Internal UW/CS Adaptive Computation TR-99-2, Aug 11, 1999.

Factor Analysis Using Batch and Online EM

Brendan J. Frey
http://www.cs.uwaterloo.ca/∼frey
Computer Science, University of Waterloo

1 Encoding Data using Linearly Combined Components

A simple way to encode input patterns is to suppose that each input can be well-approximated by a linear combination of component vectors, where the amplitudes of the vectors are modulated to match the input. For a given training set, the most appropriate set of component vectors will depend on how we expect the modulation levels to behave and how we measure the distance between the input and its approximation. These effects can be captured by a generative probability model that specifies a distribution $p(\mathbf{z})$ over modulation levels $\mathbf{z} = (z_1, \ldots, z_K)^{\mathrm{T}}$ and a distribution $p(\mathbf{x}|\mathbf{z})$ over sensors $\mathbf{x} = (x_1, \ldots, x_N)^{\mathrm{T}}$ given the modulation levels. The linear combination constraint is given by

$$\mathrm{E}[\mathbf{x}|\mathbf{z}] = \Lambda\mathbf{z}, \tag{1}$$

where $\mathrm{E}[\mathbf{x}|\mathbf{z}] = \int_{\mathbf{x}} \mathbf{x} p(\mathbf{x}|\mathbf{z}) d\mathbf{x}$ and each column of Λ is a component vector. Λ is an $N \times K$ matrix with elements λ_{nk}.

Principal component analysis (Jolliffe 1986; Linsker 1988; Oja 1989), independent component analysis (Comon, Jutten and Herault 1991; Bell and Sejnowski 1995; Amari, Cichocki and Yang 1996), and factor analysis (Rubin and Thayer 1982; Everitt 1984) can be viewed as maximum likelihood estimation in models of this type, where we assume that the appropriate modulation levels are independent and the overall distortion is given by the sum of the individual sensor distortions.

The two layer belief network that describes this process is shown in Fig. 1. The distribution over the top layer activities (modulation levels) is

$$p(\mathbf{z}) = \prod_{k=1}^{K} p(z_k), \tag{2}$$

and the distribution over the bottom layer activities (sensors) given the top layer activities is

$$p(\mathbf{x}|\mathbf{z}) = \prod_{n=1}^{N} p(x_n|\mathbf{z}). \tag{3}$$

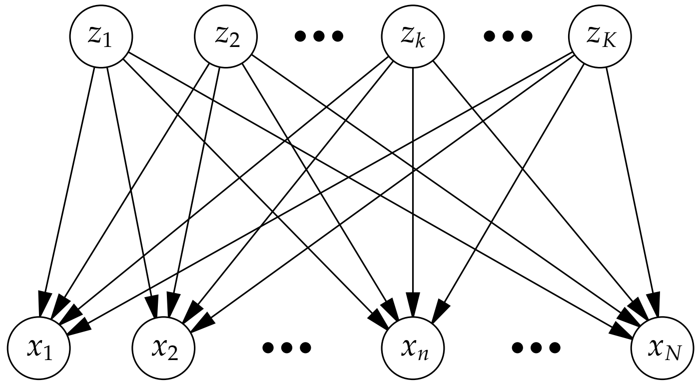

Figure 1: A two layer generative network that describes principal component analysis, independent component analysis and factor analysis.

If we choose probabilities of the form

$$p(z_k) = \mathcal{N}(z_k; 0, 1), \qquad p(x_n|\mathbf{z}) = \mathcal{N}(x_n; \textstyle\sum_{k=1}^{K} \lambda_{nk} z_k, \psi), \tag{4}$$

where the same variance ψ is used for all sensors, then maximum likelihood estimation of Λ and ψ produces components that span the first K principal components of the training data (Roweis and Ghahramani 1999; Tipping and Bishop 1999).

