

## Mixture of Gaussians

Recall the simple Gaussian mixture model

$$P(\underline{x}; \theta) = \sum_{j=1}^m P(j) N(\underline{x}; \underline{\mu}_j, \Sigma_j) \quad (1)$$

We have already discussed how to estimate the parameters of such models with the EM algorithm. The E and M steps of the algorithm were:

**(E-step)** Evaluate the posterior assignment probabilities  $p^{(l)}(j|t) = P(j|\underline{x}_t, \theta^{(l)})$  based on the current setting of the parameters  $\theta^{(l)}$ .

**(M-step)** Update the parameters according to

$$P^{(l+1)}(j) = \frac{\hat{n}(j)}{n}, \text{ where } \hat{n}(j) = \sum_{t=1}^n p^{(l)}(j|t) \quad (2)$$

$$\underline{\mu}_j^{(l+1)} = \frac{1}{\hat{n}(j)} \sum_{t=1}^n p^{(l)}(j|t) \underline{x}_t \quad (3)$$

$$\Sigma_j^{(l+1)} = \frac{1}{\hat{n}(j)} \sum_{t=1}^n p^{(l)}(j|t) (\underline{x}_t - \underline{\mu}_j^{(l+1)})(\underline{x}_t - \underline{\mu}_j^{(l+1)})^T \quad (4)$$

The fact that the EM algorithm is iterative, and can get stuck in locally optimal solutions, means that we have to pay attention to how the parameters are initialized. For example, if we initially set all the components to have identical means and covariances, then they would remain identical even after the EM iterations. In other words, they would be changed in exactly the same way. So, effectively, we would be estimating only a single Gaussian distribution. To understand this, note that the parameter updates (above) are based solely on the posterior assignments. If the parameters of any two components are identical, so will their posterior probabilities. Identical posteriors then lead to identical updates. Note that setting the prior probabilities differently while keeping the component models initialized the same would still lead to this degenerate result.

It is necessary to provide sufficient variation in the initialization step. We can still set the mixing proportions to be uniform,  $P^{(0)}(j) = 1/m$ , and let all the covariance matrices equal the overall data covariances. But we should randomly position the Gaussian components, e.g., by equating the means with randomly chosen data points (or iteratively selecting points

as means that are far from the already chosen means). Even with such initialization the algorithm would have to be run multiple times to ensure we will find a good solution. Figure 1 below exemplifies how a particular EM run with four components (with the suggested initialization) can get stuck in a locally optimal solution. The data in the figure came from a four component mixture.

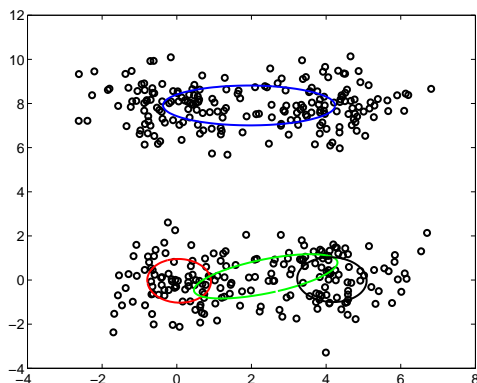


Figure 1: A locally optimal Gaussian mixture with four components

## EM theory

Let's understand the EM-algorithm a bit better in a context of a general mixture model of the form

$$P(\underline{x}; \theta) = \sum_{j=1}^m P(j)P(\underline{x}|\theta_j) \quad (5)$$

where  $\theta$  collects together both  $\{P(j)\}$  and  $\{\theta_j\}$ . The goal is to maximize the log-likelihood of data  $D = \{\underline{x}_1, \dots, \underline{x}_n\}$

$$l(D; \theta) = \sum_{t=1}^n \log P(\underline{x}_t; \theta) \quad (6)$$

As before, let's start with a complete data version of the problem and assume that we are given  $D = \{\underline{x}_1, \dots, \underline{x}_n\}$  as well as the corresponding assignments  $\mathcal{J} = \{j_1, \dots, j_n\}$ . The

complete log-likelihood of all the observations is given by:

