12 *Latent linear models*

12.1 Factor analysis

One problem with mixture models is that they only use a single latent variable to generate the observations. In particular, each observation can only come from one of K prototypes. One can think of a mixture model as using K hidden binary variables, representing a one-hot encoding of the cluster identity. But because these variables are mutually exclusive, the model is still limited in its representational power.

An alternative is to use a vector of real-valued latent variables, $\mathbf{z}_i \in \mathbb{R}^L$. The simplest prior to use is a Gaussian (we will consider other choices later):

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{12.1}$$

If the observations are also continuous, so $\mathbf{x}_i \in \mathbb{R}^D$, we may use a Gaussian for the likelihood. Just as in linear regression, we will assume the mean is a linear function of the (hidden) inputs, thus yielding

$$p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{12.2}$$

where $\mathbf{W}$ is a $D \times L$ matrix, known as the **factor loading matrix**, and $\boldsymbol{\Psi}$ is a $D \times D$ covariance matrix. We take $\boldsymbol{\Psi}$ to be diagonal, since the whole point of the model is to "force" $\mathbf{z}_i$ to explain the correlation, rather than "baking it in" to the observation's covariance. This overall model is called **factor analysis** or **FA**. The special case in which $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$ is called **probabilistic principal components analysis** or **PPCA**. The reason for this name will become apparent later.

The generative process, where $L = 1$, $D = 2$ and $\boldsymbol{\Psi}$ is diagonal, is illustrated in Figure 12.1. We take an isotropic Gaussian "spray can" and slide it along the 1d line defined by $\mathbf{w}z_i + \mu$. This induces an ellongated (and hence correlated) Gaussian in 2d.

12.1.1 FA is a low rank parameterization of an MVN

FA can be thought of as a way of specifying a joint density model on $\mathbf{x}$ using a small number of parameters. To see this, note that from Equation 4.126, the induced marginal distribution $p(\mathbf{x}_i|\boldsymbol{\theta})$ is a Gaussian:

$$p(\mathbf{x}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{x}_i|\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi})\mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)d\mathbf{z}_i \tag{12.3}$$

$$= \mathcal{N}(\mathbf{x}_i|\mathbf{W}\boldsymbol{\mu}_0 + \boldsymbol{\mu}, \boldsymbol{\Psi} + \mathbf{W}\boldsymbol{\Sigma}_0\mathbf{W}^T) \tag{12.4}$$

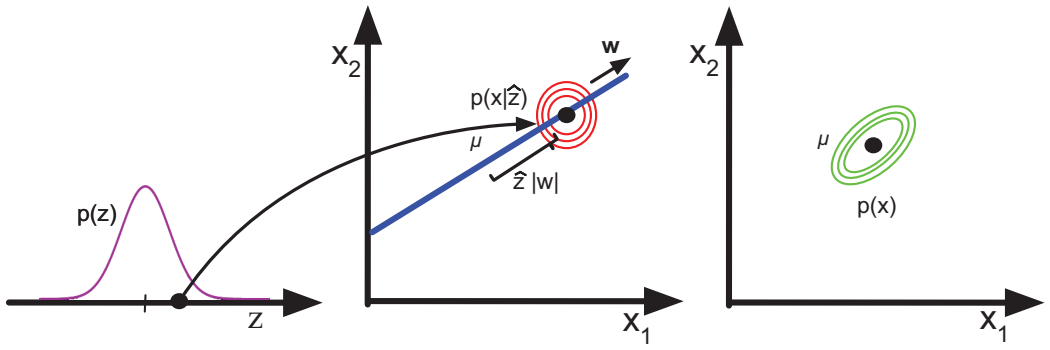


Figure 12.1 Illustration of the PPCA generative process, where we have $L = 1$ latent dimension generating $D = 2$ observed dimensions. Based on Figure 12.9 of (Bishop 2006b).

From this, we see that we can set $\boldsymbol{\mu}_0 = \mathbf{0}$ without loss of generality, since we can always absorb $\mathbf{W}\boldsymbol{\mu}_0$ into $\boldsymbol{\mu}$. Similarly, we can set $\boldsymbol{\Sigma}_0 = \mathbf{I}$ without loss of generality, because we can always "emulate" a correlated prior by using defining a new weight matrix, $\tilde{\mathbf{W}} = \mathbf{W}\boldsymbol{\Sigma}_0^{-\frac{1}{2}}$. Then we find

$$\text{cov}\left[\mathbf{x}|\boldsymbol{\theta}\right] = \tilde{\mathbf{W}}^T + \mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right] = (\mathbf{W}\boldsymbol{\Sigma}_0^{-\frac{1}{2}})\boldsymbol{\Sigma}_0(\mathbf{W}\boldsymbol{\Sigma}_0^{-\frac{1}{2}})^T + \boldsymbol{\Psi} = \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi} \tag{12.5}$$

We thus see that FA approximates the covariance matrix of the visible vector using a low-rank decomposition:

$$\mathbf{C} \triangleq \text{cov}\left[\mathbf{x}\right] = \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi} \tag{12.6}$$

This only uses $O(LD)$ parameters, which allows a flexible compromise between a full covariance Gaussian, with $O(D^2)$ parameters, and a diagonal covariance, with $O(D)$ parameters. Note that if we did not restrict $\boldsymbol{\Psi}$ to be diagonal, we could trivially set $\boldsymbol{\Psi}$ to a full covariance matrix; then we could set $\mathbf{W} = \mathbf{0}$, in which case the latent factors would not be required.

12.1.2 Inference of the latent factors

Although FA can be thought of as just a way to define a density on $\mathbf{x}$, it is often used because we hope that the latent factors $\mathbf{z}$ will reveal something interesting about the data. To do this, we need to compute the posterior over the latent factors. We can use Bayes rule for Gaussians to give

$$p(\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_i|\mathbf{m}_i, \boldsymbol{\Sigma}_i) \tag{12.7}$$

$$\boldsymbol{\Sigma}_i \triangleq (\boldsymbol{\Sigma}_0^{-1} + \mathbf{W}^T\boldsymbol{\Psi}^{-1}\mathbf{W})^{-1} \tag{12.8}$$

$$\mathbf{m}_i \triangleq \boldsymbol{\Sigma}_i(\mathbf{W}^T\boldsymbol{\Psi}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}) + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0) \tag{12.9}$$

Note that in the FA model, $\boldsymbol{\Sigma}_i$ is actually independent of i, so we can denote it by $\boldsymbol{\Sigma}$. Computing this matrix takes $O(L^3 + L^2D)$ time, and computing each $\mathbf{m}_i = \mathbb{E}\left[\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta}\right]$ takes $O(L^2 + LD)$ time. The $\mathbf{m}_i$ are sometimes called the latent **scores**, or latent **factors**.

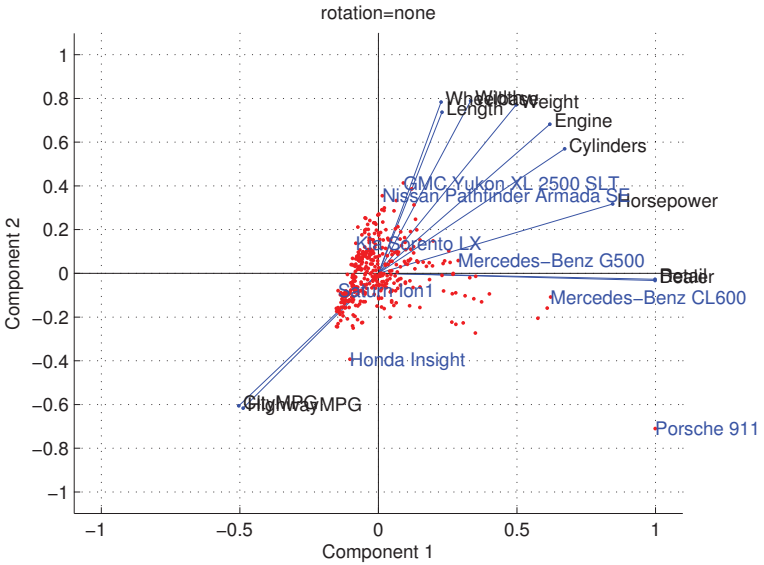


Figure 12.2 2D projection of 2004 cars data based on factor analysis. The blue text are the names of cars corresponding to certain chosen points. Figure generated by `faBiplotDemo`.

Let us give a simple example, based (Shalizi 2009). We consider a dataset of $D = 11$ variables and $N = 387$ cases describing various aspects of cars, such as the engine size, the number of cylinders, the miles per gallon (MPG), the price, etc. We first fit a $L = 2$ dimensional model. We can plot the $\mathbf{m}_i$ scores as points in $\mathbb{R}^2$, to visualize the data, as shown in Figure 12.2.

To get a better understanding of the "meaning" of the latent factors, we can project unit vectors corresponding to each of the feature dimensions, $\mathbf{e}_1 = (1, 0, \ldots, 0)$, $\mathbf{e}_2 = (0, 1, 0, \ldots, 0)$, etc. into the low dimensional space. These are shown as blue lines in Figure 12.2; this is known as a **biplot**. We see that the horizontal axis represents price, corresponding to the features labeled "dealer" and "retail", with expensive cars on the right. The vertical axis represents fuel efficiency (measured in terms of MPG) versus size: heavy vehicles are less efficient and are higher up, whereas light vehicles are more efficient and are lower down. We can "verify" this interpretation by clicking on some points, and finding the closest exemplars in the training set, and printing their names, as in Figure 12.2. However, in general, interpreting latent variable models is fraught with difficulties, as we discuss in Section 12.1.3.

12.1.3 Unidentifiability

Just like with mixture models, FA is also unidentifiable. To see this, suppose $\mathbf{R}$ is an arbitrary orthogonal rotation matrix, satisfying $\mathbf{R}\mathbf{R}^T = \mathbf{I}$. Let us define $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$; then the likelihood

function of this modified matrix is the same as for the unmodified matrix, since

$$\begin{aligned} \text{cov}\left[\mathbf{x}\right] &= \check{\mathbf{W}}\mathbb{E}\left[\mathbf{z}\mathbf{z}^T\right]\check{\mathbf{W}}^T + \mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right] &(12.10)\\ &= \mathbf{W}\mathbf{R}\mathbf{R}^T\mathbf{W}^T + \boldsymbol{\Psi} = \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi} &(12.11)\end{aligned}$$

Geometrically, multiplying $\mathbf{W}$ by an orthogonal matrix is like rotating $\mathbf{z}$ before generating $\mathbf{x}$; but since $\mathbf{z}$ is drawn from an isotropic Gaussian, this makes no difference to the likelihood. Consequently, we cannot unique identify $\mathbf{W}$, and therefore cannot uniquely identify the latent factors, either.

To ensure a unique solution, we need to remove $L(L-1)/2$ degrees of freedom, since that is the number of orthonormal matrices of size $L \times L$.[1] In total, the FA model has $D + LD - L(L-1)/2$ free parameters (excluding the mean), where the first term arises from $\boldsymbol{\Psi}$. Obviously we require this to be less than or equal to $D(D+1)/2$, which is the number of parameters in an unconstrained (but symmetric) covariance matrix. This gives us an upper bound on L, as follows:

$$L_{max} = \lfloor D + 0.5(1 - \sqrt{1+8D}) \rfloor \tag{12.12}$$

For example, $D = 6$ implies $L \leq 3$. But we usually never choose this upper bound, since it would result in overfitting (see discussion in Section 12.3 on how to choose L).

Unfortunately, even if we set $L < L_{max}$, we still cannot uniquely identify the parameters, since the rotational ambiguity still exists. Non-identifiability does not affect the predictive performance of the model. However, it does affect the loading matrix, and hence the interpretation of the latent factors. Since factor analysis is often used to uncover structure in the data, this problem needs to be addressed. Here are some commonly used solutions:

- **Forcing $\mathbf{W}$ to be orthonormal** Perhaps the cleanest solution to the identifiability problem is to force $\mathbf{W}$ to be orthonormal, and to order the columns by decreasing variance of the corresponding latent factors. This is the approach adopted by PCA, which we will discuss in Section 12.2. The result is not necessarily more interpretable, but at least it is unique.
- **Forcing $\mathbf{W}$ to be lower triangular** One way to achieve identifiability, which is popular in the Bayesian community (e.g., (Lopes and West 2004)), is to ensure that the first visible feature is only generated by the first latent factor, the second visible feature is only generated by the first two latent factors, and so on. For example, if $L = 3$ and $D = 4$, the correspond factor loading matrix is given by

$$\mathbf{W} = \begin{pmatrix} w_{11} & 0 & 0 \\ w_{21} & w_{22} & 0 \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix} \tag{12.13}$$

  We also require that $w_{jj} > 0$ for $j = 1 : L$. The total number of parameters in this constrained matrix is $D + DL - L(L-1)/2$, which is equal to the number of uniquely identifiable parameters. The disadvantage of this method is that the first L visible variables,

1. To see this, note that there are $L-1$ free parameters in $\mathbf{R}$ in the first column (since the column vector must be normalized to unit length), there are $L-2$ free parameters in the second column (which must be orthogonal to the first), and so on.

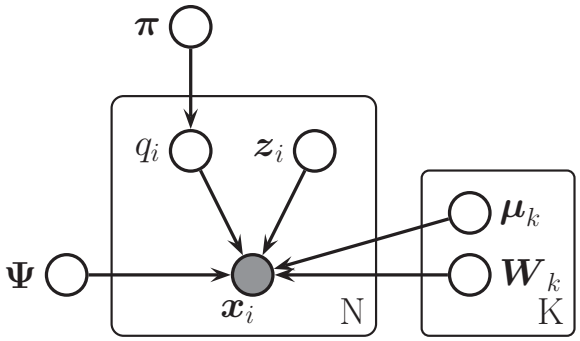


Figure 12.3 Mixture of factor analysers as a DGM.

known as the **founder variables**, affect the interpretation of the latent factors, and so must be chosen carefully.

- **Sparsity promoting priors on the weights** Instead of pre-specifying which entries in $\mathbf{W}$ are zero, we can encourage the entries to be zero, using ℓ_1 regularization (Zou et al. 2006), ARD (Bishop 1999; Archambeau and Bach 2008), or spike-and-slab priors (Rattray et al. 2009). This is called sparse factor analysis. This does not necessarily ensure a unique MAP estimate, but it does encourage interpretable solutions. See Section 13.8.
- **Choosing an informative rotation matrix** There are a variety of heuristic methods that try to find rotation matrices $\mathbf{R}$ which can be used to modify $\mathbf{W}$ (and hence the latent factors) so as to try to increase the interpretability, typically by encouraging them to be (approximately) sparse. One popular method is known as **varimax** (Kaiser 1958).
- **Use of non-Gaussian priors for the latent factors** In Section 12.6, we will dicuss how replacing $p(\mathbf{z}_i)$ with a non-Gaussian distribution can enable us to sometimes uniquely identify $\mathbf{W}$ as well as the latent factors. This technique is known as ICA.

