Chapter 7

Modeling complex data densities

In the last chapter we showed that classification with generative models is based on building simple probability models. In particular, we build class-conditional probability distributions $Pr(\mathbf{x}|w=k)$ over the observed data $\mathbf{x}$ for each value of the world state w.

In chapter 3 we introduced several probability distributions that could be used for this purpose but these were quite limited in scope. For example, it is not realistic to assume that all of the complexities of visual data are well described by the normal distribution. In this chapter, we show how to construct complex probability density functions from elementary ones using the idea of a *hidden variable*.

As a representative problem we consider *face detection*; we observe a 60×60 RGB image patch and we would like to decide whether it contains a face or not. To this end, we concatenate the RGB values to form the 10800×1 vector $\mathbf{x}$. Our goal is to take the vector $\mathbf{x}$ and return a label $w \in \{0,1\}$ indicating whether it contains background ($w=0$) or a face ($w=1$). In a real face detection system we would repeat this procedure for every possible sub-window of an image (figure 7.1).

We will start with a basic generative approach in which we describe the likelihood of the data in the presence/absence of a face with a normal distribution. We will then extend this model to address its weaknesses. We emphasize though that state-of-the-art face detection algorithms are *not* based on generative methods such as these; they are usually tackled using the discriminative methods of chapter 9. This application was selected for purely pedagogical reasons.

7.1 Normal classification model

We will take a generative approach to face detection; we will model the probability of the data $\mathbf{x}$ and parameterize this by the world state w. We will describe the data with a multivariate normal distribution so that

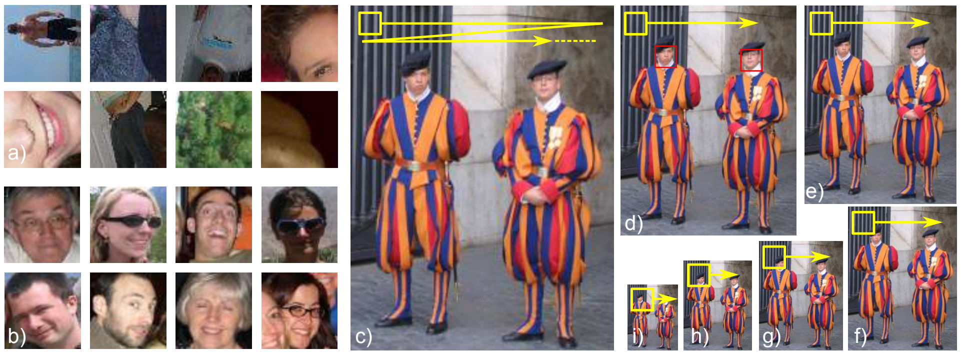


Figure 7.1 Face detection. Consider examining a small window of the image (here 60×60). We concatenate the RGB values in the window to make a data vector $\mathbf{x}$ of dimension 10800×1. The goal of face detection is to infer a label $w \in \{0, 1\}$ indicating whether the window contains (a) a background region ($w\!=\!0$) or (b) an aligned face ($w\!=\!1$). (c-i) We repeat this operation at every position and scale in the image by sweeping a fixed size window through a stack of resized images, estimating w at every point.

$$Pr(\mathbf{x}|w) = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w] \tag{7.1}$$

or treating the two possible values of the state w separately, we can explicitly write

$$\begin{aligned} Pr(\mathbf{x}|w=0) &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0] \\ Pr(\mathbf{x}|w=1) &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1]. \end{aligned} \tag{7.2}$$

These expressions are examples of *class conditional density functions*. They describe the density of the data $\mathbf{x}$ conditional on the value of the world state w.

The goal of learning is to estimate the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\}$ from example pairs of training data $\{\mathbf{x}_i, w_i\}_{i=1}^{I}$. Since parameters $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ are concerned exclusively with background regions (where $w\!=\!0$) we can learn them from the subset of training data $\mathcal{S}_0$ that belonged to the background. For example, using the maximum likelihood approach, we would seek

$$\begin{aligned} \hat{\boldsymbol{\mu}}_0, \hat{\boldsymbol{\Sigma}}_0 &= \underset{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0}{\text{argmax}} \left[\prod_{i \in \mathcal{S}_0} Pr(\mathbf{x}_i | \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \right] \\ &= \underset{\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0}{\text{argmax}} \left[\prod_{i \in \mathcal{S}_0} \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0] \right]. \end{aligned} \tag{7.3}$$

Similarly, $\boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}_1$ are concerned exclusively with faces (where $w = 1$) and can be learned from the subset $\mathcal{S}_1$ of training data which contained faces. Figure 7.2

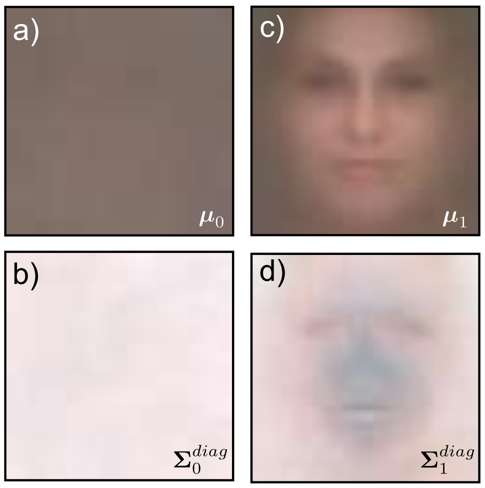

a)
$\boldsymbol{\mu}_0$
c)
$\boldsymbol{\mu}_1$
b)
$\boldsymbol{\Sigma}_0^{diag}$
d)
$\boldsymbol{\Sigma}_1^{diag}$

Figure 7.2 Class conditional density functions for normal model with diagonal covariance. Maximum likelihood fits based on 1000 training examples per class. a) Mean for background data $\boldsymbol{\mu}_0$ (reshaped from 10800×1 vector to 60×60 RGB image). b) Reshaped square root of diagonal covariance for background data $\boldsymbol{\Sigma}_0$. c) Mean for face data $\boldsymbol{\mu}_1$ d) Covariance for face data $\boldsymbol{\Sigma}_1$. The background model has little structure: the mean is uniform and the variance is high everywhere. The mean of the face model clearly captures class-specific information. The covariance of the face is larger at the edges of the image, which usually contain hair or background.

shows the maximum likelihood estimates of the parameters where we have used the diagonal form of the covariance matrix.

The goal of the inference algorithm is to take a new facial image $\mathbf{x}$ and assign a label w to it. To this end, we define a prior over the values of the world state $Pr(w) = \text{Bern}_w[\lambda]$ and apply Bayes' rule

$$Pr(w=1|\mathbf{x}) = \frac{Pr(\mathbf{x}|w=1)Pr(w=1)}{\sum_{k=0}^{1} Pr(\mathbf{x}|w=k)Pr(w=k)}. \tag{7.4}$$

All of these terms are simple to compute and so inference is very easy and will not be discussed further in this chapter.

7.1.1 Deficiencies of the multivariate normal model

Unfortunately, this model does not detect faces reliably. We will defer presenting experimental results until section 7.9.1, but for now please take it on trust that while this model achieves above-chance performance, it doesn't come close to producing a state-of-the-art result. This is hardly surprising: the success of this classifier hinges on fitting the data with a normal distribution. Unfortunately, this fit is poor for three reasons (figure 7.3).

- The normal distribution is unimodal; neither faces nor background regions are well represented by a pdf with a single peak.
- The normal distribution is not robust; a single outlier can dramatically affect the estimates of the mean and covariance.
- The normal distribution has too many parameters; here the data have $D = 10800$ dimensions. The full covariance matrix contains $D(D+1)/2$ parameters. With only 1000 training examples, we cannot even specify these parameters uniquely, so we were forced to use the diagonal form.

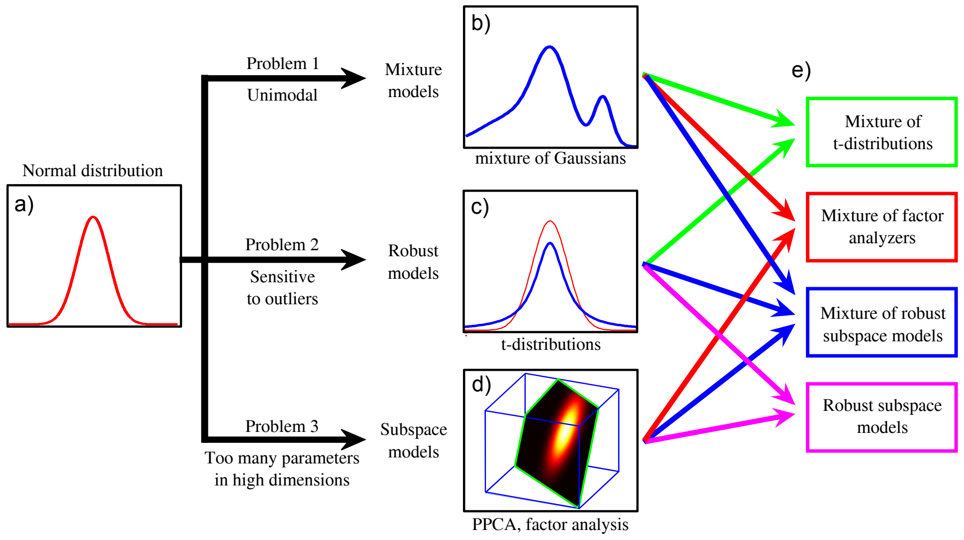


Figure 7.3 a) Problems with the multivariate normal density. b) Normal models are unimodal, but mixtures of Gaussians can model multi-modal distributions. c) Normal distributions are not robust to outliers, but t-distributions can cope with unusual observations. d) Normal models need many parameters in high dimensions but subspace models reduce this requirement. e) These solutions can be combined to form hybrid models addressing several of these problems at once.

We devote the rest of this chapter to tackling these problems. To make the density multi-modal, we introduce *mixture models*. To make the density robust, we replace the normal with the *t-distribution*. To cope with parameter estimation in high dimensions, we introduce *subspace models*.

The new models have much in common with each other. In each case we introduce a *hidden* or *latent variable* $\mathbf{h}_i$ associated with each observed data point $\mathbf{x}_i$. The hidden variable induces the more complex properties of the resulting pdf. Moreover, because the structure of the models is similar, we can use a common approach to learn the parameters.

In the following section, we present an abstract discussion of how hidden variables can be used to model complex pdfs. In section 7.3 we discuss how to learn the parameters of models with hidden variables. Then in sections 7.4, 7.5, and 7.6 we will introduce mixture models, t-distributions, and factor analysis, respectively.

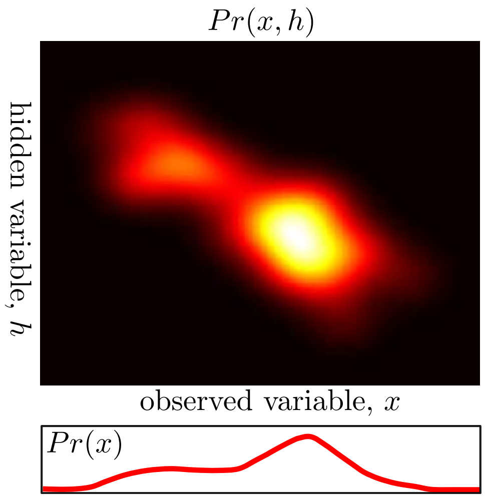


Figure 7.4 Using hidden variables to help model complex densities. One way to model the density $Pr(x)$ is to consider the joint probability distribution $Pr(x,h)$ between the observed data x and a hidden variable h. The density $Pr(x)$ can be considered as the marginalization of (integral over) this distribution with respect to the hidden variable h. As we manipulate the parameters $\boldsymbol{\theta}$ of this joint distribution, the marginal changes and the agreements with the observed data $\{x_i\}_{i=1}^I$ increases or decreases. Sometimes it is easier to fit the distribution in this indirect way than to directly manipulate $Pr(x)$.

7.2 Hidden variables

To model a complex probability density function over the variable $\mathbf{x}$, we will introduce a *hidden* or *latent* variable $\mathbf{h}$, which may be discrete or continuous. We will discuss the continuous formulation, but all of the important concepts transfer to the discrete case.

To exploit the hidden variables, we describe the final density $Pr(\mathbf{x})$ as the marginalization of the joint density $Pr(\mathbf{x},\mathbf{h})$ between $\mathbf{x}$ and $\mathbf{h}$ so that

$$Pr(\mathbf{x}) = \int Pr(\mathbf{x},\mathbf{h})\, d\mathbf{h}. \tag{7.5}$$

We now concentrate on describing the joint density $Pr(\mathbf{x},\mathbf{h})$. We can choose this so that it is relatively simple to model, but produces an expressive family of marginal distributions $Pr(\mathbf{x})$ when we integrate over $\mathbf{h}$ (see figure 7.4).

Whatever form we choose for the joint distribution, it will have some parameters $\boldsymbol{\theta}$, and so really we should write

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \int Pr(\mathbf{x},\mathbf{h}|\boldsymbol{\theta})\, d\mathbf{h}. \tag{7.6}$$

There are two possible approaches to fitting the model to training data $\{\mathbf{x}_i\}_{i=1}^I$ using the maximum likelihood method. We could directly maximize the log likelihood of the distribution $Pr(\mathbf{x})$ from the left hand side of equation 7.6 so that

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_i|\boldsymbol{\theta})\right]\right]. \tag{7.7}$$

This formulation has the advantage that we don't need to involve the hidden variables at all. However, in the models that we will consider, it will not result in a neat closed form solution. Of course we could apply a brute force nonlinear optimization technique (appendix B) but there is an alternative approach: we use the *expectation maximization* algorithm, which works directly with the right-hand side of equation 7.6 and seeks

$$\hat{\boldsymbol\theta} = \underset{\boldsymbol\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[\int Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol\theta)\, d\mathbf{h}_i \right] \right]. \tag{7.8}$$

7.3 Expectation maximization

In this section, we will present a brief description of the *expectation maximization (EM)* algorithm. The goal is to provide just enough information to use this technique for fitting models. We will return to a more detailed treatment in section 7.8.

The EM algorithm is a general-purpose tool for fitting parameters $\boldsymbol\theta$ in models of the form of equation 7.6 where

$$\hat{\boldsymbol\theta} = \underset{\boldsymbol\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[\int Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol\theta)\, d\mathbf{h}_i \right] \right]. \tag{7.9}$$

The EM algorithm works by defining a lower bound $\mathcal{B}\left[\{q_i(\mathbf{h}_i)\}, \boldsymbol\theta\right]$ on the log likelihood in equation 7.9 and iteratively increasing this bound. The lower bound is simply a function that is parameterized by $\boldsymbol\theta$ and some other quantities and is guaranteed to always return a value that is less than or equal to the log likelihood $L[\boldsymbol\theta]$ for any given set of parameters $\boldsymbol\theta$ (figure 7.5).

For the EM algorithm, the particular lower bound chosen is

$$\begin{aligned} \mathcal{B}\left[\{q_i(\mathbf{h}_i)\}, \boldsymbol\theta\right] &= \sum_{i=1}^{I} \int q_i(\mathbf{h}_i) \log\left[\frac{Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol\theta)}{q_i(\mathbf{h}_i)}\right] d\mathbf{h}_i \\ &\leq \sum_{i=1}^{I} \log\left[\int Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol\theta)\, d\mathbf{h}_i\right]. \end{aligned} \tag{7.10}$$

It is not obvious that this inequality is true, making this a valid lower bound; take this on trust for the moment and we will return to this in section 7.8.

In addition to the parameters $\boldsymbol\theta$, the lower bound $\mathcal{B}\left[\{q_i(\mathbf{h}_i)\}, \boldsymbol\theta\right]$ also depends on a set of I probability distributions $\{q_i(\mathbf{h}_i)\}_{i=1}^{I}$ over the hidden variables $\{\mathbf{h}_i\}_{i=1}^{I}$. When we vary these probability distributions, the value that the lower bound returns will change, but it will always remain less than or equal to the log likelihood.

