Mixture Models and Expectation-Maximization (EM)

Prof. Nicholas Zabaras
University of Notre Dame
Notre Dame, IN, USA

Email: nzabaras@gmail.com
URL: http://www.zabaras.com/

November 1, 2017

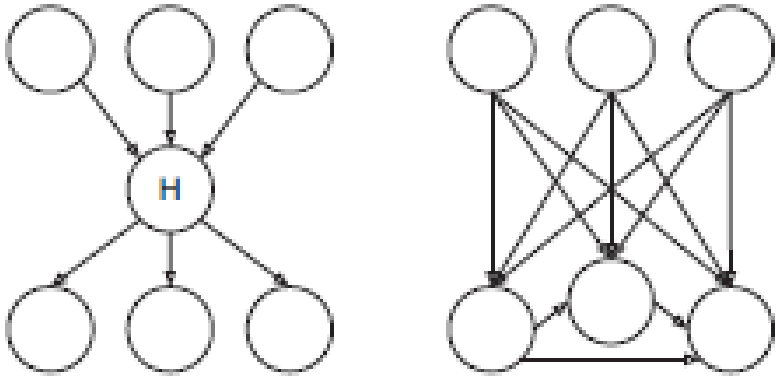
Contents

- Latent Variable Models, Introducing ideas with no probabilistic K-means clustering, K-Means, Image Compression
- Mixtures of Gaussians, responsibilities and latent variable view
- Mixtures of Bernoulli distributions, Bayesian Linear Regression, Mixture of Experts, Application to Inverse Problems, Parameter Estimation for Mixture Models and Identifiability, Computing MLE/MAP for Mixture Models is a Non-Convex Problem , EM for DGM with Hidden Variables, EM For the Student Distribution, EM For Probit Regression, MAP Estimation, EM for Bayesian Linear Regression, EM for Relevance Vector Machine
- Variational Inference, Generalizations of EM, Expectation Conditional Maximization, Incremental EM, Variational EM, Monte Carlo EM
- Fitting with Missing Data, Model Selection for Latent Variable Models

- Bishop CM, *Pattern Recognition and Machine Learning*, Springer, 2006 (Chapter 8)
- Murphy, K., Machine Learning: A Probabilistic Perspective (Chapter 11)
- M. Jordan, An Introduction to Graphical Models, unpublished (Chapters 9 and 10)

Latent Variable Models (LVMs)

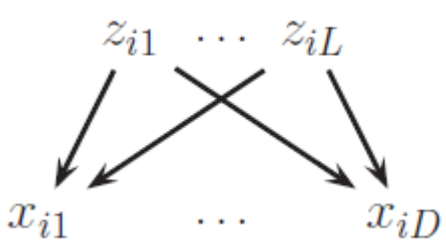
- To define high-dimensional joint probability distributions, we often assume that *the observed variables are correlated because they arise from a hidden common cause*.
- Introduce distinctions to simplify relations between observables. Viewing a distinction as a discrete random variable leads to mixture models.
- Models with hidden variables are called ***latent variable models***.
- LVMs are harder to fit than models with no latent variables.
- However, *LVMs have significant advantages*.
  - They often have fewer parameters than models that directly represent correlation in the visible space.
  - Consider below that all nodes are binary (including the hidden variable H) and all CPDs are tabular. The model on the left has 17 free parameters, whereas the model on the right has 59 free parameters.*

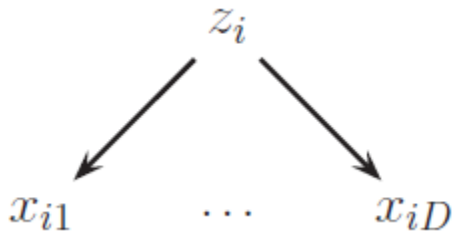


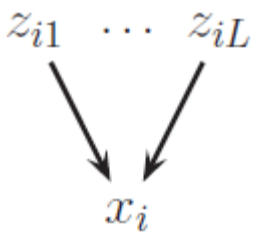
- In counting, note that $p(X_i)$ needs one parameter, $p(X_i|X_j)$ needs 2 parameters, $p(X_i|X_j,X_k,X_l)$ 8 parameters, etc.

Latent Variable Models (LVMs)

- LVMs have significant advantages.
  - The hidden variables in an LVM can serve as a **bottleneck**, which computes a compressed representation of the data (unsupervised learning).
- The general representation is shown on the right.
  By allowing $\mathbf{z}_i$ and/or $\mathbf{x}_i$ to be vector-valued, this representation can take multiple forms (many-to-many, one-to-many, many-to-one,one-to-one)







- Depending on the form of the likelihood $p(\mathbf{x}_i|\ \mathbf{z}_i)$ and the prior $p(\mathbf{z}_i)$, one can generate various models.

Gaussian Mixture Models and EM

- Roadmap:
  - Mixture of Gaussians
  - EM (informal derivation)
  - Lower bound viewpoint
  - EM revisited

- Introduce latent variables (for now discrete). Graph on an *enhanced space with conditionals all being Gaussians.*

- When you marginalize the distribution (getting rid of the latent variables), the distribution obtained is much more complex (not Gaussian)

Mixture Models and EM

- Mixture modeling can be viewed as a divide-and-conquer approach to statistical modeling.
- Additional latent variables allows to express complex marginal distributions over latent variables in terms of more tractable joint distributions over the expanded space.
- Maximum-Likelihood estimator in such a space is the Expectation-Maximization (EM) algorithm.
  - Finding clusters in a set of data points: K-Means corresponds to non-probabilistic limit of EM applied to Gaussian mixtures.
  - *The discrete latent variables define assignments of data points to specific components of the mixture.*

- Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory* **28**(2), 129–137.

Bayesian Learning

- Introduce prior distributions over parameters $p_i(\theta_i)$
- Equivalent graph with additional hidden variables
- Learning becomes inference on the expanded graph
- No distinction between variables and parameters
- Bayesian treatment uses variational inference
  - It resolves issues with MLE
  - Allows the number of components in a Gaussian mixture to be computed automatically from the data.

K-Means Clustering

K-Means Clustering: Distortion Measure

- Consider dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of a random D-dim Euclidean variable $\mathbf{x}$.
- Partition in K clusters
- Cluster prototype: $\boldsymbol{\mu}_k$ (center of cluster k)
- Main problem: Assign data to clusters & find $\boldsymbol{\mu}_k$
- Introduce binary indicator variable, 1-of-K Coding scheme

$$r_{nk} \in \{0,1\}$$

$$r_{nk} = 1 \; and \; r_{nj} = 0 \; for \;\; j \neq k$$

This is a hard assignment.

- Distortion measure $$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| \boldsymbol{x}_n - \boldsymbol{\mu}_k \right\|^2$$

Find: r_{nk} & μ_j to minimize J

K-Means Clustering: Expectation Maximization

- Find values for $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$ to minimize:

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \left\| \boldsymbol{x}_n - \boldsymbol{\mu}_k \right\|^2$$

- Use an iterative procedure:
  - Minimize J wrt r_{nk} while keeping $\{\boldsymbol{\mu}_k\}$ fixed (Expectation)

$$r_{nk} = \begin{cases} 1 & if \;\; k = \arg\min_j \left\| \boldsymbol{x}_n - \boldsymbol{\mu}_j \right\|^2 \\ 0 & otherwise \end{cases}$$

  - Minimize J wrt $\{\boldsymbol{\mu}_k\}$ while keeping $\{r_{nk}\}$ fixed (Maximization)

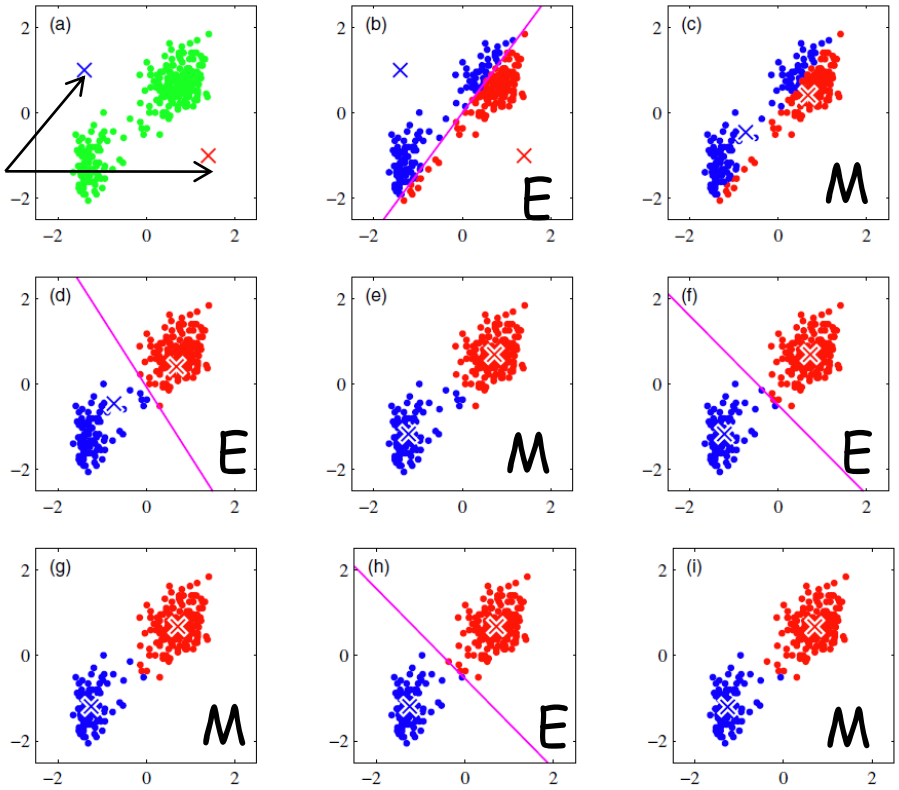
$$\boldsymbol{z}\left(\boldsymbol{\mu}_k\right) = -2\sum_{n=1}^{N} r_{nk}\left(\boldsymbol{x}_n - \boldsymbol{\mu}_k\right) = 0 \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

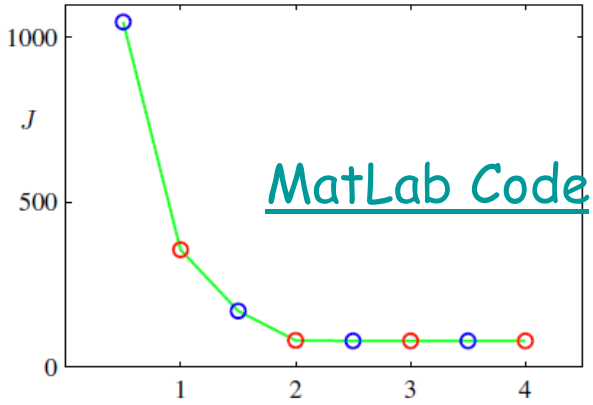
Mean of all points assigned to cluster k

K-Means Clustering: Example

- Iterate the E and M steps
- Each E or M step reduces the value of the objective function J.
- Convergence to a global or local minimum*

Initial guess of centers (use a random subset of $\mathbf{x}_n$)





- Old Faithful data (K=2)
- *Data standardized (zero mean and unit std)*
- *Classification of points based on which side they lie of the perpendicular bisector of the 2 cluster centers*

- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume I, pp. 281–297. University of California Press.

K-Means Clustering: Concluding Remarks

- The K-means is often used to initialize the parameters in a Gaussian mixture model before applying the EM algorithm
- Direct implementation of K-Means can be slow (in E-Step need to compute the distance $\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\right\|, \forall n, j$ at a cost of $\mathcal{O}$(NKD))
- For speeding up, some implementations precompute a data structure e.g. a tree such that nearby points are in the same subtree[1,2].
- Other approaches use of the triangle inequality for distances, thereby avoiding unnecessary distance calculations [3,4].

1. Ramasubramanian, V. and K. K. Paliwal (1990). A generalized optimization of the *k-d* tree for fast nearest-neighbour search. In *Proceedings Fourth IEEE Region 10 International Conference (TENCON' 89)*, pp. 565–568.
2. Moore, A. W. (2000). The anchors hierarch: using the triangle inequality to survive high dimensional data. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 397–405.
3. Hodgson, M. E. (1998). Reducing computational requirements of the minimum-distance classifier. *Remote Sensing of Environments* **25**, 117–128.
4. Elkan, C. (2003). Using the triangle inequality to accelerate *k*-means. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 147–153. AAAg

K-Means Clustering: Concluding Remarks

- Online learning (Robbins-Monro*, $\boldsymbol{z}\left(\boldsymbol{\mu}_k\right) = \partial J / \partial \boldsymbol{\mu}_k$):

$$\boldsymbol{\mu}_k^{new} = \boldsymbol{\mu}_k^{old} + \eta_n \left(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{old} \right),$$

η_n decreases as more data points are added

- K-means does not estimate the covariances –only the cluster means.

- A hard assignment of the Gaussian mixture model with general covariance matrices has been developed (*elliptical K-means*).

* For update of $\boldsymbol{\theta}$ using only one data point ,

$$\boldsymbol{\theta}^{(N)} = \boldsymbol{\theta}^{(N-1)} + a_{N-1}\boldsymbol{z}\left(\boldsymbol{\theta}^{(N-1)}\right)$$

- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman (Eds.), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume I, pp. 281–297. University of California Press.
- Sung, K. K. and T. Poggio (1998). Example-based learning for view-based human face detection. A.I. Memo 1521, MIT.

K-Means Clustering: Arbitrary Distances

- K-medoids algorithm: use general distortion (distance) measure

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \mathcal{V}(x_n, \mu_k)$$

where $\mathcal{V}(.,.)$ is *any kind of dissimilarity measure*

- E-Step Cost: $\mathcal{O}(KN)$, as in the standard K-means algorithm.

- M-Step Cost: Restrict the cluster centers to be equal to one of the data vectors assigned to that cluster (easy implementation for any $\mathcal{V}(\cdot, \cdot)$). Take each of the N_k points in the cluster k as the center of the cluster, then the M-step is to find the center that minimizes J – it requires $\mathcal{O}(N_k^2)$ evaluations of $\mathcal{V}(\cdot, \cdot)$.

- A final word: K-means does hard assignments, we need a probabilistic view (modeling uncertainty over each assignment)

K-Means: Image Segmentation

- Image segmentation: Partition an image into regions each of which has a reasonably homogeneous visual appearance.
- Treat each pixel as a separate data point.
- *Each pixel is a point in 3D comprising the intensities of the red, blue, and green channels*. This space is not Euclidean because the channel intensities are bounded by [0, 1].
- We can still apply K-means.
- In the figure shown in the next slide, *K-means* is run until convergence for different *K,* by redrawing the image replacing each pixel vector with the *{R, G,B}* intensity triplet given by the centre $\boldsymbol{\mu}_k$ to which that pixel has been assigned.
- The image is now represented with *K colors.*
- Note that K-means does not take into account spatial pixel correlation.

- Forsyth, D. A. and J. Ponce (2003). *Computer Vision: A Modern Approach*. Prentice Hall.

Image Segmentation and Compression

- Image segmentation. Smaller K *gives higher* compression at the expense of poorer image quality.

MatLab Code

$K = 2$ | $K = 3$ | $K = 10$ | Original image

Compression ratios compared to the original image

4.2% | 8.3% | 16.7% | 100%

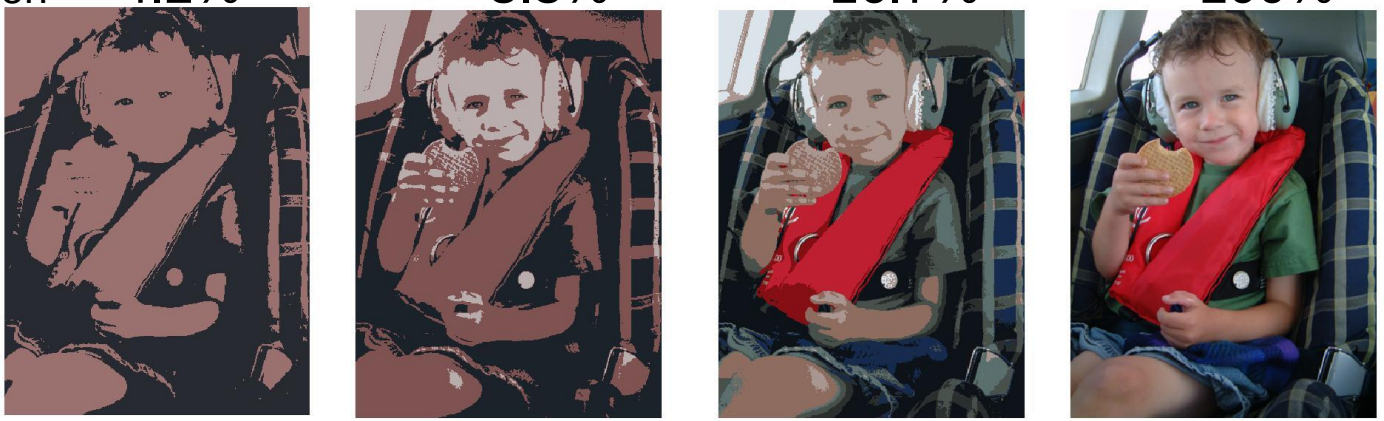


Image Compression, Vector Quantization

- The same K-means algorithm can be used for data compression:
  - Lossless data compression: reconstruct the original data exactly from the compressed representation, and
  - Lossy data compression: accept errors in reconstruction to allow higher levels of compression.
- For lossy data compression, for each data point, we store only the identity k of the cluster to which it is assigned. We also store the values of $\boldsymbol{\mu}_k$ – this requires less data for $K \leq N$.
- Each data point is then approximated by its nearest center $\boldsymbol{\mu}_k$.
- New data points can similarly be compressed by first finding the nearest $\boldsymbol{\mu}_k$ and then storing the label k instead of the original data vector.
- This is called vector quantization, and $\boldsymbol{\mu}_k$ are called code-book vectors.

Using Clustering for Data Compression

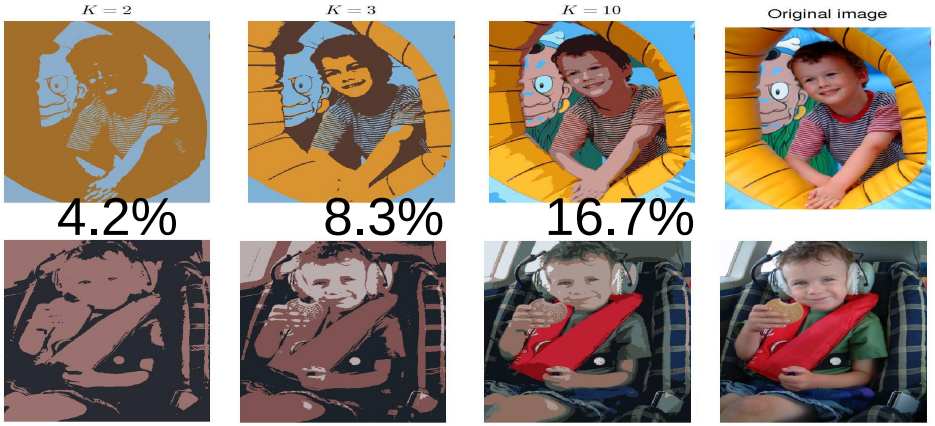
- Suppose the original image has *N* pixels comprising {R,G,B} values each of which is stored with 8 bits of precision.
- Direct transmission of the whole image requires 24N bits.
- With K-means instead of transmitting the original pixel intensity vectors we transmit the identity of the nearest $\boldsymbol{\mu}_k$.
  - Because there are *K* such vectors, this requires $log_2 K$ bits per pixel.
- We must also transmit the *K* vectors $\boldsymbol{\mu}_k$, which requires 24K bits.
- The total number of bits required to transmit the image is

$$24K + N \log_2 K$$

(rounding up to the nearest integer).

Using Clustering for Data Compression

- The image shown earlier had 240 × 180 = 43,200 pixels and requires 24 × 43,200 = 1,036,800 bits to transmit directly.
- The compressed images require 43,248 bits (K = 2), 86,472 bits (K = 3), and 173,040 bits (K = 10), respectively, to transmit.
- The compression ratios compared to the original image are 4.2%, 8.3%, and 16.7%, respectively.
- Trade-off between degree of compression & image quality.
- To produce a good image compression we need to exploit correlations in natural images in nearby pixels.

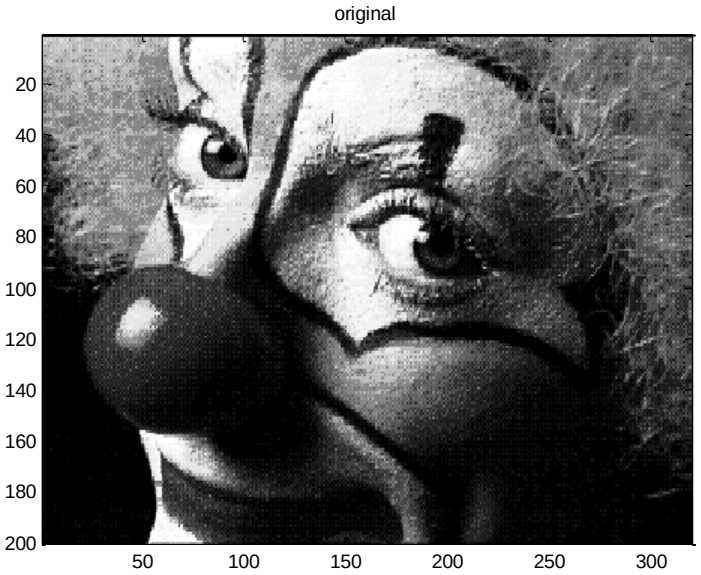


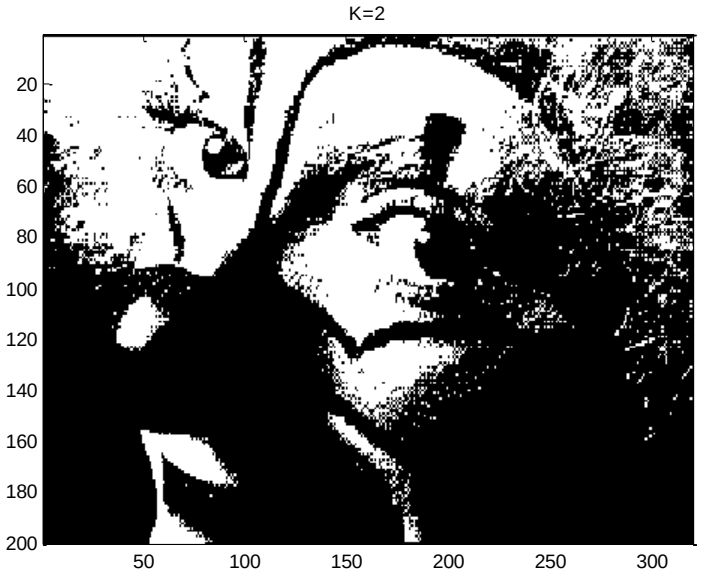
MatLab Code

Image Compression Using Vector Quantization

- K-Means is not maximizing the likelihood. It is a greedy algorithm minimizing a loss function related to data compression.
- Consider a $N = 200 \times 320 = 64{,}000$ pixel gray-scale (D=1) image. If we use 1 byte to represent each pixel (a gray-scale intensity of 0 to 255), then $C = 8$ (# bits to represent a scalar), so we need $NC = 512,000$ bits for the image.
- For the compressed image, we need $N \log_2 K + KC$ bits. For $K = 4$, this is ~128kb, factor of 4 compression. For $K = 8$, ~192kb, factor of 2.6 compression (negligible loss).
- Greater compression could be achieved if we model spatial correlation.

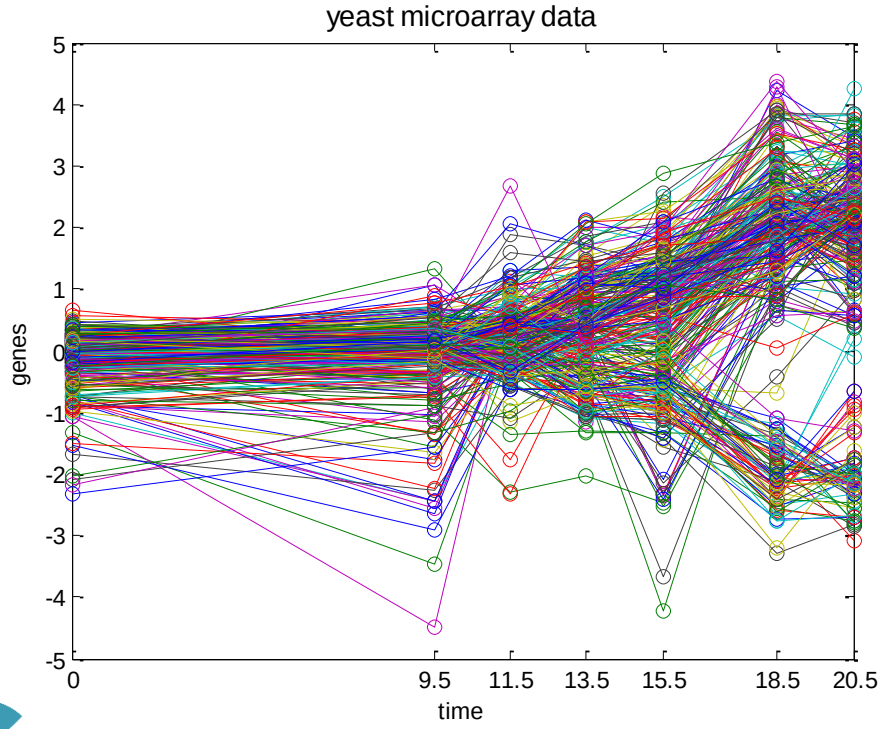
vqDemo from PMTK

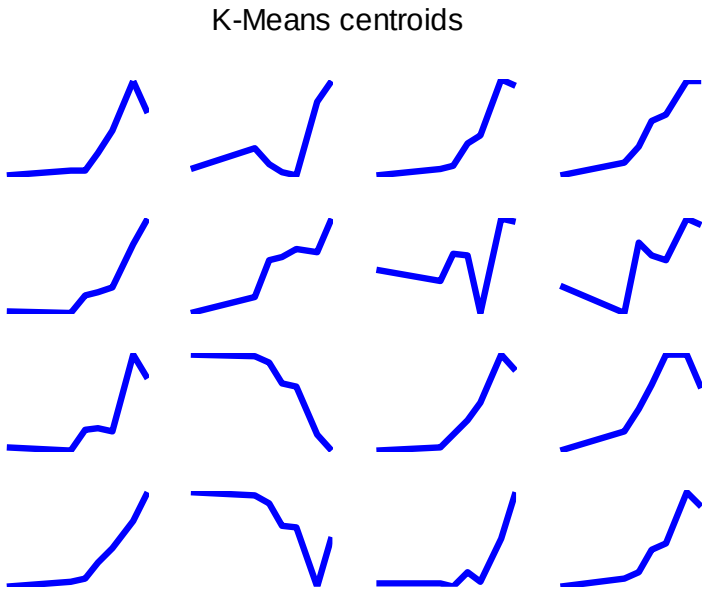




K-Means Example: Clustering of Genes

- Consider $\mathbf{x}_i \in \mathbb{R}^7$ representing the expression levels of different genes at 7 different times. We clustered them using a GMM.
- There are several kinds of genes: those whose expression level goes up monotonically over time (in response to a given stimulus); those whose expression level goes down monotonically; and more complex response patterns. We clustered into $K = 16$ groups.
- Each cluster is represented by a **prototype** or **centroid**.





kmeansYeastDemo from PMTK

Gaussian Mixture Model

The Gaussian Distribution

- Recall the multivariate Gaussian

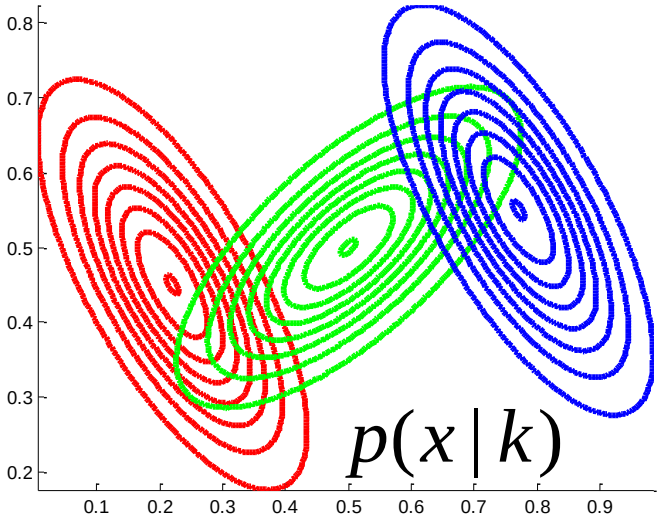
$$\mathcal{N}\left(\boldsymbol{x}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\frac{1}{\left(2\pi\right)^{D/2}\,|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)^{T}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}\right)\right\}$$

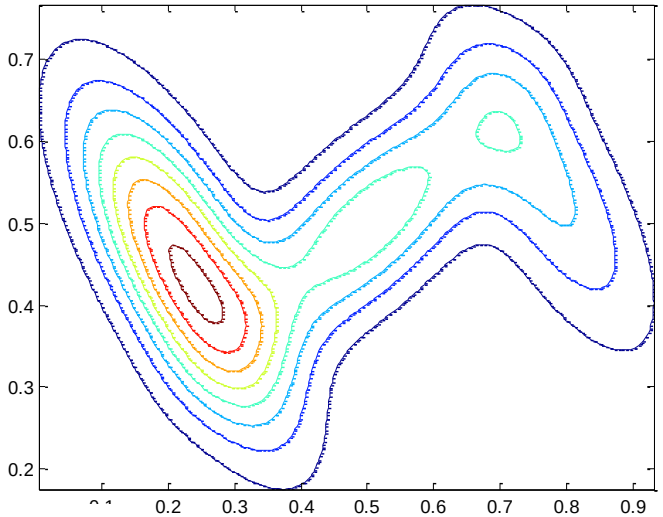
- The Maximum likelihood estimates of the mean and variance are:

$$\boldsymbol{\mu}_{ML}=\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_{n}$$

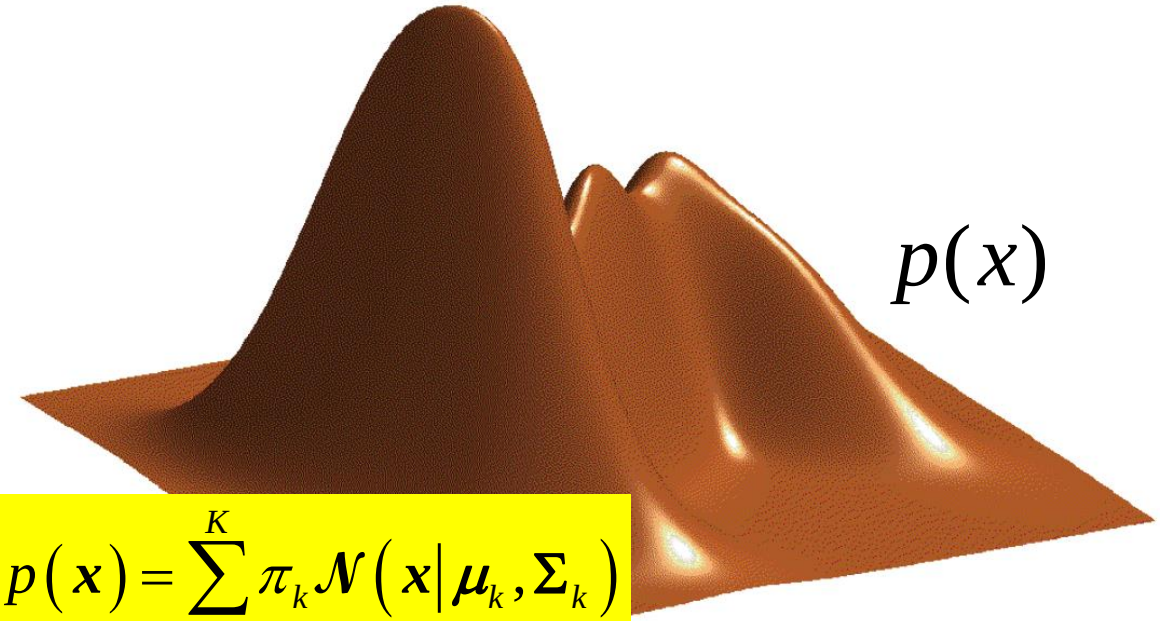
$$\boldsymbol{\Sigma}_{ML}=\frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{x}_{n}-\boldsymbol{\mu}_{ML}\right)\left(\boldsymbol{x}_{n}-\boldsymbol{\mu}_{ML}\right)^{T}$$

Example: Mixture of 3 Gaussians



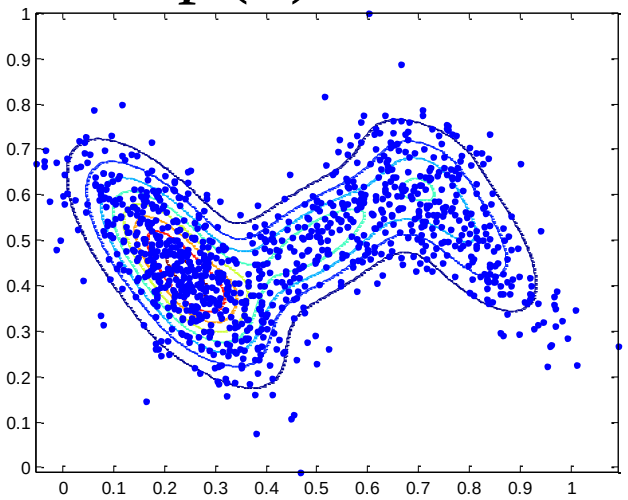


$p(x)$



$p(x)$

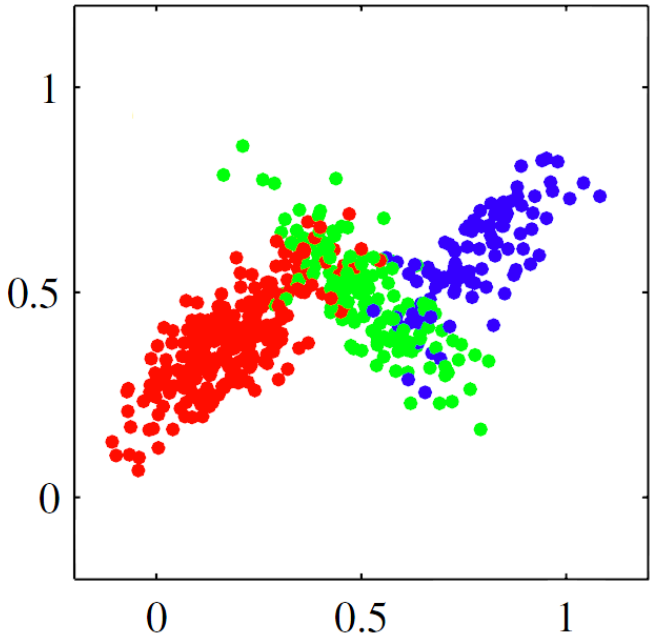
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x} \middle| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$$



mixGaussPlotDemo
from Kevin Murphys' PMTK

Sampling from a Gaussian Mixture

- To sample from a Gaussian mixture:
  - Pick one of the components with probability π_k
  - Draw a sample $\boldsymbol{x}_n$ from that component
  - Repeat the steps above for each new data point

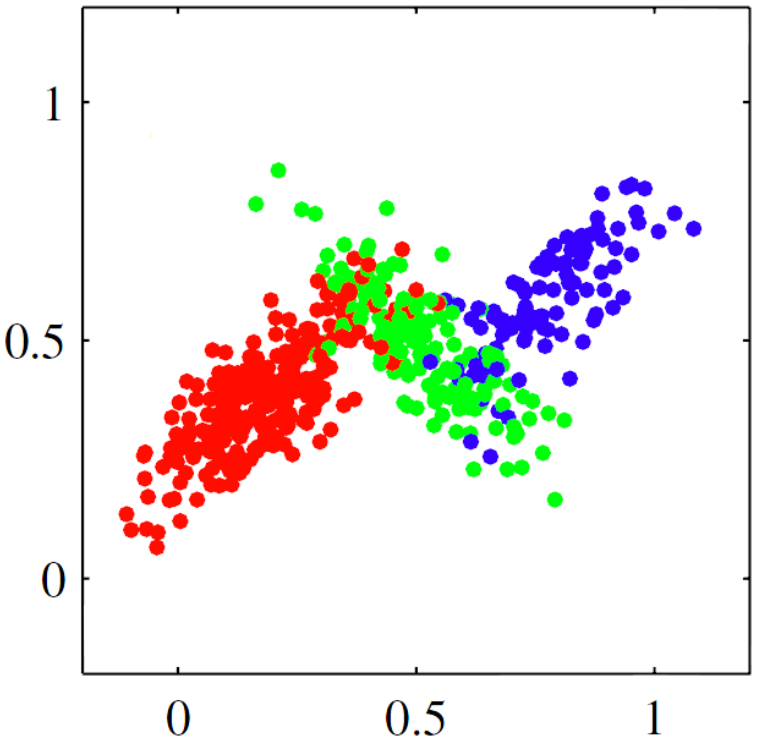


  - The data here (blue, green and red) are colored based on what component generated the data.

