Expectation-Maximization Algorithm for Clustering Multidimensional Numerical Data

Avinash Kak

Purdue University

March 3, 2024

3:19pm

An RVL Tutorial Presentation

(First presented in Summer 2012, updated Jan 2017, and reformatted in March 2024)

CONTENTS

Back to TOC

1: What Makes EM Magical?

- Despite the fact that EM can occasionally get stuck in a local maximum as you estimate the parameters by maximizing the log-likelihood of the observed data, in my mind there are three things that make it magical:
  - the ability to simultaneously optimize a large number of variables
  - the ability to find good estimates for any missing information in your data at the same time
  - and, in the context of clustering multidimensional data that lends itself to modeling by a Gaussian mixture, the ability to create both the traditional "hard" clusters and and not-so-traditional "soft" clusters.
- With regard to the ability of EM to simultaneously optimize a large number of variables, consider the case of clustering three-dimensional data:
  - Each Gaussian cluster in 3D space is characterized by the following 10 variables: the 6 unique elements of the 3×3 covariance matrix (which must be symmetric and positive-definite), the 3 unique elements of the mean, and the prior associated with the Gaussian.
  - Now let's say you expect to see six Gaussians in your data. What that means is that you would want the values for 59 variables (remember the unit-summation constraint on the class priors which reduces the overall number of variables by one) to be estimated by the algorithm that seeks to discover the clusters in your data.

  - What's amazing is that, despite the large number of variables that need to be optimized simultaneously, the chances are that the EM algorithm will give you a very good approximation to the correct answer.

- About EM returning both hard and soft clusters, by hard clusters I mean a disjoint partition of the data. This is normally what classifiers do. By soft clusters I mean allowing for a data point to belong to two or more clusters at the same time, the "level of membership" in a cluster being expressed by the posterior probabilities of the classes at the data point. (We will use the words "cluster" and "class" synonymously in this tutorial.)

Back to TOC

2: EM: The Core Notions

- EM is based on the following core ideas:
  - That there exists an analytic model for the data and that we know the functional form of the model. However, we do NOT know the values for the parameters that characterize this functional form).
  - We have a set of recorded data points in some multidimensional space, but some elements of the data are missing. (If you are mystified by this statement, do not worry. It will become clear shortly.)
  - We refer to the missing elements of the data as unobserved data.
  - While in some cases of estimation, it is easy to put your finger on what could be referred to as unobserved data, in others it can take some imagination — some other way of looking at your recorded data for you to be able to conceptualize the existence of unobserved data
  - Regardless of how you bring into play the unobserved data — whether due to the fact that you actually failed to record some of the data or whether your new way of looking at the data generation process brought into existence certain unobservables — the notion of unobserved data is central to a strict implementation of the EM algorithm.
  - Some folks refer to the unobserved data through the notion of hidden variables.
  - However, the problem with the terminology "hidden variables" is that it fails to capture the fact that some portions of the data may be missing because, say, your equipment failed to record them at the moment they became available. It's too much of a stretch of imagination to refer to such "failure to record" in terms of "hidden variables".
  - The notion of unobserved data is central to EM because that is what makes it possible to construct an iterative procedure for the maximization of the log-likelihood of the observed data.

- Obviously, we wish for EM to find the maximum-likelihood (ML) estimates for the parameters of the data model. The model parameters estimated by EM should be ML in the sense that they maximize the likelihood of all of the observed data.

- We also wish for EM to give us the best possible values (again in the most likelihood sense vis-a-vis all the observed data) for the unobserved data.

- Since folks new to EM have difficulty with the notion of unobserved data, the rest of this section presents two examples, one in which the unobserved data is literally so — that is, a part of the data that needed to be recorded was not recorded — and the other in which the unobserved data is a product of our imagination. The first example is by Duda, Hart, and Stork and the second based on a tutorial presentation of EM by Jeff Bilmes, " A Gentle Tutorial of the EM Algorithm and its Applications to Parameter Estimation for Gaussian Mixtures and Hidden Markov Models," Tech. Report, U. C. Berkeley.

Example 1 of Unobserved Data:

- Consider the case when the observed data consists of N points in a 2D plane.

- Let's say that we know *a priori* that a single bivariate Gaussian is a good model for the data. We only know the functional form of the model — we do NOT know the values for the parameters of this model.

- That is, if $\vec{\mathbf{x}}$ represents one element of the observed data, we can write

$$p(\vec{\mathbf{x}}) = \frac{1}{(2\pi)^{d/2}\,|\Sigma|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}-\vec{\mu})^T\Sigma^{-1}(\vec{\mathbf{x}}-\vec{\mu})} \tag{1}$$

where $d = 2$ and $|\Sigma|$ is the determinant of the 2×2 covariance matrix Σ. We think of $\vec{\mathbf{x}}$ as a 2-dimensional *column* vector. (The formula shown is for the general case of a d-dimensional $\vec{\mathbf{x}}$.)

– The yet-unknown mean of the observed data is represented by the 2-dimensional column vector $\vec{\mu}$.

– The yet-unknown covariance of the observed data represented by a positive-definite and symmetric 2×2 matrix Σ.

– We are therefore talking about 5 unknowns in the Gaussian model, of which three are for the symmetric 2×2 covariance matrix Σ and two for the mean vector $\vec{\mu}$.

– Given the data model as described above, let's say we are in possession of N observations, of which the last one is only partial. We consider an observation to be partial if only one of the two coordinates is known.

– Let's denote the $N-1$ complete observations by $\vec{\mathbf{x}}_1$, $\vec{\mathbf{x}}_2$, ... and $\vec{\mathbf{x}}_{N-1}$, and the last partial observation by $\vec{\mathbf{x}}_N^*$.

– The question here is: Can EM be used to estimate the parameters of the underlying Gaussian model, while at the same time, providing us with an estimate for the missing potion of the observation $\vec{\mathbf{x}}_N^*$?

Example 2 of Unobserved Data:

– Consider the following case: Our observed data can be modeled by a mixture of K Gaussians in which each Gaussian is given by

$$p(\vec{\mathbf{x}}) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}-\vec{\mu_i})^T \Sigma_i{}^{-1} (\vec{\mathbf{x}}-\vec{\mu_i})} \tag{2}$$

– In the above model, $|\Sigma_i|$ is the determinant of the $d \times d$ covariance matrix Σ_i for the i^{th} Gaussian, μ_i the mean of the same. We also associate a prior probability a_i with the i^{th} Gaussian with regard to its contribution to the mixture.

– Our goal is automatic clustering of the observations into disjoint clusters, which each cluster corresponding to a single Gaussian.

– The question here is whether EM can be used to estimate the class labels for the data elements, while, at the same time, estimating the means and the covariances of the individual Gaussians in the mixture.

– We obviously need to conceptualize the existence of unobserved data in this case. On the face of it, it is not clear as to what would constitute the unobserved data after we have recorded the N data points.

– As it turns out, we can conceptualize the needed unobserved data by thinking of the data generation process in a manner that allows a random variable to be associated with the selection of the Gaussian for each data point, as we next describe.

- We imagine the N data observations $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \ldots, \vec{\mathbf{x}}_N$ as having been generated sequentially through N different data generation events.

- Next, we bring into existence a sequence of N scalar random variables $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$ that correspond to the N observations $\mathcal{X} = \{\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \ldots, \vec{\mathbf{x}}_N\}$ on an index-by-index basis. The variable y_i will take on a random value from the set $\{1, 2, \ldots, K\}$, the value corresponding to the Gaussian that was chosen for the production of $\vec{\mathbf{x}}_i$.

- As shown in Section 4 of this tutorial, treating $\mathbf{Y} = \{y_1, y_2, \ldots, y_N\}$ as unobserved data allows us to use the EM algorithm for an iterative maximization of the log-likelihood for the data actually observed.

- In this case, it makes sense to refer to the unobserved data as the hidden variables in the estimation process.

- As mentioned earlier, the next section will present an example in which the unobserved data is literally so. Subsequently, in Section 4, we will talk about using EM for clustering Gaussian mixture data.

Back to TOC

3: An Example of EM Estimation in Which the Unobserved Data is Just the Missing Data

- This example is by Duda, Hart, and Stork (DHS) from their book "Pattern Classification," pages 126-128.

- My goal in using the DHS example is both to illustrate that the unobserved data can indeed be just the missing data, and to develop the notion of how the unobserved data facilitates the development of an iterative method for the maximization of the log-likelihood of the data actually observed.

- The observed data in this example will consist of four randomly produced points in a plane, with only the second coordinate available for the last point.

- The coordinate values for the four observed points are: $\vec{\mathbf{x}}_1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$, $\vec{\mathbf{x}}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\vec{\mathbf{x}}_3 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$, and $\vec{\mathbf{x}}_4 = \begin{pmatrix} * \\ 4 \end{pmatrix}$. Since the first coordinate of the last observation, $\vec{\mathbf{x}}_4$, is unknown, we use the symbol '*' for its value.

- We will denote the last observation $\vec{\mathbf{x}}_4 = \begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix}$, where the variable $x_{4,1}$ stands for the missing information in the data.

- So the problem is to estimate a value for $x_{4,1}$ that would be "consistent" — consistent in the maximum-likelihood sense — with the observed values for $\vec{\mathbf{x}}_1$, $\vec{\mathbf{x}}_2$, $\vec{\mathbf{x}}_3$, and for the $x_{4,2}$ coordinate of $\vec{\mathbf{x}}_4$.

- To keep the example simple, we assume the observed data can be modeled by a Gaussian with uncorrelated x and y coordinates.

- The Gaussian distribution for the data is given by

$$p(\vec{\mathbf{x}}) = \frac{1}{(2\pi)^{d/2}\,|\Sigma|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}-\vec{\mu})^T \Sigma^{-1}(\vec{\mathbf{x}}-\vec{\mu})} \tag{3}$$

  where $d = 2$, and with the covariance of this Gaussian given by

$$\Sigma \;=\; \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \tag{4}$$

- We will express the mean of the Gaussian in terms of its coordinates through:

$$\vec{\mu} \;=\; \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \tag{5}$$

- As the reader can see, there are four parameters, yet unknown, in the data model: σ_1^2, σ_2^2, μ_1 and μ_2. We will next talk about how these parameters can be estimated with EM.

- The EM algorithm requires us to iterate through the following two steps:

  1. **The Expectation Step:** Using the current best guess for the parameters of the data model, we construct an expression for the log-likelihood for all data, observed and unobserved, and, then, marginalize the expression with respect to the unobserved data. This expression will be shown to depend on both the current best guess for the model parameters and the model parameters treated as variables. [This sentence, undoubtedly confusing at its first reading, will become clear soon.]

  2. **The Maximization Step:** Given the expression resulting from the previous step, for the next guess we choose those values for the model parameters that maximize the expectation expression. These constitute our best new guess for the model parameters.

- The output of the Expectation Step codifies our expectation with regard to what model parameters are most consistent with the data actually observed and with the current guess for the parameters — provided we maximize the expression yielded by this step.

- We stop iterating through the two steps when any further change in the log-likelihood of the observed data falls below some small threshold.

- This brings us to the very important subject of the "marginalization" of the log-likelihood of all the data, observed and unobserved. By marginalization in the Expectation Step we mean integration of the log-likelihood for all data over all possibilities for the unobserved data.

- In order to give substance to the remark made at the bottom of the previous page, let's first write down an expression for log-likelihood for all data.

- Assuming the observations to have been made independently, ordinarily, the expression for the log-likelihood for the four data points would be

$$LL = \sum_{i=1}^{4} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) \tag{6}$$

where by the vector $\vec{\theta}$ we mean the model parameters:

$$\vec{\theta} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \sigma_1^2 \\ \sigma_2^2 \end{pmatrix} \tag{7}$$

- However, since we know that the last point, $\vec{\mathbf{x}}_4$, was observed only partially — that is, only the second coordinate of the last point was actually observed — we need to tease it out of the summation in Eq. (6) for special treatment later.

- So let's write the log-likelihood expression in the following form:

$$LL = \sum_{i=1}^{3} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) + \ln p(\vec{\mathbf{x}}_4|\vec{\theta}) \tag{8}$$

- As you will see later, in the Maximization Step of each iteration of EM, we would want to choose a value for the parameter vector $\vec{\theta}$ that maximizes the log-likelihood shown above. Although, in and of itself, that sounds straightforward, the reality of what needs to be maximized is a little bit more complex.