The EM algorithm manipulates both the parameters $\boldsymbol\theta$ and the distributions $\{q_i(\mathbf{h}_i)\}_{i=1}^{I}$ to increase the lower bound. It alternates between:

- updating the probability distributions $\{q_i(\mathbf{h}_i)\}_{i=1}^{I}$ to improve the bound in equation 7.10. This is called the *expectation step* or *E-step*, and
- updating the parameters $\boldsymbol\theta$ to improve the bound in equation 7.10. This is called the *maximization step* or *M-step*.

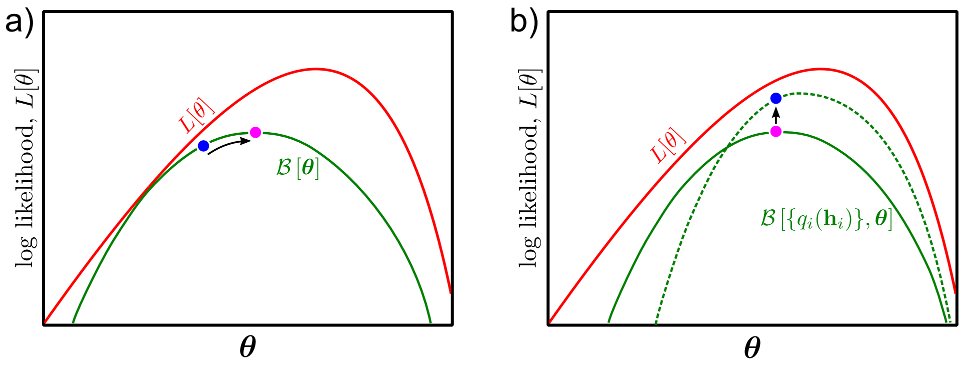


Figure 7.5 Manipulating the lower bound. a) Consider the log likelihood $L[\boldsymbol{\theta}]$ of the data $\{\mathbf{x}\}_{i=1}^{I}$ as a function of the model parameters $\boldsymbol{\theta}$ (red curve). In maximum likelihood learning our goal is to find the parameters $\boldsymbol{\theta}$ that maximize this function. A lower bound on the log likelihood is another function $\mathcal{B}[\boldsymbol{\theta}]$ of the parameters $\boldsymbol{\theta}$ that is everywhere lower or equal to the log likelihood (green curve). One way to improve the current estimate (blue dot) is to manipulate the parameters so that $\mathcal{B}[\boldsymbol{\theta}]$ increases (purple dot). This is the goal of the maximization step of the EM algorithm. b) The lower bound $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$ also depends on a set of probability distributions $\{q_i(\mathbf{h}_i)\}_{i=1}^{I}$ over hidden variables $\{\mathbf{h}_i\}$. Manipulating these probability distributions changes the value that the lower bound returns for every $\boldsymbol{\theta}$ (e.g., green curve). So a second way to improve the current estimate (purple dot) is to change the distributions in such a way that the curve increases for the current parameters (blue dot). This is the goal of the expectation step of the EM algorithm.

In the E-step at iteration t+1 we set each distribution $q_i(\mathbf{h}_i)$ to be the posterior distributions $Pr(\mathbf{h}_i|\mathbf{x}_i, \boldsymbol{\theta})$ over that hidden variable given the associated data example and the current parameters $\boldsymbol{\theta}^{[t]}$. To compute these we use Bayes' rule

$$\hat{q}_i(\mathbf{h}_i) = Pr(\mathbf{h}_i|\mathbf{x}_i, \boldsymbol{\theta}^{[t]}) = \frac{Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}^{[t]})Pr(\mathbf{h}_i|\boldsymbol{\theta}^{[t]})}{Pr(\mathbf{x}_i)}. \tag{7.11}$$

It can be shown that this choice maximizes the bound as much as possible.

In the M-step we directly maximize the bound (equation 7.10) with respect to the parameters $\boldsymbol{\theta}$. In practice we can simplify the expression for the bound to eliminate terms that do not depend on $\boldsymbol{\theta}$ and this yields

$$\hat{\boldsymbol{\theta}}^{[t+1]} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \int \hat{q}_i(\mathbf{h}_i) \log\left[Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol{\theta})\right] \, d\mathbf{h}_i \right]. \tag{7.12}$$

Each of these steps is guaranteed to improve the bound, and iterating them alternately is guaranteed to find at least a local maximum with respect to $\boldsymbol{\theta}$.

This is a practical description of the EM algorithm, but there is a lot missing: we have not demonstrated that equation 7.10 really is a bound on the log likelihood.

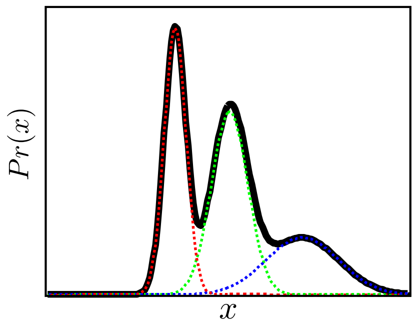


Figure 7.6 Mixture of Gaussians model in 1D. A complex multimodal probability density function (black solid curve) is created by taking a weighted sum or *mixture* of several constituent normal distributions with different means and variances (red, green and blue dashed curves). To ensure that the final distribution is a valid density, the weights must be positive and sum to one.

We have not shown that the posterior distribution $Pr(\mathbf{h}_i|\mathbf{x}_i, \boldsymbol{\theta}^{[t]})$ is the optimal choice for $q_i(\mathbf{h}_i)$ in the E-step (equation 7.11), and we have not demonstrated that the cost function for the M-step (equation 7.12) improves the bound. For now we will assume that these things are true and proceed with the main thrust of the chapter. We will return to these issues in section 7.8.

7.4 Mixture of Gaussians

The *mixture of Gaussians* (MoG) is a prototypical example of a model where learning is suited to the EM algorithm. The data are described as a weighted sum of K normal distributions

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda_k \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k], \tag{7.13}$$

where $\boldsymbol{\mu}_{1\ldots K}$ and $\boldsymbol{\Sigma}_{1\ldots K}$ are the means and covariances of the normal distributions and $\lambda_{1\ldots K}$ are positive valued weights that sum to one. The mixtures of Gaussians model describes complex multi-modal probability densities by combining simpler constituent distributions (figure 7.6).

To learn the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \lambda_k\}_{k=1}^{K}$ from training data $\{\mathbf{x}_i\}_{i=1}^{I}$ we could apply the straightforward maximum likelihood approach

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\text{argmax}} \left[\sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_i|\boldsymbol{\theta})\right]\right] \\ &= \underset{\boldsymbol{\theta}}{\text{argmax}} \left[\sum_{i=1}^{I} \log\left[\sum_{k=1}^{K} \lambda_k \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]\right]\right].\end{aligned} \tag{7.14}$$

Unfortunately, if we take the derivative with respect to the parameters $\boldsymbol{\theta}$ and equate the resulting expression to zero, it is not possible to solve the resulting equations in closed form. The sticking point is the summation inside the logarithm, which precludes a simple solution. Of course, we could use a nonlinear optimization approach, but this would be complex as we would have to maintain the constraints

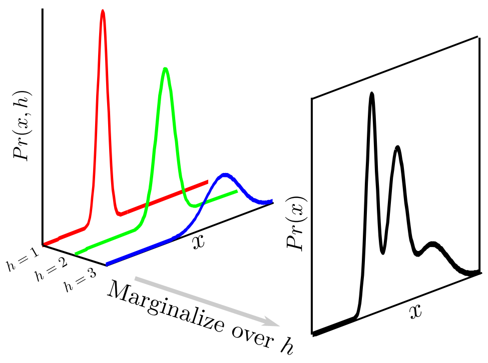


Figure 7.7 Mixture of Gaussians as a marginalization. The mixture of Gaussians can also be thought of in terms of a joint distribution $Pr(x, h)$ between the observed variable x and a discrete hidden variable h. To create the mixture density we marginalize over h. The hidden variable has a straightforward interpretation: it is the index of the constituent normal distribution.

on the parameters; the weights $\boldsymbol{\lambda}$ must sum to one and the covariances $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ must be positive definite. For a simpler approach, we express the observed density as a marginalization and use the EM algorithm to learn the parameters.

7.4.1 Mixture of Gaussians as a marginalization

The mixture of Gaussians model can be expressed as the marginalization of a joint probability distribution between the observed data $\mathbf{x}$ and a discrete hidden variable h that takes values $h \in \{1 \ldots K\}$ (figure 7.7). If we define

$$\begin{aligned} Pr(\mathbf{x}|h,\boldsymbol{\theta}) &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h] \\ Pr(h|\boldsymbol{\theta}) &= \text{Cat}_h[\boldsymbol{\lambda}], \end{aligned} \tag{7.15}$$

where $\boldsymbol{\lambda} = [\lambda_1 \ldots \lambda_K]$ are the parameters of the categorical distribution, then we can recover the original density using

$$\begin{aligned} Pr(\mathbf{x}|\boldsymbol{\theta}) &= \sum_{k=1}^{K} Pr(\mathbf{x}, h=k|\boldsymbol{\theta}) \\ &= \sum_{k=1}^{K} Pr(\mathbf{x}|h=k,\boldsymbol{\theta}) Pr(h=k|\boldsymbol{\theta}) \\ &= \sum_{k=1}^{K} \lambda_k \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]. \end{aligned} \tag{7.16}$$

Interpreting the model in this way also provides a method to draw samples from a mixture of Gaussians: we sample from the joint distribution $Pr(\mathbf{x}, h)$, and then discard the hidden variable h to leave just a data sample $\mathbf{x}$. To sample from the joint distribution $Pr(\mathbf{x}, h)$ we first sample h from the categorical prior $Pr(h)$, then sample $\mathbf{x}$ from the normal distribution $Pr(\mathbf{x}|h)$ associated with the value of h. Notice that the hidden variable h has a clear interpretation in this procedure; it determines which of the constituent normal distributions is *responsible* for the observed data point $\mathbf{x}$.

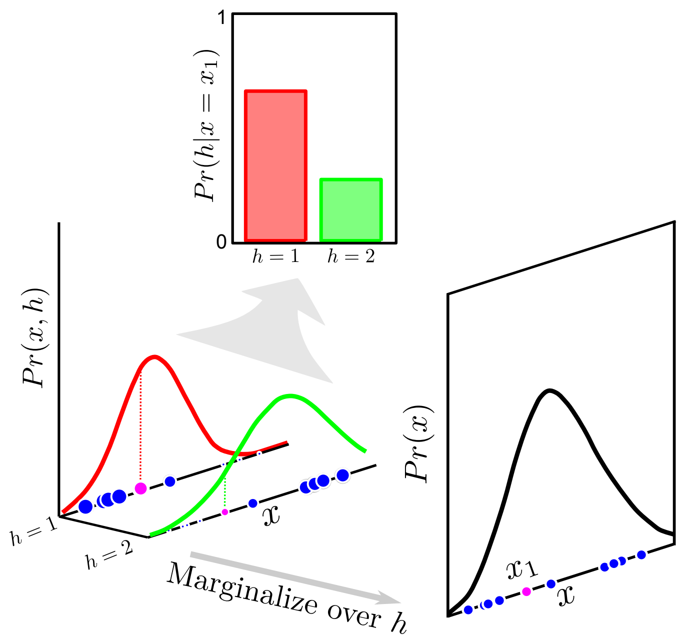


Figure 7.8 E-step for fitting the mixture of Gaussians model. For each of the I data points $\mathbf{x}_{1...I}$, we calculate the posterior distribution $Pr(h_i|\mathbf{x}_i)$ over the hidden variable h_i. The posterior probability $Pr(h_i = k|\mathbf{x}_i)$ that h_i takes value k can be understood as the responsibility of normal distribution k for data point x_i. For example, for data point x_1 (magenta circle), component 1 (red curve) is more than twice as likely to be responsible than component 2 (green curve). Note that in the joint distribution (left), the size of the projected data point indicates the responsibility.

7.4.2 Expectation maximization for fitting mixture models

Algorithm 7.1

To learn the MoG parameters $\boldsymbol{\theta} = \{\lambda_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ from training data $\{\mathbf{x}_i\}_{i=1}^{I}$ we apply the EM algorithm. Following the recipe of section 7.3, we initialize the parameters randomly and alternate between performing the E- and M-steps.

In the E-step, we maximize the bound with respect to the distributions $q_i(h_i)$ by finding the posterior probability distribution $Pr(h_i|\mathbf{x}_i)$ of each hidden variable h_i given the observation $\mathbf{x}_i$ and the current parameter settings,

$$\begin{aligned} q_i(h_i) = Pr(h_i = k|\mathbf{x}_i, \boldsymbol{\theta}^{[t]}) &= \frac{Pr(\mathbf{x}_i|h_i = k, \boldsymbol{\theta}^{[t]})Pr(h_i = k, \boldsymbol{\theta}^{[t]})}{\sum_{j=1}^{K} Pr(\mathbf{x}_i|h_i = j, \boldsymbol{\theta}^{[t]})Pr(h_i = j, \boldsymbol{\theta}^{[t]})} \\ &= \frac{\lambda_k \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]}{\sum_{j=1}^{K} \lambda_j \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j]} \\ &= r_{ik}. \end{aligned} \tag{7.17}$$

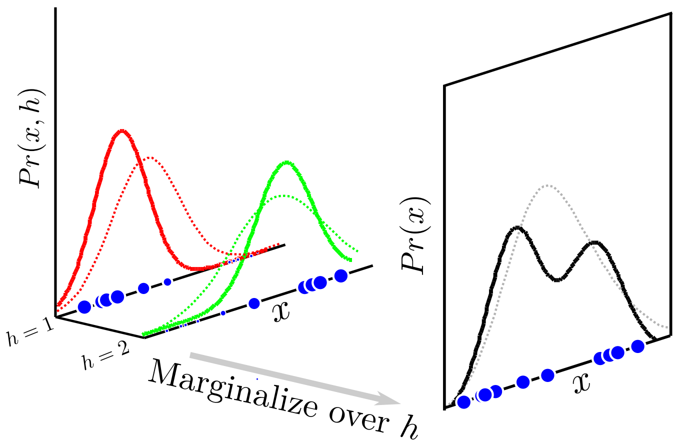


Figure 7.9 M-step for fitting the mixture of Gaussians model. For the k^{th} constituent Gaussian, we update the parameters $\{\boldsymbol{\lambda}_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$. The i^{th} data point $\mathbf{x}_i$ contributes to these updates according to the responsibility r_{ik} (indicated by size of point) assigned in the E-step; data points that are more associated with the k^{th} component have more effect on the parameters. Dashed and solid lines represent fit before and after update, respectively.

In other words we compute the probability $Pr(h_i = k|\mathbf{x}_i, \boldsymbol{\theta}^{[t]})$ that the k^{th} normal distribution was responsible for the i^{th} data point (figure 7.8). We denote this *responsibility* by r_{ik} for short.