12.1.4 Mixtures of factor analysers

The FA model assumes that the data lives on a low dimensional linear manifold. In reality, most data is better modeled by some form of low dimensional *curved* manifold. We can approximate a curved manifold by a piecewise linear manifold. This suggests the following model: let the k'th linear subspace of dimensionality L_k be represented by $\mathbf{W}_k$, for $k = 1 : K$. Suppose we have a latent indicator $q_i \in \{1, \ldots, K\}$ specifying which subspace we should use to generate the data. We then sample $\mathbf{z}_i$ from a Gaussian prior and pass it through the $\mathbf{W}_k$ matrix (where $k = q_i$), and add noise. More precisely, the model is as follows:

$$p(\mathbf{x}_i|\mathbf{z}_i, q_i = k, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k + \mathbf{W}_k\mathbf{z}_i, \boldsymbol{\Psi}) \quad (12.14)$$

$$p(\mathbf{z}_i|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I}) \quad (12.15)$$

$$p(q_i|\boldsymbol{\theta}) = \mathrm{Cat}(q_i|\boldsymbol{\pi}) \quad (12.16)$$

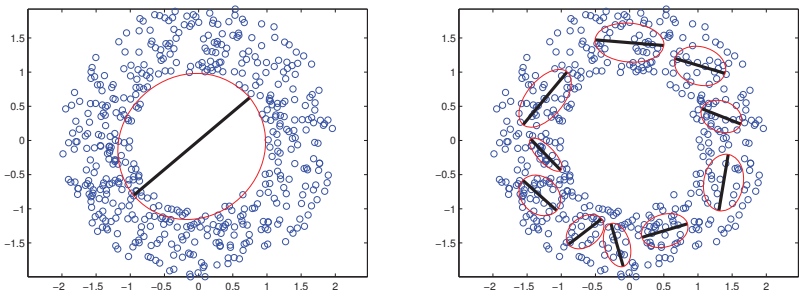


Figure 12.4 Mixture of 1d PPCAs fit to a dataset, for $K = 1, 10$. Figure generated by `mixPpcaDemoNetlab`.

This is called a **mixture of factor analysers** (MFA) (Hinton et al. 1997). The CI assumptions are represented in Figure 12.3.

Another way to think about this model is as a low-rank version of a mixture of Gaussians. In particular, this model needs $O(KLD)$ parameters instead of the $O(KD^2)$ parameters needed for a mixture of full covariance Gaussians. This can reduce overfitting. In fact, MFA is a good generic density model for high-dimensional real-valued data.

12.1.5 EM for factor analysis models

Using the results from Chapter 4, it is straightforward to derive an EM algorithm to fit an FA model. With just a little more work, we can fit a mixture of FAs. Below we state the results without proof. The derivation can be found in (Ghahramani and Hinton 1996a); however, deriving these equations yourself is a useful exercise if you want to become proficient at the math.

To obtain the results for a single factor analyser, just set $r_{ic} = 1$ and $c = 1$ in the equations below. In Section 12.2.5 we will see a further simplification of these equations that arises when fitting a PPCA model, where the results will turn out to have a particularly simple and elegant intepretation.

In the E step, we compute the posterior responsibility of cluster c for data point i using

$$r_{ic} \triangleq p(q_i = c|\mathbf{x}_i, \boldsymbol{\theta}) \propto \pi_c \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_c, \mathbf{W}_c\mathbf{W}_c^T + \boldsymbol{\Psi}) \tag{12.17}$$

The conditional posterior for $\mathbf{z}_i$ is given by

$$p(\mathbf{z}_i|\mathbf{x}_i, q_i = c, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_i|\mathbf{m}_{ic}, \boldsymbol{\Sigma}_{ic}) \tag{12.18}$$

$$\boldsymbol{\Sigma}_{ic} \triangleq (\mathbf{I}_L + \mathbf{W}_c^T\boldsymbol{\Psi}_c^{-1}\mathbf{W}_c)^{-1} \tag{12.19}$$

$$\mathbf{m}_{ic} \triangleq \boldsymbol{\Sigma}_{ic}(\mathbf{W}_c^T\boldsymbol{\Psi}_c^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_c)) \tag{12.20}$$

In the M step, it is easiest to estimate $\boldsymbol{\mu}_c$ and $\mathbf{W}_c$ at the same time, by defining $\tilde{\mathbf{W}}_c =$

$(\mathbf{W}_c, \boldsymbol{\mu}_c)$, $\tilde{\mathbf{z}} = (\mathbf{z}, 1)$, Also, define

$$\mathbf{b}_{ic} \triangleq \mathbb{E}\left[\tilde{\mathbf{z}}|\mathbf{x}_i, q_i = c\right] = \left[\mathbf{m}_{ic}; 1\right] \tag{12.21}$$

$$\mathbf{C}_{ic} \triangleq \mathbb{E}\left[\tilde{\mathbf{z}}\tilde{\mathbf{z}}^T|\mathbf{x}_i, q_i = c\right] = \begin{pmatrix} \mathbb{E}\left[\mathbf{z}\mathbf{z}^T|\mathbf{x}_i, q_i = c\right] & \mathbb{E}\left[\mathbf{z}|\mathbf{x}_i, q_i = c\right] \\ \mathbb{E}\left[\mathbf{z}|\mathbf{x}_i, q_i = c\right]^T & 1 \end{pmatrix} \tag{12.22}$$

Then the M step is as follows:

$$\hat{\tilde{\mathbf{W}}}_c = \left[\sum_i r_{ic}\mathbf{x}_i\mathbf{b}_{ic}^T\right]\left[\sum_i r_{ic}\mathbf{C}_{ic}\right]^{-1} \tag{12.23}$$

$$\hat{\boldsymbol{\Psi}} = \frac{1}{N}\mathrm{diag}\left\{\sum_{ic} r_{ic}\left(\mathbf{x}_i - \hat{\tilde{\mathbf{W}}}_c\mathbf{b}_{ic}\right)\mathbf{x}_i^T\right\} \tag{12.24}$$

$$\hat{\pi}_c = \frac{1}{N}\sum_{i=1}^{N} r_{ic} \tag{12.25}$$

Note that these updates are for "vanilla" EM. A much faster version of this algorithm, based on ECM, is described in (Zhao and Yu 2008).

12.1.6 Fitting FA models with missing data

In many applications, such as collaborative filtering, we have missing data. One virtue of the EM approach to fitting an FA/PPCA model is that it is easy to extend to this case. However, overfitting can be a problem if there is a lot of missing data. Consequently it is important to perform MAP estimation or to use Bayesian inference. See e.g., (Ilin and Raiko 2010) for details.

12.2 Principal components analysis (PCA)

Consider the FA model where we constrain $\boldsymbol{\Psi} = \sigma^2\mathbf{I}$, and $\mathbf{W}$ to be orthonormal. It can be shown (Tipping and Bishop 1999) that, as $\sigma^2 \to 0$, this model reduces to classical (non-probabilistic) **principal components analysis** (**PCA**), also known as the **Karhunen Loeve** transform. The version where $\sigma^2 > 0$ is known as **probabilistic PCA** (**PPCA**) (Tipping and Bishop 1999), or **sensible PCA** (Roweis 1997). (An equivalent result was derived independently, from a different perspective, in (Moghaddam and Pentland 1995).)

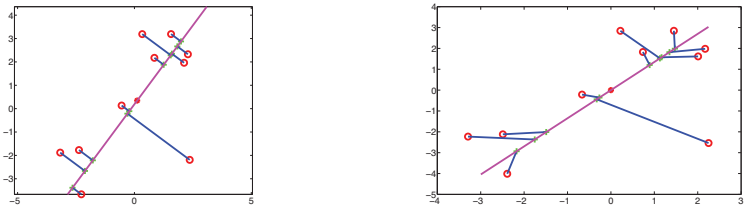
To make sense of this result, we first have to learn about classical PCA. We then connect PCA to the SVD. And finally we return to discuss PPCA.

12.2.1 Classical PCA: statement of the theorem

The **synthesis view** of classical PCA is summarized in the forllowing theorem.

Theorem 12.2.1. *Suppose we want to find an orthogonal set of L linear basis vectors $\mathbf{w}_j \in \mathbb{R}^D$, and the corresponding scores $\mathbf{z}_i \in \mathbb{R}^L$, such that we minimize the average* **reconstruction error**

$$J(\mathbf{W}, \mathbf{Z}) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{12.26}$$



(a) (b)

Figure 12.5 An illustration of PCA and PPCA where $D = 2$ and $L = 1$. Circles are the original data points, crosses are the reconstructions. The red star is the data mean. (a) PCA. The points are orthogonally projected onto the line. Figure generated by `pcaDemo2d`. (b) PPCA. The projection is no longer orthogonal: the reconstructions are shrunk towards the data mean (red star). Based on Figure 7.6 of (Nabney 2001). Figure generated by `ppcaDemo2d`.

where $\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i$*, subject to the constraint that* $\mathbf{W}$ *is orthonormal. Equivalently, we can write this objective as follows:*

$$J(\mathbf{W}, \mathbf{Z}) = ||\mathbf{X} - \mathbf{W}\mathbf{Z}^T||_F^2 \tag{12.27}$$

where $\mathbf{Z}$ *is an* $N \times L$ *matrix with the* $\mathbf{z}_i$ *in its rows, and* $||\mathbf{A}||_F$ *is the* **Frobenius norm** *of matrix* $\mathbf{A}$*, defined by*

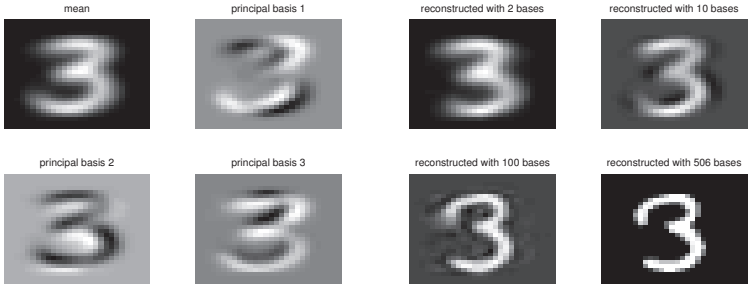
$$||\mathbf{A}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} = \sqrt{\mathrm{tr}(\mathbf{A}^T\mathbf{A})} = ||\mathbf{A}(:)||_2 \tag{12.28}$$

The optimal solution is obtained by setting $\hat{\mathbf{W}} = \mathbf{V}_L$*, where* $\mathbf{V}_L$ *contains the* L *eigenvectors with largest eigenvalues of the empirical covariance matrix,* $\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T$*. (We assume the* $\mathbf{x}_i$ *have zero mean, for notational simplicity.) Furthermore, the optimal low-dimensional encoding of the data is given by* $\hat{\mathbf{z}}_i = \mathbf{W}^T\mathbf{x}_i$*, which is an orthogonal projection of the data onto the column space spanned by the eigenvectors.*

An example of this is shown in Figure 12.5(a) for $D = 2$ and $L = 1$. The diagonal line is the vector $\mathbf{w}_1$; this is called the first principal component or principal direction. The data points $\mathbf{x}_i \in \mathbb{R}^2$ are orthogonally projected onto this line to get $z_i \in \mathbb{R}$. This is the best 1-dimensional approximation to the data. (We will discuss Figure 12.5(b) later.)

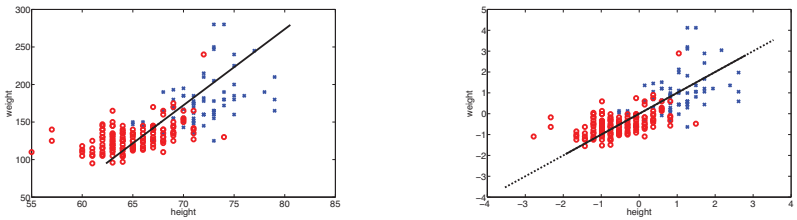
In general, it is hard to visualize higher dimensional data, but if the data happens to be a set of images, it is easy to do so. Figure 12.6 shows the first three principal vectors, reshaped as images, as well as the reconstruction of a specific image using a varying number of basis vectors. (We discuss how to choose L in Section 11.5.)

Below we will show that the principal directions are the ones along which the data shows maximal variance. This means that PCA can be "misled" by directions in which the variance is high merely because of the measurement scale. Figure 12.7(a) shows an example, where the vertical axis (weight) uses a large range than the horizontal axis (height), resulting in a line that looks somewhat "unnatural". It is therefore standard practice to standardize the data first, or



(a) (b)

Figure 12.6 (a) The mean and the first three PC basis vectors (eigendigits) based on 25 images of the digit 3 (from the MNIST dataset). (b) Reconstruction of an image based on 2, 10, 100 and all the basis vectors. Figure generated by `pcaImageDemo`.



(a) (b)

Figure 12.7 Effect of standardization on PCA applied to the height/ weight dataset. Left: PCA of raw data. Right: PCA of standardized data. Figure generated by `pcaDemoHeightWeight`.

equivalently, to work with correlation matrices instead of covariance matrices. The benefits of this are apparent from Figure 12.7(b).

12.2.2 Proof *

Proof. We use $\mathbf{w}_j \in \mathbb{R}^D$ to denote the j'th principal direction, $\mathbf{x}_i \in \mathbb{R}^D$ to denote the i'th high-dimensional observation, $\mathbf{z}_i \in \mathbb{R}^L$ to denote the i'th low-dimensional representation, and $\tilde{\mathbf{z}}_j \in \mathbb{R}^N$ to denote the $[z_{1j}, \ldots, z_{Nj}]$, which is the j'th component of all the low-dimensional vectors.

Let us start by estimating the best 1d solution, $\mathbf{w}_1 \in \mathbb{R}^D$, and the corresponding projected points $\tilde{\mathbf{z}}_1 \in \mathbb{R}^N$. We will find the remaining bases $\mathbf{w}_2$, $\mathbf{w}_3$, etc. later. The reconstruction error

is given by

$$J(\mathbf{w}_1, \mathbf{z}_1) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}_i - z_{i1}\mathbf{w}_1||^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - z_{i1}\mathbf{w}_1)^T(\mathbf{x}_i - z_{i1}\mathbf{w}_1) \tag{12.29}$$

$$= \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i^T\mathbf{x}_i - 2z_{i1}\mathbf{w}_1^T\mathbf{x}_i + z_{i1}^2\mathbf{w}_1^T\mathbf{w}_1] \tag{12.30}$$

$$= \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i^T\mathbf{x}_i - 2z_{i1}\mathbf{w}_1^T\mathbf{x}_i + z_{i1}^2] \tag{12.31}$$

since $\mathbf{w}_1^T\mathbf{w}_1 = 1$ (by the orthonormality assumption). Taking derivatives wrt z_{i1} and equating to zero gives

$$\frac{\partial}{\partial z_{i1}}J(\mathbf{w}_1, \mathbf{z}_1) = \frac{1}{N}[-2\mathbf{w}_1^T\mathbf{x}_i + 2z_{i1}] = 0 \Rightarrow z_{i1} = \mathbf{w}_1^T\mathbf{x}_i \tag{12.32}$$

So the optimal reconstruction weights are obtained by orthogonally projecting the data onto the first principal direction, $\mathbf{w}_1$ (see Figure 12.5(a)). Plugging back in gives

$$J(\mathbf{w}_1) = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i^T\mathbf{x}_i - z_{i1}^2] = \text{const} - \frac{1}{N}\sum_{i=1}^{N} z_{i1}^2 \tag{12.33}$$

Now the variance of the projected coordinates is given by

$$\text{var}\left[\tilde{\mathbf{z}}_1\right] = \mathbb{E}\left[\tilde{\mathbf{z}}_1^2\right] - \left(\mathbb{E}\left[\tilde{\mathbf{z}}_1\right]\right)^2 = \frac{1}{N}\sum_{i=1}^{N} z_{i1}^2 - 0 \tag{12.34}$$

since

$$\mathbb{E}\left[z_{i1}\right] = \mathbb{E}\left[\mathbf{x}_i^T\mathbf{w}_1\right] = \mathbb{E}\left[\mathbf{x}_i\right]^T\mathbf{w}_1 = 0 \tag{12.35}$$

because the data has been centered. From this, we see that *minimizing* the reconstruction error is equivalent to *maximizing* the variance of the projected data, i.e.,

$$\arg\min_{\mathbf{w}_1} J(\mathbf{w}_1) = \arg\max_{\mathbf{w}_1} \text{var}\left[\tilde{\mathbf{z}}_1\right] \tag{12.36}$$

This is why it is often said that PCA finds the directions of maximal variance. This is called the **analysis view** of PCA.

The variance of the projected data can be written as

$$\frac{1}{N}\sum_{i=1}^{N} z_{i1}^2 = \frac{1}{N}\sum_{i=1}^{N}\mathbf{w}_1^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{w}_1 = \mathbf{w}_1^T\hat{\mathbf{\Sigma}}\mathbf{w}_1 \tag{12.37}$$

where $\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}\sum_i \mathbf{x}_i\mathbf{x}_i^T$ is the empirical covariance matrix (or correlation matrix if the data is standardized).