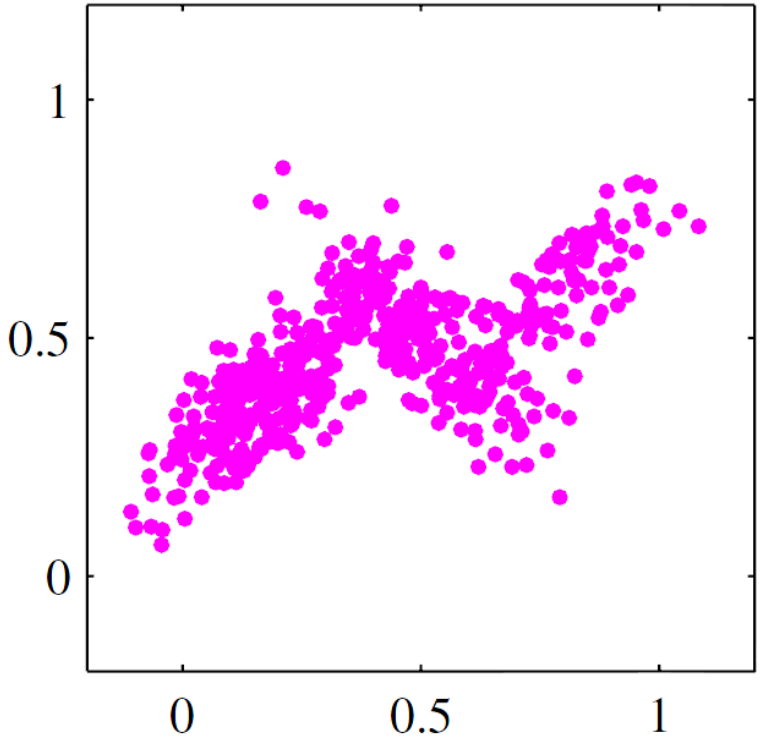
Latent Variable View of Gaussian Mixtures

- In practice, however, our goal is to solve the inverse problem:
  - Given a data set, find $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$
- Suppose we knew the colors
  - Maximum likelihood would involve fitting each component to the corresponding cluster
- Problem: the colors here represent latent (hidden) variables

Incomplete and Complete Data



Complete
Colored Data Points



Incomplete
Uncolored Data Points

Latent Variable Viewpoint of Gaussian Mixtures

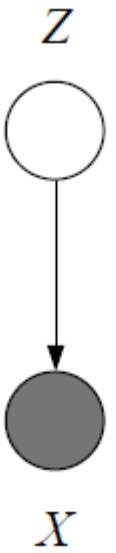
- Binary latent variables $z = \{z_{kn}\}$ describing which component in the mixture ($k=1,..,K$) generated each data point $\boldsymbol{x}_n$

For each data point : $z_k \in \{0,1\}, k = 1,...,K$ ($K = \#$ of colors)

$$\sum_k z_k = 1$$

- Example: For 3 components and 5 data points, the latent matrix is given as:

$$(z_{nk}) = \begin{array}{c} \begin{matrix} R & G & B \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \end{array} \quad \begin{matrix} \\ n=1 \\ \\ \\ \\ n=5 \end{matrix}$$



$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})\, p(\boldsymbol{x} \mid \boldsymbol{z})$

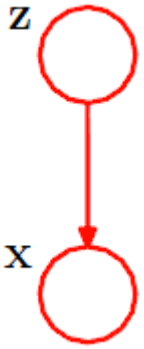
The latent variable z is a multinomial node taking one of K values.

Latent Variable Viewpoint of Gaussian Mixtures

- Gaussian Mixture Distribution:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- We introduce a latent variable **z**



$$(z_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

R G B

n=1

n=5

  - ***z*** is a binary 1-of-K coding scheme
  - $p(\boldsymbol{x},\boldsymbol{z})=p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$
  - $p(z_k=1)=p_k$ *(mixing coefficients)*

Constraints: $0 \le \pi_k \le 1, \text{ and } \sum_k \pi_k = 1, \; p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$

  - $p(\boldsymbol{x}|z_k = 1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

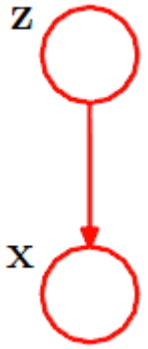
Latent Variable Viewpoint of Gaussian Mixtures

- Conditional distribution of observed variable (exponential family)

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

- Prior distribution of latent variables (exponential family)

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$



- Both distributions above are in the exponential family.
- The joint distribution is then:

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) = \prod_{k=1}^{K} \left(\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)^{z_k}$$

$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$

- Marginalizing over the latent variables (here K=3, thus for *z=(1,0,0), z=(0,1,0), (0,0,1))*, we obtain ($\mathbb{I}_{kj}$=1, k=j, otherwise 0)

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} \prod_{k=1}^{K} \left(\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)^{z_k} = \sum_{j=1}^{K} \prod_{k=1}^{K} \left(\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)^{\mathbb{I}_{kj}} = \sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- Using the joint *p(**x**,**z**)* leads to significant simplifications.

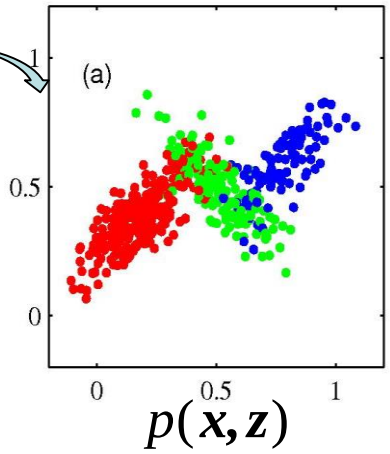
Latent Variable Viewpoint of Gaussian Mixtures

- Responsibility that component *k* takes for explaining the observation **x**:

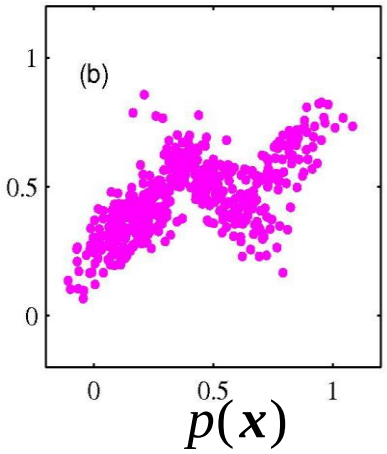
$$\gamma(z_k) \equiv p(z_k = 1 \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(z_k = 1 \mid \pi_k)\, p(\boldsymbol{x} \mid z_k = 1, \boldsymbol{\theta}_k)}{\sum_{k'} p(z_{k'} = 1 \mid \pi_{k'})\, p(\boldsymbol{x} \mid z_{k'} = 1, \boldsymbol{\theta}_{k'})} = \frac{\pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}$$

- This is the posterior probability (vs. the prior p_k). Note the relation with generative classification models.
- Generate random samples with ancestral sampling:
  - Generate: $\hat{\boldsymbol{z}}\ from\ p(\boldsymbol{z})$
  - Generate a value of $\boldsymbol{x}$ from: $p(\boldsymbol{x}|\hat{\boldsymbol{z}})$

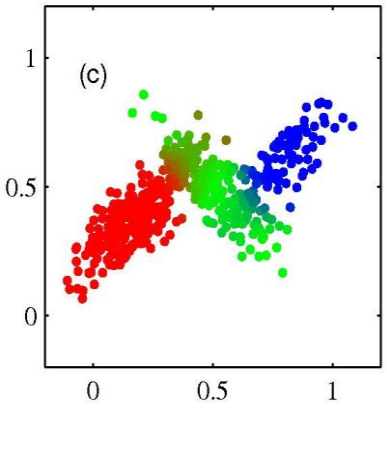
Points plot using their true identity



$p(\boldsymbol{x}, \boldsymbol{z})$



$p(\boldsymbol{x})$



Plotting each point using proportions of red, blue and green given by $\gamma(z_{nk})$,k=1,2,3

MatLab Code

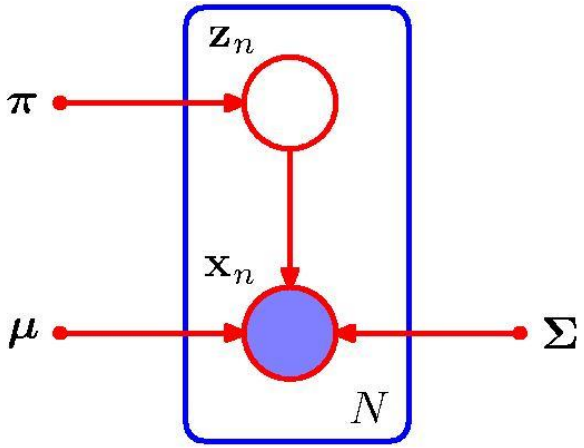
Maximum Likelihood for the Mixture of Gaussians

- Log likelihood function (N×D matrix of data set $\boldsymbol{X}$, N×K latent variable matrix of $\boldsymbol{Z}$)

$$\ln p\left(\boldsymbol{X}\middle|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\underbrace{\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}_{\mathcal{N}_{nk}}\right\}=\sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}_{nk}\right\}$$

- Sum over components appears inside the log!

- There is no closed form MLE solution



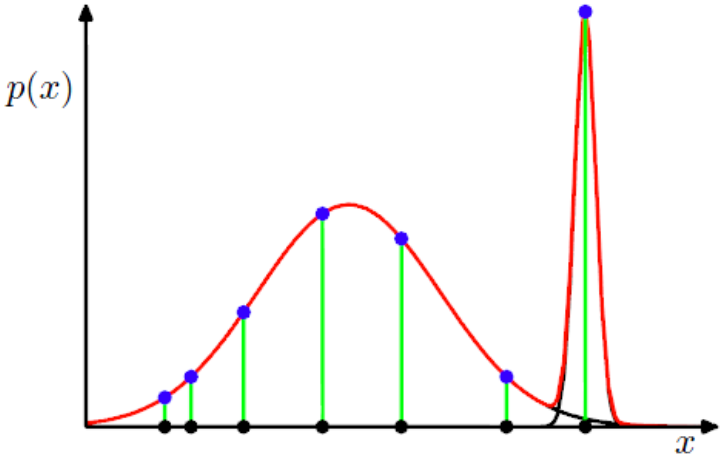
Problems in MLE for Gaussian Mixture Models

- Severe over-fitting in MLE: Infinities in likelihood function when a component 'collapses' onto a data point (mean set to a data point and variance goes to 0):

$$\mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{x}_n, \sigma_j^2 \boldsymbol{I}\right) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\sigma_j^D} \to \infty$$

with $\sigma_j \to 0$

The log-likelihood thus goes to infinity.



- When this happens, initialize the particular Gaussian with a random mean and a broad variance.
- One can also use MAP estimation instead.

Problems in MLE for Gaussian Mixture Models

- MLE cannot determine the number K of components: The higher the K, the higher the likelihood will be. This is a problem since MLE is a point estimate. That will not be the case in a Bayesian framework.
- Identifiability problem: K! equivalent solutions.[1]
- Gradient based optimization methods can be used for maximizing the likelihood. However, we concentrate next on the EM algorithm.[2,3,4]

1. Casella, G. and R. L. Berger (2002). *Statistical Inference* Second ed.). Duxbury.
2. Fletcher, R. (1987). *Practical Methods of Optimization* (Second ed.). Wiley.
3. Nocedal, J. and S. J. Wright (1999). *Numerical Optimization*. Springer.
4. Bishop, C. M. and I. T. Nabney (2008). *NetLab Algorithms for Pattern Recognition*. Springer. In preparation.

EM Algorithm for Gaussian Mixtures

$$\ln \mathcal{L} = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}_{nk} \right\} \quad \text{where : } \mathcal{N}_{nk} \equiv \mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right)$$

$$0 = \frac{\partial \ln \mathcal{L}}{\partial \boldsymbol{\mu}_j} = \sum_{n=1}^{N} \underbrace{\frac{\pi_j \mathcal{N}_{nj}}{\sum_k \pi_k \mathcal{N}_{nk}}}_{\text{Responsibilities: } \gamma(z_{nj})} \frac{1}{\mathcal{N}_{nj}} \frac{\partial \mathcal{N}_{nj}}{\partial \boldsymbol{\mu}_j}$$

$$= -\sum_{n=1}^{N} \gamma\left(z_{nj}\right) \boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{x}_n - \boldsymbol{\mu}_j \right) \Rightarrow \boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma\left(z_{nj}\right) \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)} \text{ (mean of the j component)}$$

- *Note that we don't have a closed form solution since the $\gamma(z_{nj})$ contain the $\boldsymbol{\mu}_j$'s in a complex way.*
- The responsibilities $\gamma(z_{nj})$ represent posterior probabilities.

EM Algorithm: Mixture of Gaussians

- The EM algorithm proceeds as follows:
  - E Step: Assume all the parameters and evaluate the $\gamma(z_{nj})$'s
  - M Step: Revise the parameters π_j, $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ using the $\gamma(z_{nj})$'s

- Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B* **39**(1), 1–38.
- McLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and its Extensions*. Wiley.
- McLachlan, G. J. and D. Peel (2000). *Finite Mixture Models*.
- Meng, X. L. and D. van Dyk (1997). The EM algorithm — an old folk song sung to a fast new tune (with Discussion). *J. Royal Stat. Soc. B 59*, 511–567.
- Csiszar, I. and G. Tusnady (1984). Information geometry and alternating minimization procedures. *Statistics and Decisions* **1**(1), 205–237.
- Hathaway, R. J. (1986). Another interpretation of the EM algorithm for mixture distributions. *Statistics and Probability Letters* **4**, 53–56.
- Neal, R. M. and G. E. Hinton (1999). A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 355–368. MIT Press.

EM Algorithm: Mixture of Gaussians – E Step

- *E step equation*: assume the μ_j and Σ_j and then

Compute the responsibilities:

$$\gamma\left(z_{nj}\right)=\frac{\pi_j \mathcal{N}_{nj}}{\sum_k \pi_k \mathcal{N}_{nk}}=\frac{\pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}{\sum_k \pi_k \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}$$

Responsibilities

- As we have seen, we interpret the mixing coefficients as prior probabilities

$$p(\boldsymbol{x}) = \sum_{j=1}^{K} \pi_j \mathcal{N}\left(x \middle| \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)$$

$p(j)$ $\qquad$ $p(\boldsymbol{x}|j)$

- The corresponding posterior probabilities are the responsibilities:

$$p(j|\boldsymbol{x}) = \frac{p(j)\,p(\boldsymbol{x}|j)}{p(\boldsymbol{x})} = \frac{\pi_j \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}{\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)} = \gamma_j(\boldsymbol{x})$$

- Think of $\gamma(z_{nj})$ as the posterior probability of j evaluated at the training data $\boldsymbol{x}_n$ – **the responsibility that the j^{th} Gaussian is taking in explaining point $\boldsymbol{x}_n$.**

EM Algorithm: Mixture of Gaussians – M Step

- *M step equations:* Re-estimate the parameters using the known responsibilities:

$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)\boldsymbol{x}_n}{N_j}, \; N_j = \sum_{n=1}^{N} \gamma\left(z_{nj}\right) \qquad \boldsymbol{\pi}_j = \frac{1}{N}\sum_{n=1}^{N} \gamma\left(z_{nj}\right)$$

$$\boldsymbol{\Sigma}_j = \frac{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j\right)^T}{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)}$$

- The computed variance is like a sample variance weighted by the responsibilities.

EM For Gaussian Mixtures

- Maximum of the log likelihood: derivatives of $\ln p\left(\boldsymbol{X}\middle|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right)$ wrt parameters to 0.

$$\ln p\left(\boldsymbol{X}\middle|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\sum_{n=1}^{N}\ln\sum_{k=1}^{K}\pi_k\mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)$$

- For the $\boldsymbol{\mu}_k$ we have already seen that:

$$0=\sum_{n=1}^{N}\frac{\pi_k\mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\sum_{k=1}^{K}\pi_k\mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n-\boldsymbol{\mu}_k)$$

from which:

$$\boldsymbol{\mu}_k=\frac{\sum_{n=1}^{N}\gamma\left(z_{nk}\right)\boldsymbol{x}_n}{N_k}=\frac{1}{\sum_{n}\gamma\left(z_{nk}\right)}\sum_{n=1}^{N}\gamma\left(z_{nk}\right)\boldsymbol{x}_n$$

EM For Gaussian Mixtures

- For the Σ_k:

$$\Sigma_k = \frac{1}{\sum_n \gamma(z_{nk})} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T$$

EM For Gaussian Mixtures

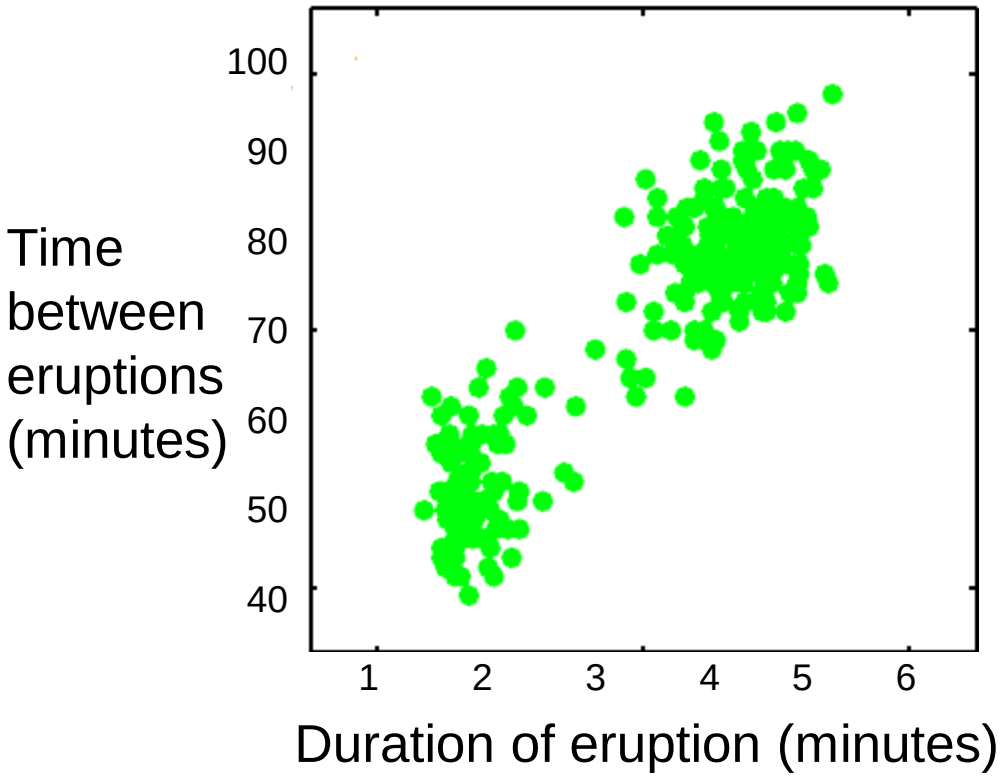
- For the π_k:
  - Take into account that $\sum_k \pi_k = 1$
  - Use Lagrange multiplier approach $\ln p\left(\mathbf{X}\middle|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}\right)+\lambda\left(\sum_k \pi_k - 1\right)$

$$0 = -\sum_{n=1}^{N}\frac{\mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\sum_{m=1}^{K}\pi_m\mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m\right)}+\lambda \Rightarrow 0 = -\sum_{k=1}^{K}\pi_k\sum_{n=1}^{N}\frac{\mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\sum_{m=1}^{K}\pi_m\mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m\right)}+\lambda\sum_{k=1}^{K}\pi_k = 0$$

$$\lambda = -N \Rightarrow 0 = -\sum_{n=1}^{N}\frac{\pi_k\mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)}{\sum_{k=1}^{K}\pi_k\mathcal{N}\left(\mathbf{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)} - N\pi_k \Rightarrow$$

$$\pi_k = \frac{\sum_n \gamma\left(z_{nk}\right)}{N} \equiv \frac{N_k}{N}$$

Old Faithful Data Set

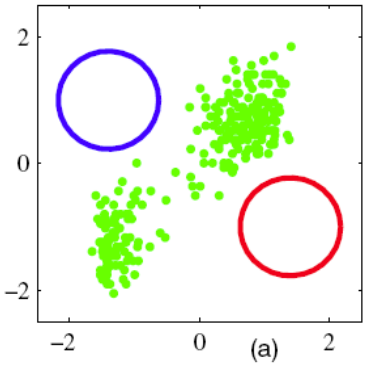


- The data are standardized – remove the mean and divide by the std.

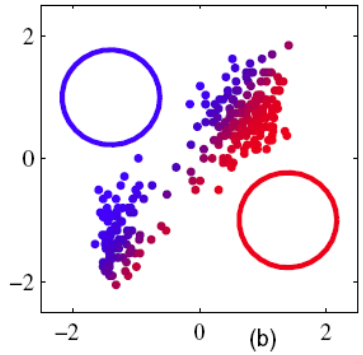
MatLab Code

EM Implementation: Mixture of Gaussians

- We initialize the problem with 2 Gaussians as shown here (mean + std).



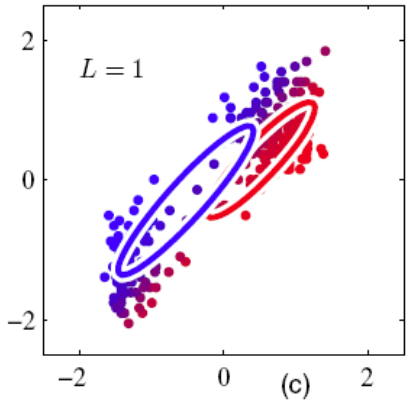
- In the E Step, we color the data points i.e. we compute the posterior probability for the red & blue components for every data point given the current parameters *(inferring the missing values given the parameters).*



- Note that the proportion of the blue ink in the points is defined by the posterior probability: $color(n) = \gamma(z_{n1})blue + \gamma(z_{n2})red$

EM Implementation: Mixture of Gaussians

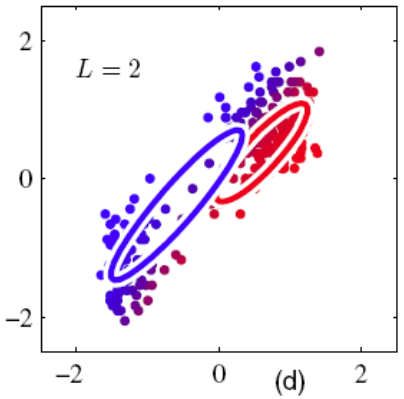
- In the M step, the γ's are fixed (fixed colors) and the parameters (mean and variance of the Gaussians) are updated:



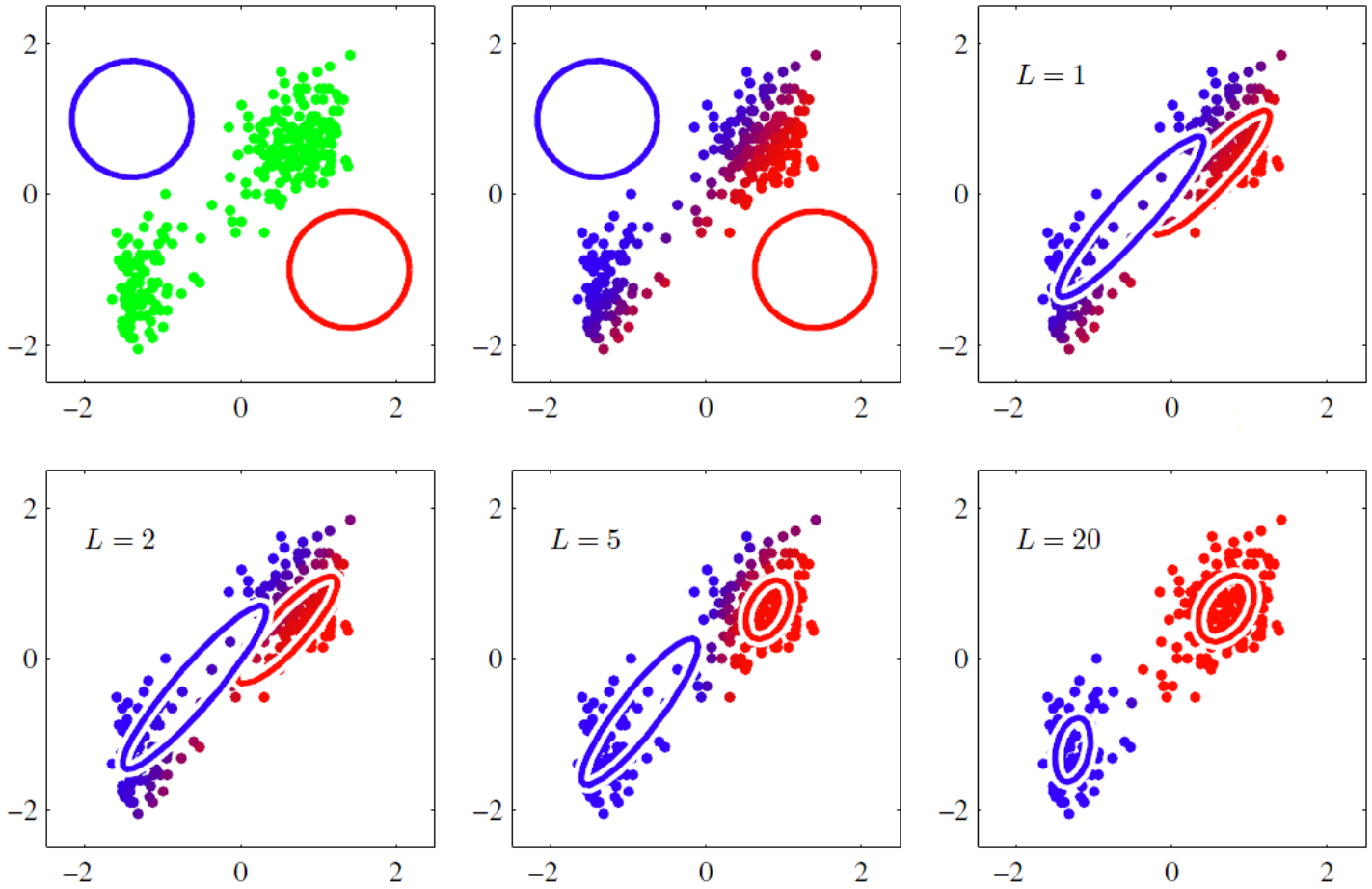
$$\boldsymbol{\mu}_j = \frac{\sum_{n=1}^{N} \gamma\left(z_{nj}\right) \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)}$$

$$\boldsymbol{\Sigma}_j = \frac{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j\right)\left(\boldsymbol{x}_n - \boldsymbol{\mu}_j\right)^T}{\sum_{n=1}^{N} \gamma\left(z_{nj}\right)}$$

- The mean of the blue component is fitted to all of the blue ink as if the red ink did not exist:
  - the E step decoupled the problem and in the M step we fit one Gaussian at a time.
- We next repeat the E step keeping the components fixed but recomputing the colors.

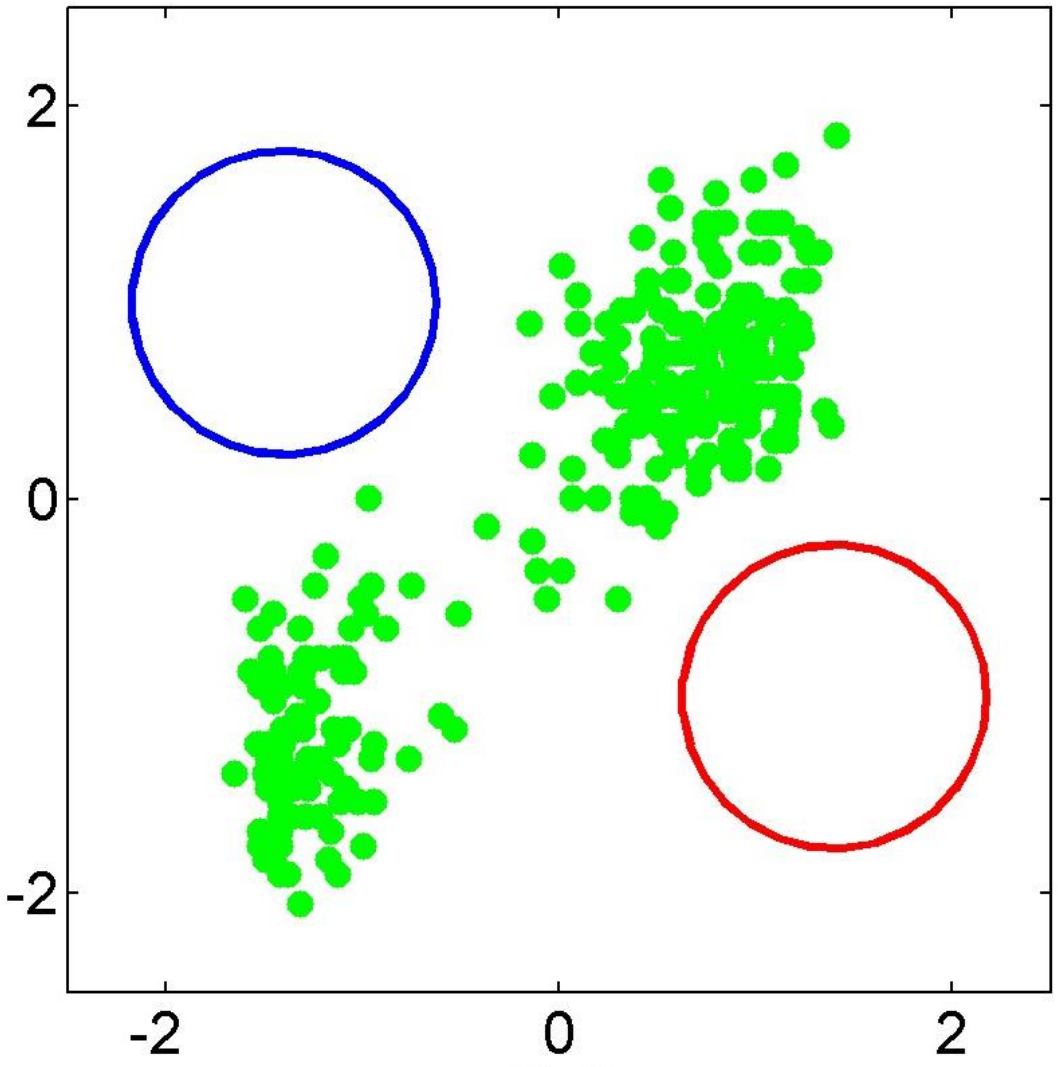


Old Faithful Data Set



- On the E step, we color the data points keeping the components fixed
- In the M step, we keep the colors the same but update the components (parameters). Work at one Gaussian at a time fitting the mean & covariance for each of them separately.

