# A Unifying Review of Linear Gaussian Models


Sam Roweis
Computation and Neural Systems
California Institute of Technology
roweis@cns.caltech.edu

Zoubin Ghahramani
Department of Computer Science
University of Toronto
zoubin@cs.toronto.edu



August 1997
— Submitted for Publication —



**Abstract**

Factor analysis, principal component analysis (PCA), mixtures of Gaussian clusters, vector quantization (VQ), Kalman filter models and hidden Markov models can all be unified as variations of unsupervised learning under a single basic generative model. This is achieved by collecting together disparate observations and derivations made by many previous authors and introducing a new way of linking discrete and continuous state models using a simple nonlinearity. Through the use of other nonlinearities we show how independent component analysis (ICA) is also a variation of the same basic generative model. We show that factor analysis and mixtures of Gaussians can be implemented in autoencoder neural networks and learned using squared error plus the *same* regularization term. We introduce a new model for static data known as sensible principal component analysis (SPCA) as well as a novel concept of spatially adaptive observation noise. We also review some of the literature involving global and local mixtures of the basic models and provide pseudo-code for inference and learning for all the basic models.

# 1 A Unifying Review

Many common statistical techniques for modeling multidimensional static datasets and multidimensional time series can be seen as variants of one underlying model. As we will show later in the paper, these include factor analysis (FA), principal component analysis (PCA), mixtures of Gaussian clusters, vector quantization (VQ), independent component analysis models (ICA), Kalman filter models (a.k.a. linear dynamical systems) and hidden Markov models (HMMs). The relationships between some of these models has been noted in passing in the recent literature. For example, Hinton et al. (1995) have noted that factor analysis and PCA are closely related and Digalakis et al. (1993) relate the forward–backward algorithm for HMMs to Kalman filtering. In this paper we unify many of the disparate observations made by previous authors (Rubin and Thayer, 1982; Delyon, 1993; Digalakis et al., 1993; Hinton et al., 1995; Elliott et al., 1995; Ghahramani and Hinton, 1996a, 1996b, 1997; Hinton and Ghahramani, 1997) and present a review of all these algorithms as instances of a single basic generative model. This unified view allows us to show some interesting relations between previously disparate algorithms. For example, factor analysis and mixtures of Gaussians can be implemented using autoencoder neural networks with different nonlinearities but learned using a squared error cost penalized by the same regularization term (section 8). ICA can be seen as a nonlinear version of factor analysis (section 7). The framework also makes it possible to derive a new model for static data known which is based on PCA but has a sensible probabilistic interpretation (section 5.3) as well as a novel concept of spatially adaptive observation noise (section 9). We also review some of the literature involving global and local mixtures of the basic models and provide pseudo-code for inference and learning for all the basic models (appendix).

# 2 The Basic Model

The basic models we will work with are discrete time linear dynamical systems with Gaussian noise. In such models we assume that the state of the process in question can at any time be summarized by a $k$-vector of *state variables* or *causes* $\mathbf{x}$ which we cannot observe directly. However, the system also produces at each time step an *output* or *observable* $p$-vector $\mathbf{y}$ to which we do have access.

The state $\mathbf{x}$ is assumed to evolve according to simple first-order Markov dynamics; each output vector $\mathbf{y}$ is generated from the current state by a simple linear observation process. Both the state evolution and the observation processes are corrupted by additive Gaussian noise which is also hidden. If we work with a continuous valued state variable $\mathbf{x}$, the basic generative model can be written[1] as:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t = \mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{1a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{1b}$$

where $\mathbf{A}$ is the $k \times k$ *state transition matrix* and $\mathbf{C}$ is the $p \times k$ *observation*, *measurement*, or *generative* matrix. The $k$-vector $\mathbf{w}$ and $p$-vector $\mathbf{v}$ are random variables representing the state evolution and observation noises respectively which are independent of each other and of the values of $\mathbf{x}$ and $\mathbf{y}$. Both of these noise sources are temporally white (uncorrelated from time step to time

[1] All vectors are column vectors. To denote the transpose of a vector or matrix we use the notation $\mathbf{x}^T$. The determinant of a matrix is denoted by $|\mathbf{A}|$ and matrix inversion by $\mathbf{A}^{-1}$. The symbol $\sim$ means "distributed according to". A multivariate normal (Gaussian) distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is written as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The same Gaussian evaluated at the point $\mathbf{x}$ is denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})|_{\mathbf{x}}$.