We can trivially maximize the variance of the projection (and hence minimize the reconstruction error) by letting $||\mathbf{w}_1|| \to \infty$, so we impose the constraint $||\mathbf{w}_1|| = 1$ and instead maximize

$$\tilde{J}(\mathbf{w}_1) = \mathbf{w}_1^T \hat{\mathbf{\Sigma}} \mathbf{w}_1 + \lambda_1(\mathbf{w}_1^T \mathbf{w}_1 - 1) \tag{12.38}$$

where λ_1 is the Lagrange multiplier. Taking derivatives and equating to zero we have

$$\frac{\partial}{\partial \mathbf{w}_1} \tilde{J}(\mathbf{w}_1) = 2\hat{\mathbf{\Sigma}}\mathbf{w}_1 - 2\lambda_1 \mathbf{w}_1 = 0 \tag{12.39}$$

$$\hat{\mathbf{\Sigma}}\mathbf{w}_1 = \lambda_1 \mathbf{w}_1 \tag{12.40}$$

Hence the direction that maximizes the variance is an eigenvector of the covariance matrix. Left multiplying by $\mathbf{w}_1$ (and using $\mathbf{w}_1^T \mathbf{w}_1 = 1$) we find that the variance of the projected data is

$$\mathbf{w}_1^T \hat{\mathbf{\Sigma}} \mathbf{w}_1 = \lambda_1 \tag{12.41}$$

Since we want to maximize the variance, we pick the eigenvector which corresponds to the largest eigenvalue.

Now let us find another direction $\mathbf{w}_2$ to further minimize the reconstruction error, subject to $\mathbf{w}_1^T \mathbf{w}_2 = 0$ and $\mathbf{w}_2^T \mathbf{w}_2 = 1$. The error is

$$J(\mathbf{w}_1, \mathbf{z}_1, \mathbf{w}_2, \mathbf{z}_2) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - z_{i1}\mathbf{w}_1 - z_{i2}\mathbf{w}_2||^2 \tag{12.42}$$

Optimizing wrt $\mathbf{w}_1$ and $\mathbf{z}_1$ gives the same solution as before. Exercise 12.4 asks you to show that $\frac{\partial J}{\partial \mathbf{z}_2} = 0$ yields $z_{i2} = \mathbf{w}_2^T \mathbf{x}_i$. In other words, the second principal encoding is gotten by projecting onto the second principal direction. Substituting in yields

$$J(\mathbf{w}_2) = \frac{1}{n} \sum_{i=1}^{N} [\mathbf{x}_i^T \mathbf{x}_i - \mathbf{w}_1^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{w}_1 - \mathbf{w}_2^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{w}_2] = \text{const} - \mathbf{w}_2^T \hat{\mathbf{\Sigma}} \mathbf{w}_2 \tag{12.43}$$

Dropping the constant term and adding the constraints yields

$$\tilde{J}(\mathbf{w}_2) = -\mathbf{w}_2^T \hat{\mathbf{\Sigma}} \mathbf{w}_2 + \lambda_2(\mathbf{w}_2^T \mathbf{w}_2 - 1) + \lambda_{12}(\mathbf{w}_2^T \mathbf{w}_1 - 0) \tag{12.44}$$

Exercise 12.4 asks you to show that the solution is given by the eigenvector with the second largest eigenvalue:

$$\hat{\mathbf{\Sigma}}\mathbf{w}_2 = \lambda_2 \mathbf{w}_2 \tag{12.45}$$

The proof continues in this way. (Formally one can use induction.) □

12.2.3 Singular value decomposition (SVD)

We have defined the solution to PCA in terms of eigenvectors of the covariance matrix. However, there is another way to obtain the solution, based on the **singular value decomposition**, or **SVD**. This basically generalizes the notion of eigenvectors from square matrices to any kind of matrix.

In particular, any (real) $N \times D$ matrix $\mathbf{X}$ can be decomposed as follows

$$\underbrace{\mathbf{X}}_{N\times D} = \underbrace{\mathbf{U}}_{N\times N}\underbrace{\mathbf{S}}_{N\times D}\underbrace{\mathbf{V}^T}_{D\times D} \tag{12.46}$$

where $\mathbf{U}$ is an $N \times N$ matrix whose columns are orthornormal (so $\mathbf{U}^T\mathbf{U} = \mathbf{I}_N$), $\mathbf{V}$ is $D \times D$ matrix whose rows and columns are orthonormal (so $\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}_D$), and $\mathbf{S}$ is a $N \times D$ matrix containing the $r = \min(N, D)$ **singular values** $\sigma_i \geq 0$ on the main diagonal, with 0s filling the rest of the matrix. The columns of $\mathbf{U}$ are the left singular vectors, and the columns of $\mathbf{V}$ are the right singular vectors. See Figure 12.8(a) for an example.

Since there are at most D singular values (assuming $N > D$), the last $N - D$ columns of $\mathbf{U}$ are irrelevant, since they will be multiplied by 0. The **economy sized SVD**, or **thin SVD**, avoids computing these unnecessary elements. Let us denote this decomposition by $\hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}$. If $N > D$, we have

$$\underbrace{\mathbf{X}}_{N\times D} = \underbrace{\hat{\mathbf{U}}}_{N\times D}\underbrace{\hat{\mathbf{S}}}_{D\times D}\underbrace{\hat{\mathbf{V}}^T}_{D\times D} \tag{12.47}$$

as in Figure 12.8(a). If $N < D$, we have

$$\underbrace{\mathbf{X}}_{N\times D} = \underbrace{\hat{\mathbf{U}}}_{N\times N}\underbrace{\hat{\mathbf{S}}}_{N\times N}\underbrace{\hat{\mathbf{V}}^T}_{N\times D} \tag{12.48}$$

Computing the economy-sized SVD takes $O(ND\min(N, D))$ time (Golub and van Loan 1996, p254).

The connection between eigenvectors and singular vectors is the following. For an arbitrary real matrix $\mathbf{X}$, if $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, we have

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{S}^T\mathbf{U}^T\,\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{V}(\mathbf{S}^T\mathbf{S})\mathbf{V}^T = \mathbf{V}\mathbf{D}\mathbf{V}^T \tag{12.49}$$

where $\mathbf{D} = \mathbf{S}^2$ is a diagonal matrix containing the squares singular values. Hence

$$(\mathbf{X}^T\mathbf{X})\mathbf{V} = \mathbf{V}\mathbf{D} \tag{12.50}$$

so the eigenvectors of $\mathbf{X}^T\mathbf{X}$ are equal to $\mathbf{V}$, the right singular vectors of $\mathbf{X}$, and the eigenvalues of $\mathbf{X}^T\mathbf{X}$ are equal to $\mathbf{D}$, the squared singular values. Similarly

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T\,\mathbf{V}\mathbf{S}^T\mathbf{U}^T = \mathbf{U}(\mathbf{S}\mathbf{S}^T)\mathbf{U}^T \tag{12.51}$$

$$(\mathbf{X}\mathbf{X}^T)\mathbf{U} = \mathbf{U}(\mathbf{S}\mathbf{S}^T) = \mathbf{U}\mathbf{D} \tag{12.52}$$

so the eigenvectors of $\mathbf{X}\mathbf{X}^T$ are equal to $\mathbf{U}$, the left singular vectors of $\mathbf{X}$. Also, the eigenvalues of $\mathbf{X}\mathbf{X}^T$ are equal to the squared singular values. We can summarize all this as follows:

$$\mathbf{U} = \text{evec}(\mathbf{X}\mathbf{X}^T),\ \mathbf{V} = \text{evec}(\mathbf{X}^T\mathbf{X}),\ \mathbf{S}^2 = \text{eval}(\mathbf{X}\mathbf{X}^T) = \text{eval}(\mathbf{X}^T\mathbf{X}) \tag{12.53}$$

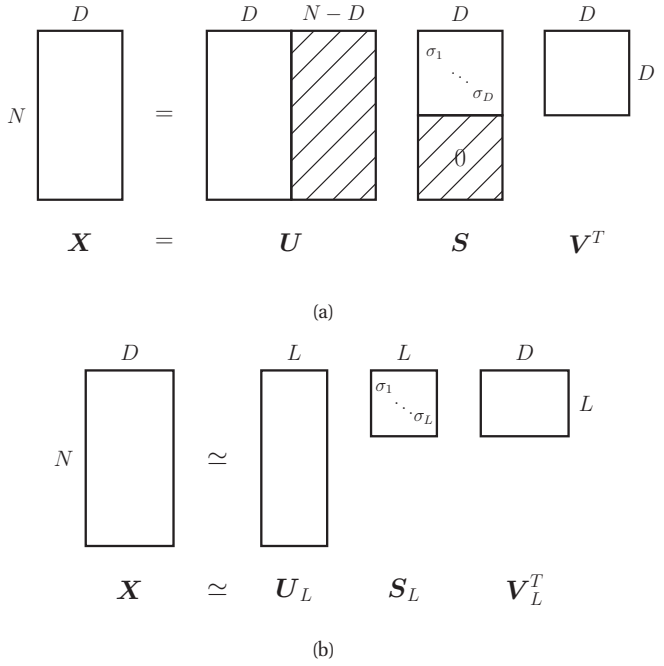


Figure 12.8 (a) SVD decomposition of non-square matrices $\mathbf{X} = \mathbf{USV}^T$. The shaded parts of $\mathbf{S}$, and all the off-diagonal terms, are zero. The shaded entries in $\mathbf{U}$ and $\mathbf{S}$ are not computed in the economy-sized version, since they are not needed. (b) Truncated SVD approximation of rank L.

Since the eigenvectors are unaffected by linear scaling of a matrix, we see that the right singular vectors of $\mathbf{X}$ are equal to the eigenvectors of the empirical covariance $\hat{\mathbf{\Sigma}}$. Furthermore, the eigenvalues of $\hat{\mathbf{\Sigma}}$ are a scaled version of the squared singular values. This means we can perform PCA using just a few lines of code (see `pcaPmtk`).

However, the connection between PCA and SVD goes deeper. From Equation 12.46, we can represent a rank r matrix as follows:

$$\mathbf{X} = \sigma_1 \begin{pmatrix} | \\ \mathbf{u}_1 \\ | \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_1^T & - \end{pmatrix} + \cdots + \sigma_r \begin{pmatrix} | \\ \mathbf{u}_r \\ | \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_r^T & - \end{pmatrix} \tag{12.54}$$

If the singular values die off quickly as in Figure 12.10, we can produce a rank L approximation to the matrix as follows:

$$\mathbf{X} \approx \mathbf{U}_{:,1:L} \ \mathbf{S}_{1:L,1:L} \ \mathbf{V}_{:,1:L}^T \tag{12.55}$$

This is called a **truncated SVD** (see Figure 12.8(b)). The total number of parameters needed to represent an $N \times D$ matrix using a rank L approximation is

$$NL + LD + L = L(N + D + 1) \tag{12.56}$$

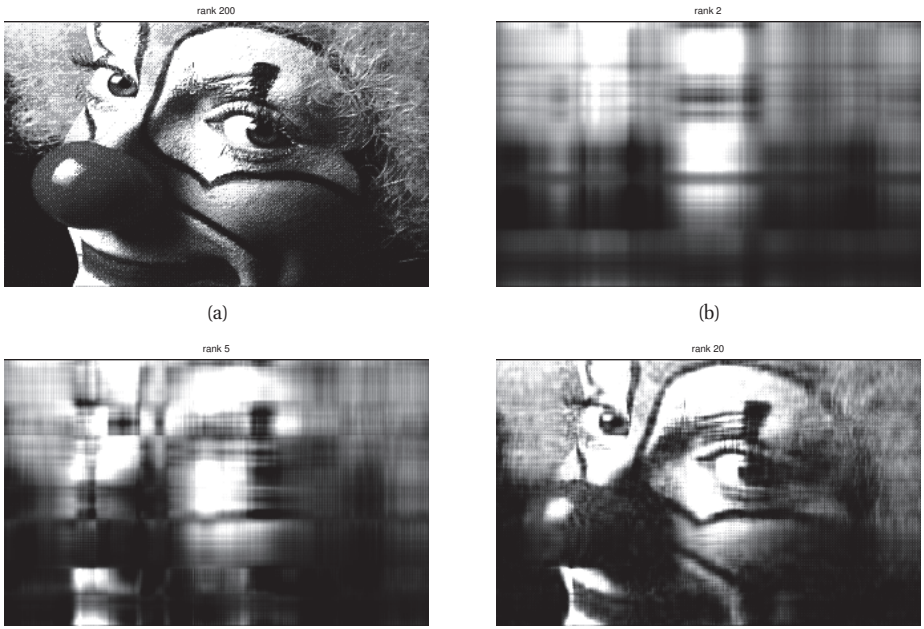


Figure 12.9 Low rank approximations to an image. Top left: The original image is of size 200×320, so has rank 200. Subsequent images have ranks 2, 5, and 20. Figure generated by `svdImageDemo`.

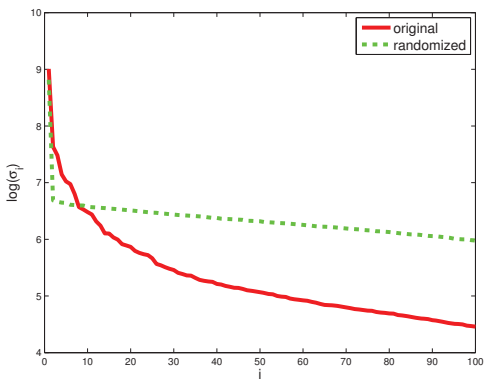


Figure 12.10 First 50 log singular values for the clown image (solid red line), and for a data matrix obtained by randomly shuffling the pixels (dotted green line). Figure generated by `svdImageDemo`.

As an example, consider the 200×320 pixel image in Figure 12.9(top left). This has 64,000 numbers in it. We see that a rank 20 approximation, with only $(200 + 320 + 1) \times 20 = 10,420$ numbers is a very good approximation.

One can show that the error in this approximation is given by

$$||\mathbf{X} - \mathbf{X}_L||_F \approx \sigma_{L+1} \tag{12.57}$$

Furthermore, one can show that the SVD offers the best rank L approximation to a matrix (best in the sense of minimizing the above Frobenius norm).

Let us connect this back to PCA. Let $\mathbf{X} = \mathbf{USV}^T$ be a truncated SVD of $\mathbf{X}$. We know that $\hat{\mathbf{W}} = \mathbf{V}$, and that $\hat{\mathbf{Z}} = \mathbf{X}\hat{\mathbf{W}}$, so

$$\hat{\mathbf{Z}} = \mathbf{USV}^T\mathbf{V} = \mathbf{US} \tag{12.58}$$

Furthermore, the optimal reconstruction is given by $\hat{\mathbf{X}} = \mathbf{Z}\hat{\mathbf{W}}^T$, so we find

$$\hat{\mathbf{X}} = \mathbf{USV}^T \tag{12.59}$$

This is precisely the same as a truncated SVD approximation! This is another illustration of the fact that PCA is the best low rank approximation to the data.

12.2.4 Probabilistic PCA

We are now ready to revisit PPCA. One can show the following remarkable result.