$$l(D, \mathcal{J}; \theta) = \sum_{j=1}^m \sum_{t=1}^n \delta(j|t) \log \left( P(j)P(\underline{x}_t|\theta_j) \right) \quad (7)$$

where  $\delta(j|t) = 1$  if  $j = j_t$  and zero otherwise. If  $\theta^{(l)}$  denote the current setting of the parameters, then the E-step of the EM-algorithm corresponds to replacing the hard assignments  $\delta(j|t)$  with soft posterior assignments  $p^{(l)}(j|t) = P(j|\underline{x}_t, \theta^{(l)})$ . This replacement step corresponds to evaluating the *expected complete log-likelihood*

$$E \{l(D, \mathcal{J}; \theta) | D, \theta^{(l)}\} = \sum_{j=1}^m \sum_{t=1}^n p^{(l)}(j|t) \log \left( P(j)P(\underline{x}_t|\theta_j) \right) \quad (8)$$

The expectation is over the assignments *given*  $D = \{\underline{x}_1, \dots, \underline{x}_n\}$  and the current setting of the parameters so that  $E\{\delta(j|t) | \underline{x}_t, \theta^{(l)}\} = P(j|\underline{x}_t, \theta^{(l)}) = p^{(l)}(j|t)$ . The rationale here is that we average over variables whose values we do not know but can guess relative to the current model, completing the incomplete observations. Note that there are two sets of parameters involved in the above expression. The current setting  $\theta^{(l)}$  that defined the posterior assignments  $p^{(l)}(j|t)$  for completing the data, and  $\theta$  that we are now free to optimize over. Once we have completed the data, i.e., evaluated the posterior assignments  $p^{(l)}(j|t)$ , they won't change as a function of  $\theta$ . The E-step simply casts the incomplete data problem back into a complete data problem. In the M-step of the EM algorithm, we treat  $D = \{\underline{x}_1, \dots, \underline{x}_n\}$  and the soft completions  $p^{(l)}(j|t)$  as if they were observed data, and maximize the expected log-likelihood with respect to  $\theta$  while keeping everything else fixed.

Let's relate the EM algorithm a bit more firmly to the goal of maximizing the log-likelihood  $l(D; \theta)$ . We will show that the EM algorithm is actually optimizing an auxiliary objective that forces the log-likelihood up from below. To this end, let  $Q = \{q(j|t)\}$  be any set of distributions over the underlying assignments, not necessarily the posterior assignments  $p^{(l)}(j|t)$ . The auxiliary objective that the EM algorithm is using is

$$l(D, Q; \theta) = \sum_{j=1}^m \sum_{t=1}^n q(j|t) \log \left( P(j)P(\underline{x}_t|\theta_j) \right) + \sum_{t=1}^n H(q(\cdot|t)) \quad (9)$$

where  $H(q(\cdot|t)) = -\sum_{j=1}^m q(j|t) \log q(j|t)$  is the entropy of the assignment distribution  $q(\cdot|t)$ . We view  $l(D, Q; \theta)$  as a function of  $Q$  and  $\theta$  and have used  $q(j|t)$  for the assignment distributions so as to emphasize that they can be set independently of the parameters  $\theta$  in this objective. The EM algorithm can be seen as an algorithm that alternately

maximizes  $l(D, Q; \theta)$ , with respect to  $Q$  for fixed  $\theta$  (E-step) and with respect to  $\theta$  for a fixed  $Q$  (M-step). To make this a bit more precise, we use  $\theta^{(l)}$  for the current setting of the parameters and let  $Q^{(l)}$  denote the assignment distributions that are really the posteriors, i.e.,  $q^{(l)}(j|t) = p^{(l)}(j|t) = P(j|\underline{x}_t, \theta^{(l)})$ . It is possible to show relatively easily that  $l(D, Q; \theta^{(l)})$  attains the maximum with respect to  $Q$  exactly when  $Q = Q^{(l)}$ , i.e., when the assignment distributions are the posteriors  $P(j|\underline{x}_t, \theta^{(l)})$ . The EM-algorithm is now defined as

$$\text{(E-step)} \quad Q^{(l)} = \arg \max_Q l(D, Q; \theta^{(l)})$$

$$\text{(M-step)} \quad \theta^{(l+1)} = \arg \max_{\theta} l(D, Q^{(l)}; \theta)$$