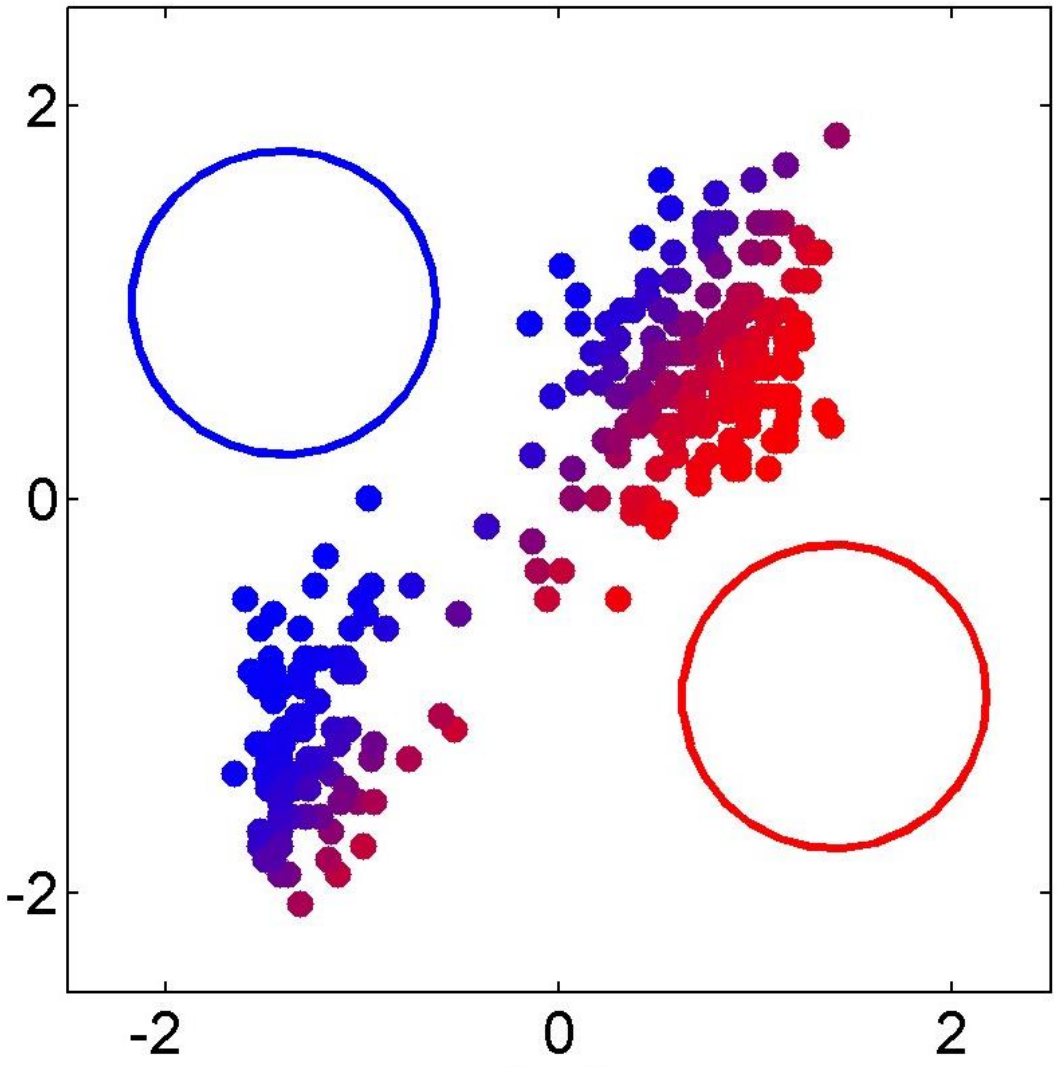
Old Faithful Data Set



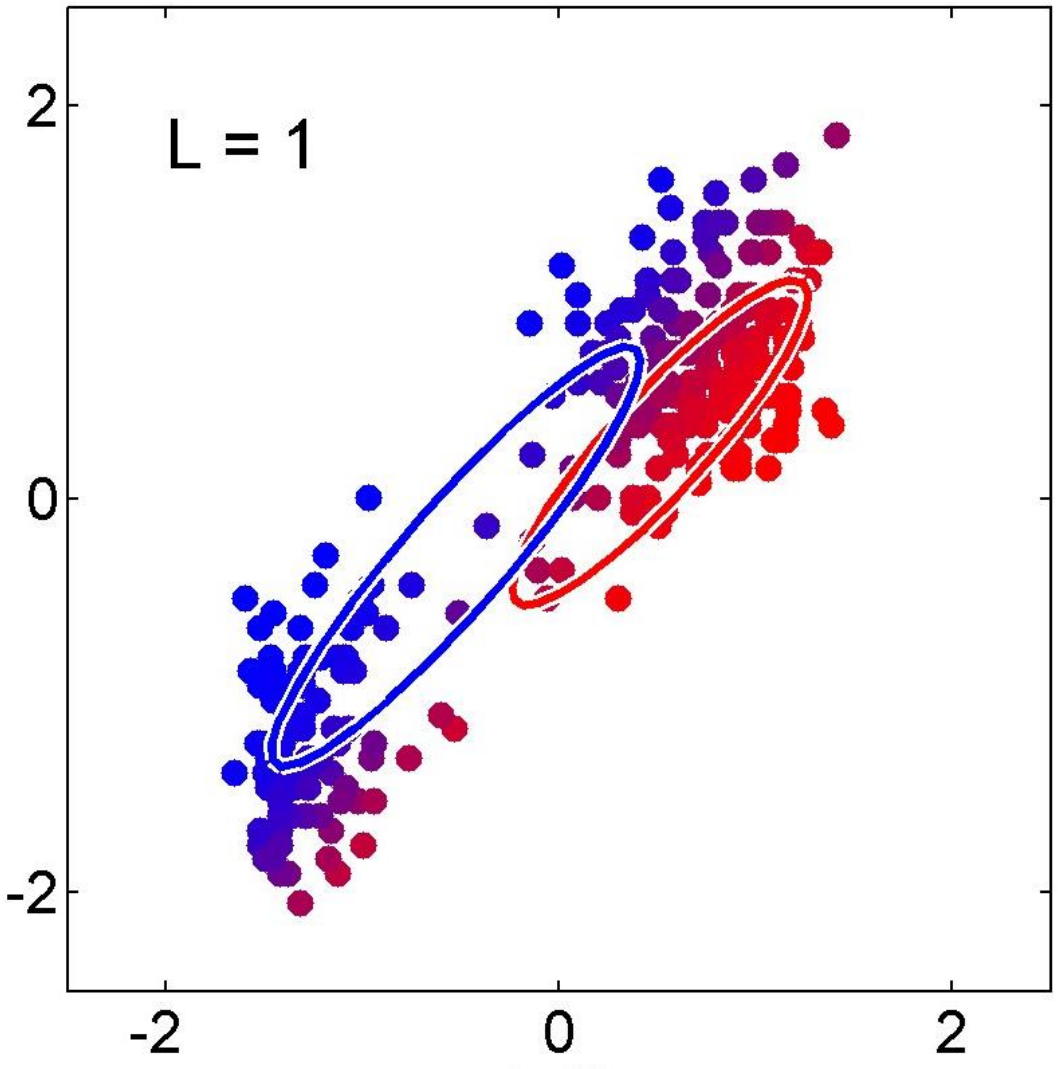
MatLab Code

Also see
mixGaussDemoFaithful
from PMTK

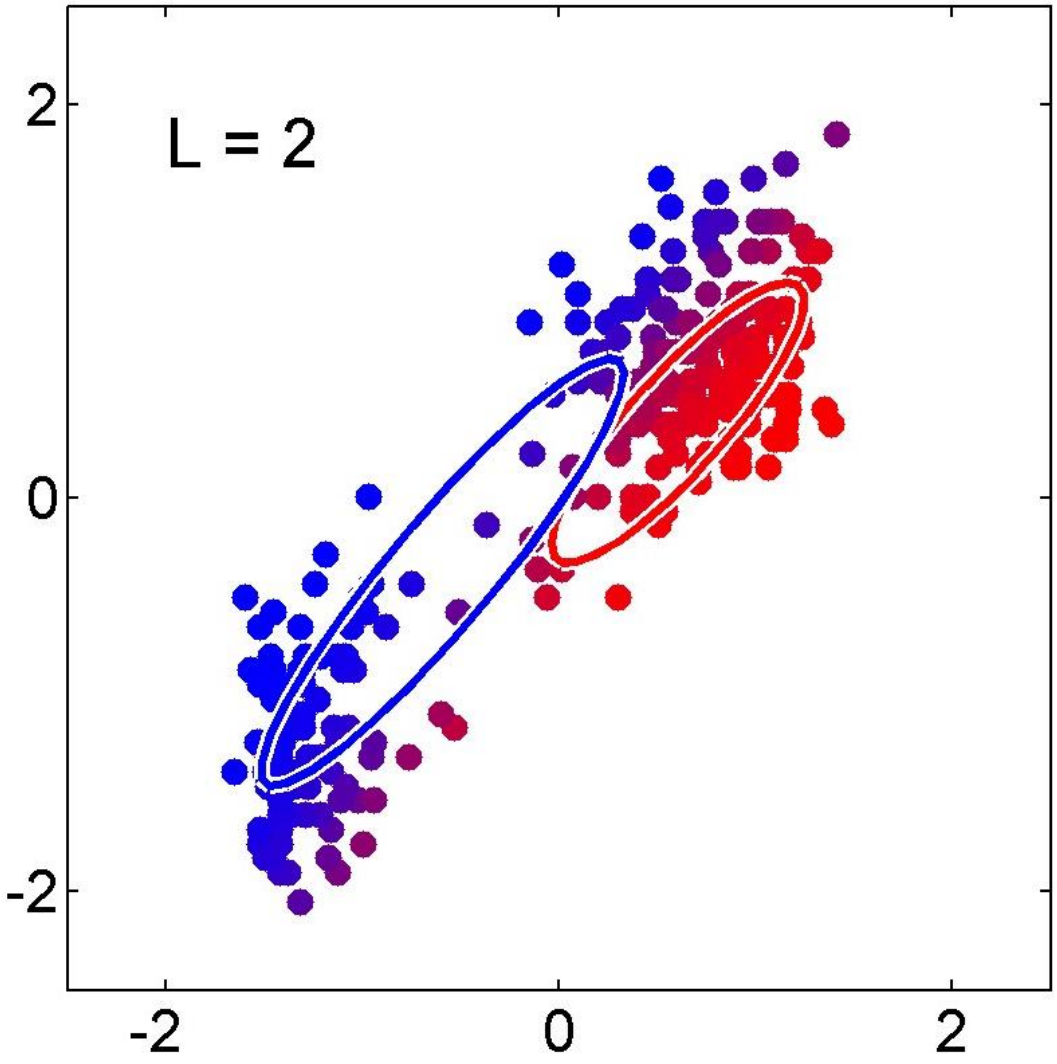
Old Faithful Data Set



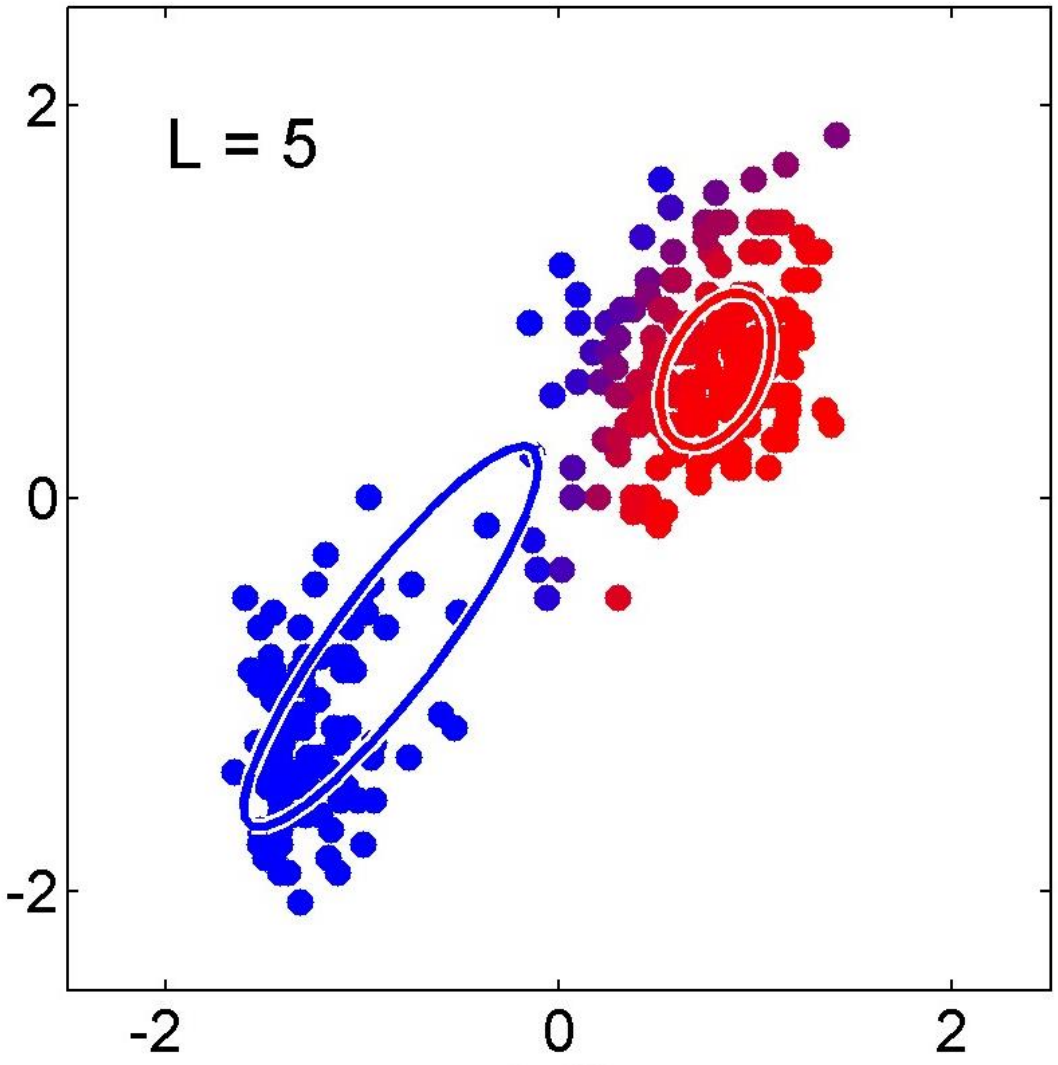
Old Faithful Data Set



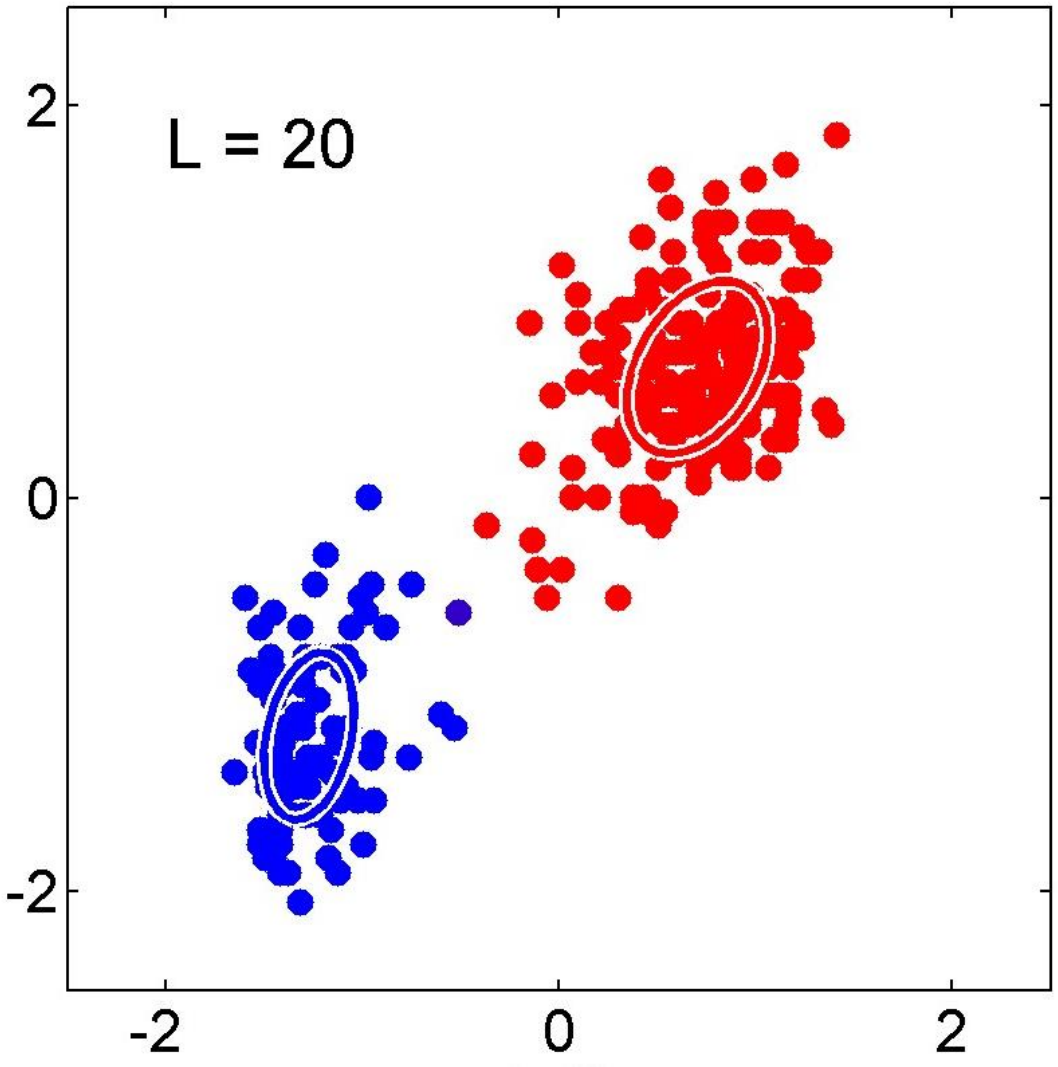
Old Faithful Data Set



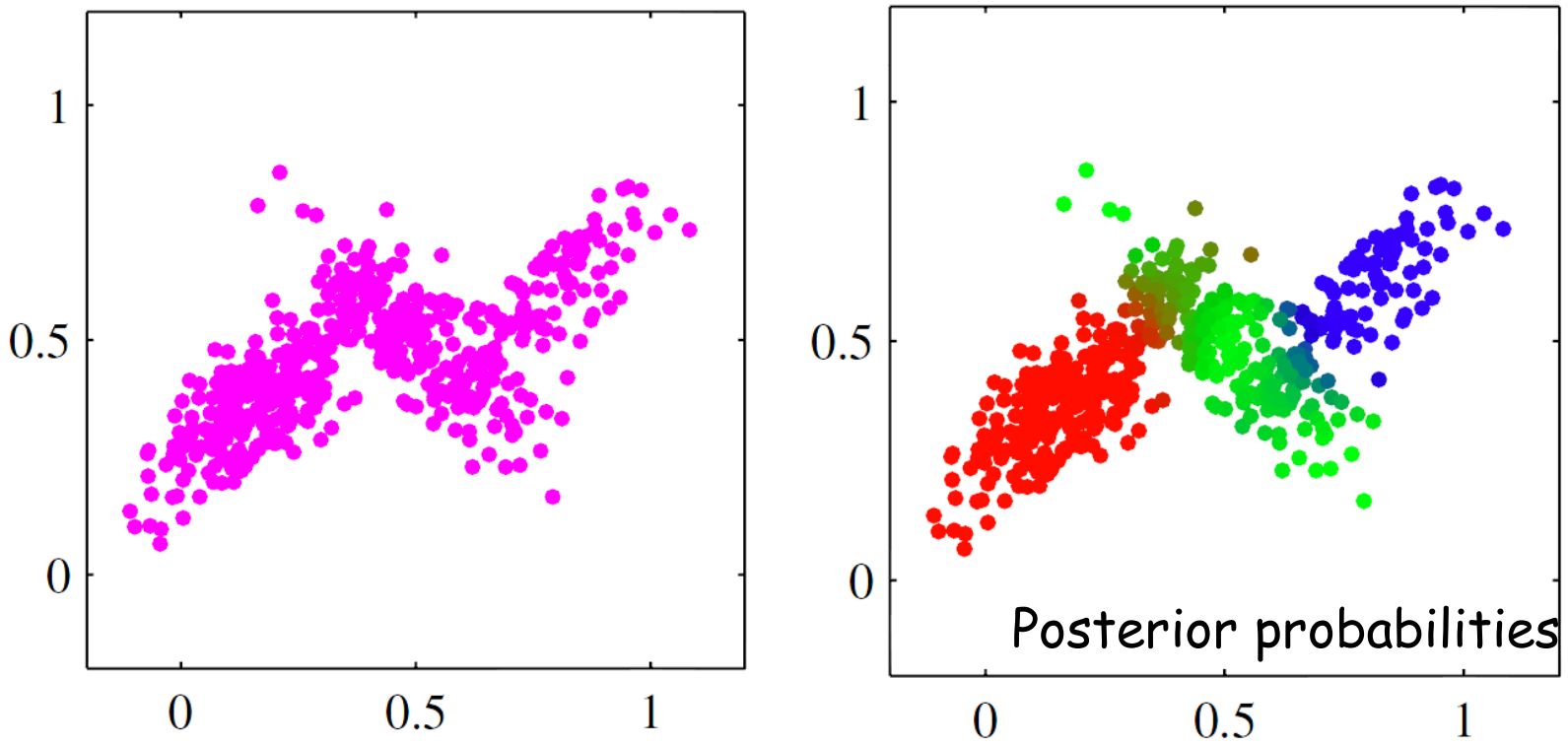
Old Faithful Data Set



Old Faithful Data Set



Posterior Probabilities (Color Coded)

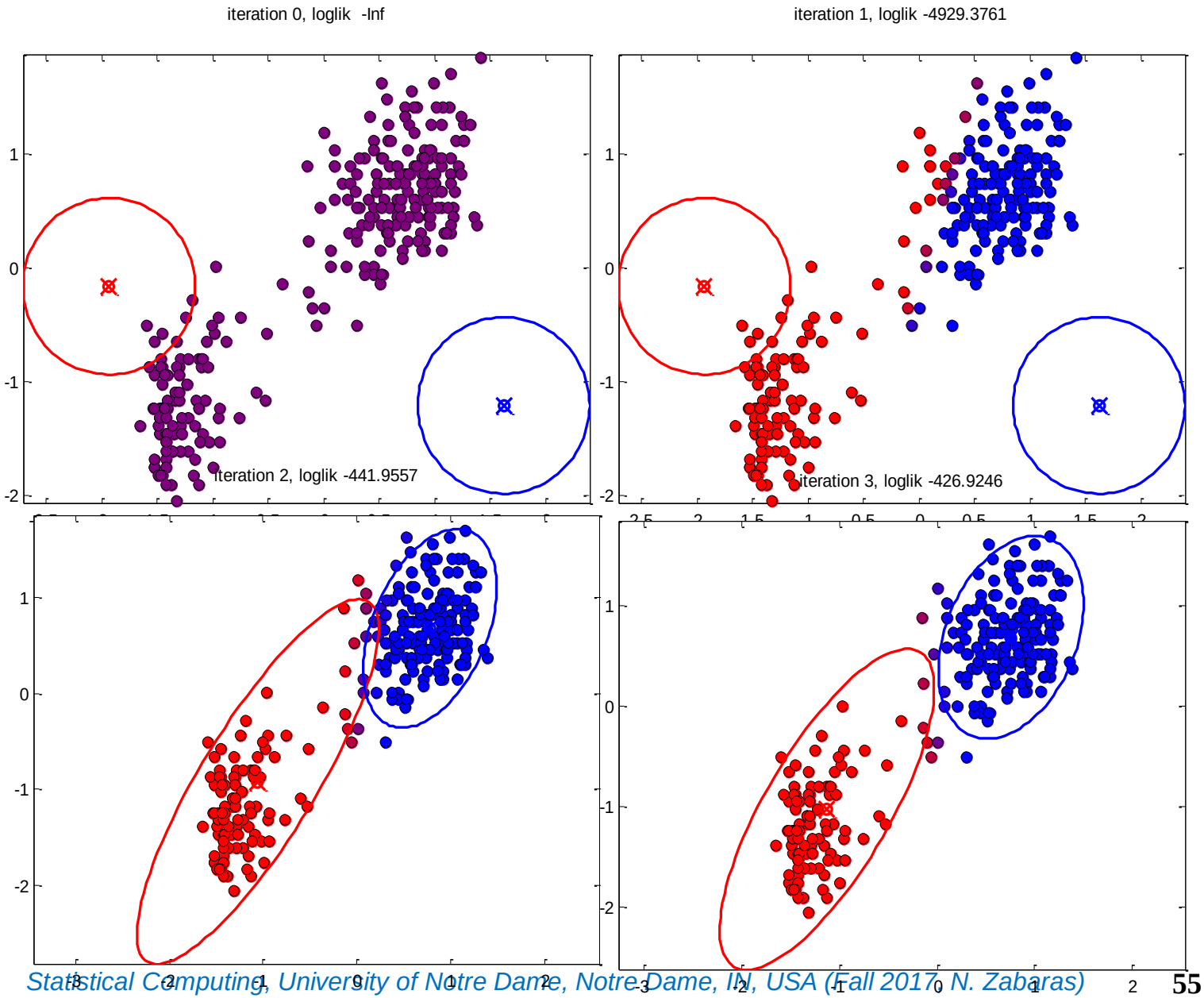


- Note that with the E Step components in the border area of e.g. blue and green or green and red are a bit ambiguous.
- In the M Step, we fix $\gamma(z_{nk})$ and maximize the expected log likelihood.

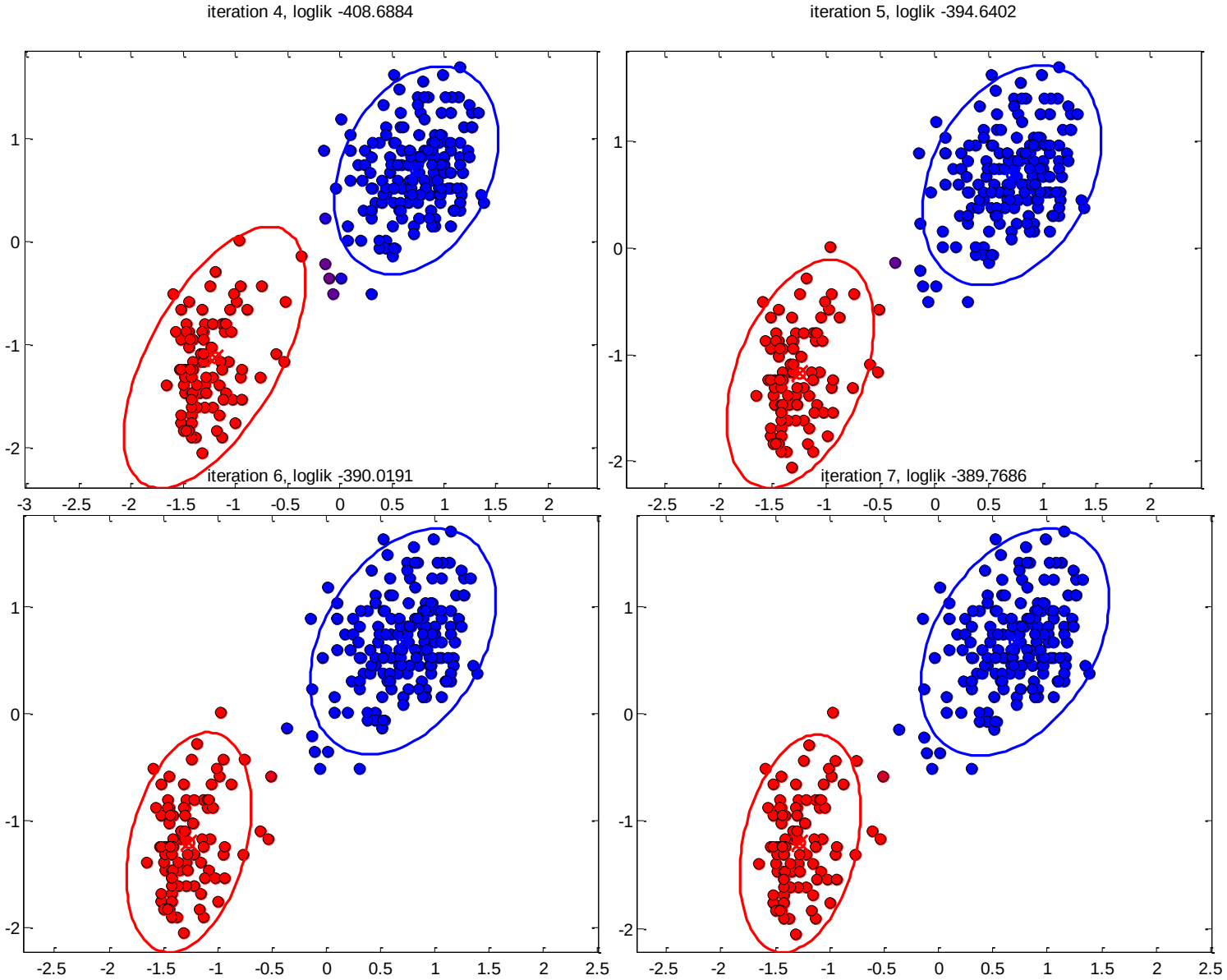
EM For Gaussian Mixtures

- No closed form solutions: $\gamma(z_{nk})$ depends on parameters
- The update equations suggest simple iterative scheme for finding maximum likelihood.
- Alternate between estimating the current $\gamma(z_{nk})$ and updating the parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$.
- More iterations needed to converge than K-means algorithm, and each cycle requires more computation
- Common, initialize parameters based on K-means run. Initialize covariance matrices to sample covariances of the clusters found by K-means.
- EM is not guaranteed to find the largest of the multiple local maxima.

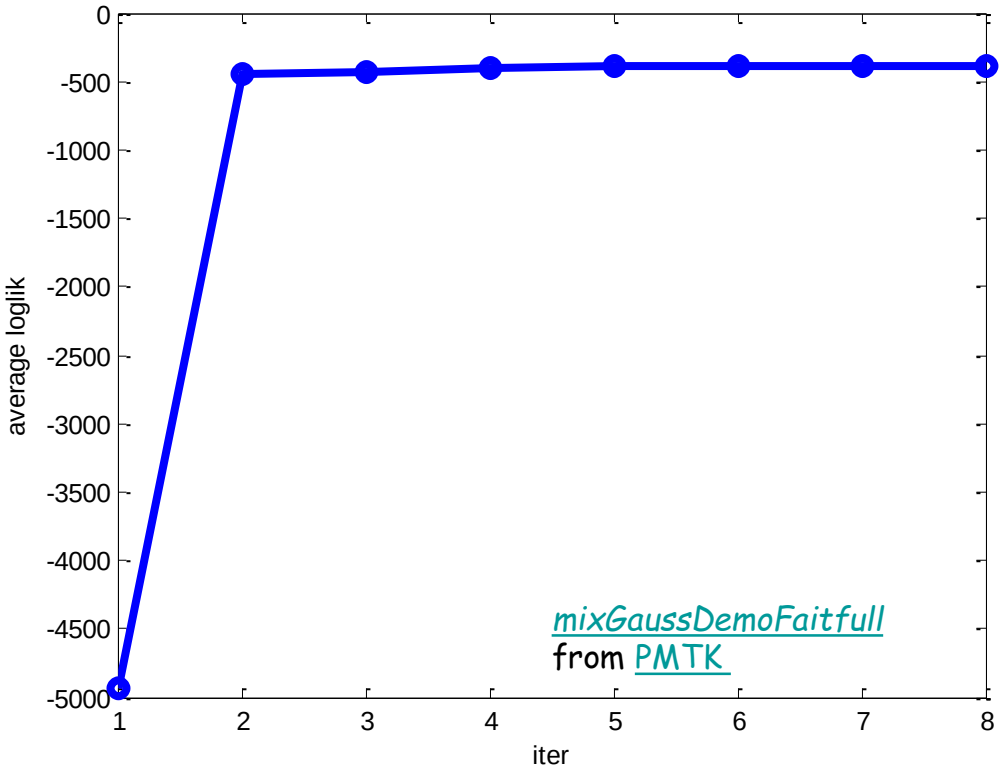
Old Faithful Data Set



Old Faithful Data Set



Old Faithful Data Set



Initialization

- Pick *K* data points at random (initial cluster centers) or
- One can pick the centers sequentially to try to cover the data.
  - Pick the initial point uniformly at random.
  - Each subsequent point is picked from the remaining points with probability proportional to its squared distance to the point's closest cluster center (*farthest point clustering, or k-means++).*
  - This trick guarantees that the distortion is never more than $\mathcal{O}(\log K)$ worse than optimal.
- The speech recognition community incrementally grows GMMs until the desired number of clusters is reached.
  - initially give each cluster a score based on its mixture weight;
  - after each round of training, split the cluster with the highest score into two, with the new centroids being random perturbations of the original centroid, and the new scores being half of the old scores.
  - The new cluster is removed if it has too small a score, or too narrow a variance.

- Gonzales, T. (1985). Clustering to minimize the maximum intercluster distance. *Theor. Comp. Sci. 38*, 293–306.
- Figueiredo, M. A. T. and A. K. Jain (2002). Unsupervised learning of finite mixture models. *IEEE Trans. On Pattern Analysis and Machine Intelligence 24*(3), 381–396. Matlab code at http://www.lx.it.pt/~mtf/mixturecode.zip.
- Arthur, D. and S. Vassilvitskii (2007). kmeans++: the advantages of careful seeding. In *Proc. 18th ACM-SIAM symp. on Discrete algorithms*, pp. 1027-1035.
- Bahmani, B., B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii (2012). Scalable k-Means++. In *VLDB*.

EM For Gaussian Mixtures: Summary

1. Initialize $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$ and evaluate log likelihood.

2. E-Step: Evaluate responsibilities $\gamma(z_{nk})$

3. M-Step: Re-estimate parameters, using current responsibilities

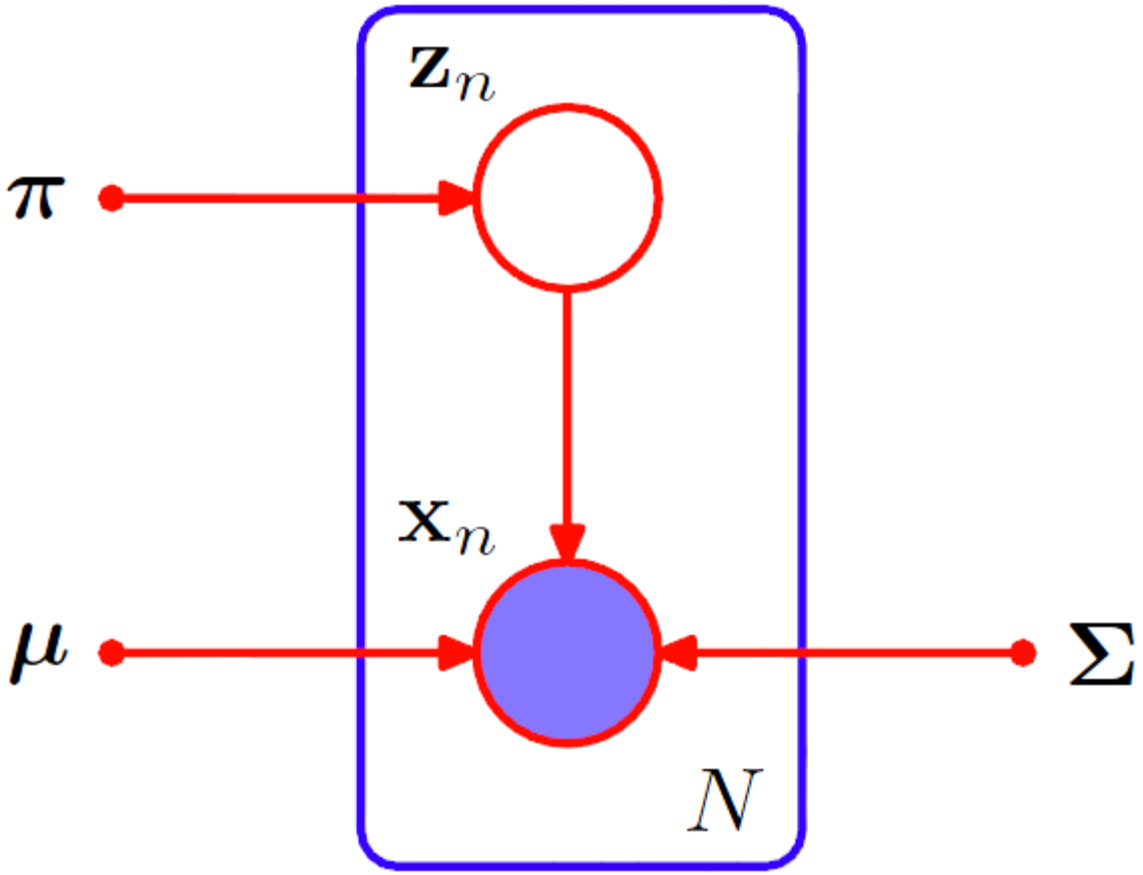
$$\boldsymbol{\mu}_k^{new} = \frac{1}{\sum_n \gamma(z_{nk})} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{\sum_n \gamma(z_{nk})} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T$$

$$\pi_k^{new} = \frac{\sum_n \gamma(z_{nk})}{N}$$

1. Evaluate log-likelihood $\ln p\left(\boldsymbol{X} \middle| \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ and check for convergence (go to Step 2)

Graphical Representation of GMM



- μ and Σ are our θ parameters – not nodes (random variables).
- They leave outside the plate (they are shared by all components)

Latent Variable View of EM

Latent Variable Viewpoint of EM

- Let $\boldsymbol{X}$ be observed data, $\boldsymbol{Z}$ latent variables and $\boldsymbol{\theta}$ parameters.
- We want to maximize the **marginal log-likelihood of observed data**

$$\ln p\left(\boldsymbol{X}|\boldsymbol{\theta}\right)=\ln\left\{\sum_{\boldsymbol{Z}} p\left(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta}\right)\right\}$$

- Optimization problematic due to log-sum in the Eq. above.
- Consider straightforward maximization for complete data.

$$\ln p\left(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta}\right)$$

- Latent **Z** is known only through $p\left(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}\right)$
- We will consider expectation of the complete-data log-likelihood

Latent Variable Viewpoint of EM

- Initialization: Choose initial set of parameters θ^{old}
- E-step: use current parameters θ^{old} to compute $p\left(\mathbf{Z}\middle|\mathbf{X},\theta^{old}\right)$ and then the expected complete-data log-likelihood for general θ

$$Q\left(\theta,\theta^{old}\right)=\sum_{\mathbf{Z}} p\left(\mathbf{Z}\middle|\mathbf{X},\theta^{old}\right)\ln p\left(\mathbf{X},\mathbf{Z}\middle|\theta\right)$$

- M-step: Determine θ^{new} by maximizing $Q\left(\theta,\theta^{old}\right)$

$$\theta^{new}=\arg\max_{\theta} Q\left(\theta,\theta^{old}\right)$$

- Check Convergence: stop, or $\theta^{old}\leftarrow\theta^{new}$ and return to E-Step.
- Note: in the definition of $Q\left(\theta,\theta^{old}\right)$ the log acts directly on $p\left(\mathbf{X},\mathbf{Z}\middle|\theta\right)$ *This step is now computationally tractable*.

Latent Variable Viewpoint of EM

- The algorithm can become Bayesian by maximizing in the M-step: $Q\left(\theta,\theta^{old}\right)+\ln p(\theta)$. This resolves the problems with MLE.

$$\theta^{new}=\arg\max_{\theta}\left\{Q\left(\theta,\theta^{old}\right)+\ln p\left(\theta\right)\right\}$$

- *The algorithm can also be applied* when the unobserved values correspond *to missing data – but only if the mechanism causing values to be missing does not depend on the unobserved values*
  - e.g. a temperature-sensor that stops recording when the temperature greater than some value

Gaussian Mixtures Revisited

- Suppose we knew the values of the latent variables (complete data $\boldsymbol{X}$, $\boldsymbol{Z}$)
  - We are interested to maximize **the complete-data log likelihood ($\boldsymbol{X}$ & $\boldsymbol{Z}$).** From $p(\boldsymbol{x},\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)^{z_k} \Rightarrow$

$$\ln p(\boldsymbol{X},\boldsymbol{Z}|\theta) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \ln\left\{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right\} = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left\{\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right\}$$

- Note *by knowing **z**, the sum is now outside the log!*
- For each n, z_{nk} is only *1* for the right coloring of the data and the algorithm leads to maximizing the individual likelihoods (see formally a proof on the next slide).
  - Trivial closed-form solution: fit each component to the corresponding set of data points
  - But only if we knew the z's.

Maximizing the Complete Data log-Likelihood

$$\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\theta\right)=\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}\ln\left\{\pi_k\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}=\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}\left\{\ln\pi_k+\ln\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}$$

- Consider maximization wrt $\left(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)$. For each data point n, the parameters z_{nk} are 0 except for a particular element which equals 1.
- We can then write: $\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\theta\right)=\sum_{k=1}^{K}\left\{\sum_{n\in X_k}\ln\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}+\ldots$
- We have K independent terms one for each component in the mixture. Maximization wrt $\left(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)$ gives the same results as MLE for the single Gaussian k.
- To maximize wrt π_k: $\max_{\pi_k}\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}\ln\pi_k+\lambda\left(\sum_{k=1}^{K}\pi_k-1\right)\Rightarrow\sum_{n=1}^{N}\frac{z_{nk}}{\pi_k}+\lambda=0$
- Multiplying by π_k and summing in k gives: $\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}+\lambda\sum_{k=1}^{K}\pi_k=0\Rightarrow\lambda=-N$
- Thus: $\sum_{n=1}^{N}\frac{z_{nk}}{\pi_k}=N\Rightarrow\sum_{n=1}^{N}z_{nk}=N\pi_k\Rightarrow$ $\pi_k=\frac{\sum_{n=1}^{N}z_{nk}}{N}$

Gaussian Mixtures Revisited

- Problem: we don't know the values of the latent variables

- Note that for a single data point:

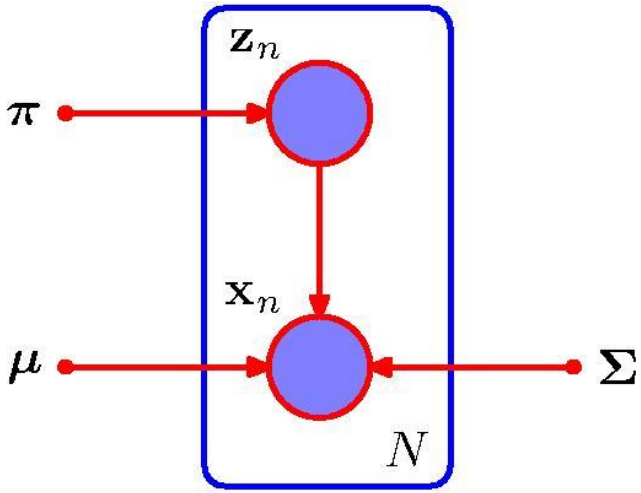
$$\mathbb{E}\left[z_{nk}\right] = p(z_{nk}=1 \mid \boldsymbol{x}_n)\times 1 + p(z_{nk}=0 \mid \boldsymbol{x}_n)\times 0 =$$

$$= \frac{p(\boldsymbol{x}_n \mid z_{nk}=1)\,p(z_{nk}=1)}{\sum_j p(\boldsymbol{x}_n \mid z_{nj}=1)\,p(z_{nj}=1)} = \frac{\mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k}{\sum_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\pi_j} \equiv \gamma\left(z_{nk}\right)$$

- Instead of maximizing the complete data log-likelihood, we **maximize the expected value of the complete data log likelihood**

Gaussian Mixtures Revisited

- We can also derive $\mathbb{E}\left[z_{nk}\right]=\gamma\left(z_{nk}\right)$ as follows:



- Using $p\left(\boldsymbol{x},\boldsymbol{z}\right)=\prod_{k=1}^{K}\pi_k^{z_k}\mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)^{z_k}\Rightarrow$

$$p\left(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}\right)=\prod_{n=1}^{N}p\left(\boldsymbol{z}_n\mid\boldsymbol{x}_n,\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}\right)$$

$$\propto\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\pi_k\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right)^{z_{nk}}$$

- *So the posterior factorizes over n and $\{\boldsymbol{z}_n\}$ are independent* given **X, μ, Σ, π.** This results from d-separation. Considering z_l and z_m, the only paths that connect them are through the parameters **μ**, **Σ**, **π** (observed since we condition on them). All these paths are tail-to-tail and since we condition on the parameters, these paths are blocked.