Theorem 12.2.2 ((Tipping and Bishop 1999)). *Consider a factor analysis model in which $\mathbf{\Psi} = \sigma^2\mathbf{I}$ and $\mathbf{W}$ is orthogonal. The observed data log likelihood is given by*

$$\log p(\mathbf{X}|\mathbf{W}, \sigma^2) = -\frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{i=1}^{N}\mathbf{x}_i^T\mathbf{C}^{-1}\mathbf{x}_i = -\frac{N}{2}\ln|\mathbf{C}| + \mathrm{tr}(\mathbf{C}^{-1}\hat{\mathbf{\Sigma}}) \tag{12.60}$$

where $\mathbf{C} = \mathbf{WW}^T + \sigma^2\mathbf{I}$ and $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T = (1/N)\mathbf{X}^T\mathbf{X}$. (We are assuming centered data, for notational simplicity.) The maxima of the log-likelihood are given by

$$\hat{\mathbf{W}} = \mathbf{V}(\mathbf{\Lambda} - \sigma^2\mathbf{I})^{\frac{1}{2}}\mathbf{R} \tag{12.61}$$

where $\mathbf{R}$ is an arbitrary $L \times L$ orthogonal matrix, $\mathbf{V}$ is the $D \times L$ matrix whose columns are the first L eigenvectors of $\mathbf{S}$, and $\mathbf{\Lambda}$ is the corresponding diagonal matrix of eigenvalues. Without loss of generality, we can set $\mathbf{R} = \mathbf{I}$. Furthermore, the MLE of the noise variance is given by

$$\hat{\sigma}^2 = \frac{1}{D-L}\sum_{j=L+1}^{D}\lambda_j \tag{12.62}$$

which is the average variance associated with the discarded dimensions.

Thus, as $\sigma^2 \to 0$, we have $\hat{\mathbf{W}} \to \mathbf{V}$, as in classical PCA. What about $\hat{\mathbf{Z}}$? It is easy to see that the posterior over the latent factors is given by

$$p(\mathbf{z}_i|\mathbf{x}_i, \hat{\boldsymbol{\theta}}) = \mathcal{N}(\mathbf{z}_i|\hat{\mathbf{F}}^{-1}\hat{\mathbf{W}}^T\mathbf{x}_i, \sigma^2\hat{\mathbf{F}}^{-1}) \tag{12.63}$$

$$\hat{\mathbf{F}} \triangleq \hat{\mathbf{W}}^T\hat{\mathbf{W}} + \hat{\sigma}^2\mathbf{I} \tag{12.64}$$

(Do not confuse $\mathbf{F} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}$ with $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$.) Hence, as $\sigma^2 \to 0$, we find $\hat{\mathbf{W}} \to \mathbf{V}$, $\hat{\mathbf{F}} \to \mathbf{I}$ and $\hat{\mathbf{z}}_i \to \mathbf{V}^T\mathbf{x}_i$. Thus the posterior mean is obtained by an orthogonal projection of the data onto the column space of $\mathbf{V}$, as in classical PCA.

Note, however, that if $\sigma^2 >$, the posterior mean is not an orthogonal projection, since it is shrunk somewhat towards the prior mean, as illustrated in Figure 12.5(b). This sounds like an undesirable property, but it means that the reconstructions will be closer to the overall data mean, $\hat{\boldsymbol{\mu}} = \overline{\mathbf{x}}$.

12.2.5 EM algorithm for PCA

Although the usual way to fit a PCA model uses eigenvector methods, or the SVD, we can also use EM, which will turn out to have some advantages that we discuss below. EM for PCA relies on the probabilistic formulation of PCA. However the algorithm continues to work in the zero noise limit, $\sigma^2 = 0$, as shown by (Roweis 1997).

Let $\tilde{\mathbf{Z}}$ be a $L \times N$ matrix storing the posterior means (low-dimensional representations) along its columns. Similarly, let $\tilde{\mathbf{X}} = \mathbf{X}^T$ store the original data along its columns. From Equation 12.63, when $\sigma^2 = 0$, we have

$$\tilde{\mathbf{Z}} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\tilde{\mathbf{X}} \tag{12.65}$$

This constitutes the E step. Notice that this is just an orthogonal projection of the data.

From Equation 12.23, the M step is given by

$$\hat{\mathbf{W}} = \left[\sum_i \mathbf{x}_i \mathbb{E}\left[\mathbf{z}_i\right]^T\right]\left[\sum_i \mathbb{E}\left[\mathbf{z}_i\right]\mathbb{E}\left[\mathbf{z}_i\right]^T\right]^{-1} \tag{12.66}$$

where we exploited the fact that $\boldsymbol{\Sigma} = \mathrm{cov}\left[\mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta}\right] = 0\mathbf{I}$ when $\sigma^2 = 0$. It is worth comparing this expression to the MLE for multi-output linear regression (Equation 7.89), which has the form $\mathbf{W} = (\sum_i \mathbf{y}_i\mathbf{x}_i^T)(\sum_i \mathbf{x}_i\mathbf{x}_i^T)^{-1}$. Thus we see that the M step is like linear regression where we replace the observed inputs by the expected values of the latent variables.

In summary, here is the entire algorithm:

- **E step** $\tilde{\mathbf{Z}} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\tilde{\mathbf{X}}$
- **M step** $\mathbf{W} = \tilde{\mathbf{X}}\tilde{\mathbf{Z}}^T(\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}^T)^{-1}$

(Tipping and Bishop 1999) showed that the only stable fixed point of the EM algorithm is the globally optimal solution. That is, the EM algorithm converges to a solution where $\mathbf{W}$ spans the same linear subspace as that defined by the first L eigenvectors. However, if we want $\mathbf{W}$ to be orthogonal, and to contain the eigenvectors in descending order of eigenvalue, we have to orthogonalize the resulting matrix (which can be done quite cheaply). Alternatively, we can modify EM to give the principal basis directly (Ahn and Oh 2003).

This algorithm has a simple physical analogy in the case $D = 2$ and $L = 1$ (Roweis 1997). Consider some points in $\mathbb{R}^2$ attached by springs to a rigid rod, whose orientation is defined by a vector $\mathbf{w}$. Let z_i be the location where the i'th spring attaches to the rod. In the E step, we hold the rod fixed, and let the attachment points slide around so as to minimize the spring energy (which is proportional to the sum of squared residuals). In the M step, we hold the attachment

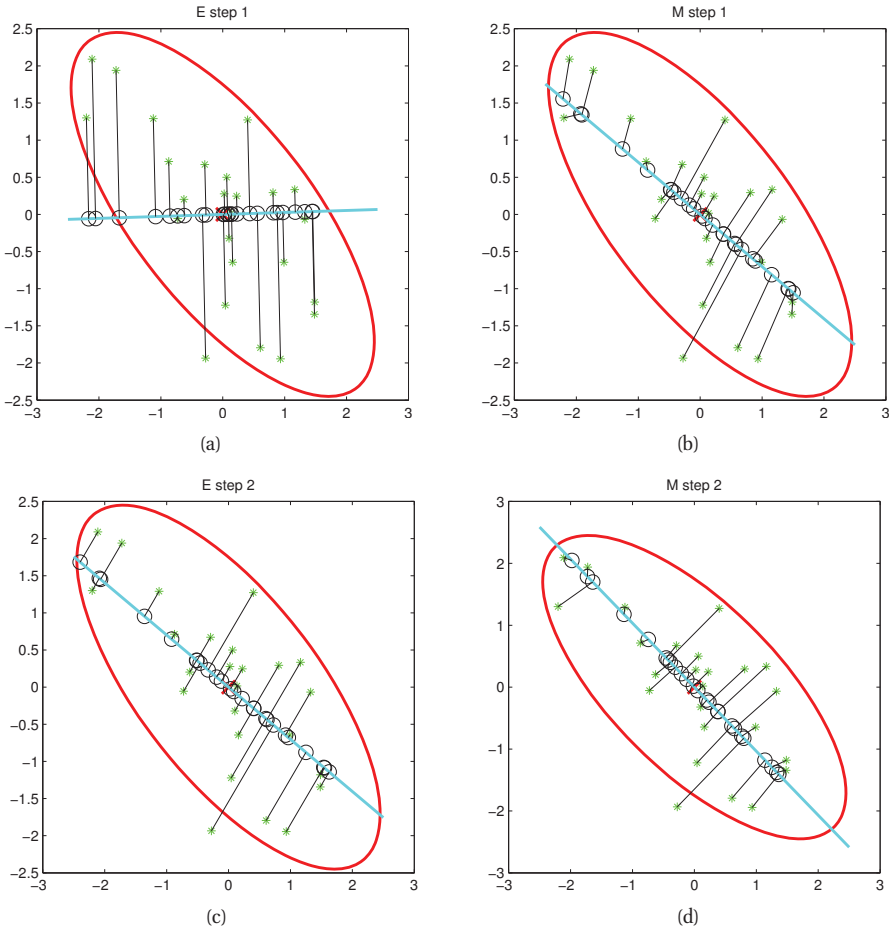


Figure 12.11 Illustration of EM for PCA when $D = 2$ and $L = 1$. Green stars are the original data points, black circles are their reconstructions. The weight vector $\mathbf{w}$ is represented by blue line. (a) We start with a random initial guess of $\mathbf{w}$. The E step is represented by the orthogonal projections. (b) We update the rod $\mathbf{w}$ in the M step, keeping the projections onto the rod (black circles) fixed. (c) Another E step. The black circles can 'slide' along the rod, but the rod stays fixed. (d) Another M step. Based on Figure 12.12 of (Bishop 2006b). Figure generated by `pcaEmStepByStep`.

points fixed and let the rod rotate so as to minimize the spring energy. See Figure 12.11 for an illustration.

Apart from this pleasing intuitive interpretation, EM for PCA has the following advantages over eigenvector methods:

- EM can be faster. In particular, assuming $N, D \gg L$, the dominant cost of EM is the projection operation in the E step, so the overall time is $O(TLND)$, where T is the number of

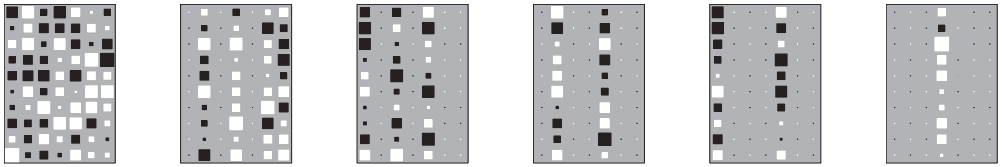


Figure 12.12 Illustration of estimating the effective dimensionalities in a mixture of factor analysers using VBEM. The blank columns have been forced to 0 via the ARD mechanism. The data was generated from 6 clusters with intrinsic dimensionalities of $7, 4, 3, 2, 2, 1$, which the method has successfully estimated. Source: Figure 4.4 of (Beal 2003). Used with kind permission of Matt Beal.

iterations. (Roweis 1997) showed experimentally that the number of iterations is usually very small (the mean was 3.6), regardless of N or D. (This results depends on the ratio of eigenvalues of the empirical covariance matrix.) This is much faster than the $O(\min(ND^2, DN^2))$ time required by straightforward eigenvector methods, although more sophisticated eigenvector methods, such as the Lanczos algorithm, have running times comparable to EM.

- EM can be implemented in an online fashion, i.e., we can update our estimate of $\mathbf{W}$ as the data streams in.
- EM can handle missing data in a simple way (see Section 12.1.6).
- EM can be extended to handle mixtures of PPCA/ FA models.
- EM can be modified to variational EM or to variational Bayes EM to fit more complex models.

12.3 Choosing the number of latent dimensions

In Section 11.5, we discussed how to choose the number of components K in a mixture model. In this section, we discuss how to choose the number of latent dimensions L in a FA/PCA model.

12.3.1 Model selection for FA/PPCA

If we use a probabilistic model, we can in principle compute $L^* = \text{argmax}_L \, p(L|\mathcal{D})$. However, there are two problems with this. First, evaluating the marginal likelihood for LVMs is quite difficult. In practice, simple approximations, such as BIC or variational lower bounds (see Section 21.5), can be used (see also (Minka 2000a)). Alternatively, we can use the cross-validated likelihood as a performance measure, although this can be slow, since it requires fitting each model F times, where F is the number of CV folds.

The second issue is the need to search over a potentially large number of models. The usual approach is to perform exhaustive search over all candidate values of L. However, sometimes we can set the model to its maximal size, and then use a technique called automatic relevancy determination (Section 13.7), combined with EM, to automatically prune out irrelevant weights.

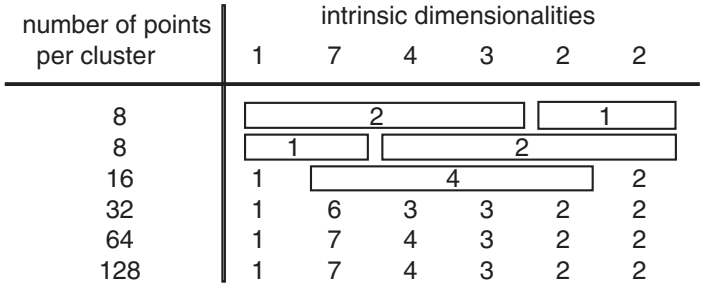


Figure 12.13 We show the estimated number of clusters, and their estimated dimensionalities, as a function of sample size. The VBEM algorithm found two different solutions when $N = 8$. Note that more clusters, with larger effective dimensionalities, are discovered as the sample sizes increases. Source: Table 4.1 of (Beal 2003). Used with kind permission of Matt Beal.

This technique will be described in a supervised context in Chapter 13, but can be adapted to the (M)FA context as shown in (Bishop 1999; Ghahramani and Beal 2000).

Figure 12.12 illustrates this approach applied to a mixture of FAs fit to a small synthetic dataset. The figures visualize the weight matrices for each cluster, using **Hinton diagrams**, where where the size of the square is proportional to the value of the entry in the matrix.[2] We see that many of them are sparse. Figure 12.13 shows that the degree of sparsity depends on the amount of training data, in accord with the Bayesian Occam's razor. In particular, when the sample size is small, the method automatically prefers simpler models, but as the sample size gets sufficiently large, the method converges on the "correct" solution, which is one with 6 subspaces of dimensionality 1, 2, 2, 3, 4 and 7.

Although the ARD/ EM method is elegant, it still needs to perform search over K. This is done using "birth" and "death" moves (Ghahramani and Beal 2000). An alternative approach is to perform stochastic sampling in the space of models. Traditional approaches, such as (Lopes and West 2004), are based on reversible jump MCMC, and also use birth and death moves. However, this can be slow and difficult to implement. More recent approaches use non-parametric priors, combined with Gibbs sampling, see e.g., (Paisley and Carin 2009).

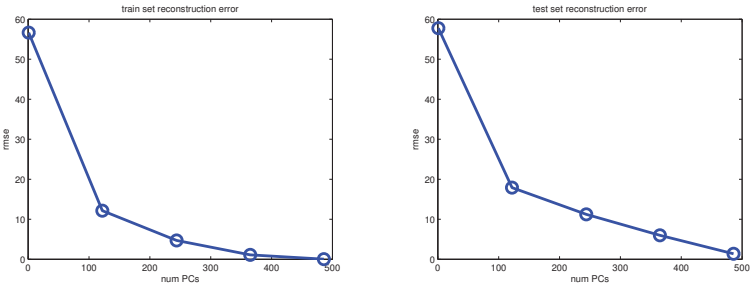
12.3.2 Model selection for PCA

Since PCA is not a probabilistic model, we cannot use any of the methods described above. An obvious proxy for the likelihood is the reconstruction error:

$$E(\mathcal{D}, L) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{12.67}$$

In the case of PCA, the reconstruction is given by by $\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}$, where $\mathbf{z}_i = \mathbf{W}^T(\mathbf{x}_i - \boldsymbol{\mu})$ and $\mathbf{W}$ and $\boldsymbol{\mu}$ are estimated from $\mathcal{D}_{\text{train}}$.

2. Geoff Hinton is an English professor of computer science at the University of Toronto.



(a) (b)

Figure 12.14 Reconstruction error on MNIST vs number of latent dimensions used by PCA. (a) Training set. (b) Test set. Figure generated by `pcaOverfitDemo`.

Figure 12.14(a) plots $E(\mathcal{D}_{\text{train}}, L)$ vs L on the MNIST training data in Figure 12.6. We see that it drops off quite quickly, indicating that we can capture most of the empirical correlation of the pixels with a small number of factors, as illustrated qualitatively in Figure 12.6.