In the M-step, we maximize the bound with respect to the parameters $\boldsymbol{\theta} = \{\lambda_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ so that

$$\begin{aligned}\hat{\boldsymbol{\theta}}^{[t+1]} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[\sum_{i=1}^{I}\sum_{k=1}^{K}\hat{q}_i(h_i=k)\log\left[Pr(\mathbf{x}_i, h_i=k|\boldsymbol{\theta})\right]\right]\\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[\sum_{i=1}^{I}\sum_{k=1}^{K}r_{ik}\log\left[\lambda_k \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]\right]\right].\end{aligned} \tag{7.18}$$

This maximization can be performed by taking the derivative of the expression with respect to the parameters, equating the result to zero and rearranging, taking care to enforce the constraint $\sum_k \lambda_k = 1$ using Lagrange multipliers. The procedure results in the update rules: Problem 7.3

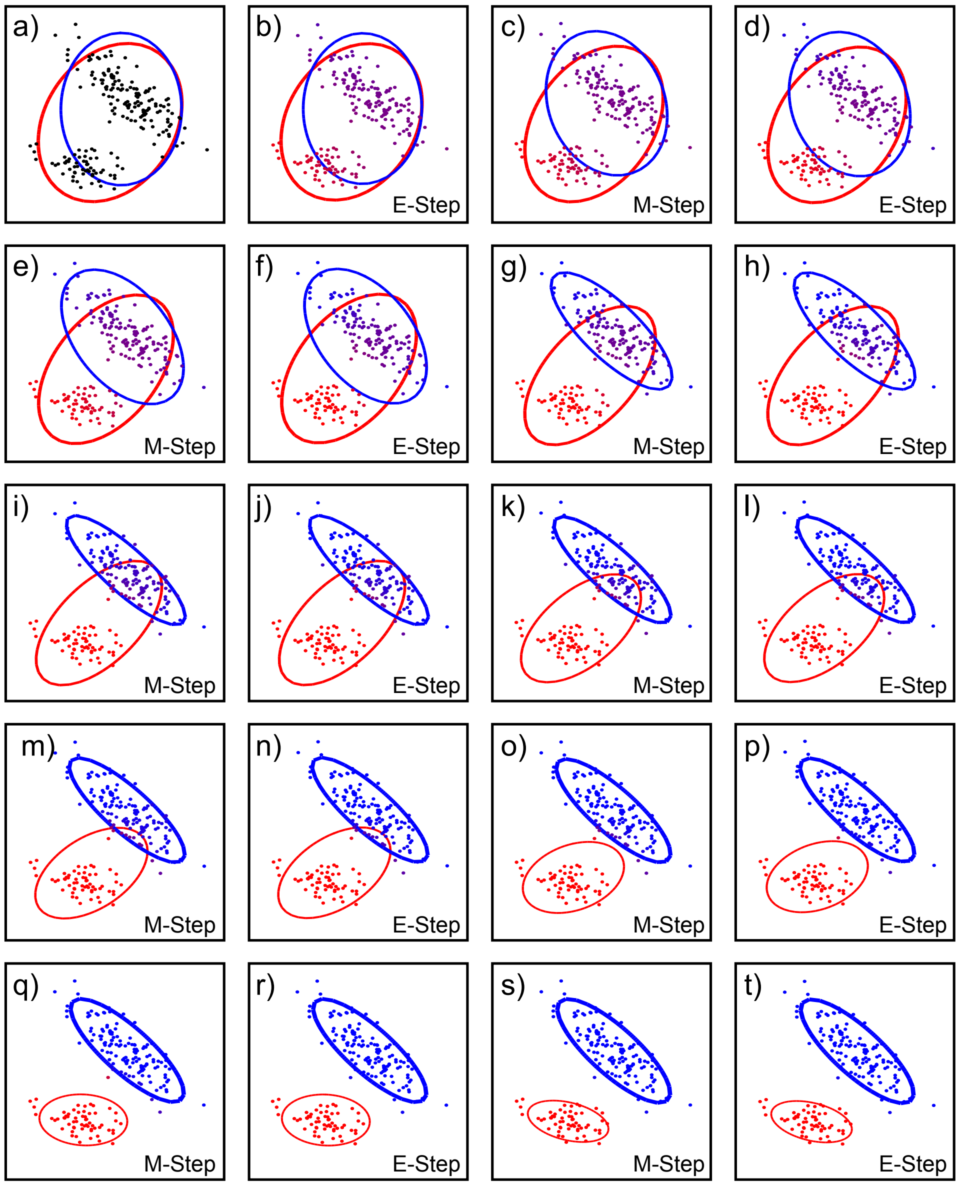


Figure 7.10 a) Initial model. b) E-step. For each data point the posterior probability that is was generated from each Gaussian is calculated (indicated by color of point). c) M-step. The mean, variance and weight of each Gaussian is updated based on these posterior probabilities. Ellipse shows Mahalanobis distance of two. Weight (thickness) of ellipse indicates weight of Gaussian. d-t) Further E-step and M-step iterations.

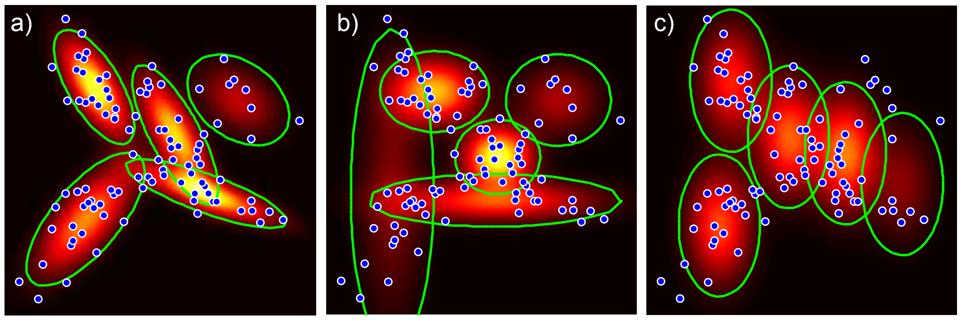


Figure 7.11 Covariance of components in mixture models. a) Full covariances. b) Diagonal covariances. c) Identical diagonal covariances.

$$\begin{aligned}
\lambda_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K}\sum_{i=1}^{I} r_{ij}} \\
\boldsymbol{\mu}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}\mathbf{x}_i}{\sum_{i=1}^{I} r_{ik}} \\
\boldsymbol{\Sigma}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}(\mathbf{x}_i-\boldsymbol{\mu}_k^{[t+1]})(\mathbf{x}_i-\boldsymbol{\mu}_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}.
\end{aligned} \tag{7.19}$$

These update rules can be easily understood (figure 7.9): we update the weights $\{\lambda_k\}_{k=1}^K$ according to the relative total responsibility of each component for the data points. We update the cluster means $\{\boldsymbol{\mu}_k\}_{k=1}^K$ by computing the weighted mean over the data points where the weights are given by the responsibilities. If component k is mostly responsible for data point $\mathbf{x}_i$, then this data point has a high weight and affects the update more. The update rule for the covariances has a similar interpretation.

In practice the E- and M-steps are alternated until the bound on the data no longer increases and the parameters no longer change. The alternating E-steps and M-steps for a two dimensional example are shown in figure 7.10. Notice that the final fit identifies the two *clusters* in the data. The mixture of Gaussians is closely related to clustering techniques such as the *K-means* algorithm (section 13.4.4).

The EM approach to estimating mixture models has three attractive features.

1. Both steps of the algorithm can be computed in closed form without the need for an optimization procedure.
2. The solution guarantees that the constraints on the parameters are respected: the weighting parameters $\{\lambda_k\}_{k=1}^K$ are guaranteed to be positive and sum to one, and the covariance matrices $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$ are guaranteed to be positive definite.

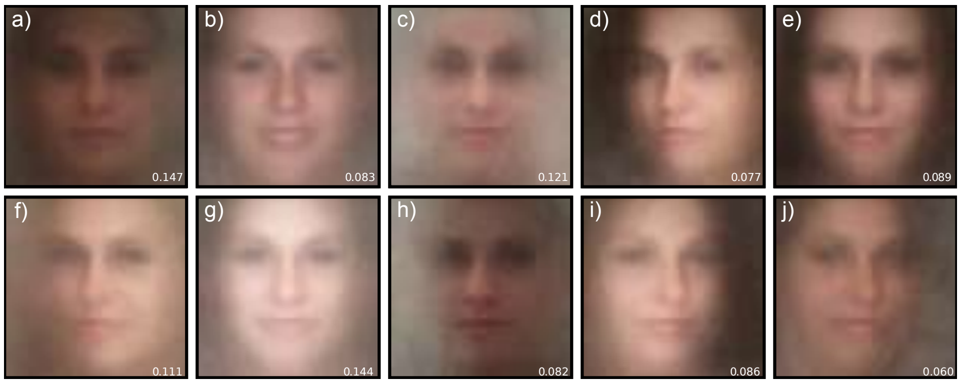


Figure 7.12 Mixtures of Gaussians model for face data. a-j) Mean vectors $\boldsymbol{\mu}_k$ for a mixture of ten Gaussians fitted to the face data set. The model has captured variation in the mean luminance and chromaticity of the face and other factors such as the pose and background color. Numbers indicate the weight of each component.

3. The method can cope with missing data. Imagine that some of the elements of training example $\mathbf{x}_i$ are missing. In the E-step, the remaining dimensions can still be used to establish a distribution over the hidden variable h. In the M-step, this data point would contribute only to the dimensions of $\{\boldsymbol{\mu}_k\}_{k=1}^K$ and $\{\boldsymbol{\Sigma}_k\}_{k=1}^K$where data were observed.

Figure 7.11 shows a mixture of five Gaussians that has been fit to a 2D data set. As for the basic multivariate normal model, it is possible to constrain the covariance matrices to be spherical or diagonal. We can also constrain the covariances to be the same for each component if desired. Figure 7.12 shows the mean vectors $\boldsymbol{\mu}_k$ for a ten component model with diagonal covariances fitted to the face data set. The clusters represent different illumination conditions as well as changes in pose, expression, and background color.

In fitting mixtures of Gaussians there are several things to consider. First, the EM algorithm does not guarantee to find a global solution to this non-convex optimization problem. Figure 7.13 shows three different solutions that were computed by starting the fitting algorithm with different initial random values for the parameters $\boldsymbol{\theta}$. The best we can do to circumvent this problem is to start fitting in different places and take the solution with the greatest log likelihood. Second, we must pre-specify the number of mixing components. Unfortunately, we cannot decide the number of components by comparing the log likelihood; models with more parameters will inevitably describe the data better. There are methods to tackle this problem, but they are beyond the scope of this volume.

Finally, although we presented a maximum likelihood approach here, it is important in practice to include priors over model parameters $Pr(\boldsymbol{\theta})$ to prevent the scenario where one of the Gaussians becomes exclusively associated with a single data point. Without a prior, the variance of this component becomes progressively

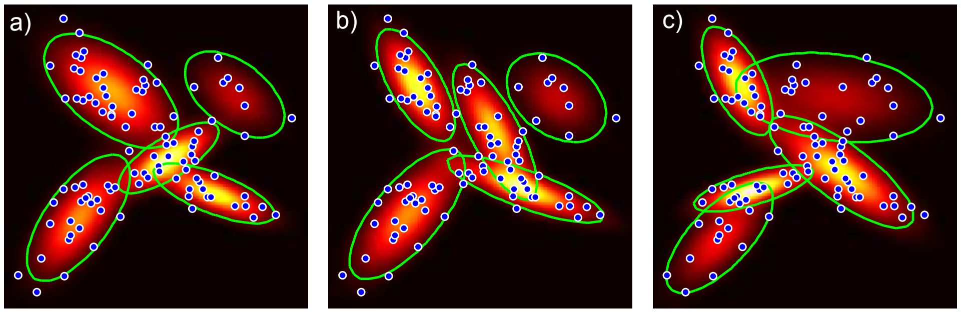


Figure 7.13 Local maxima. Repeated fitting of mixture of Gaussians model with different starting points results in different models as the fit converges to different local maxima. The log likelihoods are a) 98.76 b) 96.97 c) 94.35, respectively, indicating that (a) is the best fit.

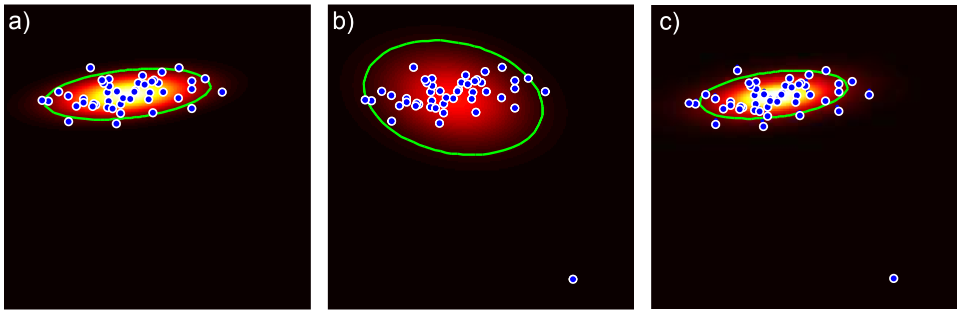


Figure 7.14 Motivation for t-distribution. a) The multivariate normal model fit to data. b) Adding a single outlier completely changes the fit. c) With the multivariate t-distribution the outlier does not have such a drastic effect.

smaller and the likelihood increases without bound.

7.5 The t-distribution

The second significant problem with using the normal distribution to describe visual data is that it is not robust: the height of the normal pdf falls off very rapidly as we move into the tails. The effect of this is that outliers (unusually extreme observations) drastically affect the estimated parameters (figure 7.14). The t-distribution is a closely related distribution in which the length of the tails is parameterized.

The univariate t-distribution (figure 7.15) has probability density function

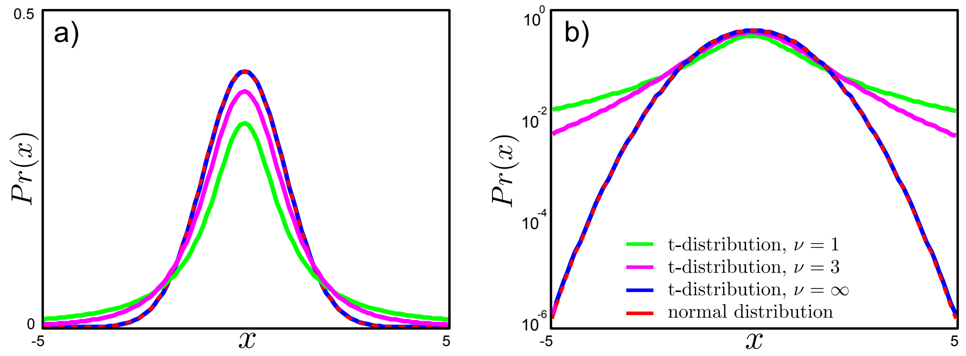


Figure 7.15 a) As well as the mean μ and scaling parameter σ^2, the t-distribution has a parameter ν which is termed the degrees of freedom. As ν decreases, the tails of the distribution become longer and the model becomes more robust. b) This is seen more clearly on a log scale.

$$\begin{aligned} Pr(x) &= \text{Stud}_{\mathbf{x}}\left[\mu, \sigma^2, \nu\right] \\ &= \frac{\Gamma\left[\frac{\nu+1}{2}\right]}{\sqrt{\nu\pi\sigma^2}\,\Gamma\left[\frac{\nu}{2}\right]}\left(1+\frac{(x-\mu)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}, \end{aligned} \tag{7.20}$$

where μ is the mean and σ^2 is the scale parameter. The degrees of freedom $\nu \in (0, \infty]$ controls the length of the tails: when ν is small there is considerable weight in the tails. For example, with $\mu = 0$ and $\sigma^2 = 1$ a data point at $x = -5$ is roughly $10^4 = 10000$ times more likely under the t-distribution with $\nu = 1$ than under the normal distribution. As ν tends to infinity, the distribution approximates a normal more and more closely and there is less weight in the tails. The variance of the distribution is given by $\sigma\nu/(\nu-2)$ for $\nu > 2$ and infinite if $0 < \nu \leq 2$.

The multivariate t-distribution has pdf

$$\begin{aligned} Pr(\mathbf{x}) &= \text{Stud}_{\mathbf{x}}\left[\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu\right] \\ &= \frac{\Gamma\left[\frac{\nu+D}{2}\right]}{(\nu\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}\Gamma\left[\frac{\nu}{2}\right]}\left(1+\frac{(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}{\nu}\right)^{-\frac{\nu+D}{2}}, \end{aligned} \tag{7.21}$$

where D is the dimensionality of the space, $\boldsymbol{\mu}$ is a $d \times 1$ mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ positive definite scale matrix, and $\nu \in [0, \infty]$ is the degrees of freedom. As for the multivariate normal distribution (figure 5.1), the scale matrix can take full, diagonal or spherical forms. The covariance of the distribution is given by $\boldsymbol{\Sigma}\nu/(\nu-2)$ for $\nu > 2$ and is infinite if $0 \leq \nu \leq 2$.