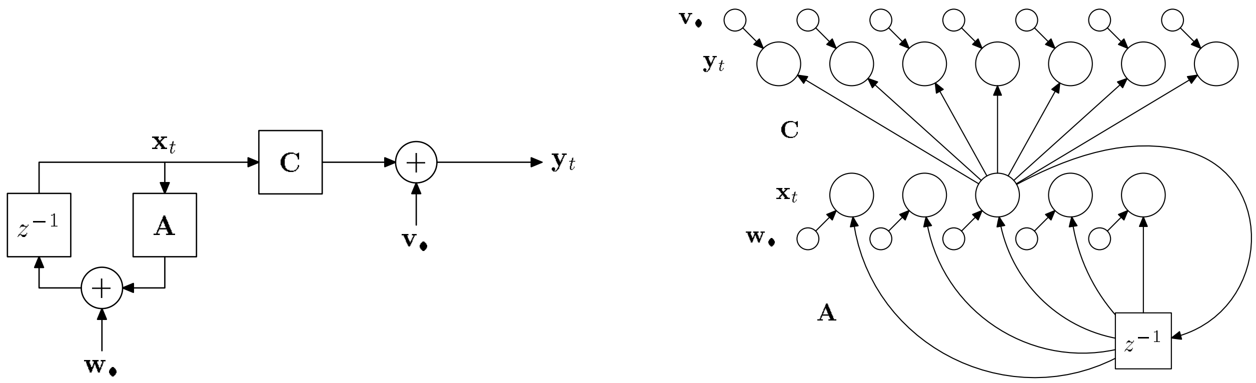


Figure 1: Linear dynamical system generative model. The $z^{-1}$ block is a unit delay. The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model on the right, the smaller circles represent noise sources and all units are linear. Outgoing weights have only been drawn from one hidden unit. This model is equivalent to a Kalman filter model (linear dynamical system).

step) and spatially Gaussian distributed[2] with zero mean and covariance matrices which we shall denote $\mathbf{Q}$ and $\mathbf{R}$ respectively. We have written $\mathbf{w}_\bullet$ and $\mathbf{v}_\bullet$ in place of $\mathbf{w}_t$ and $\mathbf{v}_t$ to emphasize that the noise processes do not have any knowledge of the time index. The restriction to zero mean noise sources is not a loss of generality.[3] Since the state evolution noise is Gaussian and its dynamics are linear, $\mathbf{x}_t$ is a *first-order Gauss-Markov random process*. Note that the noise processes are essential elements of the model: without the process noise $\mathbf{w}_\bullet$, the state $\mathbf{x}_t$ would always either shrink exponentially to zero or blow up exponentially in the direction of the leading eigenvector of $\mathbf{A}$; similarly in the absence of the observation noise $\mathbf{v}_\bullet$ the state would no longer be hidden. Figure 1 illustrates this basic model using both the engineering system block form and the network form more common in machine learning.

Notice that there is degeneracy in the model: all of the structure in the matrix $\mathbf{Q}$ can be moved into the matrices $\mathbf{A}$ and $\mathbf{C}$. This means that we can without loss of generality work with models in which $\mathbf{Q}$ is the identity matrix.[4] Of course $\mathbf{R}$ cannot be restricted in the same way since the values $\mathbf{y}_t$ are observed and hence we are not free to whiten or otherwise rescale them. Finally, the components of the state vector can be arbitrarily reordered; this corresponds to swapping the columns of $\mathbf{C}$ and also of $\mathbf{A}$. Typically we choose an ordering based on the norms of the columns of $\mathbf{C}$ which resolves this degeneracy.

The main idea of the models (1) is that the hidden state sequence $\mathbf{x}_t$ should be an *informative lower dimensional projection* or *explanation* of the complicated observation sequence $\mathbf{y}_t$. With the aid of the dynamical and noise models, the states should summarize the underlying causes of the data much more succinctly than the observations themselves. For this reason we often work with

[2] An assumption that is weakly motivated by the Central Limit Theorem but more strongly by analytic tractability.

[3] Specifically we could always add a $k+1^{st}$ dimension to the state vector which is fixed at unity. Then augmenting $\mathbf{A}$ with an extra column holding the noise mean and an extra row of zeros (except unity in the bottom right corner) takes care of a nonzero mean for $\mathbf{w}_\bullet$. Similarly, adding an extra column to $\mathbf{C}$ takes care of a nonzero mean for $\mathbf{v}_\bullet$.