Gaussian Mixtures Revisited

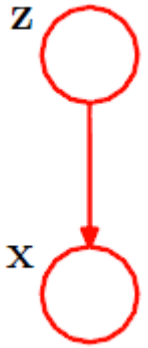
$$p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \propto \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\pi_k \mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right) \right)^{z_{nk}}$$

- The expected value of the indicator variable z_{nk} under this posterior is then:

$$\mathbb{E}\left[z_{nk} \right] = \frac{\sum_{\boldsymbol{z}_n} z_{nk} \prod_{k'} \left(\pi_{k'} \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'}) \right)^{z_{nk'}}}{\sum_{\boldsymbol{z}_n} \prod_{j} \left(\pi_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)^{z_{nj}}} = \frac{\mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \pi_k}{\sum_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \pi_j} \equiv \gamma\left(z_{nk} \right)$$

Gaussian Mixtures Revisited

- We will maximize the log of the joint distribution of latent and observed variables (complete data log likelihood), averaged with respect to the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ of the latent variables $\langle \ln p(\mathbf{X},\mathbf{Z}|\theta) \rangle$ – i.e. replace z_{nk} with the $\gamma(z_{nk})$ (E step of the EM algorithm)



$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] \text{ (responsibilities)}$$

- This will give us the M step of the EM algorithm: Maximize

$$\left\langle \ln p(\mathbf{X},\mathbf{Z}|\theta) \right\rangle = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk}) \left\{ \ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \right\}$$

Gaussian Mixtures Revisited

- Our original problem was to maximize the complete-data log likelihood:

$$\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\theta\right)=\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}\left\{\ln\pi_k+\ln\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}$$

- We change the problem statement by maximizing the log of the joint distribution of latent and observed variables, averaged with respect to the posterior distribution $p(\boldsymbol{Z}|\boldsymbol{X})$ of the latent variables $\langle \ln p(\mathbf{X},\mathbf{Z}|\theta)\rangle$ – i.e. replace z_{nk} with the $\gamma(z_{nk})$

$$\left\langle \ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\theta\right)\right\rangle=\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma\left(z_{nk}\right)\left\{\ln\pi_k+\ln\mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}$$

where

$$\gamma\left(z_{nk}\right)=\mathbb{E}\left[z_{nk}\right]\ \text{(responsibilities)}$$

Gaussian Mixtures Revisited: Summary

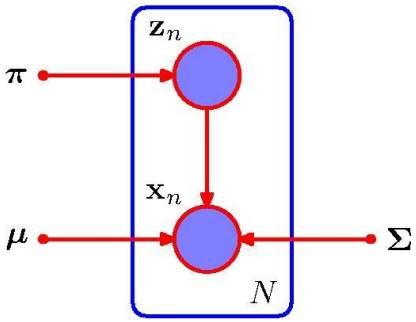
- Assume that for each $\mathbf{x}_n$ we are given the discrete variable (latent assignment variables) z_n

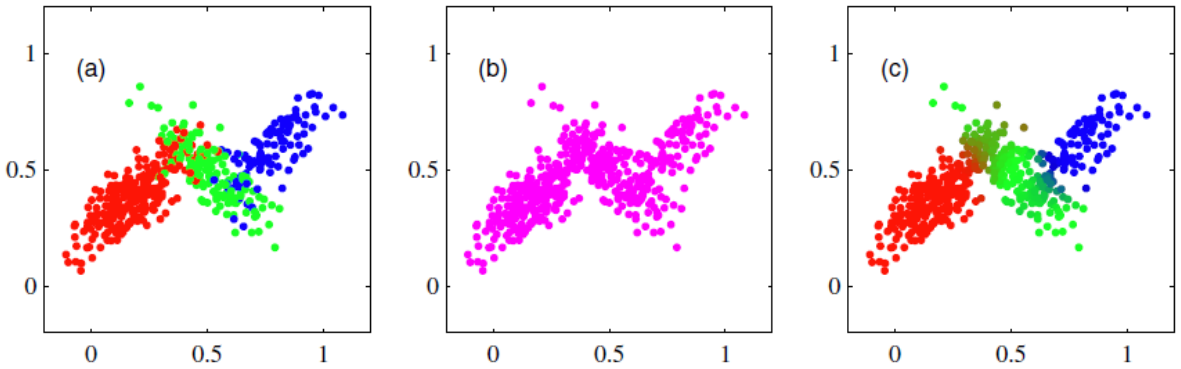
- Complete-data log-likelihood and expectation

$$p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\underbrace{\boldsymbol{\mu},\boldsymbol{\Sigma},\pi}_{\boldsymbol{\theta}}\right)=\prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{z_{nk}}\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)^{z_{nk}}$$

$$\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\boldsymbol{\theta}\right)=\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}\left\{\ln\pi_k+\ln\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}$$

$$Q\left(\boldsymbol{\theta}\right)=\mathbb{E}_{\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\boldsymbol{\theta}\right)\right]=\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\ln\pi_k+\ln\mathcal{N}\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\right\}$$





Gaussian Mixtures Revisited: Summary

- This leads to the EM algorithm for Gaussian mixtures discussed earlier.

- Choose initial values for $\boldsymbol{\mu}^{old}$, $\boldsymbol{\Sigma}^{old}$ and $\boldsymbol{\pi}^{old}$, and use these to evaluate the responsibilities (E step).

- Keep the responsibilities fixed and maximize

$$Q(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{Z}}\left[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})\right] = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\ln \pi_k + \ln \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right\}$$

  with respect to $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and π_k (M step).

- This leads to closed form solutions for $\boldsymbol{\mu}^{new}$, $\boldsymbol{\Sigma}^{new}$ and $\boldsymbol{\pi}^{new}$ identical as before (see proof of one of these next):

$$N_k = \sum_n \gamma(z_{nk}),\ \boldsymbol{\mu}_k^{new} = \frac{\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{x}_n}{N_k},\ \boldsymbol{\Sigma}_k^{new} = \frac{\sum_{n=1}^{N}\gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T}{N_k},\ \pi_k^{new} = \frac{N_k}{N}$$

Gaussian Mixtures Revisited: Summary

$$Q(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{X}, \boldsymbol{Z} \middle| \boldsymbol{\theta}\right)\right] = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\left\{\ln \pi_k + \ln \mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)\right\}$$

- We can write the rhs as follows:

$$-\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) + const$$

- Where the constant are terms independent of $\boldsymbol{\mu}_k$. Taking derivative wrt $\boldsymbol{\mu}_k$:

$$-\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\mu}_k - \boldsymbol{x}_n) = 0 \Rightarrow \boldsymbol{\mu}_k\sum_{n=1}^{N}\gamma(z_{nk}) = \sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{x}_n \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N}\gamma(z_{nk})\boldsymbol{x}_n}{N_k}$$

- Similarly, one can derive expressions for $\boldsymbol{\Sigma}^{new}$ and $\boldsymbol{\pi}^{new}$.

EM Algorithm Vs. K-Means Algorithm

- K-Means does hard (unique) assignment of each point to a class. EM makes soft assignments based on posterior probabilities (responsibilities).
- K-Means is a certain limit of EM for Gaussian mixtures.
- Consider a Gaussian model with $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi\varepsilon)^{D/2}} \exp\left\{-\frac{1}{2\varepsilon}\left\|\boldsymbol{x}-\boldsymbol{\mu}_k\right\|^2\right\}$ for all k (same ε)
- For a fixed ε and K-Gaussian mixture and assuming all $\pi_j \neq 0$, the responsibilities are:

$$\gamma(z_{nk}) = \frac{\exp\left\{-\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\right\|^2 / 2\varepsilon\right\}\pi_k}{\sum_j \exp\left\{-\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\right\|^2 / 2\varepsilon\right\}\pi_j}$$

- Consider: $\varepsilon \to 0$. Note that in this case and regardless of the $\pi_j \neq 0$ $\gamma(z_{nk}) \to r_{nk}$, where $r_{nk} = \begin{cases} 1 & if\ k = \arg\min_j \left\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\right\|^2 \\ 0 & \text{otherwise} \end{cases}$ i.e. each data point is assigned to the closest mean.

EM Algorithm Vs. K-Means Algorithm

- The EM re-estimation equation for the μ_k becomes in this case:

$$\boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma(z_{nk})} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n \rightarrow \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

- The mixing coefficients π_k are equal to the fraction of data points assigned to cluster k.

- The expected complete-data log-likelihood for $\varepsilon \rightarrow 0$ becomes:

$$Q(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{Z}} \left[\ln p\left(\boldsymbol{X}, \boldsymbol{Z} \middle| \underbrace{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \pi}_{\boldsymbol{\theta}} \right) \right] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \left\{ \ln \pi_k + \ln \mathcal{N}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right) \right\}$$

$$\rightarrow \underbrace{-\frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| \boldsymbol{x}_n - \boldsymbol{\mu}_k \right\|^2}_{J} + \text{constant}$$

- Thus maximizing $Q(\boldsymbol{\theta})$ is equivalent to mimimizing J in K-means

Mixture of Bernoulli Distributions

Mixture of Discrete Binary Variables

- Consider *D binary variables x_i, i=1,…,D each governed by a Bernoulli distribution with parameter μ_i* (x_i independent given μ):

$$p\left(\boldsymbol{x}\middle|\boldsymbol{\mu}\right)=\prod_{k=1}^{D}\mu_k^{x_k}\left(1-\mu_k\right)^{1-x_k},$$

$$\boldsymbol{x}=\left\{x_1,\ldots,x_D\right\}^T,\ \boldsymbol{\mu}=\left\{\mu_1,\ldots,\mu_D\right\}^T$$

- The mean and covariance of this distribution are:

$$\mathbb{E}\left[\boldsymbol{x}\right]=\boldsymbol{\mu},\ \mathrm{cov}\left[\boldsymbol{x}\right]=diag\left\{\mu_i\left(1-\mu_i\right)\right\}$$

- Consider a mixture of these Bernoulli distributions

$$p\left(\boldsymbol{x}\middle|\boldsymbol{\mu},\boldsymbol{\pi}\right)=\sum_{k=1}^{K}\pi_k\,p\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k\right),\quad where\quad p\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k\right)=\prod_{i=1}^{D}\mu_{ki}^{x_i}\left(1-\mu_{ki}\right)^{1-x_i},$$

$$\boldsymbol{\pi}=\left\{\pi_1,\ldots,\pi_K\right\}^T,\ \boldsymbol{\mu}=\left\{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K\right\}^T$$

- Lazarsfeld, P. F. and N. W. Henry (1968). *Latent Structure Analysis*. Houghton Mifflin.
- McLachlan, G. J. and D. Peel (2000). *Finite Mixture Models*. Wiley.

Mixture of Bernoulli Distributions

$$p(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \, p(\mathbf{x}|\boldsymbol{\mu}_k), \quad \text{where} \quad p(\mathbf{x}|\boldsymbol{\mu}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} \left(1-\mu_{ki}\right)^{1-x_i},$$

$$\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}^T, \; \boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}^T$$

- For any mixture distribution of the form $p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, p(\mathbf{x}|k)$ with mean and covariance of p(**x**|k) being $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, respectively, the mean and covariance of this mixture distribution are given as:

$$\mathbb{E}[\mathbf{x}] = \sum_{k=1}^{K} \pi_k \boldsymbol{\mu}_k$$

$$Use: \mathbb{E}[\mathbf{x}] = \sum_{k=1}^{K} p(z=k)\mathbb{E}[\mathbf{x} \mid z=k],$$

$$\text{cov}[\mathbf{x}] = \sum_{k=1}^{K} p(z=k)\mathbb{E}\left[\mathbf{x}\mathbf{x}^T \mid z=k\right] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^T$$

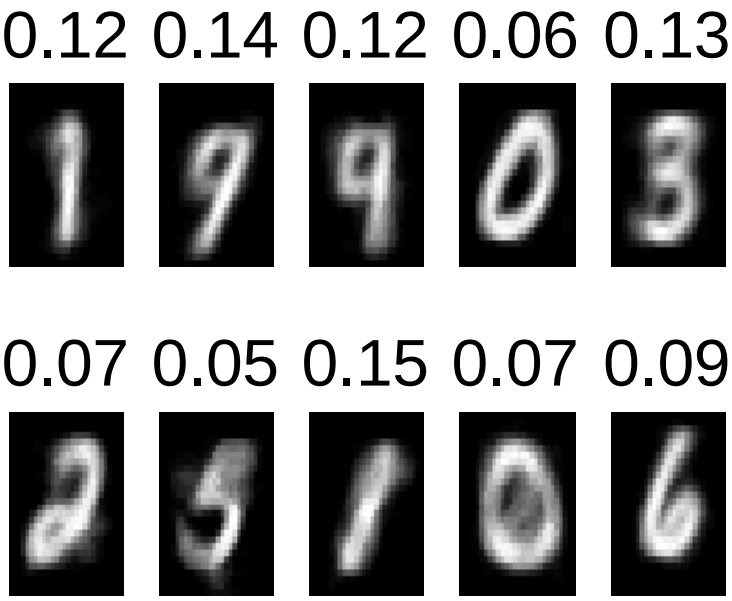
$$\text{cov}[\mathbf{x}] = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^T = \sum_{k=1}^{K} \pi_k \mathbb{E}_k\left[\mathbf{x}\mathbf{x}^T\right] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^T$$

$$= \sum_{k=1}^{K} \pi_k \left\{\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T\right\} - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^T$$

- The joint distribution is not factorized. *The mixture distribution captures correlations* between variables unlike the single product of Bernoullis model.

Example: Mixture of Bernoullis and MLE

- We fit a mixture of Bernoullis to the MNIST handwritten digit data set using $K = 10$ and visualize the centroids (*MLE* of cluster means). The numbers on the top are MLE of the mixing weights.
- This discovers some digit classes, but it creates multiple clusters for some digits and no clusters for others. The reasons for this include:
  - The model is very simple (no notion of shape or a stroke).
  - Some digits exhibit a degree of visual variety (e.g. 7's with and without the cross bar).
  - We need $K \geq 10$ clusters for this data.
  - Using a large K, we create multiple versions of the same digit. One can use model selection to prevent this.
  - The likelihood function is not convex, so we may be stuck in a local optimum.
- One must be cautious trying to interpret any clusters that are discovered by the method.
- Using informative priors can help.



mixBerMnistEM
from Kevin Murphys' PMTK

Mixture of Bernoulli Distributions

- If we are given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ then the log likelihood function for this model is given by

$$\ln p\left(\boldsymbol{X}\middle|\boldsymbol{\mu},\boldsymbol{\pi}\right) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \, p\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k\right) \right\}$$

- Note the summation inside the log. The MLE again does not have a closed solution.

- To derive the EM algorithm, we introduce a latent variable $\boldsymbol{z}$ associated with each instance of $\boldsymbol{x}$. $\mathbf{z} = (z_1, \ldots, z_K)^T$ is a binary K-dimensional variable having a single component equal to 1, with all other components equal to 0:

$$p\left(\boldsymbol{x}_n \middle| \boldsymbol{z}, \boldsymbol{\mu}\right) = \prod_{k=1}^{K} p\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_k\right)^{z_k}$$

- The prior for the latent variables is:

$$p\left(\boldsymbol{z}\middle|\boldsymbol{\pi}\right) = \prod_{k=1}^{K} \pi_k^{z_k}$$

Mixture of Bernoulli Distributions

- The complete-data log-likelihood is:

$$\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\boldsymbol{\mu},\boldsymbol{\pi}\right)=\sum_{n=1}^{N}\sum_{k=1}^{K}z_{nk}\left\{\ln\pi_k+\sum_{i=1}^{D}\left[x_{ni}\ln\mu_{ki}+(1-x_{ni})\ln(1-\mu_{ki})\right]\right\},\ \text{where}:\ \boldsymbol{X}=\{\boldsymbol{x}_n\},\ \boldsymbol{Z}=\{\boldsymbol{z}_n\}$$

- Take the expectation of the complete-data log likelihood with respect to the posterior distribution of the latent variables:

$$\mathbb{E}_{\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\boldsymbol{\mu},\boldsymbol{\pi}\right)\right]=\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\left\{\ln\pi_k+\sum_{i=1}^{D}\left[x_{ni}\ln\mu_{ki}+(1-x_{ni})\ln(1-\mu_{ki})\right]\right\},\ \text{where}:\ \gamma_{nk}=\mathbb{E}\left[z_{nk}\right]$$

$\gamma(z_{nk})$ is the posterior probability, or responsibility, of component k given data point $\boldsymbol{x}_n$.

Mixture of Bernoulli Distributions

- E-Step: Compute the Responsibilities using Bayes' rule

$$\gamma_{nk} = \mathbb{E}\left[z_{nk}\right] = \frac{\sum_{\mathbf{z}_n} z_{nk} \prod_{k'} \left(\pi_{k'} p(\mathbf{x}_n \mid \boldsymbol{\mu}_{k'})\right)^{z_{nk'}}}{\sum_{\mathbf{z}_n} \prod_{j} \left(\pi_j p(\mathbf{x}_n \mid \boldsymbol{\mu}_j)\right)^{z_{nj}}} = \frac{\pi_k p(\mathbf{x}_n \mid \boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \pi_j p(\mathbf{x}_n \mid \boldsymbol{\mu}_j)}$$

- Considering the sum in n,

$$\mathbb{E}_{\mathbf{Z}}\left[\ln p\left(\mathbf{X}, \mathbf{Z} \middle| \boldsymbol{\mu}, \boldsymbol{\pi}\right)\right] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left\{\ln \pi_k + \sum_{i=1}^{D} \left[x_{ni} \ln \mu_{ki} + (1 - x_{ni}) \ln(1 - \mu_{ki})\right]\right\}$$

we note that the responsibilities come through the following terms:

$$N_k = \sum_{n=1}^{N} \gamma_{nk}\,, \quad \bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk}\, \mathbf{x}_n$$

- *N_k is the effective number of data points associated with component k.*

Mixture of Bernoulli Distributions

- M step: Maximize the expected complete-data log likelihood with respect to the parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\pi}$.

- If we set the derivative of

$$\mathbb{E}_{\boldsymbol{Z}}\left[\ln p\left(\boldsymbol{X},\boldsymbol{Z}\middle|\boldsymbol{\mu},\boldsymbol{\pi}\right)\right]=\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\left\{\ln\pi_k+\sum_{i=1}^{D}\left[x_{ni}\ln\mu_{ki}+(1-x_{ni})\ln(1-\mu_{ki})\right]\right\}$$

with respect to $\boldsymbol{\mu}_{ki}$ equal to zero and rearrange the terms, we obtain

$$\frac{\partial}{\partial\mu_{ki}}\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\sum_{i=1}^{D}\left[x_{ni}\ln\mu_{ki}+(1-x_{ni})\ln(1-\mu_{ki})\right]=0\Rightarrow\sum_{n=1}^{N}\gamma_{nk}\left(\frac{x_{ni}}{\mu_{ki}}-\frac{1-x_{ni}}{1-\mu_{ki}}\right)=0\Rightarrow$$

$$\sum_{n=1}^{N}\frac{x_{ni}\gamma_{nk}-\mu_{ki}\gamma_{nk}}{\mu_{ki}(1-\mu_{ki})}=0\Rightarrow\sum_{n=1}^{N}x_{ni}\gamma_{nk}=\mu_{ki}\sum_{n=1}^{N}\gamma_{nk}\Rightarrow\quad\boxed{\mu_k=\bar{\boldsymbol{x}}_k\equiv\frac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}\,\boldsymbol{x}_n}$$

- The mean of component k is equal to a weighted mean of the data. The weighting coefficients are given by the responsibilities that component k takes for data points.

Mixture of Bernoulli Distributions

- For the maximization with respect to π_k, we enforce the constraint $\sum_k \pi_k = 1$.
- As for the mixture of Gaussians, we then obtain

$$\pi_k = \frac{N_k}{N}$$

- The mixing coefficient for component k is given by the effective fraction of points in the data set explained by that component.

Degenerate Case: Initialization

- Note that the following holds for the mixture of Bernoulli distributions:

$$\mathbb{E}[x] = \sum_{k=1}^{K} \pi_k \boldsymbol{\mu}_k = \sum_{k=1}^{K} \underbrace{\pi_k \frac{1}{N_k}}_{1/N} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \underbrace{\sum_{k=1}^{K} \gamma(z_{nk})}_{1} = \bar{\boldsymbol{x}}$$

- If we initialize the means by setting them to a common value $\boldsymbol{\mu}_k = \hat{\boldsymbol{\mu}}, k = 1, .., K$, then:

$$\gamma_{nk} = \frac{\pi_k p(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j)} = \frac{\pi_k}{\sum_{j=1}^{K} \pi_j} = \pi_k \ (independent\ of\ n)$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \boldsymbol{x}_n = \pi_k \frac{1}{N_k} N \bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}$$

i.e. all means converge to the MLE estimate and will never be updated.

- This is a degenerate case that needs to be avoided with proper initialization.

Mixture of Bernoulli Distributions

- In contrast to the mixture of Gaussians, there are no singularities when the likelihood function goes to infinity.

- This can be seen by noting that the likelihood function is bounded above. Indeed:

$$0 \le p(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k) \le 1, 0 \le \pi_k \le 1, \sum_k \pi_k = 1$$

- Then note that the max value of $\ln p(\mathbf{X}|\boldsymbol{\mu},\boldsymbol{\pi})$ is zero.

$$\ln p\left(\boldsymbol{X} \middle| \boldsymbol{\mu}, \boldsymbol{\pi}\right) = \sum_{n=1}^{N} \ln \underbrace{\left\{ \sum_{k=1}^{K} \pi_k \, p\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k\right) \right\}}_{\max \, value \, 1}$$

- The likelihood function can go to zero
  - These singularities would not be found provided that EM is not initialized to a pathological starting point.
  - Recall that the EM algorithm always increases the value of the likelihood function until a local maximum is found.

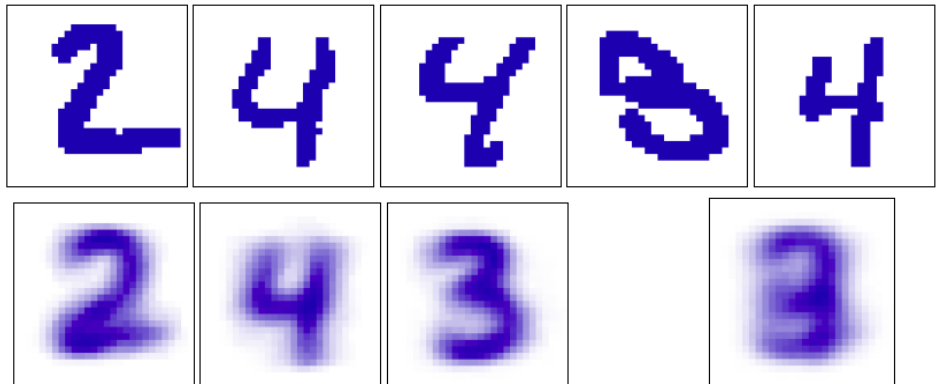
Example: Modeling of Handwritten Digits

- We illustrate the Bernoulli mixture with modeling of handwritten digits.
- We convert the digit images to binary vectors by setting all elements whose values exceed 0.5 to 1 and the remaining elements to 0.
- We fit N = 600 digits, comprising the digits '2', '3', and '4', with a *mixture of K = 3 Bernoulli distributions*. We run 10 iterations of EM.
- The mixing coefficients were initialized to $\pi_k = 1/K$, and the parameters μ_{kj} *were set to random values chosen uniformly in the range (0.25, 0.75) and then normalized* to satisfy the constraint

$$\sum_j \mu_{kj} = 1$$

Example: Modeling of Handwritten Digits

- *A mixture of 3 Bernoulli distributions* is able to find the 3 clusters in the data corresponding to the different digits.



MatLab Code

- On the top: examples from the data after converting the pixel values from grey scale to binary using a threshold of 0.5.

- On the bottom: the first three images show the parameters μ_{ki} for each of the three components in the mixture model.

- On the bottom, last image: we fit the data set using a single multivariate Bernoulli distribution with MLE. This amounts to averaging the counts in each pixel.

Mixture of Bernoulli Distributions: Summary

- Bernoulli distributions over binary data vectors

$$p\left(\boldsymbol{x}\middle|\boldsymbol{\mu}\right)=\prod_{k=1}^{D}\mu_k^{x_k}\left(1-\mu_k\right)^{1-x_k}$$

- Mixture of Bernoullis can model variable correlations

- Bernoulli is member of the exponential family
  - The model is log-linear but the mixture is not. The complete-data log-likelihood however is.

- Simple EM algorithm to find MLE parameters
  - E-Step: Compute responsibilities $\gamma(z_{nk})\propto\pi_k p\left(\boldsymbol{x}_n\middle|\boldsymbol{\mu}_k\right)$
  - M-Step: Update parameters

$$\pi_k=\sum_n\gamma(z_k)/N,\ \boldsymbol{\mu}_k=\sum_{n=1}^{N}\gamma(z_k)\boldsymbol{x}_n/\left(N\pi_k\right)$$

Mixture of Bernoulli Distributions: Extensions

- The conjugate prior for the parameters of a Bernoulli distribution is given by the beta distribution.
  - Recall that a beta prior is equivalent to introducing additional effective observations of **x.**
- We can introduce priors into the Bernoulli mixture model, and use EM to maximize the posterior probability distributions.
- Can extend the analysis of Bernoulli mixtures to multinomial binary variables having *M > 2* states by making use of the discrete distribution

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k}$$

- We can then introduce Dirichlet prior $p(\pi|\alpha)$ and Beta priors $p(\mu_k|a_k, b_k)$.

Mixture of Bernoulli Distributions: MAP

- The E-Step remains the same and in the M-step we need to maximize the following:

$$\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}\right) + \ln p(\boldsymbol{\theta}) =$$

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma_{nk}\left\{\ln \pi_k + \sum_{i=1}^{D}\left[x_{ni}\ln \mu_{ki} + (1-x_{ni})\ln(1-\mu_{ki})\right]\right\}$$

$$+\sum_{j=1}^{K}\sum_{i'=1}^{D}\left(\left(a_j - 1\right)\ln \mu_{ji'} + \left(b_j - 1\right)\ln\left(1-\mu_{ji'}\right)\right) + \sum_{l=1}^{K}\left(\alpha_l - 1\right)\ln \pi_l$$

- Maximizing wrt to $\boldsymbol{\mu}_{ki}$ gives: $\mu_{ki} = \dfrac{N_k\,\bar{\boldsymbol{x}}_{ki} + a_k - 1}{N_k + a_k - 1 + b_k - 1}$, $\bar{\boldsymbol{x}}_{ki} = \dfrac{1}{N_k}\sum_{n=1}^{N}\gamma_{nk}\,\boldsymbol{x}_{ni}$
- Maximization wrt to π_k using a Lagrange multiplier for $\sum_j \pi_j = 1$ gives:

$$\pi_k = \frac{N_k + \alpha_k - 1}{N + \alpha_0 - K}$$

MAP Estimation

- The overfitting of MLE may be severe. This can be addressed by performing MAP estimation. The new auxiliary function is the expected complete data log-likelihood plus the log prior:

$$\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}\right) = \left[\sum_i \sum_k \gamma_{ik} \log \pi_{ik} + \sum_i \sum_k \gamma_{ik} \log p\left(\mathbf{x}_i \mid \boldsymbol{\theta}_k\right)\right] + \log p(\boldsymbol{\pi}) + \sum_k \log p(\boldsymbol{\theta}_k)$$

- The E step is unchanged, but the M step needs to be modified.

- For the prior on the mixture weights, it is natural to use a Dirichlet prior, $\boldsymbol{\pi} \sim \text{Dir}(\boldsymbol{\alpha})$, since this is conjugate to the categorical distribution. The MAP estimate is given by

$$\pi_k = \frac{N_k + \alpha_k - 1}{N + \sum_k \alpha_k - K}$$

- For a uniform prior, $\alpha_k = 1$, this reduces to MLE.

MAP Estimation

- The prior on $\boldsymbol{\theta}_k$, $p(\boldsymbol{\theta}_k)$, depends on the form of the class conditional densities. We discuss the case of GMMs below.

- For simplicity, let us consider a conjugate prior of the form

$$p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \mathcal{NIW}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k | \mathbf{m}_0, \kappa_0, \nu_0, \mathbf{S}_0)$$

MAP Estimation

- Using the results from an earlier lecture on Bayesian inference for Gaussian models, the MAP estimate is given by

$$\hat{\boldsymbol{\mu}}_k = \frac{\kappa_0 \boldsymbol{m}_0 + N_k \bar{\boldsymbol{x}}_k}{N_k + \kappa_0}, \; N_k = \sum_i \gamma_{ik}, \; \bar{\boldsymbol{x}}_k = \frac{\sum_i \gamma_{ik} \boldsymbol{x}_i}{N_k}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\boldsymbol{S}_0 + \boldsymbol{S}_k + \frac{\kappa_0 N_k}{\kappa_0 + N_k} (\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)^T}{\nu_0 + N_k + D + 2}, \; \boldsymbol{S}_k = \sum_{i=1}^{N} \gamma_{ik} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k)^T$$

- We now illustrate the benefits of using MAP estimation instead of ML estimation in the context of GMMs. We apply EM to some synthetic data in *D* dimensions, using either ML or MAP estimation.

- We count the trial as a failure if there are numerical issues involving singular matrices. For each dimensionality, we conduct 5 random trials. The results are illustrated next using *N* = 100.

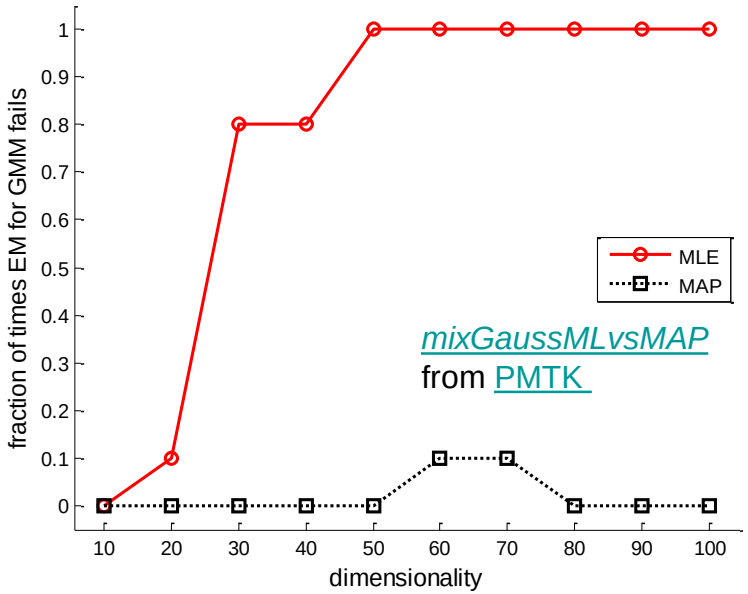
MAP Estimation

- For *D* large, ML estimation crashes, whereas MAP works.
- When using MAP estimation, we need to specify the hyper-parameters. We can set $\kappa_0 = 0$, so that the $\boldsymbol{\mu}_k$ are unregularized, since the numerical problems only arise from $\boldsymbol{\Sigma}_k$. In this case, the MAP estimates simplify to

$$\hat{\boldsymbol{\mu}}_k = \bar{\boldsymbol{x}}_k, \hat{\boldsymbol{\Sigma}}_k = \frac{\boldsymbol{S}_0 + \boldsymbol{S}_k}{\nu_0 + N_k + D + 2}$$

- Using the pooled variance s_j for each dimension j, we set:

$$\boldsymbol{S}_0 = \frac{1}{K^{1/D}} diag\left(s_1^2, ..., s_D^2\right), s_j = \frac{1}{N}\sum_{i=1}^{N}\left(x_{ij} - \bar{x}_j\right)^2$$



- Fraley, C. and A. Raftery (2002). Model-based clustering, discriminant analysis, and density estimation. *J. of the Am. Stat. Assoc.* (97), 611–631 (see pp. 163)

- With the $1/K^{1/D}$ term, the volume of each ellipsoid is then given by

$$|\boldsymbol{S}_0| = \frac{1}{K}\left|diag\left(s_1^2, ..., s_D^2\right)\right|$$

- The parameter v_0 controls how strongly we believe this prior. The weakest proper prior we can use, is to set $v_0 = D + 2$.

Other Applications of EM

EM for Bayesian Linear Regression

- Recall Bayesian Linear Regression:

$$p\left(\boldsymbol{t}\middle|\boldsymbol{w},\beta,\boldsymbol{X}\right)=\prod_{n=1}^{N}\mathcal{N}\left(t_n;\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n),\beta^{-1}\right) \quad \text{Likelihood}$$

$$p\left(\boldsymbol{w}\middle|\alpha\right)=\mathcal{N}\left(\boldsymbol{w};\alpha^{-1}\boldsymbol{I}\right) \quad \text{Prior}$$

$$p\left(\boldsymbol{t}\middle|\alpha,\beta,\boldsymbol{X}\right)=\int p\left(\boldsymbol{t}\middle|\boldsymbol{w},\beta\right)p\left(\boldsymbol{w}\,|\,\alpha\right)d\boldsymbol{w} \quad \text{Marginal Likelihood}$$

- Our goal is to maximize the evidence function $p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{\beta})$ with respect to α and β.
- Because ***w*** is marginalized out, we can regard it as a latent variable, and hence *we can optimize this marginal likelihood function using EM.*
- E step: compute the posterior distribution of **w** given the current setting of the parameters α and β and then use this to find the expected complete-data log likelihood.
- M step: maximize this quantity with respect to α *and* β.

EM for Bayesian Linear Regression

- We have already derived the posterior distribution of **w** given by

$$p(\boldsymbol{w}|\boldsymbol{t}) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{m}_N, \boldsymbol{S}_N),$$

$$\boldsymbol{m}_N = \boldsymbol{S}_N\left(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \beta\boldsymbol{\Phi}^T\boldsymbol{t}\right) \quad \boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi}, \quad \boldsymbol{S}_0^{-1} = \alpha^{-1}\boldsymbol{I}$$

- The complete-data log likelihood function is then given by

$$p(\boldsymbol{t}|\boldsymbol{w}, \beta, \boldsymbol{X}) = \prod_{n=1}^{N} \mathcal{N}\left(t_n; \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n), \beta^{-1}\right)$$

ln $p(\boldsymbol{t},\boldsymbol{w}|\alpha, \beta) = ln p(\boldsymbol{t}|\boldsymbol{w}, \beta) + ln p(\boldsymbol{w}|\alpha)$ where:

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}\left(\boldsymbol{w}; \alpha^{-1}\boldsymbol{I}\right)$$

- Taking the expectation wrt the posterior of ***w*** gives:

$$\mathbb{E}\left[\ln p(\boldsymbol{t},\boldsymbol{w}|\alpha,\beta)\right] = \frac{M}{2}\ln\left(\frac{\alpha}{2\pi}\right) - \frac{\alpha}{2}\mathbb{E}\left[\boldsymbol{w}^T\boldsymbol{w}\right] + \frac{N}{2}\ln\left(\frac{\beta}{2\pi}\right) - \frac{\beta}{2}\sum_{n=1}^{N}\mathbb{E}\left[\left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}_n\right)^2\right]$$

- M Step: Setting the derivatives wrt to α and β zero and using $\mathbb{E}\left[\boldsymbol{w}^T\boldsymbol{w}\right] = \boldsymbol{m}_N^T\boldsymbol{m}_N + Tr\left[\boldsymbol{S}_N\right]$ and $\mathbb{E}\left[\left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}_n\right)^2\right] = \left(t_n - \boldsymbol{m}_N^T\boldsymbol{\phi}_n\right)^2 + Tr\left[\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T\boldsymbol{S}_N\right]$ we obtain:

$$\alpha^{-1} = \frac{1}{M}\left(\boldsymbol{m}_N^T\boldsymbol{m}_N + Tr\left[\boldsymbol{S}_N\right]\right), \quad \beta^{-1} = \frac{1}{N}\left(\left\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{m}_N\right\| + Tr\left[\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{S}_N\right]\right)$$

EM for Bayesian Linear Regression

- The re-estimation eqs

$$\alpha^{-1} = \frac{1}{M}\left(\boldsymbol{m}_N^T \boldsymbol{m}_N + Tr\left(\boldsymbol{S}_N\right)\right), \quad \beta^{-1} = \frac{1}{N}\left(\left\|\boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{m}_N\right\| + Tr\left[\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{S}_N\right]\right)$$

  seem slightly different from the corresponding result

$$\alpha = \frac{\gamma}{\boldsymbol{m}_N^T \boldsymbol{m}_N}, \quad \gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i}$$

  derived by direct evaluation of the evidence function.

