Chapter 11
Mixture models and the EM algorithm

11.1 Latent variable models

In Chapter 10 we showed how graphical models can be used to define high-dimensional joint probability distributions. The basic idea is to model dependence between two variables by adding an edge between them in the graph. (Technically the graph represents conditional independence, but you get the point.)

An alternative approach is to assume that the observed variables are correlated because they arise from a hidden common cause. Model with hidden variables are also known as **latent variable models** or **LVM**s. As we will see in this chapter, such models are harder to fit than models with no latent variables. However, they can have significant advantages, for two main reasons.

- First, LVMs often have fewer parameters than models that directly represent correlation in the visible space.
- Second, the hidden variables in an LVM can serve as a **bottleneck**, which computes a compressed representation of the data. This forms the basis of unsupervised learning, as we will see. Figure 11.1 illustrates some generic LVM structures that can be used for this purpose.

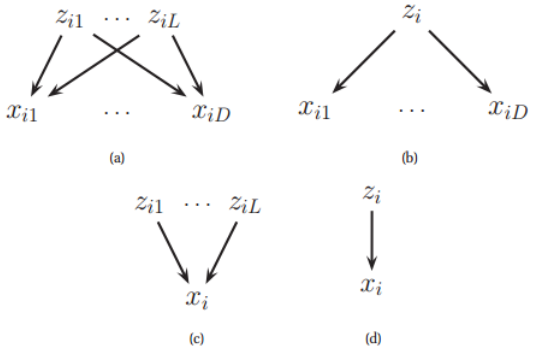


Fig. 11.1: A latent variable model represented as a DGM. (a) Many-to-many. (b) One-to-many. (c) Many-to-one. (d) One-to-one.

11.2 Mixture models

The simplest form of LVM is when $z_i \in \{1, \cdots, K\}$, representing a discrete latent state. We will use a discrete prior for this, $p(zi) = \text{Cat}(\boldsymbol{\pi})$. For the likelihood, we use $p(\boldsymbol{x}_i|z_i = k) = p_k(\boldsymbol{x}_i)$, where p_k is the k'th **base distribution** for the observations; this can be of any type. The overall model is known as a **mixture model**, since we are mixing together the K base distributions as follows:

$$p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}_i|\boldsymbol{\theta}) \tag{11.1}$$

Depending on the form of the likelihood $p(\boldsymbol{x}_i|\boldsymbol{z}_i)$ and the prior $p(\boldsymbol{z}_i)$, we can generate a variety of different models, as summarized in Table 11.1.

$p(\boldsymbol{x}_i|\boldsymbol{z}_i)$	$p(\boldsymbol{z}_i)$	**Name**	**Section**
MVN	Discrete	Mixture of Gaussians	11.2.1
Prod. Discrete	Discrete	Mixture of multinomials	11.2.2
Prod. Gaussian	Prod. Gaussian	Factor analysis/ probabilistic PCA	12.1.5
Prod. Gaussian	Prod. Laplace	Probabilistic ICA/ sparse coding	12.6
Prod. Discrete	Prod. Gaussian	Multinomial PCA	27.2.3
Prod. Discrete	Dirichlet	Latent Dirichlet allocation	27.3
Prod. Noisy-OR	Prod. Bernoulli	BN20/ QMR	10.2.3
Prod. Bernoulli	Prod. Bernoulli	Sigmoid belief net	27.7

Table 11.1: Summary of some popular directed latent variable models. Here Prod means product, so Prod. Discrete in the likelihood means a factored distribution of the form $\prod_j \text{Cat}(x_{ij}|\boldsymbol{z}_i)$, and Prod. Gaussian means a factored distribution of the form $\prod_j \mathcal{N}(x_{ij}|\boldsymbol{z}_i)$.

11.2.1 Mixtures of Gaussians

$$p_k(\boldsymbol{x}_i|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \Sigma_k) \tag{11.2}$$

11.2.2 Mixtures of multinoullis

$$p_k(\boldsymbol{x}_i|\boldsymbol{\theta}) = \prod_{j=1}^{D} \text{Ber}(x_{ij}|\mu_{jk}) = \prod_{j=1}^{D} \mu_{jk}^{x_{ij}}(1-\mu_{jk})^{1-x_{ij}} \tag{11.3}$$

where μ_{jk} is the probability that bit j turns on in cluster k.

The latent variables do not have to any meaning, we might simply introduce latent variables in order to make the model more powerful. For example, one can show that the mean and covariance of the mixture distribution are given by

$$\mathbb{E}[\boldsymbol{x}] = \sum_{k=1}^{K} \pi_k \boldsymbol{\mu}_k \tag{11.4}$$

$$\text{Cov}[\boldsymbol{x}] = \sum_{k=1}^{K} \pi_k(\Sigma_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T) - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{x}]^T \tag{11.5}$$

where $\Sigma_k = \text{diag}(\mu_{jk}(1-\mu_{jk}))$. So although the component distributions are factorized, the joint distribution is not. Thus the mixture distribution can capture correlations between variables, unlike a single product-of-Bernoullis model.

11.2.3 Using mixture models for clustering

There are two main applications of mixture models, black-box density model(see Section 14.7.3 TODO) and clustering(see Chapter 25 TODO).

