Mixture Models and the EM Algorithm

15-486/782: Artificial Neural Networks
David S. Touretzky

Fall 2006

Density Estimation

Parametric approaches require us to assume a particular distribution function, e.g., gaussian.

The real data may not fit any one simple distribution.

Non-parametric approaches use the data itself to estimate density, e.g., by histogramming (Parzen windows).

Expensive; requires us to store all the data points.

Mixture models offer a compromise approach:

Model the data as a mixture of several parameterized distributions.

A Simple Mixture Model

We can model a dataset as a collection of points generated by a mixture of Gaussians.

μ_j *mean*

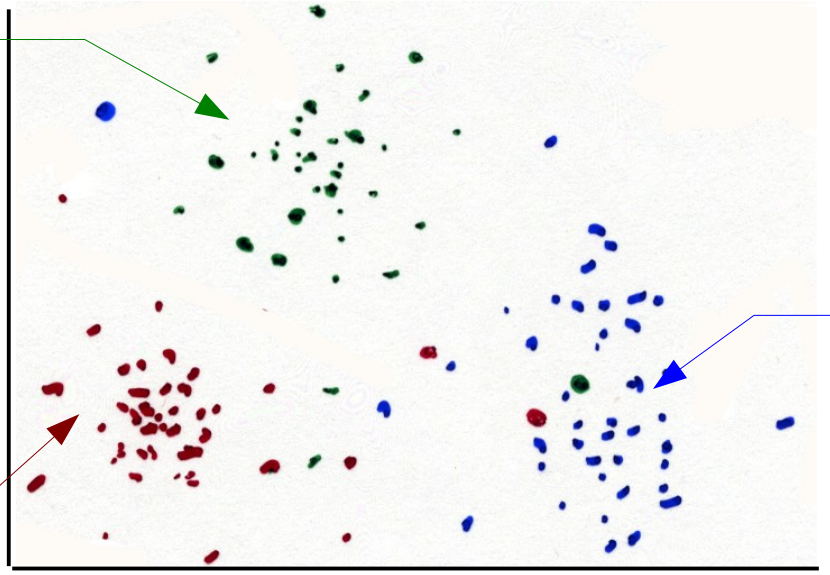
σ_j *standard deviation*

α_j *weight/prevalence/prior probability*

$$\left(\sum_j \alpha_j\right) = 1, \qquad so\ \alpha_j = P(c_j)$$

Gaussian Generators Form a Mixture

Generator for c_2



Generator for c_3

Generator for c_1

Class priors $P_j = P(c_j)$ *(mixture coefficients)*

$$\sum_j P_j = 1$$

$$\textit{Density distribution} = \sum_j P_j \cdot \exp\left[\frac{-(\mathbf{x}-\boldsymbol{\mu}_j)^2}{\sigma_j^2}\right]$$

Probability Densities

$P(j)$ = *prior probability of class* c_j

$$so \quad \sum_j P(j) = 1$$

Probability density of the mixture:

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j)$$

Posterior probability:

$$P(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{p(\mathbf{x})}$$

$$so \quad \sum_j P(j|\mathbf{x}) = 1$$

Conditional Density

Assume covariance matrix is diagonal with equal elements. Then:

$$p(\mathbf{x}|j) = \underbrace{\frac{1}{(2\pi\sigma_j^2)^{d/2}}}_{\text{normalization term}} \cdot \exp\left\{\frac{-\|\mathbf{x}-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right\}$$

How can we determine the "most probable" values of μ_j and σ_j and P(j), given the dataset $\{\mathbf{x}_i\}$?

Likelihood of a Dataset

What is the likelihood $\mathcal{L}$ that a dataset $\{\mathbf{x}_i\}$ was generated by a given mixture model?

$$p(\mathbf{x}_i) = \sum_{j=1}^{M} p(\mathbf{x}_i|j) \cdot P(j)$$

$$p(\{\mathbf{x}_i\}) = \prod_{i=1}^{n} p(\mathbf{x}_i) = \mathcal{L}$$

Log Likelihood

For gradient descent, we want a sum, not a product, because the derivative of a product is messy. So take the negative log.

$$E = -\log \mathcal{L} = -\sum_{i=1}^{n} \log p(\boldsymbol{x_i})$$

$$= -\sum_{i=1}^{n} \log \left\{ \sum_{j=1}^{M} p(\boldsymbol{x_i}|j) P(j) \right\}$$