- Each involve inversion (or eigen decomposition) of an $M \times M$ matrix and hence have comparable computational cost per iteration.

EM for Bayesian Linear Regression

- The two approaches of determining α converge to the same result (assuming they find the same local maximum of the evidence function). This can be verified by noting that the quantity γ is defined by

$$\gamma = M - \alpha \sum_i \frac{1}{\alpha + \lambda_i} = M - \alpha Tr\left[\boldsymbol{S}_N\right]$$

$$\boldsymbol{S}_N^{-1} = \boldsymbol{S}_0^{-1} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}, \quad \boldsymbol{S}_0^{-1} = \alpha^{-1}\boldsymbol{I}$$

$$\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} \boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

- At a stationary point of the evidence function, the re-estimation equation

$$\alpha = \frac{\gamma}{\boldsymbol{m}_N^T \boldsymbol{m}_N}$$

will be self-consistently satisfied and hence we can substitute for γ to give:

$$\alpha \boldsymbol{m}_N^T \boldsymbol{m}_N = \gamma = M - \alpha Tr\left[\boldsymbol{S}_N\right]$$

- Solving for α we obtain

$$\alpha^{-1} = \frac{1}{M}\left(\boldsymbol{m}_N^T \boldsymbol{m}_N + Tr\left[\boldsymbol{S}_N\right]\right)$$

EM for Relevance Vector Machine for Regression

- Consider also the relevance vector machine for regression
- We used earlier direct maximization of the marginal likelihood to derive re-estimation equations for the hyperparameters *α and β*
- Here, we consider an alternative approach in which we *view the weight vector* ***w*** *as a latent variable and apply the EM algorithm*.
- E step: find the posterior distribution over the weights, and this is given by

$$p(\boldsymbol{w} \mid \boldsymbol{t}, \boldsymbol{X}, \alpha, \beta) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}, \boldsymbol{\Sigma}), \boldsymbol{m} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{t}, \boldsymbol{\Sigma} = \left(\boldsymbol{A} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} \right)^{-1}$$

- M step: maximize the expected complete-data log likelihood, which is defined by

$$\mathbb{E}_w \left[\ln \left\{ p\left(\boldsymbol{t} \mid \boldsymbol{X}, \boldsymbol{w}, \beta\right) p(\boldsymbol{w} \mid a) \right\} \right]$$

- The expectation is taken wrt posterior distribution computed using the old parameter values.

EM for Relevance Vector Machine for Regression

- To compute the new parameter values, we maximize wrt α and β:

$$\alpha_i^{new} = \frac{1}{m_i^2 + \Sigma_{ii}}, \quad \left(\beta^{new}\right)^{-1} = \frac{\left\|\boldsymbol{t} - \boldsymbol{\Phi m}\right\|^2 + \beta^{-1}\sum_i \gamma_i}{N}$$

- These re-estimation eqs are equivalent to those obtained by direct maximization.

- Proof: To show the first Eq. note the following:

$$\mathbb{E}_w\left[\ln\left\{p\left(\boldsymbol{t}\,|\,\boldsymbol{X},\boldsymbol{w},\beta\right)p(\boldsymbol{w}\,|\,a)\right\}\right] = \mathbb{E}_w\left[\ln\mathcal{N}\left(\boldsymbol{t}\,|\,\boldsymbol{\Phi w},\beta^{-1}\boldsymbol{I}\right) + \ln\mathcal{N}\left(\boldsymbol{w}\,|\,\boldsymbol{0},\boldsymbol{A}^{-1}\right)\right]$$

$$= \frac{1}{2}\mathbb{E}_w\left[N\ln\beta - \beta\left\|\boldsymbol{t} - \boldsymbol{\Phi w}\right\|^2 + \sum_{i=1}^{M}\ln\alpha_i - Tr\left[\boldsymbol{A}\boldsymbol{w}\boldsymbol{w}^T\right]\right] + const$$

$$= \frac{1}{2}\left(N\ln\beta - \beta\left(\left\|\boldsymbol{t} - \boldsymbol{\Phi m}\right\|^2 + Tr\left[\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\right]\right) + \sum_{i=1}^{M}\ln\alpha_i - Tr\left[\boldsymbol{A}\left(\boldsymbol{m}\boldsymbol{m}^T + \boldsymbol{\Sigma}\right)\right]\right) + const$$

- Differentiation wrt to α_i gives the Eq. on the top left.

EM for Relevance Vector Machine for Regression

- To compute the new parameter values, we maximize wrt a and b:

$$\alpha_i^{new} = \frac{1}{m_i^2 + \Sigma_{ii}}, \quad \left(\beta^{new}\right)^{-1} = \frac{\left\|\boldsymbol{t} - \boldsymbol{\Phi m}\right\|^2 + \beta^{-1}\sum_i \gamma_i}{N}$$

- Proof: Differentiation wrt β gives:

$$\mathbb{E}_{\boldsymbol{w}}\left[\ln\left\{p\left(\boldsymbol{t}\,|\,\boldsymbol{X},\boldsymbol{w},\beta\right)p(\boldsymbol{w}\,|\,a)\right\}\right]$$

$$= \frac{1}{2}\left(N\ln\beta - \beta\left(\left\|\boldsymbol{t} - \boldsymbol{\Phi m}\right\|^2 + Tr\left[\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\right]\right) + \sum_{i=1}^{M}\ln\alpha_i - Tr\left[\boldsymbol{A}\left(\boldsymbol{m}\boldsymbol{m}^T + \boldsymbol{\Sigma}\right)\right]\right) + const \Rightarrow$$

$$\frac{N}{2}\frac{1}{\beta} - \frac{1}{2}\left(\left\|\boldsymbol{t} - \boldsymbol{\Phi m}\right\|^2 + Tr\left[\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\right]\right) = 0$$

- The required result follows by noting:

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Sigma} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}\left(\boldsymbol{A} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1} = \beta^{-1}\left(\boldsymbol{I} - \boldsymbol{A}\left(\boldsymbol{A} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\right) = \beta^{-1}\left(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{\Sigma}\right) \Rightarrow$$

$$tr\left[\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Sigma}\right] = \beta^{-1}tr\left(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{\Sigma}\right) = \beta^{-1}\sum_i \gamma_i, \; \gamma_i = 1 - \alpha_i\Sigma_{ii}$$

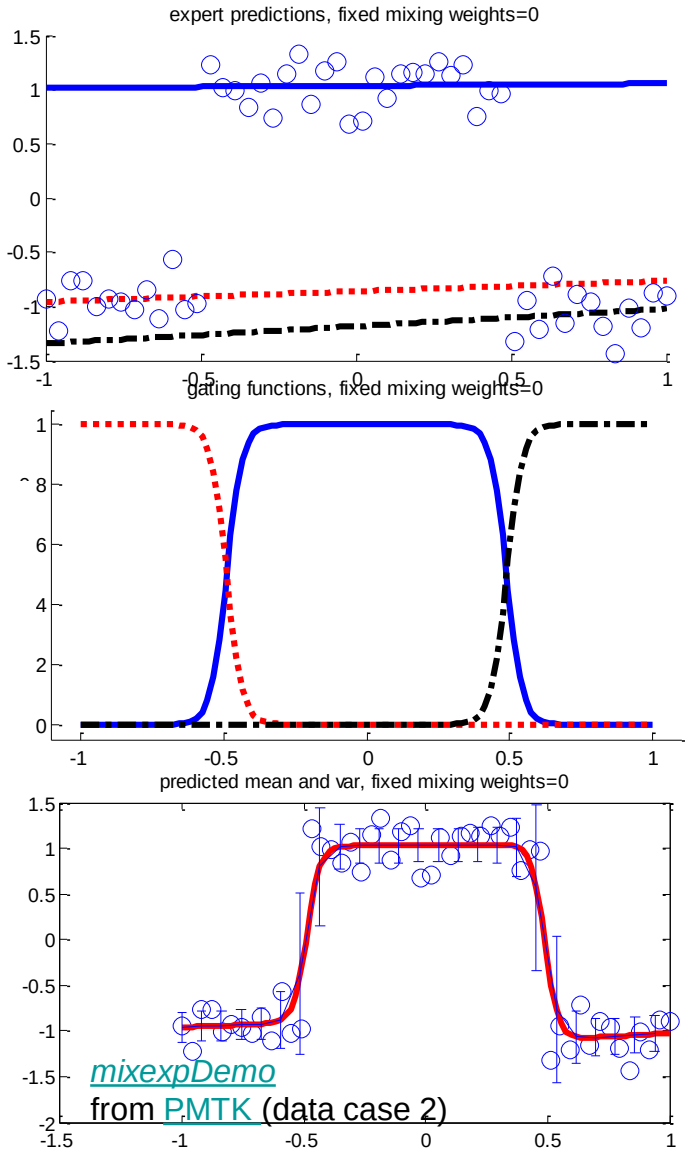
Mixture of Experts

- We can also use mixture models to also create discriminative models for classification and regression.
- Consider the data shown.
- A good model would be three linear regression functions, each applying to a different part of the input space.
- We can model this by allowing the mixing weights and the mixture densities to be input-dependent:

$$p(y_i \mid \mathbf{x}_i, z_i = k, \theta) = \mathcal{N}(y_i \mid \mathbf{w}_k^T \mathbf{x}_i, \sigma_k^2)$$

$$p(z_i \mid \mathbf{x}_i, \theta) = \mathcal{C}at(z_i \mid \mathcal{S}(\mathbf{V}^T \mathbf{x}_i))$$

- This model is called *a **mixture of experts***.
- Each submodel is an "expert" in a certain region of input space.

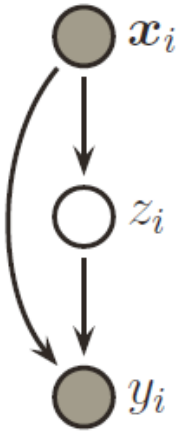


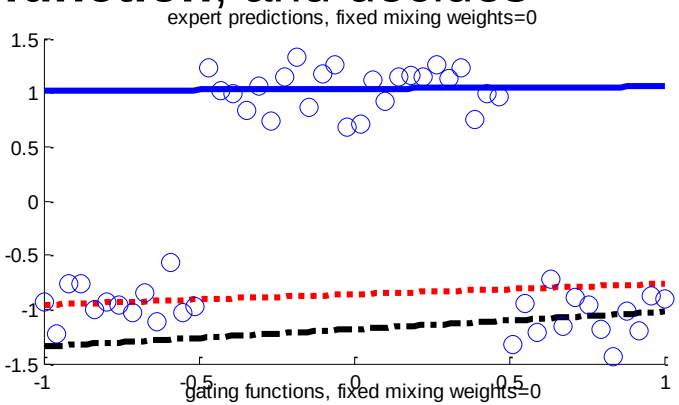
- Jordan, M. I. and R. A. Jacobs (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation 6*, 181–214.

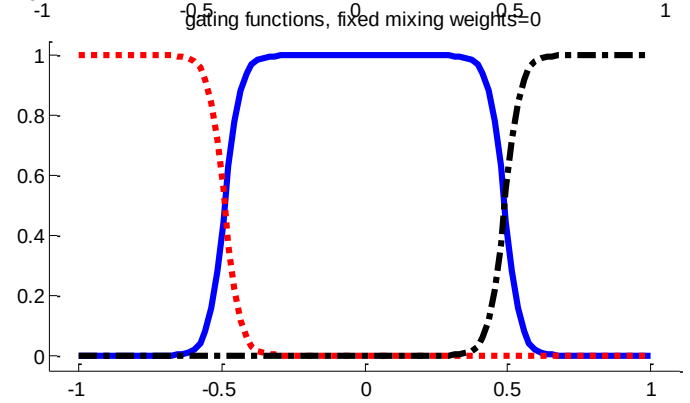
Mixture of Experts

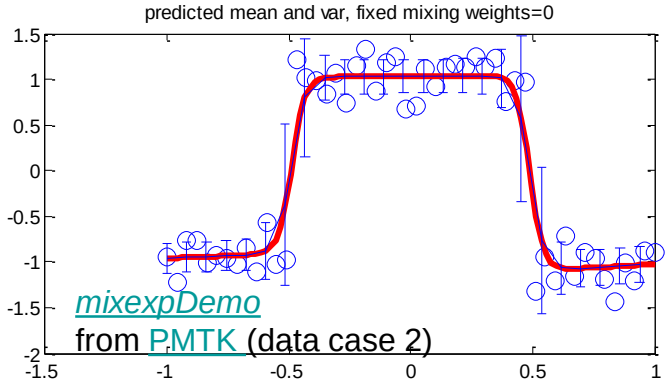
- The function $p(z_i = k|\mathbf{x}_i, \boldsymbol{\theta})$ is called a **gating function**, and decides which expert to use, depending on the input values.
- For example,
  - (b) shows how the three experts have carved up the 1d input space,
  - (a) shows the predictions of each expert individually (each expert a linear regression model), and
  - (c) shows the overall prediction of the model, obtained using

$$p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) = \sum_k p(z_i = k \mid \boldsymbol{x}_i, \boldsymbol{\theta}) p(y_i \mid \boldsymbol{x}_i, z_i = k, \boldsymbol{\theta})$$







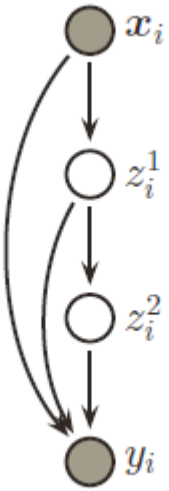


mixexpDemo from PMTK (data case 2)

- Jordan, M. I. and R. A. Jacobs (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation 6*, 181–214.

Hierarchical Mixture of Experts

- We can "plug in" any model for the expert. For example, we can use neural networks to represent both the gating functions and the experts.
- The result is known as a **mixture density network**. Such models are slower to train, but can be more flexible than mixtures of experts.
- It is also possible to make each expert be itself a mixture of experts. This gives rise to a model known as the **hierarchical mixture of experts**. See Figure for the DGM.



- Bishop, C. M. (1994). Mixture density networks. Technical Report NCRG 4288, Neural Computing Research Group, Department of Computer Science, Aston University.

EM For Mixture of Experts

- We can easily fit a mixture of experts model using EM. The expected complete data log likelihood is given by

$$Q\ \boldsymbol{\theta}, \boldsymbol{\theta}^{old}\ == \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log\left[\pi_{ik} \mathcal{N}\ y_i \mid \boldsymbol{w}_k^T \boldsymbol{x}_i, \sigma_k^2\ \right]$$

$$\pi_{ik} = \mathcal{S}\ \boldsymbol{V}^T \boldsymbol{x}_i\ _k, \gamma_{ik} \propto \pi_{ik}^{old}\ \mathcal{N}\ y_i \mid \boldsymbol{x}_i^T \boldsymbol{w}_k^{old}, \sigma_k^{old}\ ^2$$

- The E step is the same as in a standard mixture model, except we have to replace π_k with π_{ik} when computing γ_{ik}.
- In the M step, we need to maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ wrt $\mathbf{w}_k$, σ^2_k and $\mathbf{V}$. For the regression parameters for model k, the objective has the form

$$Q\ \boldsymbol{\theta}, \boldsymbol{\theta}^{old}\ = \sum_{i=1}^{N} \gamma_{ik} \left(-\frac{1}{\sigma_k^2} (y_i - \boldsymbol{w}_k^T \boldsymbol{x}_i) \right)$$

- We recognize this as a weighted least squares problem, which makes intuitive sense: if γ_{ik} is small, then data point i will be downweighted when estimating model k's parameters.
- We can immediately write down the MLE as

$$\boldsymbol{w}_k = (\boldsymbol{X}^T \boldsymbol{R}_k \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{R}_k \boldsymbol{y}, \boldsymbol{R}_k = diag(\gamma :, k)$$

EM For Mixture of Experts

- The MLE for the variance is:

$$\sigma_k^2 = \frac{\sum_{i=1}^{N} \gamma_{ik} \left(y_i - \boldsymbol{w}_k^T \boldsymbol{x}_i \right)^2}{\sum_{i=1}^{N} \gamma_{ik}}$$

- We replace the estimate of the unconditional mixing weights $\boldsymbol{\pi}$ with the estimate of the gating parameters, **V**. The objective has the form

$$\ell(V) = \sum_i \sum_k \gamma_{ik} \log \pi_{i,k}$$

- This is equivalent to the log-likelihood for multinomial logistic regression, except we have replaced the "hard" 1-of-*C* encoding $\mathbf{y}_i$ with the "soft" 1-of-*K* encoding γ_i.

- Thus *we can estimate **V** by fitting a logistic regression model to soft target labels*.

Application to Inverse Problems

- Mixtures of experts are useful in solving **inverse problems**. These are problems where we have to invert a many-to-one mapping. A typical example is in robotics, where the location of the end effector (hand) **y** is uniquely determined by the joint angles of the motors, **x**.

- However, for any given location **y**, there are many settings of the joints **x** that can produce it. Thus the inverse mapping $\mathbf{x} = f^{-1}(\mathbf{y})$ is not unique.

- Another example is **kinematic tracking** of people from video (Bo et al. 2008), where the mapping from image appearance to pose is not unique, due to self occlusion, etc.

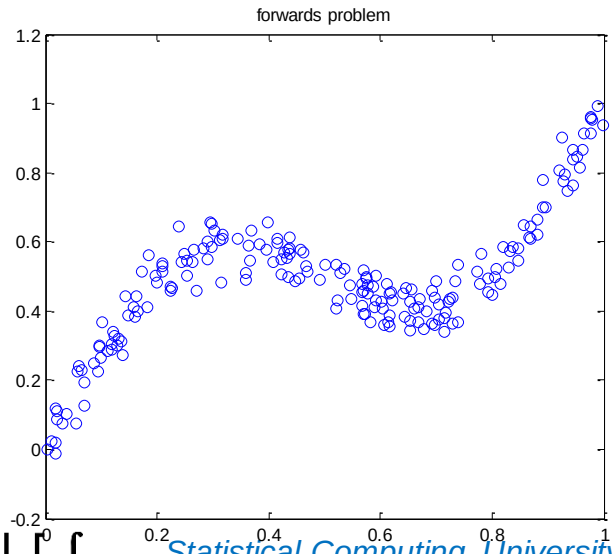
- Bo, L., C. Sminchisescu, A. Kanaujia, and D. Metaxas (2008). Fast Algorithms for Large Scale Conditional 3D Prediction. In *CVPR*.

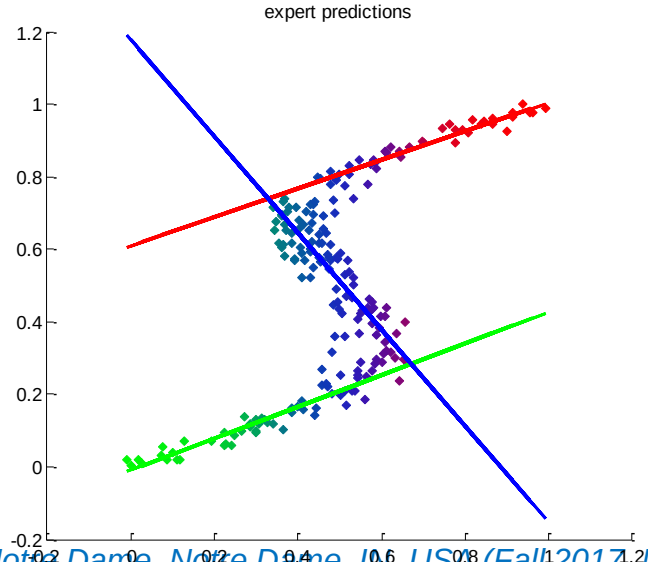
Application to Inverse Problems

- A simpler example is shown here. It defines a function, $y = f(x)$, (**forward model**). Now consider the problem of computing $x = f^{-1}(y)$. The corresponding inverse model is shown in Figure (b); this is obtained by simply interchanging the x and y axes.

- Now we see that for some values along the horizontal axis, there are multiple possible outputs, so the inverse is not uniquely defined. For example, if $y = 0.8$, then x could be 0.2 or 0.8.

- Consequently, the predictive distribution, $p(x|y, \boldsymbol{\theta})$ is multimodal.

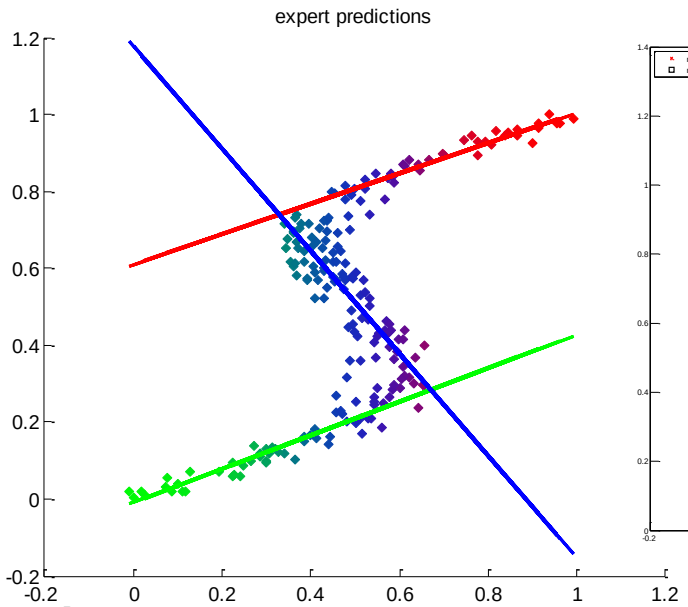


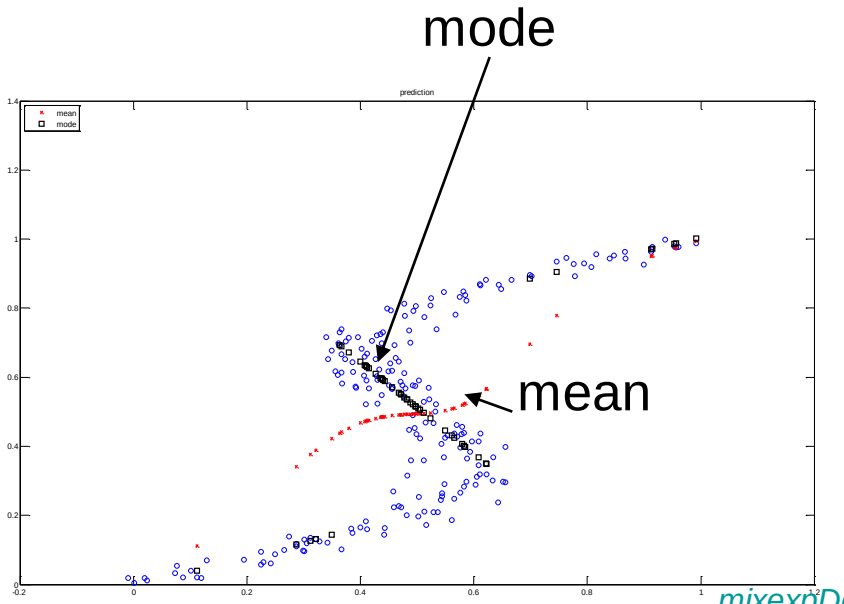


mixexpDemoOneToMany
from Kevin Murphys' PMTK

Application to Inverse Problems

- We can fit a mixture of linear experts to this data. Figure (b) shows the prediction of each expert, and (c) shows (a plugin approximation to) the posterior predictive mode and mean.
- The posterior mean does not yield good predictions. In fact, any model which is trained to minimize mean squared error — even if the model is a flexible nonlinear model, such as neural network — will work poorly on inverse problems such as this. However, the posterior mode, where the mode is input dependent, provides a reasonable approximation.

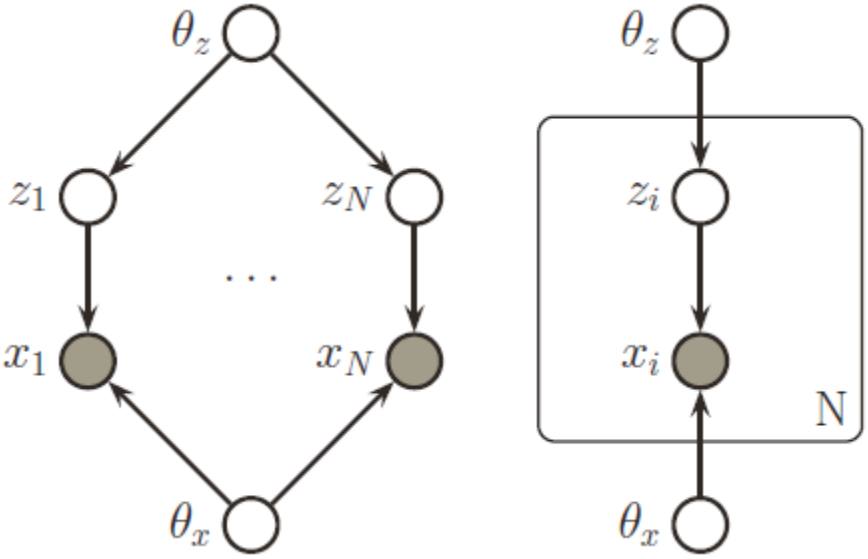




mixexpDemoOneToMany from Kevin Murphys' PMTK

Parameter Estimation for Mixture Models

- We know how to compute the posterior over the hidden variables given the observed variables when the parameters are known.
- We are interested to learn the parameters. When we have complete data and a factored prior, the posterior over the parameters also factorizes, making computation very simple.
- This is no longer true if we have hidden variables and/or missing data.



- Indeed, from the Fig., if the z_i 's were observed, then by d-separation, $\boldsymbol{\theta}_z \perp \boldsymbol{\theta}_x | \mathcal{D}$, and hence the posterior will factorize.
- But since in an LVM the z_i are hidden, the parameters are no longer independent and the posterior does not factorize, making it much harder to compute.
- This complicates the computation of the MLE and MAP.

Unidentifiability

- $p(\boldsymbol{\theta}|D)$ for an LVM may have multiple modes. Consider a Gaussian Mixture Model. If the z_i 's were all observed, we would have a unimodal posterior for the parameters:

$$p(\theta \mid \mathcal{D}) = Dir(\pi \mid \mathcal{D}) \prod_{k=1}^{K} \mathcal{NIW}(\mu_k, \Sigma_k \mid \mathcal{D})$$

- Consequently we can easily find the globally optimal MAP estimate (and hence globally optimal MLE).

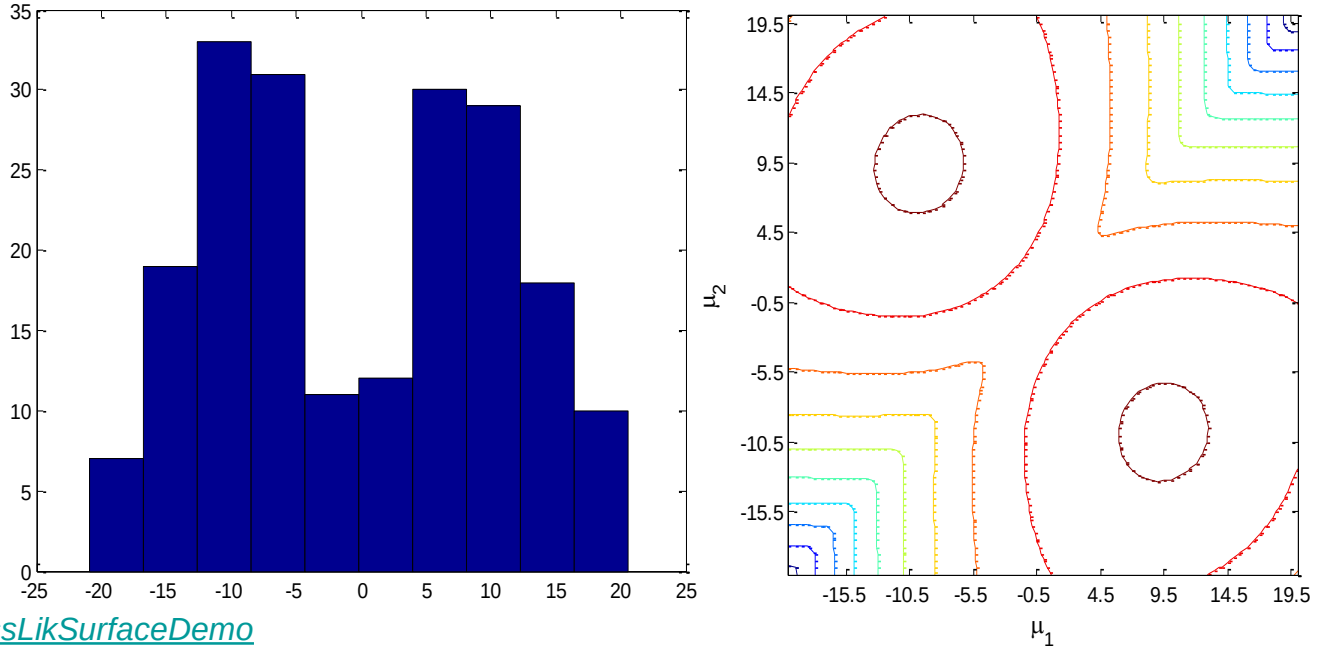
- But now suppose the z_i's are hidden. In this case, for each of the possible ways of filling in the z_i's, we get a different unimodal likelihood. Thus when we marginalize out over the z_i's, we get a multi-modal posterior for $p(\boldsymbol{\theta}|D)$.

- These modes correspond to different labelings of the clusters.

- Carreira-Perpinan, M. and C. Williams (2003). An isotropic gaussian mixture can have more modes than components. Technical Report EDI-INF-RR-0185, School of Informatics, U. Edinburgh.

Unidentifiability

- This is illustrated in (b) where the likelihood function, $p(\mathcal{D}|\mu_1, \mu_2)$, is shown for a 2D GMM with $K = 2$. The data are shown in (a).
- We see two peaks, one corresponding to $\mu_1 = -10$, $\mu_2 = 10$, and the other to $\mu_1 = 10$, $\mu_2 = -10$. *The parameters are not **identifiable**, since there is not a unique MLE.*
- Therefore there cannot be a unique MAP estimate (assuming the prior does not rule out certain labelings), and hence the posterior must be multimodal.



mixGaussLikSurfaceDemo from PMTK

Unidentifiability

- There are *K! possible labelings*, but some of the peaks might get merged. Nevertheless, there can be an exponential number, since finding the optimal MLE for a GMM is NP-hard.
- Unidentifiability can cause a problem for Bayesian inference: Suppose we draw samples from the posterior, $\boldsymbol{\theta}^{(s)} \sim p(\boldsymbol{\theta}|D)$, and then average them, to try to approximate the posterior mean

$$\overline{\boldsymbol{\theta}} = \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\theta}^{(s)}$$

- If the samples come from different modes, the average will be meaningless. Note, however, that it is reasonable to average the posterior predictive distributions,

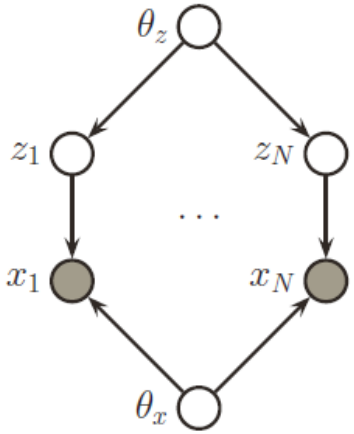
$$p(x) \approx \frac{1}{S}\sum_{s=1}^{S} p\ \ x \mid \boldsymbol{\theta}^{(s)}$$

since the likelihood function is invariant to which mode the parameters came from.

- Aloise, D., A. Deshpande, P. Hansen, and P. Popat (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning 75*, 245– 249.
- Drineas, P., A. Frieze, R. Kannan, S. Vempala, and V. Vinay (2004). Clustering large graphs via the singular value decomposition. *Machine Learning 56*, 9–33.

Unidentifiability

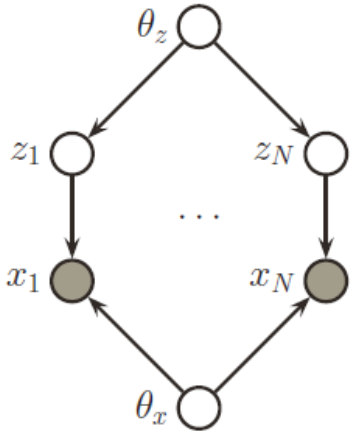
- A variety of solutions have been proposed to the unidentifiability problem depending on the model and the inference algorithm used.
- Stephens presents an approach to handling unidentifiability in mixture models using MCMC.
- Here we just compute a single local mode (approximate MAP estimation) by $\widehat{\boldsymbol{\theta}}$ rather than the posterior $p(\theta|\mathcal{D})$. Using this we evaluate $p\left(z_i|\boldsymbol{x}_i, \widehat{\boldsymbol{\theta}}\right)$
- This is common simple and reasonable approximation if the sample size is sufficiently large.
- Here we have *N* latent variables, each of which gets to "see" one data point only. However, there are only two latent parameters, each of which gets to see *N* data points. *So the posterior uncertainty about the parameters is typically much less than the posterior uncertainty about the latent variables*.



- Stephens, M. (2000). Dealing with label-switching in mixture models. *J. Royal Statistical Society, Series B 62*, 795–809.

Unidentifiability

- This justifies the common strategy of computing $p(z_i \mid x_i, \hat{\theta})$ but not bothering to compute $p(\boldsymbol{\theta}|D)$.
- We already looked at hierarchical Bayesian models, which essentially put structure on top of the parameters.
- In such models, it is important to model $p(\boldsymbol{\theta}|D)$, so that the parameters can send information between themselves. If we used a point estimate, this would not be possible.



Computing a MAP Estimate is Non Convex

- We argued that the likelihood function has multiple modes, and hence that finding an MAP or ML estimate will be hard.
- We now show this result by more algebraic means, which sheds some additional insight into the problem .
- Consider the log-likelihood for an LVM:

$$logp(\mathcal{D}|\theta) = \sum_i \log \sum_{z_i} p(x_i, z_i|\theta)$$

- This objective is hard to maximize since we cannot push the log inside the sum.
- Suppose $p(\mathbf{z}_i, \mathbf{x}_i|\boldsymbol{\theta})$ is in the exponential family:

$$p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} exp[\boldsymbol{\theta}^T \phi(\mathbf{x}, \mathbf{z})]$$

where $\phi(\mathbf{x}, \mathbf{z})$ are the sufficient statistics, and $Z(\boldsymbol{\theta})$ is the normalization constant.

- Rennie, J. (2004). Why sums are bad. Technical report, MIT.

Computing a MAP Estimate is Non Convex

- The MVN is in the exponential family, as are the Dirichlet, multinomial, Gamma, Wishart, etc. (Student's $\mathcal{T}$ distribution is an exception.) *Mixtures of exponential families are also in the exponential family, providing the mixing variables **are observed**.*
- The **complete data log likelihood** can be written as follows:

$$\ell(\boldsymbol{\theta}) = \sum_i log p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\theta}) = \boldsymbol{\theta}^T \left(\sum_i \phi(\boldsymbol{x}_i, \boldsymbol{z}_i) \right) - NZ(\boldsymbol{\theta})$$

- The 1st term is linear in $\boldsymbol{\theta}$. $Z(\boldsymbol{\theta})$ is a convex function* so the overall objective is concave (due to the - sign), and hence has a unique maximum.
- Now consider what happens when we have missing data. The **observed data log likelihood** is given by

$$\ell(\boldsymbol{\theta}) = \sum_i log \sum_{z_i} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\theta}) = \sum_i \log \left(\sum_{z_i} e^{\theta^T \phi(\boldsymbol{x}_i, \boldsymbol{z}_i)} \right) - N \log Z(\boldsymbol{\theta})$$

- Can show that the log-sum-exp function is convex, and we know that $Z(\boldsymbol{\theta})$ is convex. The difference of two convex functions is not, in general, convex. So the objective is neither convex nor concave, and has local optima.

*Boyd, S. and L. Vandenberghe (2004). Convex optimization. Cambridge.