- We want the maximization of the log-likelihood to NOT be absolute in any sense, but to be with respect to the current guess for the model parameter vector $\vec{\theta}$. (If we could solve the absolute maximization of the log-likelihood problem, we would not need the EM algorithm.)

- To address the issue raised in the previous bullet, let's denote the current guess for the model parameters by $\vec{\theta}^g$. So the question then becomes as to how to "link-up" the log-likelihood expression shown in Equation (8) with $\vec{\theta}^g$.

- The value for the log-likelihood shown in Eq. (8) depends obviously on the data coordinate $x_{4,1}$, whose value we do not know. The best way to deal with this lack of knowledge about

$x_{4,1}$ is to average out the log-likelihood with respect to $x_{4,1}$.

- In a multi-variable scenario, averaging out an entity with respect to any single variable means carrying out a marginal integration of the entity with respect to the probability density function for the variable in question.

- The question then arises as to what density function to use for the variable $x_{4,1}$. **This is where the current best guess about the data model comes in.** Recall, we represent our current best guess for the data model by the parameter vector $\vec{\theta}^g$.

- Since the parameter vector $\vec{\theta}^g$ is for a model that includes two coordinates, in and of itself, this parameter vector does not apply directly to the scalar variable $x_{4,1}$.

- So the best we can do for the needed density function for $x_{4,1}$ at the moment is to express it generally as $p(x_{4,1}|\vec{\theta}^g, x_{4,2} = 4)$.

- Now we are ready to write the log-likelihood for all of the data observations, while taking into account the missing data coordinate $x_{4,1}$.

- As stated earlier, the new log-likelihood will be a *marginalization* of the original log-likelihood over the unobserved data element:

$$LL' = \int_{-\infty}^{\infty} \left\{ \sum_{i=1}^{3} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) \;+\; \ln p(\vec{\mathbf{x}}_4|\vec{\theta}) \right\} p(x_{4,1}|\vec{\theta^g}, x_{4,2} = 4)\; dx_{4,1} \tag{9}$$

  As you can see, this *marginalization* of the log-likelihood over $x_{4,1}$ is with respect to the current best guess $\vec{\theta^g}$ for the model parameters.

- Since the observations of the four data points in the 2D plane are independent of one another, the marginalization shown above with respect to the variable $x_{4,1}$ does not affect the contribution to the log-likelihood by $\vec{\mathbf{x}}_1$, $\vec{\mathbf{x}}_2$, and $\vec{\mathbf{x}}_3$.

- The $x_{4,1}$-marginalized log-likelihood shown in Eq. (9) can therefore be simplified to:

$$LL' = \sum_{i=1}^{3} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) \;+ \int_{-\infty}^{\infty} \Big(\ln p(\vec{\mathbf{x}}_4|\vec{\theta}) \Big)\; p(x_{4,1}|\vec{\theta^g}, x_{4,2} = 4)\;\; dx_{4,1} \tag{10}$$

- We will now use Bayes' Rule to simplify the integral on the right in Eq. (10) as shown next.

- Applying the Bayes' Rule to the second term of the integrand in Eq. (10):

$$
\begin{aligned}
&\int_{-\infty}^{\infty} \left(\ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta})\right) \;\; p(x_{4,1} \mid \vec{\theta}^g, x_{4,2}=4)\; dx_{4,1} \\
&\quad = \int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \;\; \frac{p(x_{4,1}, \vec{\theta}^g, x_{4,2}=4)}{p(\vec{\theta}^g, x_{4,2}=4)} \;\; dx_{4,1} \\
&\quad = \int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \;\; \frac{p\left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g\right) \cdot p(\vec{\theta}^g)}{p(x_{4,2}=4 \mid \vec{\theta}^g) \cdot p(\vec{\theta}^g)} \;\; dx_{4,1} \\
&\quad = \int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \;\; \frac{p\left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g\right)}{p(x_{4,2}=4|\vec{\theta}^g)} \; dx_{4,1} \\
&\quad = \int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \;\; \frac{p\left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g\right)}{\int_{\infty}^{\infty} p\left(\begin{pmatrix} x'_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g\right) dx'_{4,1}} \; dx_{4,1}
\end{aligned}
\tag{11}
$$

- The final expression shown above is easy to handle since all the probabilities are now described by the Gaussian model in Eq. (3).

- The $x_{4,1}$-marginalized log-likelihood shown in Eq. (10) can therefore be expressed as:

$$
LL' = \sum_{i=1}^{3} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) \;+\; \int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \;\; \frac{p\left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g\right)}{\int_{\infty}^{\infty} p\left(\begin{pmatrix} x'_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g\right) dx'_{4,1}} \; dx_{4,1} \tag{12}
$$

Notice that this is a function of both the current guess $\vec{\theta}^g$ for the model parameters, which basically is a set of constant

values, and the variables for the new guess in the vector $\vec{\theta}$.

- The result in Equation (12) is our expectation for the log-likelihood of the observed data as a function of the variables in $\vec{\theta}$.

- Now that we have a "general" expression for the log-likelihood expectation for the model parameters, **it's time to get to the business of applying the EM algorithm to the problem at hand — for both the purpose of estimating the model parameters and the missing** $x_{4,1}$.

- In the discussion that follows, we will refer to the Expectation Step as the E-Step and the Maximization Step as the M-Step.

- For the very first iteration through the two steps, we must make a reasonable random guess for the model parameters. We will choose

$$\vec{\theta}^g = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{13}$$

- In other words, we are choosing zero mean and unit variance as the initial guess for the model parameters.

- For the invocation of the E-Step in the first iteration of the EM algorithm, we plug the guess in Eq. (13) in Eq. (12) and we get

$$LL' = \sum_{i=1}^{3} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) \; + \; \frac{1}{D} \int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \; p\left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g \right) \; dx_{4,1} \tag{14}$$

  where the constant D stands for the denominator integral on the right hand side in Eq. (12). It is given by

$$\begin{aligned} D &= \int_{-\infty}^{\infty} p\left(\begin{pmatrix} x'_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \right) \; dx'_{4,1} \\ &= \int_{-\infty}^{\infty} \frac{1}{2\pi} e^{\frac{-1}{2}(x_{4,1}^2 + 4^2)} \; dx'_{4,1} \\ &= \frac{e^{-8}}{\sqrt{2\pi}} \end{aligned} \tag{15}$$

- We will now simplify the integral in Eq. (14) as follows:

$$\int_{-\infty}^{\infty} \ln p(\vec{\mathbf{x}}_4 \mid \vec{\theta}) \; p\left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \vec{\theta}^g \right) \; dx_{4,1}$$

$$
\begin{aligned}
&= \int_{-\infty}^{\infty} \ln p \left(\begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix} \;\middle|\; \begin{pmatrix} \mu_1 \\ \mu_2 \\ \sigma_1^2 \\ \sigma_2^2 \end{pmatrix} \right) \cdot \frac{1}{2\pi} e^{-\frac{1}{2}(x_{4,1}^2+4^2)} dx_{4,1} \\
&= \int_{-\infty}^{\infty} \ln \left(\frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{1}{2}\left((x_{4,1}-\mu_1)^2\sigma_1^2+(4-\mu_2)^2\sigma_2^2\right)} \right) \cdot \frac{1}{2\pi} e^{-\frac{1}{2}(x_{4,1}^2+4^2)} dx_{4,1} \\
&= \int_{-\infty}^{\infty} \left\{ \ln \left(\frac{1}{2\pi\sigma_1\sigma_2} \right) - \frac{1}{2}\left[(x_{4,1}-\mu_1)^2\sigma_1^2 + (4-\mu_2)^2\sigma_2^2 \right] \right\} \cdot \frac{1}{2\pi} e^{-\frac{1}{2}(x_{4,1}^2+4^2)} dx_{4,1} \\
&= \int_{-\infty}^{\infty} \ln \left(\frac{1}{2\pi\sigma_1\sigma_2} \right) \frac{1}{2\pi} e^{-\frac{1}{2}(x_{4,1}^2+4^2)} dx_{4,1} - \frac{1}{2} \int_{-\infty}^{\infty} \left[(x_{4,1}-\mu_1)^2\sigma_1^2 + (4-\mu_2)^2\sigma_2^2 \right] \cdot \frac{1}{2\pi} e^{-\frac{1}{2}(x_{4,1}^2+4^2)} dx_{4,1} \\
&= \ln \left(\frac{1}{2\pi\sigma_1\sigma_2} \right) \cdot \frac{e^{-8}}{\sqrt{2\pi}} \frac{1}{2} \left[(1+\mu_1^2)^2\sigma_1^2 \;+\; (16-8\mu_2+\mu_2^2)\sigma_2^2 \right] \cdot \frac{e^{-8}}{\sqrt{2\pi}}
\end{aligned}
\tag{16}
$$

- We used the following two properties of the Gaussian integrals in deriving the final expression shown in Eq. (16):

$$
\begin{aligned}
\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{z^2}{2}} dz &= 1 \\
\int_{-\infty}^{\infty} z^2 e^{-z^2} dz &= \sqrt{2\pi}
\end{aligned}
\tag{17}
$$

  where the second property is a simplification of the following result concerning Gaussian integrals (see Wikipedia page on "Gaussian integrals"):

$$
\int_0^{\infty} z^{2n} e^{-\frac{z^2}{a^2}} dz = \sqrt{\pi} \frac{(2n)!}{n!} \left(\frac{a}{2} \right)^{2n+1}
$$

- Substituting Eq. (16) in Eq. (14), we get for our marginalized log-likelihood:

$$LL' = \sum_{i=1}^{3} \ln p(\vec{\mathbf{x}}_i|\vec{\theta}) + \frac{1}{2}\left[(1+\mu_1^2)^2\sigma_1^2 + (16-8\mu_2+\mu_2^2)\sigma_2^2\right] - \ln(2\pi\sigma_1\sigma_2) \tag{18}$$

  That completes the E-Step in the first iteration of the EM algorithm.

- For the M-Step, we substitute in the first term of the log-likelihood expression of Eq. (18) the values of $p(\vec{\mathbf{x}})$ evaluated at $\vec{\mathbf{x}}_1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$, $\vec{\mathbf{x}}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and $\vec{\mathbf{x}}_3 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$. We then take the partial derivatives of the log-likelihood with respect to the parameters μ_1, μ_2, σ_1^2 and σ_2^2. Setting these partial derivatives to zero, we obtain the following values for the new guess according to the result shown on page 127 of Duda, Hart, and Stork:

$$\vec{\theta}^{new} = \begin{pmatrix} 0.75 \\ 2.0 \\ 0.918 \\ 2.0 \end{pmatrix} \tag{19}$$

- **This completes one iteration of the EM algorithm.**

- Duda, Hart, and Stork tell us that the EM algorithm converges in three iterations to the values shown below for μ_1, μ_2, σ_1^2, and σ_2^2:

$$\vec{\theta}^{new} = \begin{pmatrix} 1.0 \\ 2.0 \\ 0.667 \\ 2.0 \end{pmatrix} \tag{20}$$

- Subsequently, we may construct an ML estimate for the missing $x_{4,1}$ by substituting $\vec{\mathbf{x}} = \begin{pmatrix} x_{4,1} \\ 4 \end{pmatrix}$ and the values estimated by EM for the parameter $\vec{\mu}$ and Σ in Eq. (3), taking the natural log of the resulting expression to obtained the log-likelihood, and setting its derivative with respect to $x_{4,1}$ to 0. In our case, this returns the answer $x_{4,1} = 1.0$.

- The example presented in this section represents a strict interpretation of what is meant by an Expectation-Maximization algorithm. In general, it is possible to come up with "looser" interpretations in which we relax the condition that each new guess for the parameters being estimated be optimum (from the standpoint of optimizing the log-likelihood of the observed data).

- If we had to re-work the example presented so far in accordance with a looser interpretation of EM, after the initial guess $\vec{\theta}^g$, we would obtain an ML estimate for the missing $x_{4,1}$ in the manner described previously. That would constitute our E-Step. Subsequently, we would use the estimated value for $x_{4,1}$ to construct an ML estimate for the next guess $\vec{\theta}^{new}$, and so on.

We would continue the iterations as long as the log-likelihood of the actually observed data continues to increase.

- In both the strict and the loose interpretations (the loose interpretation is also referred to as the Generalized Expectation Maximization algorithm), the algorithm is guaranteed to converge to a local maximum of the log-likelihood of the observed data.