Gradient Descent on E

$$E = -\sum_{i=1}^{n} \log p(\mathbf{x_i}) = -\sum_{i=1}^{n} \log \left(\sum_{j=1}^{M} p(\mathbf{x_i}|j) P(j) \right)$$

$$\frac{\partial E}{\partial \mu_j} = -\sum_{i=1}^{n} \left(\frac{1}{p(\mathbf{x_i})} \cdot \sum_{k=1}^{M} P(k) \; \frac{\partial}{\partial \boldsymbol{\mu_j}} p(\mathbf{x_i}|k) \right)$$

$$= -\sum_{i=1}^{n} \left(\frac{1}{p(\mathbf{x_i})} \cdot p(\mathbf{x_i}|j) P(j) \cdot \frac{\|\mathbf{x_i} - \boldsymbol{\mu_j}\|}{\sigma} \right)$$

$$= -\sum_{i=1}^{n} P(j|\mathbf{x_i}) \cdot \frac{\|\mathbf{x_i} - \boldsymbol{\mu_j}\|}{\sigma}$$

Gradient Descent (cont.)

$E = -\log \mathcal{L}$ *is our error function.*

Do gradient descent on E:

$$\frac{\partial E}{\partial \mu_j} = -\sum_{i=1}^{n} P(j|\boldsymbol{x_i}) \cdot \frac{\|\boldsymbol{x_i} - \boldsymbol{\mu_j}\|}{\sigma_j^2}$$

$$\frac{\partial E}{\partial \sigma_j} = \sum_{i=1}^{n} \left(P(j|\boldsymbol{x_i}) \cdot \left\{ \frac{d}{\sigma_j} - \frac{\|\boldsymbol{x_i} - \boldsymbol{\mu_j}\|^2}{\sigma_j^3} \right\} \right)$$

Constrained Gradient Descent

There's a problem:

We can't just nudge μ_i and σ_j around by small amounts, since their values affect P(j).

Must satisfy these constraints:

$$\sum_{j=1}^{M} P(j) = 1$$

$$0 \leqslant P(j) \leqslant 1$$

The EM Algorithm (Expectation-Maximization)

Iterative algorithm for optimizing μ_j *and* σ_j, *and* P_j *to minimize* E.

Definitions:

$$P_{ji} = P(j|\boldsymbol{x_i}) = \frac{P(\boldsymbol{x_i}|j) \cdot P(j)}{p(\boldsymbol{x_i})}$$

Class priors:

$$P(j) = P_j = \frac{1}{N}\sum_{i=1}^{N} P_{ji}$$

$$p(\boldsymbol{x}) = \sum_{j=1}^{M} p(\boldsymbol{x}|j)\cdot P(j)$$

Expectation (E) Step

Calculate P_{ji} *for all points* i *and gaussians* j*,*

using current parameter values $\boldsymbol{\mu_j}$*,* σ_j^2*, and* P_j*.*

Maximization (M) Step

$$P_j \leftarrow \frac{1}{N}\sum_{i=1}^{N} P_{ji}$$

$$\boldsymbol{\mu_j} \leftarrow \frac{\sum_i P_{ji} \cdot \boldsymbol{x_i}}{\sum_i P_{ji}} = \frac{\sum_i P_{ji} \cdot \boldsymbol{x_i}}{P_j}$$

$$\sigma_j^2 \leftarrow \frac{\frac{1}{d}\sum_i P_{ji} \cdot \|\boldsymbol{x_i} - \boldsymbol{\mu_j}\|^2}{P_j}$$

Repeat E and M steps until convergence.

EM Finds a Local Minimum in E (or Local Maximum in $\mathcal{L}$)