Exercise 12.5 asks you to prove that the residual error from only using L terms is given by the sum of the discarded eigenvalues:

$$E(\mathcal{D}_{\text{train}}, L) \quad = \quad \sum_{j=L+1}^{D} \lambda_j \tag{12.68}$$

Therefore an alternative to plotting the error is to plot the retained eigenvalues, in decreasing order. This is called a **scree plot**, because "the plot looks like the side of a mountain, and 'scree' refers to the debris fallen from a mountain and lying at its base".[3] This will have the same shape as the residual error plot.

A related quantity is the **fraction of variance explained**, defined as

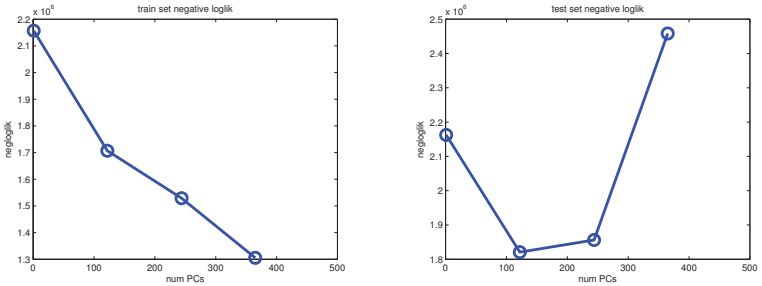
$$F(\mathcal{D}_{\text{train}}, L) = \frac{\sum_{j=1}^{L} \lambda_j}{\sum_{j'=1}^{L_{max}} \lambda_{j'}} \tag{12.69}$$

This captures the same information as the scree plot.

Of course, if we use $L = \text{rank}(\mathbf{X})$, we get zero reconstruction error on the training set. To avoid overfitting, it is natural to plot reconstruction error on the test set. This is shown in Figure 12.14(b). Here we see that the error continues to go down even as the model becomes more complex! Thus we do not get the usual U-shaped curve that we typically expect to see.

What is going on? The problem is that PCA is not a proper generative model of the data. It is merely a compression technique. If you give it more latent dimensions, it will be able to approximate the test data more accurately. By contrast, a probabilistic model enjoys a Bayesian Occam's razor effect (Section 5.3.1), in that it gets "punished" if it wastes probability mass on parts of the space where there is little data. This is illustrated in Figure 12.15, which plots the

3. Quotation from `http://janda.org/workshop/factoranalysis/SPSSrun/SPSS08.htm`.



(a) (b)

Figure 12.15 Negative log likelihood on MNIST vs number of latent dimensions used by PPCA. (a) Training set. (b) Test set. Figure generated by `pcaOverfitDemo`.

negative log likelihood, computed using PPCA, vs L. Here, on the test set, we see the usual U-shaped curve.

These results are analogous to those in Section 11.5.2, where we discussed the issue of choosing K in the K-means algorithm vs using a GMM.

12.3.2.1 Profile likelihood

Although there is no U-shape, there is sometimes a "regime change" in the plots, from relatively large errors to relatively small. One way to automate the detection of this is described in (Zhu and Ghodsi 2006). The idea is this. Let λ_k be some measure of the error incurred by a model of size k, such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{L_{max}}$. In PCA, these are the eigenvalues, but the method can also be applied to K-means. Now consider partitioning these values into two groups, depending on whether $k < L$ or $k > L$, where L is some threshold which we will determine. To measure the quality of L, we will use a simple change-point model, where $\lambda_k \sim \mathcal{N}(\mu_1, \sigma^2)$ if $k \leq L$, and $\lambda_k \sim \mathcal{N}(\mu_2, \sigma^2)$ if $k > L$. (It is important that σ^2 be the same in both models, to prevent overfitting in the case where one regime has less data than the other.) Within each of the two regimes, we assume the λ_k are iid, which is obviously incorrect, but is adequate for our present purposes. We can fit this model for each $L = 1 : L_{max}$ by partitioning the data and computing the MLEs, using a pooled estimate of the variance:

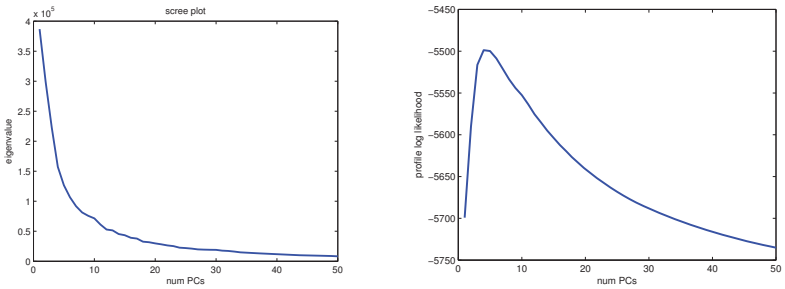
$$\mu_1(L) = \frac{\sum_{k \leq L} \lambda_k}{L}, \; \mu_2(L) = \frac{\sum_{k > L} \lambda_k}{N - L} \tag{12.70}$$

$$\sigma^2(L) = \frac{\sum_{k \leq L} (\lambda_k - \mu_1(L))^2 + \sum_{k > L} (\lambda_k - \mu_2(L))^2}{N} \tag{12.71}$$

We can then evaluate the **profile log likelihood**

$$\ell(L) = \sum_{k=1}^{L} \log \mathcal{N}(\lambda_k | \mu_1(L), \sigma^2(L)) + \sum_{k=L+1}^{K} \log \mathcal{N}(\lambda_k | \mu_2(L), \sigma^2(L)) \tag{12.72}$$

Finally, we choose $L^* = \arg\max \ell(L)$. This is illustrated in Figure 12.16. On the left, we plot the scree plot, which has the same shape as in Figure 12.14(a). On the right, we plot the profile



(a) (b)

Figure 12.16 (a) Scree plot for training set, corresponding to Figure 12.14(a). (b) Profile likelihood. Figure generated by `pcaOverfitDemo`.

likelihood. Rather miraculously, we see a fairly well-determined peak.

12.4 PCA for categorical data

In this section, we consider extending the factor analysis model to the case where the observed data is categorical rather than real-valued. That is, the data has the form $y_{ij} \in \{1, \ldots, C\}$, where $j = 1 : R$ is the number of observed response variables. We assume each y_{ij} is generated from a latent variable $\mathbf{z}_i \in \mathbb{R}^L$, with a Gaussian prior, which is passed through the softmax function as follows:

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{12.73}$$

$$p(\mathbf{y}_i|\mathbf{z}_i, \boldsymbol{\theta}) = \prod_{r=1}^{R} \mathrm{Cat}(y_{ir}|\mathcal{S}(\mathbf{W}_r^T \mathbf{z}_i + \mathbf{w}_{0r})) \tag{12.74}$$

where $\mathbf{W}_r \in \mathbb{R}^{L \times M}$ is the factor loading matrix for response j, and $\mathbf{w}_{0r} \in \mathbb{R}^M$ is the offset term for response r, and $\boldsymbol{\theta} = (\mathbf{W}_r, \mathbf{w}_{0r})_{r=1}^R$. (We need an explicit offset term, since clamping one element of $\mathbf{z}_i$ to 1 can cause problems when computing the posterior covariance.) As in factor analysis, we have defined the prior mean to be $\mathbf{m}_0 = \mathbf{0}$ and the prior covariance $\mathbf{V}_0 = \mathbf{I}$, since we can capture non-zero mean by changing $\mathbf{w}_{0j}$ and non-identity covariance by changing $\mathbf{W}_r$. We will call this **categorical PCA**. See Chapter 27 for a discussion of related models.

It is interesting to study what kinds of distributions we can induce on the observed variables by varying the parameters. For simplicity, we assume there is a single ternary response variable, so y_i lives in the 3d probability simplex. Figure 12.17 shows what happens when we vary the parameters of the prior, $\mathbf{m}_0$ and $\mathbf{V}_0$, which is equivalent to varying the parameters of the likelihood, $\mathbf{W}_1$ and $\mathbf{w}_{01}$. We see that this can define fairly complex distributions over the simplex. This induced distribution is known as the **logistic normal** distribution (Aitchison 1982).

We can fit this model to data using a modified version of EM. The basic idea is to infer a Gaussian approximation to the posterior $p(\mathbf{z}_i|\mathbf{y}_i, \boldsymbol{\theta})$ in the E step, and then to maximize $\boldsymbol{\theta}$ in the M step. The details for the multiclass case, can be found in (Khan et al. 2010) (see

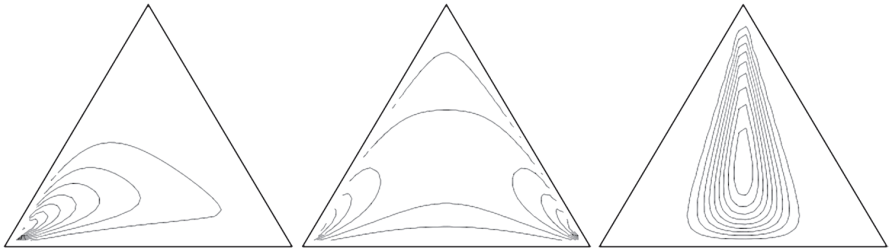
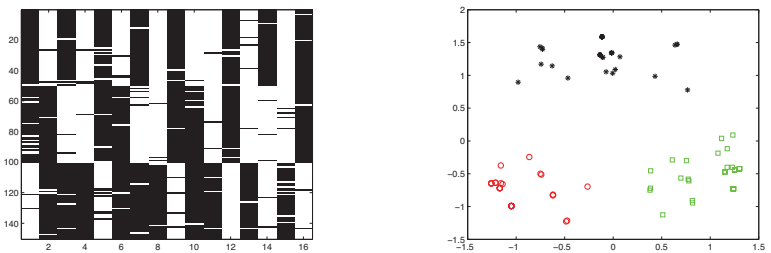


Figure 12.17 Some examples of the logistic normal distribution defined on the 3d simplex. (a) Diagonal covariance and non-zero mean. (b) Negative correlation between states 1 and 2. (c) Positive correlation between states 1 and 2. Source: Figure 1 of (Blei and Lafferty 2007). Used with kind permission of David Blei.



(a) (b)

Figure 12.18 Left: 150 synthetic 16 dimensional bit vectors. Right: the 2d embedding learned by binary PCA, using variational EM. We have color coded points by the identity of the true "prototype" that generated them. Figure generated by `binaryFaDemoTipping`.

also Section 21.8.1.1). The details for the binary case for the the sigmoid link can be found in Exercise 21.9, and for the probit link in Exercise 21.10.

One application of such a model is to visualize high dimensional categorical data. Figure 12.18(a) shows a simple example where we have 150 6-dimensional bit vectors. It is clear that each sample is just a noisy copy of one of three binary prototypes. We fit a 2d catFA to this model, yielding approximate MLEs $\hat{\boldsymbol{\theta}}$. In Figure 12.18(b), we plot $\mathbb{E}\left[\mathbf{z}_i|\mathbf{x}_i, \hat{\boldsymbol{\theta}}\right]$. We see that there are three distinct clusters, as is to be expected.

In (Khan et al. 2010), we show that this model outperforms finite mixture models on the task of imputing missing entries in design matrices consisting of real and categorical data. This is useful for analysing social science survey data, which often has missing data and variables of mixed type.

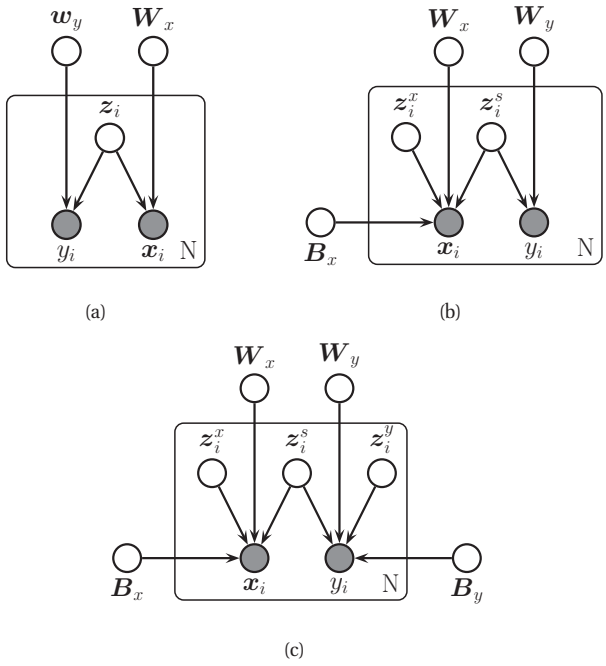


Figure 12.19 Gaussian latent factor models for paired data. (a) Supervised PCA. (b) Partial least squares. (c) Canonical correlation analysis.

12.5 PCA for paired and multi-view data

It is common to have a pair of related datasets, e.g., gene expression and gene copy number, or movie ratings by users and movie reviews. It is natural to want to combine these together into a low-dimensional embedding. This is an example of **data fusion**. In some cases, we might want to predict one element of the pair, say $\mathbf{x}_{i1}$, from the other one, $\mathbf{x}_{i2}$, via the low-dimensional "bottleneck".

Below we discuss various latent Gaussian models for these tasks, following the presentation of (Virtanen 2010). The models easily generalize from pairs to sets of data, $\mathbf{x}_{im}$, for $m = 1 : M$. We focus on the case where $\mathbf{x}_{im} \in \mathbb{R}^{D_m}$. In this case, the joint distribution is multivariate Gaussian, so we can easily fit the models using EM, or Gibbs sampling.

We can generalize the models to handle discrete and count data by using the exponential family as a response distribution instead of the Gaussian, as we explain in Section 27.2.2. However, this will require the use of approximate inference in the E step (or an analogous modification to MCMC).

12.5.1 Supervised PCA (latent factor regression)

Consider the following model, illustrated in Figure 12.19(a):

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{0}, \mathbf{I}_L) \tag{12.75}$$
$$p(y_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{w}_y^T \mathbf{z}_i + \mu_y, \sigma_y^2) \tag{12.76}$$
$$p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{W}_x \mathbf{z}_i + \boldsymbol{\mu}_x, \sigma_x^2 \mathbf{I}_D) \tag{12.77}$$

In (Yu et al. 2006), this is called **supervised PCA**. In (West 2003), this is called **Bayesian factor regression**. This model is like PCA, except that the target variable y_i is taken into account when learning the low dimensional embedding. Since the model is jointly Gaussian, we have

$$y_i|\mathbf{x}_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w}, \sigma_y^2 + \mathbf{w}_y^T \mathbf{C} \mathbf{w}_y) \tag{12.78}$$

where $\mathbf{w} = \boldsymbol{\Psi}^{-1} \mathbf{W}_x \mathbf{C} \mathbf{w}_y$, $\boldsymbol{\Psi} = \sigma_x^2 \mathbf{I}_D$, and $\mathbf{C}^{-1} = \mathbf{I} + \mathbf{W}_x^T \boldsymbol{\Psi}^{-1} \mathbf{W}_x$. So although this is a joint density model of $(y_i, \mathbf{x}_i)$, we can infer the implied conditional distribution.

We now show an interesting connection to Zellner's g-prior. Suppose $p(\mathbf{w}_y) = \mathcal{N}(\mathbf{0}, \frac{1}{g}\boldsymbol{\Sigma}^2)$, and let $\mathbf{X} = \mathbf{R}\mathbf{V}^T$ be the SVD of $\mathbf{X}$, where $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{R}^T\mathbf{R} = \boldsymbol{\Sigma}^2 = \text{diag}(\sigma_j^2)$ contains the squared singular values. Then one can show (West 2003) that

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, g\mathbf{V}^{-T}\boldsymbol{\Sigma}^{-2}\mathbf{V}^{-1}) = \mathcal{N}(\mathbf{0}, g(\mathbf{X}^T\mathbf{X})^{-1}) \tag{12.79}$$

So the dependence of the prior for $\mathbf{w}$ on $\mathbf{X}$ arises from the fact that $\mathbf{w}$ is derived indirectly by a joint model of $\mathbf{X}$ and $\mathbf{y}$.