Soft clustering

$$\begin{aligned} r_{ik} &\triangleq p(z_i = k|\boldsymbol{x}_i, \boldsymbol{\theta}) = \frac{p(z_i = k, \boldsymbol{x}_i|\boldsymbol{\theta})}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} \\ &= \frac{p(z_i = k|\boldsymbol{\theta})p(\boldsymbol{x}_i|z_i = k, \boldsymbol{\theta})}{\sum_{k'=1}^{K} p(z_i = k'|\boldsymbol{\theta})p(\boldsymbol{x}_i|z_i = k', \boldsymbol{\theta})} \end{aligned} \tag{11.6}$$

where r_{ik} is known as the **responsibility** of cluster k for point i.

Hard clustering

$$z_i^* \triangleq \arg\max_k r_{ik} = \arg\max_k p(z_i = k|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{11.7}$$

The difference between generative classifiers and mixture models only arises at training time: in the mixture case, we never observe z_i, whereas with a generative classifier, we do observe y_i(which plays the role of z_i).

11.2.4 Mixtures of experts

Section 14.7.3 TODO described how to use mixture models in the context of generative classifiers. We can also use them to create discriminative models for classification and regression. For example, consider the data in Figure 11.2(a). It seems like a good model would be three different linear regression functions, each applying to a different part of the input space. We can model this by allowing the mixing weights and the mixture densities to be input-dependent:

$$p(y_i|\boldsymbol{x}_i, z_i = k, \boldsymbol{\theta}) = \mathcal{N}(y_i|\boldsymbol{w}_k^T \boldsymbol{x}, \sigma_k^2) \tag{11.8}$$

$$p(z_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \text{Cat}(z_i|\mathcal{S}(\boldsymbol{V}^T \boldsymbol{x}_i)) \tag{11.9}$$

See Figure 11.3(a) for the DGM.

This model is called a **mixture of experts** or MoE (Jordan and Jacobs 1994). The idea is that each submodel is considered to be an expert in a certain region of input space. The function $p(z_i|\boldsymbol{x}_i, \boldsymbol{\theta})$ is called a **gating function**, and decides which expert to use, depending on the input values. For example, Figure 11.2(b) shows how the three experts have carved up the 1d input space, Figure 11.2(a) shows the predictions of each expert individually (in this case, the experts are just linear regression models), and Figure 11.2(c) shows the overall prediction of the model, obtained using

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \sum_{k=1}^{K} p(z_i = k|\boldsymbol{x}_i, \boldsymbol{\theta})p(y_i|\boldsymbol{x}_i, z_i = k, \boldsymbol{\theta}) \tag{11.10}$$

We discuss how to fit this model in Section TODO 11.4.3.

11.3 Parameter estimation for mixture models

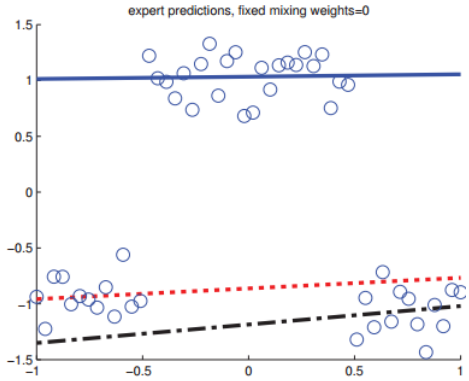
11.3.1 Unidentifiability

11.3.2 Computing a MAP estimate is non-convex

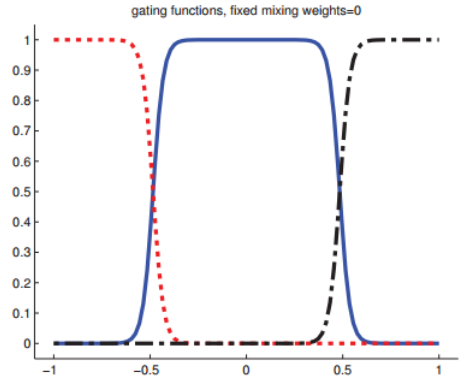
11.4 The EM algorithm

11.4.1 Introduction

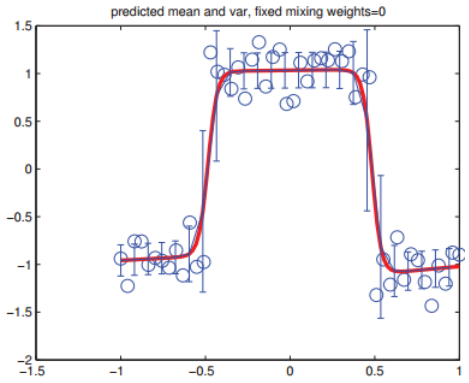
For many models in machine learning and statistics, computing the ML or MAP parameter estimate is easy provided we observe all the values of all the relevant random



(a)

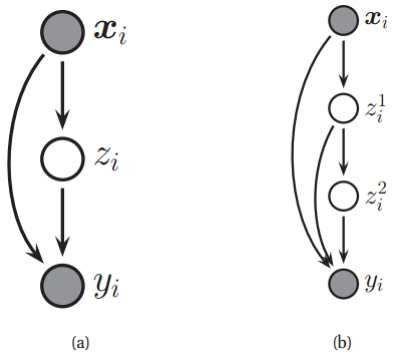


(b)



(c)

Fig. 11.2: (a) Some data fit with three separate regression lines. (b) Gating functions for three different experts. (c) The conditionally weighted average of the three expert predictions.