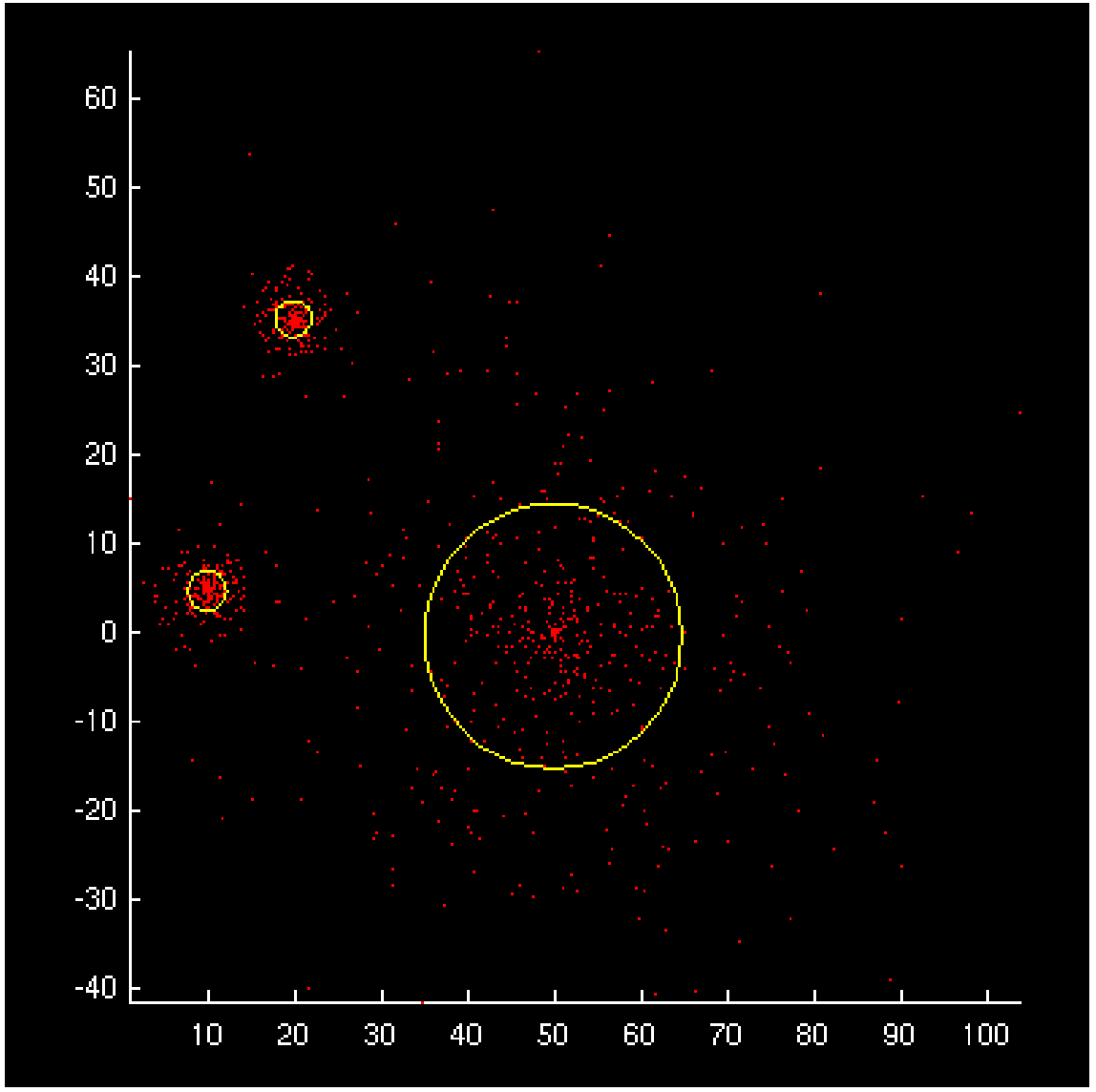
$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = 0 = -\sum_{i=1}^{n} P(j|\boldsymbol{x}_i) \cdot \frac{\|\boldsymbol{x}_i - \mu_j\|}{\sigma_j^2}$$

$$\boldsymbol{\mu}_j \sum_{i=1}^{n} \frac{P(j|\boldsymbol{x}_i)}{\sigma_j^2} = \frac{\sum_{i=1}^{n} P(j|\boldsymbol{x}_i)\boldsymbol{x}_i}{\sigma_j^2}$$

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{n} P(j|\boldsymbol{x}_i)\boldsymbol{x}_i}{\sum_{i=1}^{n} P(j|\boldsymbol{x}_i)}$$

μ_j *is the weighted mean of the* $\boldsymbol{x}_i$*'s credited to the jth mixture component.*

