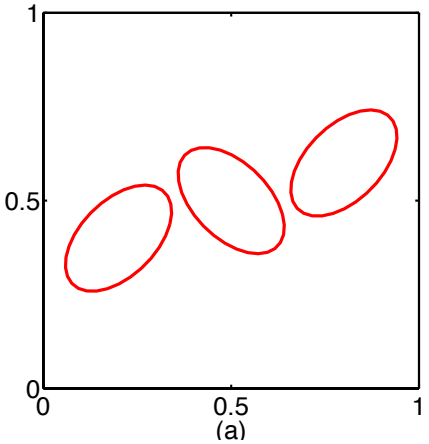
Lecture 11: Mixture models. Expectation maximization (EM).

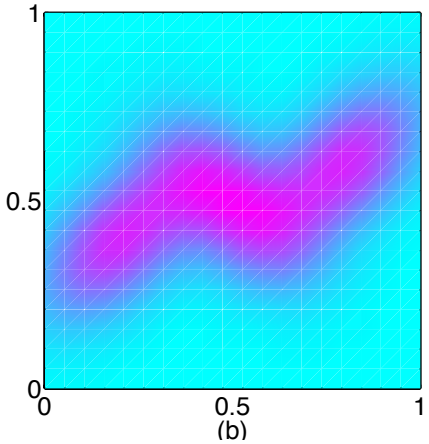
- Clustering as a missing data problem
- Mixture of Gaussians model
- Expectation Maximization (EM)
- EM for Gaussian mixture models

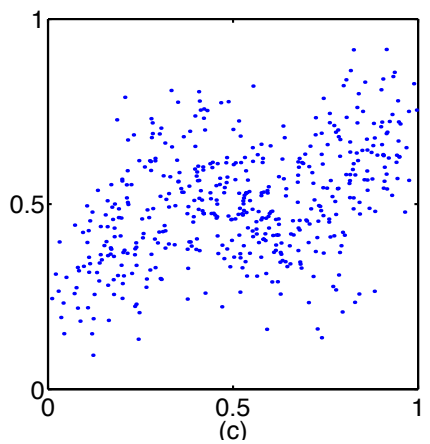
A different view of clustering

- Suppose you had a classification data set, with data coming from K classes
- But someone erased all the class labels!
- You would like to know to what class each example belongs
- This is exactly the problem solved by k-means! k-means gives the maximum likelihood solution under a very restrictive assumption: each class has a Gaussian distribution with its own mean but with unit covariance matrix
- Now we relax this assumption: each class will have a Gaussian distribution with its own mean and its own arbitrary covariance matrix
- This model is called a *mixture of Gaussians*

What do mixtures of Gaussian look like?







- Can be very expressive if you have enough components
- But too many components may lead to overfitting
- We first discuss learning parameters for a mixture with known number of components

Mixtures of Gaussian: definition

- A Gaussian mixture with k components is parametrized by k mixing weights $p_1, \cdots, p_k$ and k means $\mu_1, \cdots, \mu_k$ and covariance matrices $\Sigma_1, \cdots, \Sigma_k$.
- Probabilistic model:
  - Discrete random variable h with $\mathbb{P}(h = i) = p_i$ for $i = 1, \cdots, k$ (corresponds to choosing a component of the mixture)
  - x given $h = i$ follows a normal distribution $\mathcal{N}(\mu_i, \Sigma_i)$ for each i.
- Drawing a point from a Gaussian mixture
  1. Draw h from the discrete distribution defined by the p_i
  2. Draw x from $\mathcal{N}(\mu_h, \Sigma_h)$
- Our first example of a latent variable model.

More generally: Missing values / semi-supervised learning

- Suppose we have a generative model of supervised learning data with parameters θ, but the values y are missing in some of the instances
- The log likelihood of the data can be written as:

$$\log L(\theta) = \sum_{\text{complete data}} \log P(\mathbf{x}_i, y_i|\theta) + \sum_{\text{incomplete data}} \log P(\mathbf{x}_i|\theta)$$

- For the second term, we must consider *all possible values* for y:

$$\sum_{\text{incomplete data}} \log P(\mathbf{x}_i|\theta) = \sum_{\text{incomplete data}} \log \left(\sum_y P(\mathbf{x}_i, y|\theta) \right)$$

Data likelihood with missing values

- Log likelihood of the data:

$$\log L(\theta) = \sum_{\text{complete data}} \log P(\mathbf{x}_i, y_i|\theta) + \sum_{\text{incomplete data}} \log \left(\sum_y P(\mathbf{x}_i, y|\theta) \right)$$

- In semi-supervised learning, some y's are observed, some are not. Even so, using the $\mathbf{x}$'s with unobserved y's can be helpful.
- In the clustering problem, y is *never* observed, so we only have the second term above.

Maximum likelihood learning with missing values

Complete data	Missing values
Parameters of the model can be estimated locally, and independently	Parameters cannot be estimated independently
Log-likelihood has a unique maximum	There are many local maxima! Maximizing the likelihood becomes a non-linear optimization problem
Under certain assumptions, there is a nice, closed-form solution for the parameters	Closed-form solutions cannot be obtained

Two solutions

1. *Gradient ascent:* follow the gradient of the likelihood with respect to the parameters
2. *Expectation minimization:* use the current parameter setting to construct a local approximation of the likelihood which is "nice" and can be optimized easily

Gradient ascent

- Move parameters in the direction of the gradient of the log-likelihood
- Pros:
  - It is easy to compute the gradient at any parameter setting, provided we work with usual distributions
  - We already know how to do this!
- Cons:
  - The gradient needs to be *projected on the space of legal parameters* (i.e., we need to ensure that we get "legal" probability distributions or probability density functions)
  - Sensitive to parameters (e.g. learning rates) and possibly slow
- More efficient gradient-style have been developed recently, e.g. sub-gradients, contrastive divergence

Expectation Maximization (EM)

- A general purpose method for learning from incomplete data
- Main idea:
  - If we had complete data we could easily maximize the likelihood
  - But because the data is incomplete, we get a summation inside the $\log$, which makes the optimization much harder
  - So in the case of missing values, we will "fantasize" what they should be, based on the current parameter setting
  - In other words, we *fill in the missing values based on our current expectation*
  - Then we *compute new parameters, which maximize the likelihood* of the completed data
  - In terms of the previous slide, we estimate y given θ, then we reestimate θ given y, then we reestimate y given the new θ, . . .

Gaussian discriminant analysis

- Inputs $\mathbf{x}_i$ are real-valued vectors, each with a class $y_i \in \{1, 2, \ldots, K\}$. (We saw $y_i \in \{0, 1\}$, previously.)
- The parameters of the model are:
  - The prior probabilities, $p_k = P(y = k)$.
  - Mean and covariance matrix, μ_k, Σ_k, defining a multivariate Gaussian distribution for examples in class k.
- All parameters chosen to maximize the log likelihood of the data:

$$\log L = \sum_i (\log P(y_i) + \log P(\mathbf{x}_i | y_i))$$

Maximum likelihood solution

- Let $\delta_{ik} = 1$ if $y_i = k$ and 0 otherwise
- The class probabilities are determined by the empirical frequency of examples in each class:

$$p_k = \frac{\sum_i \delta_{ik}}{\sum_k \sum_i \delta_{ik}} = \frac{1}{m} \sum_i \delta_{ik}$$

- The mean and covariance matrix for class k are the empirical mean and covariance of the examples in that class:

$$\begin{aligned} \mu_k &= \frac{\sum_i \delta_{ik} \mathbf{x}_i}{\sum_i \delta_{ik}} \\ \Sigma_k &= \frac{\sum_i \delta_{ik} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T}{\sum_i \delta_{ik}} \end{aligned}$$

EM for Mixture of Gaussians

- We start with an initial guess for the parameters p_k, μ_k, Σ_k
- We will alternate an:
  - *expectation step (E-step)*, in which we "complete" the data—estimating the y_i
  - *maximization step (M-step)*, in which we re-compute the parameters p_k, μ_k, Σ_k
- In the *hard EM* version, completing the data means that each data point is assumed to be generated by *exactly one Gaussian*—taken to be the most likely assignment.
  (This is roughly equivalent to the setting of K-means clustering.)
- In the *soft EM* version (also usually known as EM), we assume that each data point could have been generated from *any component*
  - We estimate probabilities $P(y_i = k) = P(\delta_{ik} = 1) = E(\delta_{ik})$
  - Each $\mathbf{x}_i$ contributes to the mean and variance estimate of each component.

Hard EM for Mixture of Gaussians

1. Guess initial parameters p_k, μ_k, Σ_k for each class k
2. Repeat until convergence:
   (a) *E-step:* For each instance i, compute the most likely class:

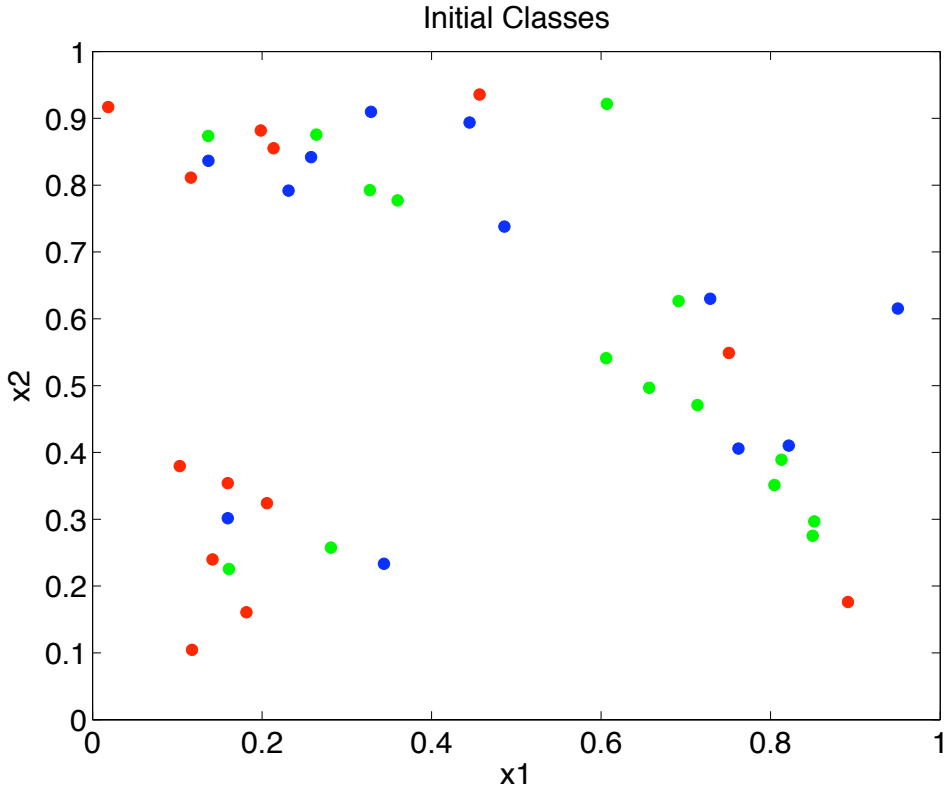
$$y_i = \arg\max_k P(y = k|\mathbf{x}_i) = \arg\max_k P(\mathbf{x}_i|y = k)P(y = k)$$

   (b) *M-step:* Update the parameters of the model $(p_k, \mu_k, \Sigma_k, \forall k = 1 \ldots K)$ to maximize the likelihood of the data:

$$p_k = \frac{1}{m}\sum_{i=1}^{m}\delta_{ik} \qquad \mu_k = \frac{\sum_{i=1}^{m}\delta_{ik}\mathbf{x}_i}{\sum_{i=1}^{m}\delta_{ik}}$$

