

1. Compute MRF-MAP estimate (hard labels)
2. Calculate posterior distribution of labels:

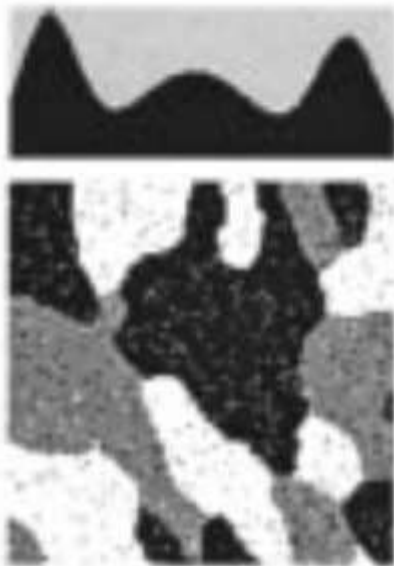
$$\rho_{ik} = p(z_{ik} = 1 | x_i) = \frac{p(x_i | z_{ik} = 1) p(z_{ik} = 1 | N(i))}{p(x_i)}$$

- **M-step:**

- Use existing update rules to estimate μ_k and σ_k^2
- Explicit mixing weights π_k no longer used

Results of HMRF-EM

- Binary potentials greatly reduce misclassification on simulated example:



MCR: 5.82%

GMM Result
(low noise)



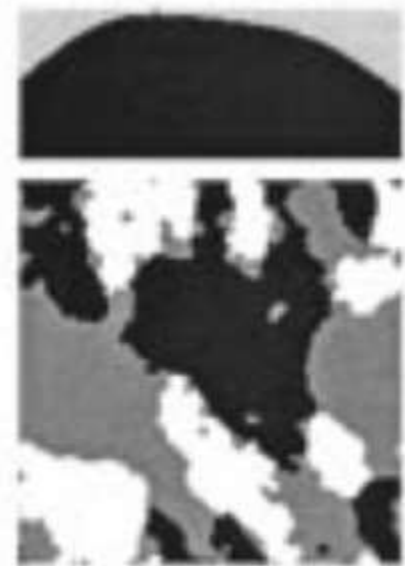
MCR: 0.12%

HMRF-EM
(low noise)



MCR: 1.04%

HMRF-EM
(med. noise)