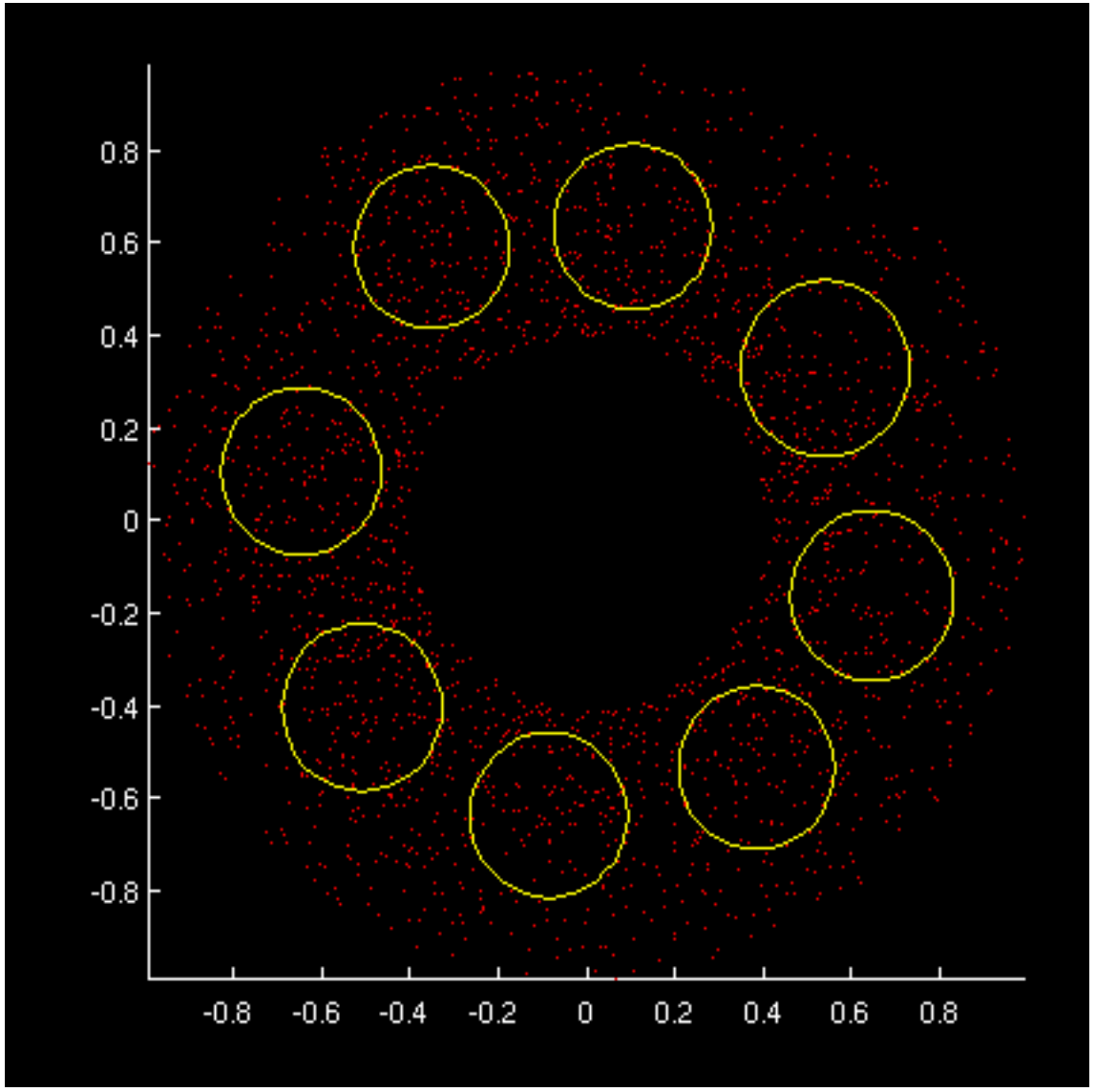
EM Demo



EmDataSet =4

emdemo

EM Demo



EmDataSet = 3

emdemo

Cheapo Heuristic

Not as good as split/merge (defined later), but easy to program:

If a component captures fewer than 1/(2M) points, reset its μ to a random $\mathbf{x}_i$ and recalculate its σ^2.

Assumes the P(j) values are roughly equal.

EmHeuristic = 1

emdemo

Characteristics of EM

Learns in a small number of iterations.

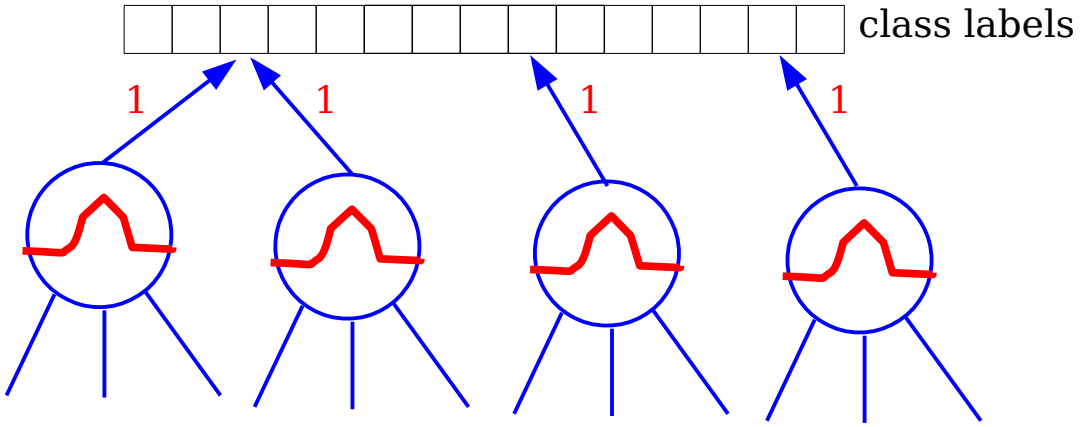
Can get stuck in local minima.