If $p(z_k)$ is a heavy-tailed distribution and *no* distortion is allowed (achieving this generally requires using a complete or over-complete basis, where $K \geq N$), maximum likelihood estimation of Λ is equivalent to independent component analysis (Pearlmutter and Parra 1997; MacKay 1997). For example, we may take

$$p(z_k) \propto 1/(e^{z_k} + e^{-z_k}), \qquad p(x_n|\mathbf{z}) = \delta(x_n - \textstyle\sum_{k=1}^{K} \lambda_{nk} z_k). \tag{5}$$

The advantage of independent component analysis over principal component analysis and factor analysis is the use of heavy-tailed distributions over the modulation levels. These distributions effectively produce sparse representations of the data, since a heavy-tailed distribution places probability mass near zero and at large values (*i.e.*, it has high kurtosis).

1.1 Factor Analysis. If both $p(z_k)$ and $p(x_n|\mathbf{z})$ are Gaussian and each sensor has its own noise variance,

$$p(z_k) = \mathcal{N}(z_k; 0, 1), \qquad p(x_n|\mathbf{z}) = \mathcal{N}(x_n; \textstyle\sum_{k=1}^{K} \lambda_{nk} z_k, \psi_n), \tag{6}$$

then maximum likelihood estimation of Λ and ψ_n, $n = 1, \ldots, N$ performs factor analysis. In factor analysis, the modulation levels are called "factors" and Λ is called the "factor loading matrix".

In contrast to principal component analysis, factor analysis can model a different level of noise on each sensor. In contrast to independent component analysis, the noise model used in factor analysis permits a more compact latent representation, where $K < N$.

The factor analysis model (factor analyzer) can be written

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) = (2\pi)^{-N/2} \exp[-\mathbf{z}^{\mathrm{T}}\mathbf{z}/2] \\
p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}; \Lambda\mathbf{z}, \Psi) = |2\pi\Psi|^{-1/2} \exp[-(\mathbf{x} - \Lambda\mathbf{z})^{\mathrm{T}}\Psi^{-1}(\mathbf{x} - \Lambda\mathbf{z})/2]
\end{aligned} \tag{7}
$$

where Ψ is an $N \times N$ diagonal covariance matrix whose nth diagonal entry is ψ_n and $\mathbf{I}$ is the $K \times K$ identity matrix. The marginal distribution over $\mathbf{x}$ is

$$
\begin{aligned}
p(\mathbf{x}) &= \int_{\mathbf{z}} \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})\mathcal{N}(\mathbf{x}; \Lambda\mathbf{z}, \Psi) d\mathbf{z} = \mathcal{N}(\mathbf{x}; \mathbf{0}, \Lambda\Lambda^{\mathrm{T}} + \Psi) \\
&= |2\pi(\Lambda\Lambda^{\mathrm{T}} + \Psi)|^{-1/2} \exp[-\mathbf{x}^{\mathrm{T}}(\Lambda\Lambda^{\mathrm{T}} + \Psi)^{-1}\mathbf{x}/2]
\end{aligned} \tag{8}
$$

Directly computing the determinant and the inverse covariance matrix takes $O(N^3)$ operations and once they are determined, directly computing $p(\mathbf{x})$ for a given $\mathbf{x}$ takes $O(N^2)$ operations.

Since usually $K << N$, the above procedure can be sped up *very significantly* using linear algebra tricks. $p(\mathbf{x})$ can be rexpressed thus:

$$
\begin{aligned}
p(\mathbf{x}) =& (2\pi)^{-N/2}|\Lambda^{\mathrm{T}}\Psi^{-1}\Lambda + \mathbf{I}|^{-1/2}|\Psi|^{-1/2} \\
&\cdot \exp[-\mathbf{x}^{\mathrm{T}}\big(\Psi^{-1} - \Psi^{-1}\Lambda(\mathbf{I} + \Lambda^{\mathrm{T}}\Psi^{-1}\Lambda)^{-1}\Lambda^{\mathrm{T}}\Psi^{-1}\big)\mathbf{x}/2].
\end{aligned} \tag{9}
$$

Using this expression, computing the determinant and the inverse covariance matrix takes just $O(K^2N)$ operations – this is $(N/K)^2$ times faster than the direct method. Once the determinant and inverse covariance matrix are determined, computing $p(\mathbf{x})$ for a given $\mathbf{x}$ takes just $O(KN)$ operations – this is N/K times faster than the direct method. In practice, it is often a good idea to compute $\log p(\mathbf{x})$ to avoid numerical underflow.