The E-step above recovers the posterior assignments,  $q^{(l)}(j|t) = p^{(l)}(j|t)$ , and the M-step, with these posterior assignments, maximizes

$$l(D, Q^{(l)}; \theta) = \sum_{j=1}^m \sum_{t=1}^n p^{(l)}(j|t) \log \left( P(j) P(\underline{x}_t | \theta_j) \right) + \sum_{t=1}^n H(p^{(l)}(\cdot|t)) \quad (10)$$

which is exactly as before since the entropy term is fixed for the purpose of optimizing  $\theta$ . Since each step in the algorithm is a maximization step, the objective  $l(D, Q; \theta)$  has to increase monotonically. This monotone increase is precisely what underlies the monotone increase of the log-likelihood  $l(D; \theta)$  in the EM algorithm. Indeed, we claim that our auxiliary objective equals the log-likelihood after any E-step. In other words,

$$l(D, Q^{(l)}; \theta^{(l)}) = l(D; \theta^{(l)}) \quad (11)$$

It is easy (but a bit tedious) to verify this by substituting in the posterior assignments:

$$l(D, Q^{(l)}; \theta^{(l)}) = \sum_{j=1}^m \sum_{t=1}^n p^{(l)}(j|t) \log \left( P^{(l)}(j) P(\underline{x}_t | \theta_j^{(l)}) \right) + \sum_{t=1}^n H(p^{(l)}(\cdot|t)) \quad (12)$$

$$= \sum_{j=1}^m \sum_{t=1}^n p^{(l)}(j|t) \log \left( P^{(l)}(j) P(\underline{x}_t | \theta_j^{(l)}) \right) - \sum_{t=1}^n \sum_{j=1}^m p^{(l)}(j|t) \log p^{(l)}(j|t) \quad (13)$$

$$= \sum_{j=1}^m \sum_{t=1}^n p^{(l)}(j|t) \log \frac{P^{(l)}(j) P(\underline{x}_t | \theta_j^{(l)})}{p^{(l)}(j|t)} \quad (14)$$

$$= \sum_{j=1}^m \sum_{t=1}^n P(j|\underline{x}_t, \theta^{(l)}) \log \frac{P^{(l)}(j) P(\underline{x}_t | \theta_j^{(l)})}{P(j|\underline{x}_t, \theta^{(l)})} \quad (15)$$

$$= \sum_{j=1}^m \sum_{t=1}^n P(j|\underline{x}_t, \theta^{(l)}) \log P(\underline{x}_t; \theta^{(l)}) \quad (16)$$

$$= \sum_{t=1}^n \log P(\underline{x}_t; \theta^{(l)}) = l(D; \theta^{(l)}) \quad (17)$$

Note that the entropy term in the auxiliary objective is necessary for this to happen. Now, we are finally ready to state that

$$\overbrace{l(D, Q^{(l)}; \theta^{(l)})}^{l(D; \theta^{(l)})} \text{M-step} \leq l(D, Q^{(l)}; \theta^{(l+1)}) \leq \overbrace{l(D, Q^{(l+1)}; \theta^{(l+1)})}^{l(D; \theta^{(l+1)})} \text{M-step} \leq \dots \quad (18)$$

which demonstrates that the EM-algorithm monotonically increases the log-likelihood. The equality above holds only at convergence.

## Additional mixture topics: regularization

Regularization plays an important role in the context of any flexible model and this is true for mixtures as well. The number of parameters in an  $m$ -component mixture of Gaussians model in  $d$ -dimensions is exactly  $m - 1 + md + md(d + 1)/2$  and can easily become a problem when  $d$  and/or  $m$  are large. To include regularization we need to revise the basic EM-algorithm formulated for maximum likelihood estimation. From the point of view of deriving new update equations, the effect of regularization will be the same in the case of complete data (with again the resulting  $\delta(j|t)$ 's replaced by the posteriors). Let us therefore

simplify the setting a bit and consider regularizing the model when the observations are complete. The key terms to regularize are the covariance matrices.

