Mixture Models and the EM Algorithm

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

Fall 2006

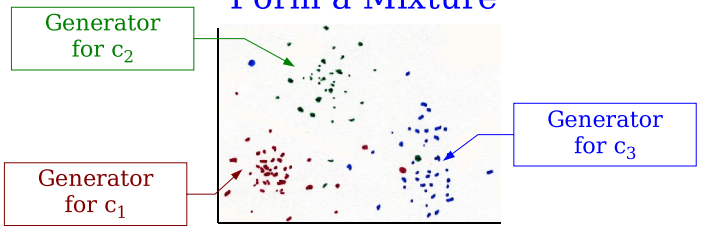
Mixture Models

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



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

$$\sum_j P_j = 1$$

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

Probability Densities

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

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

Probability density of the mixture:

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

Posterior probability:

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

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

Conditional Density

Assume covariance matrix is diagonal with equal elements. Then:

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

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 L that a dataset {**x**i} was generated by a given mixture model?

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

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

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.

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

Gradient Descent on E

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

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

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

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

Gradient Descent (cont.)

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

Do gradient descent on E:

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

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

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 μ_j, σ_j^2, *and* P_j.

Maximization (M) Step

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

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

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

Repeat E and M steps until convergence.

EM Finds a Local Minimum in E (or Local Maximum in L)

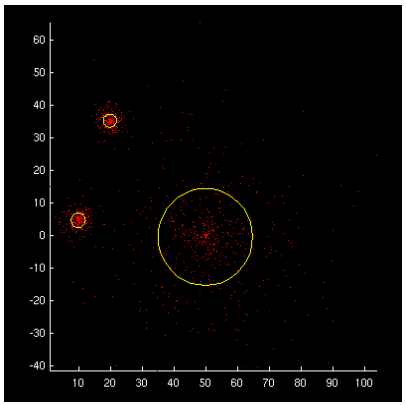
$$\frac{\partial E}{\partial \mu_j} = 0 = -\sum_{i=1}^{n} P(j|\boldsymbol{x_i})\cdot\frac{\|\boldsymbol{x_i}-\mu_j\|}{\sigma_j^2}$$

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

$$\mu_j = \frac{\sum_{i=1}^{n} P(j|\boldsymbol{x_i})\,\boldsymbol{x_i}}{\sum_{i=1}^{n} P(j|\boldsymbol{x_i})}$$

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

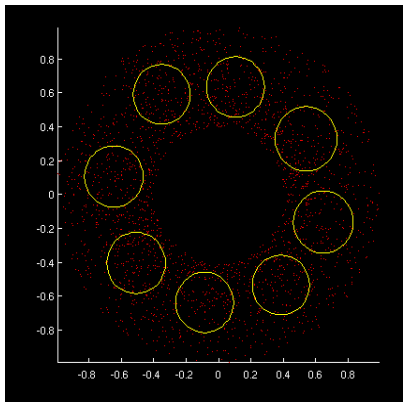
EM Demo



EmDataSet =4

emdemo

EM Demo

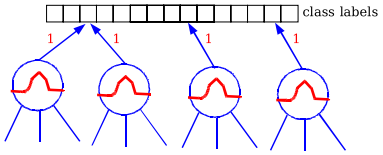


EmDataSet = 3

emdemo

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.

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.

"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 σ²

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

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

On-line Calculation of μ

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

$$\begin{aligned}\mu_{n+1} &= \frac{n\cdot\mu_n + x_{n+1}}{n+1} \\ &= \frac{n}{n+1}\mu_n + \frac{x_n}{n+1} \\ &= \left(1-\frac{1}{n+1}\right)\mu_n + \frac{1}{n+1}x_{n+1}\end{aligned}$$

On-line Calculation of σ²

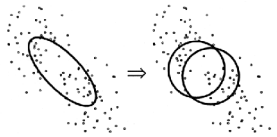
$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_i-\mu_n\right)^2$$

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

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

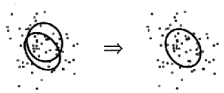
Split and Merge EM
(Ueda, Nakano, Gharamani, and Hinton)

- 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: i j k ...

New components: i' j' k' ...

merge

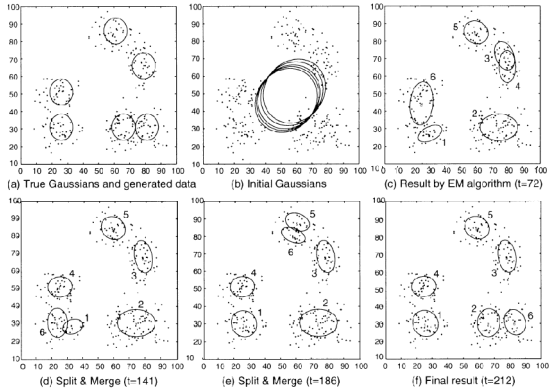
split

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.

Performance of SMEM



Mixture of Factor Analyzers

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

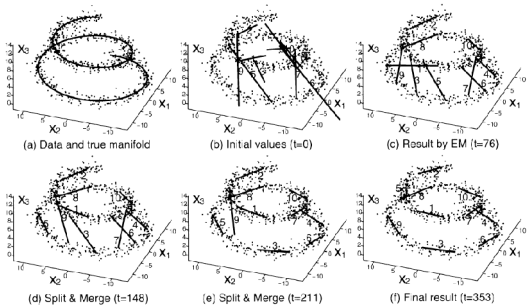
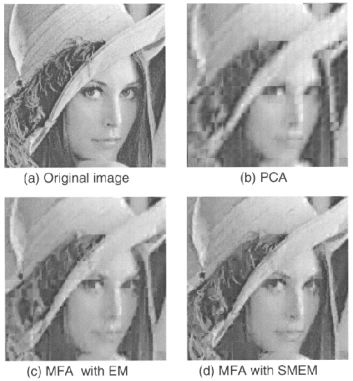


Image Reconstruction



Cheapo Heuristic

Not as good as SMEM, 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