[4] In particular, since is it a covariance matrix, $\mathbf{Q}$ is symmetric positive semi-definite and thus can be diagonalized to the form $\mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$ (where $\mathbf{E}$ is a rotation matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues). Thus for any model in which $\mathbf{Q}$ is not the identity matrix, we can generate an *exactly equivalent* model using a new state vector $\mathbf{x}' = \mathbf{\Lambda}^{-1/2}\mathbf{E}^T\mathbf{x}$ with $\mathbf{A}' = (\mathbf{\Lambda}^{-1/2}\mathbf{E}^T)\mathbf{A}(\mathbf{E}\mathbf{\Lambda}^{1/2})$ and $\mathbf{C}' = \mathbf{C}(\mathbf{E}\mathbf{\Lambda}^{1/2})$ such that the new covariance of $\mathbf{x}'$ is the identity matrix: $\mathbf{Q}' = \mathbf{I}$.

state dimensions much smaller than the number of observables — in other words $k \ll p$.[5] We assume that both $\mathbf{A}$ and $\mathbf{C}$ are of rank $k$ and that $\mathbf{Q}$,$\mathbf{R}$ and $\mathbf{Q}_1$ (introduced below) are always full rank.

# 3 Probability Computations

The popularity of linear Gaussian models comes from two fortunate analytical properties of Gaussian processes: the sum of two independent Gaussian distributed quantities is also Gaussian distributed[6] and the output of a linear system whose input is Gaussian distributed is again Gaussian distributed. This means that if we assume the initial state $\mathbf{x}_1$ of the system to be Gaussian distributed:

$$\mathbf{x}_1 \sim \mathcal{N}\left(\boldsymbol{\mu}_1, \mathbf{Q}_1\right) \tag{2}$$

then all future states $\mathbf{x}_t$ and observations $\mathbf{y}_t$ will also be Gaussian distributed. In fact, we can write explicit formulae for the conditional expectations of the states and observables:

$$P\left(\mathbf{x}_{t+1}|\mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{A}\mathbf{x}_t, \mathbf{Q}\right)|_{\mathbf{x}_{t+1}} \tag{3a}$$

$$P\left(\mathbf{y}_t|\mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{C}\mathbf{x}_t, \mathbf{R}\right)|_{\mathbf{y}_t} \tag{3b}$$

Furthermore, because of the Markov properties of the model and the Gaussian assumptions about the noise and initial distributions it is easy to write an expression for the joint probability of a sequence of $\tau$ states and outputs:

$$P\left(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right) = P\left(\mathbf{x}_1\right) \prod_{t=1}^{\tau-1} P\left(\mathbf{x}_{t+1}|\mathbf{x}_t\right) \prod_{t=1}^{\tau} P\left(\mathbf{y}_t|\mathbf{x}_t\right) \tag{4}$$

The negative log probability (cost) is just the sum of matrix quadratic forms:

$$\begin{aligned}
-2\log P\left(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right) = &\sum_{t=1}^{\tau}\left[(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t)^T\mathbf{R}^{-1}(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t) + \log|\mathbf{R}|\right] + \\
&\sum_{t=1}^{\tau-1}\left[(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t)^T\mathbf{Q}^{-1}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t) + \log|\mathbf{Q}|\right] + \\
&(\mathbf{x}_1 - \boldsymbol{\mu}_1)^T\mathbf{Q}_1^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1) + \log|\mathbf{Q}_1| + \tau(p+k)\log 2\pi
\end{aligned} \tag{5}$$

[5] More precisely, in a model where all the matrices are full-rank, the problem of inferring the state from a sequence of $\tau$ consecutive observations is well-defined as long $k \leq \tau p$ (a notion related to *observability* in systems theory (Goodwin and Sin, 1984)). For this reason, in dynamic models it is sometimes useful to use state spaces of larger dimension than the observations, $k > p$, in which case the state provides a compact representation of a *sequence* of observations.

[6] In other words the convolution of two Gaussians is again a Gaussian. In particular, the convolution of $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ is $\mathcal{N}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)$. This is not the same as the (false) statement that the sum of two Gaussians is a Gaussian but is the same as the (Fourier domain equivalent) statement that the multiplication of two Gaussians is a Gaussian (although no longer normalized).