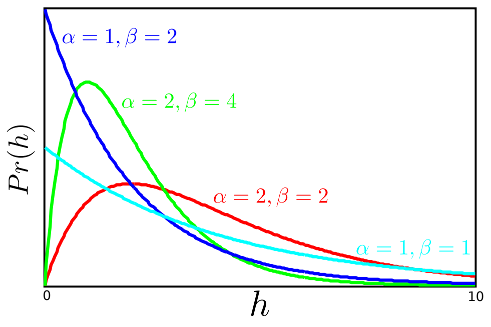


Figure 7.16 The gamma distribution is defined on positive real values and has two parameters α, β. The mean of the distribution is $\text{E}[h] = \alpha/\beta$ and the variance is $\text{E}[(h-\text{E}[h])^2] = \alpha/\beta^2$. The t-distribution can be thought of as a weighted sum of normal distributions with the same mean, but covariances that depend inversely on the gamma distribution.

7.5.1 Student t-distribution as a marginalization

As for the mixtures of Gaussians, it is also possible to understand the t-distribution in terms of hidden variables. We define

Problem 7.5
Problem 7.6

$$\begin{aligned} Pr(\mathbf{x}|h) &= \text{Norm}_x[\boldsymbol{\mu}, \boldsymbol{\Sigma}/h] \\ Pr(h) &= \text{Gam}_h[\nu/2, \nu/2], \end{aligned} \tag{7.22}$$

where h is a scalar hidden variable and $\text{Gam}[\alpha, \beta]$ is the gamma distribution with parameters α, β (figure 7.16). The gamma distribution is a continuous probability distribution defined on the positive real axis with probability density function

$$\text{Gam}_h[\alpha, \beta] = \frac{\beta^\alpha}{\Gamma[\alpha]} \exp[-\beta h] h^{\alpha-1}, \tag{7.23}$$

where $\Gamma[\bullet]$ is the gamma function.

The t-distribution is the marginalization with respect to the hidden variable h of the joint distribution between the data $\mathbf{x}$ and h (figure 7.17),

$$\begin{aligned} Pr(\mathbf{x}) = \int Pr(\mathbf{x}, h) dh &= \int Pr(\mathbf{x}|h) Pr(h) dh \\ &= \int \text{Norm}_x[\boldsymbol{\mu}, \boldsymbol{\Sigma}/h] \text{Gam}_h[\nu/2, \nu/2] dh \\ &= \text{Stud}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu]. \end{aligned} \tag{7.24}$$

This formulation also provides a method to generate data from the t-distribution; we first generate h from the gamma distribution and then generate $\mathbf{x}$ from the associated normal distribution $Pr(\mathbf{x}|h)$. Hence the hidden variable has a simple interpretation: it tells us which one of the continuous family of underlying normal distributions was responsible for this data point.

7.5.2 Expectation maximization for fitting t-distributions

Since the pdf takes the form of a marginalization of the joint distribution with a hidden variable (equation 7.24), we can use the EM algorithm to learn the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu\}$ from a set of training data $\{\mathbf{x}_i\}_{i=1}^I$.

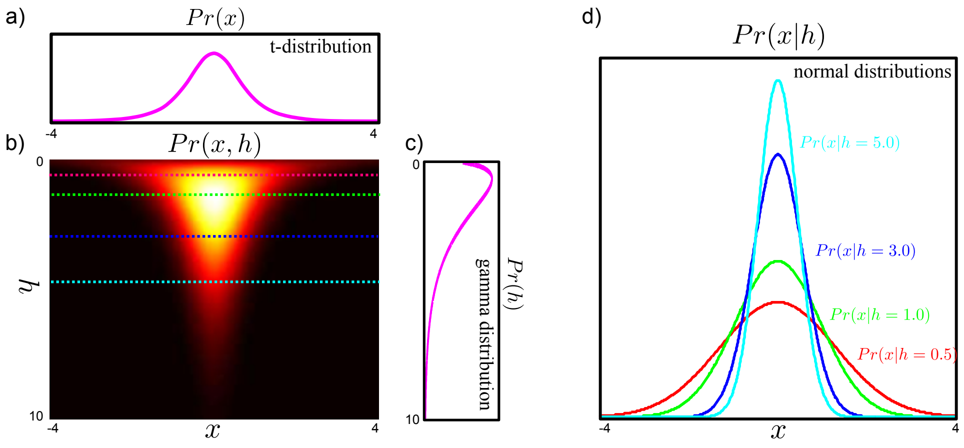


Figure 7.17 a) The t-distribution has a similar form to the normal distribution but longer tails. b) The t-distribution is the marginalization of the joint distribution $Pr(x,h)$ between the observed variable x and a hidden variable h. c) The prior distribution over the hidden variable h has a gamma distribution. d) The conditional distribution $Pr(x|h)$ is normal with a variance that depends on h. So the t-distribution can be considered as an infinite weighted sum of normal distributions with variances determined by the gamma prior (equation 7.24).

Algorithm 7.2

Problem 7.7

In the E-step (figure 7.18a-b) we maximize the bound with respect to the distributions $q_i(h_i)$ by finding the posterior $Pr(h_i|\mathbf{x}_i,\boldsymbol{\theta}^{[t]})$ over each hidden variable h_i given associated observation $\mathbf{x}_i$ and the current parameter settings. By Bayes' rule, we get

$$\begin{aligned} q_i(h_i) = Pr(h_i|\mathbf{x}_i,\boldsymbol{\theta}^{[t]}) &= \frac{Pr(\mathbf{x}_i|h_i,\boldsymbol{\theta}^{[t]})Pr(h_i)}{Pr(\mathbf{x}_i|\boldsymbol{\theta}^{[t]})} \\ &= \frac{\text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu},\boldsymbol{\Sigma}/h_i]\text{Gam}_{h_i}[\nu/2,\nu/2]}{Pr(\mathbf{x}_i)} \\ &= \text{Gam}_{h_i}\left[\frac{\nu+D}{2},\frac{(\mathbf{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})}{2}+\frac{\nu}{2}\right], \end{aligned} \tag{7.25}$$

where we have used the fact that the gamma distribution is conjugate to the scaling factor for the normal variance. The E-step can be understood as follows: we are treating each data point $\mathbf{x}_i$ as if it were generated from one of the normals in the infinite mixture where the hidden variable h_i determines which normal. So, the E-step computes a distribution over h_i, which hence determines a distribution over which normal created the data.

We now compute the following expectations (section 2.7) with respect to the distribution in equation 7.25:

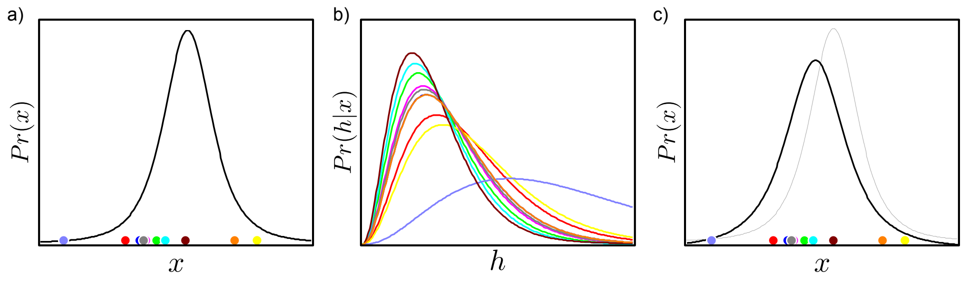


Figure 7.18 Expectation maximization for fitting t-distributions. a) Estimate of distribution before update. b) In the E-step we calculate the posterior distribution $Pr(h_i|x_i)$ over the hidden variable h_i for each data point x_i. The color of each curve corresponds to that of the original data point in (a). c) In the M-step we use these distributions over h to update the estimate of the parameters $\theta = \{\mu, \sigma^2, \nu\}$.

$$\begin{aligned} \mathrm{E}[h_i] &= \frac{(\nu + D)}{\nu + (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})} & (7.26)\\ \mathrm{E}[\log[h_i]] &= \Psi\left[\frac{\nu + D}{2}\right] - \log\left[\frac{\nu + (\mathbf{x}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})}{2}\right], & (7.27) \end{aligned}$$

where $\Psi[\bullet])$ is the *digamma* function. These will be needed in the E-step.

In the M-step (figure 7.18c) we maximize the bound with respect to the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu\}$ so that

$$\begin{aligned} \hat{\boldsymbol{\theta}}^{[t+1]} &= \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \left[\sum_{i=1}^{I} \int \hat{q}_i(h_i) \log\left[Pr(\mathbf{x}_i, h_i|\boldsymbol{\theta})\right] dh_i\right]\\ &= \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \left[\sum_{i=1}^{I} \int \hat{q}_i(h_i) \left(\log\left[Pr(\mathbf{x}_i|h_i, \boldsymbol{\theta})\right] + \log\left[Pr(h_i)\right]\right) dh_i\right]\\ &= \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \left[\sum_{i=1}^{I} \int Pr(h_i|\mathbf{x}_i, \boldsymbol{\theta}^{[t]}) \left(\log\left[Pr(\mathbf{x}_i|h_i, \boldsymbol{\theta})\right] + \log\left[Pr(h_i)\right]\right) dh_i\right]\\ &= \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \left[\sum_{i=1}^{I} E\left[\log\left[Pr(\mathbf{x}_i|h_i, \boldsymbol{\theta})\right]\right] + E\left[\log\left[Pr(h_i)\right]\right]\right], & (7.28) \end{aligned}$$

where the expectation is taken relative to the posterior distribution $Pr(h_i|\mathbf{x}_i, \boldsymbol{\theta}^{[t]})$. Substituting in the expressions for the normal likelihood $Pr(\mathbf{x}_i|h_i)$ and the gamma prior $Pr(h_i)$, we find that

$$\begin{aligned} E\left[\log\left[Pr(\mathbf{x}_i|h_i,\boldsymbol{\theta})\right]\right] &= \frac{D\mathrm{E}[\log h_i]-D\log 2\pi-\log|\boldsymbol{\Sigma}|-(\mathbf{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\mathrm{E}[h_i]}{2}\\ E\left[\log\left[Pr(h_i)\right]\right] &= \frac{\nu}{2}\log\left[\frac{\nu}{2}\right]-\log\Gamma\left[\frac{\nu}{2}\right]+\left(\frac{\nu}{2}-1\right)\mathrm{E}[\log h_i]-\frac{\nu}{2}\mathrm{E}[h_i]. \end{aligned} \tag{7.29}$$

To optimize $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we take derivatives of equation 7.28, set the resulting expressions to zero and rearrange to yield update equations

$$\begin{aligned} \boldsymbol{\mu}^{[t+1]} &= \frac{\sum_{i=1}^{I}\mathrm{E}[h_i]\mathbf{x}_i}{\sum_{i=1}^{I}\mathrm{E}[h_i]}\\ \boldsymbol{\Sigma}^{[t+1]} &= \frac{\sum_{i=1}^{I}\mathrm{E}[h_i](\mathbf{x}_i-\boldsymbol{\mu}^{[t+1]})(\mathbf{x}_i-\boldsymbol{\mu}^{[t+1]})^T}{\sum_{i=1}^{I}\mathrm{E}[h_i]}. \end{aligned} \tag{7.30}$$

These update equations have an intuitive form: for the mean, we are computing a weighted sum of the data. Outliers in the data set will tend to be explained best by the normal distributions in the infinite mixture which have larger covariances: for these distributions h is small (h scales the normal covariance inversely). Consequently, $\mathrm{E}[h]$ is small, and they are weighted less in the sum. The update for the $\boldsymbol{\Sigma}$ has a similar interpretation.

Unfortunately, there is no closed form solution for the degrees of freedom ν. We hence perform a one-dimensional line search to maximize equation 7.28 having substituted in the updated values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, or use one of the optimization techniques described in chapter 9.

When we fit a t-distribution with a diagonal scale matrix $\boldsymbol{\Sigma}$ to the face data set, the mean $\boldsymbol{\mu}$ and scale matrix $\boldsymbol{\Sigma}$ (not shown) look visually similar to those for the normal model (figure 7.2). However, the model is not the same. The fitted degrees of freedom ν is 6.6. This low value indicates that the distribution has significantly longer tails than the normal model.

In conclusion, the multivariate t-distribution provides an improved description of data with outliers (figure 7.14). It has just one more parameter than the normal (the degrees of freedom, ν), and subsumes the normal as a special case (where ν becomes very large). However, this generality comes at a cost: there is no closed form solution for the maximum likelihood parameters and so we must resort to more complex approaches such as the EM algorithm to fit the distribution.

7.6 Factor analysis

We now address the final problem with the normal distribution. Visual data are often very high dimensional; in the face detection task the data comes in the form of 60×60 RGB images and is hence characterized as a $60\times60\times3=10800$ dimensional vector. To model this data with the full multivariate normal distribution, we

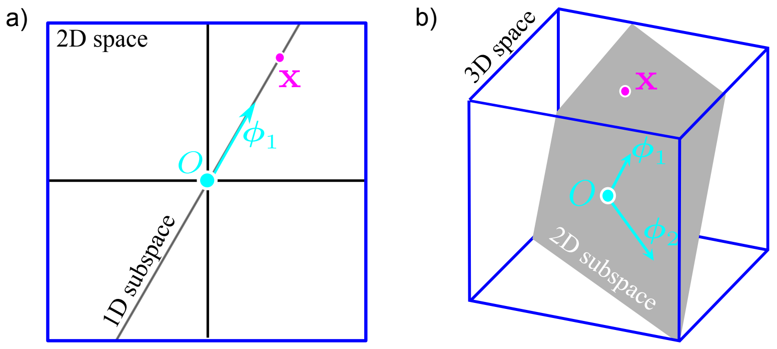


Figure 7.19 Linear subspaces a) A one dimensional subspace (a line through the origin, O) is embedded in a two dimensional space. Any point $\mathbf{x}$ in the subspace can be reached by weighting the single basis vector $\boldsymbol{\phi}_1$ appropriately. b) A two dimensional subspace (a plane through the origin, O) is embedded in a three dimensional space. Any point $\mathbf{x}$ in the subspace can be reached using a linear combination $\mathbf{x} = \alpha\boldsymbol{\phi}_1 + \beta\boldsymbol{\phi}_2$ of the two basis functions $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$ that describe the subspace. In general a K-dimensional subspace can be described using K basis functions.

require a covariance matrix of dimensions 10800×10800: we would need a very large number of training examples to get good estimates of all of these parameters in the absence of prior information. Furthermore, to store the covariance matrix we will need a large amount of memory, and there remains the problem of inverting this large matrix when we evaluate the normal likelihood (equation 5.1).

Of course, we could just use the diagonal form of the covariance matrix which contains only 10800 parameters. However, this is too great a simplification: we are assuming that each dimension of the data is independent and for face images this is clearly not true. For example, in the cheek region, the RGB values of neighboring pixels covary very closely. A good model should capture this information.

Factor analysis provides a compromise in which the covariance matrix is structured so that it contains fewer unknown parameters than the full matrix but more than the diagonal form. One way to think about the covariance of a factor analyzer is that it models part of the high-dimensional space with a full model and mops up remaining variation with a diagonal model.

More precisely, the factor analyzer describes a *linear subspace* with a full covariance model. A linear subspace is a subset of a high dimensional space that can be reached by taking linear combinations (weighted sums) of a fixed set of basis functions (figure 7.19). So, a line through the origin is a subspace in two dimensions as we can reach any point on it by weighting a single basis vector. A line through the origin is also a subspace in three dimensions, but so is a plane through the origin: we can reach any point on the plane by taking linear combinations of two basis vectors. In general, a D-dimensional space contains subspaces of dimensions $1, 2, \ldots, D-1$.