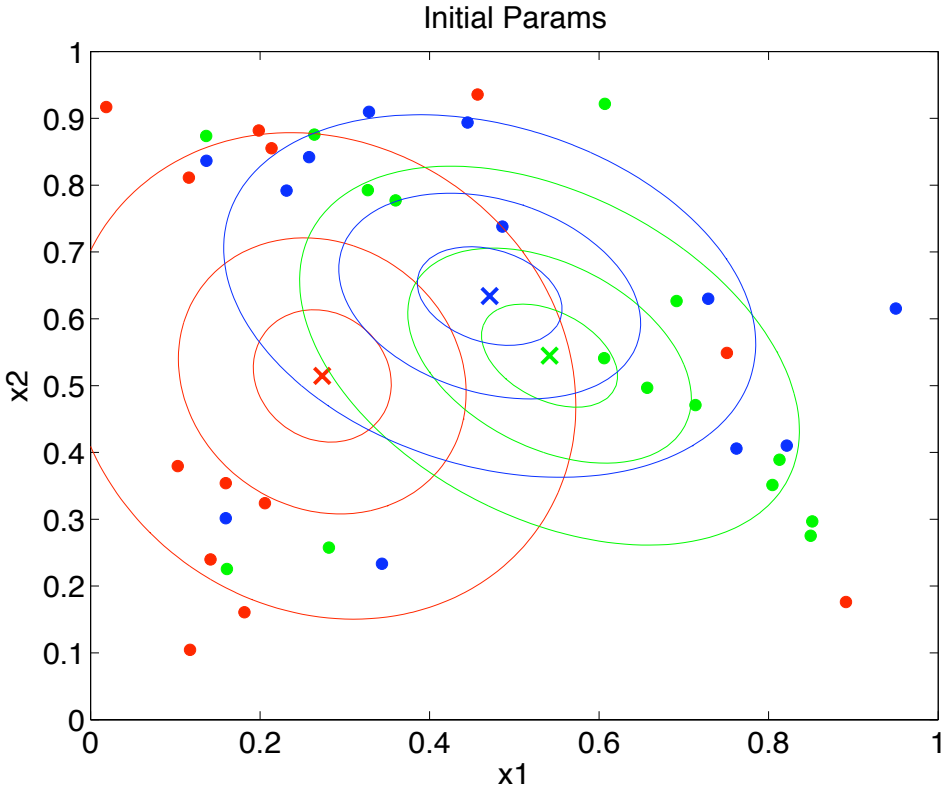
$$\Sigma_k = \frac{\sum_{i=1}^{m}\delta_{ik}\left(\mathbf{x}_i - \mu_k\right)\left(\mathbf{x}_i - \mu_k\right)^T}{\sum_{i=1}^{m}\delta_{ik}}$$

Hard EM for Mixture of Gaussians: Example



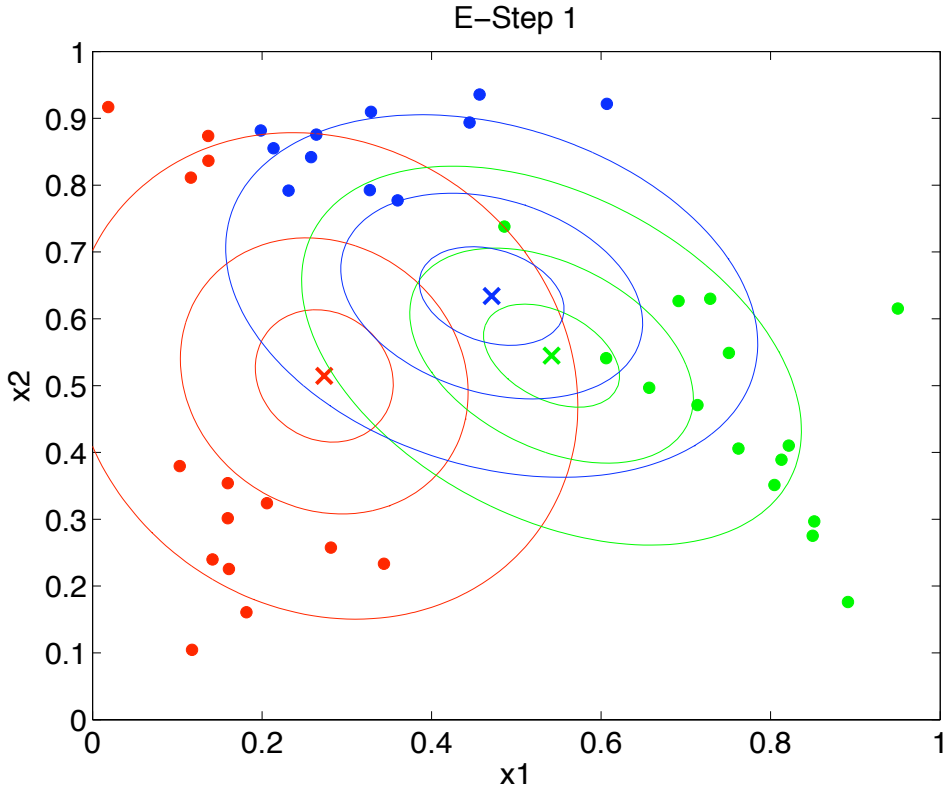
$K = 3$, initial assignment of points to components is random

Hard EM for Mixture of Gaussians: Example

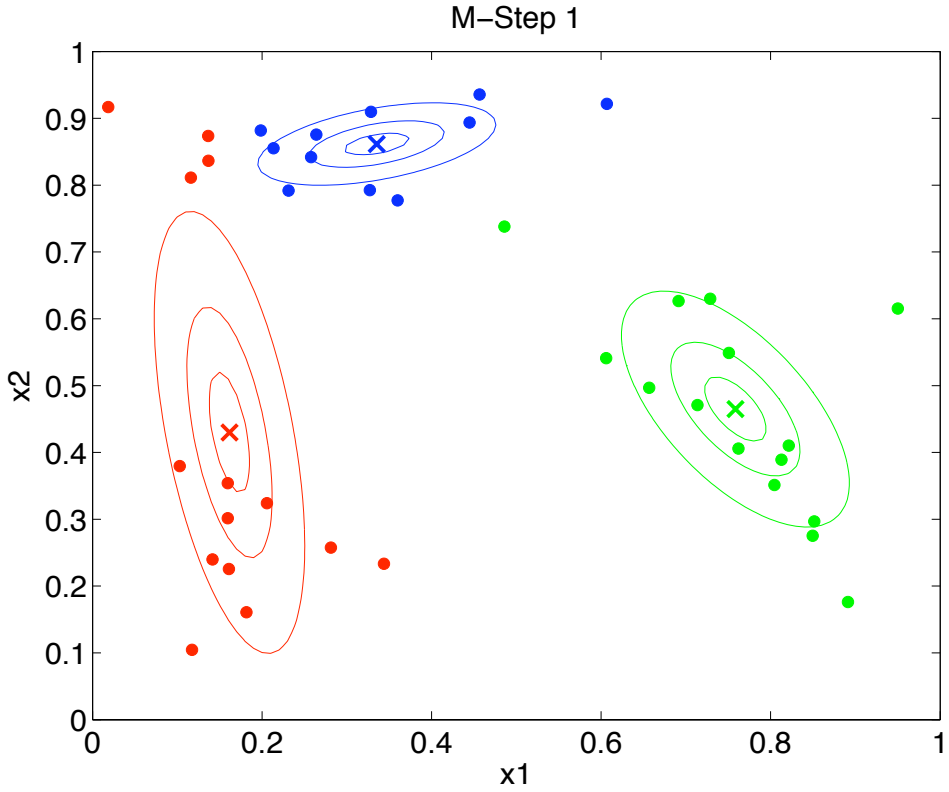


Initial parameters (means and variances) computed from initial assignments

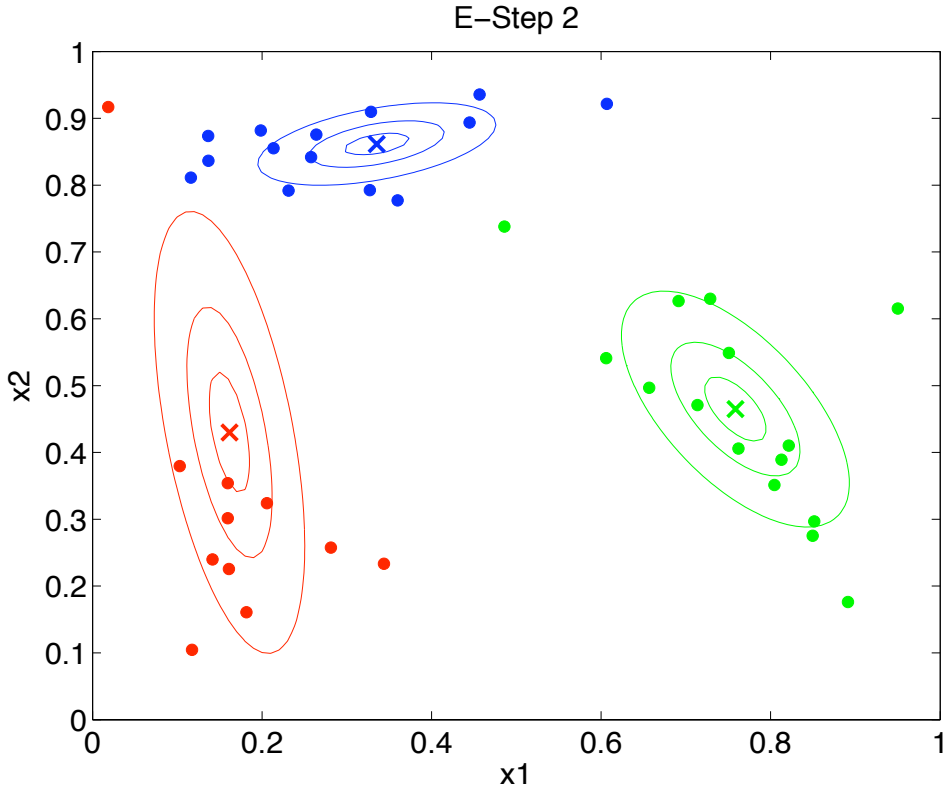
Hard EM for Mixture of Gaussians: Example



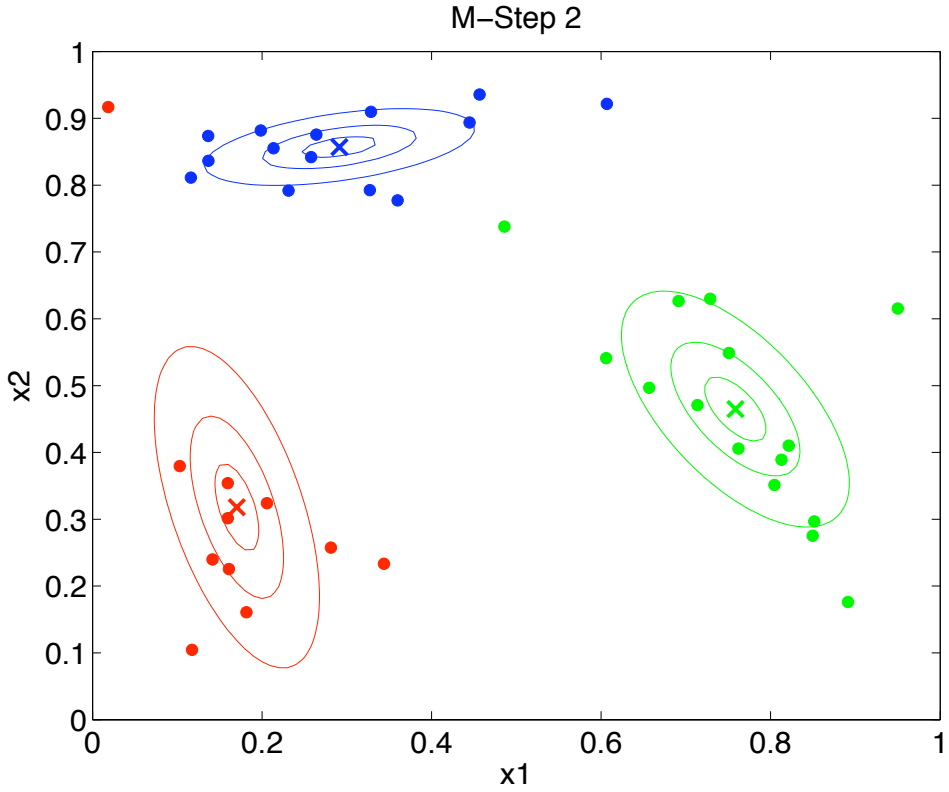
Hard EM for Mixture of Gaussians: Example