# 4 Learning and Estimation Problems

Latent variable models have a wide spectrum of application in data analysis. In some scenarios we know exactly what the hidden states are supposed to be and just want to estimate them. For example in a vision problem the hidden states may be the location or pose of an object; in a tracking problem the states may be positions and momenta. In these cases we can often write down *a priori* the observation and/or state evolution matrices based on our knowledge of the problem structure or physics. The emphasis is on accurate inference of the unobserved information from the data we do have, for example, from an image of an object. In other scenarios we are trying to discover explanations or causes for our data and have no preconceptions about what these causes should be. The observation and state evolution processes are mostly or entirely unknown. The emphasis is instead on robustly learning a few parameters which model the observed data well (assign it high likelihood). Speech modeling is a good example of such a situation — our goal is to find economical models that perform well for recognition tasks but the particular values of the hidden states may not be meaningful or important to us. These two goals, estimating the hidden states given observations and a model, and learning the model parameters, typically manifest themselves in the solution of two distinct problems: *inference* and *system identification.*

## 4.1 Inference: Filtering and Smoothing

The first problem is that of *inference* or *filtering/smoothing* which asks: *given* fixed model parameters $\{\mathbf{A}, \mathbf{C}, \mathbf{Q}, \mathbf{R}, \boldsymbol{\mu}_1, \mathbf{Q}_1\}$ what can be said about the unknown hidden state sequence given some observations? This question is typically made precise in several ways. A very basic quantity we would like to be able to compute is the total likelihood of an observation sequence:

$$P\left(\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right) = \int_{\text{all possible } \{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}} P\left(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right) \, d\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\} \tag{6}$$

This requires an efficient way of integrating (or summing) the joint probability (easily computed by (5) or similar formulae) over all possible paths through state space.

Once this integral is available it is simple to compute the conditional distribution any *one* proposed hidden state sequence given the observations by dividing the joint probability by the total likelihood of the observations:

$$P\left(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\} | \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right) = \frac{P\left(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right)}{P\left(\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right)} \tag{7}$$

Often we are interested in the distribution of the hidden state at a particular time $t$. In *filtering* we attempt to compute this conditional posterior probability:

$$P\left(\mathbf{x}_t | \{\mathbf{y}_1 \ldots \mathbf{y}_t\}\right) \tag{8}$$

given all the observations up to and including time $t$. In *smoothing* we compute the distribution over $\mathbf{x}_t$:

$$P\left(\mathbf{x}_t | \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}\right) \tag{9}$$

given the entire sequence of observations. (It is also possible to ask for the conditional state expectation given observations that extend only a few time steps into the future – *partial smoothing* or that end a few time steps before the current time – *partial prediction.*) These conditional calculations are closely related to the computation of (6) and often the intermediate values of a recursive method used to compute (6) give the desired distributions (8) or (9). Filtering and smoothing have been extensively studied for continuous state models in the signal processing community, starting with the seminal works of Kalman (Kalman, 1960; Kalman and Bucy, 1961) and Rauch (Rauch, 1963; Rauch et al., 1965) although this literature is often not well known in the machine learning community. For the discrete state models much of the literature stems from the work of Baum and colleagues (Baum and Petrie, 1966; Baum and Eagon, 1967; Baum et al., 1970; Baum, 1972) on hidden Markov models and of Viterbi (Viterbi, 1967) and others on optimal decoding. The recent book by Elliott and colleagues (1995) contains a thorough mathematical treatment of filtering and smoothing for many general models.

### 4.2 Learning (System Identification)

The second problem of interest with linear Gaussian models is the *learning* or *system identification* problem: given only an observed sequence (or perhaps several sequences) of outputs $\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}$ find the parameters $\{\mathbf{A}, \mathbf{C}, \mathbf{Q}, \mathbf{R}, \boldsymbol{\mu}_1, \mathbf{Q}_1\}$ which maximize the likelihood of the observed data as computed by (6).

The learning problem has been investigated extensively by neural network researchers for static models and also for some discrete state dynamic models such as hidden Markov models or the more general Bayesian belief networks. There is a corresponding area of study in control theory known as *system identification* which investigates learning in continuous state models. For linear Gaussian models there are several approaches to system identification (Ljung and Söderström, 1983), but to clarify the relationship between these models and the others we review in this paper, we focus on system identification methods based on the *EM* algorithm, described below. The *EM* algorithm for linear Gaussian dynamical systems was originally derived by Shumway and Stoffer (1982) and recently reintroduced (and extended) in the neural computation field by Ghahramani and Hinton (1996a, 1996b). Digalakis *et al.* (1993) made a similar reintroduction and extension in the speech processing community. Once again we mention the book by Elliott et al. (1995) which also covers learning in this context.