The probability density function of a factor analyzer is given by

$$Pr(\mathbf{x}) = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}], \tag{7.31}$$

where the covariance matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}$ contains a sum of two terms. The first term $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ describes a full covariance model over the subspace: the K columns of the portrait[1] rectangular matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_K]$ are termed *factors*. The factors are basis vectors that determine the subspace modeled. When we fit the model to data the factors will span the set of directions where the data covary the most. The second term $\boldsymbol{\Sigma}$ is a diagonal matrix that accounts for all remaining variation.

Notice that this model has $K \times D$ parameters to describe $\boldsymbol{\Phi}$ and another D parameters to describe the diagonal matrix $\boldsymbol{\Sigma}$. If the number of factors K is much less than the dimensionality of the data D then this model has fewer parameters than a normal with full covariance and hence can be learned from fewer training examples.

When $\boldsymbol{\Sigma}$ is a constant multiple of the identity matrix (i.e., models spherical covariance) the model is called *probabilistic principal component analysis*. This simpler model has slightly fewer parameters and can be fit in closed form (i.e., without the need for the EM algorithm), but otherwise it has no advantages over factor analysis (see section 17.5.1 for more details). We will hence restrict ourselves to a discussion of the more general factor analysis model.

7.6.1 Factor analysis as a marginalization

As for the mixtures of Gaussians and the t-distribution, it is possible to view the factor analysis model as a marginalization of a joint distribution between the observed data $\mathbf{x}$ and a K-dimensional hidden variable $\mathbf{h}$. We define

$$\begin{aligned} Pr(\mathbf{x}|\mathbf{h}) &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}, \boldsymbol{\Sigma}] \\ Pr(\mathbf{h}) &= \text{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}], \end{aligned} \tag{7.32}$$

Problem 7.8

where $\mathbf{I}$ represents the identity matrix. It can be shown (but is not obvious) that

$$\begin{aligned} Pr(\mathbf{x}) = \int Pr(\mathbf{x}, \mathbf{h}) d\mathbf{h} &= \int Pr(\mathbf{x}|\mathbf{h}) Pr(\mathbf{h})\, d\mathbf{h} \\ &= \int \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}, \boldsymbol{\Sigma}] \text{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}]\, d\mathbf{h} \\ &= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}], \end{aligned} \tag{7.33}$$

which was the original definition of the factor analyzer (equation 7.31).

Expressing factor analysis as a marginalization reveals a simple method to draw samples from the distribution. We first draw a hidden variable $\mathbf{h}$ from the normal prior. We then draw the sample $\mathbf{x}$ from a normal distribution with mean $\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}$ and diagonal covariance $\boldsymbol{\Sigma}$ (see equation 7.32).

[1]That is, it is tall and thin, as opposed to landscape which would be short and wide

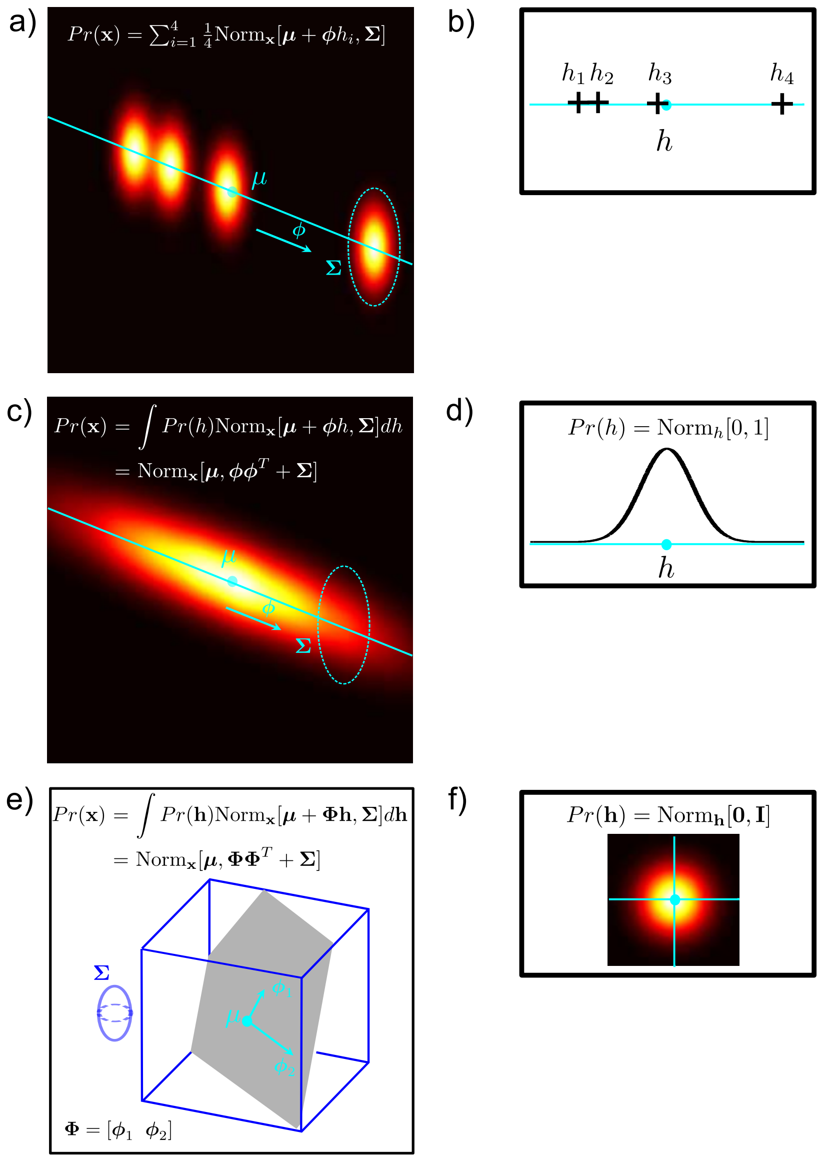

a)
$Pr(\mathbf{x}) = \sum_{i=1}^{4} \frac{1}{4}\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu} + \boldsymbol{\phi} h_i, \boldsymbol{\Sigma}]$
μ
ϕ
Σ
b)
$h_1\, h_2$
h_3
h_4
h
c)
$Pr(\mathbf{x}) = \int Pr(h)\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu} + \boldsymbol{\phi} h, \boldsymbol{\Sigma}]dh$
$= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\phi}\boldsymbol{\phi}^T + \boldsymbol{\Sigma}]$
μ
ϕ
Σ
d)
$Pr(h) = \text{Norm}_h[0, 1]$
h
e)
$Pr(\mathbf{x}) = \int Pr(\mathbf{h})\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}, \boldsymbol{\Sigma}]d\mathbf{h}$
$= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}]$
Σ
ϕ_1
μ
ϕ_2
$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \;\; \boldsymbol{\phi}_2]$
f)
$Pr(\mathbf{h}) = \text{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}]$

Figure 7.20 Relationship between factor analysis and mixtures of Gaussians (MoG). a) Consider a MoG model where each component has identical diagonal covariance $\boldsymbol{\Sigma}$. We could describe variation in a particular direction $\boldsymbol{\phi}$ by parameterizing the mean of each Gaussian as $\boldsymbol{\mu}_i = \boldsymbol{\mu} + \boldsymbol{\phi} h_i$. b) Different values of the scalar hidden variable h_i determine different positions along direction $\boldsymbol{\phi}$. c) Now we replace the MoG with an infinite sum (integral) over a continuous family of Gaussians, each of which is determined by a certain value of h. d) If we choose the prior over the hidden variable to be normal, then this integral has a closed form solution and is a factor analyzer. e) More generally we want to describe variance in a set of directions $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_K]$ in a high dimensional space. f) To this end we use a K-dimensional hidden variable $\mathbf{h}$ and an associated normal prior $Pr(\mathbf{h})$.

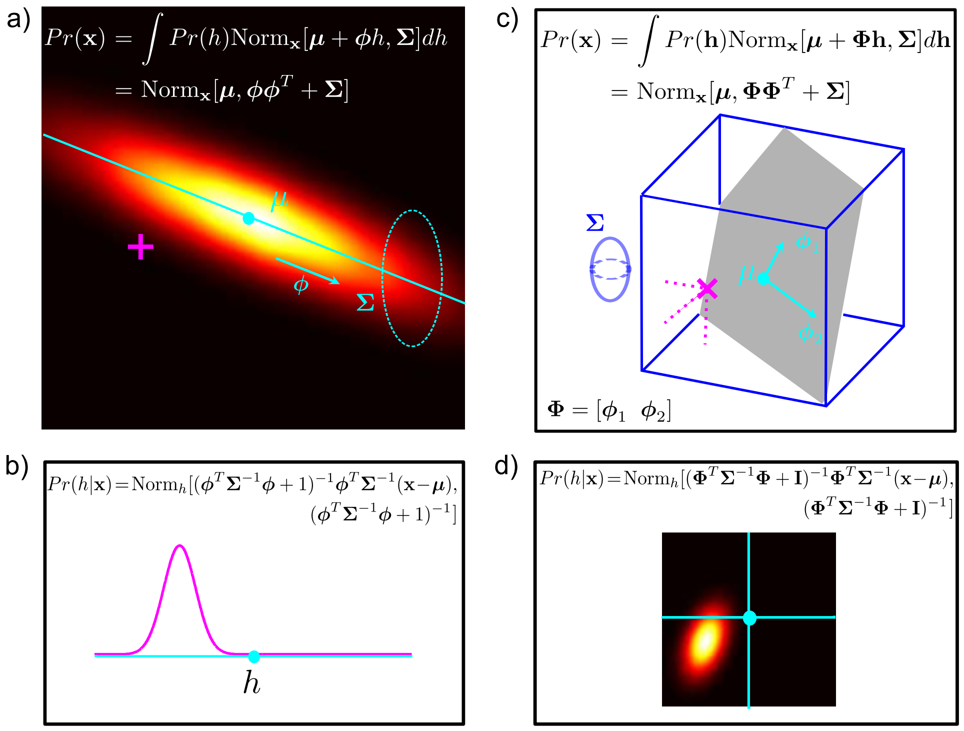

a)
$Pr(\mathbf{x}) = \int Pr(h)\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}+\boldsymbol{\phi}h, \boldsymbol{\Sigma}]dh$
$= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\phi}\boldsymbol{\phi}^T + \boldsymbol{\Sigma}]$
μ
ϕ
Σ
b)
$Pr(h|\mathbf{x}) = \text{Norm}_h[(\boldsymbol{\phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}+1)^{-1}\boldsymbol{\phi}^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}),$
$(\boldsymbol{\phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\phi}+1)^{-1}]$
h
c)
$Pr(\mathbf{x}) = \int Pr(\mathbf{h})\text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}+\boldsymbol{\Phi}\mathbf{h}, \boldsymbol{\Sigma}]d\mathbf{h}$
$= \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}]$
Σ
ϕ_1
μ
ϕ_2
$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2]$
d)
$Pr(h|\mathbf{x}) = \text{Norm}_h[(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}+\mathbf{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}),$
$(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}+\mathbf{I})^{-1}]$

Figure 7.21 E-step for expectation maximization algorithm for factor analysis. (a) Two dimensional case with one factor. We are given a data point $\mathbf{x}$ (purple cross). (b) In the E-step we seek a distribution over possible values of the associated hidden variable h. It can be shown that this posterior distribution over h is itself normally distributed. (c) Three-dimensional case with two factors. Given a data point $\mathbf{x}$ (purple cross), we aim to find a distribution (d) over possible values of the associated hidden variable $\mathbf{h}$. Once more this posterior is normally distributed.

This leads us to a simple interpretation of the hidden variable $\mathbf{h}$: each element h_k weights the associated basis function $\boldsymbol{\phi}_k$ in the matrix $\boldsymbol{\Phi}$ and hence defines a point on the subspace (figure 7.19). The final density (equation 7.31) is hence an infinite weighted sum of normal distributions with the same diagonal covariance $\boldsymbol{\Sigma}$ and means $\boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}$ that are distributed over the subspace. The relationship between mixture models and factor analysis is explored further in figure 7.20.

7.6.2 Expectation maximization for learning factor analyzers

Algorithm 7.3

Since the factor analyzer can be expressed as a marginalization of a joint distribution between the observed data $\mathbf{x}$ and a hidden variable $\mathbf{h}$ (equation 7.33), it is possible to use the EM algorithm to learn the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}\}$. Once more, we follow the recipe described in section 7.3.

In the E-step (figure 7.21) we optimize the bound with respect to the distributions $q_i(\mathbf{h}_i)$. To do this we compute the posterior probability distribution $Pr(\mathbf{h}_i|\mathbf{x}_i)$ over each hidden variable $\mathbf{h}_i$ given the associated observed data $\mathbf{x}_i$ and the current values of the parameters $\boldsymbol{\theta}^{[t]}$. To this end we apply Bayes' rule:

Problem 7.9

$$\begin{aligned}\hat{q}(\mathbf{h}_i) &= Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta}^{[t]}) \qquad (7.34)\\ &= \frac{Pr(\mathbf{x}_i|\mathbf{h}_i,\boldsymbol{\theta}^{[t]})Pr(\mathbf{h}_i)}{Pr(\mathbf{x}_i|\boldsymbol{\theta}^{[t]})}\\ &= \frac{\text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}+\boldsymbol{\Phi}\mathbf{h}_i,\boldsymbol{\Sigma}]\text{Norm}_{\mathbf{h}_i}[\mathbf{0},\mathbf{I}]}{Pr(\mathbf{x}_i|\boldsymbol{\theta}^{[t]})}\\ &= \text{Norm}_{\mathbf{h}_i}[(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}+\mathbf{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu}),(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}+\mathbf{I})^{-1}]\end{aligned}$$

where we have made use of the change of variables relation (section 5.7) and then the fact that the product of two normals is proportional to a third normal (section 5.6). The resulting constant of proportionality exactly cancels out with the term $Pr(\mathbf{x})$ ensuring that the result is a valid probability distribution.

The E-step computes a probability distribution over the possible causes $\mathbf{h}$ for the observed data. This implicitly defines a probability distribution over the positions $\boldsymbol{\Phi}\mathbf{h}$ on the subspace that might have generated this example.

We extract the following expectations from the posterior distribution (equation 7.34) as they will be needed in the M-step:

$$\begin{aligned}\text{E}[\mathbf{h}_i] &= (\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}+\mathbf{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu})\\ \text{E}[\mathbf{h}_i\mathbf{h}_i^T] &= E\left[(\mathbf{h}_i-\text{E}[\mathbf{h}_i])(\mathbf{h}_i-\text{E}[\mathbf{h}_i])^T\right]+\text{E}[\mathbf{h}_i]\text{E}[\mathbf{h}_i]^T\\ &= (\boldsymbol{\Phi}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}+\mathbf{I})^{-1}+\text{E}[\mathbf{h}_i]\text{E}[\mathbf{h}_i]^T. \qquad (7.35)\end{aligned}$$

In the M-step, we optimize the bound with respect to the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu},\boldsymbol{\Phi},\boldsymbol{\Sigma}\}$ so that

$$\begin{aligned}\hat{\boldsymbol{\theta}}^{[t+1]} &= \underset{\boldsymbol{\theta}}{\text{argmax}}\left[\sum_{i=1}^{I}\int\hat{q}_i(\mathbf{h}_i)\log\left[Pr(\mathbf{x},\mathbf{h}_i,\boldsymbol{\theta})\right]\,d\mathbf{h}_i\right]\\ &= \underset{\boldsymbol{\theta}}{\text{argmax}}\left[\sum_{i=1}^{I}\int\hat{q}_i(\mathbf{h}_i)\left[\log\left[Pr(\mathbf{x}|\mathbf{h}_i,\boldsymbol{\theta})\right]+\log\left[Pr(\mathbf{h}_i)\right]\right]\,d\mathbf{h}_i\right]\\ &= \underset{\boldsymbol{\theta}}{\text{argmax}}\left[\sum_{i=1}^{I}\int\hat{q}_i(\mathbf{h}_i)\log\left[Pr(\mathbf{x}|\mathbf{h}_i,\boldsymbol{\theta})\right]\,d\mathbf{h}_i\right]\\ &= \underset{\boldsymbol{\theta}}{\text{argmax}}\left[\sum_{i=1}^{I}E\left[\log Pr(\mathbf{x}|\mathbf{h}_i,\boldsymbol{\theta})\right]\right], \qquad (7.36)\end{aligned}$$

where we have removed the term $\log[Pr(\mathbf{h}_i)]$ as it is not dependent on the variables $\boldsymbol{\theta}$. The expectations $\text{E}[\bullet]$ are taken with respect to the relevant posterior distributions $\hat{q}_i(\mathbf{h}_i) = Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta}^{[t]})$. The expression for $\log[Pr(\mathbf{x}_i|\mathbf{h}_i)]$ is given by

$$\log[Pr(\mathbf{x}_i|\mathbf{h}_i)] = -\frac{D\log(2\pi) + \log|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu} - \boldsymbol{\Phi}\mathbf{h}_i)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu} - \boldsymbol{\Phi}\mathbf{h}_i)}{2}, \quad (7.37)$$

where D is the dimensionality of the data.