- But you can add heuristics to help unstick the algorithm.

Must decide in advance how many Gaussians.

Williamson: Gaussian ARTMAP

1. Use an RBF network to do pattern classification:



Each unit votes for one class. Tally votes from all active units. The class with the most votes wins.
No LMS training.

2. Use a variant of EM to train the gaussians.

“Match Tracking” in ARTMAP

Establish a match threshold ρ.

Units count as “active” only if $P(\boldsymbol{x}_i|j) > \rho$.

All other units are reset to zero; they do not vote.

If the network guesses the wrong class, increase ρ slightly and try again.

If ρ gets too high and all units are reset, then add a new unit to handle this data point.

Performance of Gaussian ARTMAP

Note: EM is a batch (offline) learning algorithm. Guassian ARTMAP uses an online variant.

Did well on several tasks:

- Letter image classification
- Landsat satellite image segmentation
- Speaker-independent vowel recognition

Match tracking helps Gaussian ARTMAP outperform EM by “backpropagating” the effects of erroneous classifications.

Offline Calculation of μ and σ^2

$$\mu = \langle x \rangle$$

$$\begin{aligned}
\sigma^2 &= \langle (x-\mu)^2 \rangle \\
&= \langle x^2 - 2x\mu + \mu^2 \rangle \\
&= \langle x^2 \rangle - 2\langle x \rangle \mu + \langle \mu^2 \rangle \\
&= \langle x^2 \rangle - 2\mu^2 + \mu^2 \\
&= \langle x^2 \rangle - \langle x \rangle^2
\end{aligned}$$

On-line Calculation of μ

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\mu_{n+1} = \frac{n \cdot \mu_n + x_{n+1}}{n+1}$$

$$= \frac{n}{n+1}\mu_n + \frac{x_n}{n+1}$$

$$= \left(1 - \frac{1}{n+1}\right)\mu_n + \frac{1}{n+1}x_{n+1}$$

On-line Calculation of σ^2

$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_i-\mu_n\right)^2$$

$$\sigma_{n+1}^2 = \left(1-\frac{1}{n+1}\right)\sigma_n^2 + \frac{1}{n+1}\left(x_{n+1}-\mu_{n+1}\right)^2$$

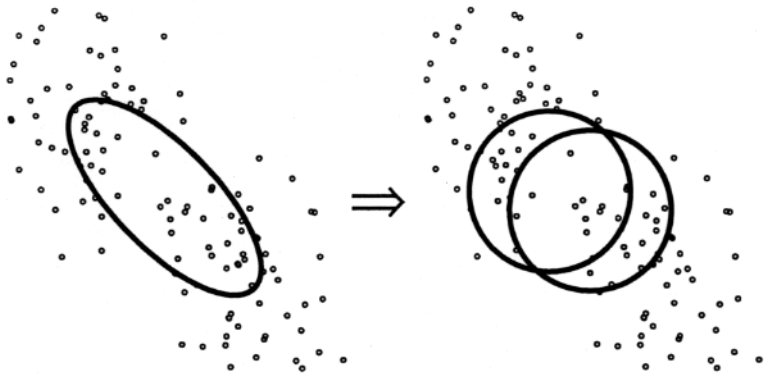
σ_n^2 is slightly biased because μ_n changes, but the effect is not significant.

Split and Merge EM
(Ueda, Nakano, Gharamani, and Hinton)

Split a component if it does a poor job estimating the local density.

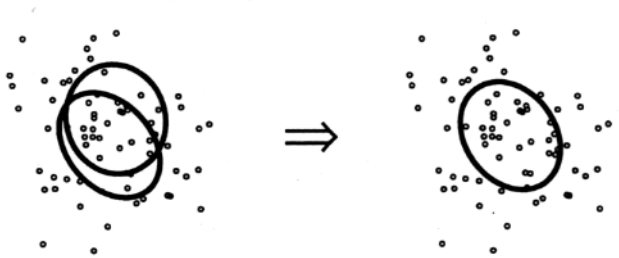
Example: a component stuck between two clusters will have low density near its mean and high density near the true cluster centers.