The above discussion focussed on regression. (Guo 2009) generalizes CCA to the exponential family, which is more appropriate if $\mathbf{x}_i$ and/or y_i are discrete. Although we can no longer compute the conditional $p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$ in closed form, the model has a similar interpretation to the regression case, namely that we are predicting the response via a latent "bottleneck".

The basic idea of compressing $\mathbf{x}_i$ to predict $\mathbf{y}_i$ can be formulated using information theory. In particular, we might want to find an encoding distribution $p(\mathbf{z}|\mathbf{x})$ such that we minimize

$$\mathbb{I}(X; Z) - \beta \mathbb{I}(X; Y) \tag{12.80}$$

where $\beta \geq 0$ is some parameter controlling the tradeoff between compression and predictive accuracy. This is known as the **information bottleneck** (Tishby et al. 1999). Often Z is taken to be discrete, as in clustering. However, in the Gaussian case, IB is closely related to CCA (Chechik et al. 2005).

We can easily generalize CCA to the case where y_i is a vector of responses to be predicted, as in multi-label classification. (Ma et al. 2008; Williamson and Ghahramani 2008) used this model to perform collaborative filtering, where the goal is to predict $y_{ij} \in \{1, \ldots, 5\}$, the rating person i gives to movie j, where the "side information" $\mathbf{x}_i$ takes the form of a list of i's friends. The intuition behind this approach is that knowledge of who your friends are, as well as the ratings of all other users, should help predict which movies you will like. In general, any setting where the tasks are correlated could benefit from CCA. Once we adopt a probabilistic view, various extensions are straightforward. For example, we can easily generalize to the semi-supervised case, where we do not observe $\mathbf{y}_i$ for all i (Yu et al. 2006).

12.5.1.1 Discriminative supervised PCA

One problem with this model is that it puts as much weight on predicting the inputs $\mathbf{x}_i$ as the outputs $\mathbf{y}_i$. This can be partially alleviated by using a weighted objective of the following form (Rish et al. 2008):

$$\ell(\boldsymbol{\theta}) = \prod_i p(\mathbf{y}_i|\boldsymbol{\eta}_{iy})^{\alpha_y} p(\mathbf{x}_i|\boldsymbol{\eta}_{ix})^{\alpha_x} \tag{12.81}$$

where the α_m control the relative importance of the data sources, and $\boldsymbol{\eta}_{im} = \mathbf{W}_m \mathbf{z}_i$. For Gaussian data, we can see that α_m just controls the noise variance:

$$\ell(\boldsymbol{\theta}) \propto \prod_i \exp(-\frac{1}{2}\alpha_x ||\mathbf{x}_i^T - \boldsymbol{\eta}_{ix}||^2) \exp(-\frac{1}{2}\alpha_y ||\mathbf{y}_i^T - \boldsymbol{\eta}_{iy}||^2) \tag{12.82}$$

This interpretation holds more generally for the exponential family. Note, however, that it is hard to estimate the α_m parameters, because changing them changes the normalization constant of the likelihood. We give an alternative approach to weighting $\mathbf{y}$ more heavily below.

12.5.2 Partial least squares

The technique of **partial least squares** (**PLS**) (Gustafsson 2001; Sun et al. 2009) is an asymmetric or more "discriminative" form of supervised PCA. The key idea is to allow some of the (co)variance in the input features to be explained by its own subspace, $\mathbf{z}_i^x$, and to let the rest of the subspace, $\mathbf{z}_i^s$, be shared between input and output. The model has the form

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i^s|\mathbf{0}, \mathbf{I}_{L_s})\mathcal{N}(\mathbf{z}_i^x|\mathbf{0}, \mathbf{I}_{L_x}) \tag{12.83}$$

$$p(\mathbf{y}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{W}_y \mathbf{z}_i^s + \boldsymbol{\mu}_y, \sigma^2 \mathbf{I}_{D_y}) \tag{12.84}$$

$$p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{W}_x \mathbf{z}_i^s + \mathbf{B}_x \mathbf{z}_i^x + \boldsymbol{\mu}_x, \sigma^2 \mathbf{I}_{D_x}) \tag{12.85}$$

See Figure 12.19(b). The corresponding induced distribution on the visible variables has the form

$$p(\mathbf{v}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{v}_i|\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I})d\mathbf{z}_i = \mathcal{N}(\mathbf{v}_i|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}) \tag{12.86}$$

where $\mathbf{v}_i = (\mathbf{x}_i; \mathbf{y}_i)$, $\boldsymbol{\mu} = (\boldsymbol{\mu}_y; \boldsymbol{\mu}_x)$ and

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_y & \mathbf{0} \\ \mathbf{W}_x & \mathbf{B}_x \end{pmatrix} \tag{12.87}$$

$$\mathbf{W}\mathbf{W}^T = \begin{pmatrix} \mathbf{W}_y\mathbf{W}_y^T & \mathbf{W}_x\mathbf{W}_x^T \\ \mathbf{W}_x\mathbf{W}_x^T & \mathbf{W}_x\mathbf{W}_x^T + \mathbf{B}_x\mathbf{B}_x^T \end{pmatrix} \tag{12.88}$$

We should choose L large enough so that the shared subspace does not capture covariate-specific variation.

This model can be easily generalized to discrete data using the exponential family (Virtanen 2010).

12.5.3 Canonical correlation analysis

Canonical correlation analysis or **CCA** is like a symmetric unsupervised version of PLS: it allows each view to have its own "private" subspace, but there is also a shared subspace. If we have two observed variables, $\mathbf{x}_i$ and $\mathbf{y}_i$, then we have three latent variables, $\mathbf{z}_i^s \in \mathbb{R}^{L_0}$ which is shared, $\mathbf{z}_i^x \in \mathbb{R}^{L_x}$ and $\mathbf{z}_i^y \in \mathbb{R}^{L_y}$ which are private. We can write the model as follows (Bach and Jordan 2005):

$$p(\mathbf{z}_i) = \mathcal{N}(\mathbf{z}_i^s|\mathbf{0}, \mathbf{I}_{L_s})\mathcal{N}(\mathbf{z}_i^x|\mathbf{0}, \mathbf{I}_{L_x})\mathcal{N}(\mathbf{z}_i^y|\mathbf{0}, \mathbf{I}_{L_y}) \tag{12.89}$$

$$p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{x}_i|\mathbf{B}_x\mathbf{z}_i^x + \mathbf{W}_x\mathbf{z}_i^s + \boldsymbol{\mu}_x, \sigma^2\mathbf{I}_{D_x}) \tag{12.90}$$

$$p(\mathbf{y}_i|\mathbf{z}_i) = \mathcal{N}(\mathbf{y}_i|\mathbf{B}_y\mathbf{z}_i^y + \mathbf{W}_y\mathbf{z}_i^s + \boldsymbol{\mu}_y, \sigma^2\mathbf{I}_{D_y}) \tag{12.91}$$

See Figure 12.19(c). The corresponding observed joint distribution has the form

$$p(\mathbf{v}_i|\boldsymbol{\theta}) = \int \mathcal{N}(\mathbf{v}_i|\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I})d\mathbf{z}_i = \mathcal{N}(\mathbf{v}_i|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}_D) \tag{12.92}$$

where

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_x & \mathbf{B}_x & \mathbf{0} \\ \mathbf{W}_y & \mathbf{0} & \mathbf{B}_y \end{pmatrix} \tag{12.93}$$

$$\mathbf{W}\mathbf{W}^T = \begin{pmatrix} \mathbf{W}_x\mathbf{W}_x^T + \mathbf{B}_x\mathbf{B}_x^T & \mathbf{W}_x\mathbf{W}_y^T \\ \mathbf{W}_y\mathbf{W}_y^T & \mathbf{W}_y\mathbf{W}_y^T + \mathbf{B}_y\mathbf{B}_y^T \end{pmatrix} \tag{12.94}$$

One can compute the MLE for this model using EM. (Bach and Jordan 2005) show that the resulting MLE is equivalent (up to rotation and scaling) to the classical, non-probabilistic view. However, the advantages of the probabilistic view are many: we can trivially generalize to $M > 2$ observed variables; we can create mixtures of CCA (Viinikanoja et al. 2010); we can create sparse versions of CCA using ARD (Archambeau and Bach 2008); we can generalize to the exponential family (Klami et al. 2010); we can perform Bayesian inference of the parameters (Wang 2007; Klami and Kaski 2008); we can handle non-parametric sparsity-promoting priors for $\mathbf{W}$ and $\mathbf{B}$ (Rai and Daume 2009); and so on.

12.6 Independent Component Analysis (ICA)

Consider the following situation. You are in a crowded room and many people are speaking. Your ears essentially act as two microphones, which are listening to a linear combination of the different speech signals in the room. Your goal is to deconvolve the mixed signals into their constituent parts. This is known as the **cocktail party problem**, and is an example of **blind signal separation** (BSS), or **blind source separation**, where "blind" means we know "nothing" about the source of the signals. Besides the obvious applications to acoustic signal processing, this problem also arises when analysing EEG and MEG signals, financial data, and any other dataset (not necessarily temporal) where latent sources or factors get mixed together in a linear way.

We can formalize the problem as follows. Let $\mathbf{x}_t \in \mathbb{R}^D$ be the observed signal at the sensors at "time" t, and $\mathbf{z}_t \in \mathbb{R}^L$ be the vector of source signals. We assume that

$$\mathbf{x}_t = \mathbf{W}\mathbf{z}_t + \boldsymbol{\epsilon}_t \tag{12.95}$$

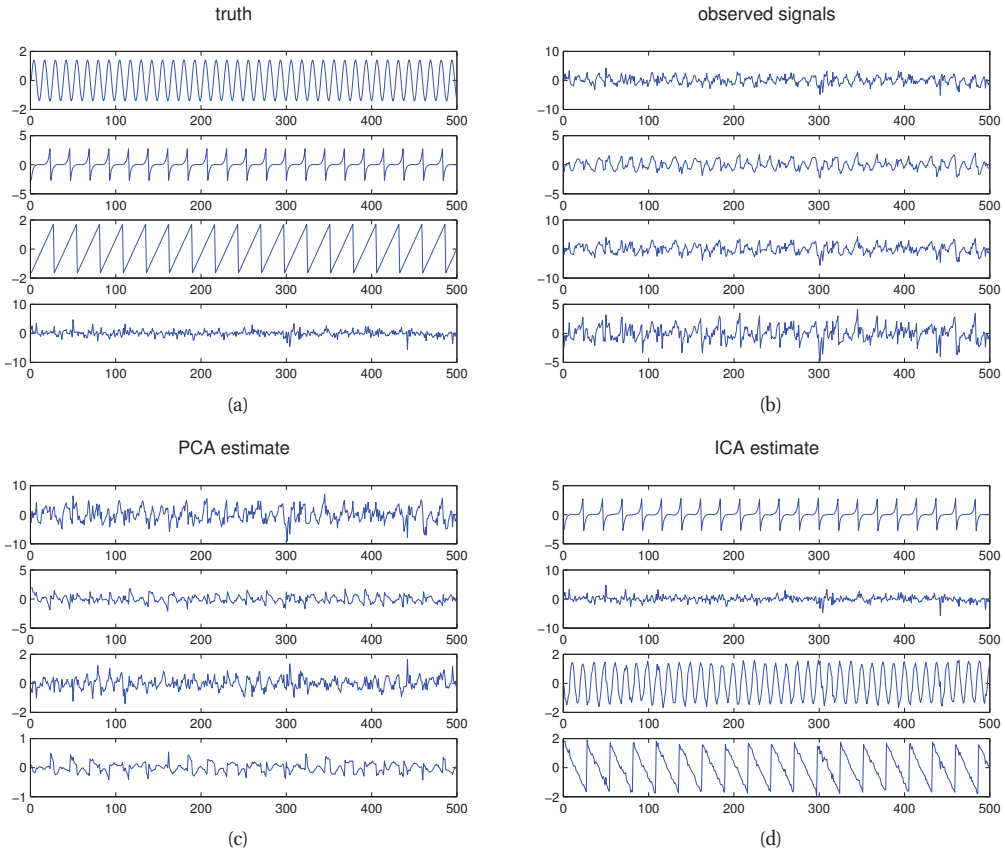


Figure 12.20 Illustration of ICA applied to 500 iid samples of a 4d source signal. (a) Latent signals. (b) Observations. (c) PCA estimate. (d) ICA estimate. Figure generated by `icaDemo`, written by Aapo Hyvarinen.

where $\mathbf{W}$ is an $D \times L$ matrix, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$. In this section, we treat each time point as an independent observation, i.e., we do not model temporal correlation (so we could replace the t index with i, but we stick with t to be consistent with much of the ICA literature). The goal is to infer the source signals, $p(\mathbf{z}_t|\mathbf{x}_t, \boldsymbol{\theta})$, as illustrated in Figure 12.20. In this context, $\mathbf{W}$ is called the **mixing matrix**. If $L = D$ (number of sources = number of sensors), it will be a square matrix. Often we will assume the noise level, $|\boldsymbol{\Psi}|$, is zero, for simplicity.

So far, the model is identical to factor analysis (or PCA if there is no noise, except we don't in general require orthogonality of $\mathbf{W}$). However, we will use a different prior for $p(\mathbf{z}_t)$. In PCA, we assume each source is independent, and has a Gaussian distribution

$$p(\mathbf{z}_t) = \prod_{j=1}^{L} \mathcal{N}(z_{tj}|0, 1) \tag{12.96}$$

We will now relax this Gaussian assumption and let the source distributions be any *non-Gaussian*

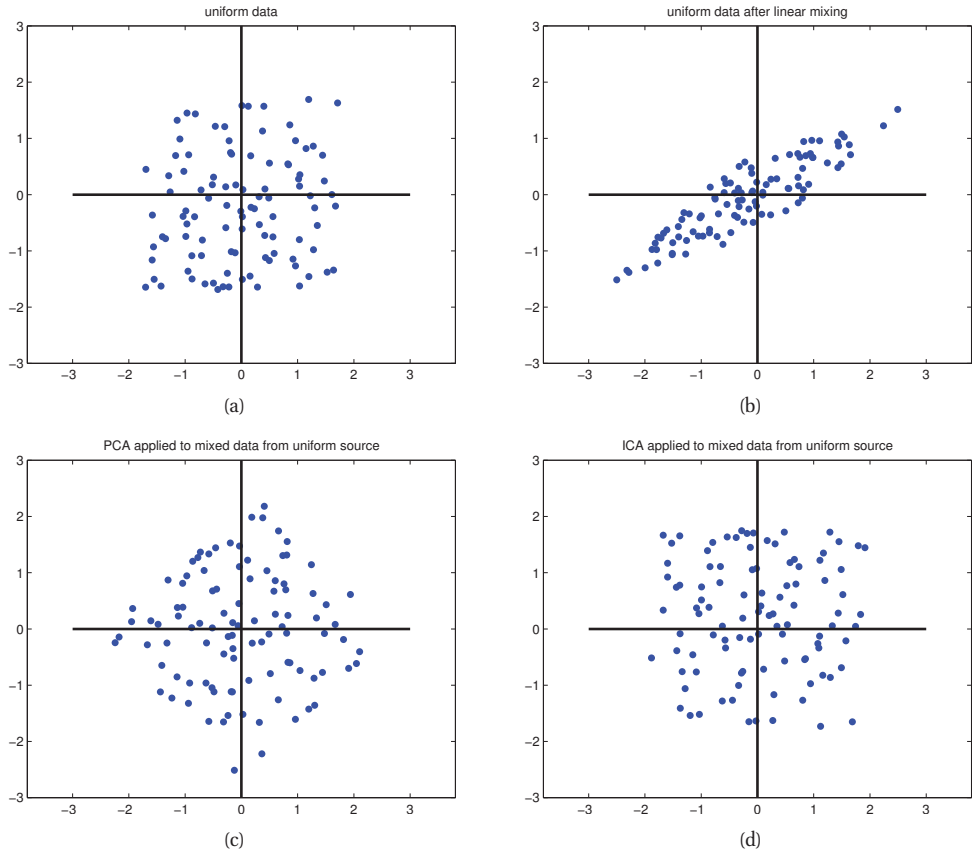


Figure 12.21 Illustration of ICA and PCA applied to 100 iid samples of a 2d source signal with a uniform distribution. (a) Latent signals. (b) Observations. (c) PCA estimate. (d) ICA estimate. Figure generated by `icaDemoUniform`, written by Aapo Hyvarinen.

distribution

$$p(\mathbf{z}_t) = \prod_{j=1}^{L} p_j(z_{tj}) \tag{12.97}$$

Without loss of generality, we can constrain the variance of the source distributions to be 1, because any other variance can be modelled by scaling the rows of $\mathbf{W}$ appropriately. The resulting model is known as **independent component analysis** or **ICA**.