Although the factor analyzer is not able to represent the rich, nonlinear, hierarachical processes that can be learned by nonlinear binary networks (Hinton and Sejnowski 1986; Neal 1992; Hinton et al. 1995; Dayan et al. 1995) and nonlinear real-valued networks (Frey 1997; Hinton and Ghahramani 1997; Frey 1998; Frey and Hinton 1999; Attias 1999), the factor analyzer can be trained using the exact expectation maximization algorithm.

1.2 Nonisotropic Noise. Principal component analysis assumes that the distortion on each sensor is measured in the same way, *i.e.*, the same variance parameter ψ is used for each sensor in (4). In cases where some sensors are significantly noisier than other sensors, principal component analysis tends to extract components that represent the noise instead of components that represent the dependencies between sensors.

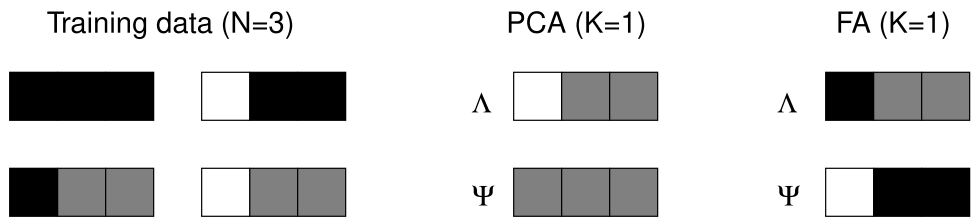

Figure 2: The component Λ and sensor noise levels Ψ found by single-component principal component analysis (PCA) and factor analysis (FA) applied to a small data set (black = 0, white = 1).

For example, Fig. 2 shows 4 training cases for a simple problem with 3 sensors, where the intensity of the gray level in a box corresponds to the numerical value of the sensor (black = 0, white = 1). Clearly, the first sensor is independent of the other sensors, although it has high variance. The second two sensors are highly dependent and in fact always have exactly equal values. When single-component principal component analysis is applied to this data, the component erroneously represents a correlation between all 3 sensors and the component is dominated by the first sensor. Also, the sensor noise level represents significant noise on the second two sensors, even though the fact that the second two sensors are exactly equal in the training set indicates that they aren't corrupted by independent noise.

In some cases, the sensitivity of principal component analysis to independent sensor noise can be corrected by preprocessing the data by hand or by automatically preprocessing the data using assumptions about the sensor noise. For example, researchers interested in classifying frontal images of faces usually try to mask out the hair and background clutter, which often contain higher pixel variances than the facial features (Turk and Pentland 1991; Moghaddam and Pentland 1997).

In contrast to principal component analysis, when single-component factor analysis is applied to the data in Fig. 2 the component accurately represents the dependence between the second two sensors. The sensor noise levels accurately represent the large noise level on the first sensor and the absence of noise on the second two sensors. Whereas principal component analysis finds directions of maximum variance in the input space, factor analysis seeks directions of maximum variance *after* subtracting off an estimate of the independent sensor noise.

Independent component analysis assumes the sensors are noise free (see (5)), so for limited amounts of data the components will come to represent noise as well as structure in the data. In some applications that use large sets of images, the components come to represent useful low-level wavelet-like features. However, for a given input image, the factors will represent noise as well as structure, since the components span the input space and the data is assumed to be noise free. Factor analysis can discount the independent sensor noise before computing the appropriate factors.