Back to TOC

4: EM for Clustering Data That Can be Modeled as a Gaussian Mixture

- Let's say we have observed the following N data points in a d-dimensional space:

$$\mathcal{X} \;=\; \{\vec{\mathbf{x}}_1,\; \vec{\mathbf{x}}_2,\; \ldots \vec{\mathbf{x}}_N\} \tag{21}$$

  We will assume that these N points are drawn from K Gaussian distributions, with the ℓ^{th} distribution characterized by the parameters $\theta_\ell = \{\vec{\mu}_\ell, \Sigma_\ell\}$, where μ_ℓ is the mean and Σ_ℓ the covariance. We also assume that the different Gaussian distributions do not carry equal weight with regard to their contributions to the observed data. We will represent this fact by associating a prior probability a_ℓ with the ℓ^{th} Gaussian. Obviously, $\sum_{\ell=1}^{K} a_\ell = 1$.

- We further assume that do not know which element of $\mathcal{X}$ was drawn from which of the Gaussians. However, we do know that *each element* of the dataset $\mathcal{X}$ is characterized by the following mixture probability density function:

$$p(\vec{\mathbf{x}} \mid \Theta) \;\;=\;\; \sum_{\ell=1}^{K} a_\ell \cdot p_\ell(\vec{\mathbf{x}} \mid \theta_\ell) \tag{22}$$

where, as mentioned above, a_ℓ is the prior associated with the ℓ^{th} Gaussian, and Θ represents all the parameters involved in the description of the mixture:

$$\Theta = (a_1, ..., a_K; \theta_1, ..., \theta_K) \tag{23}$$

As you would expect, the ℓ^{th} Gaussian in the mixture is given by

$$p_\ell(\vec{\mathbf{x}}|\theta_\ell) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_\ell|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}-\vec{\mu_\ell})^T \Sigma_\ell^{-1}(\vec{\mathbf{x}}-\vec{\mu_\ell})} \tag{24}$$

where $\theta_\ell = (\vec{\mu}_\ell, \Sigma_\ell)$ represents the parameters for just the ℓ^{th} Gaussian.

- If we assume that the N observations in $\mathcal{X}$ are independent, we can write the following expression for the probability distribution for all of the observations in $\mathcal{X}$:

$$\begin{aligned} p(\mathcal{X}|\Theta) &= \prod_{i=1}^{N} p(\vec{\mathbf{x}}_i \mid \Theta) \\ &= \prod_{i=1}^{N} \left(\sum_{\ell=1}^{K} a_\ell \cdot p_\ell(\vec{\mathbf{x}}_i \mid \theta_\ell) \right) \end{aligned} \tag{25}$$

- Substituting the individual Gaussians from Eq. (24) in Eq. (25), we can write for the probability distribution for all of our dataset:

$$p(\mathcal{X}|\Theta) \;=\; \prod_{i=1}^{N}\left(\sum_{\ell=1}^{K} a_\ell \cdot \frac{1}{(2\pi)^{d/2}\,|\Sigma_\ell|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})^T\Sigma_\ell^{-1}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})}\right) \tag{26}$$

- Focusing on Eq. (26), if we knew the parameter set Θ, then $p(\mathcal{X}|\Theta)$ is obviously a probability distribution for the dataset $\mathcal{X}$. However, if our goal is to estimate Θ from a given set of observations $\mathcal{X} = \{\vec{\mathbf{x}}_1,\ \vec{\mathbf{x}}_2,\ \ldots \vec{\mathbf{x}}_N\}$, then we prefer to think of the right hand side in Eq. (26) as the *likelihood* that tells us how likely the known observations in $\mathcal{X}$ are for candidate values for the elements of Θ. To make this fact more explicit, we rewrite Eq. (26) as

$$L(\Theta \mid \mathcal{X}) \;=\; \prod_{i=1}^{N}\left(\sum_{\ell=1}^{K} a_\ell \cdot \frac{1}{(2\pi)^{d/2}\,|\Sigma_\ell|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})^T\Sigma_\ell^{-1}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})}\right) \tag{27}$$

- Our goal is to construct a Maximum Likelihood estimate for Θ by seeking Θ^* that maximizes the log-likelihood:

$$\Theta^* \;=\; \underset{\Theta}{\text{argmax}} \quad \ln\left(L(\Theta \mid \mathcal{X})\right) \tag{28}$$

- Substituting Eq. (27) in Eq. (28), we get

$$\begin{aligned}
\Theta^* &= \underset{\Theta}{\text{argmax}} \ln\left[\prod_{i=1}^{N}\left(\sum_{\ell=1}^{K} a_\ell \cdot \frac{1}{(2\pi)^{d/2}\,|\Sigma_\ell|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})^T\Sigma_\ell^{-1}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})}\right)\right] \\
&= \underset{\Theta}{\text{argmax}} \sum_{i=1}^{N} \ln\left[\left(\sum_{\ell=1}^{K} a_\ell \cdot \frac{1}{(2\pi)^{d/2}\,|\Sigma_\ell|^{1/2}} e^{-\frac{1}{2}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})^T\Sigma_\ell^{-1}(\vec{\mathbf{x}}_i-\vec{\mu_\ell})}\right)\right]
\end{aligned} \tag{29}$$

- Looking at the expression shown on the right hand side above, our goal of finding a Θ that maximizes the log-likelihood is made difficult by the fact that we are dealing with the logarithm of a summation of exponentials. So the usual convenience afforded by the fact that the logarithm of an isolated Gaussian distributions reduces to a simple quadratic cannot help us here.

- This is where the EM algorithm comes to our rescue.

- In order to use EM, we obviously need to conceptualize the existence of unobserved data in this case. As mentioned earlier in the second half of Section 3, we can conceptualize the needed unobserved data by thinking of the data generation process in a manner that allows a random variable to be associated with the selection of the Gaussian for each data point. We imagine the N observations $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \ldots, \vec{\mathbf{x}}_N$ to be the results of N different data generation events.

- Next, we bring into existence a sequence of N scalar random variables $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$, with y_i corresponding to the data generation event that resulted in us making the observation $\vec{\mathbf{x}}_i$.

- Think of each y_i as a Gaussian-selector random variable. There are N of them and each such variable takes on a value from the set $\{1, 2, \ldots, K\}$.

- The value of the random variable y_i being $y_i = \ell$ implies that the i^{th} data element was generated by the Gaussian $p_\ell()$ whose mean is μ_ℓ and covariance Σ_ℓ.

- Treating $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$ as unobserved data allows us to use the EM algorithm for an iterative maximization of the log-likelihood for the data actually observed.

- While previously we wanted to estimate Θ by maximizing the likelihood $L(\Theta|\mathcal{X})$, we now want to subject the estimation of Θ to the maximization of $L(\Theta \mid \mathcal{X}, \mathcal{Y})$. We note:

$$\begin{aligned} L(\Theta \mid \mathcal{X}, \mathcal{Y}) &= p(\mathcal{X}, \mathcal{Y} \mid \Theta) \\ &= \prod_{i=1}^{N} p(\vec{\mathbf{x}}_i, y_i \mid \Theta) \\ &= \prod_{i=1}^{N} p(\vec{\mathbf{x}}_i \mid y_i, \Theta) \cdot p(y_i) \end{aligned} \tag{30}$$

- Note that we are using the notation $p()$ generically, in the sense that that $p()$ in Eq. (30) simply stands for the probability of its argument. So in the second term in Eq. (30), $p(y_i)$ stands for the probability that a Gaussian-selector random variable will take on the value y_i.

- Since the conditioning of the probability $p(\vec{\mathbf{x}}_i \mid y_i, \Theta)$ jointly on the Gaussian-selector random variable y_i and Θ is the same as

choosing just the parameters θ_{y_i} that correspond to the y_i^{th} Gaussian, $L(\Theta|\mathcal{X},\mathcal{Y})$ can be further simplified as follows

$$\begin{aligned} L(\Theta \mid \mathcal{X},\mathcal{Y}) &= \prod_{i=1}^{N} p_{y_i}(\vec{\mathbf{x}}_i \mid \theta_{y_i}) \cdot p(y_i) \\ &= \prod_{i=1}^{N} a_{y_i} \cdot p_{y_i}(\vec{\mathbf{x}}_i \mid \theta_{y_i}) \end{aligned} \tag{31}$$

- Eq. (31) makes explicit the fact that the likelihood for all data — both observed and unobserved — depends on the values taken on by the Gaussian-selector random variables, y_i's, for each of the data production events.

- Of course, in practice, we want to maximize the log-likelihood. So we rewrite the result in Eq. (31) in the following form:

$$LL = \sum_{i=1}^{N} \ln \Big(a_{y_i} \cdot p_{y_i}(\vec{\mathbf{x}}_i \mid \theta_{y_i}) \Big) \tag{32}$$

- At this point, the reader would be well served by reviewing Section 3 of this tutorial. In our discussion here, we are about to invoke one of the most central notions of EM — the marginalization of the log-likelihood of all data over the unobserved data — that was developed in that section.

- At this point in our derivation, we are at about the same point as where we were just prior to Eq. (9) in Section 3. If you understood the discussion in that section, you know that we next need to marginalize the log-likelihood shown in Eq. (32) with respect to the unobserved data $\mathcal{Y}$. This we do by writing:

$$LL' = \sum_{y_1=1}^{K} \ldots \sum_{y_N=1}^{K} \left(\sum_{i=1}^{N} \ln \left(a_{y_i} \cdot p_{y_i}(\vec{\mathbf{x}}_i \mid \theta_{y_i}) \right) \right) \cdot p(y_1 \ldots y_N \mid \Theta^g, \mathcal{X}) \qquad (33)$$

  where Θ^g represents the current guess for the parameters we need to estimate. Again using the notation $p()$ generically, the notation $p(y_1, \ldots, y_N)$ is for the joint probability of the N Gaussian-selector random variables taking on the values y_1 through y_N.

- The $\mathcal{Y}$-marginalized log-likelihood in Eq. (33) is obviously the **expectation** of the **observed-data log-likelihood** with respect to the hidden variables in $\mathcal{Y}$. On the other hand, Eq. (32) shows the complete-data log-likelihood. By this time you obviously know that by "complete data" I mean both the observed data $\mathcal{X}$ and the unobserved data $\mathcal{Y}$.

- With regard to the $y_1 \ldots y_N$-summations in Eq. (33), note the role played by the unknown distribution $p(y_1, \ldots, y_N)$. This distribution is obviously related to the Gaussian priors a_i's for the K Gaussians in the mixture. Just imagine the extreme case when, say, a_0 is 0.99 and all other $K-1$ a_i's are close to zero.

For such an extreme case, most of the N y_i's will be set to 0. This dependence on the K priors, a_i's, must somehow be reflected in the distribution $p(y_1, \ldots, y_N)$.

- Since our data generation model is based on the assumption that the choice of the Gaussian at each of the N data production events is made randomly (and independently of the other such choices), we can write

$$\begin{aligned} p(y_1 \ldots y_N \mid \Theta^g, \mathcal{X}) &= \prod_{i=1}^{N} p(y_i \mid \Theta^g, \mathcal{X}) \\ &= \prod_{i=1}^{N} p(y_i \mid \vec{\mathbf{x}}_i, \Theta^g) \end{aligned} \tag{34}$$

  where the second expression again is based on the independence of data generation at each of the N different data production events.