To split, make two copies, and perturb each one away from the mean by a small amount.



Split and Merge EM (cont.)

Merge two components if their parameters are close. Set the merged component's parameters to the weighted average.

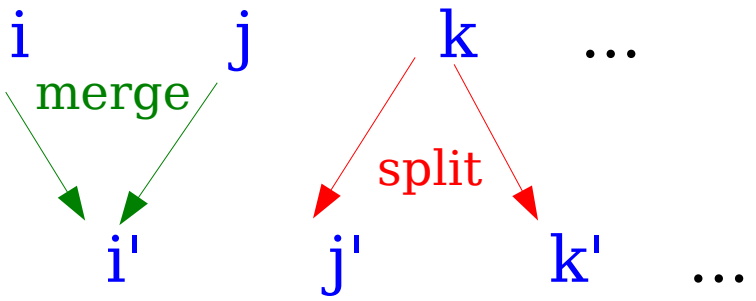


Combine one merge step with one split step, so the number of mixture components M stays the same.

Combined Split and Merge Steps

Old components:

New components:



Run a mini-EM step to adapt elements i', j', and k'. Then run full EM to asymptote.

If overall likelihood is not improved, undo the split/merge and try a different set of candidates.

Candidates are ranked heuristically; only need to look at about 5.

When To Merge

Define $P_i(\Theta^*) = \langle P(i|x_1;\Theta^*),\ldots,P(i|x_N;\Theta^*)\rangle^T$

Merge criterion:

$$J_{merge}(i,j;\Theta^*) = \frac{P_i(\Theta^*)^T P_j(\Theta^*)}{\|P_i(\Theta^*)\| \ \|P_j(\Theta^*)\|}$$

J_{merge} will be high if the vectors $P_i(\Theta^*)$ and $P_j(\Theta^*)$ point in roughly the same direction, meaning their gaussians capture roughly the same set of points.

Choose candidate pairs (i,j) to give the highest value of $J_{merge}(i,j;\Theta^*)$

When To Split

Estimate the local density of kth model around x:

$$f_k(x;\Theta^*) = \frac{\sum_{n=1}^{N} \delta(x-x_n) P(k|x_n;\Theta^*)}{\sum_{n=1}^{N} P(k|x_n;\Theta^*)}$$