Computing a MAP Estimate is Non Convex

- For non-convex functions is hard to find their global optimum.
- Simulated annealing or genetic algorithms claim to find the global optimum but under unrealistic assumptions (e.g., if they are allowed to be cooled "infinitely slowly", or allowed to run "infinitely long").
- In practice, we will run a local optimizer, perhaps using **multiple random restarts** to increase out chance of finding a "good" local optimum.
- A convex method for fitting mixtures of Gaussians has been proposed. The idea is to assign one cluster per data point, and select from amongst them, using a convex ℓ_1-type penalty, rather than trying to optimize the locations of the cluster centers. This is essentially an unsupervised version of the approach used in sparse kernel logistic regression.
- Note, however, that the ℓ_1 penalty, although convex, is not necessarily a good way to promote sparsity. *Some of the best sparsity-promoting methods use non-convex penalties, and use EM to optimize them.*

- Lashkari, D. and P. Golland (2007). Convex clustering with examplar-based models. In *NIPS*.

EM For DGM with Hidden Variables

- We can generalize the ideas behind EM for mixtures of experts to compute the MLE or MAP estimate for an arbitrary DGM. We could use gradient-based methods but it is much simpler to use EM.
- In the E step, we estimate the hidden variables, and in the M step, we compute the MLE using these filled-in values.
- Here we assume all CPDs are tabular. Let us write each CPT as follows:

$$p(x_{it} \mid \boldsymbol{x}_{i,pa(t)}, \boldsymbol{\theta}_t) = \prod_{c=1}^{K_{pa(t)}} \prod_{k=1}^{K_t} \theta_{tck}^{\mathbb{I}(x_{it}=k, \boldsymbol{x}_{i,pa(t)}=c)}$$

- The log-likelihood of the complete data is given by

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{t=1}^{V} \sum_{c=1}^{K_{pa(t)}} \sum_{k=1}^{K_t} N_{tck} \log \theta_{tck}, \; N_{tck} = \sum_{i=1}^{N} \mathbb{I}(x_{it} = k, \boldsymbol{x}_{i,pa(t)} = c)$$

- N_{tck} are the empirical counts. Hence the expected complete data log-likelihood has the form ($\mathcal{D}_i$ are all the visible variables at case i)

$$\mathbb{E} \log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{t=1}^{V} \sum_{c=1}^{K_{pa(t)}} \sum_{k=1}^{K_t} \overline{N}_{tck} \log \theta_{tck}, \; N_{tck} = \sum_{i=1}^{N} \mathbb{E}\left[\mathbb{I}(x_{it} = k, \boldsymbol{x}_{i,pa(t)} = c)\right] = \sum_{i=1}^{N} p(x_{it} = k, \boldsymbol{x}_{i,pa(t)} = c \mid \mathcal{D}_i)$$

EM For DGM with Hidden Variables

- The $p(x_{it} = k, \boldsymbol{x}_{i,pa(t)} = c \mid \mathcal{D}_i)$ is known as a **family marginal**, and can be computed using any GM inference algorithm. The $\bar{N}_{tck}$ are the expected sufficient statistics, and constitute the output of the E step.
- Given these ESS, the M step has the simple form

$$\hat{\theta}_{tck} = \frac{\bar{N}_{tck}}{\sum_{k'} \bar{N}_{tjk'}}$$

- This can be proved by adding Lagrange multipliers to enforce

$$\sum_k \theta_{tjk} = 1$$

  to the expected complete data log likelihood, and then optimizing each parameter vector $\boldsymbol{\theta}_{tc}$ separately.
- We can modify this to perform MAP estimation with a Dirichlet prior by simply adding pseudo counts to the expected counts.

- Binder, J., D. Koller, S. J. Russell, and K. Kanazawa (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning 29*, 213– 244.
- Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis 19*, 191–201.

EM For the Student Distribution

- The Gaussian distribution is sensitive to outliers since the log probability only decays quadratically with distance from the center. A more robust alternative is the Student's $\mathcal{T}$ distribution.

- There is no closed form formula for the MLE of a Student's, even if we have no missing data, so we must resort to iterative optimization methods. The easiest is to use is EM, since it automatically enforces the constraints that v is positive and $\mathbf{\Sigma}$ symmetric positive definite.

- The resulting algorithm turns out to have a simple intuitive form.

- At first blush, it might not be apparent why EM can be used, since there is no missing data. The key idea is to introduce an "artificial" hidden or auxiliary variable in order to simplify the algorithm. In particular, we will exploit the fact that a Student distribution can be written as a **Gaussian scale mixture**:

$$\mathcal{T}(x_i \mid \mu, \Sigma, \nu) = \int \mathcal{N}(x_i \mid \mu, \Sigma / z_i) \mathcal{G}a\left(z_i \mid \frac{\nu}{2}, \frac{\nu}{2}\right) dz_i$$

EM For the Student Distribution

- This can be thought of as an "infinite" mixture of Gaussians, each one with a slightly different covariance matrix.

- Treating the z_i as missing data, we can write the complete data log likelihood as

$$\ell_c(\theta) = \sum_{i=1}^{N}\left[\log \mathcal{N}(x_i \mid \mu, \Sigma / z_i) + \log \mathcal{G}a(z_i \mid \nu/2,\ \nu/2)\right]$$

$$= \sum_{hi=1}^{N}\left[-\frac{D}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{z_i}{2}\delta_i + \frac{\nu}{2}\log\frac{\nu}{2} - \log\Gamma(\frac{\nu}{2}) + \frac{\nu}{2}(\log z_i - z_i) + (\frac{D}{2} - 1)\log z_i\right]$$

- We have defined the Mahalanobis distance to be

$$\delta_i = (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})$$

EM For the Student Distribution

$$\ell_c(\theta) =$$

$$\sum_{hi=1}^{N}\left[-\frac{D}{2}log(2\pi)-\frac{1}{2}log|\mathbf{\Sigma}|-\frac{z_i}{2}\delta_i+\frac{\nu}{2}log\frac{\nu}{2}-log\Gamma(\frac{\nu}{2})+\frac{\nu}{2}(logz_i-z_i)+(\frac{D}{2}-1)logz_i\right]$$

- We can partition this into two terms, one involving $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$, and the other involving ν. We have, dropping irrelevant constants,

$$\ell_c(\theta) = L_N(\mu,\Sigma) + L_G(\nu)$$

$$L_N(\mu,\Sigma) = -\frac{1}{2}Nlog|\Sigma| - \frac{1}{2}\sum_{i=1}^{N} z_i\delta_i$$

$$L_G(\nu) = -Nlog\Gamma(\nu/2)+\frac{1}{2}N\nu log(\nu/2)+\frac{1}{2}\nu\sum_{i=1}^{N}(logz_i-z_i)$$

EM with ν known

- Let us first derive the algorithm with ν assumed known. In this case, we can ignore the L_G term, so we only need to figure out how to compute $\mathbb{E}[z_i]$ wrt the old parameters. From the joint log-likelihood we can see:

$$p(z_i \mid \boldsymbol{x}_i, \theta) = \mathcal{G}a\left(z_i \mid \frac{\nu + D}{2}, \frac{\nu + \delta_i}{2}\right)$$

- Now if $z_i \sim \mathcal{G}a(a, b)$, then $\mathbb{E}[z_i] = a/b$. Hence the E step at iteration t is

$$\bar{z}_i^{(t)} = \mathbb{E}\left[z_i \mid \boldsymbol{x}_i, \theta^{(t)}\right] = \frac{\nu^{(t)} + D}{\nu^{(t)} + \delta_i^{(t)}}$$

- The M step is obtained by maximizing $\mathbb{E}[L_N(\boldsymbol{\mu}, \boldsymbol{\Sigma})]$ to yieldType equation here.

$$\hat{\boldsymbol{\mu}}^{(t+1)} = \frac{\sum_i \bar{z}_i^{(t)} \boldsymbol{x}_i}{\sum_i \bar{z}_i^{(t)}}$$

$$\hat{\boldsymbol{\Sigma}}^{(t+1)} = \sum_i \bar{z}_i^{(t)} \left(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}^{(t+1)}\right)\left(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}^{(t+1)}\right)^T = \frac{1}{N}\sum_i \bar{z}_i^{(t)} \boldsymbol{x}_i \boldsymbol{x}_i^T - \left(\sum_i \bar{z}_i^{(t)}\right) \hat{\boldsymbol{\mu}}^{(t+1)} \left(\hat{\boldsymbol{\mu}}^{(t+1)}\right)^T$$

- z_i is the precision of measurement i, so if it is small, the corresponding data point is down-weighted when estimating the mean and covariance. This is how the Student's achieves robustness to outliers.

EM with ν Unknown

- To compute the MLE for the degrees of freedom, we first need to compute the expectation of $LG(v)$, which involves z_i and $\log z_i$. Now if $z_i \sim \mathcal{G}a(a, b)$, then one can show that

$$\bar{\ell}_i^{(t)} = \mathbb{E}\left[logz_i \mid \theta^{(t)} \right] = \Psi(a) - logb$$

where $\Psi(x) = \frac{d}{dx} log\ \Gamma(x)$ is the digamma function. Hence, we have

$$\bar{\ell}_i^{(t)} = \Psi\left(\frac{\nu^{(t)} + D}{2}\right) - log\left(\frac{\nu^{(t)} + \delta_i^{(t)}}{2}\right) = log(\bar{z}_i^{(t)}) + \Psi\left(\frac{\nu^{(t)} + D}{2}\right) - log\left(\frac{\nu^{(t)} + D}{2}\right)$$

where we used: $\bar{z}_i^{(t)} = \frac{\nu^{(t)} + D}{\nu^{(t)} + \delta_i^{(t)}} \Rightarrow \nu^{(t)} + \delta_i^{(t)} = \frac{\nu^{(t)} + D}{\bar{z}_i^{(t)}}$

- Substituting, we have

$$\mathbb{E}[L_G(\nu)] = -Nlog\ \Gamma(\nu/2) + \frac{N\nu}{2} log(\nu/2) + \frac{\nu}{2} \sum_i (\bar{\ell}_i^{(t)}(t) - \bar{z}_i^{(t)})$$

- The gradient of this expression is equal to

$$\frac{d}{dv}\mathbb{E}[L_G(\nu)] = -\frac{N}{2}\Psi(\nu/2) + \frac{N}{2} log(\nu/2) + \frac{N}{2} + \frac{1}{2}\sum_i (\bar{\ell}_i^{(t)}(t) - \bar{z}_i^{(t)})$$

EM with ν Unknown

$$\frac{d}{dv}\mathbb{E}[L_G(\nu)] = -\frac{N}{2}\Psi(\nu/2) + \frac{N}{2}log(\nu/2) + \frac{N}{2} + \frac{1}{2}\sum_i (\bar{\ell}_i^{(t)}(t) - \bar{z}_i^{(t)})$$

- This has a unique solution in the interval (0,+∞] which can be found using a 1d constrained optimizer
- Performing a gradient-based optimization in the M step, rather than a closed-form update, is an example of what is known as the **generalized EM** algorithm.
- One can show that EM will still converge to a local optimum even if we only perform a "partial" improvement to the parameters in the M step.

Mixtures of Student Distributions

- It is easy to extend the above methods to fit a mixture of Student distributions (see Lo, 2009, pp. 57).
- We have two latent variables for each data point: z_i which defines the cluster to which each data point belongs and $u_i \sim \mathcal{G}a\left(\frac{\nu_k}{2}, \frac{\nu_k}{2}\right)$ which is the latent scale for point i.
- The complete data log-likelihood is: $\ell_c(\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik}\left(log\left(\pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k/u_i)\right) + \right.$

- Lo, C. H. (2009). *Statistical methods for high throughput genomics*. Ph.D. thesis, UBC.

Mixtures of Student Distributions

- We need to compute the expected complete data log-likelihood: $Q(\boldsymbol{\theta}) = \mathbb{E}[\ell_c(\boldsymbol{\theta})]$, using $\tilde{z}_{ik} = \mathbb{E}[z_{ik}|\boldsymbol{x_i}, \boldsymbol{\theta}] = p(z_i = k|\boldsymbol{x_i}, \boldsymbol{\theta}), \tilde{u}_{ik} = \mathbb{E}[u_i|\boldsymbol{x_i}, z_{ik} = 1, \boldsymbol{\theta}] = p(z_i = k|\boldsymbol{x_i}, \boldsymbol{\theta})$. These can be computed as follows:

$$\tilde{z}_{ik} = \frac{\pi_k T(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \nu_k)}{\sum_{k'=1}^{K} \pi_{k'} T(\boldsymbol{x}_i|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'}, \nu_{k'})}, \tilde{u}_{ik} = \frac{\nu_k + D}{\nu_k + \delta_{ik}},$$

$$\tilde{s}_{ik} = log\tilde{u}_{ik} + \Psi\left(\frac{\nu_k + D}{2}\right) - log\left(\frac{\nu_k + D}{2}\right)$$

$$\pi_k = \frac{\gamma_k}{N}, \gamma_k = \sum_{i=1}^{N} \tilde{z}_{ik}, \mu_k = \frac{\sum_{i=1}^{N} \tilde{z}_{ik}\tilde{u}_{ik}\boldsymbol{x}_i}{\sum_{i}^{N} \tilde{z}_{ik}\tilde{u}_{ik}},$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{N} \tilde{z}_{ik}\tilde{u}_{ik}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\gamma_k}$$

- In the M-step, the optimization for ν_k is not exact. The needed gradient of $Q(\boldsymbol{\theta})$ is:

$$\frac{dQ}{d\nu_k} = \frac{N_k}{2}\left(log\frac{\nu}{2} + 1 - \Psi\left(\frac{\nu}{2}\right)\right) + \frac{1}{2}\sum_{i=1}^{N} \tilde{z}_{ik}(\tilde{s}_{ik} - \tilde{u}_{ik})$$

  - Lo, C. H. (2009). Sta $\tilde{s}_{ik}$tistical methods for high throughput genomics. Ph.D. thesis, UBC.

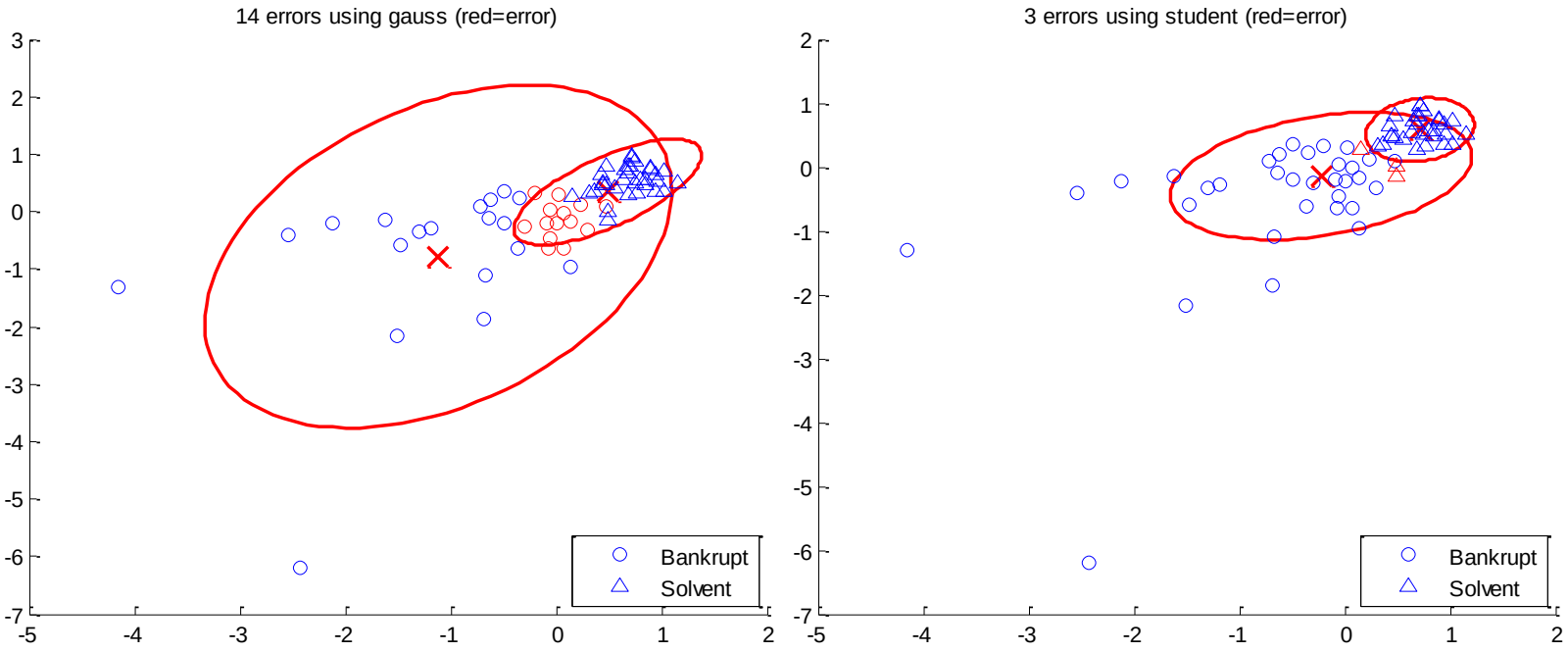
Mixtures of Student Distributions

- We can identify outliers on this model by looking for small values of $\tilde{u}_{ik}$. This takes values on the interval $(0,1 + D/\nu_k)$ and for moderate ν_k its mean is about 1. A threshold of 0.5 is good.
- In the example we have $N = 66$, $D = 2$ data set regarding the bankruptcy patterns of certain companies. The first feature specifies the ratio of retained earnings (RE) to total assets, and the second feature specifies the ratio of earnings before interests and taxes (EBIT) to total assets. We fit two models to this data, ignoring the class labels: a mixture of 2 Gaussians, and a mixture of 2 Students.
- We then use each fitted model to classify the data. We compute the most probable cluster membership and treat this as $\hat{y}_i$. We then compare $\hat{y}_i$ to the true labels y_i and compute an error rate.
- If this is more than 50%, we permute the latent labels (i.e., we consider cluster 1 to represent class 2 and vice versa), and then recompute the error rate. Points which are misclassified are then shown in red.

- Lo, C. H. (2009). *Statistical methods for high throughput genomics*. Ph.D. thesis, UBC.

Mixtures of Student Distributions

- Points which are misclassified are shown in red.
- We see that the Student model made 4 errors, the Gaussian model made 21. This is because the class-conditional densities contain some extreme values, causing the Gaussian to be a poor choice.



mixStudentBankruptcyDemo from PMTK

- Lo, C. H. (2009). *Statistical methods for high throughput genomics*. Ph.D. thesis, UBC.

EM For Probit Regression

- Probit regression has the form $p(y_i = 1|z_i) = \mathbb{I}(z_i > 0)$, where $z_i \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_i, 1)$ is latent.
- We fit this model using EM. Let ϕ and Φ the pdf and cdf of $\mathcal{N}(\mathbf{w}|0, 1)$.
- The complete data log likelihood has the following form, assuming a $\mathcal{N}(\mathbf{0}, \mathbf{V}_0)$ prior on $\mathbf{w}$:

$$\begin{aligned}\ell(\boldsymbol{z}, \boldsymbol{w} \mid \boldsymbol{V}_0) &= \log p(\boldsymbol{y} \mid \boldsymbol{z}) + \log \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{X}\boldsymbol{w}, \boldsymbol{I}) + \log \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{0}, \boldsymbol{V}_0) \\ &= \sum_i \log p(y_i \mid z_i) - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{z} - \boldsymbol{X}\boldsymbol{w}) - \frac{1}{2}\boldsymbol{w}^T \boldsymbol{V}_0^{-1} \boldsymbol{w} + const\end{aligned}$$

- The posterior in the E step is a **truncated Gaussian**:

$$p(z_i \mid y_i, \boldsymbol{x}_i, w) = \begin{cases} \mathcal{N}(z_i \mid \boldsymbol{w}^T \boldsymbol{x}_i, 1)\mathbb{I}(z_i > 0) \text{ if } y_i = 1 \\ \mathcal{N}(z_i \mid \boldsymbol{w}^T \boldsymbol{x}_i, 1)\mathbb{I}(z_i < 0) \text{ if } y = 0 \end{cases}$$

- We see that $\mathbf{w}$ only depends linearly on $\mathbf{z}$, so we just need to compute $\mathbb{E}[z_i|y_i, \mathbf{x}_i, \mathbf{w}]$. One can show that:

$$\mathbb{E}[z_i \mid \boldsymbol{w}, \boldsymbol{x}_i] = \begin{cases} -\mu_i + \frac{\phi(\mu_i)}{1 - \Phi(-\mu_i)} = \mu_i + \frac{\phi(\mu_i)}{\Phi(\mu_i)} \text{ if } y_i = 1 \\ \mu_i - \frac{\phi(\mu_i)}{\Phi(-\mu_i)} = \mu_i - \frac{\phi(\mu_i)}{1 - \Phi(\mu_i)} \text{ if } y_i = 0 \end{cases}, \text{where } \mu_i = \boldsymbol{w}^T \boldsymbol{x}_i$$

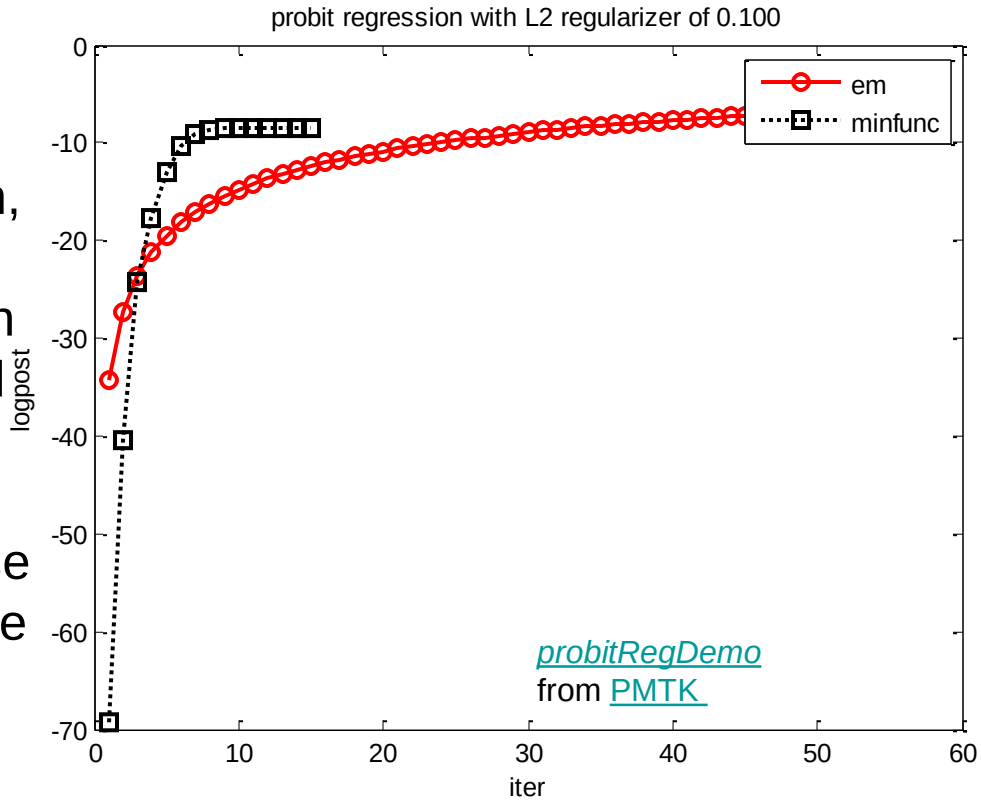
- In the M step, we estimate $\mathbf{w}$ using ridge regression, where $\boldsymbol{\mu} = \mathbb{E}[\mathbf{z}]$ is the output we are trying to predict.

EM For Probit Regression

- Specifically, we have

$$\widehat{\boldsymbol{w}} = (\boldsymbol{V}_0^{-1} + \boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\mu}$$

- The EM algorithm is simple, but can be much slower than direct gradient methods.
- This is because the posterior entropy in the E step is quite high, since we only observe that *z* is positive or negative, but are given no information from the likelihood about its magnitude.
- Using a stronger regularizer can help speed convergence, because it constrains the range of plausible *z* values.
- In addition, one can use various speedup tricks, such as data augmentation.



- van Dyk, D. and X.-L. Meng (2001). The Art of Data Augmentation. *J. Computational and Graphical Statistics* *10*(1), 1–50.

The EM Algorithm Revisited

The EM Algorithm in General

- Let **X** be the observed variables, **Z** denote all latent variables and θ the set of all parameters.
- Our goal is as before to maximize the likelihood:

$$p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \int_{\boldsymbol{Z}} p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta}) d\boldsymbol{Z}$$

- We assume that $p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})$ is easier to compute than $p(\boldsymbol{X} \mid \boldsymbol{\theta})$.
- *Introduce an arbitrary distribution q(**Z**) over the latent variables*. One can then show:

$$\ln p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q \| p)$$

where:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \int q(\boldsymbol{Z}) \ln \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\theta})}{q(\boldsymbol{Z})} \right\} d\boldsymbol{Z}$$

$$KL(q \| p) = -\int q(\boldsymbol{Z}) \ln \left\{ \frac{p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{Z})} \right\} d\boldsymbol{Z}, \; KL(q \| p) \geq 0, \; KL(q \| p) = 0 \; \text{if} \; q(\boldsymbol{Z}) = p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})$$

The EM Algorithm in General

- For the proof of the identity shown earlier, note that:

$$\mathcal{L}(q,\boldsymbol{\theta}) = \int q(\boldsymbol{Z}) \ln \left\{ \frac{p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta})}{q(\boldsymbol{Z})} \right\} d\boldsymbol{Z}$$

$$= \int q(\boldsymbol{Z}) \ln \Big(p(\boldsymbol{X},\boldsymbol{Z}\mid\boldsymbol{\theta}) \Big) d\boldsymbol{Z} - \int q(\boldsymbol{Z}) \ln q(\boldsymbol{Z}) d\boldsymbol{Z}$$

$$= \int q(\boldsymbol{Z}) \Big[\ln \Big(p(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\theta}) \Big) + \ln p(\boldsymbol{X}\mid\boldsymbol{\theta}) \Big] d\boldsymbol{Z} - \int q(\boldsymbol{Z}) \ln q(\boldsymbol{Z}) d\boldsymbol{Z}$$

$$= \int q(\boldsymbol{Z}) \ln \Big(p(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\theta}) \Big) d\boldsymbol{Z} + \ln p(\boldsymbol{X}\mid\boldsymbol{\theta}) - \int q(\boldsymbol{Z}) \ln q(\boldsymbol{Z}) d\boldsymbol{Z}$$

$$= \int q(\boldsymbol{Z}) \ln \left(\frac{p(\boldsymbol{Z}\mid\boldsymbol{X},\boldsymbol{\theta})}{q(\boldsymbol{Z})} \right) d\boldsymbol{Z} + \ln p(\boldsymbol{X}\mid\boldsymbol{\theta}) = \ln p(\boldsymbol{X}\mid\boldsymbol{\theta}) - KL(q \,\|\, p)$$

Lower Bound on Model Evidence p(X|θ)

- KL(q||p) is Kullback-Leibler distance between q and the posterior p(**Z**|**X**,**q**)

$$KL(q\|p) = -\int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} d\mathbf{Z}, \; KL(q\|p) \geq 0, \; KL(q\|p) = 0 \; if \; q(\mathbf{Z}) = p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$$

- From

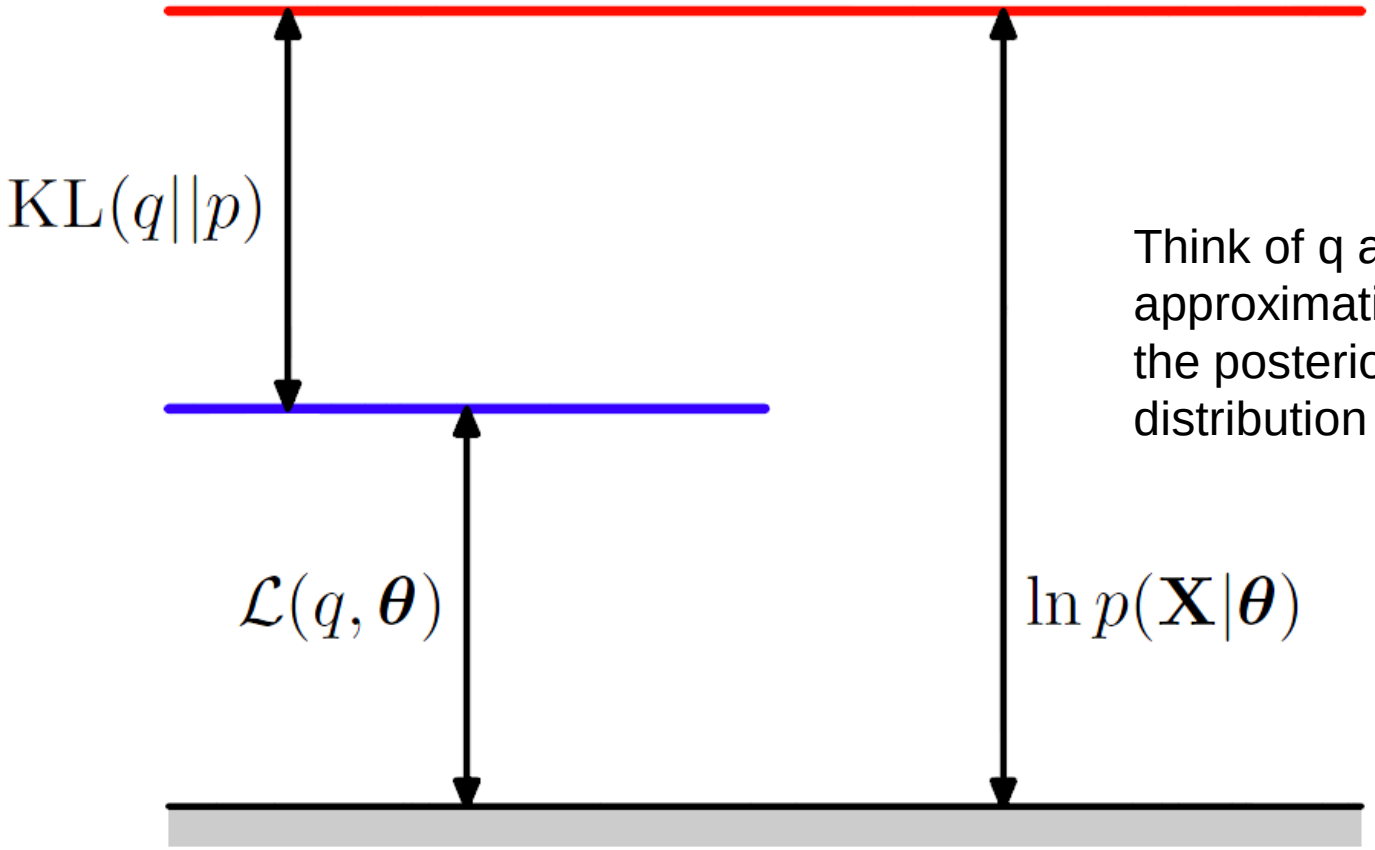
$$\ln p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q\|p)$$

it follows that $\mathcal{L}(q, \boldsymbol{\theta})$ *is a lower bound of lnp(**X**|**q**):*

$$\ln p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q\|p) \geq \mathcal{L}(q, \boldsymbol{\theta})$$

- Maximizing $\mathcal{L}(q, \boldsymbol{\theta})$ over *q(**Z**)* would give the true posterior but this is not computationally tractable.

Variational Lower Bound



Think of q as an approximation to the posterior distribution $p(\boldsymbol{Z} \mid \boldsymbol{X}, \boldsymbol{\theta})$

Working with lnp(**X|q**) is intractable – e.g. it is log of sum of Gaussians- Work instead with $\mathcal{L}(q,\boldsymbol{\theta})$

Approximate $\ln p(\boldsymbol{X} \mid \boldsymbol{\theta})$ *with its lower bound* $\mathcal{L}(q,\boldsymbol{\theta})$

Make $\mathcal{L}(q,\boldsymbol{\theta})$ *as big as possible or equivalently*

make $KL(q \,||\, p)$ *as small as possible*

The EM Algorithm in General

- Maximizing $\mathcal{L}(q,\boldsymbol{\theta})$ over a free form q would give the true posterior but this is not computationally tractable

$$q(\boldsymbol{Z}) = p(\boldsymbol{Z} \mid \boldsymbol{X}, \theta)$$

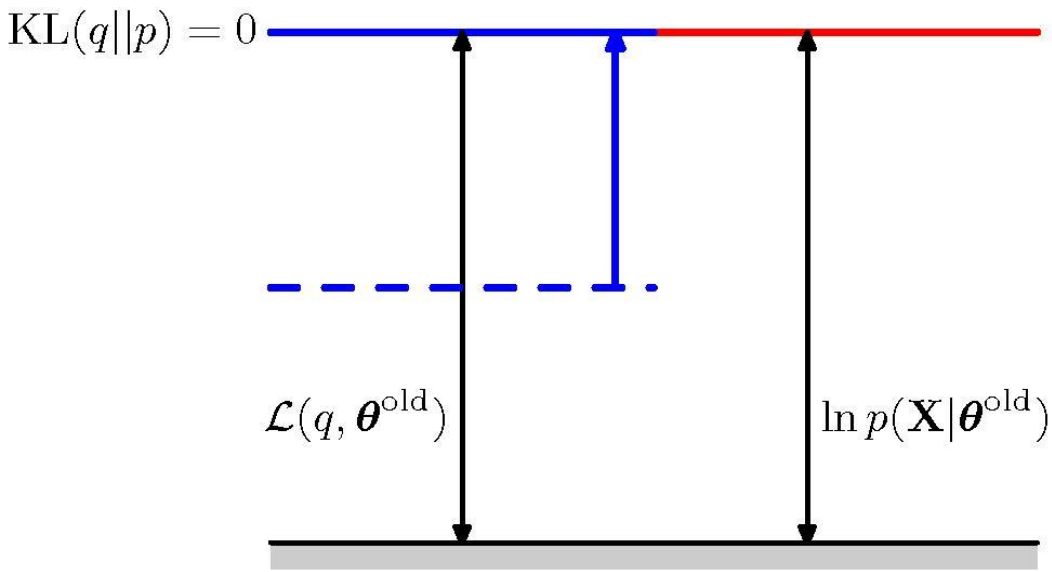
- The EM algorithm is a two-stage iterative optimization technique for finding maximum likelihood solutions. We can use the decomposition

$$\ln p(\boldsymbol{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q,\boldsymbol{\theta}) + KL(q \| p)$$

to define the EM algorithm and to demonstrate that it does indeed maximize the log likelihood.

The EM Algorithm in General

- Suppose that the current value of the parameter vector is θ^{old}.
- In the E step, the lower bound $\mathcal{L}(q, \theta^{old})$ *is maximized with respect to* $q(\mathbf{Z})$ *while holding* θ^{old} *fixed*.
- The solution to this maximization problem is easily seen by noting that the value of $\ln p(\mathbf{X}|\theta^{old})$ *does not depend on* $q(\mathbf{Z})$ *and so the largest value of* $\mathcal{L}(q, \theta^{old})$ *will occur when the KL divergence vanishes, i.e. when* $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta^{old})$.
- In this case, the lower bound will equal the log likelihood.



The EM Algorithm in General

- The lower bound then becomes:

$$\mathcal{L}(q,\boldsymbol{\theta}) = \sum_{\mathbf{Z}} \underbrace{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}_{old})}_{\text{play the role of responsibilities}} \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})}{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}_{old})} \right\}$$

- This as a function of θ is the expected complete-data log likelihood up to an additive constant.

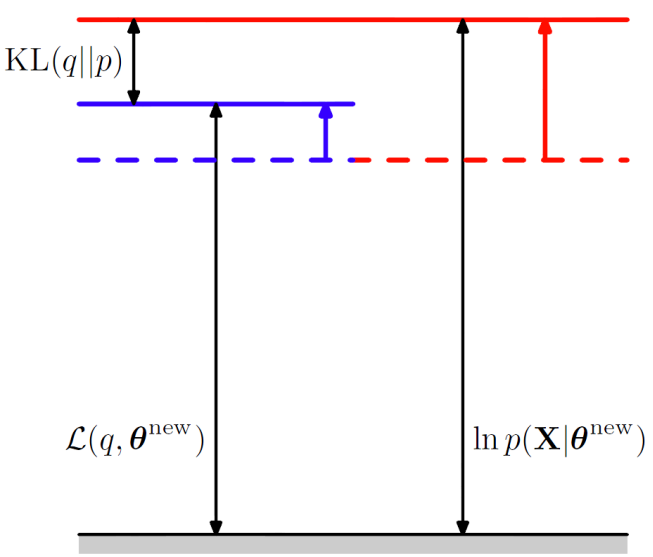
The EM Algorithm in General

- In the subsequent M step, the distribution q(**Z**) is held fixed and $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized with respect to $\boldsymbol{\theta}$ to give some new value $\boldsymbol{\theta}^{\text{new}}$.

- This causes $\mathcal{L}$ to increase (unless it is already at a maximum), which will cause the corresponding log likelihood function to increase.

- Because q is determined using the old parameter values rather than the new values and is held fixed during the M step, it will not equal the new posterior distribution $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{new}})$, and hence there will be a nonzero KL divergence.

- The increase in the log likelihood function is therefore greater than the increase in the lower bound, as shown.



The EM Algorithm in General

- We can show that the maximization in the M step is that of the expected value of the complete data log likelihood.