1.3 Nonorthogonal Components. The columns of Λ produced by factor analysis may or may not be orthogonal. In fact, Λ is not even identifiable by maximum likelihood factor analysis. If Λ is replaced by $\Lambda\mathbf{R}$, where $\mathbf{R}$ is a $K \times K$ unitary "rotation" matrix, the resulting model leaves the marginal density over $\mathbf{x}$ unchanged. (Try replacing Λ with $\Lambda\mathbf{R}$ in (8) and using the property of a unitary matrix, $\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1}$.) It follows that maximum likelihood factor analysis can only find components Λ up to an arbitrary rotation applied to the *rows* of Λ. In contrast, direct application of singular value decomposition to compute the principal component subspace of some training data produces components that are orthogonal.

Although the nonidentifiability of maximum likelihood factor analysis is often considered to be undesirable, it can also be considered beneficial. Higher levels of processing may perform better using factors produced by nonorthogonal components. For example, in facial expression modeling, components that represent different expressions and different parts of expressions will very likely *not* be orthogonal.

2 Inferring the Factors in a Factor Analyzer

Since the factor analyzer is a Gaussian model, the distribution over the factors given the sensor activities is a Gaussian. The posterior is characterized by the expected value of $\mathbf{z}$ and its covariance matrix, which are obtained by normalizing $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ with respect to $\mathbf{z}$ using linear algebra:

$$\mathrm{E}[\mathbf{z}|\mathbf{x}] = (\Lambda^{\mathrm{T}}\Psi^{-1}\Lambda + \mathbf{I})^{-1}\Lambda^{\mathrm{T}}\Psi^{-1}\mathbf{x}, \tag{10}$$

$$\mathrm{COV}[\mathbf{z}|\mathbf{x}] = (\Lambda^{\mathrm{T}}\Psi^{-1}\Lambda + \mathbf{I})^{-1}. \tag{11}$$

Notice that the covariance matrix does not depend on the *value* of the input, although it does depend on an input being observed. (Recall that $\mathrm{COV}[\mathbf{z}] = \mathbf{I}$.) From these, we can also compute $\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}|\mathbf{x}]$:

$$\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}|\mathbf{x}] = \mathrm{COV}[\mathbf{z}|\mathbf{x}] + E[\mathbf{z}|\mathbf{x}]E[\mathbf{z}|\mathbf{x}]^{\mathrm{T}} \tag{12}$$

3 Factor Analysis Using Batch EM

Given i.i.d. training cases $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$, maximum likelihood factor analysis tries to find the set of parameters (Λ and Ψ) that maximizes

$$\mathcal{L} = \log p(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}) = \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)}). \tag{13}$$

Substituting $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \Lambda\Lambda^{\mathrm{T}} + \Psi)$, we have

$$\mathcal{L} = \sum_{t=1}^{T} -\frac{1}{2} \log |2\pi(\Lambda\Lambda^{\mathrm{T}} + \Psi)| + \frac{1}{2} \mathbf{x}^{(t)\mathrm{T}} (\Lambda\Lambda^{\mathrm{T}} + \Psi)^{-1} \mathbf{x}^{(t)}. \tag{14}$$

The derivatives of $\mathcal{L}$ with respect to Λ and Ψ give complicated expressions that can be evaluated, but not solved directly. So, an iterative Newton-type method can be applied (Fletcher 1987).

Since the factor analyzer is a latent variable model, we can instead apply the expectation maximization (EM) algorithm, as first described in (Rubin and Thayer 1982). Suppose we have an arbitrary distribution $q(\mathbf{z})$ over the factors $\mathbf{z}$. Using Jensen's inequality, we can produce a lower bound on $\mathcal{L}$:

$$\begin{aligned}
\mathcal{L} &= \sum_{t=1}^{T} \log p(\mathbf{x}^{(t)}) \\
&= \sum_{t=1}^{T} \log \int_{\mathbf{z}} p(\mathbf{x}^{(t)}, \mathbf{z}) d\mathbf{z} \\
&= \sum_{t=1}^{T} \log \int_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{x}^{(t)}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \\
&\geq \sum_{t=1}^{T} \int_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}^{(t)}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} = \mathcal{F}
\end{aligned} \tag{15}$$