Problem 7.10

We optimize equation 7.36 by taking derivatives with respect to the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}\}$, equating the resulting expressions to zero and rearranging to yield

$$\begin{aligned}
\hat{\boldsymbol{\mu}} &= \frac{\sum_{i=1}^{I}\mathbf{x}_i}{I}\\
\hat{\boldsymbol{\Phi}} &= \left(\sum_{i=1}^{I}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})\mathrm{E}[\mathbf{h}_i]^T\right)\left(\sum_{i=1}^{I}\mathrm{E}[\mathbf{h}_i\mathbf{h}_i^T]\right)^{-1}\\
\hat{\boldsymbol{\Sigma}} &= \frac{1}{I}\sum_{i=1}^{I}\mathrm{diag}\left[(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T - \hat{\boldsymbol{\Phi}}\mathrm{E}[\mathbf{h}_i](\mathbf{x}^T - \boldsymbol{\mu})\right],
\end{aligned} \quad (7.38)$$

where the function diag[$\bullet$] is the operation of setting all elements of the matrix argument to zero except those on the diagonal.

Figure 7.22 shows the parameters of a factor analysis model fitted to the face data using ten iterations of the EM algorithm. The different factors encode different *modes of variation* of the data set which often have real-world interpretations such as changes in pose or lighting.

In conclusion, the factor analyzer is an efficient model for capturing the covariance in high-dimensional data. It devotes one set of parameters $\boldsymbol{\Phi}$ to describing the directions in which the data are most correlated and a second set $\boldsymbol{\Sigma}$ describes the remaining variation.

7.7 Combining models

The mixture of Gaussians, t-distribution, and factor analysis models are constructed similarly: each is a weighted sum or integral of a set of constituent normal distributions. Mixture of Gaussian models consist of a weighted sum of K normal distributions with different means and variances. The t-distribution consists of an infinite weighted sum of normal distributions with the same mean, but different covariances. Factor analysis models consist of an infinite weighted sum of normal distributions with different means, but the same diagonal covariance.

In light of these similarities, it is perhaps unsurprising then that the models can be easily combined. If we combine mixture models and factor analyzers, we get a *mixture of factor analyzers (MoFA)* model. This is a weighted sum of factor analyzers, each of which has a different mean and allocates high probability density to a different subspace. Similarly, combining mixture models and t-distributions results in a *mixture of t-distributions* or *robust mixture model.* Combining t-distributions and factor analyzers, we can construct a *robust subspace model*, which models data

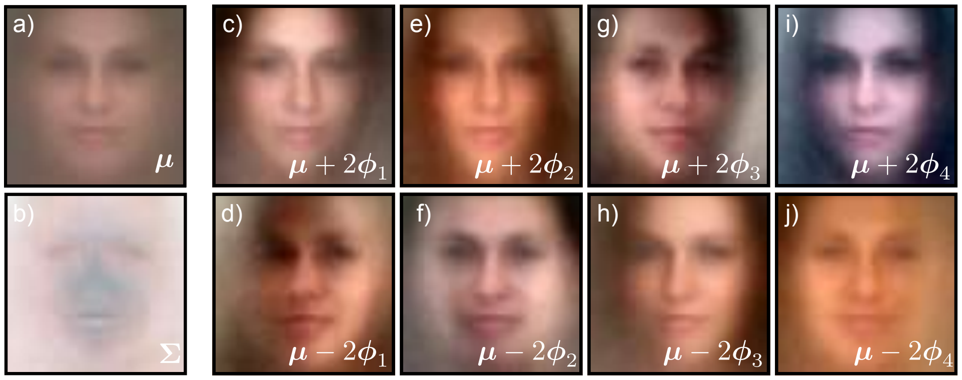


Figure 7.22 Factor analyzer with ten factors (four shown) for face classes. a) Mean $\boldsymbol{\mu}$ for face model. b) Diagonal covariance component $\boldsymbol{\Sigma}$ for face model. To visualize the effect of the first factor $\boldsymbol{\phi}_1$ we add (c) or subtract (d) a multiple of it from the mean: we are moving along one axis of the 10D subspace that seems to encode mainly the mean intensity. Other factors (e-j) encode changes in the hue and the pose of the face.

that lie primarily in a subspace but is tolerant to outliers. Finally, combining all three models, we get a *mixture of robust subspace models*. This has the combined benefits of all three approaches (it is multi-modal and robust and makes efficient use of parameters). The associated density function is:

$$Pr(\mathbf{x}) \quad = \quad \sum_{k=1}^{K} \lambda_k \text{Stud}_{\mathbf{x}} \left[\boldsymbol{\mu}_k, \boldsymbol{\Phi}_k \boldsymbol{\Phi}_k^T + \boldsymbol{\Sigma}_k, \nu_k \right], \tag{7.39}$$

where $\boldsymbol{\mu}_k$,$\boldsymbol{\Phi}_k$ and $\boldsymbol{\Sigma}_k$ represent the mean, factors, and diagonal covariance matrix belonging to the k^{th} component, λ_k represents the weighting of the k^{th} component and ν_k represents the degrees of freedom of the k^{th} component. To learn this model, we would use a series of interleaved expectation maximization algorithms.

7.8 Expectation maximization in detail

Throughout this chapter, we have employed the expectation maximization (EM) algorithm, using the recipe from section 7.3. We now examine the EM algorithm in detail to understand why this recipe works.

The EM algorithm is used to find maximum likelihood or MAP estimates of model parameters $\boldsymbol{\theta}$ where the likelihood $Pr(\mathbf{x}|\boldsymbol{\theta})$ of the data $\mathbf{x}$ can be written as

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_k Pr(\mathbf{x}, h = k|\boldsymbol{\theta}) = \sum_k Pr(\mathbf{x}|h = k, \boldsymbol{\theta}) Pr(h = k) \tag{7.40}$$

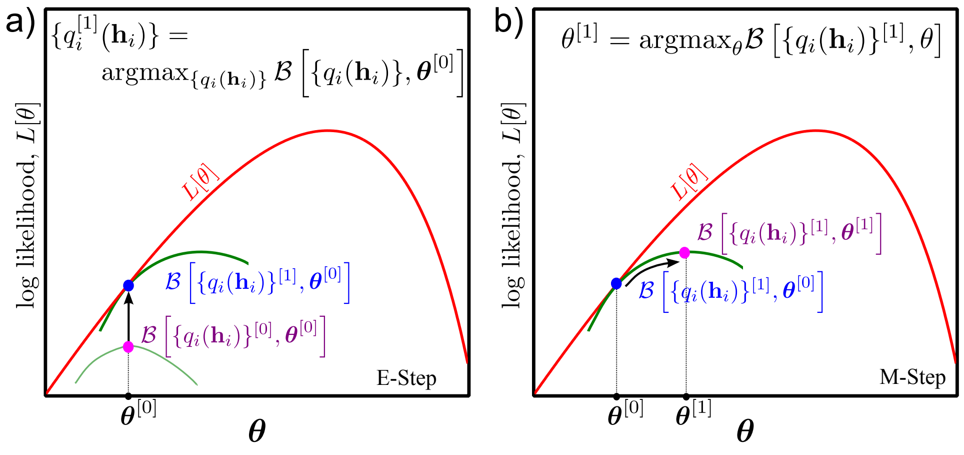


Figure 7.23 E-step and M-step. a) In the E-step, we manipulate the distributions $\{q_i(\mathbf{h}_i)\}$ to find the best new lower bound given parameters $\boldsymbol{\theta}$. This optimal lower bound will touch the log likelihood at the current parameter values $\boldsymbol{\theta}$ (we cannot do better than this!). b) In the M-step, we hold $\{q_i(\mathbf{h}_i)\}$ constant and optimize $\boldsymbol{\theta}$ with respect to the new bound.

and

$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \int Pr(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta})\, d\mathbf{h} = \int Pr(\mathbf{x}|\mathbf{h}, \boldsymbol{\theta}) Pr(\mathbf{h})\, d\mathbf{h} \qquad (7.41)$$

for discrete and continuous hidden variables, respectively. In other words, the likelihood $Pr(\mathbf{x}|\boldsymbol{\theta})$ is a marginalization of a joint distribution over the data and the hidden variables. We will work with the continuous case.

The EM algorithm relies on the idea of a lower bounding function (or *lower bound*), $\mathcal{B}[\boldsymbol{\theta}]$ on the log likelihood. This is a function of the parameters $\boldsymbol{\theta}$ that is always guaranteed to be equal to or lower than the log likelihood. The lower bound is carefully chosen so that it is easy to maximize with respect to the parameters.

This lower bound is also parameterized by a set of probability distributions $\{q_i(\mathbf{h}_i)\}_{i=1}^I$ over the hidden variables, so we write it as $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$. Different probability distributions $q_i(\mathbf{h}_i)$ predict different lower bounds $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$ and hence different functions of $\boldsymbol{\theta}$ that lie everywhere below the true log likelihood (figure 7.5b).

In the EM algorithm we alternate between expectation steps (E-steps) and maximization steps (M-steps) where

- in the E-step (figure 7.23a) we fix $\boldsymbol{\theta}$ and find the best lower bound $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$ with respect to the distributions $q_i(\mathbf{h}_i)$. In other words, at iteration t

$$q_i^{[t]}[\mathbf{h}_i] = \underset{q_i[\mathbf{h}_i]}{\text{argmax}} \left[\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \theta^{[t-1]}]\right]. \qquad (7.42)$$

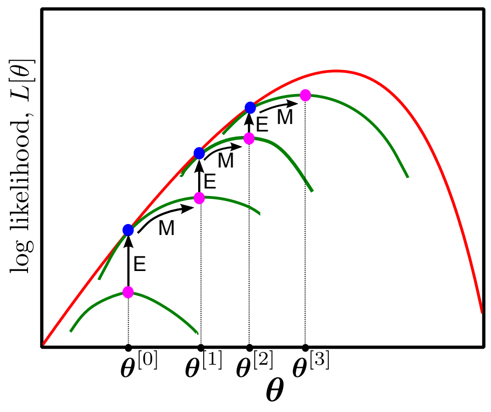


Figure 7.24 Expectation maximization algorithm. We iterate the expectation and maximization steps by alternately changing the distributions $q_i(\mathbf{h}_i)$ and the parameter $\boldsymbol{\theta}$ so that the bound increases. In the E-step, the bound is maximized with respect to $q_i(\mathbf{h}_i)$ for fixed parameters $\boldsymbol{\theta}$: the new function with respect to $\boldsymbol{\theta}$ touches the true log likelihood at $\boldsymbol{\theta}$. In the M-step, we find the maximum of this function. In this way we are guaranteed to reach a local maximum in the likelihood function.

The best lower bound will be a function that is as high as possible at the current parameter estimates $\boldsymbol{\theta}$. Since it must be everywhere equal to or lower than the log likelihood, the highest possible value is the log likelihood itself. So the bound touches the log-likelihood curve for the current parameters $\boldsymbol{\theta}$.

- in the M-step (figure 7.23b), we fix $q_i(\mathbf{h}_i)$ and find the values of $\boldsymbol{\theta}$ that maximize this bounding function $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$. In other words, we compute

$$\boldsymbol{\theta}^{[t]} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\mathcal{B}[\{q_i^{[t]}(\mathbf{h}_i)\}, \boldsymbol{\theta}]\right]. \tag{7.43}$$

By iterating these steps, the (local) maximum of the actual log likelihood is approached (figure 7.24). To complete our picture of the EM algorithm, we must

- define $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}^{[t-1]}]$ and show that it always lies below the log likelihood,
- show which probability distribution $q_i(\mathbf{h}_i)$ optimizes the bound in the E-step,
- show how to optimize the bound with respect to $\boldsymbol{\theta}$ in the M-step.

These three issues are tackled in sections 7.8.1, 7.8.2 and 7.8.3, respectively.

7.8.1 Lower bound for EM algorithm

We define the lower bound $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$ to be

$$\begin{aligned}
\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}] &= \sum_{i=1}^{I} \int q_i(\mathbf{h}_i) \log\left[\frac{Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right] d\mathbf{h}_i \\
&\leq \sum_{i=1}^{I} \log\left[\int q_i(\mathbf{h}_i) \frac{Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\, d\mathbf{h}_i\right] \\
&= \sum_{i=1}^{I} \log\left[\int Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol{\theta})\, d\mathbf{h}_i\right],
\end{aligned} \tag{7.44}$$

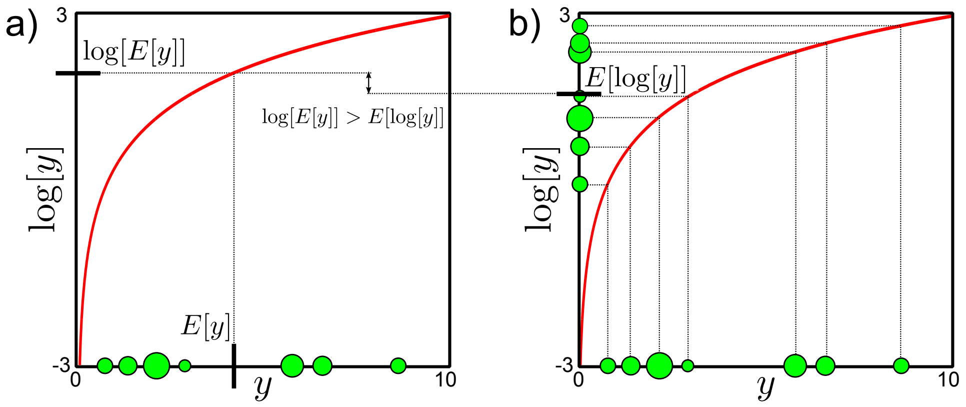


Figure 7.25 Jensen's inequality for the logarithmic function (discrete case). a) Taking a weighted average of examples E[y] and passing them through the log function. b) Passing the samples through the log function and taking a weighted average E[log[y]]. The latter case always produces a smaller value than the former (E[log[y]] $\leq$ log(E[y])): higher valued examples are relatively compressed by the concave log function.

where we have used *Jensen's inequality* between the first and second lines. This states that because the logarithm is a concave function, we can write

$$\int Pr(y)\log[y]dy \leq \log\left[\int yPr(y)dy\right] \tag{7.45}$$

or E[log[y]] $\leq$ log(E[y]). Figure 7.25 illustrates Jensen's inequality for a discrete variable.