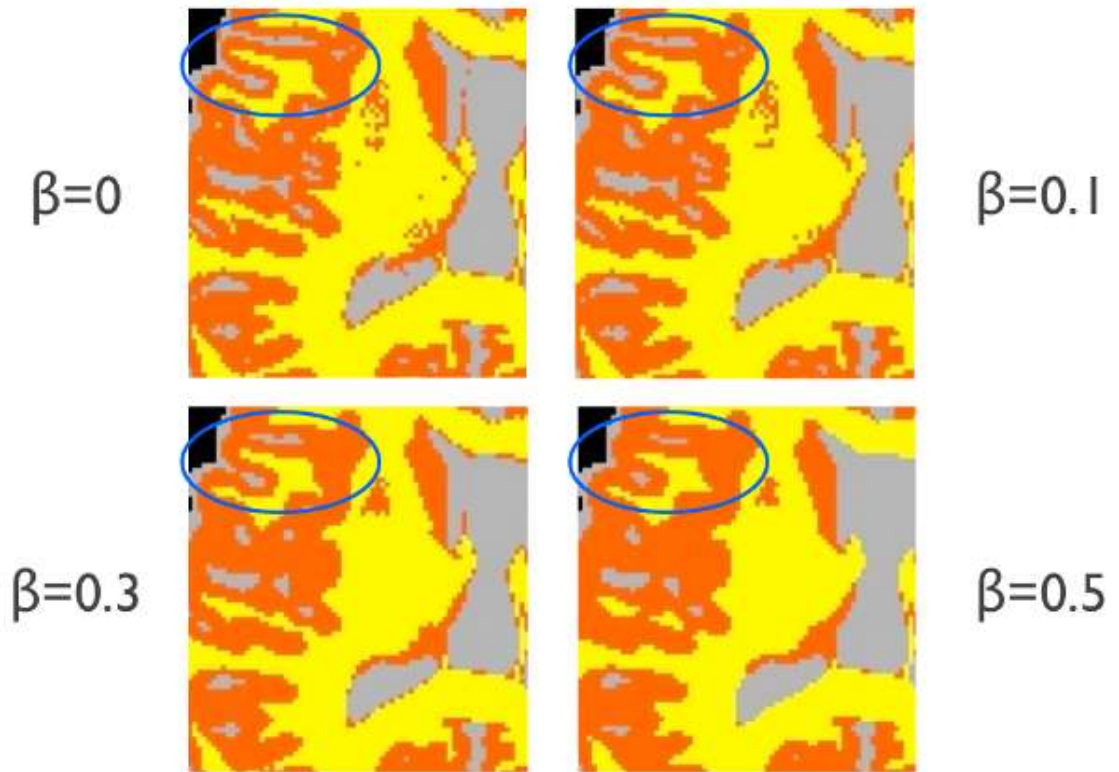


MCR: 8.73%

HMRF-EM
(high noise)

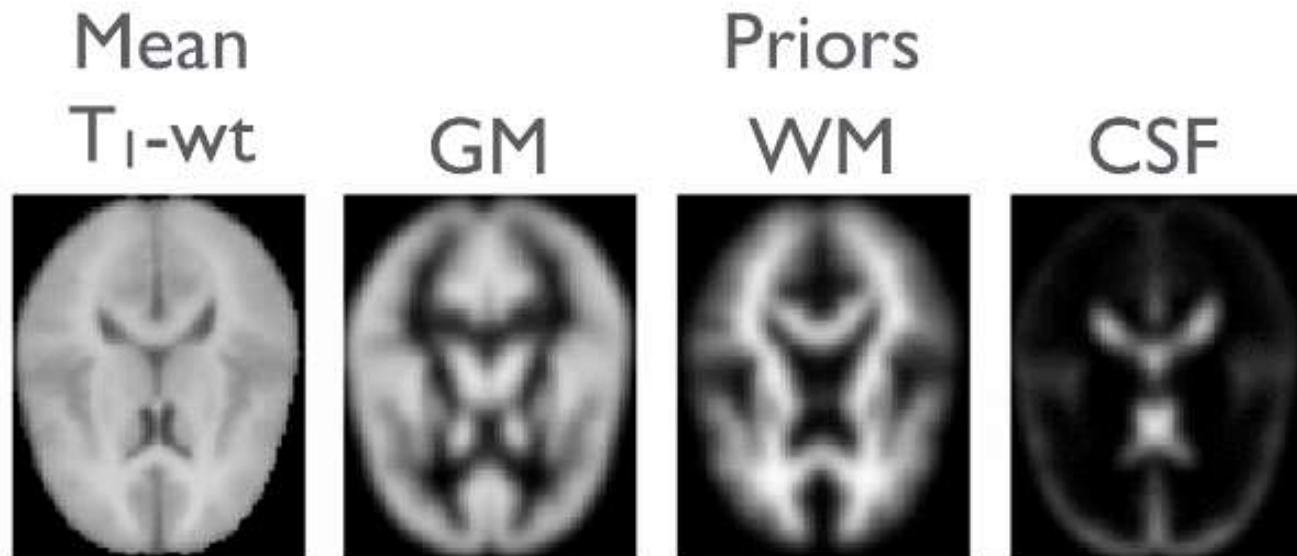
Data vs. Smoothness Tradeoff

- The result of MRF segmentation depends on the relative weight of the smoothness and data terms
 - In generalized Potts model $V_{ij}(z_i, z_j) = u_{ij}\delta(z_i \neq z_j)$, defined by u_{ij}
 - Results with $u_{ij} = \beta$ for all i, j :
 - Detail view:



Priors from Atlas

- In addition to smoothness, we can use tissue class probabilities from an atlas as a prior
 - Can help in difficult cases (e.g., strong bias field)
 - Requires registration to the atlas
 - Cave: Misregistration can skew the results