(a) (b)

Fig. 11.3: (a) A mixture of experts. (b) A hierarchical mixture of experts.

variables, i.e., if we have complete data. However, if we have missing data and/or latent variables, then computing the ML/MAP estimate becomes hard.

One approach is to use a generic gradient-based optimizer to find a local minimum of the NLL($\boldsymbol{\theta}$). However, we often have to enforce constraints, such as the fact that covariance matrices must be positive definite, mixing weights must sum to one, etc., which can be tricky. In such cases, it is often much simpler (but not always faster) to use an algorithm called **expectation maximization**,or **EM** for short (Dempster et al. 1977; Meng and van Dyk 1997; McLachlan and Krishnan 1997). This is is an efficient iterative algorithm to compute the ML or MAP estimate in the presence of missing or hidden data, often with closed-form updates at each step. Furthermore, the algorithm automatically enforce the required constraints.

See Table 11.2 for a summary of the applications of EM in this book.

Table 11.2: Some models discussed in this book for which EM can be easily applied to find the ML/ MAP parameter estimate.

Model	**Section**
Mix. Gaussians	11.4.2
Mix. experts	11.4.3
Factor analysis	12.1.5
Student T	11.4.5
Probit regression	11.4.6
DGM with hidden variables	11.4.4
MVN with missing data	11.6.1
HMMs	17.5.2
Shrinkage estimates of Gaussian means	Exercise 11.13

11.4.2 Basic idea

EM exploits the fact that if the data were fully observed, then the ML/ MAP estimate would be easy to compute. In particular, each iteration of the EM algorithm consists of two processes: The E-step, and the M-step.

- In the **E-step**, the missing data are inferred given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation, explaining the choice of terminology.
- In the **M-step**, the likelihood function is maximized under the assumption that the missing data are known. The missing data inferred from the E-step are used in lieu of the actual missing data.

Let $\boldsymbol{x}_i$ be the visible or observed variables in case i, and let $\boldsymbol{z}_i$ be the hidden or missing variables. The goal is to maximize the log likelihood of the observed data:

$$\ell(\boldsymbol{\theta}) = \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \sum_{\boldsymbol{z}_i} p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{\theta}) \tag{11.11}$$

Unfortunately this is hard to optimize, since the log cannot be pushed inside the sum.

EM gets around this problem as follows. Define the **complete data log likelihood** to be

$$\ell_c(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{\theta}) \tag{11.12}$$

This cannot be computed, since $\boldsymbol{z}_i$ is unknown. So let us define the **expected complete data log likelihood** as follows:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) \triangleq \mathbb{E}_{\boldsymbol{z}|\mathcal{D},\boldsymbol{\theta}^{t-1}}\left[\ell_c(\boldsymbol{\theta})\right] = \mathbb{E}\left[\ell_c(\boldsymbol{\theta})|\mathcal{D}, \boldsymbol{\theta}^{t-1}\right] \tag{11.13}$$

where t is the current iteration number. Q is called the **auxiliary function**(see Section 11.4.9 for derivation). The expectation is taken wrt the old parameters, $\boldsymbol{\theta}^{t-1}$, and the observed data $\mathcal{D}$. The goal of the E-step is to compute $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$, or rather, the parameters inside of it which the MLE(or MAP) depends on; these are known as the **expected sufficient statistics** or **ESS**. In the M-step, we optimize the Q function wrt $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^t = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) \tag{11.14}$$

To perform MAP estimation, we modify the M-step as follows:

$$\boldsymbol{\theta}^t = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) + \log p(\boldsymbol{\theta}) \tag{11.15}$$

The E step remains unchanged.

In summary, the EM algorithm's pseudo code is as follows

input : observed data $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$,joint distribution $P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$
output: model's parameters $\boldsymbol{\theta}$
// 1. identify hidden variables $\boldsymbol{z}$, write out the log likelihood function $\ell(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$
$\boldsymbol{\theta}^{(0)}$ = ... // initialize
while *(!convergency)* **do**
  // 2. E-step: plug in $P(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$, derive the formula of $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$
  $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \mathbb{E}\left[\ell_c(\boldsymbol{\theta})|\mathcal{D}, \boldsymbol{\theta}^{t-1}\right]$
  // 3. M-step: find $\boldsymbol{\theta}$ that maximizes the value of $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$
  $\boldsymbol{\theta}^t = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$
end

Algorithm 3: EM algorithm

Below we explain how to perform the E and M steps for several simple models, that should make things clearer.