7.8.2 The E-step

In the E-step, we update the bound $\mathcal{B}[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}]$ with respect to the distributions $q_i(\mathbf{h}_i)$. To see how to do this, we manipulate the expression for the bound as follows:

$$
\begin{aligned}
\mathcal{B}[\{q_i(\mathbf{h}_i)\},\boldsymbol{\theta}] &= \sum_{i=1}^{I}\int q_i(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{x}_i,\mathbf{h}_i|\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right]\,d\mathbf{h}_i\\
&= \sum_{i=1}^{I}\int q_i(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})Pr(\mathbf{x}_i|\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right]\,d\mathbf{h}_i\\
&= \sum_{i=1}^{I}\int q_i(\mathbf{h}_i)\log\left[Pr(\mathbf{x}_i|\boldsymbol{\theta})\right]\,d\mathbf{h}_i - \sum_{i=1}^{I}\int q_i(\mathbf{h}_i)\log\left[\frac{q_i(\mathbf{h}_i)}{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}\right]\,d\mathbf{h}_i\\
&= \sum_{i=1}^{I}\log\left[Pr(\mathbf{x}_i|\boldsymbol{\theta})\right] - \sum_{i=1}^{I}\int q_i(\mathbf{h}_i)\log\left[\frac{q_i(\mathbf{h}_i)}{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}\right]\,d\mathbf{h}_i, \qquad (7.46)
\end{aligned}
$$

where the hidden variables from the first term are integrated out between the last two lines. The first term in this expression is constant with respect to the distributions $q_i(\mathbf{h}_i)$ and so to optimize the bound we must find the distributions $\hat{q}_i(\mathbf{h}_i)$ that satisfy

$$
\begin{aligned}
\hat{q}_i(\mathbf{h}_i) &= \underset{q_i(\mathbf{h}_i)}{\operatorname{argmax}}\left[-\int q_i(\mathbf{h}_i)\log\left[\frac{q_i(\mathbf{h}_i)}{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}\right]\,d\mathbf{h}_i\right]\\
&= \underset{q_i(\mathbf{h}_i)}{\operatorname{argmax}}\left[\int q_i(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right]\,d\mathbf{h}_i\right]\\
&= \underset{q_i(\mathbf{h}_i)}{\operatorname{argmin}}\left[-\int q_i(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right]\,d\mathbf{h}_i\right]. \qquad (7.47)
\end{aligned}
$$

This expression is known as the *Kullback-Leibler* divergence between $q_i(\mathbf{h}_i)$ and $Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})$. It is a measure of distance between probability distributions. We can use the inequality $\log[y] \leq y-1$ (plot the functions to convince yourself of this!) to show that this cost function (including the minus sign) is always positive,

$$
\begin{aligned}
\int q_i(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right]\,d\mathbf{h}_i &\leq \int q_i(\mathbf{h}_i)\left(\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}-1\right)\,d\mathbf{h}_i\\
&= \int Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta}) - q_i(\mathbf{h}_i)\,d\mathbf{h}_i\\
&= 1-1=0, \qquad (7.48)
\end{aligned}
$$

implying that when we reintroduce the minus sign the cost function *must* be positive. So the criteria in equation 7.46 will be maximized when the Kullback-Leibler divergence is zero. This value is reached when $q_i(\mathbf{h}_i)=Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})$ so that

$$
\begin{aligned}
\int q_i(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}{q_i(\mathbf{h}_i)}\right]\,d\mathbf{h}_i &= \int Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})\log\left[\frac{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}{Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})}\right]\,d\mathbf{h}_i\\
&= \int Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})\log\left[1\right]\,d\mathbf{h}_i = 0. \qquad (7.49)
\end{aligned}
$$

In other words, to maximize the bound with respect to $q_i(\mathbf{h}_i)$ we set this to be the posterior distribution $Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta})$ over the hidden variables $\mathbf{h}_i$ given the current set of parameters. In practice, this is computed using Bayes' rule,

$$Pr(\mathbf{h}_i|\mathbf{x}_i,\boldsymbol{\theta}) = \frac{Pr(\mathbf{x}_i|\mathbf{h}_i,\boldsymbol{\theta})Pr(\mathbf{h}_i)}{Pr(\mathbf{x}_i)}. \tag{7.50}$$

So the E-step consists of computing the posterior distribution for each hidden variable using Bayes' rule.

7.8.3 The M-step

In the M-step we maximize the bound with respect to the parameters $\boldsymbol{\theta}$ so that

$$\begin{aligned}
\boldsymbol{\theta}^{[t]} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\mathcal{B}[\{q_i^{[t]}(\mathbf{h}_i)\},\boldsymbol{\theta}]\right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\int q_i^{[t]}(\mathbf{h}_i)\log\left[\frac{Pr(\mathbf{x}_i,\mathbf{h}_i|\boldsymbol{\theta})}{q_i^{[t]}(\mathbf{h}_i)}\right] d\mathbf{h}_i\right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\int q_i^{[t]}(\mathbf{h}_i)\log\left[Pr(\mathbf{x}_i,\mathbf{h}_i|\boldsymbol{\theta})\right] - q_i^{[t]}(\mathbf{h}_i)\log\left[q_i^{[t]}(\mathbf{h}_i)\right] d\mathbf{h}_i\right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\int q_i^{[t]}(\mathbf{h}_i)\log\left[Pr(\mathbf{x}_i,\mathbf{h}_i|\boldsymbol{\theta})\right] d\mathbf{h}_i\right],
\end{aligned} \tag{7.51}$$

where we have omitted the second term as it does not depend on the parameters. If you look back at the algorithms in this chapter, you will see that we have maximized exactly this criterion.

7.9 Applications

The models in this chapter have many uses in computer vision. We now present a cross-section of applications. As an example of two-class classification using mixtures of Gaussians densities, we reconsider the face detection application that has been a running theme throughout this chapter. To illustrate multi-class classification, we describe an object recognition model based on t-distributions. We also describe a segmentation application which is an example of unsupervised learning: we do not have labeled training data to build our model.

To illustrate the use of the factor analyzer for classification, we present a face recognition example. To illustrate its use for regression, we consider the problem of changing a face image from one pose to another. Finally, we highlight the fact that hidden variables can take on real-world interpretations by considering a model that explains the weak spatial alignment of digits.

7.9.1 Face detection

In face detection we attempt to infer a discrete label $w \in \{0, 1\}$ indicating whether a face is present or not based on observed data $\mathbf{x}$. We will describe the likelihood for each world state with a mixtures of Gaussians model, where the covariances of the Gaussian components are constrained to be diagonal so that

$$Pr(\mathbf{x}|w = m) = \sum_{k=1}^{K} \lambda_{mk} \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_{km}, \boldsymbol{\Sigma}_{km}]. \tag{7.52}$$

where m indexes the world state and k indexes the component of the mixture distribution.

We will assume that we have no prior knowledge about whether the face is present or not so that $Pr(w = 0) = Pr(w = 1) = 0.5$. We fit the two likelihood terms using a set of labeled training pairs $\{\mathbf{x}_i, w_i\}$. In practice this means learning one mixtures of Gaussians model for the non-faces based on the data where $w_i = 0$ and a separate model for the faces based on the data where $w = 1$.

For test data $\mathbf{x}^*$, we compute the posterior probability over w using Bayes' rule

$$Pr(w^* = 1|\mathbf{x}^*) = \frac{Pr(\mathbf{x}^*|w^* = 1)Pr(w^* = 1)}{\sum_{k=0}^{1} Pr(\mathbf{x}^*|w^* = k)Pr(w^* = k)}. \tag{7.53}$$

Table 7.1 shows percent correct classification for 100 test examples The results are based on models learned from 1000 training examples of each class.

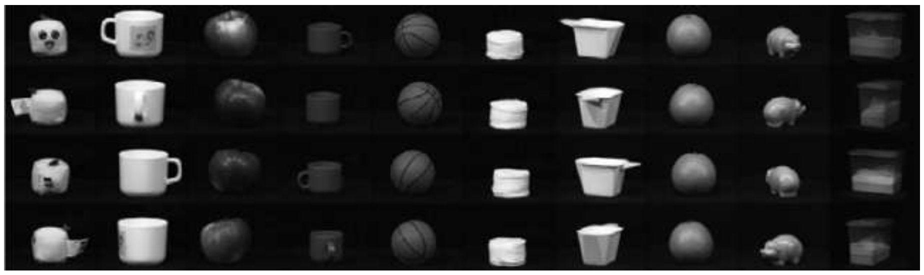
	Color	Grayscale	Equalized
Single Gaussian	76%	79%	80%
Mixture of 10 Gaussians	81%	85%	89%

Table 7.1: Percent correct classification rates for two different models and three different types of preprocessing. In each case, the data $\mathbf{x}^*$ was assumed to be a face if the posterior probability $Pr(w = 1|\mathbf{x}^*)$ was greater than 0.5.

The first column shows results where the data vector consists of the RGB values with a 24×24 region (the running example in this chapter used 60×60 pixel regions, but this is unnecessarily large). The results are compared to classification based on a single normal distribution. The subsequent columns of the table show results for systems trained and tested with grayscale 24×24 pixel regions and grayscale 24×24 regions that have been histogram equalized (section 13.1.2).

There are two insights to be gleaned from these classification results. First, the choice of model does make a difference; the mixtures of Gaussians density always results in better classification performance than the single Gaussian model. Second, the choice of *preprocessing* is also critical to the final performance. This book concerns models for vision, but it is important to understand that this is not the only thing that determines the final performance of real-world systems. A brief summary of preprocessing methods is presented in chapter 13.

a)



b)

74	66	66	90	61	97	47	62	43	71
74	25	44	91	43	80	63	24	08	55

Figure 7.26 Object recognition. a) The training database consists of a series of different views of ten different objects. The goal is to learn a class-conditional density function for each object and classify new examples using Bayes' rule. b) Percent correct results for class conditional densities based on the t-distribution (top row) and the normal distributions (bottom row). The robust model performs better, especially on objects with specularities. Images from Amsterdam library (Guesebroek *et al.* 2005).

The reader should not depart with the impression that this is a sensible approach to face detection. Even the best performance of 89% is far below what would be required in a real face detector: consider that in a single image we might classify patches at 10000 different positions and scales, so an 11% error rate will be unacceptable. Moreover, evaluating each patch under both class conditional density functions is too computationally expensive to be practical. In practice, face detection would normally be achieved using discriminative methods (chapter 9).

7.9.2 Object recognition

In object recognition the goal is to assign a discrete world vector $w_i \in \{1, 2, \ldots, M\}$ indicating which of M categories is present based on observed data $\mathbf{x}$. To this end, Aeschliman *et al.* (2010) split each image into 100 10×10 pixel regions arranged in a regular grid. The grayscale pixel data from the j^{th} region were concatenated to form a 100×1 vector $\mathbf{x}_{ij}$. They treated the regions independently, and described each with a t-distribution so that the class conditional density functions were

$$Pr(\mathbf{x}_i|w=m) = \prod_{j=1}^{J} \text{Stud}_{\mathbf{x}_{ij}}[\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}, \nu_{jm}]. \tag{7.54}$$

Figure 7.26 shows results based on training with 10 classes from the Amsterdam library of images (Guesebroek *et al.* 2005). Each class consists of 72 images taken at 5 degree intervals around the object. The data was divided randomly into 36 test images and 36 training images for each class. The prior probabilities of the classes were set to uniform, and the posterior distribution $Pr(w_i|\mathbf{x}_i)$ was calculated using

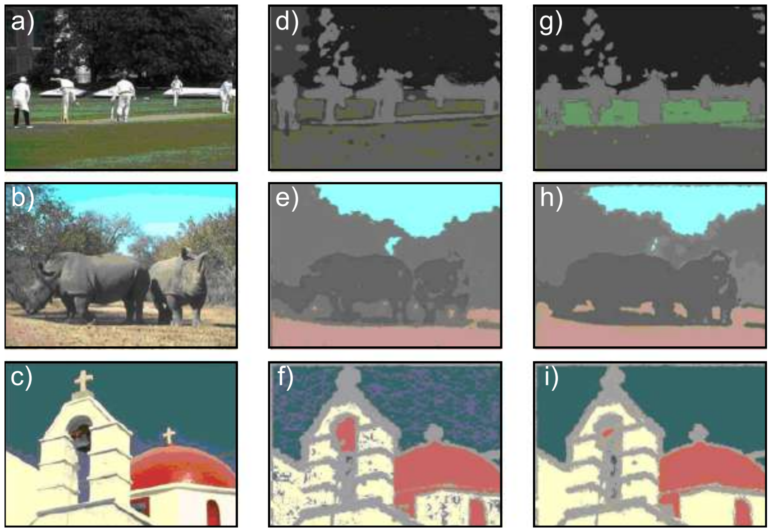


Figure 7.27 Segmentation. a-c) Original images. d-f) Segmentation results based on a mixture of five normal distributions. The pixels associated with the k^{th} component are colored with the mean RGB values of the pixels that are assigned to this value g-i) Segmentation results based on a mixture of K t-distributions. The segmentation here is less noisy than for the MoG model. Results from Sfikas *et al.* (2007) ©IEEE 2007.

Bayes' rule. A test object was classified according to the class with the highest posterior probability.

The results show the superiority of the t-distribution: for almost every class the percent correct performance is better, and this is especially true for objects such as the china pig where the specularities act as outliers. By adding just one more parameter per patch, the performance increases from a mean of 51% to 68%.

7.9.3 Segmentation

The goal of segmentation is to assign a discrete label $\{w_n\}_{n=1}^N$ which takes one of K values $w_n \in \{1, 2, \ldots, K\}$ to each of the N pixels in the image so that regions that belong to the same object are assigned the same label. The segmentation model depends on observed data vectors $\{\mathbf{x}_n\}_{n=1}^N$ at each of the N pixels that would typically include the RGB pixel values, the (x, y) position of the pixel and other information characterizing local texture.

We will frame this problem as *unsupervised learning*. In other words, we do not have the luxury of having training images where the state of the world is known. We must both learn the parameters $\boldsymbol{\theta}$ and estimate the world states $\{w_i\}_{i=1}^I$ from the image data $\{\mathbf{x}_n\}_{n=1}^N$.

We will assume that the k^{th} object is associated with a normal distribution with parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ and prevalence λ_k so that

$$\begin{aligned} Pr(w_n) &= \text{Cat}_{w_n}[\boldsymbol{\lambda}] \\ Pr(\mathbf{x}_i|w_i=k) &= \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]. \end{aligned} \tag{7.55}$$

Marginalizing out the world state w, we have

$$Pr(\mathbf{x}_{1\ldots N}) = \prod_{n=1}^{N}\sum_{k=1}^{K}\lambda_k \text{Norm}_{\mathbf{x}_n}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]. \tag{7.56}$$

To fit this model, we find the parameters $\boldsymbol{\theta} = \{\lambda_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ using the EM algorithm. To assign a class to each pixel, we then find the value of the world state that have the highest posterior probability given the observed data

$$\hat{w}_i = \underset{w_i}{\text{argmax}}\left[Pr(w_i|\mathbf{x}_i)\right], \tag{7.57}$$

where the posterior is computed as in the E-step.

Figure 7.27 shows results from this model and a similar mixture model based on t-distributions from Sfikas *et al.* (2007). The mixture models manage to partition the image quite well into different regions. Unsurprisingly, the t-distribution results are rather less noisy than those based on the normal distribution.

7.9.4 Frontal face recognition

The goal of face identification (figure 7.28) is to assign a label $w \in \{1\ldots M\}$ indicating which of M possible identities the face belongs to based on a data vector $\mathbf{x}$. The model is learned from labeled training data $\{\mathbf{x}_i, w_i\}_{i=1}^{I}$ where the identity is known. In a simple system, the data vector might consist of the concatenated grayscale values from the face image, which should be reasonably large (say 50×50 pixels) to ensure that the identity is well represented.

Since the data are high dimensional, a reasonable approach is to model each class conditional density function with a factor analyzer

$$Pr(\mathbf{x}_i|w_i=k) = \text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Phi}_k\boldsymbol{\Phi}_k^T + \boldsymbol{\Sigma}_k] \tag{7.58}$$

where the parameters for the k^{th} identity $\boldsymbol{\theta}_k = \boldsymbol{\mu}_k, \boldsymbol{\Phi}_k, \boldsymbol{\Sigma}_k$ can be learned from the subset of data that belongs to that identity using the EM algorithm. We also assign priors $P(w=k)$ according to the prevalence of each identity in the database.