Summary: Markov Random Fields

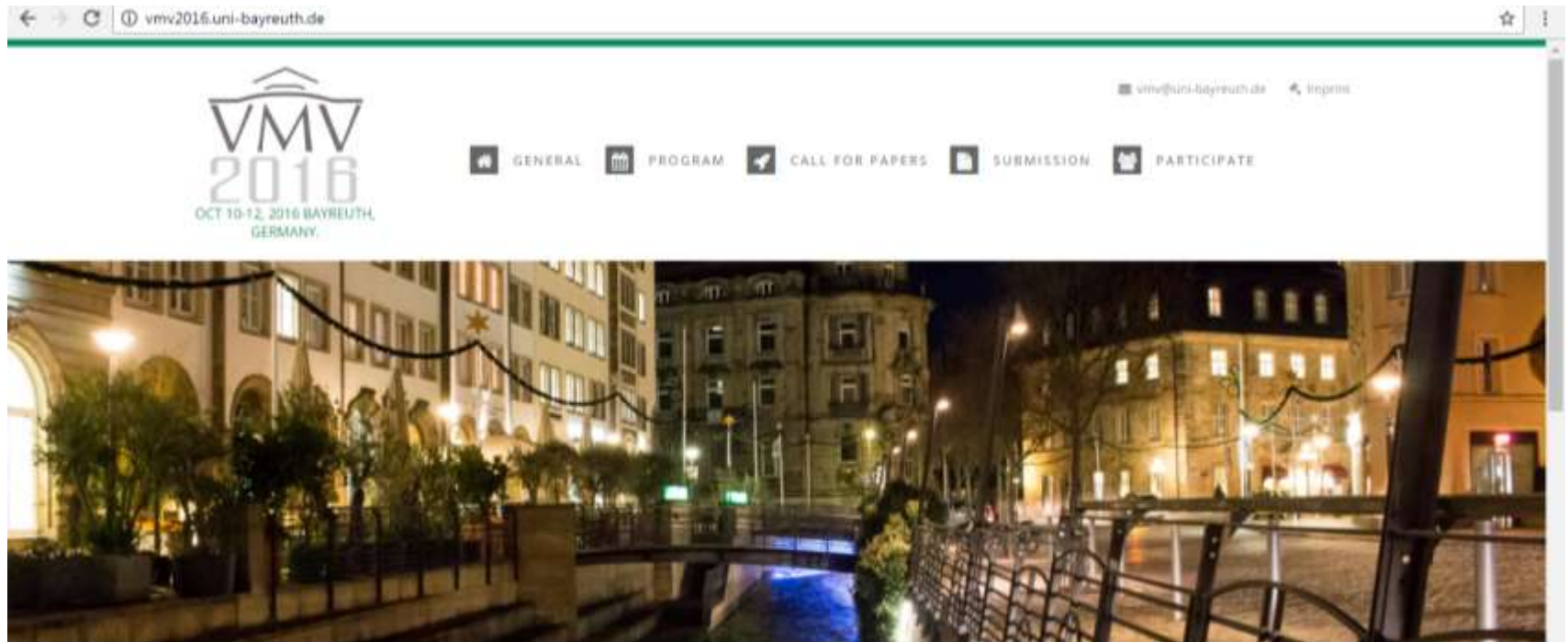
- Successfully dealing with **image degradation** requires modeling statistical dependencies between labels of neighboring pixels
- **Markov Random Fields** provide a convenient formalism, since they
 - Limit dependence to immediate neighbors
 - Allow for factorization using clique potentials
- **Maximum a posteriori (MAP)** estimation is usually addressed using approximative approaches
 - Iterated Conditional Modes (ICM) is a simple option
- Additional **prior knowledge** (e.g., from atlas) can be included

Announcements

- Nobody contacted us about looking for a group for project 2
 - This means you are free to form groups of three
- Due to a lack of electricity in the B-IT building, we have to cancel the lecture next week (January 17)
 - We will adapt the assignment sheet accordingly to include self-study

Job Opportunity

- We are looking for an experienced **web designer** who is willing to set up a relatively simple page for the VMV 2017 symposium



WELCOME TO VMV 2016!

Welcome to the official website of the 21st International Symposium on Vision, Modeling and Visualization (VMV 2016).

The VMV Symposium is the annual symposium of the Computer Graphics Special Interest Group of the German Informatics Society (Jahrestagung des

Further Reading: Markov Random Fields

- Y. Zhang et al.: *Segmentation of Brain MR Images Through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm*. IEEE Trans. Medical Imaging 20(1):45-57, 2001
- W.M. Wells, III et al.: *Adaptive Segmentation of MRI Data*. IEEE Trans. Medical Imaging 15(4):429-442, 1996
- R. Guillemaud, R. Brady: *Estimating the Bias Field of MR Images*. IEEE Trans. Medical Imaging 16(3):238-251, 1997