The distribution $q^*(\mathbf{z})$ that maximizes this bound is found by taking the derivative of $\mathcal{F}$ with respect to $q(\mathbf{z})$ and using a Lagrange multiplier to ensure that $\int_{\mathbf{z}} q(\mathbf{z}) d\mathbf{z} = 1$. For training case $\mathbf{x}^{(t)}$ we obtain

$$\begin{aligned}
q^*(\mathbf{z}) &= p(\mathbf{z}|\mathbf{x}^{(t)}), \\
\mathcal{F}^* &= \mathcal{L}
\end{aligned} \tag{16}$$

The distribution that maximizes the bound is equal to the exact posterior distribution and the resulting bound is tight.

Each iteration of EM consists of computing q^* using the current parameters and then adjusting the parameters to maximize $\mathcal{F}$. (This view was introduced in (Neal and Hinton 1998) and provides a general way to prove that the EM algorithm (Dempster, Laird and Rubin 1977) consistently increases the log-probability of the data.)

For factor analysis, the EM algorithm is

- Initialize Λ to small random values; initialize Ψ so that ψ_n matches the variance of x_n in the training set
- E-Step: Process the training set one case at a time and accumulate the following sufficient statistics, using equations (10), (11) and (12)
  - $\sum_t \mathbf{x}^{(t)} \mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}]^{\mathrm{T}}$

- $\sum_t \mathrm{E}[\mathbf{z}\mathbf{z}^\mathrm{T}|\mathbf{x}^{(t)}]$
- $\sum_t \mathrm{diag}\big((\mathbf{x}^{(t)} - \Lambda\mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}])(\mathbf{x}^{(t)} - \Lambda\mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}])^\mathrm{T} + \Lambda\mathrm{COV}[\mathbf{z}|\mathbf{x}^{(t)}]\Lambda^\mathrm{T}\big)$

diag() sets all off-diagonal elements of a matrix to zero

- M-Step: Update the parameters as follows
  - $\Lambda^{\mathrm{new}} = \big(\sum_t \mathbf{x}^{(t)}\mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}]^\mathrm{T}\big)\big(\sum_t \mathrm{E}[\mathbf{z}\mathbf{z}^\mathrm{T}|\mathbf{x}^{(t)}]\big)^{-1}$
  - $\Psi^{\mathrm{new}} = \sum_t \mathrm{diag}\big((\mathbf{x}^{(t)} - \Lambda\mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}])(\mathbf{x}^{(t)} - \Lambda\mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}])^\mathrm{T} + \Lambda\mathrm{COV}[\mathbf{z}|\mathbf{x}^{(t)}]\Lambda^\mathrm{T}\big)$
- Go back to the E-Step until a stopping criterion is satisfied

In fact, a slightly more efficient EM algorithm is obtained by using Λ^{new} to update Ψ in the M-Step:

$$\Psi^{\mathrm{new}} = \textstyle\sum_t \mathrm{diag}(\mathbf{x}^{(t)}\mathbf{x}^{(t)\mathrm{T}}) - \mathrm{diag}\big(\Lambda^{\mathrm{new}} \sum_t \mathbf{x}^{(t)}\mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}]^\mathrm{T}\big)$$

This update also uses a simpler sufficient statistic, $\sum_t \mathrm{diag}(\mathbf{x}^{(t)}\mathbf{x}^{(t)\mathrm{T}})$.

4 Factor Analysis Using Online EM

Sometimes, we are not given a single batch of training data, but instead receive patterns one at a time and would like to adjust the factor analyzer online. The EM algorithm described in the previous section is a batch method – each iteration processes the batch of training data.

One way to convert a batch learning method to an online learning method is to view each newly received pattern (or small number of patterns) as a "batch" and then only *partly* optimize the model parameters using the "batch" learning method. The degree to which the parameters are modified can be controlled by a "learning rate", $\eta \in [0, 1]$. Large values of η will cause the method to adapt quickly (and possibly be unstable or overfit the current input), whereas small values of η will cause the method to adapt slowly.