The basis of all the learning algorithms presented by these authors is the powerful *EM* or *expectation-maximization* algorithm (Baum and Petrie, 1966; Dempster et al., 1977). The objective of the *EM* algorithm is to maximize the likelihood of the observed data (6) in the presence of hidden variables. Let us denote the observed data by $\mathbf{Y} = \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}$, the hidden variables by $\mathbf{X} = \{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}$, and the parameters of the model by $\theta$. Maximizing the likelihood as a function of $\theta$ is equivalent to maximizing the log likelihood:

$$\mathcal{L}(\theta) = \log P(\mathbf{Y}|\theta) = \log \int_{\mathbf{X}} P(\mathbf{X}, \mathbf{Y}|\theta)\, d\mathbf{X} \tag{10}$$

Using *any* distribution $Q$ over the hidden variables, we can obtain a lower bound on $\mathcal{L}$:

$$\begin{aligned}
\log \int_{\mathbf{X}} P(\mathbf{Y}, \mathbf{X}|\theta)\, d\mathbf{X} &= \log \int_{\mathbf{X}} Q(\mathbf{X}) \frac{P(\mathbf{X}, \mathbf{Y}|\theta)}{Q(\mathbf{X})}\, d\mathbf{X} && (11) \\
&\geq \int_{\mathbf{X}} Q(\mathbf{X}) \log \frac{P(\mathbf{X}, \mathbf{Y}|\theta)}{Q(\mathbf{X})}\, d\mathbf{X} && (12) \\
&= \int_{\mathbf{X}} Q(\mathbf{X}) \log P(\mathbf{X}, \mathbf{Y}|\theta)\, d\mathbf{X} - \int_{\mathbf{X}} Q(\mathbf{X}) \log Q(\mathbf{X})\, d\mathbf{X} && (13) \\
&= \mathcal{F}(Q, \theta) && (14)
\end{aligned}$$

where the middle inequality is known as Jensen's inequality and can be proven using the concavity of the log function. If we define the *energy* of a global configuration $(\mathbf{X}, \mathbf{Y})$ to be $-\log P(\mathbf{X}, \mathbf{Y}|\theta)$, then some readers may notice that the lower bound $\mathcal{F}(Q, \theta) \leq \mathcal{L}(\theta)$ is the negative of a quantity known in statistical physics as the *free energy*: the expected energy under $Q$ minus the entropy of $Q$ (Neal and Hinton, 1993). The *EM* algorithm alternates between maximizing $\mathcal{F}$ with respect to the *distribution* $Q$ and the *parameters* $\theta$, respectively, holding the other fixed. Starting from some initial parameters $\theta_0$:

$$\textbf{E step:} \qquad Q_{k+1} \leftarrow \arg\max_{Q} \; \mathcal{F}(Q, \theta_k) \tag{15}$$

$$\textbf{M step:} \qquad \theta_{k+1} \leftarrow \arg\max_{\theta} \; \mathcal{F}(Q_{k+1}, \theta) \tag{16}$$

It is easy to show that the maximum in the E step results when $Q$ is exactly the marginal distribution of $\mathbf{X}$: $Q_{k+1}(\mathbf{X}) = P(\mathbf{X}|\mathbf{Y}, \theta_k)$, at which point the bound becomes an equality: $\mathcal{F}(Q_{k+1}, \theta_k) = \mathcal{L}(\theta_k)$. The maximum in the M step is obtained by maximizing the the first term in (13), since the entropy of $Q$ does not depend on $\theta$:

$$\textbf{M step:} \qquad \theta_{k+1} \leftarrow \arg\max_{\theta} \int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Y}, \theta_k) \log P(\mathbf{X}, \mathbf{Y}|\theta)\, d\mathbf{X}.$$

This is the expression most often associated with the *EM* algorithm, but it obscures the elegant interpretation of *EM* as coordinate ascent in $\mathcal{F}$ (Neal and Hinton, 1993). Since $\mathcal{F} = \mathcal{L}$ at the beginning of each M step, and since the E step does not change $\theta$, we are guaranteed not to decrease the likelihood after each combined *EM* step.