- Note that $p(y_i \mid \vec{\mathbf{x}}_i, \Theta^g)$ stands for

  the probability that a Gaussian selector has value y_i given that the observed data element has value $\vec{\mathbf{x}}_i$ and that the parameters of the mixture are set to Θ^g

- Using Bayes' Rule, the above probability can be expressed in the following form:

the probability that the observation is $\vec{\mathbf{x}}_i$ and the mixture parameters are Θ^g given that the Gaussian-selector has value y_i

multiplied by

the probability that the Gaussian-selector has value y_i

and divided by

the probability that the observation is $\vec{\mathbf{x}}_i$ and the mixture parameters are Θ^g

- Expressed mathematically, we can therefore write

$$\begin{aligned} p(y_i \mid \vec{\mathbf{x}}_i, \Theta^g) &= \frac{p_{y_i}(\vec{\mathbf{x}}_i, \Theta^g | y_i) \cdot a^g_{y_i}}{p(\vec{\mathbf{x}}_i, \Theta^g)} \\ &= \frac{p_{y_i}(\vec{\mathbf{x}}_i \mid \Theta^g, y_i) \cdot p(\Theta^g) \cdot a^g_{y_i}}{p(\vec{\mathbf{x}}_i | \Theta^g) \cdot p(\Theta^g)} \\ &= \frac{p_{y_i}(\vec{\mathbf{x}}_i | \theta^g_{y_i}) \cdot a^g_{y_i}}{p(\vec{\mathbf{x}}_i \mid \Theta^g)} \\ &= \frac{p_{y_i}(\vec{\mathbf{x}}_i | \theta^g_{y_i}) \cdot a^g_{y_i}}{\sum_{i=1}^{K} a^g_i p_i(\vec{\mathbf{x}}_i \mid \theta^g_i)} \end{aligned} \tag{35}$$

- We will now go back to Eq. (33) for the expected log-likelihood for the observed data and simplify it further. Substituting Eq. (34) in Eq. (33), we get

$$LL' = \sum_{y_1=1}^{K} \cdots \sum_{y_N=1}^{K} \left(\sum_{i=1}^{N} \ln \left(a_{y_i} \cdot p_{y_i}(x_i \mid \theta_{y_i}) \right) \right) \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) \tag{36}$$

- We will next isolate out the inner summation shown above, $\sum_{i=1}^{N} \ln \left(a_{y_i} \cdot p_{y_i}(x_i \mid \theta_{y_i}) \right)$, over the N observed-data generation events

from the summations over the variables in $\mathcal{Y}$ by first expressing, as shown below, the inner summation as a double summation with the help of the Kronecker delta function δ_{ℓ,y_i} (which equals 1 when $\ell = y_i$ and zero otherwise):

$$\sum_{i=1}^{N} \ln \Big(a_{y_i} \cdot p_{y_i}(x_i \mid \theta_{y_i}) \Big) \quad = \quad \sum_{\ell=1}^{K} \sum_{i=1}^{N} \delta_{\ell,y_i} \ln \Big(a_\ell \cdot p_\ell(x_i \mid \theta_\ell) \Big) \tag{37}$$

- Note that the above re-write of the inner summation in Eq. (36) merely expresses the choice that the Gaussian selector at the data generation event indexed i is set to ℓ.

- Substituting Eq. (37) in Eq. (36), we can write

$$LL' \quad = \quad \sum_{\ell=1}^{K} \sum_{i=1}^{N} \ln \Big(a_\ell \cdot p_\ell(x_i \mid \theta_\ell) \Big) \sum_{y_1=1}^{K} \ldots \sum_{y_N=1}^{K} \delta_{\ell,y_i} \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) \tag{38}$$

- We will now focus on the just the portion $\sum_{y_1=1}^{K} \ldots \sum_{y_N=1}^{K} \delta_{\ell,y_i} \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g)$ of the right hand side of Eq. (38) and show that it simplifies to just $p(\ell \mid \vec{\mathbf{x}}_i, \Theta^g)$, where $p(\ell \mid \vec{\mathbf{x}}_i, \Theta^g)$ is the probability that the value of the Gaussian-selector is ℓ given the observation $\vec{\mathbf{x}}_i$ and given that the mixture parameters are Θ^g.

- The portion $\sum_{y_1=1}^{K} \ldots \sum_{y_N=1}^{K} \delta_{\ell,y_i} \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g)$ of Eq. (38) can be simplified in the following manner:

$$\sum_{y_1=1}^{K} \cdots \sum_{y_N=1}^{K} \left(\delta_{\ell,y_i} \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) \right)$$

$$= \left(\sum_{y_1=1}^{K} \cdots \sum_{y_{i-1}=1}^{K} \sum_{y_{i+1}=1}^{K} \cdots \sum_{y_N=1}^{K} \prod_{j=1, j\neq i}^{N} p(y_j|\vec{\mathbf{x}}_j, \Theta^g) \right) \cdot p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \tag{39}$$

Note how the summation with respect to the variable y_i collapsed to just the term $p(\ell|\vec{\mathbf{x}}_i, \Theta^g)$ on account of δ_{ℓ,y_i}. This term is placed outside the large parentheses. So the summations are now, first, with respect to the variables y_1 through y_{i-1}, and then with respect to the variables y_{i+1} through y_N.

- Expressing the summations over the product as a product over a summation, we can re-express the result in Eq. (39) in the following form:

$$\sum_{y_1=1}^{K} \cdots \sum_{y_N=1}^{K} \left(\delta_{\ell,y_i} \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) \right)$$

$$= \prod_{j=1, j\neq i}^{N} \left(\sum_{y_j=1}^{K} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) \right) \cdot p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \tag{40}$$

- However, since $\sum_{y_j=1}^{K} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) = 1$, we end up with the simplification:

$$\sum_{y_1=1}^{K} \cdots \sum_{y_N=1}^{K} \left(\delta_{\ell,y_i} \cdot \prod_{j=1}^{N} p(y_j \mid \vec{\mathbf{x}}_j, \Theta^g) \right) = p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \tag{41}$$

- We substitute this result in Eq. (38) to obtain a much simpler form for the expected log-likelihood, as shown next.

- Substituting Eq. (41) in Eq. (38), we get for $\mathcal{Y}$-marginalized log-likelihood:

$$LL' = \sum_{\ell=1}^{K}\sum_{i=1}^{N} \ln\Big(a_\ell \cdot p_\ell(x_i \mid \theta_\ell)\Big) \cdot p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \tag{42}$$

  This is the expected value of the log-likelihood of the observed data $\mathcal{X}$, with the expectation having been carried out over the unobserved data $\mathcal{Y}$.

- The result shown above lends itself to the following further simplification:

$$LL' = \sum_{\ell=1}^{K}\sum_{i=1}^{N} \ln\big(a_\ell\big) \cdot p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \;+\; \sum_{\ell=1}^{K}\sum_{i=1}^{N} \ln\big(p_\ell(x_i \mid \theta_\ell)\big) \cdot p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \tag{43}$$

- The form shown in Eq. (43) is particularly convenient for the maximization of the log-likelihood because it separates out the contributions from the a_ℓ terms and those from the θ_ℓ terms. Recall our goal is to find the best values for a_ℓ and θ_ℓ for all ℓ from 1 through K since these constitute the parameters of our data model.

- That brings to an end the analytical work for the Expectation Step. We now have an expression for the expectation of the log-likelihood of the observed data $\mathcal{X}$, with the expectation having been carried out over the unobserved data $\mathcal{Y}$.

- Our next task obviously is the Maximization Step, which calls on us to find the model parameters that maximize the expected log-likelihood as expressed by Eq. (43).

- We will start with obtaining an update formula for a_ℓ's, the Gaussian priors. We want to obtain a_ℓ's through a maximization of the log-likelihood in Eq. (43) while keeping in the mind the constraint that $\sum_{\ell=1}^{K} a_\ell = 1$. This calls for the use of a Lagrange multiplier λ in the following equation:

$$\frac{\partial}{\partial a_{\ell'}}\Big[LL' \;+\; \lambda\Big(\sum_{\ell=1}^{K} a_\ell = 1\Big)\Big] \;\;=\;\; 0 \tag{44}$$

  for the estimation of the prior a_ℓ for $\ell = \ell'$.

- Substituting Eq. (43) in Eq. (44), we get:

$$\frac{\partial}{\partial a_{\ell'}}\left[\left(\sum_{\ell=1}^{K}\sum_{i=1}^{N} \ln(a_\ell)\cdot p(\ell|\vec{\mathbf{x}}_i, \Theta^g)\right) \;+\; \lambda\Big(\sum_{\ell=1}^{K} a_\ell = 1\Big)\right] \;\;=\;\; 0 \tag{45}$$

- The partial derivative with respect to $a_{\ell'}$ yields the following equation for each $\ell' = 1 \ldots K$:

$$\sum_{i=1}^{N} \frac{\partial}{\partial a_{\ell'}} \Bigg(\ln(a_{\ell'}) \cdot p(\ell' | \vec{\mathbf{x}}_i, \Theta^g) \Bigg) \; + \; \lambda \;\; = \;\; 0 \tag{46}$$

- Since $\frac{\partial \ln x}{\partial x} = \frac{1}{x}$, the K equations shown above reduce to

$$\sum_{i=1}^{N} p(\ell' | \vec{\mathbf{x}}_i, \Theta^g) \; + \; \lambda \cdot a_{\ell'} \;\; = \;\; 0 \tag{47}$$

- Summing both sides of the K equations shown above and recognizing that $\sum_{\ell'=1}^{K} p(\ell' | \vec{\mathbf{x}}_i, \Theta^g) = 1$ and $\sum_{\ell'=1}^{K} a_{\ell'} = 1$, we end up with

$$\lambda \;\; = \;\; -N \tag{48}$$

- Substituting the result in Eq. (48) in Eq. (47), we get following formula:

$$a_{\ell'} \;\; = \;\; \frac{1}{N} \sum_{i=1}^{N} p(\ell' | \vec{\mathbf{x}}_i, \Theta^g) \tag{49}$$

- Eq. (49) serves as a formula for EM-based updating of the values of the priors a_ℓ's. This is how the formula needs to be interpreted: Using the guess Θ^g for the mixture parameters (recall that Θ^g includes the guesses for the priors $a_\ell, \;\; \ell = 1 \ldots K$), calculate the posterior "class" probabilities $p(\ell | \vec{\mathbf{x}}_i, \Theta^g)$ at each of the data points $\vec{\mathbf{x}}_i, \, i = 1 \ldots N$ and then use the formula shown above to update the values for the priors. To emphasize the fact that the formula is to be used for updating the values of the priors, we re-express Eq. (49) as

$$a_\ell^{new} = \frac{1}{N}\sum_{i=1}^{N} p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \tag{50}$$

for $\ell = 1 \ldots K$.

- Note that we referred to $p(\ell|\vec{\mathbf{x}}_i, \Theta^g)$ as posterior class probabilities. This is in keeping with the traditional description of such probabilities in the pattern classification literature. We are evidently thinking of each Gaussian in the mixture as defining a class.

- That leaves us with having to develop the EM update formulas for the means and the covariances of the Gaussians.

- For both of these updates, we can ignore the first of the two terms on the right hand side in the log-likelihood formula in Eq. (43) because it does not involve the means and the covariances. Denoting the rest of the log-likelihood by LL'', we can write

$$\begin{aligned} LL'' &= \sum_{\ell=1}^{K}\sum_{i=1}^{N} \ln\Big(p_\ell(\vec{\mathbf{x}}_i \mid \theta_i)\Big) \cdot p\big(\ell \mid \vec{\mathbf{x}}_i, \Theta^g\big) \\ &= \sum_{\ell=1}^{K}\sum_{i=1}^{N} \left[\frac{-1}{2}(\vec{\mathbf{x}}_i - \vec{\mu_\ell})^T\Sigma_\ell^{-1}(\vec{\mathbf{x}}_i - \vec{\mu_\ell}) - \frac{\ln|\Sigma_\ell|}{2} + \ln\frac{1}{(2\pi)^{d/2}}\right] \\ &\qquad\qquad \cdot\ p\big(\ell \mid \vec{\mathbf{x}}_i, \Theta^g\big) \end{aligned} \tag{51}$$

- Taking the partial derivative of right hand side above with respect to $\vec{\mu}_{\ell'}$ and setting it to zero for the maximization of LL'' yields:

$$\sum_{i=1}^{N} \Sigma_{\ell'}^{-1} \cdot (\vec{\mathbf{x}}_i - \vec{\mu_{\ell'}}) \cdot p\big(\ell' \mid \vec{\mathbf{x}}_i, \Theta^g\big) \quad = \quad 0 \tag{52}$$

- Eq. (52), which must be true for all ℓ', $\ell' = 1 \ldots K$, follows from the fact that when a matrix A is square-symmetric, the partial derivative $\frac{\partial}{\partial \vec{\mathbf{x}}}$ of the quadratic $\frac{\partial \vec{\mathbf{x}}^T A \vec{\mathbf{x}}}{\partial \vec{\mathbf{x}}} = 2A\vec{\mathbf{x}}$.

- Eq. (52) gives the following formula for updating the means of the Gaussians:

$$\vec{\mu}_{\ell'} \quad = \quad \frac{\sum_{i=1}^{N} \vec{\mathbf{x}}_i \cdot p\big(\ell' \mid \vec{\mathbf{x}}_i, \Theta^g\big)}{\sum_{i=1}^{N} p\big(\ell' \mid \vec{\mathbf{x}}_i, \Theta^g\big)} \tag{53}$$

  for $\ell' = 1 \ldots K$.