11.4.3 EM for GMMs

11.4.3.1 Auxiliary function

$$\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) &= \mathbb{E}_{z|\mathcal{D},\boldsymbol{\theta}^{t-1}}\left[\ell_c(\boldsymbol{\theta})\right] \\
&= \mathbb{E}_{z|\mathcal{D},\boldsymbol{\theta}^{t-1}}\left[\sum_{i=1}^{N} \log p(\boldsymbol{x}_i, z_i|\boldsymbol{\theta})\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_{z|\mathcal{D},\boldsymbol{\theta}^{t-1}}\left\{\log\left[\prod_{k=1}^{K} (\pi_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k))^{\mathbb{I}(z_i=k)}\right]\right\} \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}[\mathbb{I}(z_i=k)] \log\left[\pi_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)\right] \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} p(z_i=k|\boldsymbol{x}_i, \boldsymbol{\theta}^{t-1}) \log\left[\pi_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)\right] \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log \pi_k + \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log p(\boldsymbol{x}_i|\boldsymbol{\theta}_k)
\end{aligned} \tag{11.16}$$

where $r_{ik} \triangleq \mathbb{E}[\mathbb{I}(z_i=k)] = p(z_i=k|\boldsymbol{x}_i, \boldsymbol{\theta}^{t-1})$ is the **responsibility** that cluster k takes for data point i. This is computed in the E-step, described below.

11.4.3.2 E-step

The E-step has the following simple form, which is the same for any mixture model:

$$r_{ik} = p(z_i = k|\boldsymbol{x}_i, \boldsymbol{\theta}^{t-1}) = \frac{p(z_i = k, \boldsymbol{x}_i|\boldsymbol{\theta}^{t-1})}{p(\boldsymbol{x}_i|\boldsymbol{\theta}^{t-1})} = \frac{\pi_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{t-1})}{\sum_{k'=1}^{K} \pi_{k'} p(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{t-1})} \tag{11.17}$$

11.4.3.3 M-step

In the M-step, we optimize Q wrt $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_k$.

For $\boldsymbol{\pi}$, grouping together only the terms that depend on π_k, we find that we need to maximize $\sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log \pi_k$. However, there is an additional constraint $\sum_{k=1}^{K} \pi_k = 1$, since they represent the probabilities $\boldsymbol{\pi}_k = P(z_i = k)$. To deal with the constraint we construct the Lagrangian

$$\mathcal{L}(\boldsymbol{\pi}) = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log \pi_k + \beta \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

where β is the Lagrange multiplier. Taking derivatives, we find

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik}}{N} \tag{11.18}$$

This is the same for any mixture model, whereas $\boldsymbol{\theta}_k$ depends on the form of $p(\boldsymbol{x}|\boldsymbol{\theta}_k)$.

For $\boldsymbol{\theta}_k$, plug in the pdf to Equation 11.16

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log \pi_k - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} [\log |\boldsymbol{\Sigma}_k| + (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)]$$

Take partial derivatives of Q wrt $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ and let them equal to 0, we can get

$$\frac{\partial Q}{\partial \boldsymbol{\mu}_k} = -\frac{1}{2}\sum_{i=1}^{N} r_{ik} \left[(\boldsymbol{\Sigma}_k^{-1} + \boldsymbol{\Sigma}_k^{-T})(\boldsymbol{x}_i - \boldsymbol{\mu}_k) \right] = -\sum_{i=1}^{N} r_{ik} \left[\boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \right] = 0 \Rightarrow$$

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik} \boldsymbol{x}_i}{\sum_{i=1}^{N} \hat{r}_{ik}} \tag{11.19}$$

$$\frac{\partial Q}{\partial \boldsymbol{\Sigma}_k} = -\frac{1}{2}\sum_{i=1}^{N} r_{ik} \left[\frac{1}{\boldsymbol{\Sigma}_k} - \frac{1}{\boldsymbol{\Sigma}_k^2} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \right] = 0 \Rightarrow$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \hat{r}_{ik}} \tag{11.20}$$

$$= \frac{\sum_{i=1}^{N} \hat{r}_{ik} \boldsymbol{x}_i \boldsymbol{x}_i^T}{\sum_{i=1}^{N} \hat{r}_{ik}} - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T \tag{11.21}$$

11.4.3.4 Algorithm pseudo code

input : observed data $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$,GMM
output: GMM's parameters $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
// 1. initialize
$\boldsymbol{\pi}^{(0)}$ = ...
$\boldsymbol{\mu}^{(0)}$ = ...
$\boldsymbol{\Sigma}^{(0)}$ = ...
t = 0
while *(!convergency)* **do**
  // 2. E-step
  $\hat{r}_{ik} = \frac{\pi_k p(\boldsymbol{x}_i|\boldsymbol{\theta}_k^{t-1})}{\sum_{k'=1}^{K} \pi_{k'} p(\boldsymbol{x}_i|\boldsymbol{\mu}_k^{t-1}, \boldsymbol{\Sigma}_k^{t-1})}$
  // 3. M-step
  $\hat{\boldsymbol{\pi}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik}}{N}$
  $\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik} \boldsymbol{x}_i}{\sum_{i=1}^{N} \hat{r}_{ik}}$
  $\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{i=1}^{N} \hat{r}_{ik} \boldsymbol{x}_i \boldsymbol{x}_i^T}{\sum_{i=1}^{N} \hat{r}_{ik}} - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T$
  ++t
end