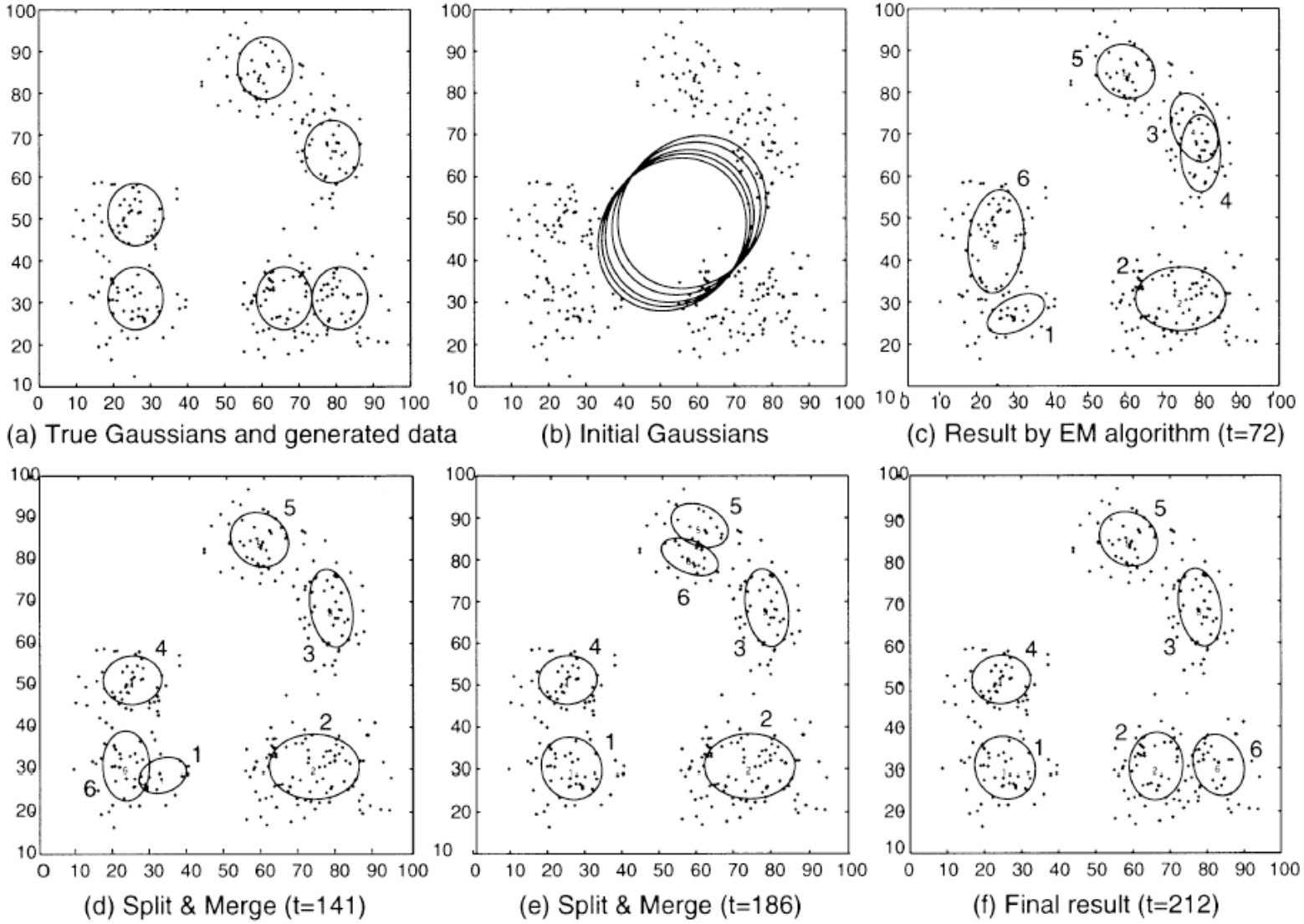
where δ is the empirical density distribution.

Check Kullback-Leibler divergence of local data density estimate f_k with density p_k of kth model as specified by current parameter estimate Θ^*:

$$J_{split}(k;\Theta^*) = \int f_k(x;\Theta^*) \log \frac{f_k(x;\Theta^*)}{p_k(x;\Theta^*)} dx$$

A large value of J_{split} suggests that model k has a poor estimate of local density, so splitting it might be helpful.

Performance of SMEM



Factor Analyzers

- More complex approach than EM.

- Able to handle high-dimensional data with low-dimensional embedded structure.

Mixture of Factor Analyzers

Project high-D space down to lower-D space. Compute a mixture of low-D functions.