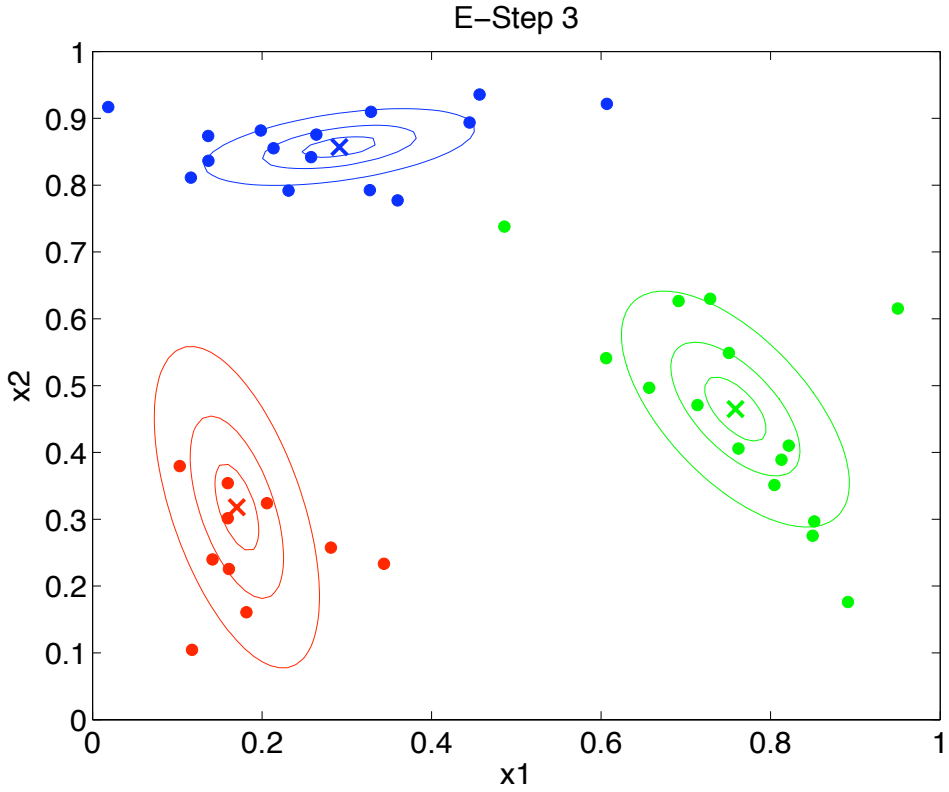
Hard EM for Mixture of Gaussians: Example



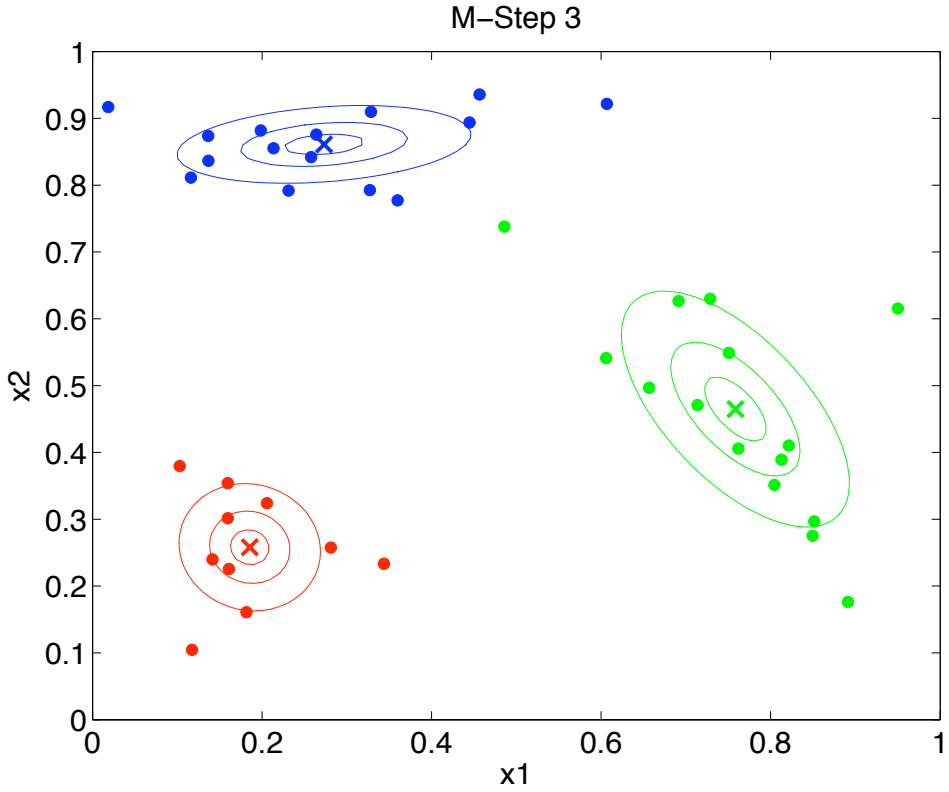
Hard EM for Mixture of Gaussians: Example



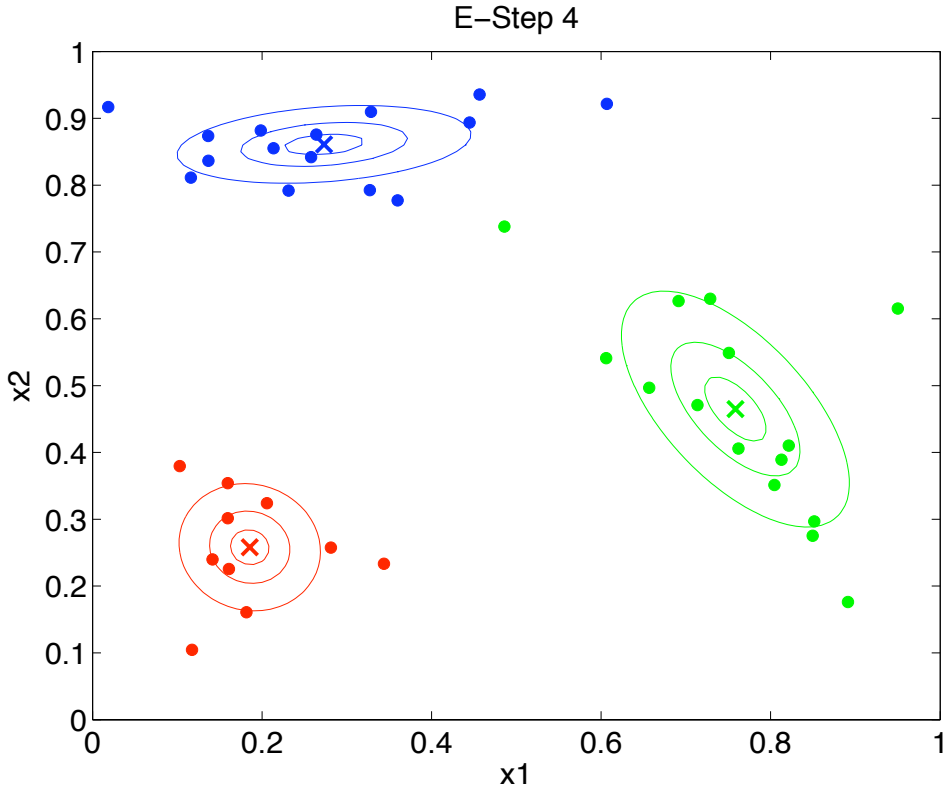
Hard EM for Mixture of Gaussians: Example



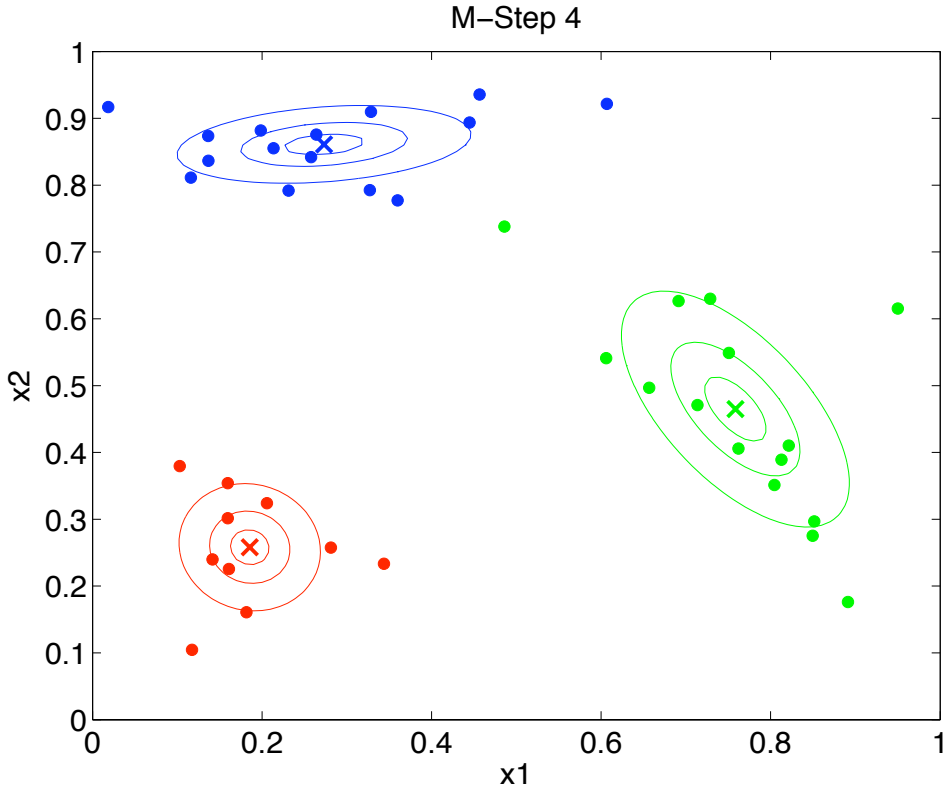
Hard EM for Mixture of Gaussians: Example



Hard EM for Mixture of Gaussians: Example



Hard EM for Mixture of Gaussians: Example



Soft EM for Mixture of Gaussians

1. Guess initial parameters p_k, μ_k, Σ_k for each class k
2. Repeat until convergence:
   (a) *E-step:* For each instance i and class k, compute the probabilities of class membership:

$$w_{ik} = P(y_i = k|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = k)P(y_i = k)}{P(\mathbf{x}_i)}$$