We can assign a Wishart prior over the covariance matrices. This is a conjugate prior over covariance matrices and will behave nicely in terms of the estimation equations. The prior distribution depends on two parameters, one of which is a common covariance matrix  $S$  towards which the estimates of  $\Sigma_j$ 's will be pulled. The degree of "pull" is specified by  $n'$  known as the *equivalent sample size*.  $n'$  specifies the number of data points we would have to observe for the data to influence the solution as much as the prior. More formally, the log-wishart penalty is given by

$$\log P(\Sigma_j | S, n') = \text{const.} - \frac{n'}{2} \text{Trace}(\Sigma_j^{-1} S) - \frac{n'}{2} \log |\Sigma_j| \quad (19)$$

Given data  $D = \{\underline{x}_1, \dots, \underline{x}_n\}$ , and assignments  $\mathcal{J} = \{j_1, \dots, j_n\}$ , the penalized complete log-likelihood function is given by

$$l(D, \mathcal{J}; \theta) = \sum_{j=1}^m \left( \sum_{t=1}^n \delta(j|t) \right) \log P(j) + \sum_{j=1}^m \left( \sum_{t=1}^n \delta(j|t) \log N(\underline{x}_t; \underline{\mu}_j, \Sigma_j) + \log P(\Sigma_j | S, n') \right) \quad (20)$$

The solution (steps not provided) changes only slightly and, naturally, only in terms of the covariance matrices<sup>1</sup>:

$$\hat{P}(j) = \frac{\hat{n}(j)}{n}, \quad \text{where } \hat{n}(j) = \sum_{t=1}^n \delta(j|t) \quad (21)$$

$$\hat{\underline{\mu}}_j = \frac{1}{\hat{n}(j)} \sum_{t=1}^n \delta(j|t) \underline{x}_t \quad (22)$$

$$\hat{\Sigma}_j = \frac{1}{\hat{n}(j) + n'} \left[ \sum_{t=1}^n \delta(j|t) (\underline{x}_t - \hat{\underline{\mu}}_j)(\underline{x}_t - \hat{\underline{\mu}}_j)^T + n' S \right] \quad (23)$$

The resulting covariance updates can be used as part of the EM algorithm simply by replacing  $\delta(j|t)$  with the posterior assignments  $p^{(l)}(j|t)$ , as before. The choice of  $S$  depends on the problem but could be, e.g., either the identity matrix or  $\hat{\Sigma}$  (the overall data covariance).

---

<sup>1</sup>Note that the regularization penalty corresponds to having observed  $n'$  samples with covariance  $S$  from the *same* Gaussian. This update rule is therefore not cast in terms of inverse covariance matrices as it would be when combining noisy sources with different noise covariances.

## Additional mixture topics: sequential estimation

Mixture models serve two different purposes. They provide efficiently parameterized densities (distributions) over  $\underline{x}$  but can also help uncover structure in the data, i.e., identify which data points belong to which groups. We will discuss the latter more in the context of clustering and focus here only on predicting  $\underline{x}$ . In this setting, we can estimate mixture models a bit more efficiently in stages, similarly to boosting. In other words, suppose we have already estimated an  $m - 1$  mixture

$$\hat{P}_{m-1}(\underline{x}) = \sum_{j=1}^{m-1} \hat{P}(j)P(\underline{x}|\hat{\theta}_j) \quad (24)$$

The component distributions could be Gaussians or other densities or distributions. We can now imagine adding one more component while keeping  $\hat{P}_{m-1}(\underline{x})$  fixed. The resulting  $m$ -component mixture can be parameterized as

$$P(\underline{x}; p_m, \theta_m) = (1 - p_m)\hat{P}_{m-1}(\underline{x}) + p_mP(\underline{x}|\theta_m) \quad (25)$$

$$= \sum_{j=1}^{m-1} [(1 - p_m)\hat{P}(j)]P(\underline{x}|\hat{\theta}_j) + p_mP(\underline{x}|\theta_m) \quad (26)$$