- To make more explicit the fact that it is an update formula for the mean vectors, we can express it in the following form

$$\vec{\mu}_{\ell}^{new} \quad = \quad \frac{\sum_{i=1}^{N} \vec{\mathbf{x}}_i \cdot p\big(\ell \mid \vec{\mathbf{x}}_i, \Theta^g\big)}{\sum_{i=1}^{N} p\big(\ell \mid \vec{\mathbf{x}}_i, \Theta^g\big)} \tag{54}$$

  This formula tells us that, using the current guess Θ^g (which includes guesses for the means), we first estimate the posterior

class probabilities $p(\ell|\vec{\mathbf{x}}_i, \Theta^g)$ at each of the data points $\vec{\mathbf{x}}_i$, $i = 1 \ldots N$, and then use the formula shown above to update the means.

- That brings us to the derivation of an update formula for the covariances. As was the case for the means, we only need to maximize the second of the two terms in the summation in Eq. (43) for estimating the covariances.

- That is, we only need to maximize the expression for LL'' shown in Eq. (51). Ignoring the constant term inside the square brackets in Eq. (51), we now re-express LL'' as shown below

$$LL'' = \sum_{\ell=1}^{K} \sum_{i=1}^{N} \left[\frac{-1}{2} (\vec{\mathbf{x}}_i - \vec{\mu}_\ell)^T \Sigma_\ell^{-1} (\vec{\mathbf{x}}_i - \vec{\mu}_\ell) - \frac{\ln |\Sigma_\ell|}{2} \right] \cdot p\big(\ell \mid \vec{\mathbf{x}}_i, \Theta^g\big) \tag{55}$$

- Making use of the identity that $\vec{\mathbf{x}}^T A \vec{\mathbf{x}} = tr(A \vec{\mathbf{x}} \vec{\mathbf{x}}^T)$, where $tr(A)$ denotes the trace of a square matrix A, we can rewrite the above equation as

$$\begin{aligned} LL'' \;\; = \;\; \sum_{\ell=1}^{K} \Bigg[& \frac{1}{2} \ln \left(|\Sigma_\ell^{-1}| \right) \Big(\sum_{i=1}^{N} p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \Big) \;\; - \\ & \frac{1}{2} \sum_{i=1}^{N} p(\ell|\vec{\mathbf{x}}_i, \Theta^g) \cdot tr\Big(\Sigma_\ell^{-1} \cdot N_{\ell,i} \Big) \Bigg] \end{aligned} \tag{56}$$

where the $d \times d$ matrix $N_{\ell,i} = (\vec{\mathbf{x}}_i - \vec{\mu}_\ell)(\vec{\mathbf{x}}_i - \vec{\mu}_\ell)^T$. Recall that d is the dimensionality of our data space. In the first term above, we also made use of the identity $|A|^{-1} = |A^{-1}|$.

- Equation (56) is in a form that lends itself to differentiation with respect to $\Sigma_{\ell'}^{-1}, \ell' = 1 \ldots K$, for the maximization of LL''. We need to concern ourselves with two derivatives involving $\Sigma_{\ell'}^{-1}$: $\frac{\partial \ln(|\Sigma_{\ell'}^{-1}|)}{\partial \Sigma_{\ell'}^{-1}}$ and $\frac{\partial tr(|\Sigma_{\ell'}^{-1}|)}{\partial \Sigma_{\ell'}^{-1}}$. The first of these can be solved using the identity $\frac{\partial \ln|A|}{\partial A} = 2A^{-1} - diag(A^{-1})$ and the second by $\frac{\partial tr(AB)}{\partial A} = B + B^T - diag(B)$. Substituting these derivatives in Eq. (56) and setting the result to zero gives us the following equation:

$$\frac{1}{2}\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big)\cdot\Big(2\Sigma_{\ell'} - diag(\Sigma_{\ell'})\Big) \;-\; \frac{1}{2}\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big)\cdot\Big(2N_{\ell',i} - diag(N_{\ell',i})\Big) \;=\; 0 \tag{57}$$

- The above equation can be recast in the following form:

$$\frac{1}{2}\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big)\cdot\Big(2M_{\ell',i} \;-\; diag(M_{\ell',i})\Big) \;=\; 0 \tag{58}$$

  where $M_{\ell',i} = \Sigma_{\ell'} - N_{\ell',i}$ for $\ell' = 1 \ldots N$.

- The result in Eq. (58) is of the form

$$2S \;-\; diag(S) \;=\; 0 \tag{59}$$

  with

$$S \;=\; \frac{1}{2}\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big)\cdot M_{\ell',i} \tag{60}$$

- For the identity in Eq. (59) to hold, it must be case that $S = 0$, implying that

$$\frac{1}{2}\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big) \cdot M_{\ell',i} \;\; = \;\; 0 \tag{61}$$

  which yields the following solution

$$\Sigma_{\ell'} \;\; = \;\; \frac{\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big) \cdot N_{\ell',i}}{\sum_{i=1}^{N} p\big(\ell'|\vec{\mathbf{x}}_i, \Theta^g\big)} \tag{62}$$

  for $\ell' = 1 \ldots K$

- Substituting in Eq. (62) the value of $N_{\ell',i} = (\vec{\mathbf{x}}_i - \vec{\mu}_\ell)(\vec{\mathbf{x}}_i - \vec{\mu}_\ell)^T$ that was defined just after Eq. (56) and casting the result in the form of a update formula as we did earlier for the priors and means, we can write

$$\Sigma_\ell^{new} \;\; = \;\; \frac{\sum_{i=1}^{N} p\big(\ell|\vec{\mathbf{x}}_i, \Theta^g\big) \cdot (\vec{\mathbf{x}}_i - \vec{\mu}_\ell)(\vec{\mathbf{x}}_i - \vec{\mu}_\ell)^T}{\sum_{i=1}^{N} p\big(\ell|\vec{\mathbf{x}}_i, \Theta^g\big)} \tag{63}$$

  for $\ell = 1 \ldots K$.

- Since you are likely to update the covariances after you have updated the means, perhaps a more precise way to express this formula is:

$$\Sigma_\ell^{new} \;\; = \;\; \frac{\sum_{i=1}^{N} p\big(\ell|\vec{\mathbf{x}}_i, \Theta^g\big) \cdot (\vec{\mathbf{x}}_i - \vec{\mu}_\ell^{new})(\vec{\mathbf{x}}_i - \vec{\mu}_\ell^{new})^T}{\sum_{i=1}^{N} p\big(\ell|\vec{\mathbf{x}}_i, \Theta^g\big)} \tag{64}$$

for $\ell = 1 \ldots K$.

- As with the update formulas shown earlier in Eqs. (49) and (54), the update formula in Eq. (64) tells us that, using the guessed parameter values in Θ^g (this guess obviously includes values for Σ_ℓ, $\ell = 1 \ldots K$), we first calculate the posterior "class" probabilities at each of the data points $\vec{\mathbf{x}}_i$. Subsequently, using Eq. (64), we update the covariances for each of the Gaussians in the mixture.

- The three update formulas in Eqs. (49), (54), and (64) all require us to the compute using the current guess the posterior class probabilities at each of the data points. This we can do with the help of Bayes' Rule as follows, as shown below.

- The posterior class probabilities at each data point $\vec{\mathbf{x}}_i$, $i = 1 \ldots N$, is given by

$$\begin{aligned} p(\ell|\vec{\mathbf{x}}_i, \Theta^g) &= \frac{p(\vec{\mathbf{x}}_i|\ell, \Theta^g) \cdot p(\ell|\Theta^g)}{p(\ell, \vec{\mathbf{x}}_i, \Theta^g)} \\ &= \frac{p(\vec{\mathbf{x}}_i|\ell, \theta_\ell) \cdot a_\ell}{p(\ell, \vec{\mathbf{x}}_i, \Theta^g)} \\ &= \frac{p(\vec{\mathbf{x}}_i|\ell, \theta_\ell) \cdot a_\ell}{numerator\ normalizer} \end{aligned} \tag{65}$$

- Expressing the denominator as *numerator normalizer* is meant to convey the very important point that, from a computational perspective, there is never a need to explicitly calculate the probabilities $p(\ell, \vec{\mathbf{x}}_i, \Theta^g)$. After we have calculated the numerators for all $p(\ell|\vec{\mathbf{x}}_i, \Theta^g)$ on the left hand side above, we estimate the denominator by insisting that $\sum_{\ell=1}^{K} p(\ell, \vec{\mathbf{x}}_i, \Theta^g) = 1$.

Back to TOC

5: Algorithm::ExpectationMaximization — a Perl Module

- The goal of this section is to introduce you to my Perl module `Algorithm::ExpectationMaximization` for the clustering of multidimensional numerical data that can be modeled as a Gaussian mixture. This module can be downloaded from (all in one line):

  `http://search.cpan.org/~avikak/Algorithm-ExpectationMaximization/lib/Algorithm/ExpectationMaximization.pm`

- If unable to directly click on the URL shown above or this URL is difficult to copy and paste in your browser window, you can also reach the module by carrying out a Google search on a string like "Avi Kak EM Algorithm". Make sure you have reached the CPAN open-source archive and that you have Version 1.22 of the module.

- **IMPORTANT:** if you are NOT a Perl programmer, the easiest way for you to use this module for data clustering would be to use one of the canned scripts in the `examples` directory of the module. Section 6 presents a catalog of these scripts. All you would need to do would be to modify one of these scripts to suit your needs. The rest of this section is for those who would like

to write their own Perl scripts for data clustering using this module.

- The module expects that the data that needs to be clustered to be made available through a text file whose contents should look like:

```
c20   9    10.7087017   9.6352838    10.951215   ...
c7    23   12.8025925   10.6126270   10.522848   ...
b9    6    7.6011820    5.0588924    5.828417    ...
....
....
```

- Your file is allowed to have as many columns as you wish, but one of the columns must contain a symbolic tag for each data record. In the above example, the entries in the first column, whose column index is 0, are the symbolic tags.

- You must inform the module as to which column contains the tag and which columns to use for clustering. This you do by defining a mask variable an example of which is shown below:

```
my $mask = "N0111";
```

for the case when your data file has five columns in it, the tag for each record is in the first column, and you want only the **last** three columns to be used for clustering. The position of the character 'N' corresponds to the column with the data tags.

In this case, since 'N' is at the position of column index 0, that's the column with the symbolic tags. An entry of 0 in the mask means to NOT use that column data for clustering.

- After you have set up your data file in the manner described above, you need to create an instance of the module in your own Perl script. This you do by invoking the module constructor `new()` as shown below. The example call shown below is for the case when you expect to see 3 clusters, you want cluster seeding to be random, and you want to set an upper limit of 300 on EM iterations:

```
my $clusterer = Algorithm::ExpectationMaximization->new(
                            datafile            => $datafile,
                            mask                => $mask,
                            K                   => 3,
                            max_em_iterations   => 300,
                            seeding             => 'random',
                            terminal_output     => 1,
                            debug               => 0,
                );
```

- The choice `random` for cluster seeding in the call to the constructor shown above means that the clusterer will randomly select `K` data points to serve as initial cluster centers.

- Other possible choices for the constructor parameter `seeding` are `kmeans` and `manual`. With the `kmeans` option for `seeding`,

the output of a K-means clusterer is used for the cluster seeds and the initial cluster covariances. If you use the `manual` option for seeding, you must also specify the data elements to use for seeding the clusters. See the CPAN documentation page for the module for an example of manual seeding.

- After the invocation of the constructor, the following calls are mandatory for reasons that should be obvious from the names of the methods:

```
$clusterer->read_data_from_file();
srand(time);
$clusterer->seed_the_clusters();
$clusterer->EM();
$clusterer->run_bayes_classifier();
my $clusters = $clusterer->return_disjoint_clusters();
```

- In the sequence of mandatory calls shown in the previous bullet, it is the call to `EM()` that invokes the Expectation-Maximization algorithm for the clustering of data using the three update formulas presented in Section 4.

- The call to `srand(time)` is to seed the pseudo random number generator afresh for each run of the cluster seeding procedure. If you want to see repeatable results from one run to another of the algorithm with random seeding, you would obviously not invoke `srand(time)`.

- The call `run_bayes_classifier()` shown previously as one of the mandatory calls carries out a disjoint clustering of all the data points using the naive Bayes' classifier.