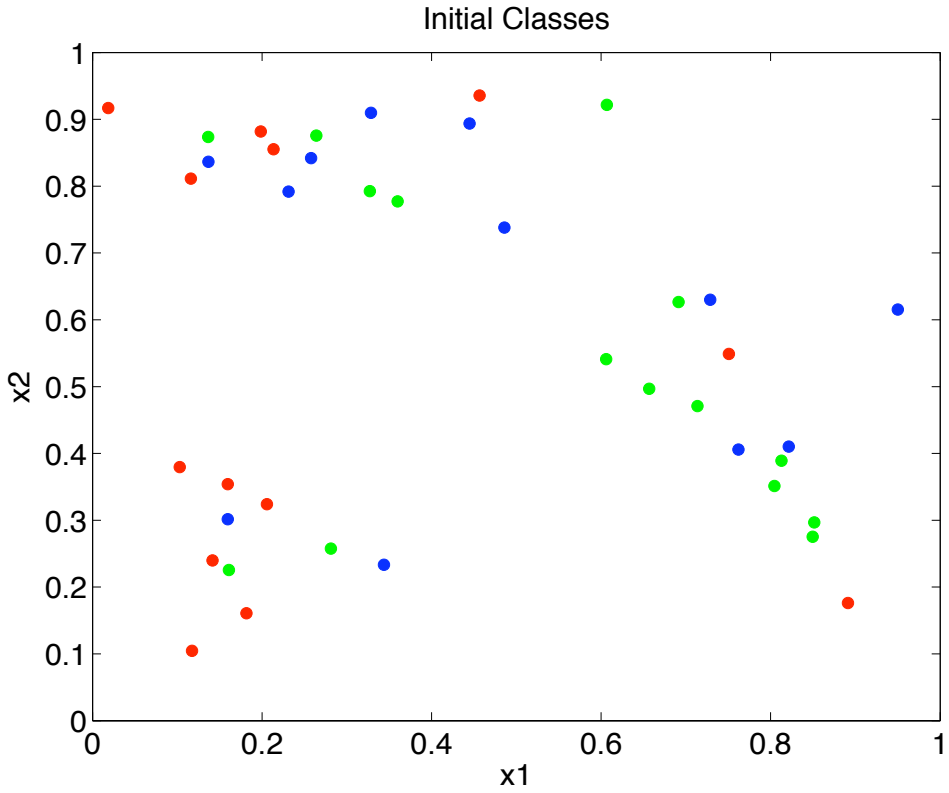
   I.e., instances are "partially assigned" to each class, according to w_{ik}

   (b) *M-step:* Update the parameters of the model to maximize the likelihood of the data:

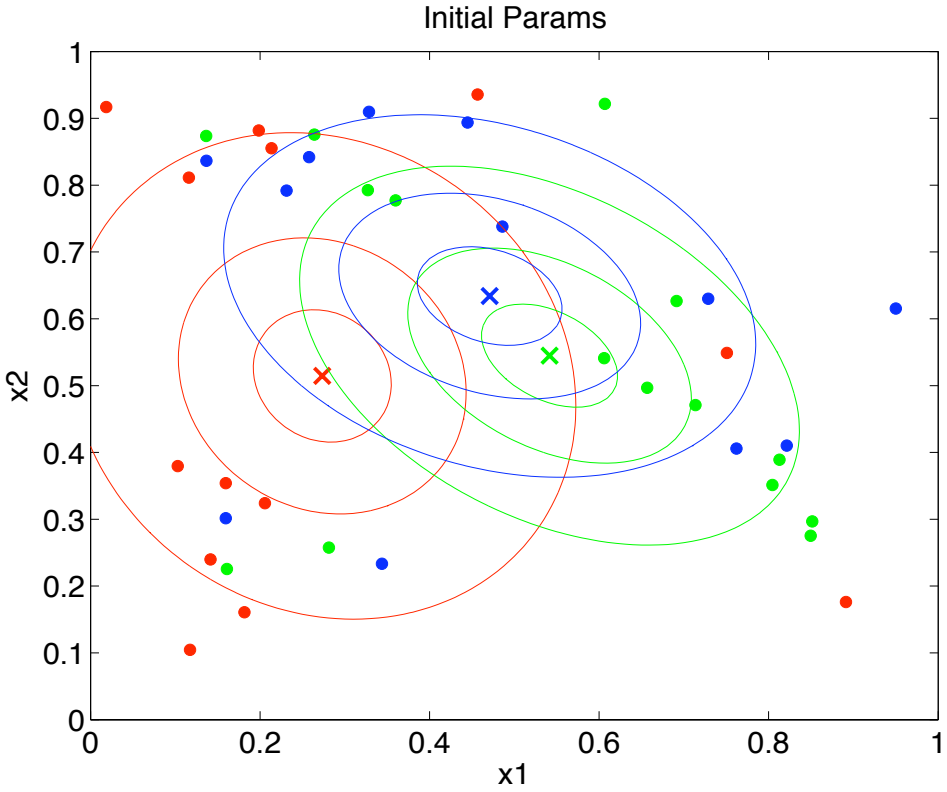
$$p_k = \frac{1}{m}\sum_{i=1}^{m} w_{ik} \qquad \mu_k = \frac{\sum_{i=1}^{m} w_{ik}\mathbf{x}_i}{\sum_{i=1}^{m} w_{ik}}$$

$$\Sigma_k = \frac{\sum_{i=1}^{m} w_{ik}\left(\mathbf{x}_i - \mu_k\right)\left(\mathbf{x}_i - \mu_k\right)^T}{\sum_{i=1}^{m} w_{ik}}$$

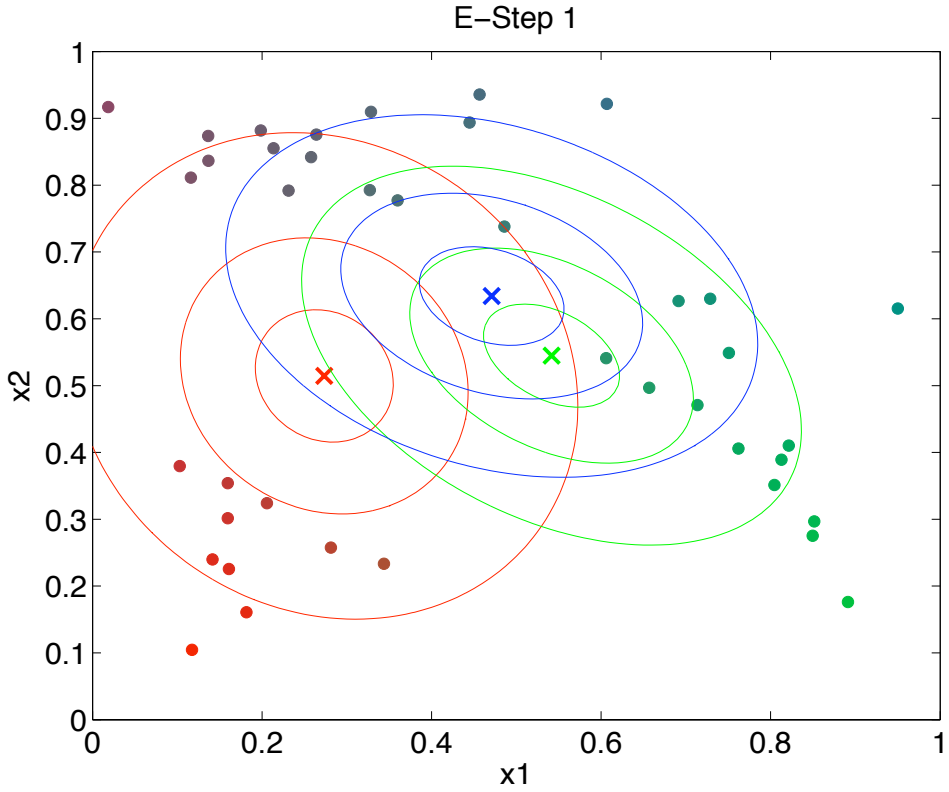
Soft EM for Mixture of Gaussians: Example



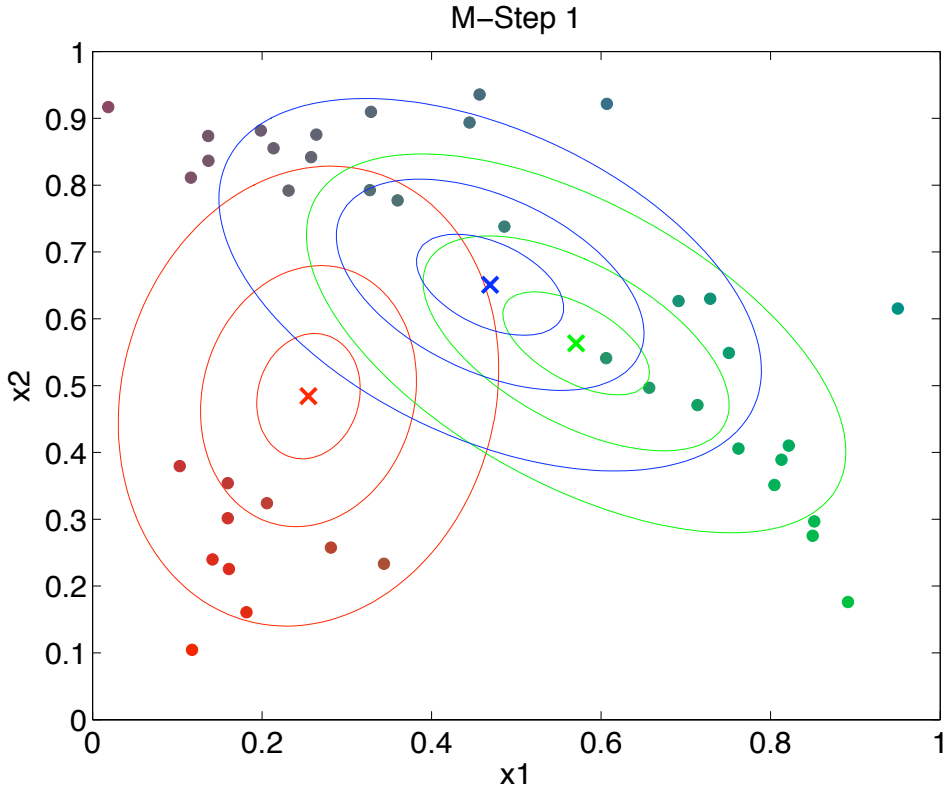
Soft EM for Mixture of Gaussians: Example



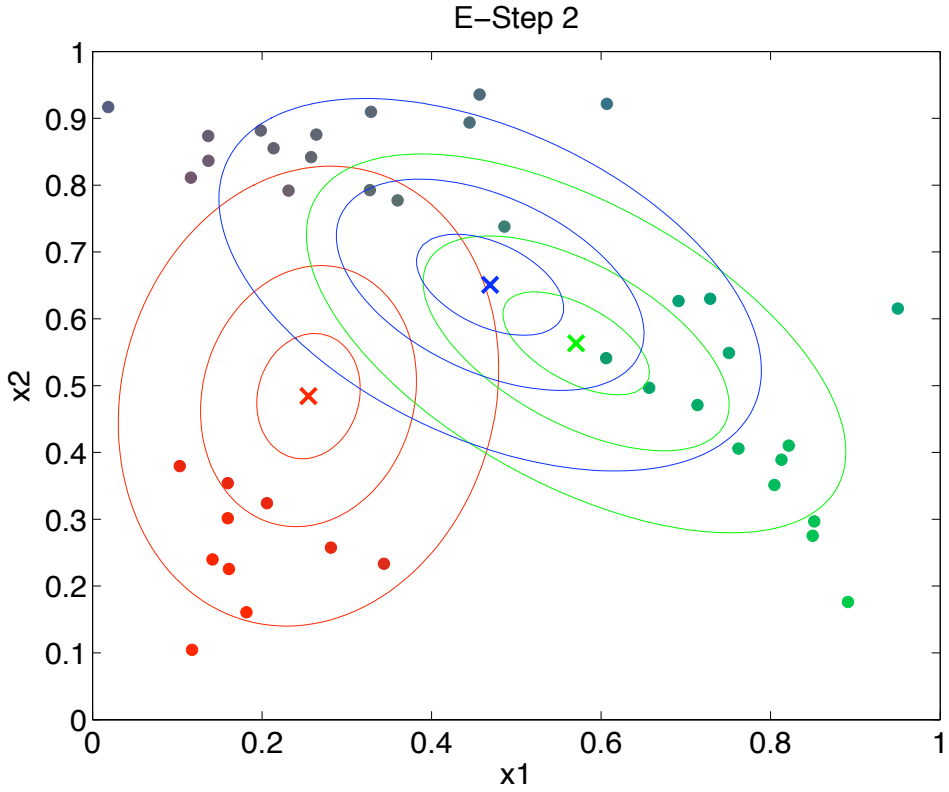
Soft EM for Mixture of Gaussians: Example