The reason the Gaussian distribution is disallowed as a source prior in ICA is that it does not permit unique recovery of the sources, as illustrated in Figure 12.20(c). This is because the PCA likelihood is invariant to any orthogonal transformation of the sources $\mathbf{z}_t$ and mixing matrix $\mathbf{W}$. PCA can recover the best linear subspace in which the signals lie, but cannot uniquely recover the signals themselves.

To illustrate this, suppose we have two independent sources with uniform distributions, as shown in Figure 12.21(a). Now suppose we have the following mixing matrix

$$\mathbf{W} = \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \tag{12.98}$$

Then we observe the data shown in Figure 12.21(b) (assuming no noise). If we apply PCA followed by scaling to this, we get the result in Figure 12.21(c). This corresponds to a whitening of the data. To uniquely recover the sources, we need to perform an additional rotation. The trouble is, there is no information in the symmetric Gaussian posterior to tell us which angle to rotate by. In a sense, PCA solves "half" of the problem, since it identifies the linear subspace; all that ICA has to do is then to identify the appropriate rotation. (Hence we see that ICA is not that different from methods such as varimax, which seek good rotations of the latent factors to enhance interpretability.)

Figure 12.21(d) shows that ICA can recover the source, up to a permutation of the indices and possible sign change. ICA requires that $\mathbf{W}$ is square and hence invertible. In the non-square case (e.g., where we have more sources than sensors), we cannot uniquely recover the true signal, but we can compute the posterior $p(\mathbf{z}_t|\mathbf{x}_t, \hat{\mathbf{W}})$, which represents our beliefs about the source. In both cases, we need to estimate $\mathbf{W}$ as well as the source distributions p_j. We discuss how to do this below.

12.6.1 Maximum likelihood estimation

In this section, we discuss ways to estimate square mixing matrices $\mathbf{W}$ for the noise-free ICA model. As usual, we will assume that the observations have been centered; hence we can also assume $\mathbf{z}$ is zero-mean. In addition, we assume the observations have been whitened, which can be done with PCA.

If the data is centered and whitened, we have $\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] = \mathbf{I}$. But in the noise free case, we also have

$$\text{cov}\left[\mathbf{x}\right] = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] = \mathbf{W}\mathbb{E}\left[\mathbf{z}\mathbf{z}^T\right]\mathbf{W}^T = \mathbf{W}\mathbf{W}^T \tag{12.99}$$

Hence we see that $\mathbf{W}$ must be orthogonal. This reduces the number of parameters we have to estimate from D^2 to $D(D-1)/2$. It will also simplify the math and the algorithms.

Let $\mathbf{V} = \mathbf{W}^{-1}$; these are often called the **recognition weights**, as opposed to $\mathbf{W}$, which are the **generative weights**.[4]

Since $\mathbf{x} = \mathbf{W}\mathbf{z}$, we have, from Equation 2.89,

$$p_x(\mathbf{W}\mathbf{z}_t) = p_z(\mathbf{z}_t)|\det(\mathbf{W}^{-1})| = p_z(\mathbf{V}\mathbf{x}_t)|\det(\mathbf{V})| \tag{12.100}$$

Hence we can write the log-likelihood, assuming T iid samples, as follows:

$$\frac{1}{T}\log p(\mathcal{D}|\mathbf{V}) = \log|\det(\mathbf{V})| + \frac{1}{T}\sum_{j=1}^{L}\sum_{t=1}^{T}\log p_j(\mathbf{v}_j^T\mathbf{x}_t) \tag{12.101}$$

4. In the literature, it is common to denote the generative weights by $\mathbf{A}$ and the recognition weights by $\mathbf{W}$, but we are trying to be consistent with the notation used earlier in this chapter.

where $\mathbf{v}_j$ is the j'th row of $\mathbf{V}$. Since we are constraining $\mathbf{V}$ to be orthogonal, the first term is a constant, so we can drop it. We can also replace the average over the data with an expectation operator to get the following objective

$$\mathrm{NLL}(\mathbf{V}) = \sum_{j=1}^{L} \mathbb{E}\left[G_j(z_j)\right] \tag{12.102}$$

where $z_j = \mathbf{v}_j^T \mathbf{x}$ and $G_j(z) \triangleq -\log p_j(z)$. We want to minimize this subject to the constraint that the rows of $\mathbf{V}$ are orthogonal. We also want them to be unit norm, since this ensures that the variance of the factors is unity (since, with whitened data, $\mathbb{E}\left[\mathbf{v}_j^T \mathbf{x}\right] = ||\mathbf{v}_j||^2$), which is necessary to fix the scale of the weights. In otherwords, $\mathbf{V}$ should be an orthonormal matrix.

It is straightforward to derive a gradient descent algorithm to fit this model; however, it is rather slow. One can also derive a faster algorithm that follows the natural gradient; see e.g., (MacKay 2003, ch 34) for details. A popular alternative is to use an approximate Newton method, which we discuss in Section 12.6.2. Another approach is to use EM, which we discuss in Section 12.6.3.

12.6.2 The FastICA algorithm

We now describe the **fast ICA** algorithm, based on (Hyvarinen and Oja 2000), which we will show is an approximate Newton method for fitting ICA models.

For simplicity of presentation, we initially assume there is only one latent factor. In addition, we initially assume all source distributions are known and are the same, so we can just write $G(z) = -\log p(z)$. Let $g(z) = \frac{d}{dz} G(z)$. The constrained objective, and its gradient and Hessian, are given by

$$f(\mathbf{v}) = \mathbb{E}\left[G(\mathbf{v}^T \mathbf{x})\right] + \lambda(1 - \mathbf{v}^T \mathbf{v}) \tag{12.103}$$

$$\nabla f(\mathbf{v}) = \mathbb{E}\left[\mathbf{x} g(\mathbf{v}^T \mathbf{x})\right] - \beta \mathbf{v} \tag{12.104}$$

$$\mathbf{H}(\mathbf{v}) = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T g'(\mathbf{v}^T \mathbf{x})\right] - \beta \mathbf{I} \tag{12.105}$$

where $\beta = 2\lambda$ is a Lagrange multiplier. Let us make the approximation

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^T g'(\mathbf{v}^T \mathbf{x})\right] \approx \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] \mathbb{E}\left[g'(\mathbf{v}^T \mathbf{x})\right] = \mathbb{E}\left[g'(\mathbf{v}^T \mathbf{x})\right] \tag{12.106}$$

This makes the Hessian very easy to invert, giving rise to the following Newton update:

$$\mathbf{v}^* \triangleq \mathbf{v} - \frac{\mathbb{E}\left[\mathbf{x} g(\mathbf{v}^T \mathbf{x})\right] - \beta \mathbf{v}}{\mathbb{E}\left[g'(\mathbf{v}^T \mathbf{x})\right] - \beta} \tag{12.107}$$

One can rewrite this in the following way

$$\mathbf{v}^* \triangleq \mathbb{E}\left[\mathbf{x} g(\mathbf{v}^T \mathbf{x})\right] - \mathbb{E}\left[g'(\mathbf{v}^T \mathbf{x})\right] \mathbf{v} \tag{12.108}$$

(In practice, the expectations can be replaced by Monte Carlo estimates from the training set, which gives an efficient online learning algorithm.) After performing this update, one should project back onto the constraint surface using

$$\mathbf{v}^{new} \triangleq \frac{\mathbf{v}^*}{||\mathbf{v}^*||} \tag{12.109}$$

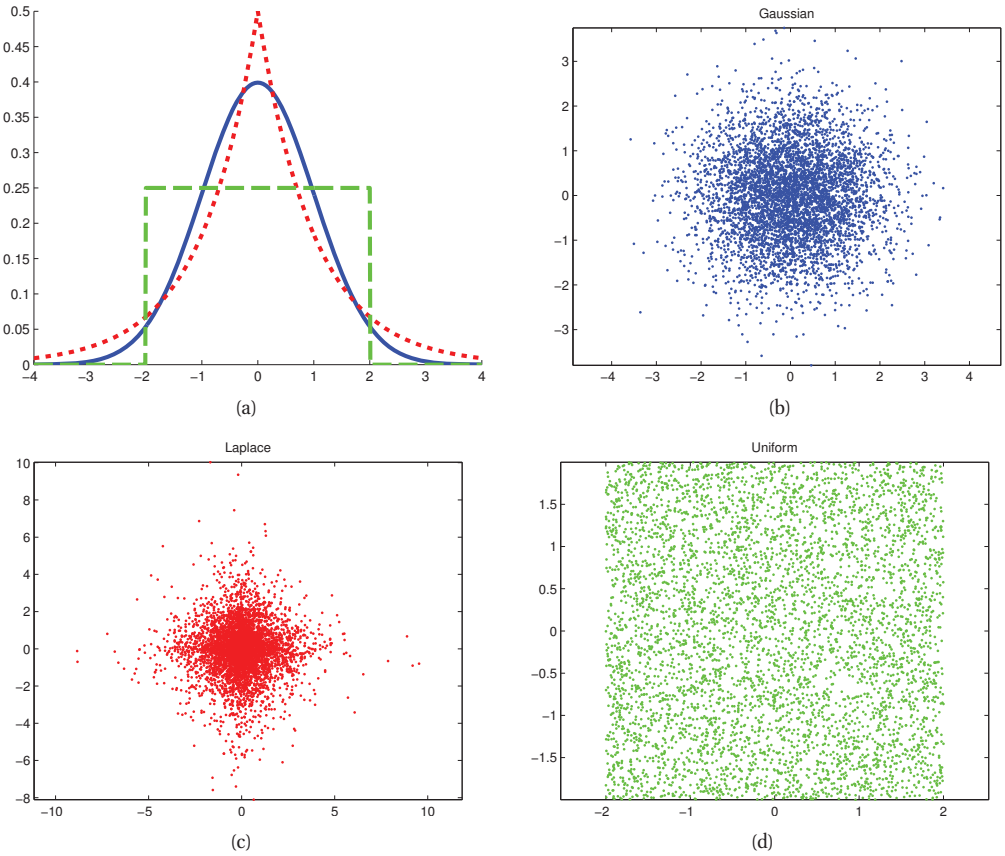


Figure 12.22 Illustration of Gaussian, sub-Gaussian (uniform) and super-Gaussian (Laplace) distributions in 1d and 2d. Figure generated by `subSuperGaussPlot`, written by Kevin Swersky.

One iterates this algorithm until convergence. (Due to the sign ambiguity of $\mathbf{v}$, the values of $\mathbf{v}$ may not converge, but the direction defined by this vector should converge, so one can assess convergence by monitoring $|\mathbf{v}^T\mathbf{v}^{new}|$, which should approach 1.)

Since the objective is not convex, there are multiple local optima. We can use this fact to learn multiple different weight vectors or **features**. We can either learn the features sequentially and then project out the part of $\mathbf{v}_j$ that lies in the subspace defined by earlier features, or we can learn them in parallel, and orthogonalize $\mathbf{V}$ in parallel. This latter approach is usually preferred, since, unlike PCA, the features are not ordered in any way. So the first feature is not "more important" than the second, and hence it is better to treat them symmetrically.

12.6.2.1 Modeling the source densities

So far, we have assumed that $G(z) = -\log p(z)$ is known. What kinds of models might be reasonable as signal priors? We know that using Gaussians (which correspond to quadratic functions for G) won't work. So we want some kind of non-Gaussian distribution. In general, there are several kinds of non-Gaussian distributions, such as the following:

- **Super-Gaussian distributions** These are distributions which have a big spike at the mean, and hence (in order to ensure unit variance) have heavy tails. The Laplace distribution is a classic example. See Figure 12.22. Formally, we say a distribution is **super-Gaussian** or **leptokurtic** ("lepto" coming from the Greek for "thin") if $\mathrm{kurt}(z) > 0$, where $\mathrm{kurt}(z)$ is the **kurtosis** of the distribution, defined by

$$\mathrm{kurt}(z) \triangleq \frac{\mu_4}{\sigma^4} - 3 \tag{12.110}$$

  where σ is the standard deviation, and μ_k is the k'th **central moment**, or moment about the mean:

$$\mu_k \triangleq \mathbb{E}\left[(X - \mathbb{E}[X])^k\right] \tag{12.111}$$

  (So $\mu_1 = \mu$ is the mean, and $\mu_2 = \sigma^2$ is the variance.) It is conventional to subtract 3 in the definition of kurtosis to make the kurtosis of a Gaussian variable equal to zero.
- **Sub-Gaussian distributions** A **sub-Gaussian** or **platykurtic** ("platy" coming from the Greek for "broad") distribution has negative kurtosis. These are distributions which are much flatter than a Gaussian. The uniform distribution is a classic example. See Figure 12.22.
- **Skewed distributions** Another way to "be non-Gaussian" is to be asymmetric. One measure of this is **skewness**, defined by

$$\mathrm{skew}(z) \triangleq \frac{\mu_3}{\sigma^3} \tag{12.112}$$

  An example of a (right) skewed distribution is the gamma distribution (see Figure 2.9).

When one looks at the empirical distribution of many natural signals, such as images and speech, when passed through certain linear filters, they tend to be very super-Gaussian. This result holds both for the kind of linear filters found in certain parts of the brain, such as the simple cells found in the primary visual cortex, as well as for the kinds of linear filters used in signal processing, such as wavelet transforms. One obvious choice for modeling natural signals with ICA is therefore the Laplace distribution. For mean zero and variance 1, this has a log pdf given by

$$\log p(z) = -\sqrt{2}|z| - \log(\sqrt{2}) \tag{12.113}$$

Since the Laplace prior is not differentiable at the origin, it is more common to use other, smoother super-Gaussian distributions. One example is the logistic distribution. The corresponding log pdf, for the case where the mean is zero and the variance is 1 (so $\mu = 0$ and $s = \frac{\sqrt{3}}{\pi}$), is given by the following:

$$\log p(z) = -2\log\cosh\left(\frac{\pi}{2\sqrt{3}}z\right) - \log\frac{4\sqrt{3}}{\pi} \tag{12.114}$$

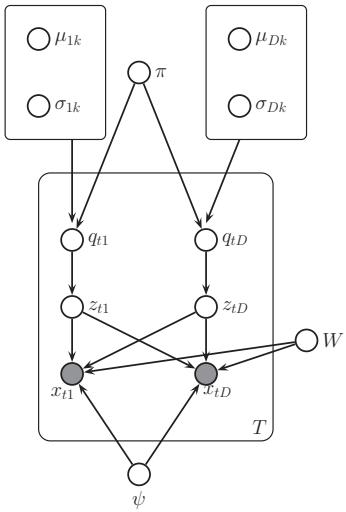


Figure 12.23 Modeling the source distributions using a mixture of univariate Gaussians (the independent factor analysis model of (Moulines et al. 1997; Attias 1999)).