- After you have run `run_bayes_classifier()`, a call to `return_disjoint_clusters()` returns the clusters thus formed to you. Once you have obtained access to the clusters in this manner, you can display them in your terminal window by

```
foreach my $index (0..@$clusters-1) {
  print ``Cluster $index (Naive Bayes):  \
                                    @{$clusters->[$index]}\n\n''
}
```

- If you would like to also see the clusters purely on the basis of the posterior class probabilities exceeding a threshold `$theta1`, you call

```
my $theta1 = 0.5;
my $posterior_prob_clusters =
  $clusterer->return_clusters_with_posterior_\
                              probs_above_threshold($theta1);
```

  where you can obviously set the threshold `$theta1` to any value you wish.

- When you cluster the data with a call to `return_clusters_with_posterior_probs_above_threshold($theta1)` as shown

in the previous bullet, in general you will end up with clusters that overlap. You can display them in your terminal window in the same manner as shown previously for the naive Bayes' clusters.

- You can write the naive Bayes' clusters out to files, one cluster per file, by calling

```
$clusterer->write_naive_bayes_clusters_to_files();
```

  The clusters are placed in files with names like

```
naive_bayes_cluster1.dat
naive_bayes_cluster2.dat
...
```

- You can write out the posterior-probability based clusters to files by calling:

```
$clusterer->write_posterior_prob_clusters_above_\
                                    threshold_to_files($theta1);
```

- The threshold `$theta1` in the call shown in the previous bullet sets the probability threshold for deciding which data elements to place in a cluster. The clusters themselves are placed in files with names like

```
posterior_prob_cluster1.dat
posterior_prob_cluster2.dat
...
```

- **The module allows you to visualize the clusters when you do clustering in 2D and 3D spaces.**

- In order to visualize the output of clustering, you must first set the mask for cluster visualization. This mask tells the module which 2D or 3D subspace of the original data space you wish to visualize the clusters in.

- After you have decided on a mask, visualization typically involves making the following calls:

```
my $visualization_mask = ''111'';
$clusterer->visualize_clusters($visualization_mask);
$clusterer->visualize_distributions($visualization_mask);
```

  where the first call for the visualization of naive Bayes' clusters and the second for the visualization of posterior-probability based clusters.

- When clustering in 2D or 3D spaces, you can also directly create image PNG files that correspond to terminal visualization of the output of clustering:

```
$clusterer->plot_hardcopy_clusters($visualization_mask);
$clusterer->plot_hardcopy_distributions($visualization_mask);
```

- The PNG image of the posterior probability distributions is written out to a file named `posterior_prob_plot.png` and

the PNG image of the disjoint Naive Bayes' clusters to a file called `cluster_plot.png`.

- **The module also contains facilities for synthetic data generation for experimenting with EM based clustering.**

- The data generation is controlled by the contents of a parameter file that is supplied as an argument to the data generator method of the module. The priors, the means, and the covariance matrices in the parameter file must be according to the syntax shown in the `param1.txt` file in the `examples` directory. It is best to edit a copy of this file for your synthetic data generation needs.

- Here is an example of a sequence of calls that you would use for generating synthetic data:

```
my $parameter_file = ``param1.txt'';
my $out_datafile = ``mydatafile1.dat'';
Algorithm::ExpectationMaximization->cluster_data_generator(
                      input_parameter_file => $parameter_file,
                      output_datafile => $out_datafile,
                      total_number_of_data_points => $N );
```

  where the value of `$N` is the total number of data points you would like to see generated for all of the Gaussians. How this total number is divided up amongst the Gaussians is decided by

the prior probabilities for the Gaussian components as declared in input parameter file.

- The synthetic data may be visualized in a terminal window and the visualization written out as a PNG image to a disk file by

```
my $data_visualization_mask = ‘‘11’’;
$clusterer->visualize_data($data_visualization_mask);
$clusterer->plot_hardcopy_data($data_visualization_mask);
```

Back to TOC

6: Convenience Scripts in the `examples` Directory of the Module Algorithm::ExpectationMaximization

- Even if you are not a Perl programmer, you can use the module `Algorithm::ExpectationMaximization` for clustering your data through the convenience scripts that you will find in the **`examples`** directory of the module. You would just need to edit one of the scripts to suit your needs and then all you have to do is to execute the script.

- And even if you are a Perl programmer, becoming familiar with the scripts in the **`examples`** directory is possibly the best strategy for becoming familiar with this module (and its future versions).

- The rest of this section presents a brief introduction to the five scripts in the **`examples`** directory:

  - **canned_example1.pl**:
    The goal of this script is to show EM-based clustering of overlapping clusters starting with randomly selected seeds. As programmed, this script clusters the data in the datafile `mydatafile.dat`. The mixture data in the file corresponds to three overlapping Gaussian components in a star-shaped pattern.

– **canned_example2.pl**:
This goal of this script is to use the output of the K-Means clusterer to serve as seeds for the EM-based clusterer. The data fed to this script consists of two well separated blobs.

– **canned_example3.pl**:
This script gives a demonstration of how you would structure a call to the module constructor when you want to specify the cluster seeds manually.

– **canned_example4.pl**:
Whereas the three previous scripts demonstrate EM based clustering of 2D data, this script uses the module to cluster 3D data. This script is meant to demonstrate how the EM algorithm works on well-separated but highly anisotropic clusters in 3D.

– **canned_example5.pl**:
This script also demonstrates clustering in 3D but now we have one Gaussian cluster that "cuts" through the other two Gaussian clusters.

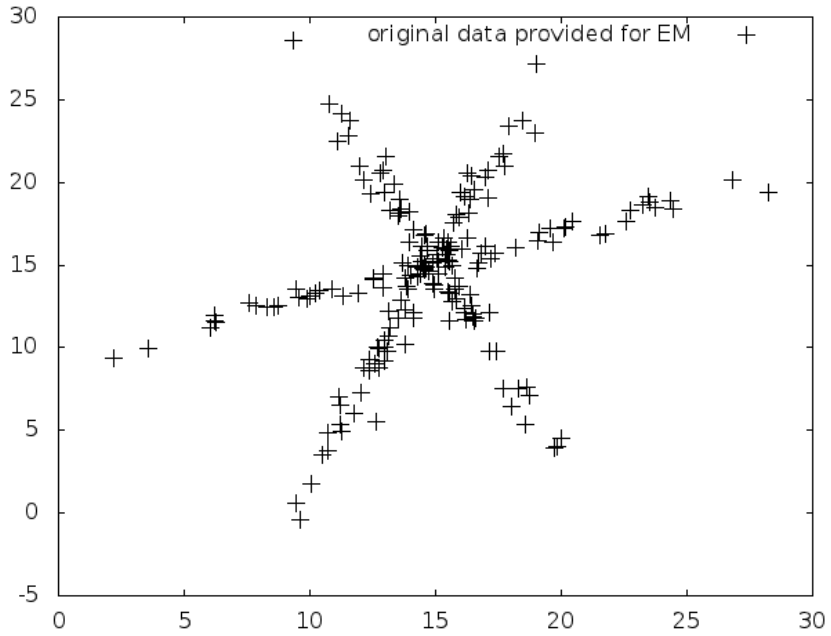
Back to TOC

7: Some Clustering Results Obtained with `Algorithm::ExpectationMaximization`

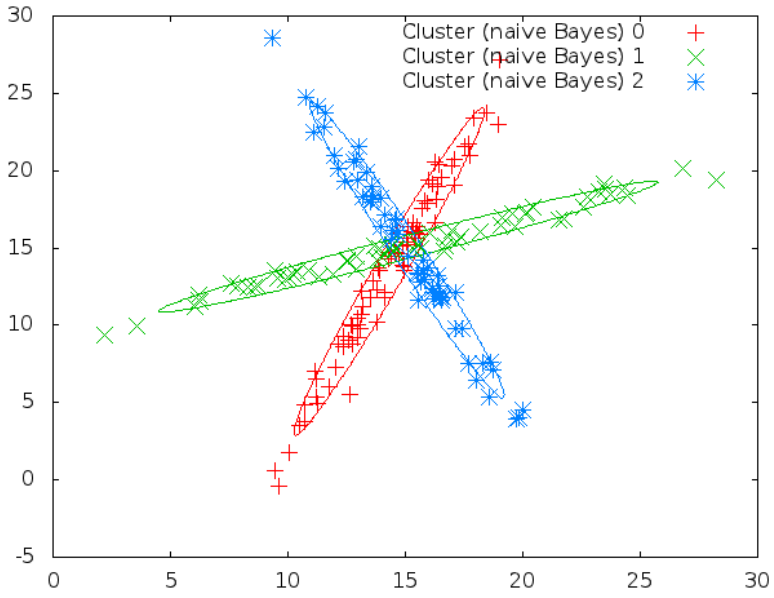
This section presents some results obtained with the `Algorithm::ExpectationMaximization` module. We will show five different results obtained with the five `canned_example` scripts in the **`examples`** directory.

Results Produced by the Script `canned_example1.pl`:

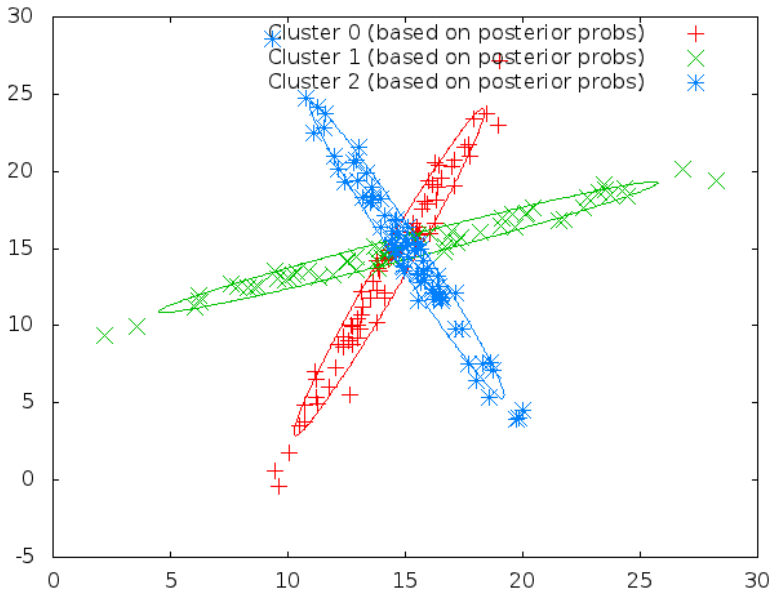
Shown next is a 2D scatter plot that consists of three overlapping Gaussian clusters that was fed into the script `canned_example1.pl`. As the reader will recall from the previous section, this canned script uses random seeding for the initialization of the cluster centers.



When you execute the script `canned_example1.pl` using the data shown in the scatter plot in the previous figure, you will get two types of clusters: the disjoint Naive Bayes' clusters and the clusters based on the posterior class probabilities exceeding a specified threshold. Shown in the next figure are the three Naive Bayes' clusters as produced by the call to `canned_example1.pl`.



And for the same input data, shown in the next figure are the three posterior-probability based clusters produced by the module.

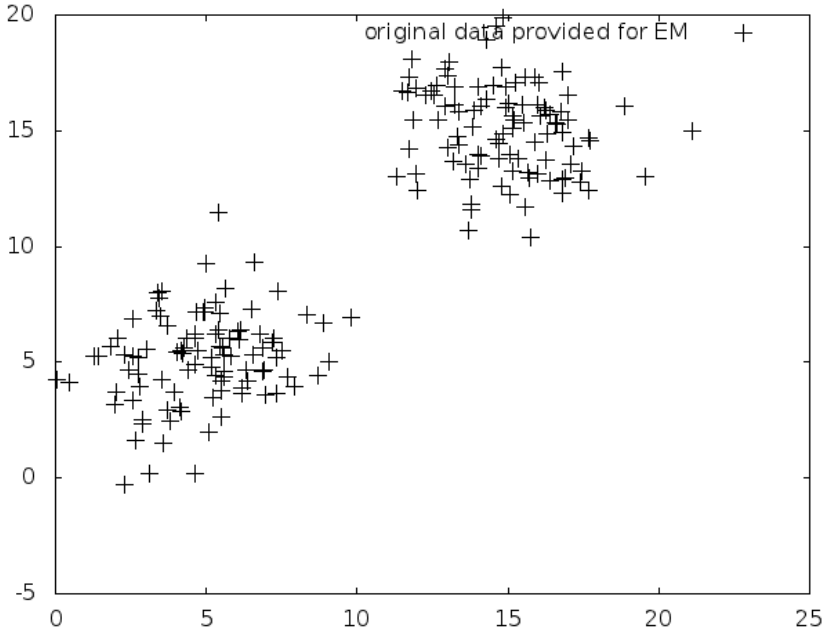


For the Naive Bayes' clusters shown at the top on the previous page, each data point gets a single label (a single color in our case). On the other hand, for the posterior-probability clusters shown above, it is possible for a single pixel to acquire multiple class labels. In trying to visually discern the color identities of the pixels in the figure on this page, note that when a color is assigned multiple colors, it is likely that the color one actually sees at the pixel is the last color that was applied to the pixel. It is important to mention that this ambiguity would exist only in the visualization of the clusters. On the other hand, when you actually write out the clusters to disk files, you can see all the pixels in each cluster.