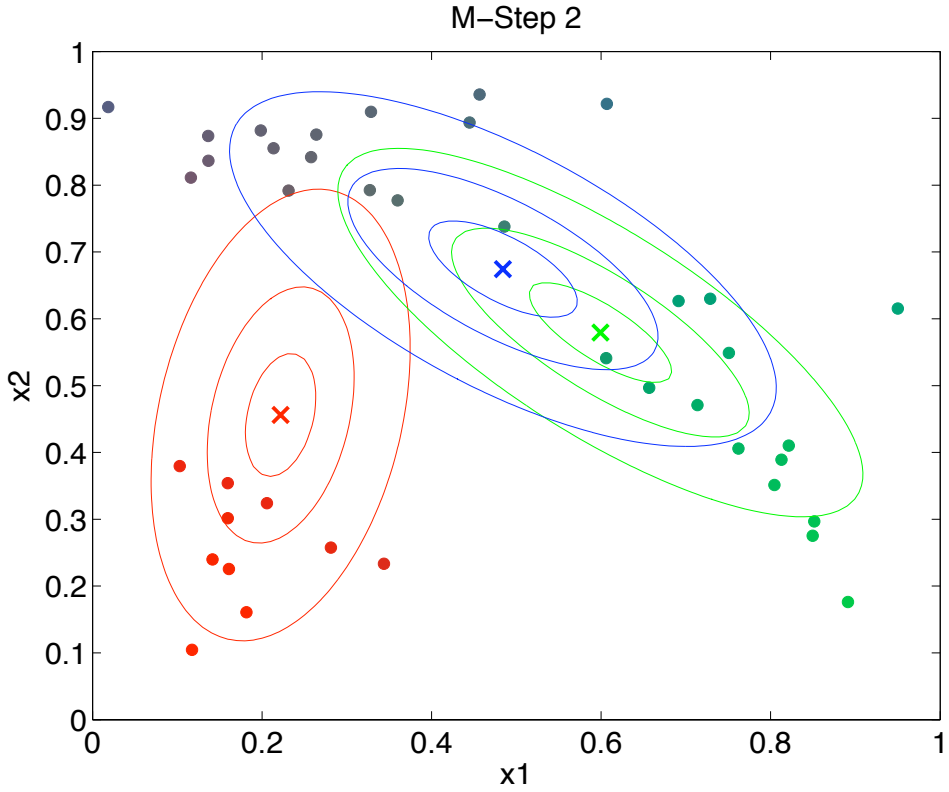
Soft EM for Mixture of Gaussians: Example



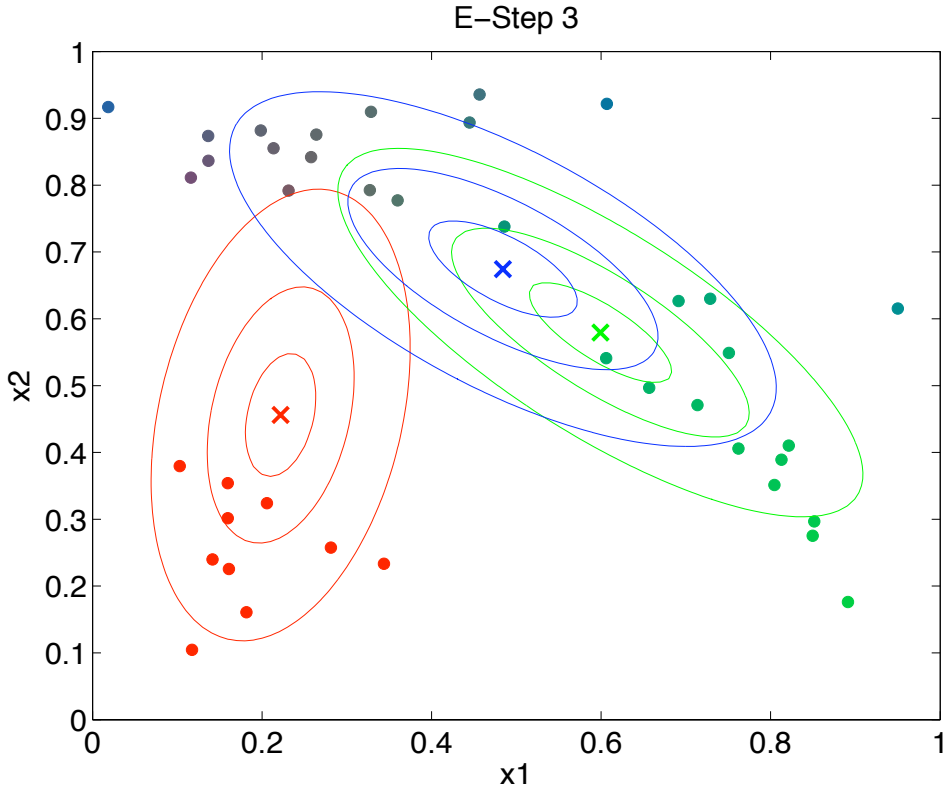
Soft EM for Mixture of Gaussians: Example



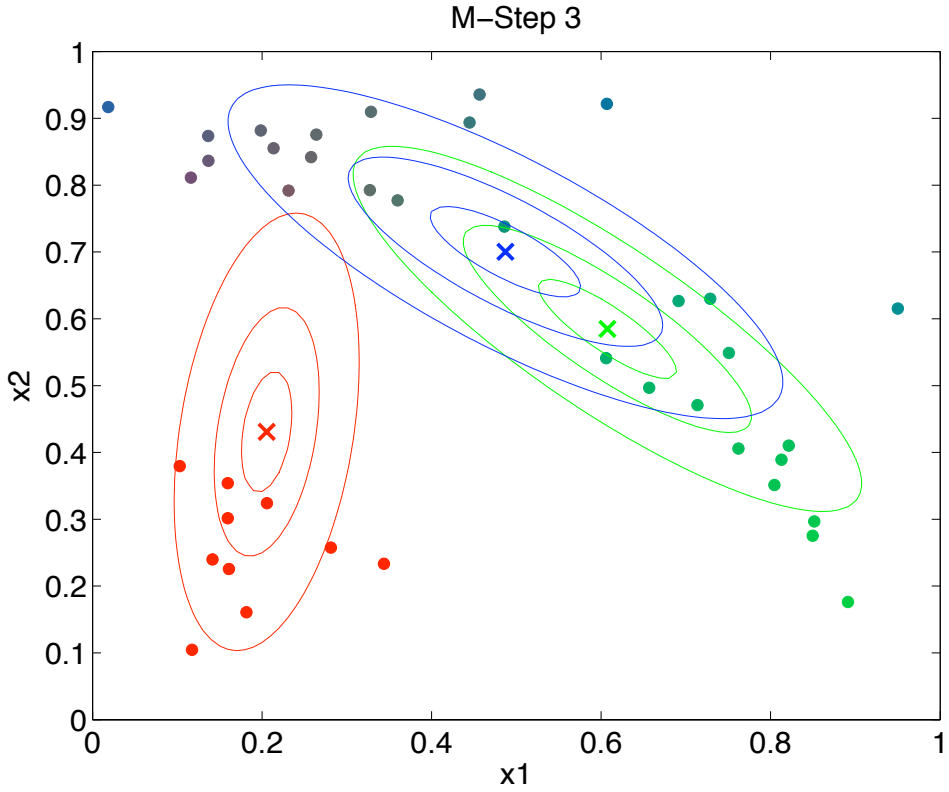
Soft EM for Mixture of Gaussians: Example



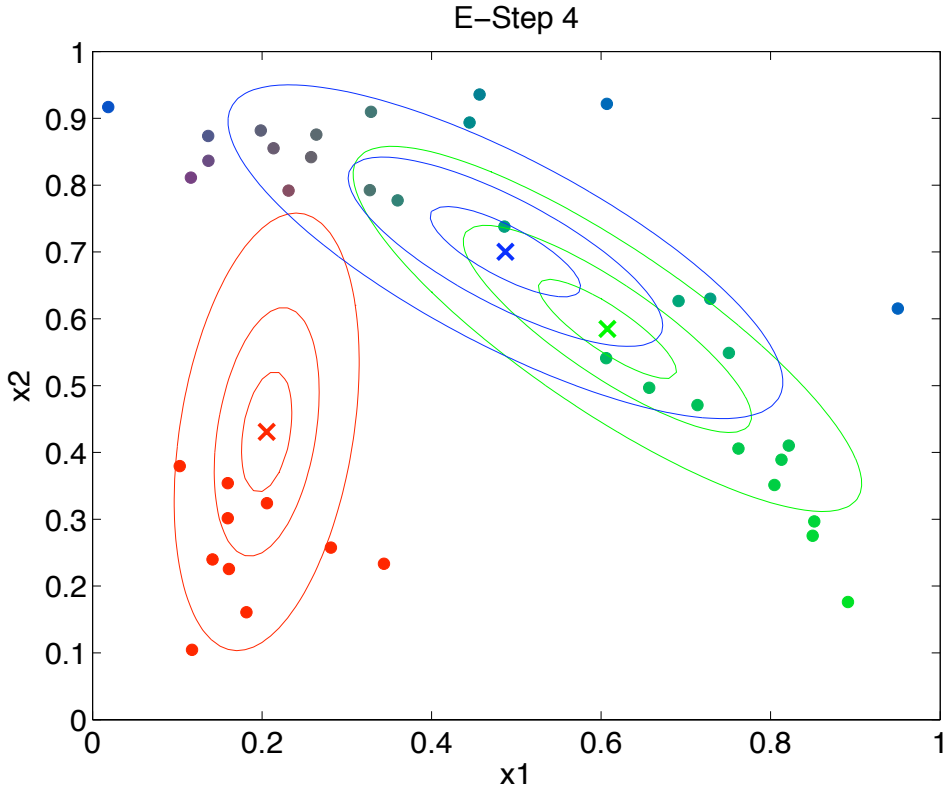
Soft EM for Mixture of Gaussians: Example



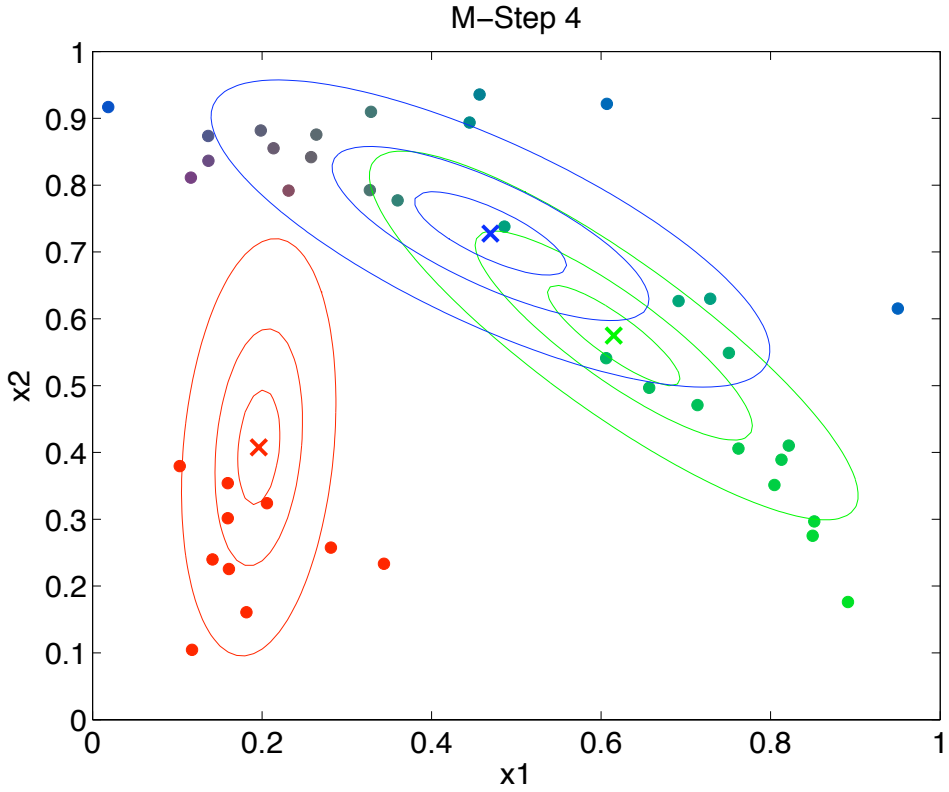
Soft EM for Mixture of Gaussians: Example