Algorithm 4: EM algorithm for GMM

11.4.3.5 MAP estimation

As usual, the MLE may overfit. The overfitting problem is particularly severe in the case of GMMs. An easy solution to this is to perform MAP estimation. The new auxiliary function is the expected complete data log-likelihood plus the log prior:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log \pi_k + \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \log p(\boldsymbol{x}_i|\boldsymbol{\theta}_k) + \log p(\boldsymbol{\pi}) + \sum_{k=1}^{K} \log p(\boldsymbol{\theta}_k) \tag{11.22}$$

It is natural to use conjugate priors.

$$p(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$$
$$p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \text{NIW}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k|\boldsymbol{m}_0, \kappa_0, \nu_0, \boldsymbol{S}_0)$$

From Equation 3.28 and Section 4.6.3, the MAP estimate is given by

$$\hat{\pi}_k = \frac{\sum_{i=1}^N r_{ik} + \alpha_k - 1}{N + \sum_{k=1}^K \alpha_k - K} \tag{11.23}$$

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^N r_{ik}\boldsymbol{x}_i + \kappa_0 \boldsymbol{m}_0}{\sum_{i=1}^N r_{ik} + \kappa_0} \tag{11.24}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\boldsymbol{S}_0 + \boldsymbol{S}_k + \frac{\kappa_0 r_k}{\kappa_0 + r_k}(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)^T}{\nu_0 + r_k + D + 2} \tag{11.25}$$

where $r_k \triangleq \sum_{i=1}^N r_{ik}, \bar{\boldsymbol{x}}_k \triangleq \frac{\sum_{i=1}^N r_{ik}\boldsymbol{x}_i}{r_k}$,

$$\boldsymbol{S}_k \triangleq \sum_{i=1}^N r_{ik}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k)^T$$

11.4.4 EM for K-means

11.4.4.1 Representation

$$y_j = k \text{ if } \|\boldsymbol{x}_j - \boldsymbol{\mu}_k\|_2^2 \text{ is minimal} \tag{11.26}$$

where $\boldsymbol{\mu}_k$ is the centroid of cluster k.

11.4.4.2 Evaluation

$$\arg\min_{\boldsymbol{\mu}} \sum_{j=1}^N \sum_{k=1}^K \gamma_{jk} \|\boldsymbol{x}_j - \boldsymbol{\mu}_k\|_2^2 \tag{11.27}$$

The hidden variable is γ_{jk}, which's meanining is:

$$\gamma_{jk} = \begin{cases} 1, & \text{if } \|\boldsymbol{x}_j - \boldsymbol{\mu}_k\|_2 \text{ is minimal for } \boldsymbol{\mu}_k \\ 0, & \text{otherwise} \end{cases}$$

11.4.4.3 Optimization

E-Step:

$$\gamma_{jk}^{(i+1)} = \begin{cases} 1, & \text{if } \|\boldsymbol{x}_j - \boldsymbol{\mu}_k^{(i)}\|_2 \text{ is minimal for } \boldsymbol{\mu}_k^{(i)} \\ 0, & \text{otherwise} \end{cases} \tag{11.28}$$

M-Step:

$$\boldsymbol{\mu}_k^{(i+1)} = \frac{\sum_{j=1}^N \gamma_{jk}^{(i+1)} \boldsymbol{x}_j}{\sum \gamma_{jk}^{(i+1)}} \tag{11.29}$$

11.4.4.4 Tricks

Choosing k

TODO

Choosing the initial centroids(seeds)

1. **K-means++**.
   The intuition that spreading out the k initial cluster centers is a good thing is behind this approach: the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center[21].
   The exact algorithm is as follows:
   a. Choose one center uniformly at random from among the data points.
   b. For each data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the nearest center that has already been chosen.
   c. Choose one new data point at random as a new center, using a weighted probability distribution where a point $\boldsymbol{x}$ is chosen with probability proportional to $D(\boldsymbol{x})^2$.
   d. Repeat Steps 2 and 3 until k centers have been chosen.
   e. Now that the initial centers have been chosen, proceed using standard k-means clustering.
2. TODO

11.4.5 EM for mixture of experts

11.4.6 EM for DGMs with hidden variables

11.4.7 EM for the Student distribution *

11.4.8 EM for probit regression *

11.4.9 Derivation of the Q function

Theorem 11.1. ***(Jensen's inequality)*** *Let f be a convex function(see Section A.1) defined on a convex set $\mathcal{S}$. If $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n \in \mathcal{S}$ and $\lambda_1, \lambda_2, \cdots, \lambda_n \geq 0$ with $\sum_{i=1}^n \lambda_i = 1$,*

$$f\left(\sum_{i=1}^n \lambda_i \boldsymbol{x}_i\right) \leq \sum_{i=1}^n \lambda_i f(\boldsymbol{x}_i) \tag{11.30}$$

Proposition 11.1.

$$\log\left(\sum_{i=1}^n \lambda_i \boldsymbol{x}_i\right) \geq \sum_{i=1}^n \lambda_i \log(\boldsymbol{x}_i) \tag{11.31}$$