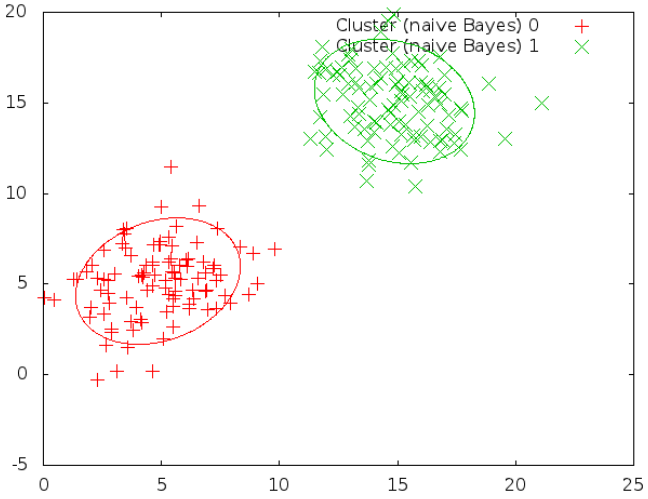
Results Produced by the Script `canned_example2.pl`:

I'll now show results obtained with the script `canned_example2.pl` with K-Means based seeding for the initialization of the cluster centers and the cluster covariances. The figure shown below is a scatter

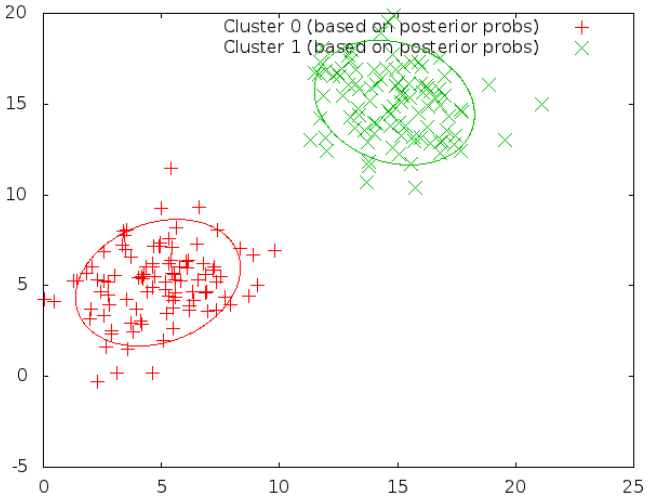
plot of the data that we will feed into `canned_example2.pl`.



When we run `canned_example2.pl`, as with the previous example, we end with two different clustering results (which in this case look identical because the clusters are so well separated). One of these is for Naive Bayes' clustering, which we show in the figure below:



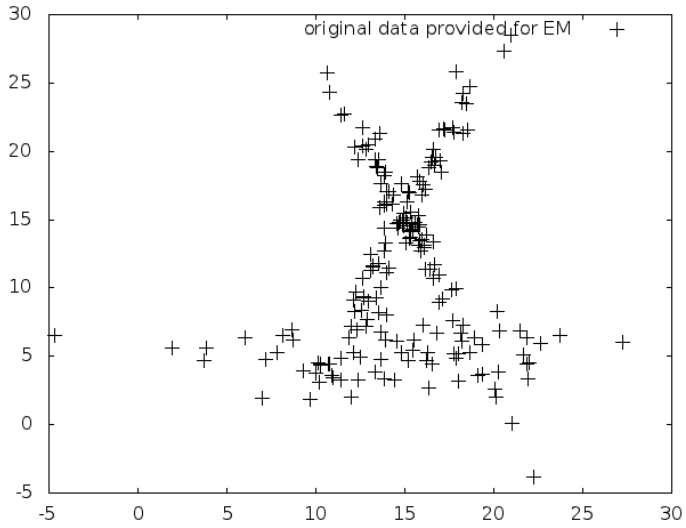
And the other is clustering on the basis of the posterior probabilities exceeding a given threshold. We show this result in the figure shown at the top on the next figure when the threshold is set to 0.2.



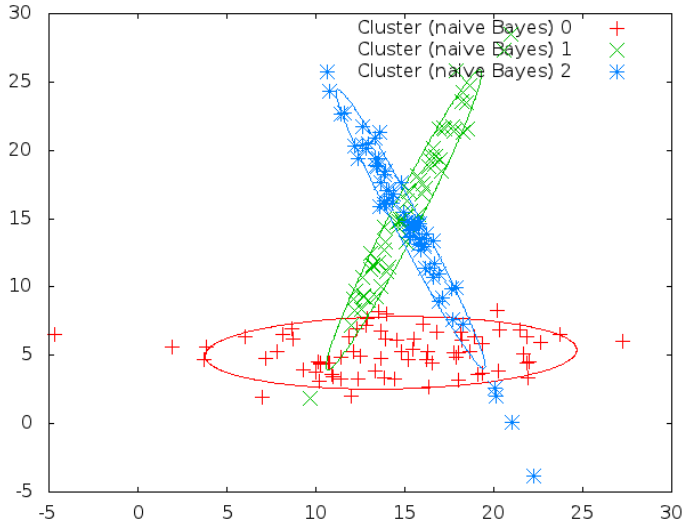
As is to be expected for this example, since the two clusters are so widely separated, the Naive Bayes' clusters and the posterior-probability based clusters here are identical.

Results Produced by the Script `canned_example3.pl`:

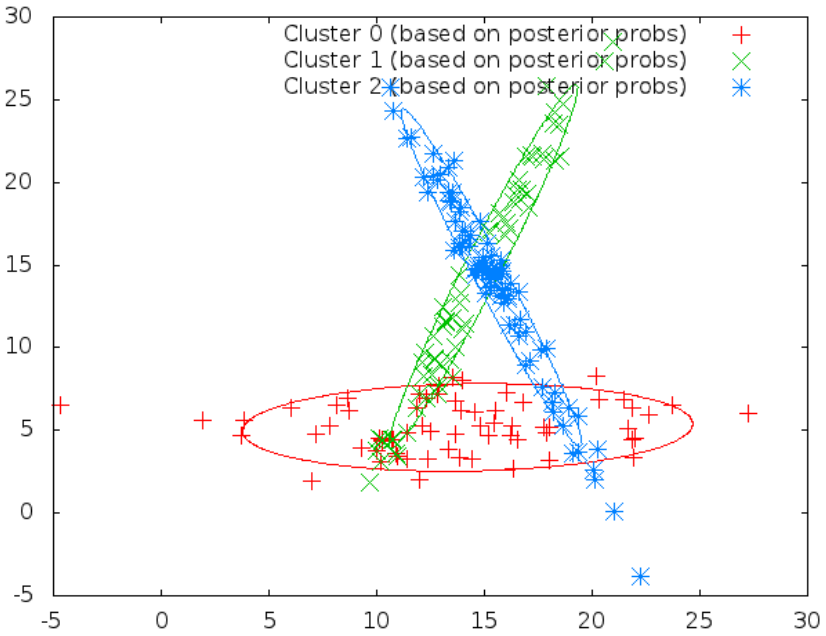
Our next demonstration involves manual seeding of the clusters for the data that is displayed in the scatter plot figure shown next. The data consists of three overlapping clusters, with the cluster at the bottom cutting across the other two.



When you execute `canned_example3.pl`, you'll again get two outputs for the two different types of clusters the module outputs. Shown below are the Naive Bayes' clusters for this data:



And shown below are the clusters based on the posterior class probabilities exceeding 0.2 threshold.



In the clusters shown above, if you look carefully at some of the data points in the overlap regions, you will notice that some of the pixels have more than one colored marker, illustrating the notion of soft clustering that is achieved with posterior class probabilities.

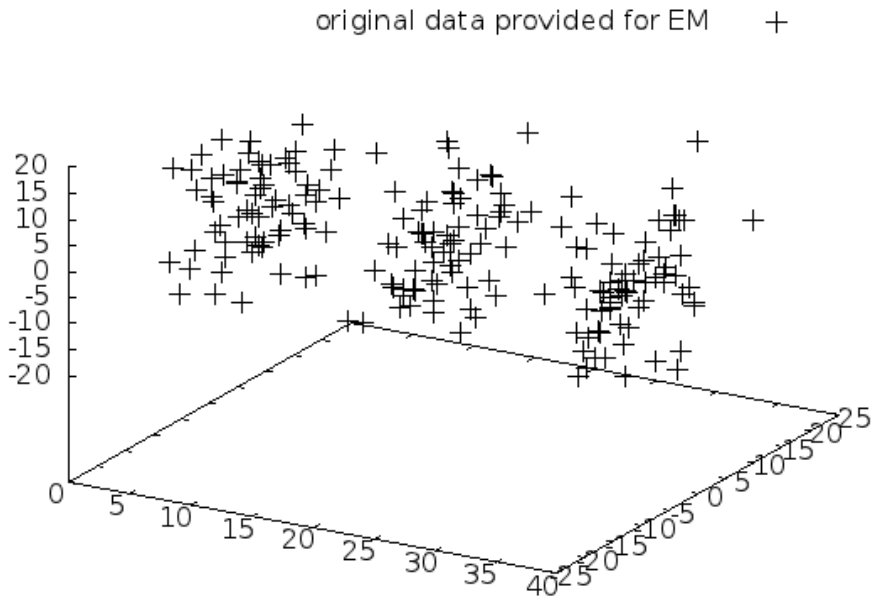
Results Produced by the Script `canned_example4.pl`:

The three clustering examples we have presented so far have all dealt with 2D data. Now I'll show some results of clustering 3D data. When clustering is carried out in 3D or a larger number of dimensions, visualization of result of clustering becomes a bigger challenge — especially when you show the visualizations in hardcopy as we do here. Talking about 3D, when you visualize the clusters on a computer terminal, you can at least rotate the figure and look at the clusters from different viewpoints to gain a better sense of how good the clusters look. In hardcopy, as is the case with

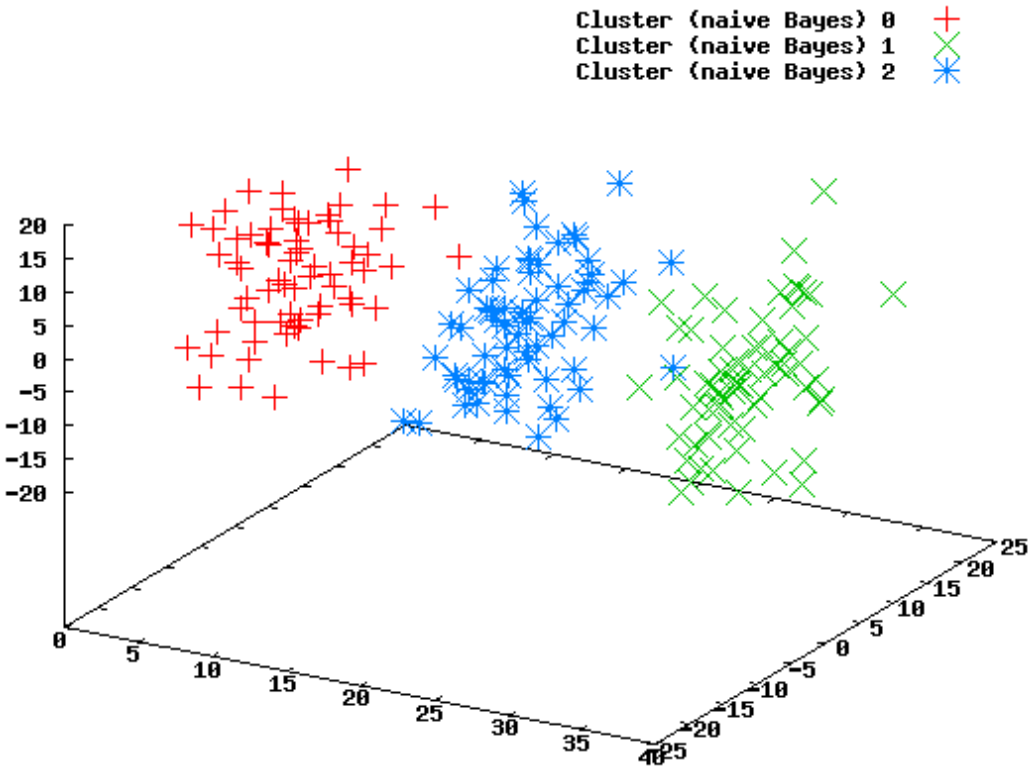
us here, all you can do is to show the results from one viewpoint.