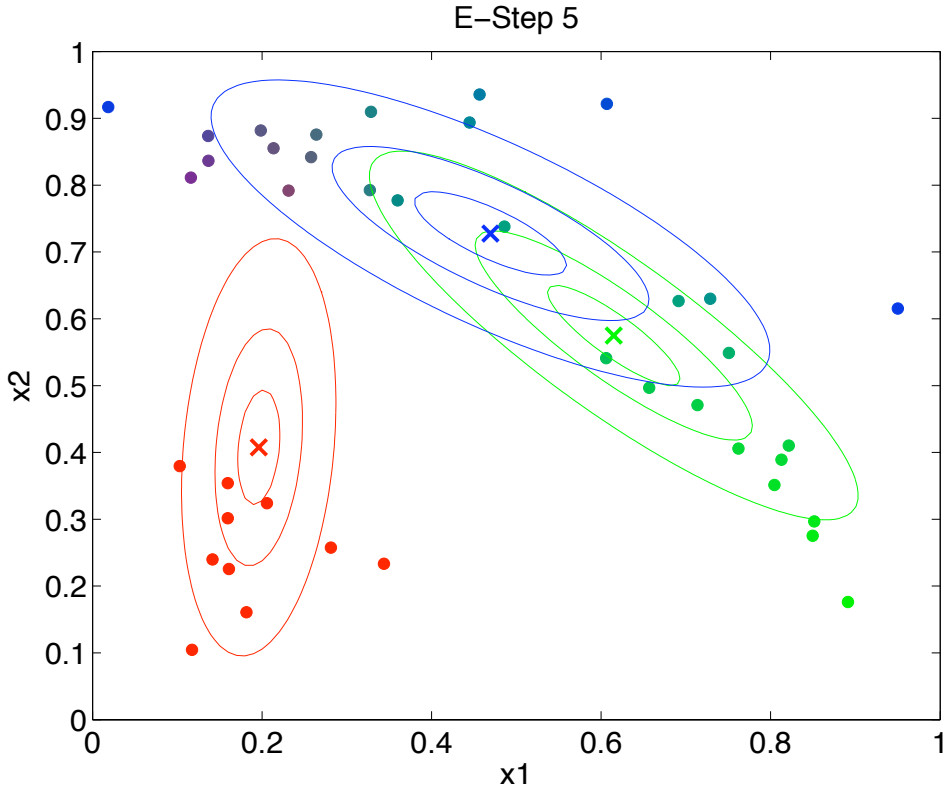
Soft EM for Mixture of Gaussians: Example



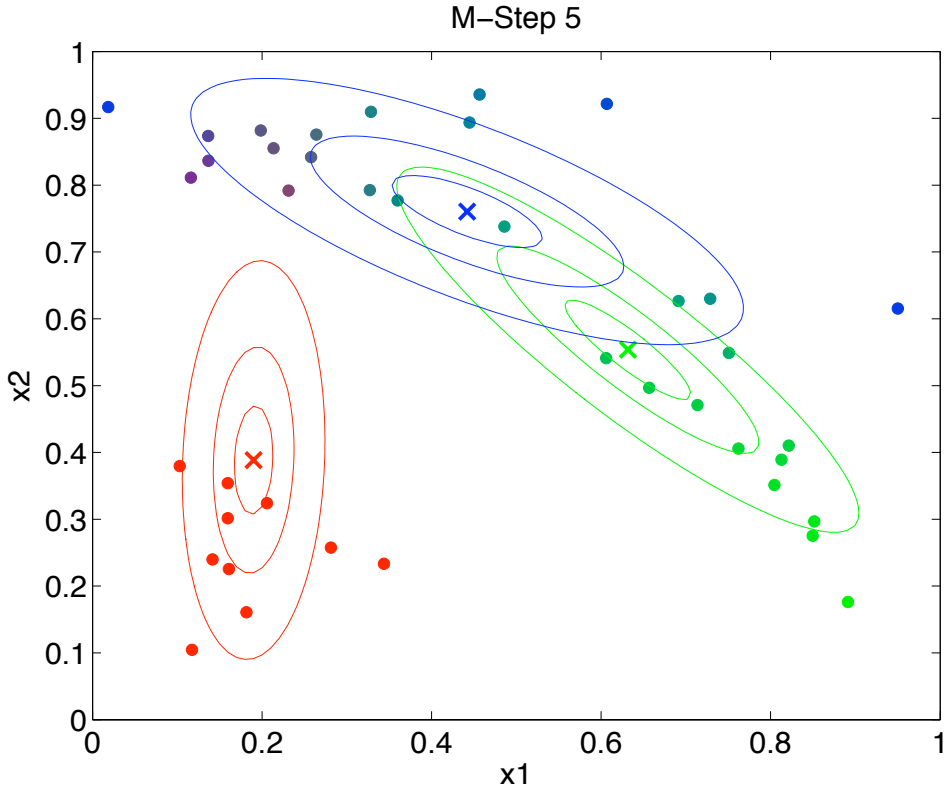
Soft EM for Mixture of Gaussians: Example



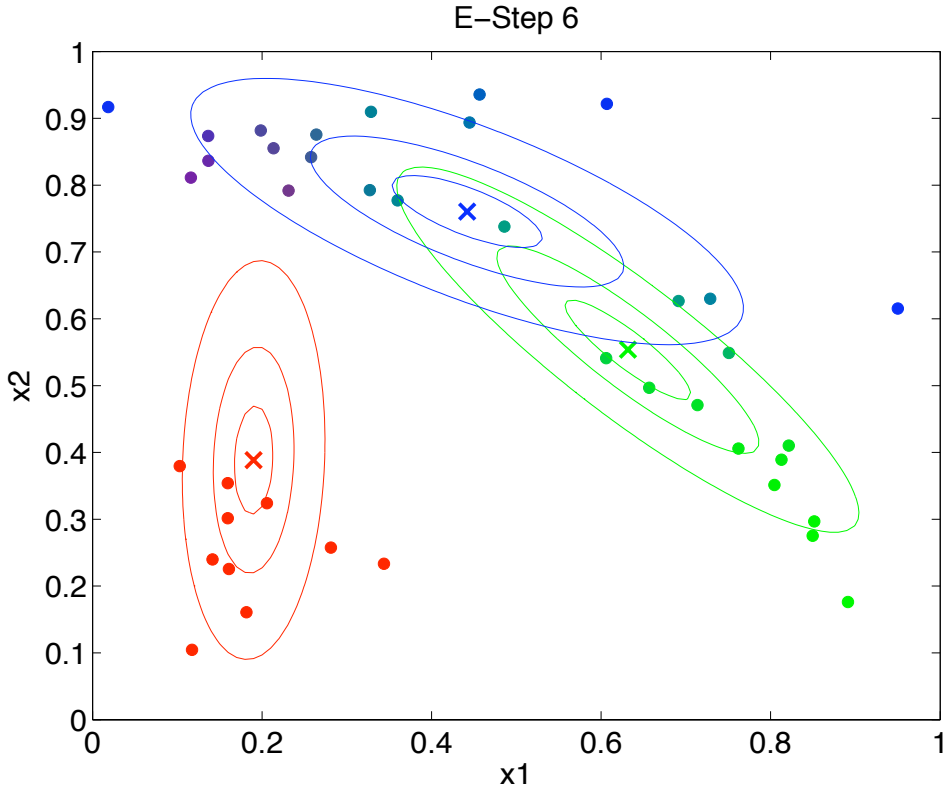
Soft EM for Mixture of Gaussians: Example



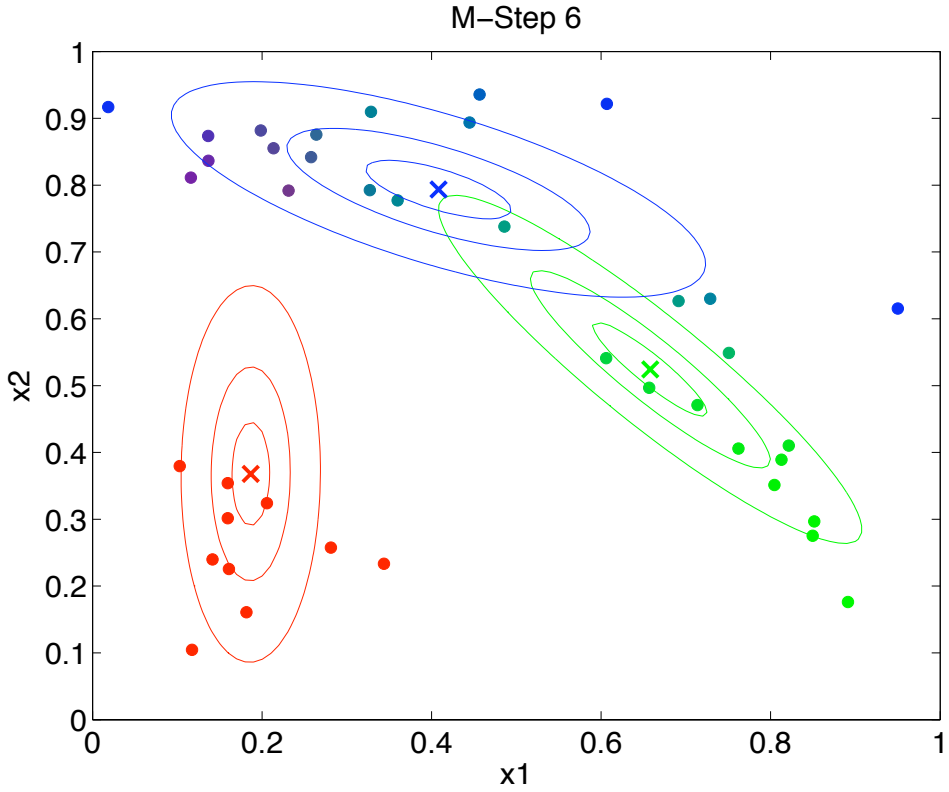
Soft EM for Mixture of Gaussians: Example