[21] http://en.wikipedia.org/wiki/K-means++

Now let's proof why the Q function should look like Equation 11.13:

$$
\begin{aligned}
\ell(\boldsymbol{\theta}) &= \log P(\mathcal{D}|\boldsymbol{\theta}) \\
&= \log \sum_{\boldsymbol{z}} P(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta}) \\
&= \log \sum_{\boldsymbol{z}} P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta}) \\
\ell(\boldsymbol{\theta}) - \ell(\boldsymbol{\theta}^{t-1}) &= \log \left[\sum_{\boldsymbol{z}} P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta}) \right] - \log P(\mathcal{D}|\boldsymbol{\theta}^{t-1}) \\
&= \log \left[\sum_{\boldsymbol{z}} P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta}) \frac{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1})}{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1})} \right] \\
&\quad - \log P(\mathcal{D}|\boldsymbol{\theta}^{t-1}) \\
&= \log \left[\sum_{\boldsymbol{z}} P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \frac{P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1})} \right] \\
&\quad - \log P(\mathcal{D}|\boldsymbol{\theta}^{t-1}) \\
&\geq \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \log \left[\frac{P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1})} \right] \\
&\quad - \log P(\mathcal{D}|\boldsymbol{\theta}^{t-1}) \\
&= \sum_{\boldsymbol{z}} \{ P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \\
&\quad \log \left[\frac{P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) P(\mathcal{D}|\boldsymbol{\theta}^{t-1})} \right] \Big\}
\end{aligned}
$$

$$
\begin{aligned}
B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) &\triangleq \ell(\boldsymbol{\theta}^{t-1}) + \\
&\sum_{\boldsymbol{z}} P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \log \left[\frac{P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) P(\mathcal{D}|\boldsymbol{\theta}^{t-1})} \right] \\
&\Rightarrow
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{\theta}^t &= \arg\max_{\boldsymbol{\theta}} B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) \\
&= \arg\max_{\boldsymbol{\theta}} \{ \ell(\boldsymbol{\theta}^{t-1}) + \\
&\quad \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \log \left[\frac{P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta})}{P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) P(\mathcal{D}|\boldsymbol{\theta}^{t-1})} \right] \Big\}
\end{aligned}
$$

Now drop terms which are constant w.r.t. $\boldsymbol{\theta}$

$$
\begin{aligned}
&= \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \log \left[P(\mathcal{D}|\boldsymbol{z}, \boldsymbol{\theta}) P(\boldsymbol{z}|\boldsymbol{\theta}) \right] \right\} \\
&= \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}) \log \left[P(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta}) \right] \right\} \\
&= \arg\max_{\boldsymbol{\theta}} \left\{ \mathbb{E}_{\boldsymbol{z}|\mathcal{D}, \boldsymbol{\theta}^{t-1}} \log \left[P(\mathcal{D}, \boldsymbol{z}|\boldsymbol{\theta}) \right] \right\} \qquad (11.32) \\
&\triangleq \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) \qquad (11.33)
\end{aligned}
$$

11.4.10 Convergence of the EM Algorithm *

11.4.10.1 Expected complete data log likelihood is a lower bound

Note that $\ell(\boldsymbol{\theta}) \geq B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$, and $\ell(\boldsymbol{\theta}^{t-1}) \geq B(\boldsymbol{\theta}^{t-1}, \boldsymbol{\theta}^{t-1})$, which means $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$ is an lower bound of $\ell(\boldsymbol{\theta})$. If we maximize $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$, then $\ell(\boldsymbol{\theta})$ gets maximized, see Figure 11.4.

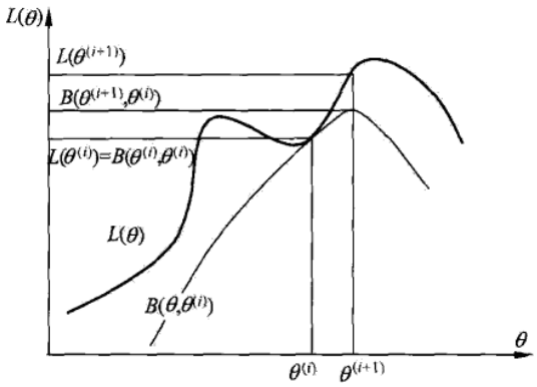


Fig. 11.4: Graphical interpretation of a single iteration of the EM algorithm: The function $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$ is bounded above by the log likelihood function $\ell(\boldsymbol{\theta})$. The functions are equal at $\boldsymbol{\theta} = \boldsymbol{\theta}^{t-1}$. The EM algorithm chooses $\boldsymbol{\theta}^{t-1}$ as the value of $\boldsymbol{\theta}$ for which $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$ is a maximum. Since $\ell(\boldsymbol{\theta}) \geq B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$ increasing $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$ ensures that the value of the log likelihood function $\ell(\boldsymbol{\theta})$ is increased at each step.

Since the expected complete data log likelihood Q is derived from $B(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1})$ by dropping terms which are constant w.r.t. $\boldsymbol{\theta}$, so it is also a lower bound to a lower bound of $\ell(\boldsymbol{\theta})$.

11.4.10.2 EM monotonically increases the observed data log likelihood

11.4.11 Generalization of EM Algorithm *

EM algorithm can be interpreted as F function's maximization-maximization algorithm, based on this interpretation there are many variations and generalization, e.g., generalized EM Algorithm(GEM).