Shown in the next figure is a 3D scatter-plot of the data for the script `canned_example4.pl`.

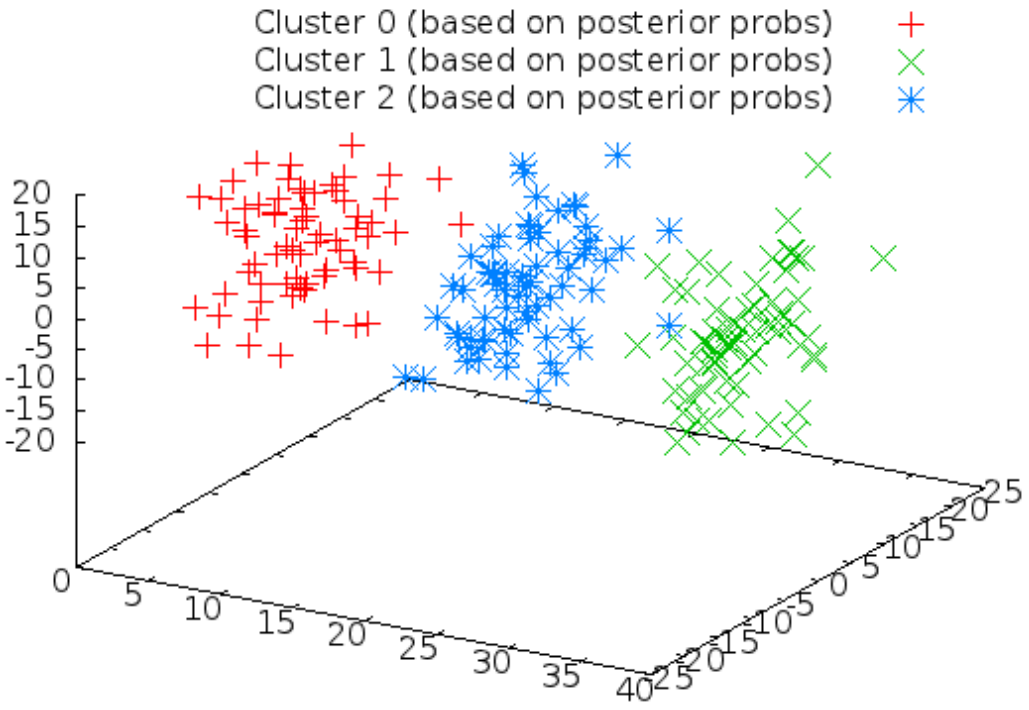
As you can see, in this case the data consists of three isolated clusters. We will ask the script `canned_example4.pl` to cluster this data with random seeding for initialization.



When you run the script, you will again two clustering outputs, one for the Naive Bayes' clusters and the posterior-probability based clusters. Shown at the top on the next figure are the Naive Bayes' clusters for this data.



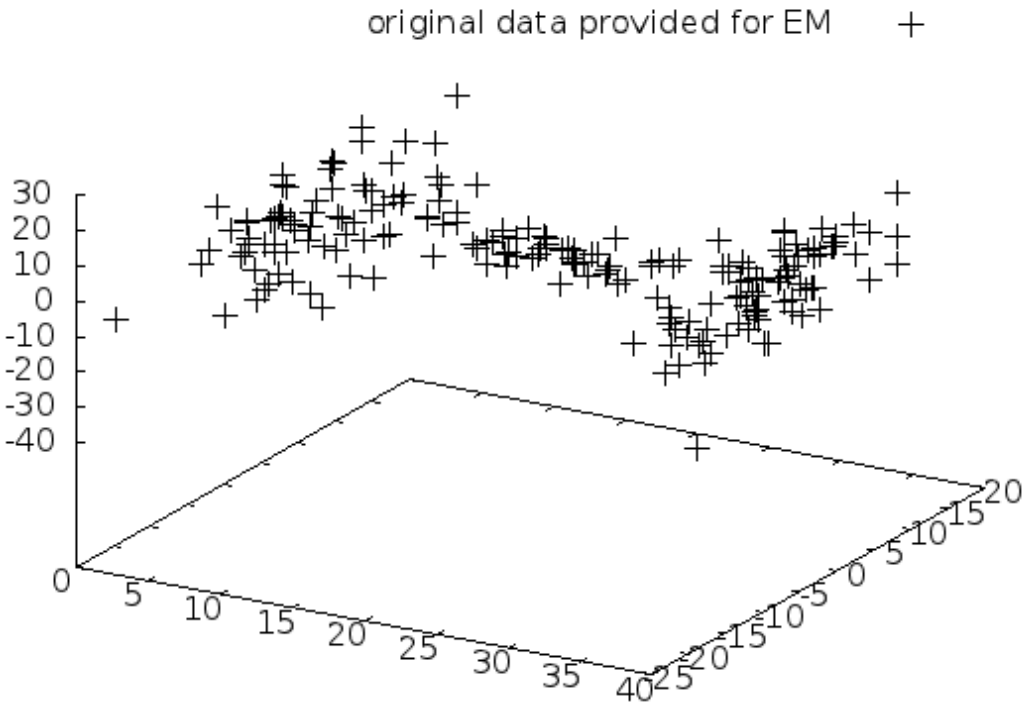
And shown in the next figure are the clusters based on the posterior class probabilities exceeding 0.2 threshold.



Both the Naive Bayes' clusters and the posterior-probability-based clusters are identical in this case, as you would expect, since the clusters are so well separated.

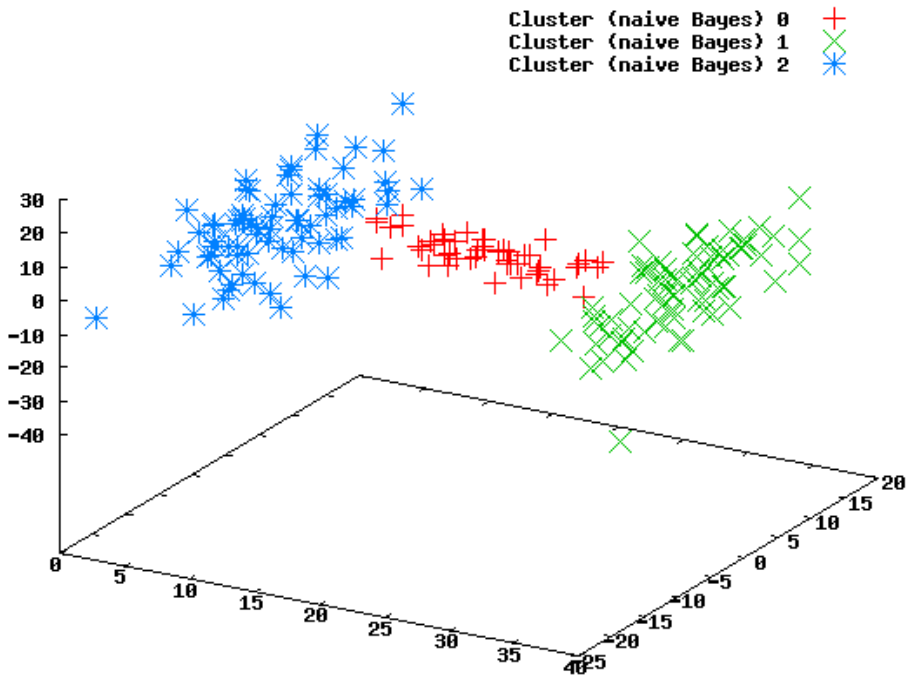
Results Produced by the Script `canned_example5.pl`:

That brings us to the final example of this section. While the previous example showed clustering of 3D data consisting of well separated clusters, now let's take up the case of overlapping 3D clusters.

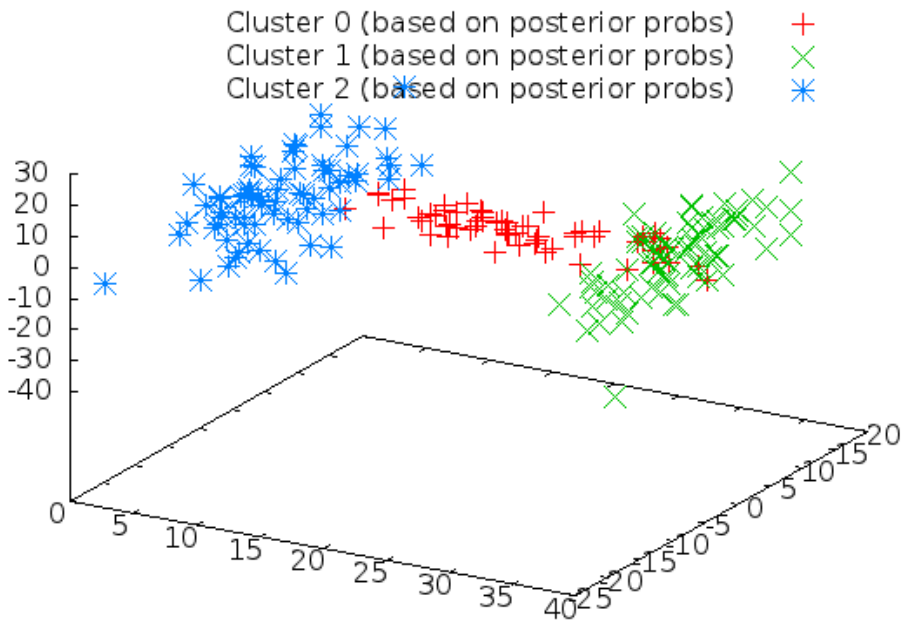


Shown in the previous figure is a scatter plot of the 3D data we will now take up. The data consists of three elongated blobs, two oriented parallel to each other and one perpendicular to the other two.

When we run the script `canned_example5.pl` with the data shown in the previous figure, for the Naive Bayes' clusters we get the clusters shown below:



And shown below are the clusters based on the posterior class probabilities exceeding 0.2 threshold.



If you look carefully at the rightmost cluster shown above, you will notice some red points interspersed with the green points. Of the various demos included in this section, this is the probability the best example of the difference between Naive Bayes' and posterior-probability based clustering.

Back to TOC

8: Acknowledgments

If you enjoyed the discussion in Section 3, the credit for that must go entirely to Richard Duda, Peter Hart, and David Stork for their introduction to EM on pages 126 through 128 of their book "Pattern Classification."

And if you enjoyed the discussion in Section 4, the credit for that must go entirely to Jeff Bilmes for his technical report entitled "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models" (TR-97-021, University of California, Berkeley).

The discussion presented in Sections 3 and 4 is merely a further elaboration of the material at the two sources mentioned above. If you are a reader who believes that the original authors listed above had already explained the material in sufficient detail and that my tutorial here merely amounts to belaboring the obvious, please accept my apologies.

If there is any pedagogical merit to this tutorial, it lies in using the EM example presented by Duda et al. as a stepping stone to my explanation of Bilmes's derivation of EM for Gaussian Mixture Models.

I was first exposed to EM a very long time back by Jennifer Dy, a former Ph.D. student (co-supervised by Carla Brodley and myself) who is now a well-known professor at Northeastern University. Jennifer was using EM for unsupervised feature selection in the context of content-based image retrieval. Here is a citation to that work: Jennifer Dy, Carla Brodley, Avi Kak, Lynn Broderick, and Alex Aisen "*Unsupervised Feature Selection Applied to Content-Based Retrieval of Lung Images*," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003.