- Indeed, if we substitute $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{X},\theta)$

  in the lower bound $\mathcal{L}(q,\theta) = \int q(\boldsymbol{Z}) \ln \left\{ \frac{p(\boldsymbol{X},\boldsymbol{Z}|\theta)}{q(\boldsymbol{Z})} \right\} d\boldsymbol{Z}$

  we see that the lower bound after the E step becomes

$$\mathcal{L}(q,\boldsymbol{\theta}) = \int q(\boldsymbol{Z}) \ln \left\{ \frac{p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} \right\} d\boldsymbol{Z} = \int p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{old}) \ln \frac{p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta})}{p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{old})} d\boldsymbol{Z}$$

$$= \int p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{old}) \ln p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta}) d\boldsymbol{Z} - \int p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{old}) \ln p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta}^{old})$$

$$= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) + const$$

  where the constant is the entropy of q and therefore independent of θ.

- $\mathcal{Q}(\mathbf{q},\mathbf{q}^{old})$ is the expectation of the complete data log likelihood wrt posterior of the latent variables.

The EM Algorithm in General

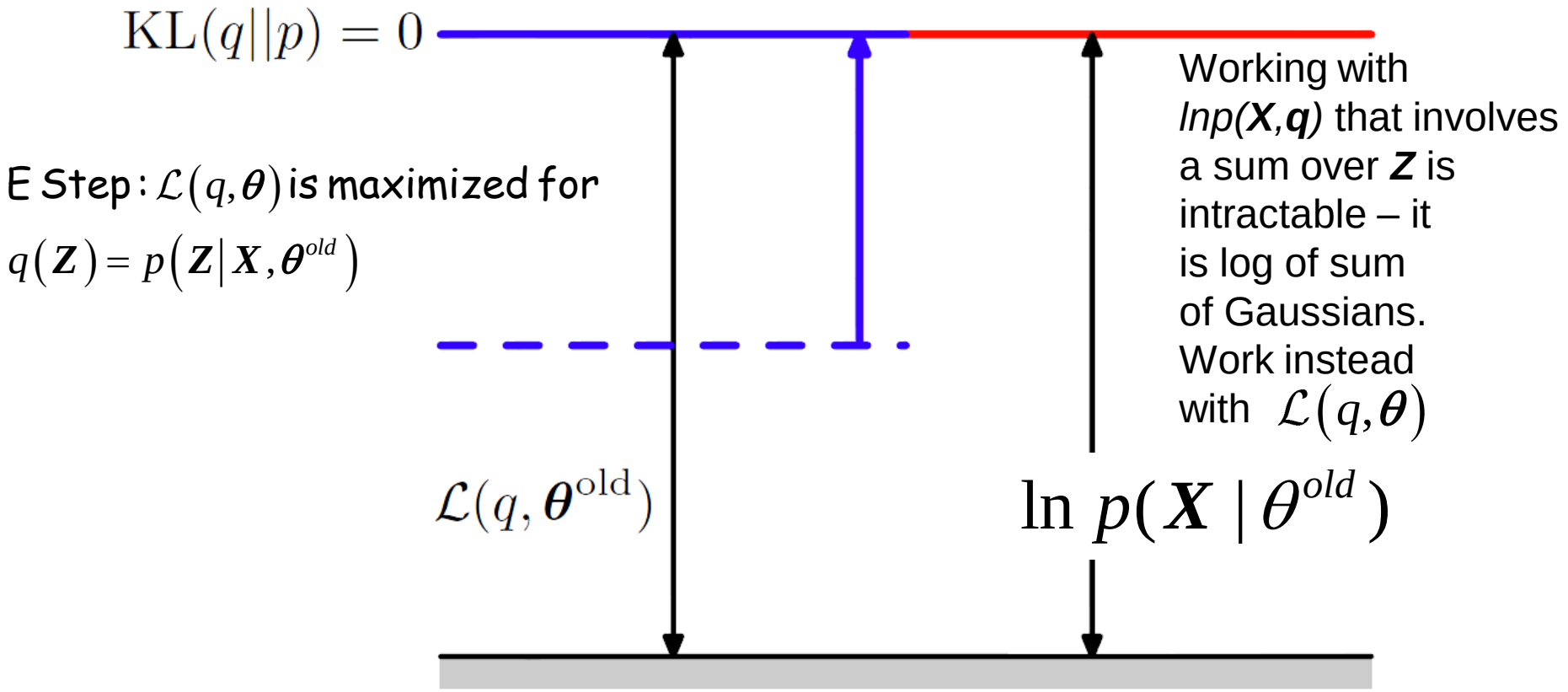
- Thus in the M step, we maximize the expectation of the complete-data log likelihood, as we saw earlier in the case of mixtures of Gaussians.

$$\mathcal{L}(q,\boldsymbol{\theta}) = \int p\left(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{old}\right)\ln p\left(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}\right)d\mathbf{Z} - \int p\left(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{old}\right)\ln p\left(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}^{old}\right)$$
$$= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) + const$$

- Note that *the variable **θ** over which we are optimizing appears only inside the logarithm.*

- *If p(**Z**,**X**|**θ**) is from the exponential family, then the log cancels the exponential* leading to an M step that will be simpler than the maximization of p(**X**|**θ**).

E-Step

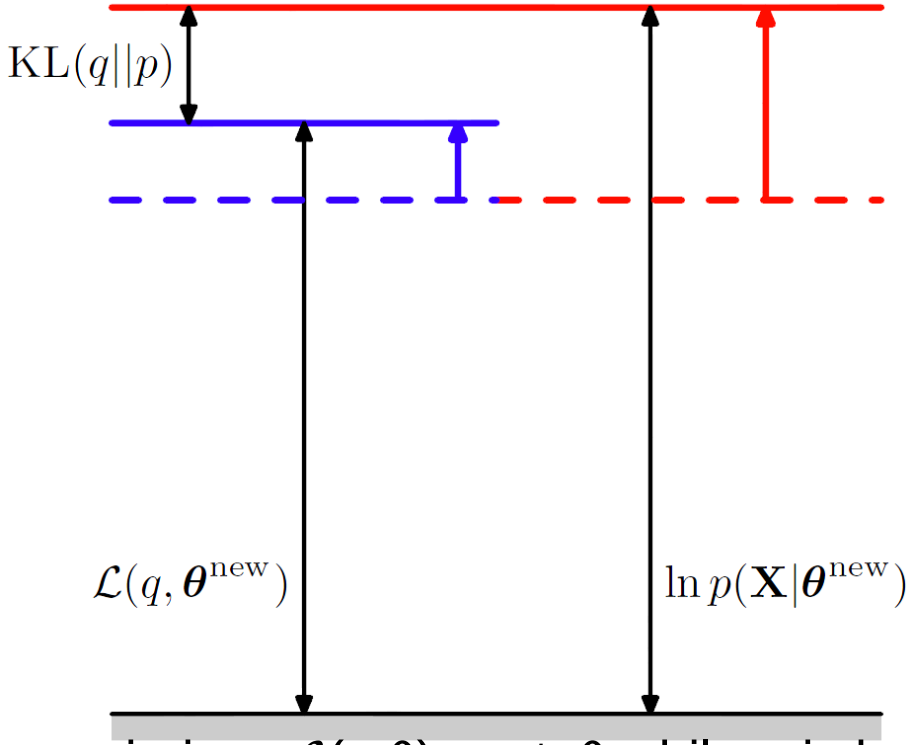


- E-step: Maximizes $\mathcal{L}(q,\theta)$ w.r.t. q for fixed θ

$$\mathcal{L}(q,\boldsymbol{\theta}) = \ln p(\mathbf{X}|\boldsymbol{\theta}) - KL\left(q(\mathbf{Z}) \| p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})\right)$$

- At every step, the EM algorithm increases this lower bound on the log probability on the data (log-likelihood function)

M-Step



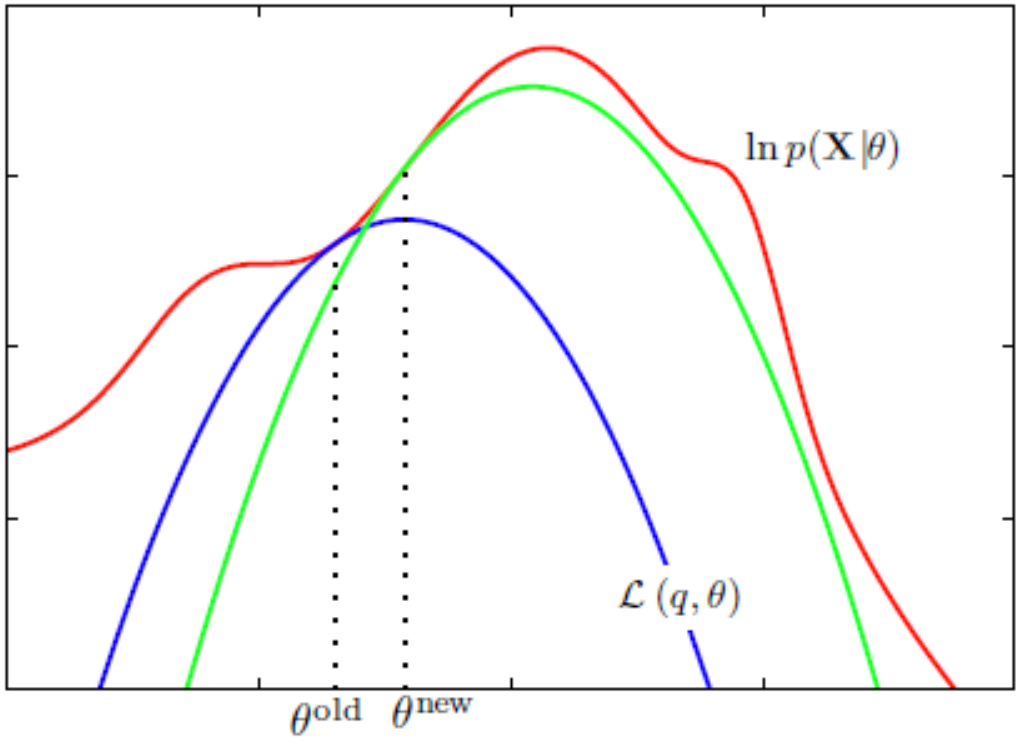
- The M-step, maximizes $\mathcal{L}(q,\theta)$ w.r.t. θ while q is kept fixed (function of θ^{old}).

$$\mathcal{L}(q,\theta) = \int q(\boldsymbol{Z}) \ln p\left(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}\right) d\boldsymbol{Z} - \int q(\boldsymbol{Z}) \ln q(\boldsymbol{Z}) d\boldsymbol{Z}$$

- The $\ln p\left(\boldsymbol{X}|\theta_{new}\right)$ goes up at least as much as $\mathcal{L}$ (q,θ^{new}) creating *KL(q||p).*
- *$\mathcal{L}$ maximized for* $\boldsymbol{\theta} = \arg\max_{\theta} \int q(\boldsymbol{Z}) \ln p\left(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}\right) d\boldsymbol{Z}$

EM in the Space of Parameters

- Can view the EM algorithm in the space of parameters.
- The red curve is the incomplete data log likelihood function whose value we wish to maximize.
- We start with some initial parameter value $\theta^{(old)}$, and in the first E step we evaluate the posterior distribution over latent variables, which gives rise to a lower bound $\mathcal{L}(q, \theta^{(old)})$ whose value equals the log likelihood at $\theta^{(old)}$ as shown by the blue curve.



emLogLikelihoodMax
from PMTK

EM in the Space of Parameters

- Note that the bound $\mathcal{L}(q,\theta)$ with $q(\mathbf{Z})=p(\mathbf{Z}|\mathbf{X},\theta^{(old)})$ is tangent to the log likelihood $\ln p(\mathbf{X}|\theta)$ at $\theta^{(old)}$ i.e. that both curves have the same gradient.

- This is obvious after noting that KL(q||p) is at its minimum (i.e. 0) when $q(\mathbf{Z})=p(\mathbf{Z}|\mathbf{X},\theta^{(old)})$.

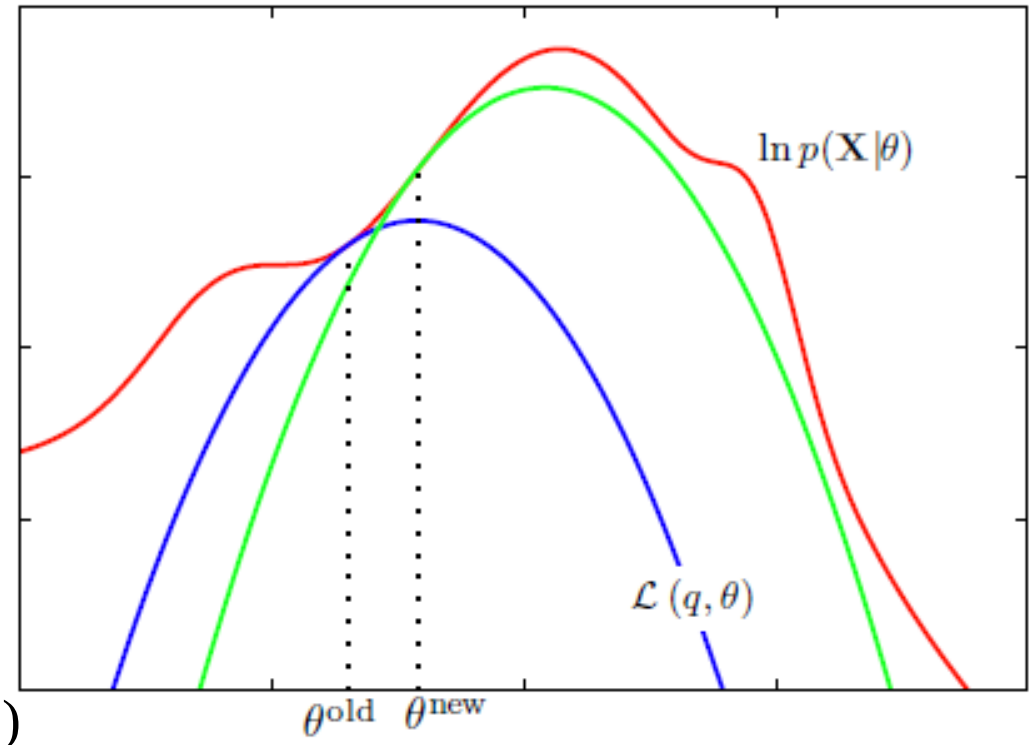
- This means that:

$$\frac{\partial}{\partial\theta} KL(q\,||\,p) = 0$$

since $p(\mathbf{Z}|\mathbf{X},\theta)$ depends on θ.

- From $\mathcal{L}(q,\theta) = \ln p(\boldsymbol{X}\,|\,\boldsymbol{\theta}) - KL(q\,||\,p)$ we conclude that:

$$\frac{\partial}{\partial\theta}\mathcal{L}\left(q,\boldsymbol{\theta}^{(old)}\right) = \frac{\partial}{\partial\theta}\ln p(\boldsymbol{X}\,|\,\boldsymbol{\theta}^{(old)})$$

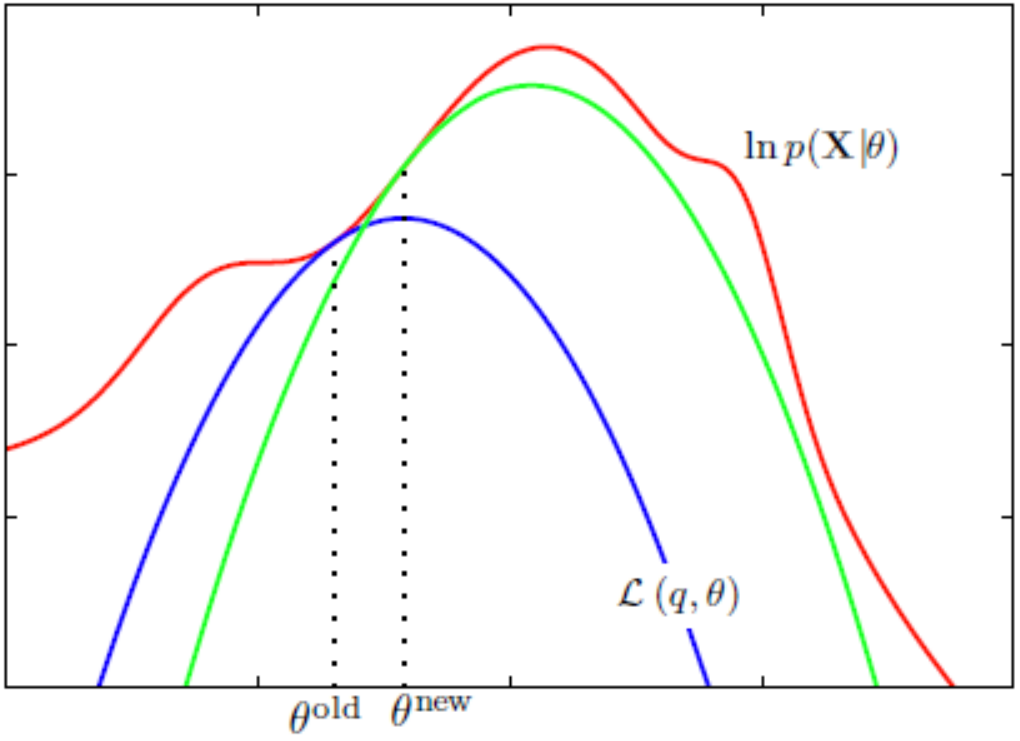


EM in the Space of Parameters

- *The lower bound $\mathcal{L}(q,\theta)$ is a convex function having a unique maximum* (for mixture components from the exponential family).

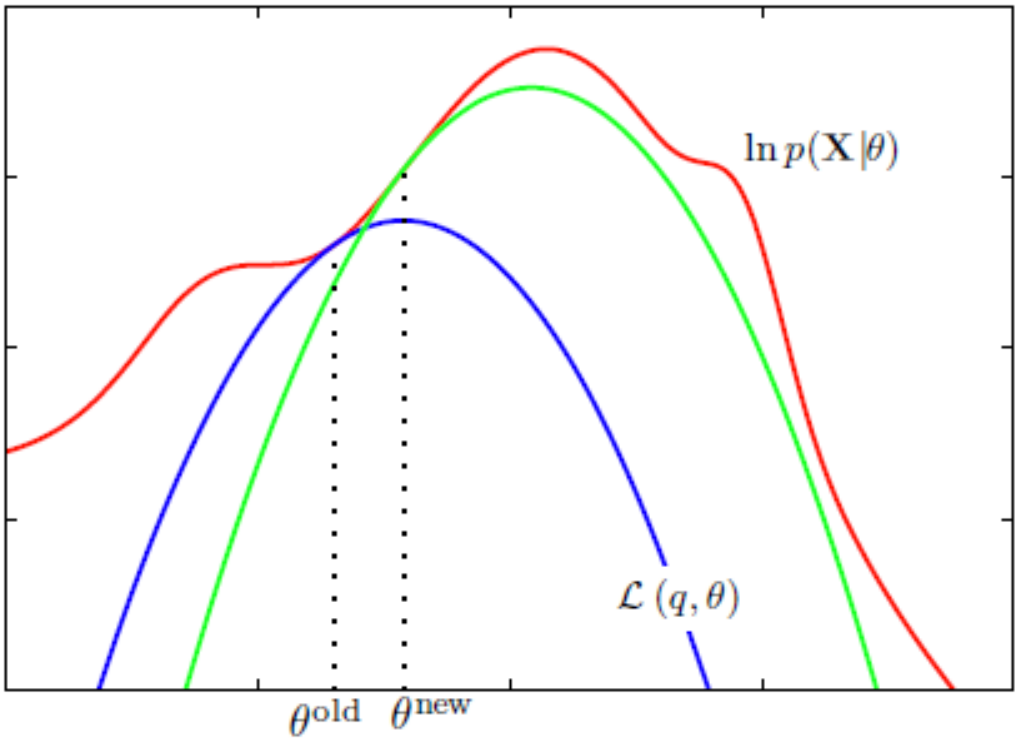
- In the M step, the bound $\mathcal{L}(q,\theta)$ is maximized giving the value $\theta^{(new)}$ which gives a larger value of log likelihood than $\theta^{(old)}$

- *The subsequent E step then constructs a bound $\mathcal{L}(q,\theta^{(new)})$ that is tangential at $\theta^{(new)}$ as shown by the green curve.*



EM in General: Parameter Space Representation

- E-Step resets bound $\mathcal{L}(q,\theta)$ on $\ln p(\mathbf{X}|\theta)$ at $\theta=\theta^{old}$, it is
  - Tight at $\theta=\theta^{old}$,
  - Tangetial at $\theta=\theta^{old}$,
  - Convex (easy) in θ for exponential family mixture components



EM In General

- Consider an i.i.d. data set, **X** that comprises of N data points $\{\mathbf{x}_n\}$. **Z** comprises N corresponding latent variables $\{\mathbf{z}_n\}$, n = 1, . . . , N.

- From the independence assumption, we have

$$p(\boldsymbol{X},\boldsymbol{Z}) = \prod_n p\left(\boldsymbol{x}_n,\boldsymbol{z}_n\right)$$

and, by marginalizing over $\{\mathbf{z}_n\}$ we have

$$p(\boldsymbol{X}) = \prod_n p\left(\boldsymbol{x}_n\right)$$

- Using the sum and product rules, we see that the posterior probability that is evaluated in the E step takes the form

$$p(\boldsymbol{Z} \mid \boldsymbol{X},\theta) = \frac{p(\boldsymbol{X},\boldsymbol{Z} \mid \theta)}{\sum_{\boldsymbol{Z}} p(\boldsymbol{X},\boldsymbol{Z} \mid \theta)} = \frac{\prod_{n=1}^{N} p(\boldsymbol{x}_n,\boldsymbol{z}_n \mid \theta)}{\sum_{\boldsymbol{Z}} \prod_{n=1}^{N} p(\boldsymbol{x}_n,\boldsymbol{z}_n \mid \theta)} = \prod_{n=1}^{N} p(\boldsymbol{z}_n \mid \boldsymbol{x}_n,\theta)$$

EM In General

$$p(\mathbf{Z} \mid \mathbf{X}, \theta) = \prod_{n=1}^{N} p(\mathbf{z}_n \mid \mathbf{x}_n, \theta)$$

- *Thus the posterior distribution of the latent variables also factorizes with respect to n.*

- For the Gaussian mixture model*: the responsibility that each of the mixture components takes for a particular $\mathbf{x}_n$ depends only on the value of $\mathbf{x}_n$ and on θ, not on the values of the other data points.*

- We have seen that both the E and the M steps of the EM algorithm are increasing the value of a well-defined bound on the log likelihood function and that the complete EM cycle will change the model parameters in such a way as to cause the log likelihood to increase (unless it is already at a maximum, in which case the parameters remain unchanged).

Using EM to Maximize p(θ|X)

- We *can also use the EM algorithm to maximize the posterior distribution p(θ|X)* for models in which we have introduced a prior p(θ) over the parameters.

- Note that as a function of θ, we have p(θ|X) = p(θ,X)/p(X) and so

$$\ln p(\boldsymbol{\theta}|\mathbf{X}) = \ln p(\boldsymbol{\theta},\mathbf{X}) - \ln p(\mathbf{X})$$

- Making use of the decomposition $\ln p(\mathbf{X}|\theta) = \mathcal{L}(q,\theta) + KL(q\|p)$

$$\begin{aligned}\ln p(\boldsymbol{\theta}|\mathbf{X}) &= \mathcal{L}(q, \boldsymbol{\theta}) + KL(q\|p) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X}) \\ &\geq \mathcal{L}(q, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X})\end{aligned}$$

  where ln p(**X**) is a constant.

- We can again optimize the right-hand side alternately with respect to q and θ.

Using EM to Maximize p(θ|X)

$$\ln p(\boldsymbol{\theta}|\mathbf{X}) = \mathcal{L}(q, \boldsymbol{\theta}) + KL(q||p) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X})$$
$$\geq \mathcal{L}(q, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X})$$

$$\mathcal{L}(q,\boldsymbol{\theta}) = \int p\left(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{old}\right)\ln p\left(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}\right)d\mathbf{Z} - \int p\left(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{old}\right)\ln p\left(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}^{old}\right)$$
$$= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{old}) + const$$

- Since q appears only in $\mathcal{L}(q, \boldsymbol{\theta})$, optimization with respect to q gives rise to the *same E step equations as for the standard EM*.
- *The M-step equations are modified through the introduction of the prior $\ln p(\boldsymbol{\theta})$, which requires only a small modification to the standard MLE M-step equations.*

Generalizations of the EM Algorithm

Generalizations of the EM Algorithm

- The EM algorithm breaks down the difficult problem of maximizing the likelihood function into two stages: the E and M steps, each of which are often simpler.
- For complex models, the E step and/or the M steps may be intractable.
- This leads to multiple possible extensions of the EM algorithm.

Generalized EM (GEM): Intractable M Step

- GEM addresses the problem of an intractable M step.
- Instead of maximizing $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, we instead change $\boldsymbol{\theta}$ in such a way as to increase its value.
- Since $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on the log likelihood function, each complete EM cycle of the GEM algorithm is guaranteed to increase the value of the log likelihood (unless the parameters already correspond to a local maximum).
- Use nonlinear optimization strategies (e.g. conjugate gradients algorithm) during the M step.
- This is called the *generalized EM or GEM algorithm*.
  - This is an unfortunate name since EM can be generalized in many other ways.

Generalized E Step

- We can similarly generalize the E step of the EM algorithm by *performing a partial, rather than complete, optimization of $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $q(\mathbf{Z})$.*

- We have seen that for any value of $\boldsymbol{\theta}$ there is a unique maximum of $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $q(\mathbf{Z})$ that corresponds to the posterior $q_{\theta}(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})$ and that for this choice of $q(\mathbf{Z})$, $\mathcal{L}(q, \boldsymbol{\theta})$ is equal to $\ln p(\mathbf{X}|\boldsymbol{\theta})$.

- It follows that any algorithm that converges to the global maximum of $\mathcal{L}(q, \boldsymbol{\theta})$ will find a value of $\boldsymbol{\theta}$ that is also a global maximum of the log likelihood $\ln p(\mathbf{X}|\boldsymbol{\theta})$.

- Provided $p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$ is a continuous function of $\boldsymbol{\theta}$ then, by continuity, *any local maximum of $\mathcal{L}(q, \boldsymbol{\theta})$ will also be a local maximum of $\ln p(\mathbf{X}|\boldsymbol{\theta})$.*

  - Neal, R. M. and G. E. Hinton (1999). A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 355–368. MIT Press.

ECM: Expectation Conditional Maximization

- Expectation conditional maximization, or ECM, algorithm, involves making several constrained optimizations within each M step.

- Optimize the parameters in the M step sequentially:

  - Partition the parameters into groups, and

  - *Break the M step into multiple steps each of which involves optimizing one of the subset of the parameters with the remainder held fixed.*

- Meng, X. L. and D. B. Rubin (1993). Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika* **80**, 267–278.

ECM Either (ECME)

$$\text{Complete Data Log Likelihood}: \ell_c\ \boldsymbol{\theta} = \sum_{i=1}^{N} \log p\ \boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\theta}$$

$$\text{Expected Complete Data Log Likelihood}: Q\ \boldsymbol{\theta}, \boldsymbol{\theta}^{t-1} = \mathbb{E}\left[\ell_c\ \boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{\theta}^{t-1}\right]$$

- This is a variant of ECM in which we maximize the expected complete data log likelihood (the *Q* function) as usual, or the observed data log likelihood, during one or more of the conditional maximization steps.
- The latter can be much faster since it ignores the results of the E step, and *directly optimizes the objective of interest*.
- An example of this is when fitting the Student's $\mathcal{T}$ distribution.

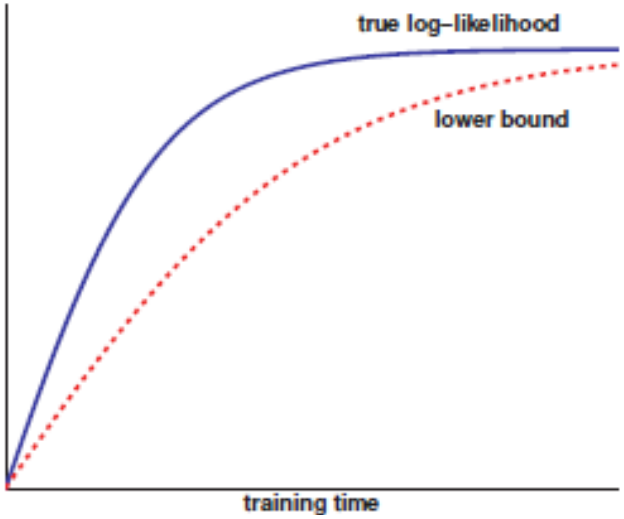
For fixed v, we can update $\boldsymbol{\Sigma}$ as usual, but then to update v, we replace the standard update of the form
$v_{t+1} = \arg\max_v Q((\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}, v), \boldsymbol{\theta}_t)$ with

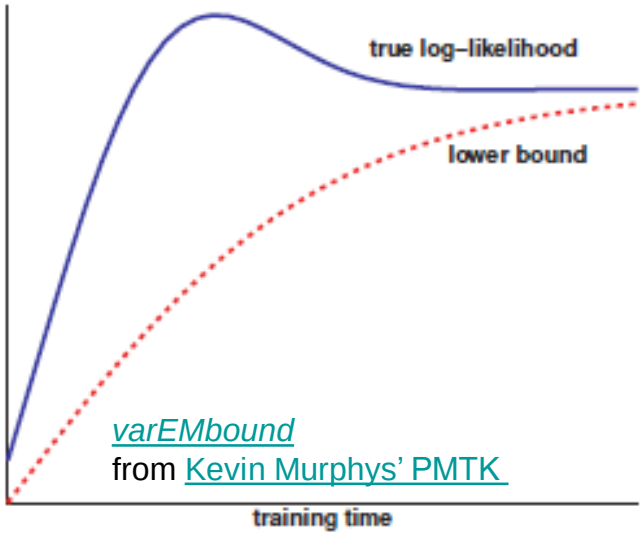
$$v_{t+1} = \arg\max_v \log p(\mathcal{D} \mid \boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}, v).$$

- Liu, C. and D. Rubin (1995). ML Estimation of the T distribution using EM and its extensions, ECM and ECME. *Statistica Sinica 5*, 19–39.
- McLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and Extensions*. Wiley.

Variational EM

- If we can ensure that the E step is performing inference based on a lower bound to the likelihood, then the M step can be seen as monotonically increasing this lower bound.
- This is called **variational EM.**
- On the top, the lower bound increases at each iteration and the same is true for the likelihood.
- On the bottom, the lower bound increases but the likelihood decreases. The algorithm is closing the gap between the approximate and true posterior (regularizing effect).





varEMbound
from Kevin Murphys' PMTK

- Neal, R. M. and G. E. Hinton (1998). A new view of the EM algorithm that justifies incremental and other variants. In M. Jordan (Ed.), *Learning in Graphical Models*. MIT Press..
- Saul, L., T. Jaakkola, and M. Jordan (1996). Mean Field Theory for Sigmoid Belief Networks. *J. of AI Research 4*, 61–76.

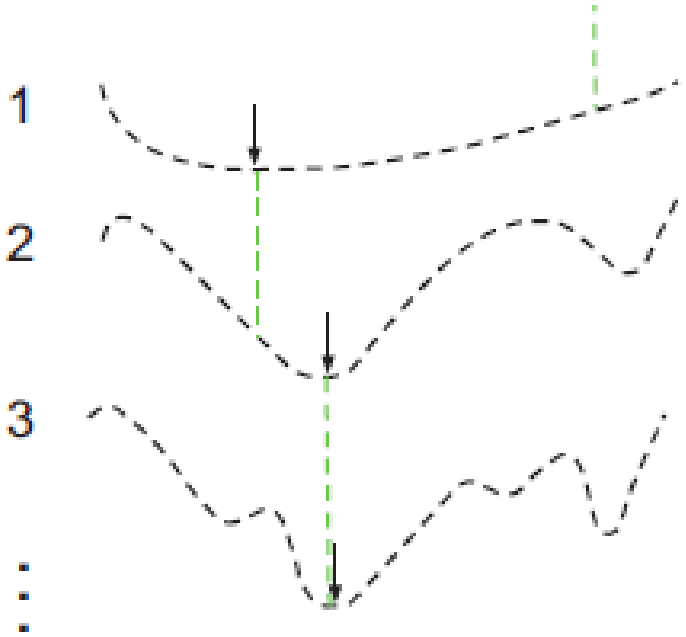
Monte Carlo EM (MCEM)

- In the E step, one can use MC approximation to the expected sufficient statistics.
- Draw $\mathbf{z}^s_i \sim p(\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta}^t)$ and then compute the sufficient statistics for each completed vector $(\mathbf{x}_i, \mathbf{z}^s_i)$ and then average the results.[1]
- If we only draw a single sample, it is called **stochastic EM.**[2]
- Use MCMC even though for convergence inside each E step, the method becomes very slow.
- *An alternative is to use stochastic approximation*: only perform brief sampling in the E step, followed by a partial parameter update. This is called **stochastic approximation EM** and tends to work better than MCEM. [3]
- *Another alternative* is to apply MCMC to infer the parameters as well as the latent variables (a fully Bayesian approach), thus *eliminating the distinction between E and M steps*.

1. Wei, G. and M. Tanner (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *J. of the Am. Stat. Assoc. 85*(411), 699–704.
2. Celeux, G. and J. Diebolt (1985). The SEM algorithm: A probabilistic teacher derive from the EM algorithm for the mixture problem. *Computational Statistics Quarterly 2*, 73–82.
3. Delyon, B., M. Lavielle, and E. Moulines (1999). Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics 27* (1), 94–128.

Annealing EM

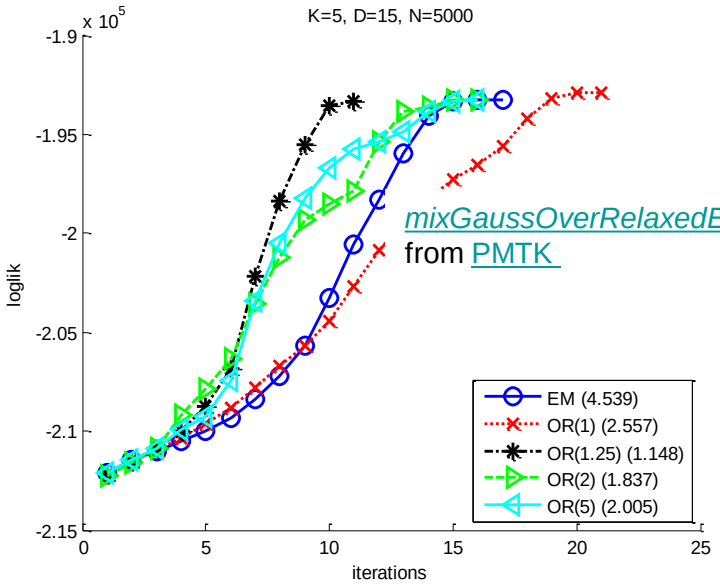
- In general, EM converges to a local maximum. To increase the chance of finding the global maximum, we can use a variety of methods.
- One approach is to use a method known as **deterministic annealing**.
- Smooth the posterior landscape by raising the temperature and then gradually start cooling while tracking the global maximum.



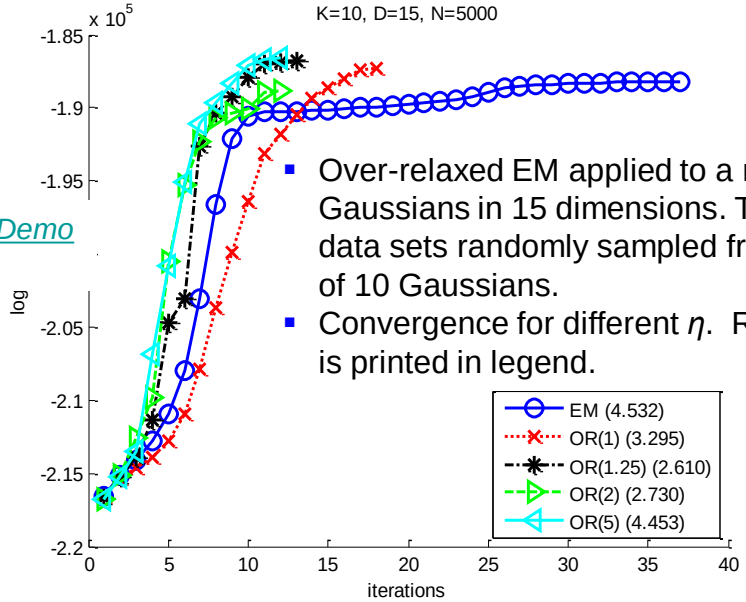
- McLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and Extensions*. Wiley.
- Rose, K. (1998, November). Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. IEEE 80*, 2210–2239
- Ueda, N. and R. Nakano (1998). Deterministic annealing EM algorithm. *Neural Networks 11*, 271–282.