11.4.11.1 F function's maximization-maximization algorithm

Definition 11.1. Given the probability distribution of the hidden variable Z is $\tilde{P}(Z)$, define **F function** as the following:

$$F(\tilde{P},\boldsymbol{\theta}) = \mathbb{E}_{\tilde{P}}[\log P(X,Z|\boldsymbol{\theta})] + H(\tilde{P}) \tag{11.34}$$

Where $H(\tilde{P}) = -\mathbb{E}_{\tilde{P}} \log \tilde{P}(Z)$, which is $\tilde{P}(Z)$'s entropy. Usually we assume that $P(X,Z|\boldsymbol{\theta})$ is continuous w.r.t. $\boldsymbol{\theta}$, therefore $F(\tilde{P},\boldsymbol{\theta})$ is continuous w.r.t. $\tilde{P}$ and $\boldsymbol{\theta}$.

Lemma 11.1. *For a fixed $\boldsymbol{\theta}$, there is only one distribution $\tilde{P}_\theta$ which maximizes $F(\tilde{P},\boldsymbol{\theta})$*

$$\tilde{P}_\theta(Z) = P(Z|X,\boldsymbol{\theta}) \tag{11.35}$$

and $\tilde{P}_\theta$ is continuous w.r.t. $\boldsymbol{\theta}$.

Proof. Given a fixed $\boldsymbol{\theta}$, we can get $\tilde{P}_\theta$ which maximizes $F(\tilde{P},\boldsymbol{\theta})$. we construct the Lagrangian

$$\begin{aligned}\mathcal{L}(\tilde{P},\boldsymbol{\theta}) &= \mathbb{E}_{\tilde{P}}[\log P(X,Z|\boldsymbol{\theta})] - \mathbb{E}_{\tilde{P}} \log \tilde{P}_\theta(Z) \\ &\quad + \lambda \left[1 - \sum_Z \tilde{P}(Z)\right]\end{aligned} \tag{11.36}$$

Take partial derivative with respect to $\tilde{P}_\theta(Z)$ then we get

$$\frac{\partial \mathcal{L}}{\partial \tilde{P}_\theta(Z)} = \log P(X,Z|\boldsymbol{\theta}) - \log \tilde{P}_\theta(Z) - 1 - \lambda$$

Let it equal to 0, we can get

$$\lambda = \log P(X,Z|\boldsymbol{\theta}) - \log \tilde{P}_\theta(Z) - 1$$

Then we can derive that $\tilde{P}_\theta(Z)$ is proportional to $P(X,Z|\boldsymbol{\theta})$

$$\begin{aligned}\frac{P(X,Z|\boldsymbol{\theta})}{\tilde{P}_\theta(Z)} &= e^{1+\lambda} \\ \Rightarrow \tilde{P}_\theta(Z) &= \frac{P(X,Z|\boldsymbol{\theta})}{e^{1+\lambda}} \\ \sum_Z \tilde{P}_\theta(Z) = 1 \Rightarrow \sum_Z \frac{P(X,Z|\boldsymbol{\theta})}{e^{1+\lambda}} &= 1 \Rightarrow P(X|\boldsymbol{\theta}) = e^{1+\lambda} \\ \tilde{P}_\theta(Z) &= \frac{P(X,Z|\boldsymbol{\theta})}{e^{1+\lambda}} = \frac{P(X,Z|\boldsymbol{\theta})}{P(X|\boldsymbol{\theta})} = P(Z|X,\boldsymbol{\theta})\end{aligned}$$

Lemma 11.2. *If $\tilde{P}_\theta(Z) = P(Z|X,\boldsymbol{\theta})$, then*

$$F(\tilde{P},\boldsymbol{\theta}) = \log P(X|\boldsymbol{\theta}) \tag{11.37}$$

Theorem 11.2. *One iteration of EM algorithm can be implemented as F function's maximization-maximization.*

Assume $\boldsymbol{\theta}^{t-1}$ is the estimation of $\boldsymbol{\theta}$ in the $(t-1)$-th iteration, $\tilde{P}^{t-1}$ is the estimation of $\tilde{P}$ in the $(t-1)$-th iteration. Then in the t-th iteration two steps are:

1. for fixed $\boldsymbol{\theta}^{t-1}$, find $\tilde{P}^t$ that maximizes $F(\tilde{P},\boldsymbol{\theta}^{t-1})$;
2. for fixed $\tilde{P}^t$, find $\boldsymbol{\theta}^t$ that maximizes $F(\tilde{P}^t,\boldsymbol{\theta})$.