To perform recognition, we compute the posterior distribution $Pr(w^*|\mathbf{x}^*)$ for the new data example $\mathbf{x}^*$ using Bayes' rule. We assign the identity that maximizes this posterior distribution.

This approach works well if there are sufficient examples of each gallery individual to learn a factor analyzer, and if the poses of all of the faces are similar. In the next example, we develop a method to change the pose of faces, so that we can cope with the case where the poses differ.

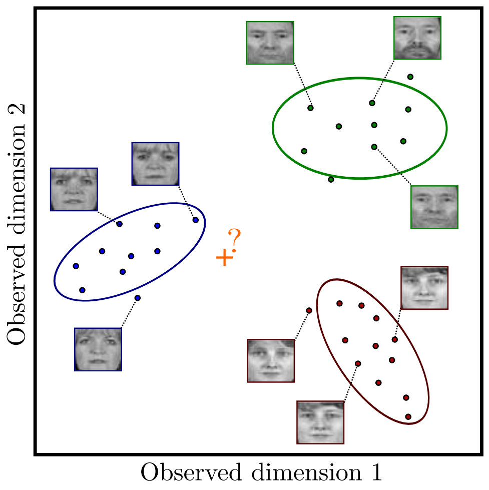


Figure 7.28 Face recognition. Our goal is to take the RGB values of a facial image $\mathbf{x}$ and assign a label $w \in \{1 \ldots K\}$ corresponding to the identity. Since the data are high-dimensional , we model the class conditional density function $Pr(\mathbf{x}|w = k)$ for each individual in the database as a factor analyzer. To classify a new face, we apply Bayes' rule with suitable priors $Pr(w)$ to compute the posterior distribution $Pr(w|\mathbf{x})$. We choose the label $\hat{w} = \text{argmax}_w[Pr(w = k|\mathbf{x})]$ that maximizes the posterior. This approach assumes that there are sufficient training examples to learn a factor analyzer for each class.

7.9.5 Changing face pose (regression)

To change the pose of the face, we predict the RGB values $\mathbf{w}$ of the face in the new pose given a face $\mathbf{x}$ in the old pose. This is different from the previous examples in that it is a regression problem: the output $\mathbf{w}$ is a continuous multidimensional variable rather than a class label.

We form a compound variable $\mathbf{z} = [\mathbf{x}^T \; \mathbf{w}^T]^T$ by concatenating the RGB values from the two poses together. We now model the joint density as $Pr(\mathbf{z}) = Pr(\mathbf{x}, \mathbf{w})$ with a factor analyzer

$$Pr(\mathbf{x}, \mathbf{w}) = Pr(\mathbf{z}) = \text{Norm}_{\mathbf{z}}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}], \tag{7.59}$$

which we learn for paired training examples $\{\mathbf{x}_i, \mathbf{w}_i\}_{i=1}^I$ where the identity is known to be the same for each pair.

To find the non-frontal face $\mathbf{w}^*$ corresponding to a new frontal face $\mathbf{x}^*$ we use the approach of section **??**: the posterior over $\mathbf{w}^*$ is just the conditional distribution $Pr(\mathbf{w}|\mathbf{x})$. Since the joint distribution is normal, we can compute the posterior distribution in closed form using equation 5.5. Using the notation

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{x}} \\ \boldsymbol{\mu}_{\mathbf{w}} \end{bmatrix} \text{ and } \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Phi}_{\mathbf{x}}^T + \boldsymbol{\Sigma}_{\mathbf{x}} & \boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Phi}_{\mathbf{w}}^T \\ \boldsymbol{\Phi}_{\mathbf{w}}\boldsymbol{\Phi}_{\mathbf{x}}^T & \boldsymbol{\Phi}_{\mathbf{w}}\boldsymbol{\Phi}_{\mathbf{w}}^T + \boldsymbol{\Sigma}_{\mathbf{w}} \end{bmatrix}, \tag{7.60}$$

the posterior is given by

$$\begin{aligned} Pr(\mathbf{w}^*|\mathbf{x}^*) \;=\; & \text{Norm}_{\mathbf{w}^*}[\boldsymbol{\mu}_{\mathbf{w}} + \boldsymbol{\Phi}_{\mathbf{w}}\boldsymbol{\Phi}_{\mathbf{x}}^T(\boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Phi}_{\mathbf{x}}^T + \boldsymbol{\Sigma}_{\mathbf{x}})^{-1}(\mathbf{x}^* - \boldsymbol{\mu}_{\mathbf{x}}), \\ & \boldsymbol{\Phi}_{\mathbf{w}}\boldsymbol{\Phi}_{\mathbf{w}}^T + \boldsymbol{\Sigma}_{\mathbf{w}} - \boldsymbol{\Phi}_{\mathbf{w}}\boldsymbol{\Phi}_{\mathbf{x}}^T(\boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Phi}_{\mathbf{x}}^T + \boldsymbol{\Sigma}_{\mathbf{x}})^{-1}\boldsymbol{\Phi}_{\mathbf{x}}\boldsymbol{\Phi}_{\mathbf{w}}^T], \end{aligned} \tag{7.61}$$

and the most probable $\mathbf{w}^*$ is at the mean of this distribution.

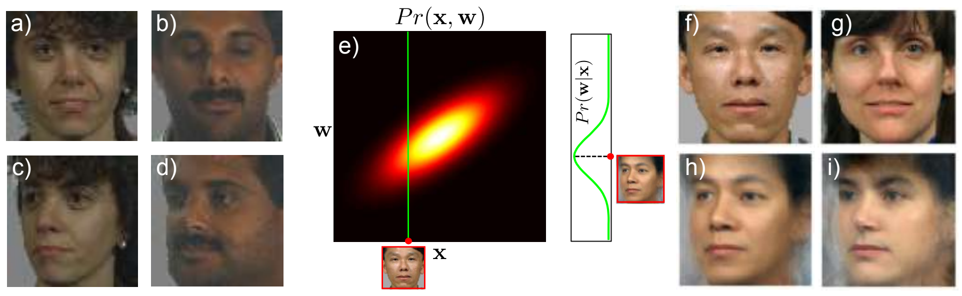


Figure 7.29 Regression example: we aim to predict quarter left face image $\mathbf{w}$ from the frontal image $\mathbf{x}$. To this end, we take paired examples of frontal faces (a-b) and quarter left faces (c-d) and learn a joint probability model $Pr(\mathbf{x}, \mathbf{w})$ by concatenating the variables to form $\mathbf{z} = [\mathbf{x}^T, \mathbf{w}^T]^T$ and fitting a factor analyzer to $\mathbf{z}$. e) Since the factor analyzer has a normal density (equation 7.31), we can predict the conditional distribution $Pr(\mathbf{w}|\mathbf{x})$ of a quarter-left face given a frontal face which will also be normal (see section 5.5). f-g) Two frontal faces. h-i) MAP predictions for non-frontal faces (the mean of the normal distribution $Pr(\mathbf{w}|\mathbf{x})$).

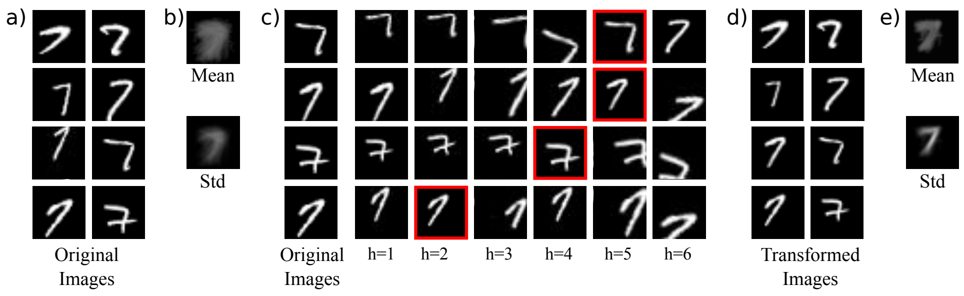


Figure 7.30 Modeling transformations with hidden variables. a) The original set of digit images are only weakly aligned. b) The mean and standard deviation images are consequently blurred out. The probability density model does not fit well. c) Each possible value of a discrete hidden variable represents a different transformation (here inverse transformations are shown). The red square highlights the most likely choice of hidden variable after ten iterations. d) The inversely transformed digits (based on most likely hidden variable). d) The new mean and standard deviation images are more focused: the probability density function fits better.

7.9.6 Transformations as hidden variables

Finally, we consider a model that is closely related to the mixture of Gaussians, where the hidden variables have a clear real-world interpretation. Consider the problem of building a density function for a set of poorly aligned images of digits (figure 7.30). A simple normal distribution with diagonal covariance produces only a very poor representation of the data because most of the variation is due to the poor alignment.

We construct a generative model that first draws an *aligned* image $\mathbf{x}'$ from a normal distribution, and then translates it using one of a discrete set $\{\text{trans}_k[\bullet]\}_{k=1}^{K}$ of K possible transformations to explain the poorly aligned image $\mathbf{x}$. In mathematical terms, we have

$$\begin{aligned} Pr(\mathbf{x}') &= \text{Norm}_{\mathbf{x}'}[\boldsymbol{\mu}, \boldsymbol{\Sigma}] \\ Pr(h) &= \text{Cat}_h[\lambda] \\ \mathbf{x} &= \text{trans}_h[\mathbf{x}'] \end{aligned} \tag{7.62}$$

where $h \in \{1, \ldots, K\}$ is a hidden variable that denotes which of the possible transformations warped this example.

This model can be learned using an EM-like procedure. In the E-step, we compute a probability distribution $Pr(h_i|\mathbf{x}_i)$ over the hidden variables by applying all of the inverse transformations to each example and evaluating how likely the result is under the current parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. In the M-step, we update these parameters by taking weighted sums of the inverse transformed images.

Summary

In this chapter we have introduced the idea of the *hidden variable* to induce structure in density models. The main approach to learning models of this kind is the expectation maximization algorithm. This is an iterative approach that is only guaranteed to find a local maximum. We have seen that although these models are more sophisticated than the normal distribution, they are still not really good representations of the density of high-dimensional visual data.

Notes

Expectation maximization: The EM algorithm was originally described by Dempster *et al.* (1977) although the presentation in this chapter owes more to the perspective espoused in Neal & Hinton (1999). A comprehensive summary of the EM algorithm and its extensions can be found in McLachlan & Krishnan (2008).

Mixtures of Gaussians: The mixtures of Gaussians model is closely related to the *K-means* algorithm (discussed in section 13.4.4), which is a pure clustering algorithm but does not have a probabilistic interpretation. Mixture models are used extensively in computer vision. Common applications include skin detection (e.g., Jones & Rehg 2002) and background subtraction (e.g., Stauffer & Grimson 1999).

t-distributions: General information about the t distribution can be found in Kotz & Nadarajah (2004). The EM algorithm for fitting the t-distribution is given in Liu & Rubin (1995), and other fitting methods are discussed in Nadarajah & Kotz (2008) and Aeschliman *et al.* (2010). Applications of t-distributions in vision include object recognition (Aeschliman *et al.* 2010) and tracking (Loxam & Drummond 2008; Aeschliman *et al.* 2010), and they are also used as the building blocks of sparse image priors (Roth & Black 2009).

Subspace models: The EM algorithm to learn the factor analyzer is due to Rubin & Thayer (1982). Factor analysis is closely related to several other models including probabilistic principal component analysis (Tipping & Bishop 1999), which is discussed in section 17.5.1 and principal component analysis which is discussed in section 13.4.2. Subspace models have been extended to the nonlinear case by Lawrence (2004). This is discussed in detail in section 17.8. In chapter 18 we present a series of models based on factor analysis, which explicitly encode the identity and style of the object.

Combining models: Ghahramani & Hinton (1996c) introduced the mixtures of factor analyzers model. Peel & McLachlan (2000) present an algorithm for learning robust mixture models (mixtures of t-distributions). Khan & Dellaert (2004) and Zhao & Jiang (2006) both present subspace models based on the t-distribution. De Ridder & Franc (2003) combined mixture models, subspace models and t-distributions to create a distribution that is mutli-modal, robust and oriented along a subspace.

Face detection, face recognition, object recognition: Models based on subspace distributions were used in early methods for face and object recognition (e.g., Moghaddam & Pentland 1997; Murase & Nayar 1995). Modern face detection methods mainly rely on discriminative methods (chapter 9), and the current state of the art in object recognition relies on bag of words approaches (chapter 20). Face recognition applications do not usually have the luxury of having many examples of each individual and so cannot build a separate density for each. However, modern approaches are still largely based on subspace methods (see chapter 18).

Segmentation: Belongie *et al.* (1998) use a mixtures of Gaussians scheme similar to that described to segment the image as part of a content-based image retrieval system. A modern approach to segmentation based on the mixture of Gaussians model can be found in Ma *et al.* (2007). Sfikas *et al.* (2007) compared the segmentation performance of mixtures of Gaussians and mixtures of t-distributions.

Other uses for hidden variables: Frey & Jojic (1999a) and Frey & Jojic (1999b) used hidden variables to model unseen transformations applied to data from mixture models and subspace models, respectively. Jojic & Frey (2001) used discrete hidden variables to represent the index of the layer in a multi-layered model of a video. Jojic *et al.*

(2003) presented a structured mixture model, where the mean and variance parameters are represented as images and the hidden variable indexes the starting position of subpatch from this image.

Problems

Problem 7.1 Consider a computer vision system for machine inspection of oranges in which the goal is to tell if the orange is ripe. For each image we separate the orange from the background and calculate the average color of the pixels, which we describe as a 3×1 vector $\mathbf{x}$. We are given training pairs $\{\mathbf{x}_i, w_i\}$ of these vectors, each with an associated binary variable $w \in \{0, 1\}$ that indicates that this training example was unripe ($w = 0$) or ripe ($w = 1$). Describe how to build a generative classifier that could classify new examples $\mathbf{x}^*$ as being ripe or unripe.

Problem 7.2 It turns out that a small subset of the training labels w_i in the previous example were wrong. How could you modify your classifier to cope with this situation?

Problem 7.3 Derive the M-step equations for the mixtures of Gaussians model (equation 7.19).

Problem 7.4 Consider modeling some univariate continuous visual data $x \in [0, 1]$ using a *mixture of beta distributions*. Write down an equation for this model. Describe in words what will occur in (i) the E-step and (ii) the M-step.

Problem 7.5 Prove that the student t-distribution over x is the marginalization with respect to h of the joint distribution $Pr(x, h)$ between x and a hidden variable h where

$$\text{Stud}_x[\mu, \sigma^2, \nu] = \int \text{Norm}_x[\mu, \sigma^2/h]\text{Gam}_h[\nu/2, \nu/2]dh.$$

Problem 7.6 Show that the peak of the Gamma distribution $\text{Gam}_z[\alpha, \beta]$ is at

$$\hat{z} = \frac{\alpha - 1}{\beta}.$$

Problem 7.7 Show that the Gamma distribution is conjugate to the inverse scaling factor of the variance in a normal distribution so that

$$\text{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}, \boldsymbol{\Sigma}/h_i]\text{Gam}_{h_i}[\nu/2, \nu/2] = \kappa\text{Gam}_{h_i}[\alpha, \beta],$$

and find the constant of proportionality κ and the new parameters α and β.

Problem 7.8 The model for factor analysis can be written as

$$\mathbf{x}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \boldsymbol{\epsilon}_i,$$

where $\mathbf{h}_i$ is distributed normally with mean zero and identity covariance and $\boldsymbol{\epsilon}_i$ is distributed normally with mean zero and covariance $\boldsymbol{\Sigma}$. Determine expressions for

1. $E[\mathbf{x}_i]$,

2. $E[(\mathbf{x}_i - E[\mathbf{x}_i])(\mathbf{x}_i - E[\mathbf{x}_i])^T]$.

Problem 7.9 Derive the E-step for factor analysis (equation 7.34).

Problem 7.10 Derive the M-step for factor analysis (equation 7.38).