For factor analysis, one online EM algorithm is

- Initialize Λ to small random values; initialize Ψ so that ψ_n matches the *anticipated* variance of x_n in the training sequence
- Online E-Step: For the current input $\mathbf{x}$, compute the following statistics, using equations (10) and (11)
  - $\mathrm{E}[\mathbf{z}|\mathbf{x}]$
  - $\mathrm{COV}[\mathbf{z}|\mathbf{x}]$
- Online M-Step: Update the parameters as follows
  - $\Lambda^{\mathrm{new}} = \Lambda + \eta\Psi^{-1}\big((\mathbf{x} - \Lambda\mathrm{E}[\mathbf{z}|\mathbf{x}])\mathrm{E}[\mathbf{z}|\mathbf{x}]^\mathrm{T} - \Lambda\mathrm{COV}[\mathbf{z}|\mathbf{x}]\big)$

– $\Psi^{\mathrm{new}} = (1-\eta)\Psi +$
$\eta \mathrm{diag}\big((\mathbf{x}^{(t)} - \Lambda \mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}])(\mathbf{x}^{(t)} - \Lambda \mathrm{E}[\mathbf{z}|\mathbf{x}^{(t)}])^{\mathrm{T}} + \Lambda \mathrm{COV}[\mathbf{z}|\mathbf{x}^{(t)}]\Lambda^{\mathrm{T}}\big)$

- Go back to the Online E-Step

References

Amari, S.-I., Cichocki, A., and Yang, H. 1996. A new learning algorithm for blind signal separation. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge MA.

Attias, H. 1999. Independent factor analysis. *Neural Computation*, 11(4):803–852.

Bell, A. J. and Sejnowski, T. J. 1995. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159.

Comon, P., Jutten, C., and Herault, J. 1991. Blind separation of sources. *Signal Processing*, 24:11–20.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. 1995. The Helmholtz machine. *Neural Computation*, 7:889–904.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38.

Everitt, B. S. 1984. *An Introduction to Latent Variable Models*. Chapman and Hall, New York NY.

Fletcher, R. 1987. *Practical methods of optimization*. John Wiley & Sons, New York NY.

Frey, B. J. 1997. Continuous sigmoidal belief networks trained using slice sampling. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*. MIT Press, Cambridge MA. Available at `http://www.cs.utoronto.ca/~frey`.

Frey, B. J. 1998. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge MA. See `http://www.cs.utoronto.ca/~frey`.

Frey, B. J. and Hinton, G. E. 1999. Variational learning in non-linear Gaussian belief networks. *Neural Computation*, 11(1):193–214. Available at `http://www.cs.utoronto.ca/~frey`.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. 1995. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161.

Hinton, G. E. and Ghahramani, Z. 1997. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London B*, 352:1177–1190.

Hinton, G. E. and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume I, pages 282–317. MIT Press, Cambridge MA.

Jolliffe, I. T. 1986. *Principal Component Analysis*. Springer-Verlag, New York NY.

Linsker, R. 1988. Self organization in a perceptual network. *Computer*, 21:105–128.

MacKay, D. J. C. 1997. Maximum likelihood and covariant algorithms for independent component analysis. Unpublished manuscript available at `http://wol.ra.phy.cam.ac.uk/mackay`.

Moghaddam, B. and Pentland, A. 1997. Probabilistic visual learning for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710.

Neal, R. M. 1992. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113.

Neal, R. M. and Hinton, G. E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, Norwell MA.

Oja, E. 1989. Neural networks, principal components and subspaces. *International Journal of Neural Systems*, 1:61–68.

Pearlmutter, B. and Parra, L. 1997. Maximum likelihood blind source separation: A context-sensitive generalization of ICA. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*. MIT Press, Cambridge MA.

Roweis, S. and Ghahramani, Z. 1999. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–346.

Rubin, D. and Thayer, D. 1982. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76.

Tipping, M. E. and Bishop, C. M. 1999. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482.

Turk, M. and Pentland, A. 1991. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1).