Note that by scaling down  $\hat{P}_{m-1}(\underline{x})$  with  $1 - p_m$  we can keep the overall mixture proportions to sum to one regardless of  $p_m \in [0, 1]$ . We can now estimate  $p_m$  and  $\theta_m$  via the EM algorithm. The benefit is that this estimation problem is much simpler than estimating the full mixture; we only have one component to adjust as well as the weight assigned to that component. The EM-algorithm for estimating the component to add is given by

**(E-step)** Evaluate the posterior assignment probabilities pertaining to the new component:

$$p^{(l)}(m|t) = \frac{p_m^{(l)}P(\underline{x}_t|\theta_m^{(l)})}{(1 - p_m^{(l)})\hat{P}_{m-1}(\underline{x}_t) + p_m^{(l)}P(\underline{x}_t|\theta_m^{(l)})} \quad (27)$$

**(M-step)** Obtain new parameters  $p_m^{(l+1)}$  and  $\theta_m^{(l+1)}$

$$p_m^{(l+1)} = \frac{\sum_{t=1}^n p^{(l)}(m|t)}{n} \quad (28)$$

$$\theta_m^{(l+1)} = \arg \max_{\theta_m} \left\{ \sum_{t=1}^n p^{(l)}(m|t) \log P(\underline{x}_t|\theta_m) \right\} \quad (29)$$

Note that in Eq.(29) we maximize the *expected log-likelihood* where the hard assignments  $\delta(m|t)$  have been replaced with their expected values, i.e., soft posterior assignments  $p^{(l)}(m|t)$ . This is exactly how the updates were obtained in the case of Gaussian mixtures as well.

Note that it is often necessary to regularize the mixture components even if they are added sequentially.

## Additional mixture topics: conditional mixtures

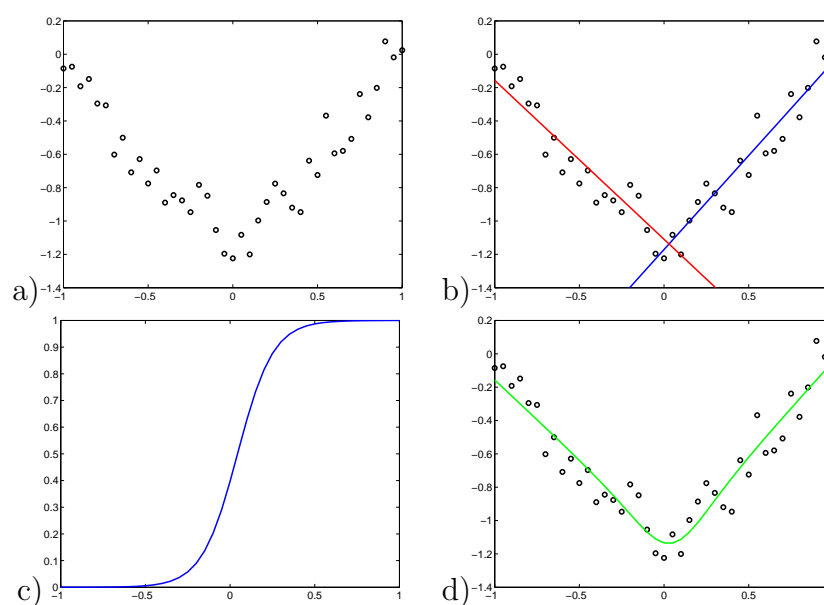


Figure 2: a) A regression problem, b) linear regression models (experts) and their assigned regions, c) gating network output (probability that we select the blue expert on the right), d) the mean output from the conditional mixture distribution as a function of the input.

We can also use mixture models for regression. These are known as *conditional mixtures* or *mixtures of experts models*. Consider a simple linear regression model

$$P(y|\underline{x}, \theta, \sigma^2) = N(y; \theta^T \underline{x}, \sigma^2) \quad (30)$$