Various ways of estimating $G(Z) = -\log p(z)$ are discussed in the seminal paper (Pham and Garrat 1997). However, when fitting ICA by maximum likelihood, it is not critical that the exact shape of the source distribution be known (although it is important to know whether it is sub or super Gaussian). Consequently, it is common to just use $G(z) = \sqrt{z}$ or $G(z) = \log\cosh(z)$ instead of the more complex expressions above.

12.6.3 Using EM

An alternative to assuming a particular form for $G(z)$, or equivalently for $p(z)$, is to use a flexible non-parametric density estimator, such as a mixture of (uni-variate) Gaussians:

$$p(q_j = k) = \pi_k \tag{12.115}$$

$$p(z_j|q_j = k) = \mathcal{N}(\mu_{j,k}, \sigma^2_{j,k}) \tag{12.116}$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z}, \mathbf{\Psi}) \tag{12.117}$$

This approach was proposed in (Moulines et al. 1997; Attias 1999), and the corresponding graphical model is shown in Figure 12.23.

It is possible to derive an exact EM algorithm for this model. The key observation is that it is possible to compute $\mathbb{E}[\mathbf{z}_t|\mathbf{x}_t, \boldsymbol{\theta}]$ exactly by summing over all K^L combinations of the $\mathbf{q}_t$ variables, where K is the number of mixture components per source. (If this is too expensive, one can use a variational mean field approximation (Attias 1999).) We can then estimate all the source distributions in parallel by fitting a standard GMM to $\mathbb{E}[\mathbf{z}_t]$. When the source GMMs are

known, we can compute the marginals $p_j(z_j)$ very easily, using

$$p_j(z_j) = \sum_{k=1}^{K} \pi_{j,k} \mathcal{N}(z_j|\mu_{j,k}, \sigma_{j,k}^2) \tag{12.118}$$

Given the p_j's, we can then use an ICA algorithm to estimate $\mathbf{W}$. Of course, these steps should be interleaved. The details can be found in (Attias 1999).

12.6.4 Other estimation principles *

It is quite common to estimate the parameters of ICA models using methods that seem different to maximum likelihood. We will review some of these methods below, because they give additional insight into ICA. However, we will also see that these methods in fact are equivalent to maximum likelihood after all. Our presentation is based on (Hyvarinen and Oja 2000).

12.6.4.1 Maximizing non-Gaussianity

An early approach to ICA was to find a matrix $\mathbf{V}$ such that the distribution $\mathbf{z} = \mathbf{V}\mathbf{x}$ is as far from Gaussian as possible. (There is a related approach in statistics called **projection pursuit**.) One measure of non-Gaussianity is kurtosis, but this can be sensitive to outliers. Another measure is the **negentropy**, defined as

$$\text{negentropy}(z) \triangleq \mathbb{H}\left(\mathcal{N}(\mu, \sigma^2)\right) - \mathbb{H}(z) \tag{12.119}$$

where $\mu = \mathbb{E}[z]$ and $\sigma^2 = \text{var}[z]$. Since the Gaussian is the maximum entropy distribution, this measure is always non-negative and becomes large for distributions that are highly non-Gaussian.

We can define our objective as maximizing

$$J(\mathbf{V}) = \sum_j \text{negentropy}(z_j) = \sum_j \mathbb{H}\left(\mathcal{N}(\mu_j, \sigma_j^2)\right) - \mathbb{H}(z_j) \tag{12.120}$$

where $\mathbf{z} = \mathbf{V}\mathbf{x}$. If we fix $\mathbf{V}$ to be orthogonal, and if we whiten the data, the covariance of $\mathbf{z}$ will be $\mathbf{I}$ independently of $\mathbf{V}$, so the first term is a constant. Hence

$$J(\mathbf{V}) = \sum_j -\mathbb{H}(z_j) + \text{const} = \sum_j \mathbb{E}[\log p(z_j)] + \text{const} \tag{12.121}$$

which we see is equal (up to a sign change, and irrelevant constants) to the log-likelihood in Equation 12.102.

12.6.4.2 Minimizing mutual information

One measure of dependence of a set of random variables is the **multi-information**:

$$I(\mathbf{z}) \triangleq \mathbb{KL}\left(p(\mathbf{z})||\prod_j p(z_j)\right) = \sum_j \mathbb{H}(z_j) - \mathbb{H}(\mathbf{z}) \tag{12.122}$$

We would like to minimize this, since we are trying to find independent components. Put another way, we want the best possible factored approximation to the joint distribution.

Now since $\mathbf{z} = \mathbf{V}\mathbf{x}$, we have

$$I(\mathbf{z}) = \sum_j \mathbb{H}(z_j) - \mathbb{H}(\mathbf{V}\mathbf{x}) \tag{12.123}$$

If we constrain $\mathbf{V}$ to be orthogonal, we can drop the last term, since then $\mathbb{H}(\mathbf{V}\mathbf{x}) = \mathbb{H}(\mathbf{x})$ (since multiplying by $\mathbf{V}$ does not change the shape of the distribution), and $\mathbb{H}(\mathbf{x})$ is a constant which is is solely determined by the empirical distribution. Hence we have $I(\mathbf{z}) = \sum_j \mathbb{H}(z_j)$. Minimizing this is equivalent to maximizing the negentropy, which is equivalent to maximum likelihood.

12.6.4.3 Maximizing mutual information (infomax)

Instead of trying to minimize the mutual information between the components of $\mathbf{z}$, let us imagine a neural network where $\mathbf{x}$ is the input and $y_j = \phi(\mathbf{v}_j^T \mathbf{x}) + \epsilon$ is the noisy output, where ϕ is some nonlinear scalar function, and $\epsilon \sim \mathcal{N}(0, 1)$. It seems reasonable to try to maximize the information flow through this system, a principle known as **infomax**. (Bell and Sejnowski 1995). That is, we want to maximize the mutual information between $\mathbf{y}$ (the internal neural representation) and $\mathbf{x}$ (the observed input signal). We have $\mathbb{I}(\mathbf{x}; \mathbf{y}) = \mathbb{H}(\mathbf{y}) - \mathbb{H}(\mathbf{y}|\mathbf{x})$, where the latter term is constant if we assume the noise has constant variance. One can show that we can approximate the former term as follows

$$\mathbb{H}(\mathbf{y}) = \sum_{j=1}^{L} \mathbb{E}\left[\log \phi'(\mathbf{v}_j^T \mathbf{x})\right] + \log|\det(\mathbf{V})| \tag{12.124}$$

where, as usual, we can drop the last term if $\mathbf{V}$ is orthogonal. If we define $\phi(z)$ to be a cdf, then $\phi'(z)$ is its pdf, and the above expression is equivalent to the log likelihood. In particular, if we use a logistic nonlinearity, $\phi(z) = \mathrm{sigm}(z)$, then the corresponding pdf is the logistic distribution, and $\log \phi'(z) = \log\cosh(z)$ (ignoring irrelevant constants). Thus we see that infomax is equivalent to maximum likelihood.

Exercises

Exercise 12.1 M step for FA

For the FA model, show that the MLE in the M step for $\mathbf{W}$ is given by Equation 12.23.

Exercise 12.2 MAP estimation for the FA model

Derive the M step for the FA model using conjugate priors for the parameters.

Exercise 12.3 Heuristic for assessing applicability of PCA

(Source: (Press 2005, Q9.8).). Let the empirical covariance matrix Σ have eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$. Explain why the variance of the evalues, $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(\lambda_i - \overline{\lambda})^2$ is a good measure of whether or not PCA would be useful for analysing the data (the higher the value of σ^2 the more useful PCA).

Exercise 12.4 Deriving the second principal component

a. Let

$$J(\mathbf{v}_2, \mathbf{z}_2) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - z_{i1}\mathbf{v}_1 - z_{i2}\mathbf{v}_2)^T(\mathbf{x}_i - z_{i1}\mathbf{v}_1 - z_{i2}\mathbf{v}_2) \tag{12.125}$$

Show that $\frac{\partial J}{\partial \mathbf{z}_2} = 0$ yields $z_{i2} = \mathbf{v}_2^T\mathbf{x}_i$.

b. Show that the value of $\mathbf{v}_2$ that minimizes

$$\tilde{J}(\mathbf{v}_2) = -\mathbf{v}_2^T\mathbf{C}\mathbf{v}_2 + \lambda_2(\mathbf{v}_2^T\mathbf{v}_2 - 1) + \lambda_{12}(\mathbf{v}_2^T\mathbf{v}_1 - 0) \tag{12.126}$$

is given by the eigenvector of $\mathbf{C}$ with the second largest eigenvalue. Hint: recall that $\mathbf{C}\mathbf{v}_1 = \lambda_1\mathbf{v}_1$ and $\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$.

Exercise 12.5 Deriving the residual error for PCA

a. Prove that

$$||\mathbf{x}_i - \sum_{j=1}^{K} z_{ij}\mathbf{v}_j||^2 = \mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{K}\mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j \tag{12.127}$$

Hint: first consider the case $K = 2$. Use the fact that $\mathbf{v}_j^T\mathbf{v}_j = 1$ and $\mathbf{v}_j^T\mathbf{v}_k = 0$ for $k \neq j$. Also, recall $z_{ij} = \mathbf{x}_i^T\mathbf{v}_j$.

b. Now show that

$$J_K \triangleq \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{K}\mathbf{v}_j^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{v}_j\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^T\mathbf{x}_i - \sum_{j=1}^{K}\lambda_j \tag{12.128}$$

Hint: recall $\mathbf{v}_j^T\mathbf{C}\mathbf{v}_j = \lambda_j\mathbf{v}_j^T\mathbf{v}_j = \lambda_j$.

c. If $K = d$ there is no truncation, so $J_d = 0$. Use this to show that the error from only using $K < d$ terms is given by

$$J_K = \sum_{j=K+1}^{d}\lambda_j \tag{12.129}$$

Hint: partition the sum $\sum_{j=1}^{d}\lambda_j$ into $\sum_{j=1}^{K}\lambda_j$ and $\sum_{j=K+1}^{d}\lambda_j$.

Exercise 12.6 Derivation of Fisher's linear discriminant

Show that the maximum of $J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}$ is given by $\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$ where $\lambda = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}$. Hint: recall that the derivative of a ratio of two scalars is given by $\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{f'g - fg'}{g^2}$, where $f' = \frac{d}{dx}f(x)$ and $g' = \frac{d}{dx}g(x)$. Also, recall that $\frac{d}{d\mathbf{x}}\mathbf{x}^T\mathbf{A}\mathbf{x} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$.

Exercise 12.7 PCA via successive deflation

Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ be the first k eigenvectors with largest eigenvalues of $\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$, i.e., the principal basis vectors. These satisfy

$$\mathbf{v}_j^T\mathbf{v}_k = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases} \tag{12.130}$$

We will construct a method for finding the $\mathbf{v}_j$ sequentially.

As we showed in class, $\mathbf{v}_1$ is the first principal eigenvector of $\mathbf{C}$, and satisfies $\mathbf{C}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$. Now define $\tilde{\mathbf{x}}_i$ as the orthogonal projection of $\mathbf{x}_i$ onto the space orthogonal to $\mathbf{v}_1$:

$$\tilde{\mathbf{x}}_i = \mathbf{P}_{\perp \mathbf{v}_1}\, \mathbf{x}_i = (\mathbf{I} - \mathbf{v}_1 \mathbf{v}_1^T)\mathbf{x}_i \tag{12.131}$$

Define $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1; ...; \tilde{\mathbf{x}}_n]$ as the **deflated matrix** of rank $d-1$, which is obtained by removing from the d dimensional data the component that lies in the direction of the first principal direction:

$$\tilde{\mathbf{X}} = (\mathbf{I} - \mathbf{v}_1 \mathbf{v}_1^T)^T \mathbf{X} = (\mathbf{I} - \mathbf{v}_1 \mathbf{v}_1^T)\mathbf{X} \tag{12.132}$$

a. Using the facts that $\mathbf{X}^T \mathbf{X} \mathbf{v}_1 = n\lambda_1 \mathbf{v}_1$ (and hence $\mathbf{v}_1^T \mathbf{X}^T \mathbf{X} = n\lambda_1 \mathbf{v}_1^T$) and $\mathbf{v}_1^T \mathbf{v}_1 = 1$, show that the covariance of the deflated matrix is given by

$$\tilde{\mathbf{C}} \triangleq \frac{1}{n} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} = \frac{1}{n} \mathbf{X}^T \mathbf{X} - \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T \tag{12.133}$$

b. Let $\mathbf{u}$ be the principal eigenvector of $\tilde{\mathbf{C}}$. Explain why $\mathbf{u} = \mathbf{v}_2$. (You may assume $\mathbf{u}$ is unit norm.)

c. Suppose we have a simple method for finding the leading eigenvector and eigenvalue of a pd matrix, denoted by $[\lambda, \mathbf{u}] = f(\mathbf{C})$. Write some pseudo code for finding the first K principal basis vectors of $\mathbf{X}$ that only uses the special f function and simple vector arithmetic, i.e., your code should not use SVD or the `eig` function. Hint: this should be a simple iterative routine that takes 2–3 lines to write. The input is $\mathbf{C}$, K and the function f, the output should be $\mathbf{v}_j$ and λ_j for $j = 1 : K$. Do not worry about being syntactically correct.

Exercise 12.8 Latent semantic indexing

(Source: de Freitas.). In this exercise, we study a technique called **latent semantic indexing**, which applies SVD to a document by term matrix, to create a low-dimensional embedding of the data that is designed to capture semantic similarity of words.

The file `lsiDocuments.pdf` contains 9 documents on various topics. A list of all the 460 unique words/terms that occur in these documents is in `lsiWords.txt`. A document by term matrix is in `lsiMatrix.txt`.

a. Let X be the transpose of `lsiMatrix`, so each column represents a document. Compute the SVD of X and make an approximation to it $\hat{X}$ using the first 2 singular values/ vectors. Plot the low dimensional representation of the 9 documents in 2D. You should get something like Figure 12.24.

b. Consider finding documents that are about alien abductions. If If you look at `lsiWords.txt`, there are 3 versions of this word, term 23 ("abducted"), term 24 ("abduction") and term 25 ("abductions"). Suppose we want to find documents containing the word "abducted". Documents 2 and 3 contain it, but document 1 does not. However, document 1 is clearly related to this topic. Thus LSI should also find document 1. Create a test document q containing the one word "abducted", and project it into the 2D subspace to make $\hat{q}$. Now compute the cosine similarity between $\hat{q}$ and the low dimensional representation of all the documents. What are the top 3 closest matches?

Exercise 12.9 Imputation in a FA model

Derive an expression for $p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta})$ for a FA model.

Exercise 12.10 Efficiently evaluating the PPCA density

Derive an expression for $p(\mathbf{x}|\hat{\mathbf{W}}, \hat{\sigma}^2)$ for the PPCA model based on plugging in the MLEs and using the matrix inversion lemma.

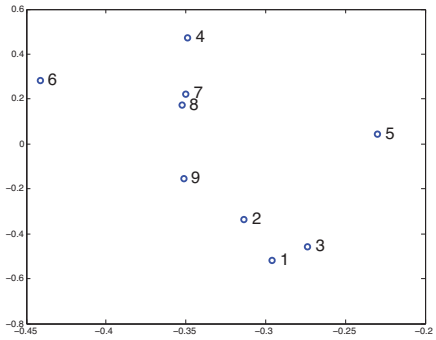


Figure 12.24 Projection of 9 documents into 2 dimensions. Figure generated by `lsiCode`.

Exercise 12.11 PPCA vs FA

(Source: Exercise 14.15 of (Hastie et al. 2009), due to Hinton.). Generate 200 observations from the following model, where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$: $x_{i1} = z_{i1}$, $x_{i2} = z_{i1} + 0.001 z_{i2}$, $x_{i3} = 10 z_{i3}$. Fit a FA and PCA model with 1 latent factor. Hence show that the corresponding weight vector $\mathbf{w}$ aligns with the maximal variance direction (dimension 3) in the PCA case, but with the maximal correlation direction (dimensions 1+2) in the case of FA.