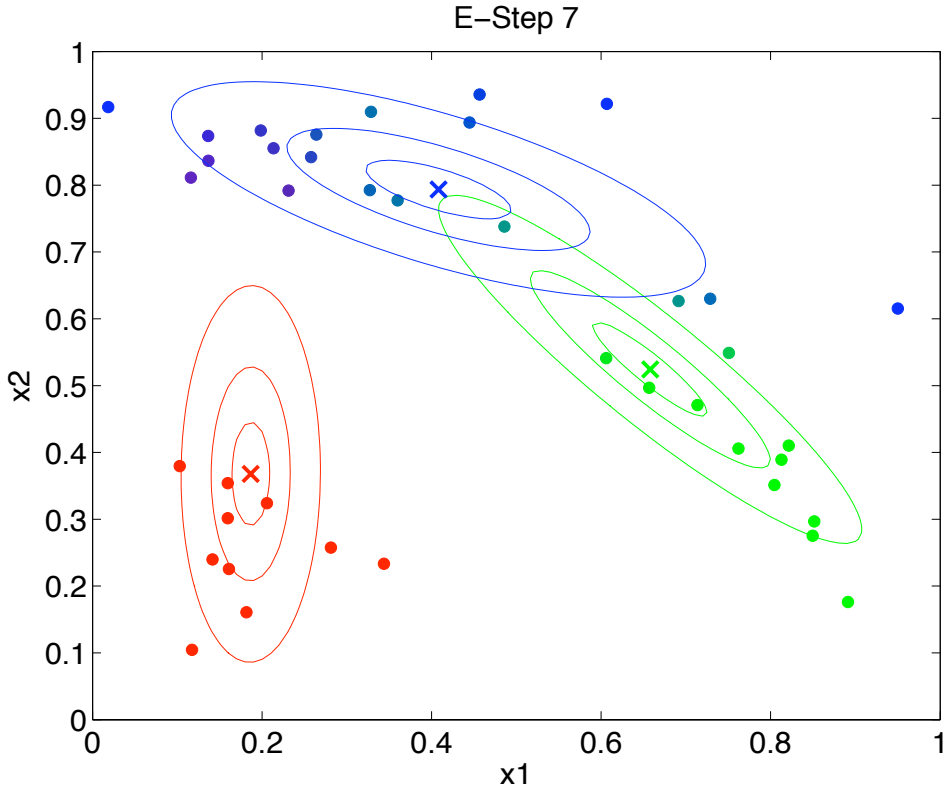
Soft EM for Mixture of Gaussians: Example



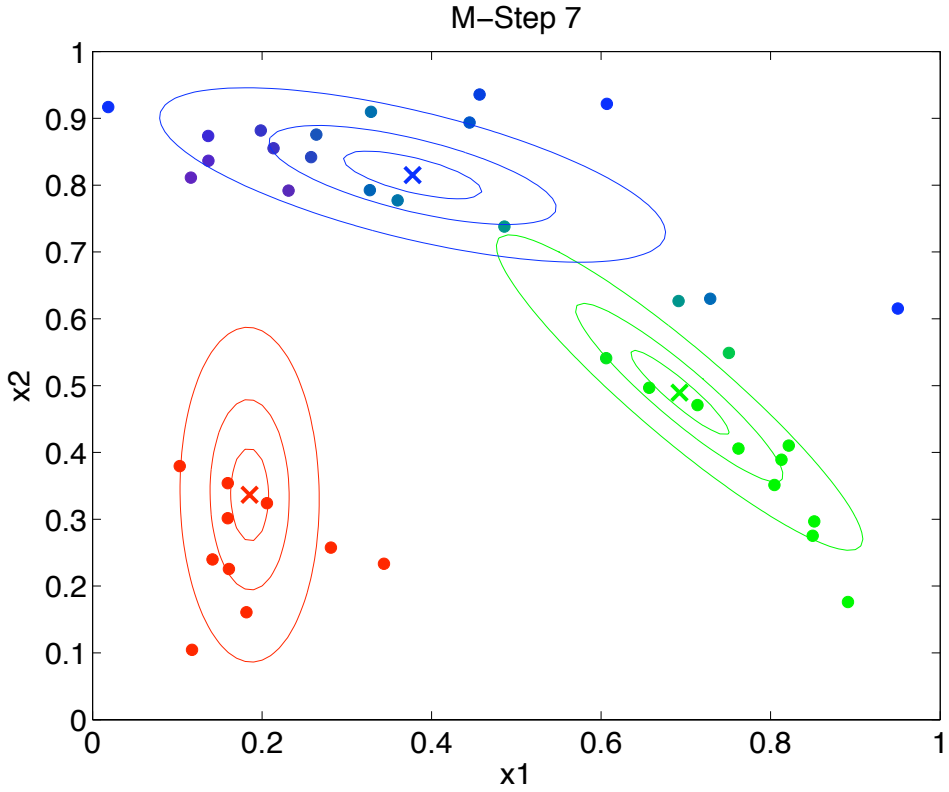
Soft EM for Mixture of Gaussians: Example



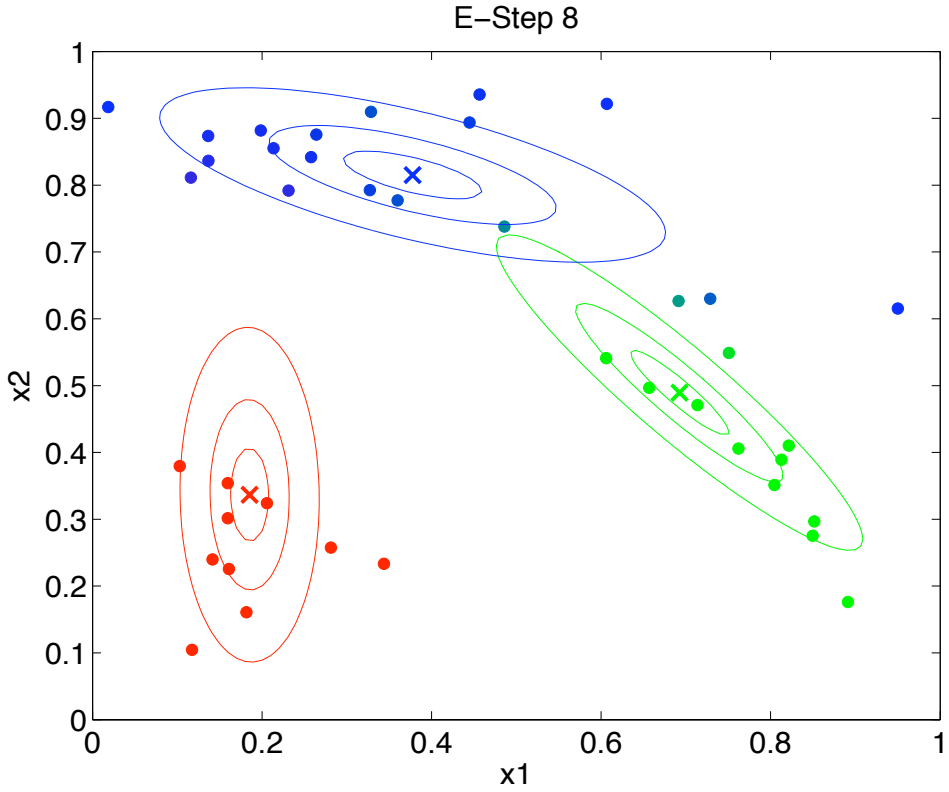
Soft EM for Mixture of Gaussians: Example



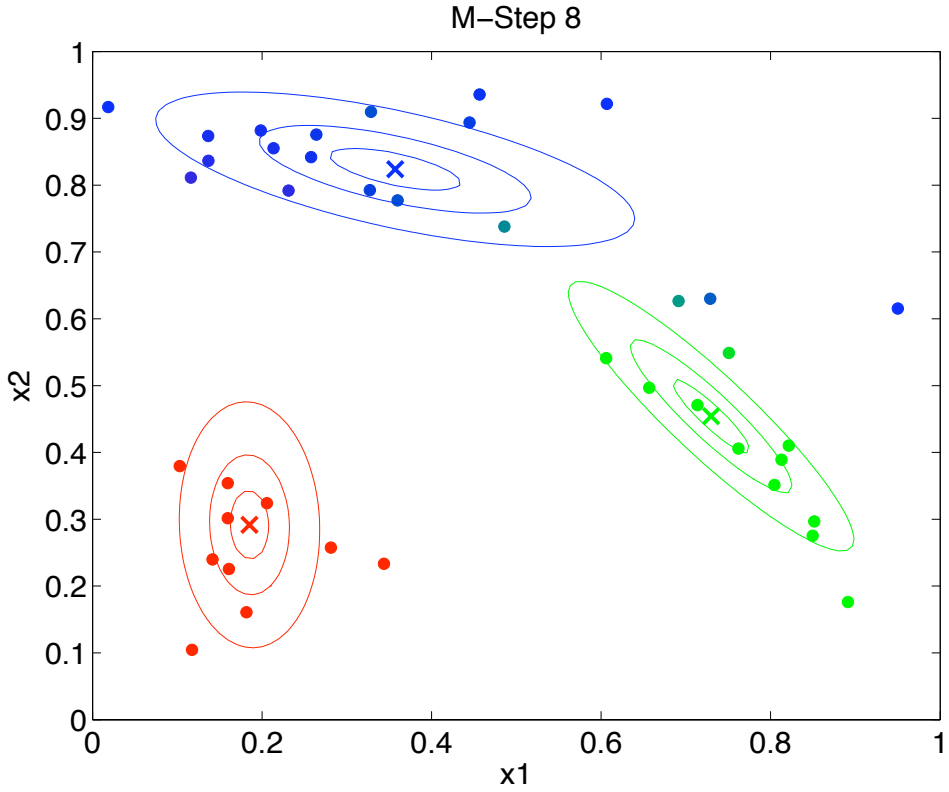
Soft EM for Mixture of Gaussians: Example



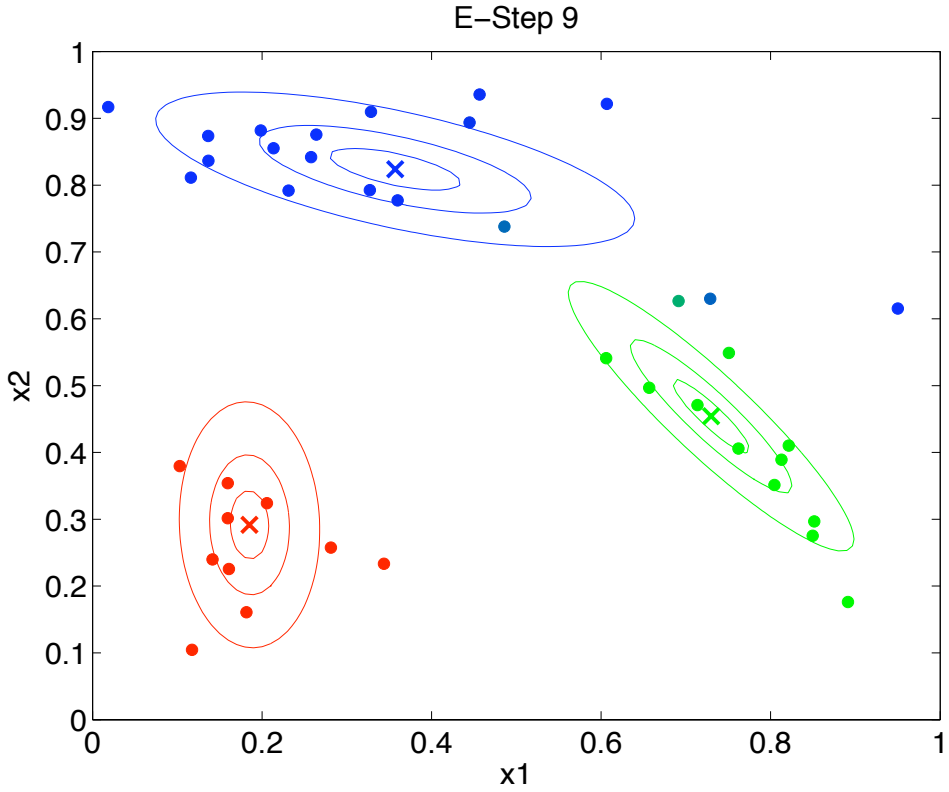
Soft EM for Mixture of Gaussians: Example