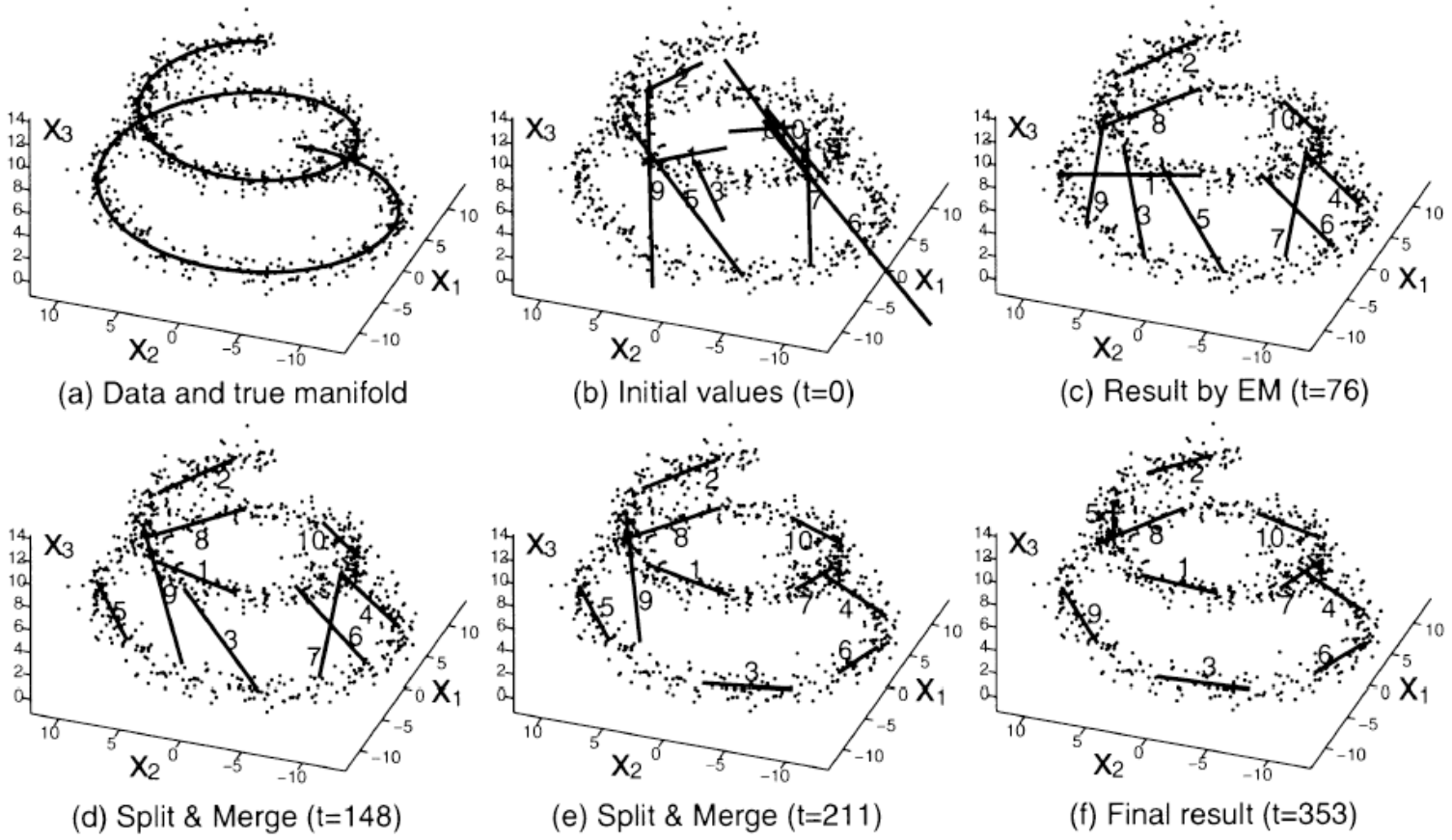
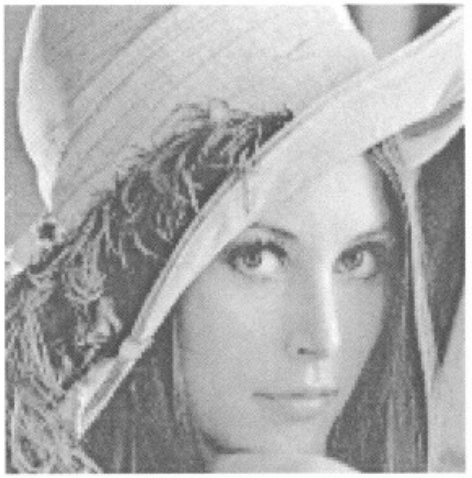
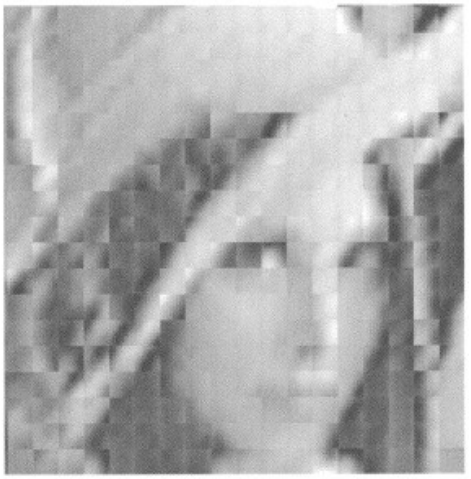


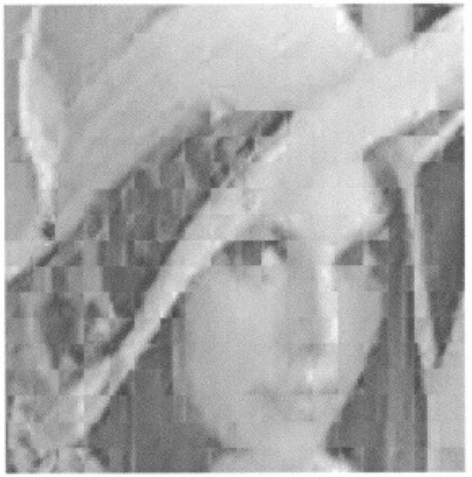
Image Reconstruction



(a) Original image



(b) PCA



(c) MFA with EM

(d) MFA with SMEM