Over-Relaxed EM

- EM is slow if there is lots of missing data. In the adaptive **over-relaxed EM algorithm** we update $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta(M(\boldsymbol{\theta}_t) - \boldsymbol{\theta}_t)$, where η is a step-size parameter and $M(\boldsymbol{\theta}_t)$ is the usual M-step update.
- This reduces to EM if $\eta = 1$, but using larger η results in faster convergence. But for large η the algorithm fails to converge.
- In closing, note that EM is a special case of **bound optimization** or **MM** algorithms (MM stands for **minorize-maximize**).



mixGaussOverRelaxedEmDemo from PMTK



- Over-relaxed EM applied to a mixture of 5 Gaussians in 15 dimensions. Two different data sets randomly sampled from a mixture of 10 Gaussians.
- Convergence for different η. Running time is printed in legend.

- Salakhutdinov, R. and S. Roweis (2003). Adaptive overrelaxed bound optimization methods. In *Proceedings of the International Conference on Machine Learning*, Volume 20, pp. 664–671.
- Hunter, D. R. and K. Lange (2004). A Tutorial on MM Algorithms. *The American Statistician 58*, 30–37.

Online EM

- When dealing with large or streaming datasets, it is important to be able to learn online.
- There are two main approaches to **online EM** in the literature.
- **Incremental EM**[1], optimizes the lower bound $\mathcal{L}(\boldsymbol{\theta}, q_1, \ldots, q_N)$ one q_i at a time. This requires storing the expected sufficient statistics for each data case.
- **Stepwise EM**[2-5], is based on stochastic approximation theory, and only requires constant memory use.

1. Neal, R. M. and G. E. Hinton (1998). A new view of the EM algorithm that justifies incremental and other variants. In M. Jordan (Ed.), *Learning in Graphical Models*. MIT Press.
2. Sato, M. and S. Ishii (2000). On-line EM algorithm for the normalized Gaussian network. *Neural Computation 12*, 407–432.
3. Cappe, O. and E. Mouline (2009, June). Online EM Algorithm for Latent Data Models. *J. of Royal Stat. Soc. Series B 71*(3), 593–613.
4. Cappe, O. (2010). Online Expectation Maximisation. In K. Mengersen, M. Titterington, and C. Robert (Eds.), *Mixtures*.
5. Liang, P. and D. Klein. Online EM for Unsupervised Models. In *Proc. NAACL Conference*

Batch EM Review

- Let $\phi(\mathbf{x}, \mathbf{z})$ be a vector of sufficient statistics for a single data case.
  - For example, for a mixture of multinoullis, this would be the count vector $a(j)$, which is the number of cluster j was used in $\mathbf{z}$, plus the matrix $B(j, v)$, which is of the number of times the hidden state was j and the observed letter was v.
- Let $\mathbf{s}_i$ be the expected sufficient statistics for case i, and $\boldsymbol{\mu} = \sum_{i=1}^{N} \boldsymbol{s}_i$ the sum of the ESS. Given $\boldsymbol{\mu}$, we can derive an MLE or MAP estimate of the parameters $\boldsymbol{\theta}(\boldsymbol{\mu})$ in the M step.
- For example, in the case of mixtures of Multinoullis, we just need to normalize $\mathbf{a}$ and each row of $\mathbf{B}$

```
1 initialize μ;
2 repeat
3     μ^new = 0 ;
4     for each example i = 1 : N do
5         s_i := Σ_z p(z|x_i, θ(μ)) φ(x_i, z) ;
6         μ^new := μ^new + s_i; ;
7     μ := μ^new;
8 until converged;
```

Incremental EM

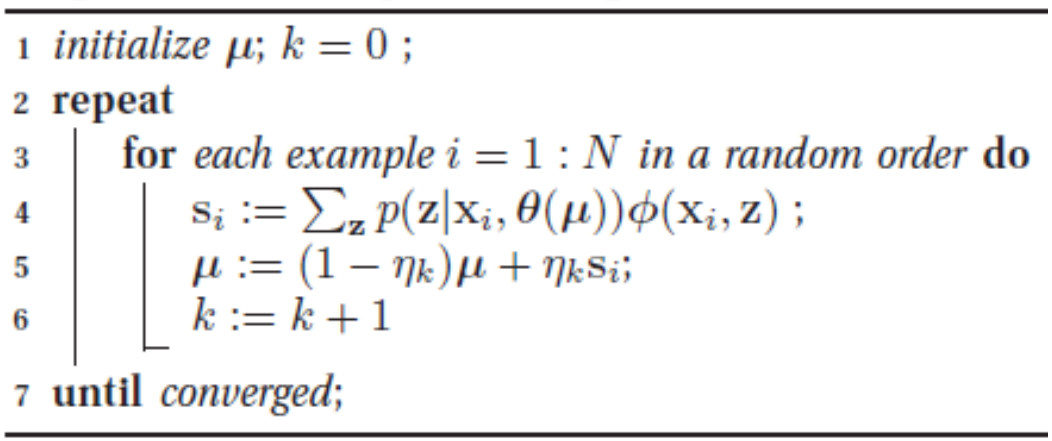
- *In incremental EM, we keep track of* $\boldsymbol{\mu}$ *as well as the* $\mathbf{s}_i$. When we come to a data case, we swap out the old $\mathbf{s}_i$ and replace it with the new $\mathbf{s}_i^{new}$
- Note that we can exploit the sparsity of $\mathbf{s}_i^{new}$ to speedup the computation of $\boldsymbol{\theta}$, since most components of $\boldsymbol{\mu}$ will not have changed.
- We maximize the lower bound $\mathcal{Q}(\boldsymbol{\theta}, q_1, \ldots, q_N)$ by *optimizing* q_1, *then* $\boldsymbol{\theta}$, *then* q_2, *then* $\boldsymbol{\theta}$, *etc.* Recall in the batch mode the update of $\boldsymbol{\theta}$ only occurs after all data were processed.
- This method is guaranteed to monotonically converge to a local maximum of the lower bound and of the log likelihood itself.

```
1 initialize s_i for i = 1 : N;
2 μ = Σ_i s_i;
3 repeat
4     for each example i = 1 : N in a random order do
5         s_i^new := Σ_z p(z|x_i, θ(μ)) φ(x_i, z) ;
6         μ := μ + s_i^new − s_i;
7         s_i := s_i^new;
8 until converged;
```

- Neal, R. M. and G. E. Hinton (1998). A new view of the EM algorithm that justifies incremental and other variants. In M. Jordan (Ed.), *Learning in Graphical Models*. MIT Press.

Stepwise EM Algorithm

- *Whenever we compute a new $\mathbf{s}_i$, we move $\boldsymbol{\mu}$ towards it.*
- At iteration k, the stepsize η_k must satisfy the Robbins-Monro conditions, e.g. $\eta_k = (2+k)^{-\kappa}$ for $0.5 < \kappa \leq 1$.
- Can get better speed up by using a minibatch of size m before each update. Optimize m and κ to maximize the training set likelihood. Try different values in parallel for an initial period.
- Liang and Klein compared batch, incremental, and stepwise EM on four different unsupervised language modeling tasks. They found that *stepwise EM ($\kappa \approx 0.7$, $m \approx 1000$) was faster than incremental EM, and both were much faster than batch EM.*
- *Stepwise EM was often better than batch EM; incremental EM was often worse than either of the other methods.*



- Liang, P. and D. Klein. Online EM for Unsupervised Models. In *Proc. NAACL Conference*

Incremental EM Algorithm

- Consider the case of N independent data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with corresponding latent variables $\mathbf{z}_1, \ldots, \mathbf{z}_N$.
- The joint distribution $p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$ factorizes over the data points, and this structure can be exploited in an incremental form of EM in which at each EM cycle *only one data point is processed at a time.*
- *In the E step*, instead of recomputing the responsibilities for all of the data points, *we just re-evaluate the responsibilities for one data point.*
- It might appear that the subsequent M step would require computation involving the responsibilities for all of the data points. However, if the mixture components are members of the exponential family, then *the responsibilities enter only through simple sufficient statistics, and these can be updated efficiently.*

Incremental EM Algorithm

- Consider e.g. for the case of a Gaussian mixture performing an update for data point m in which the old and new values of the responsibilities are denoted $\gamma^{old}(z_{mk})$ and $\gamma^{new}(z_{mk})$.
- In the M step, the required sufficient statistics can be updated incrementally. For instance, for the means the sufficient statistics are defined by

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\,\mathbf{x}_n, \quad N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

from which we obtain (see proof next)

$$\boldsymbol{\mu}_k^{new} = \boldsymbol{\mu}_k^{old} + \left(\frac{\gamma^{new}(z_{mk}) - \gamma^{old}(z_{mk})}{N_k^{new}}\right)\left(\mathbf{x}_m - \boldsymbol{\mu}_k^{old}\right)$$

$$N_k^{new} = N_k^{old} + \gamma^{new}(z_{mk}) - \gamma^{old}(z_{mk})$$

- The results for the mixing coefficients and covariances are:

$$\pi_k^{new} = \pi_k^{old} - \frac{\gamma^{old}(z_{mk})}{N} + \frac{\gamma^{new}(z_{mk})}{N}$$

$$\boldsymbol{\Sigma}_k^{new} = \boldsymbol{\Sigma}_k^{old} - \frac{\gamma^{old}(z_{mk})}{N_k^{new}}\left(\left(\mathbf{x}_m - \boldsymbol{\mu}_k^{old}\right)\left(\mathbf{x}_m - \boldsymbol{\mu}_k^{old}\right)^T - \boldsymbol{\Sigma}_k^{old}\right) + \frac{\gamma^{new}(z_{mk})}{N_k^{new}}\left(\left(\mathbf{x}_m - \boldsymbol{\mu}_k^{new}\right)\left(\mathbf{x}_m - \boldsymbol{\mu}_k^{new}\right)^T - \boldsymbol{\Sigma}_k^{old}\right)$$

Incremental EM Algorithm

- Start with $N_k^{old} = \sum_n \gamma^{old}(z_{nk})$ and obtain N_k^{new} by updating $\gamma^{new}(z_{mk})$ of the data point $\mathbf{x}_m$:

$$N_k^{new} = \sum_{n \neq m} \gamma^{old}(z_{nk}) + \gamma^{new}(z_{mk}) = N_k^{old} - \gamma^{old}(z_{mk}) + \gamma^{new}(z_{mk})$$

- Similarly start with $\boldsymbol{\mu}_k^{old} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma^{old}(z_{nk}) \boldsymbol{x}_n$ and obtain $\boldsymbol{\mu}_k^{new}$ by updating the responsibilities $\gamma^{new}(z_{mk})$ of the data point $\mathbf{x}_m$:

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k^{new}} \left(\sum_{n \neq m}^{N} \gamma^{old}(z_{nk}) \boldsymbol{x}_n + \gamma^{new}(z_{mk}) \boldsymbol{x}_m \right) = \frac{1}{N_k^{new}} \left(\sum_{n}^{N} \gamma^{old}(z_{nk}) \boldsymbol{x}_n - \gamma^{old}(z_{mk}) \boldsymbol{x}_m + \gamma^{new}(z_{mk}) \boldsymbol{x}_m \right)$$

$$= \frac{1}{N_k^{new}} \left(N_k^{old} \boldsymbol{\mu}_k^{old} - \gamma^{old}(z_{mk}) \boldsymbol{x}_m + \gamma^{new}(z_{mk}) \boldsymbol{x}_m \right)$$

$$= \frac{1}{N_k^{new}} \left(\left(N_k^{new} - \gamma^{new}(z_{mk}) + \gamma^{old}(z_{mk}) \right) \boldsymbol{\mu}_k^{old} - \gamma^{old}(z_{mk}) \boldsymbol{x}_m + \gamma^{new}(z_{mk}) \boldsymbol{x}_m \right)$$

$$= \boldsymbol{\mu}_k^{old} + \left(\frac{\gamma^{new}(z_{mk}) - \gamma^{old}(z_{mk})}{N_k^{new}} \right) \left(\boldsymbol{x}_m - \boldsymbol{\mu}_k^{old} \right)$$

Incremental EM Algorithm

- *Thus both the E step and the M step take fixed time that is independent of the total number of data points.*

- Because the parameters are revised after each data point, rather than waiting until after the whole data set is processed, *this incremental version can converge faster than the batch version.*

- Each E or M step in this incremental algorithm is increasing the value of $\mathcal{L}(q, \boldsymbol{\theta})$ and, as we have shown above, if the algorithm converges to a local (or global) maximum of $\mathcal{L}(q, \boldsymbol{\theta})$, this will correspond to a local (or global) maximum of the log likelihood function ln p($\mathbf{X}|\boldsymbol{\theta}$).

EM For Missing Data Problems

Fitting Models with Missing Data

- We want to fit a joint density model by MLE but we have holes in our data matrix due to missing data (NaNs). Let $O_{ij} = 1$ if component j of data case i is observed, and let $O_{ij} = 0$ otherwise. Let $\mathbf{X}_v$ be the visible data, and $\mathbf{X}_h$ be the missing (hidden) data:

$$\boldsymbol{X}_v = \left\{ x_{ij} : O_{ij} = 1 \right\}, \boldsymbol{X}_h = \left\{ x_{ij} : O_{ij} = 0 \right\}$$

- Our goal is to compute

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} p(\boldsymbol{X}_v | \boldsymbol{\theta}, \boldsymbol{O})$$

- Under the *missing at random assumption*, we have

$$p(\boldsymbol{X}_v | \boldsymbol{\theta}, \boldsymbol{O}) = \prod_{i=1}^{N} p(\boldsymbol{x}_{iv} | \boldsymbol{\theta})$$

  Here $\mathbf{x}_{iv}$ is a vector created from row i and the columns $\{j : O_{ij} = 1\}$.

- Hence the log-likelihood has the form

$$\log p(\boldsymbol{X}_v | \boldsymbol{\theta}, \boldsymbol{O}) = \sum_i \log p(\boldsymbol{x}_{iv} | \boldsymbol{\theta}), \text{ where } : p(\boldsymbol{x}_{iv} | \boldsymbol{\theta}) = \sum_{\boldsymbol{x}_{ih}} p(\boldsymbol{x}_{iv}, \boldsymbol{x}_{ih} | \boldsymbol{\theta})$$

- We finally obtain our familiar form of LVMs:

$$\log p(\boldsymbol{X}_v | \boldsymbol{\theta}, \boldsymbol{O}) = \sum_i \log \sum_{\boldsymbol{x}_{ih}} p(\boldsymbol{x}_{iv}, \boldsymbol{x}_{ih} | \boldsymbol{\theta})$$

Fitting Models with Missing Data

- We want to fit an MVN by MLE based on those rows of the data matrix that are fully observed. If there are no such rows, we can use some ad-hoc imputation procedures, and then compute an initial MLE.
- **E step:** Once we have $\boldsymbol{\theta}^{t-1}$, we can compute the expected complete data log likelihood at iteration t as follows:

$$Q\left(\theta,\theta^{t-1}\right)=\mathbb{E}\left[\sum_{i=1}^{N}\log\mathcal{N}\left(\boldsymbol{x}_i\mid\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\mid\mathcal{D},\theta^{t-1}\right]=$$

$$=-\frac{N}{2}\log\left|2\pi\boldsymbol{\Sigma}\right|-\frac{1}{2}\sum_i\mathbb{E}\left[\left(\boldsymbol{x}_i-\boldsymbol{\mu}\right)^T\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}_i-\boldsymbol{\mu}\right)\right]$$

$$=-\frac{N}{2}\log\left|\boldsymbol{\Sigma}\right|-\frac{ND}{2}\log\left(2\pi\right)-\frac{1}{2}tr\left(\boldsymbol{\Sigma}^{-1}\mathbb{E}\left[S(\boldsymbol{\mu})\right]\right)$$

where:

$$\mathbb{E}\left[S(\boldsymbol{\mu})\right]=\sum_i\mathbb{E}\left[\left(\boldsymbol{x}_i-\boldsymbol{\mu}\right)\left(\boldsymbol{x}_i-\boldsymbol{\mu}\right)^T\right]=\sum_i\left(\mathbb{E}\left[\boldsymbol{x}_i\boldsymbol{x}_i^T\right]+\boldsymbol{\mu}\boldsymbol{\mu}^T-2\boldsymbol{\mu}\mathbb{E}\left[\boldsymbol{x}_i\right]^T\right)$$

- To simplify the notation, we drop the conditioning of the expectation on $\mathcal{D}$ and $\boldsymbol{\theta}^{t-1}$. We need to compute to expected sufficient statistics.
- We use the results for the conditionals of a MVN from an earlier lecture.

$$\boldsymbol{x}_{ih}\mid\boldsymbol{x}_{iv},\theta\sim\mathcal{N}\left(\boldsymbol{m}_i,\boldsymbol{V}_i\right)$$

$$\boldsymbol{m}_i=\boldsymbol{\mu}_h+\boldsymbol{\Sigma}_{hv}\boldsymbol{\Sigma}_{vv}^{-1}\left(\boldsymbol{x}_{iv}-\boldsymbol{\mu}_v\right),\boldsymbol{V}_i=\boldsymbol{\Sigma}_{hh}-\boldsymbol{\Sigma}_{hv}\boldsymbol{\Sigma}_{vv}^{-1}\boldsymbol{\Sigma}_{vh}$$

Fitting Models with Missing Data

- Hence the expected sufficient statistics are

$$\mathbb{E}[\boldsymbol{x}_i] = (\mathbb{E}[\boldsymbol{x}_{ih}]; \boldsymbol{x}_{iv}) = (\boldsymbol{m}_i; \boldsymbol{x}_{iv}), \mathbb{E}[\boldsymbol{x}_i \boldsymbol{x}_i^T] = \mathbb{E}\left[\begin{pmatrix} \boldsymbol{x}_{ih} \\ \boldsymbol{x}_{iv} \end{pmatrix} (\boldsymbol{x}_{ih} \quad \boldsymbol{x}_{iv})\right] = \begin{pmatrix} \mathbb{E}[\boldsymbol{x}_{ih}\boldsymbol{x}_{ih}^T] & \mathbb{E}[\boldsymbol{x}_{ih}]\boldsymbol{x}_{iv}^T \\ \boldsymbol{x}_{iv}\mathbb{E}[\boldsymbol{x}_{ih}]^T & \boldsymbol{x}_{iv}\boldsymbol{x}_{iv}^T \end{pmatrix}$$

$$\mathbb{E}[\boldsymbol{x}_{ih}\boldsymbol{x}_{ih}^T] = \mathbb{E}[\boldsymbol{x}_{ih}]\mathbb{E}[\boldsymbol{x}_{ih}]^T + \boldsymbol{V}_i = \boldsymbol{m}_i\boldsymbol{m}_i^T + \boldsymbol{V}_i$$

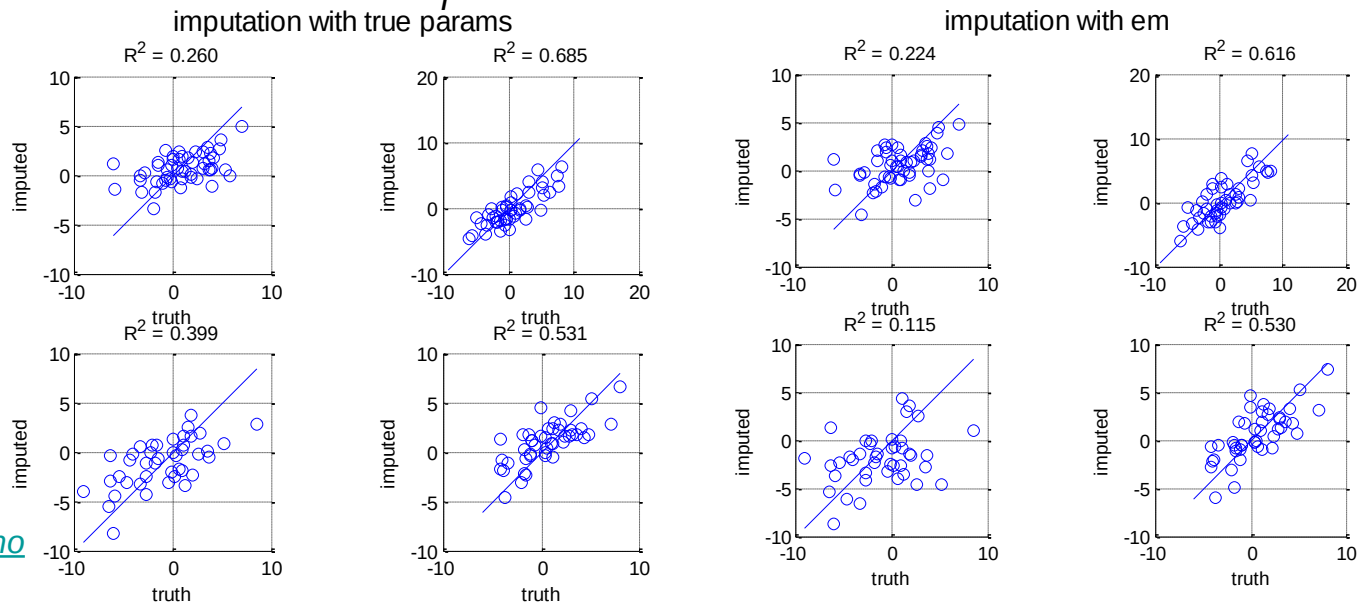
- To simplify the notation we assume that the unobserved variables come before the observed variables in the node ordering.
- M-Step: By solving $\nabla Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) = 0$, we can show that *the M step is equivalent to plugging the ESS into the MLE equations:*

$$\boldsymbol{\mu}^t = \frac{1}{N}\sum_i \mathbb{E}[\boldsymbol{x}_i], \boldsymbol{\Sigma}^t = \frac{1}{N}\sum_i \mathbb{E}[\boldsymbol{x}_i \boldsymbol{x}_i^T] - \boldsymbol{\mu}^t (\boldsymbol{\mu}^t)^T$$

- EM is *not* equivalent to simply replacing variables by their expectations and plugging into the standard MLE formula; that ignores the posterior variance and results in incorrect estimates. *Instead we must compute the expectation of the sufficient statistics and plug that into the usual equation for the MLE.*
- We can now easily modify the algorithm to perform MAP estimation.

Fitting Models with Missing Data

- Consider the imputation problem with N = 100 10-dim data cases, with 50% missing data. We fit the parameters using EM. Call the resulting parameters $\hat{\theta}$. *We make predictions as* $\mathbb{E}\left[\boldsymbol{x}_{ih} | \boldsymbol{x}_{iv}, \hat{\theta}\right]$
- The results obtained using the learned parameters are as good as with the true parameters. Performance improves with more data, or with less missing data.
- One can also fit a mixture of Gaussians in the presence of partially observed data vectors $\mathbf{x}_i$.



gaussImputationDemo from PMTK

Model Selection

Model Selection for Latent Variable Models

- In LVMs we specify the number of latent variables (model complexity). Choosing these parameters is a model selection.
- The optimal Bayesian approach is to *pick the model with the largest marginal likelihood,* $K^* = \text{argmax}_k\, p(\mathcal{D} \mid K)$.
- Evaluating the marginal likelihood for LVMs is quite difficult and simple approximations, such as BIC, can be used.
- Alternatively, we can use the cross-validated likelihood as a performance measure, although this can be slow, since it requires fitting each model F times (F being the # of CV folds).
- We need to search over a potentially large number of models. One can perform exhaustive search over all values of K. Often, we can set the model to its maximal size and using the Occam's razor principle kill unwanted components.

- Fraley, C. and A. Raftery (2002). Model-based clustering, discriminant analysis, and density estimation. *J. of the Am. Stat. Assoc.* (97), 611–631.

Model Selection for Latent Variable Models

- One can also perform stochastic sampling in the space of models, e.g. use reversible jump MCMC (introduce birth/death moves to propose new centers or kill existent ones).
- A simpler approach is to use a Dirichlet process mixture model, which can be fit using Gibbs sampling, but allows for an unbounded number of mixture components.
- These sampling methods are faster than evaluating the model for each *K* separately. Fitting the model for each *K* is often slow. By contrast, sampling methods often quickly determine that a certain value of *K* is poor, and thus they do not waste time in that part of the posterior.

- Green, P. (1998). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*, 711–732.
- Green, P. (2003). Tutorial on transdimensional MCMC. In P. Green, N. Hjort, and S. Richardson (Eds.), *Highly Structured Stochastic Systems*. OUP.
- Lunn, D., N. Best, and J. Whittaker (2009). Generic reversible jump MCMC using graphical models. *Statistics and Computing 19*(4), 395–408.

Model Selection for Non-Probabilistic Methods

- How do we choose *K* for the *K-means* algorithm? There is no probability model or likelihood for K-means, so none of the methods discussed can be used.
- Use the reconstruction error. Define the squared reconstruction error of a data set $\mathcal{D}$, using model complexity *K*, as follows:

$$E(\mathcal{D}, K) = \frac{1}{|\mathcal{D}|} \sum_{i \in D} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2$$

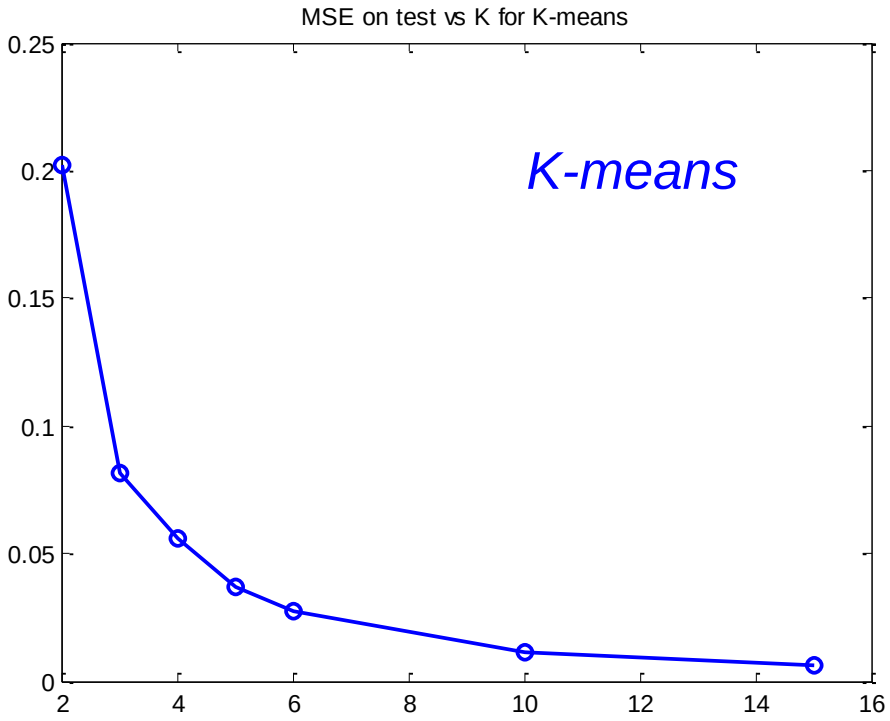
- In the case of K-means,

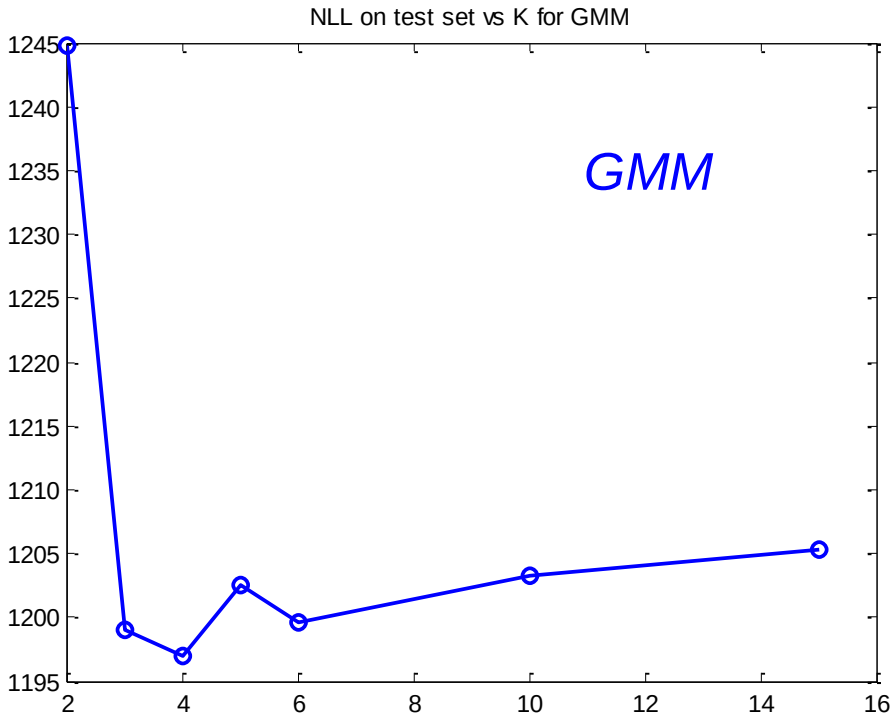
$$\hat{\boldsymbol{x}}_i = \boldsymbol{\mu}_{z_i},\ z_i = \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_2^2$$

- We plot next the reconstruction error on the *test set* for K-means.
- *We notice that the MSE error decreases with increasing model complexity contrary to what we have discussed before for model selection!*

Model Selection for Non-Probabilistic Methods

- Test set performance vs *K* for data generated from a mixture of 3 Gaussians in 1d
  - ✓ (a) MSE on test set *for K-means*.
  - ✓ (b) Negative log likelihood on test set *for GMM*.

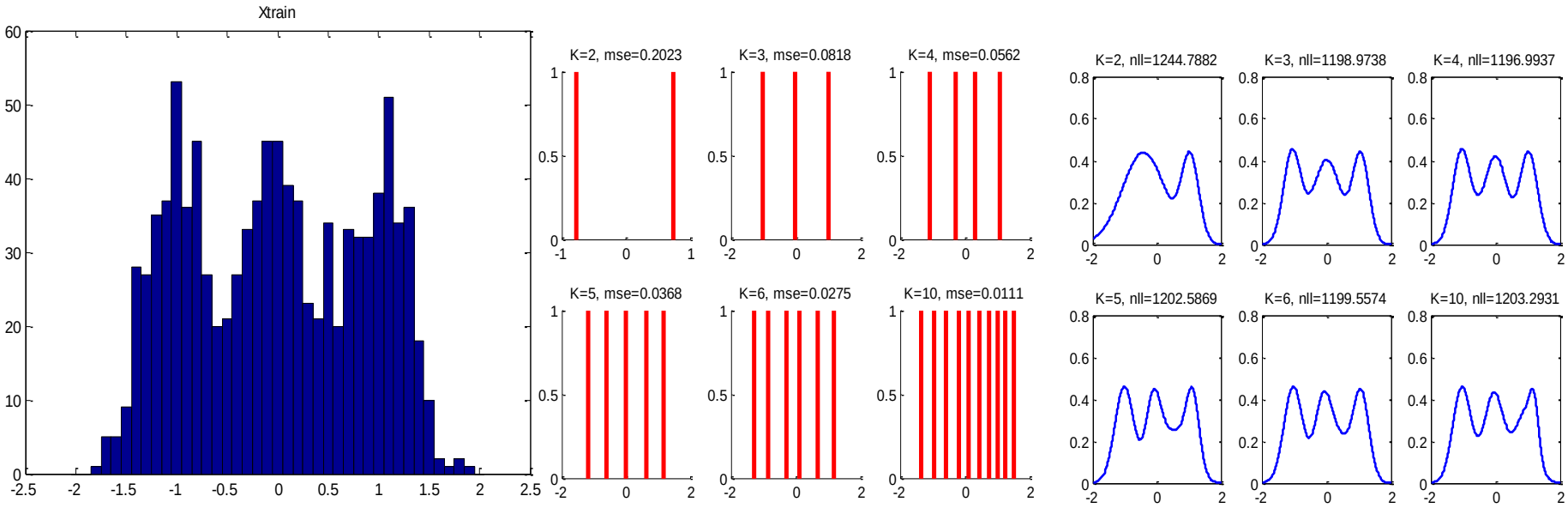




kmeansModelSel1d
from PMTK

Model Selection for Non-Probabilistic Methods

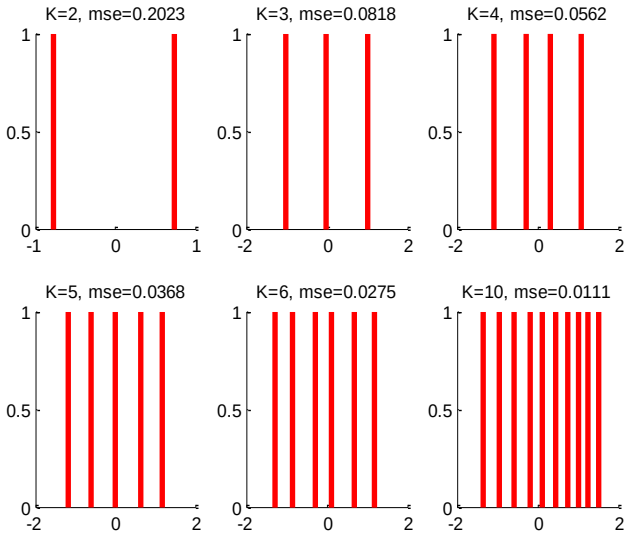
- Synthetic data generated from a mixture of 3 Gaussians in 1d.
  - ✓ (a) Histogram of training data (Test data looks essentially the same)
  - ✓ (b) Centroids estimated by K-means for $K \in \{2, 3, 4, 5, 6, 10\}$.
  - ✓ (c) GMM density model estimated by EM for various K.

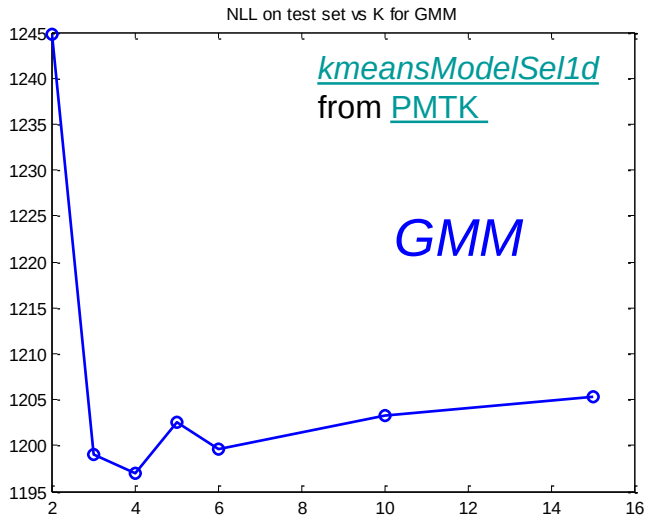


kmeansModelSel1d from PMTK

Model Selection for Non-Probabilistic Methods

- When we add more and more centroids to *K*-means, we densely cover the space.
- K-Means: Hence any given test vector is more likely to find a close prototype to accurately represent it as *K* increases, thus decreasing reconstruction error.
- *In supervised learning, we can always use cross validation to select between non-probabilistic models of different complexity, but this is not the case with unsupervised learning.*
- In probabilistic models such as the GMM: the negative log-likelihood has the usual U-shaped curve on the test set.





- Tibshirani, R., G. Walther, and T. Hastie (2001). Estimating the number of clusters in a dataset via the gap statistic. *J. of Royal Stat. Soc. Series B 32*(2), 411–423.

Model Selection for Non-Probabilistic Methods

- Given that cross validation doesn't work for unsupervised models how *can one choose K when using a non probabilistic model?*

- We can ***plot the reconstruction error on the training set versus K***, and ***identify a kink K^* in the curve***.

  - For $K < K^*$ (K^* being the unknown true number of clusters), the rate of decrease in the error function will be high, since we are splitting apart things that should not be grouped together.

  - However, for $K > K^*$, we are splitting apart "natural" clusters, which does not reduce the error by as much.

- This kink-finding process can be automated by use of the gap statistic. Identifying such kinks can be hard, since the loss function usually drops off gradually.

  - Tibshirani, R., G. Walther, and T. Hastie (2001). Estimating the number of clusters in a dataset via the gap statistic. *J. of Royal Stat. Soc. Series B 32*(2), 411–423.

Summary: The EM Algorithm

- (local) maxima of $\mathcal{L}(q,\theta)$ correspond to those of ln $p(\mathbf{X}|\mathbf{q})$.
- EM converges to (local) maximum of likelihood
  - Coordinate ascent on $\mathcal{L}(q,\theta)$ and $\mathcal{L} = \ln p(\mathbf{X}|\theta)$ after E-step
- Alternative schemes to optimize the bound
  - Generalized EM: relax M-step from maximizing to increasing $\mathcal{L}$
  - Expectation Conditional Maximization: M-step maximizes w.r.t. groups of parameters in turn
  - Incremental EM: E-step per data point, incremental M-step
  - Variational EM: relax E-step from maximizing to increasing $\mathcal{L}$
    - no longer $\mathcal{L} = \ln p(\mathbf{X}|\theta)$ after E-step
- Same applies for MAP estimation $p(\theta|\mathbf{X}) = p(\theta)p(\mathbf{X}|\theta)/p(\mathbf{X})$
  - Bound second term: $\ln p(\theta|\mathbf{X}) \geq \ln p(\theta) + \mathcal{L}(q,\theta) - \ln p(\mathbf{X})$