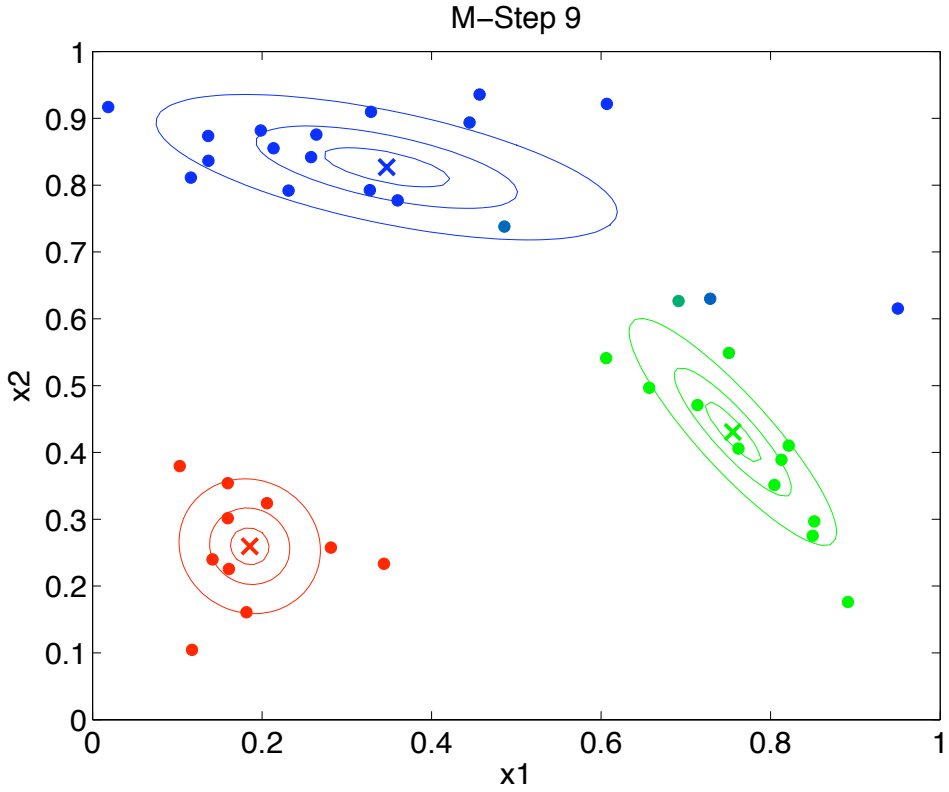
Soft EM for Mixture of Gaussians: Example



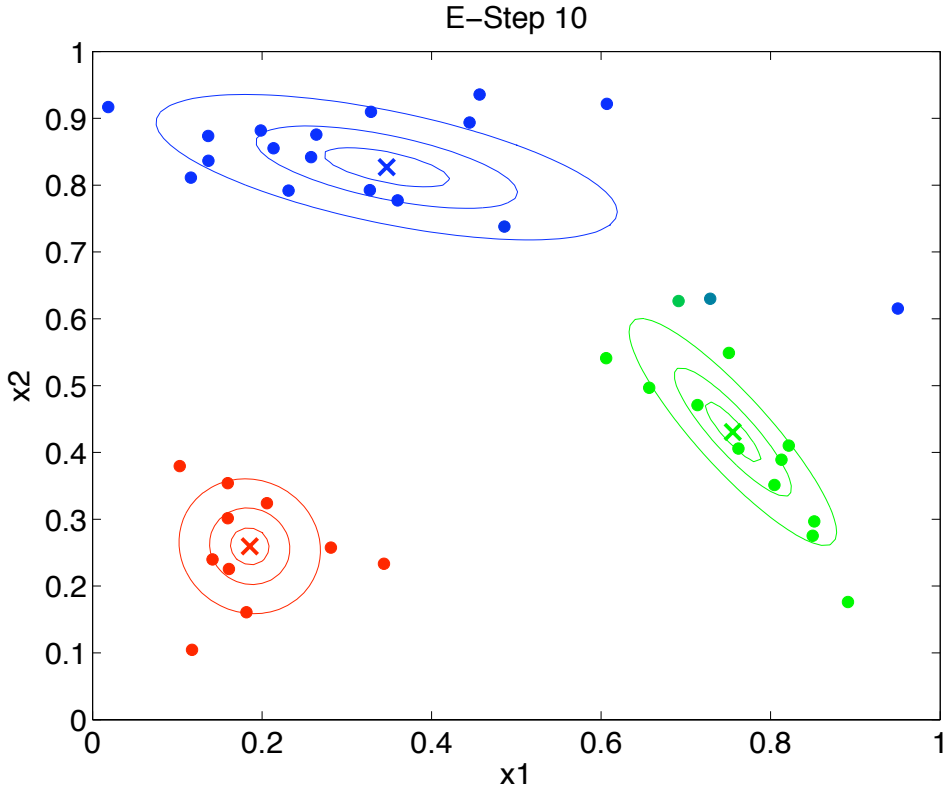
Soft EM for Mixture of Gaussians: Example



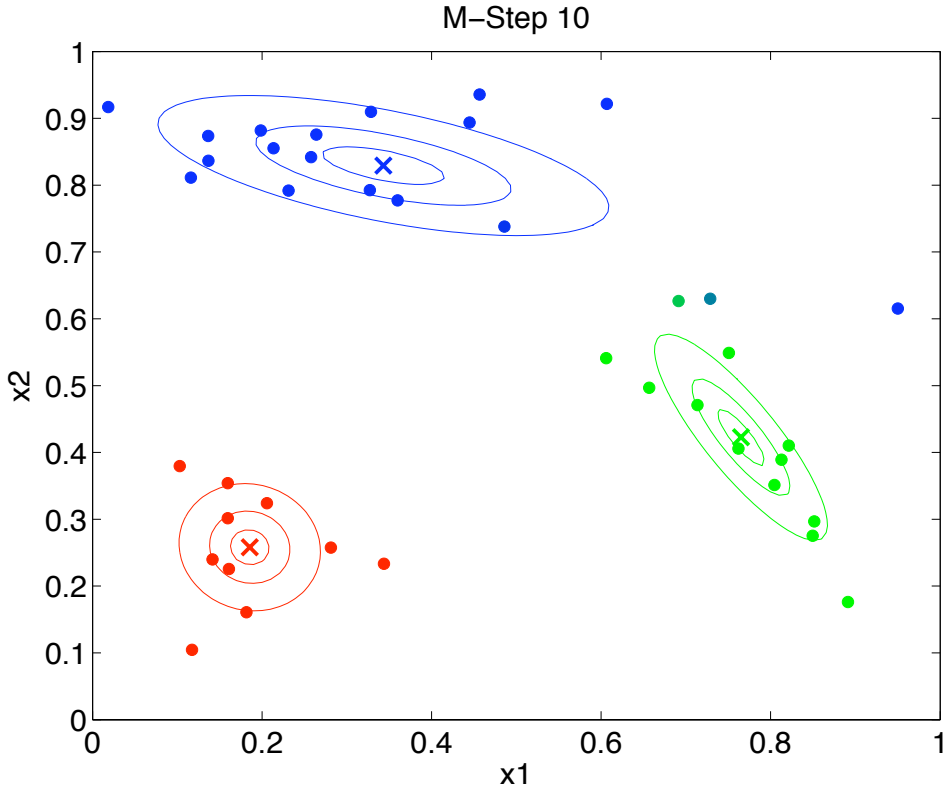
Soft EM for Mixture of Gaussians: Example



Soft EM for Mixture of Gaussians: Example



Soft EM for Mixture of Gaussians: Example

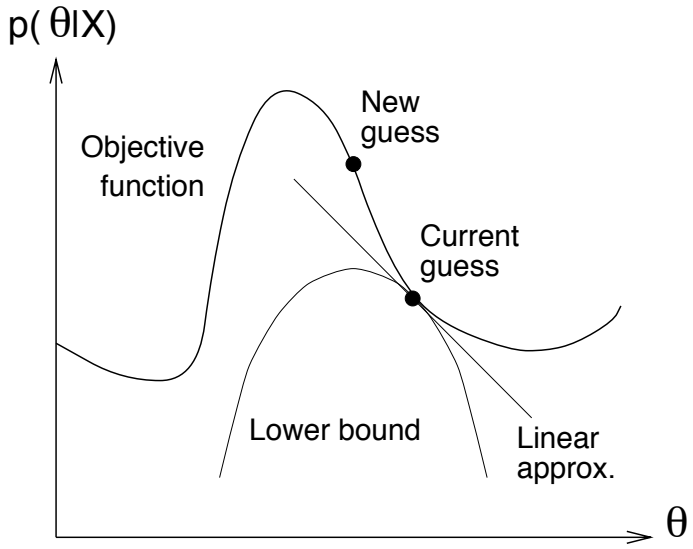


Note that some points still have "mixed" colour, indicating that they belong with significant probability to more than one component/cluster.

Comparison of hard EM and soft EM

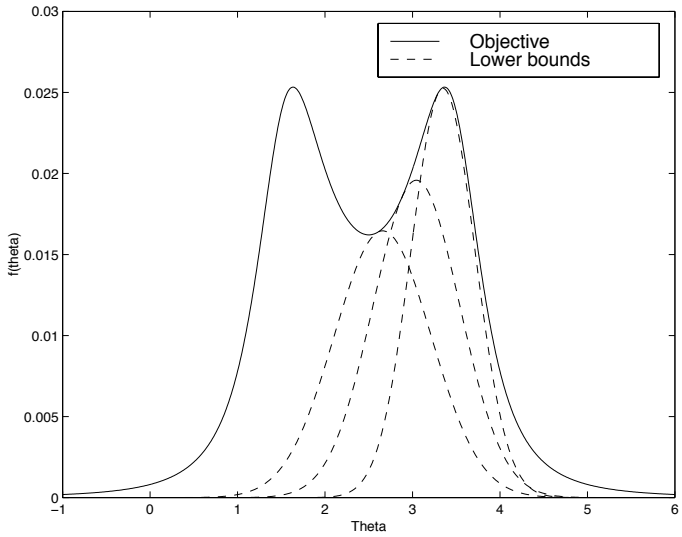
- Soft EM does not commit to a particular value of the missing item. Instead, it considers all possible values, with some probability
- This is a pleasing property, given the uncertainty in the value
- Soft EM is almost always the method of choice (and often when people say "EM", they mean the soft version)
- The complexity of each iteration of the two versions is pretty much the same.
- Soft EM might take more iterations, if we stop it based on having a change in the parameter values below a threshold

What EM does



- The algorithm works by making a *convex approximation to the log-likelihood* (by filling in the data)
- Note the difference to using a linear approximation of the objective function, like in gradient methods
- This approximation is a *lower bound* on the log-likelihood
- The maximization step maximizes the lower bound on the log-likelihood

Theoretical properties of EM



[1]

- Each iteration improves the likelihood of the data
  - Straightforward for Hard EM.
  - Less obvious for Soft EM (proved using Jensen's inequality)
- If the parameters do not change in one iteration, then the gradient of the log-likelihood function is zero
- Hence, EM converges to a locally optimal solution

[1] Minka, 1998 tutorial

Variations

- If only some of the data is incomplete, the likelihood will have one component based on the complete instances and another ones based on incomplete instances
- Sparse EM: Only compute probability at a few data points (most values will be close to 0 anyway)
- EM can be used if you have labeled data but which is possibly unreliable, in order to get better labels.
- Instead of a complete M-step, just improve the likelihood a bit
- Note that EM can be stuck in *local optima*, so it has to be restarted!
- It works very well for low-dimensional problems, but can have problems if θ is high-dimensional.

Summary of EM

- EM is guaranteed to converge to a *local optimum of the likelihood function*
- Can be used for virtually any application with missing data/latent variables (more on this later)
- Starting with different values of the initial parameters is necessary in order to get a good solution
- The algorithm can be stopped when no more improvement is achieved between iterations.
- A big hammer that fits all sorts of practical problems

Finding the number of components in a Gaussian Mixture Model

- The number of components controls the bias-variance trade-off: the more components, the smaller the bias, the larger the log-likelihood on training data, and the higher the variance / risk of overfitting
- Validation (using log-likelihood of a validation set) can be used to decide number of components
- Penalty/regularization methods (minimum description length) are also used to encourage small models
- Bayesian (parametric and non-parametric) methods can be used to put a prior on the number of components