Proof. (1) According to Lemma 11.1, we can get

$$\tilde{P}^t(Z) = P(Z|X,\boldsymbol{\theta}^{t-1})$$

(2) According above, we can get

$$\begin{aligned}F(\tilde{P}^t,\boldsymbol{\theta}) &= \mathbb{E}_{\tilde{P}^t}[\log P(X,Z|\boldsymbol{\theta})] + H(\tilde{P}^t) \\ &= \sum_Z P(Z|X,\boldsymbol{\theta}^{t-1}) \log P(X,Z|\boldsymbol{\theta}) + H(\tilde{P}^t) \\ &= Q(\boldsymbol{\theta},\boldsymbol{\theta}^{t-1}) + H(\tilde{P}^t)\end{aligned}$$

Then

$$\boldsymbol{\theta}^t = \arg\max_\theta F(\tilde{P}^t,\boldsymbol{\theta}) = \arg\max_\theta Q(\boldsymbol{\theta},\boldsymbol{\theta}^{t-1})$$

11.4.11.2 The Generalized EM Algorithm(GEM)

In the formulation of the EM algorithm described above, $\boldsymbol{\theta}^t$ was chosen as the value of $\boldsymbol{\theta}$ for which $Q(\boldsymbol{\theta},\boldsymbol{\theta}^{t-1})$ was maximized. While this ensures the greatest increase in $\ell(\boldsymbol{\theta})$, it is however possible to relax the requirement of maximization to one of simply increasing $Q(\boldsymbol{\theta},\boldsymbol{\theta}^{t-1})$ so that $Q(\boldsymbol{\theta}^t,\boldsymbol{\theta}^{t-1}) \geq Q(\boldsymbol{\theta}^{t-1},\boldsymbol{\theta}^{t-1})$. This approach, to simply increase and not necessarily maximize $Q(\boldsymbol{\theta}^t,\boldsymbol{\theta}^{t-1})$ is known as the Generalized Expectation Maximization (GEM) algorithm and is often useful in cases where the maximization is difficult. The convergence of the GEM algorithm is similar to the EM algorithm.

11.4.12 Online EM

11.4.13 Other EM variants *

11.5 Model selection for latent variable models

When using LVMs, we must specify the number of latent variables, which controls the model complexity. In particular, in the case of mixture models, we must specify K, the number of clusters. Choosing these parameters is an example of model selection. We discuss some approaches below.

11.5.1 Model selection for probabilistic models

The optimal Bayesian approach, discussed in Section 5.3, is to pick the model with the largest marginal likelihood, $K^* = \arg\max_k p(\mathcal{D}|k)$.

There are two problems with this. First, evaluating the marginal likelihood for LVMs is quite difficult. In practice, simple approximations, such as BIC, can be used (see e.g., (Fraley and Raftery 2002)). Alternatively, we can use the cross-validated likelihood as a performance measure, although this can be slow, since it requires fitting each model F times, where Fis the number of CV folds.

The second issue is the need to search over a potentially large number of models. The usual approach is to perform exhaustive search over all candidate values ofK. However, sometimes we can set the model to its maximal size, and then rely on the power of the Bayesian Occams razor to kill off unwanted components. An example of this will be shown in Section TODO 21.6.1.6, when we discuss variational Bayes.

An alternative approach is to perform stochastic sampling in the space of models. Traditional approaches, such as (Green 1998, 2003; Lunn et al. 2009), are based on reversible jump MCMC, and use birth moves to propose new centers, and death moves to kill off old centers. However, this can be slow and difficult to implement. A simpler approach is to use a Dirichlet process mixture model, which can be fit using Gibbs sampling, but still allows for an unbounded number of mixture components; see Section TODO 25.2 for details.

Perhaps surprisingly, these sampling-based methods can be faster than the simple approach of evaluating the quality of eachKseparately. The reason is that fitting the model for each K is often slow. By contrast, the sampling methods can often quickly determine that a certain value of K is poor, and thus they need not waste time in that part of the posterior.

11.5.2 Model selection for non-probabilistic methods

What if we are not using a probabilistic model? For example, how do we choose K for the K-means algorithm? Since this does not correspond to a probability model, there is no likelihood, so none of the methods described above can be used.

An obvious proxy for the likelihood is the **reconstruction error**. Define the squared reconstruction error of a data set $\mathcal{D}$, using model complexity K, as follows:

$$E(\mathcal{D},K) \triangleq \frac{1}{|\mathcal{D}|}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 \tag{11.38}$$

In the case of K-means, the reconstruction is given by $\hat{\boldsymbol{x}}_i = \boldsymbol{\mu}_{z_i}$, where $z_i = \arg\min_k \|\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k\|^2$, as explained in Section 11.4.2.6 TODO.

In supervised learning, we can always use cross validation to select between non-probabilistic models of different complexity, but this is not the case with unsupervised learning. The most common approach is to plot the reconstruction error on the training set versus K, and to try to identify a **knee** or **kink** in the curve.

11.6 Fitting models with missing data

Suppose we want to fit a joint density model by maximum likelihood, but we have holes in our data matrix, due to missing data (usually represented by NaNs). More formally, let $O_{ij} = 1$ if component j of data case i is observed, and let $O_{ij} = 0$ otherwise. Let $\boldsymbol{X}_v = \{x_{ij} : \boldsymbol{O}_{ij} = 1\}$ be the visible data, and $X_h = \{x_{ij} : \boldsymbol{O}_{ij} = 0\}$ be the missing or hidden data. Our goal is to compute

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{X}_v|\boldsymbol{\theta},\boldsymbol{O}) \tag{11.39}$$

Under the missing at random assumption (see Section 8.6.2), we have
TODO

11.6.1 EM for the MLE of an MVN with missing data

TODO