The goal is to predict the responses as a function of the input  $\underline{x}$ . In many cases the responses are not linear functions of the input (see Figure 2a). This leaves us with a few choices. We

could try to find a feature mapping  $\phi(\underline{x})$  so as to ensure that a linear function in the feature space can capture the non-linearities that relate  $\underline{x}$  and  $y$ . It may not be always clear what the feature vector should be, however (in case of Figure 2a the relevant feature mapping would be  $\phi(x) = |x|$ ). Another approach is to divide the input space into regions such that within each region the relation between  $\underline{x}$  and  $y$  can be captured with a linear function. Such a model requires us to be able to predict how to assign linear functions (experts) to regions. In conditional mixtures, this assignment is carried out by another model, so called *gating network*. The gating network could, for example, be a logistic regression model

$$P(j = 1|\underline{x}, \eta, \eta_0) = g(\eta^T \underline{x} + \eta_0) \quad (31)$$

and  $P(j = 2|\underline{x}, \eta, \eta_0) = 1 - g(\eta^T \underline{x} + \eta_0)$  (we would have to use a *softmax* function for dividing the space into  $m$  regions). The overall model for the responses  $y$  given  $\underline{x}$  would then be a conditional mixture

$$P(y|\underline{x}) = \sum_{j=1}^2 P(j|\underline{x}, \eta, \eta_0) N(y; \theta_j^T \underline{x}, \sigma_j^2) \quad (32)$$

Note that the real difference between this and a simple mixture is that the mixing proportions as well as the component models are conditioned on the input. Such models can be again estimated via the EM-algorithm:

**(E-step)** Evaluate the posterior assignment probabilities:

$$p^{(l)}(j|t) = \frac{P(j|\underline{x}_t, \eta^{(l)}, \eta_0^{(l)}) N(y_t; (\theta_j^{(l)})^T \underline{x}_t, (\sigma_j^{(l)})^2)}{P^{(l)}(y_t|\underline{x}_t)} \quad (33)$$

Note that the assignments are based on both  $\underline{x}_t$  and  $y_t$ .

**(M-step)** Obtain new parameters by maximizing the expected log-likelihoods relative to the posterior assignments:

$$\{\eta^{(l+1)}, \eta_0^{(l+1)}\} = \arg \max_{\eta, \eta_0} \sum_{j=1}^2 \sum_{t=1}^n p^{(l)}(j|t) \log P(j|\underline{x}_t, \eta, \eta_0) \quad (34)$$

$$\{\theta_j^{(l+1)}, \sigma_j^{(l+1)}\} = \arg \max_{\theta_j, \sigma_j} \sum_{t=1}^n p^{(l)}(j|t) \log N(y_t; \theta_j^T \underline{x}_t, \sigma_j^2) \quad (35)$$

These estimation equations are readily obtained from the complete log-likelihood version pertaining to the different conditional probabilities and by replacing  $\delta(j|t)$  with the posterior assignments.

Figure 2 illustrates the resulting mixture of experts model in the toy problem.

## Mixture models and clustering

We have so far used mixture models as flexible ways of constructing probability models for prediction tasks. The motivation behind the mixture model was that the available data may include unobserved (sub)groups and by incorporating such structure in the model we could obtain more accurate predictions. We are also interested in uncovering that group structure. This is a *clustering* problem. For example, in the case of modeling exam scores it would be useful to understand what types of students there are. In a biological context, it would be useful to uncover which genes are active (expressed) in which cell types when the measurements are from tissue samples involving multiple cell types in unknown quantities. Clustering problems are ubiquitous.

Mixture models as generative models require us to articulate the type of clusters or subgroups we are looking to identify. The simplest type of clusters we could look for are spherical Gaussian clusters, i.e., we would be estimating Gaussian mixtures of the form

$$P(\underline{x}; \theta, m) = \sum_{j=1}^m P(j) N(\underline{x}; \underline{\mu}_j, \sigma_j^2 I) \quad (36)$$

where the parameters  $\theta$  include  $\{P(j)\}$ ,  $\{\underline{\mu}_j\}$ , and  $\{\sigma_j^2\}$ . Note that we are estimating the mixture models with a different objective in mind. We are more interested in finding where the clusters are than how good the mixture model is as a generative model.

There are many questions to resolve. For example, how many such spherical Gaussian clusters are there in the data? This is a model selection problem. If such clusters exist, do we have enough data to identify them? If we have enough data, can we hope to find the clusters via the EM algorithm? Is our approach robust, i.e., does our method degrade gracefully when data contain “background samples” or impurities along with spherical Gaussian clusters? Can we make the clustering algorithm more robust?