Therefore, at the heart of the recursive *EM* learning procedures is the following idea: use the solutions to the filtering/smoothing problem to estimate the unknown hidden states given the observations and the current model parameters. Then use this fictitious "complete" or "joint" data to solve for new model parameters. Given the estimated states obtained from the inference algorithm, it is usually easy to solve for new parameters. For linear Gaussian models this typically involves minimizing quadratic forms such as (5) which can be done with linear regression. This process is repeated, using these new model parameters to infer the hidden states again, and so on. We shall review the details of particular algorithms as we present the various cases below, however we now touch upon one general point which often causes confusion. Our goal is to maximize the *total* likelihood (6) (or equivalently maximize the total log likelihood) of the observed data with respect to the model parameters. This means integrating (or summing) over all ways in which the generative model could have produced the data. As a consequence of using the *EM* algorithm to do this maximization we find ourselves needing to compute (and maximize) the *expected* log likelihood of the joint data, where the expectation is taken over the distribution of hidden values predicted by the current model parameters and the observations. Thus, it appears that we are maximizing the incorrect quantity, but doing so is in fact guaranteed to increase (or keep the same) the quantity of interest at each iteration of the algorithm.

# 5 Continuous state linear Gaussian systems

Having described the basic model and learning procedure we now focus on specific linear instances of the model in which the hidden state variable $\mathbf{x}$ is continuous and the noise processes are Gaussian. This will allow us to elucidate the relationship between factor analysis, PCA, and Kalman filter models. We divide our discussion into models which generate *static data* and those which generate *dynamic data*. Static data has no temporal dependence – no information would be lost by permuting the ordering of the data points $\mathbf{y}_t$; whereas for dynamic data the time ordering of the data points is crucial.

## 5.1 Static Data Modeling: Factor Analysis, SPCA and PCA

In many situations we have reason to believe (or at least to assume) that each point in our dataset was generated independently and identically. In other words, there is no natural (temporal) ordering to the data points, they merely a form a collection. In such cases we assume that the underlying state vector $\mathbf{x}$ has no dynamics, *i.e.* the matrix $\mathbf{A}$ is the zero matrix, and therefore $\mathbf{x}$ is simply a constant (which we take without loss of generality to be the zero vector) corrupted by noise. The new generative model then becomes:

$$\mathbf{A} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x}_\bullet = \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{17a}$$

$$\mathbf{y}_\bullet = \mathbf{C}\mathbf{x}_\bullet + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{17b}$$

Notice that since $\mathbf{x}_t$ is driven only by the noise $\mathbf{w}_\bullet$ and since $\mathbf{y}_t$ depends only on $\mathbf{x}_t$ all temporal dependence has disappeared. This is the motivation for the term *static* and for the notations $\mathbf{x}_\bullet$ and $\mathbf{y}_\bullet$ above. We also no longer use a separate distribution for the initial state: $\mathbf{x}_1 \sim \mathbf{x}_\bullet \sim \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$.

This model is illustrated in figure 2. We can analytically integrate (6) to obtain the marginal distribution of $\mathbf{y}_\bullet$, which is the Gaussian

$$\mathbf{y}_\bullet \sim \mathcal{N}\left(\mathbf{0}, \mathbf{C}\mathbf{Q}\mathbf{C}^T + \mathbf{R}\right) \tag{18}$$

Two things are important to notice. First, the degeneracy mentioned above persists between the structure in $\mathbf{Q}$ and $\mathbf{C}$.[7] This means there is no loss of generality in restricting $\mathbf{Q}$ to be diagonal. Furthermore, there is arbitrary sharing of scale between a diagonal $\mathbf{Q}$ and $\mathbf{C}$: typically we either restrict the columns of $\mathbf{C}$ to be unit vectors or make $\mathbf{Q}$ the identity matrix to resolve this degeneracy.

Second, the covariance matrix $\mathbf{R}$ of the observation noise *must* be restricted in some way for the model to capture any interesting or informative projections in the state $\mathbf{x}_\bullet$. If $\mathbf{R}$ were not

[7] If we diagonalize $\mathbf{Q}$ and rewrite the covariance of $\mathbf{y}_\bullet$ the degeneracy becomes clear: $\mathbf{y}_\bullet \sim \mathcal{N}\left(\mathbf{0}, (\mathbf{C}\mathbf{E}\mathbf{\Lambda}^{1/2})(\mathbf{C}\mathbf{E}\mathbf{\Lambda}^{1/2})^T + \mathbf{R}\right)$. To make $\mathbf{Q}$ diagonal we simply replace $\mathbf{C}$ with $\mathbf{CE}$.

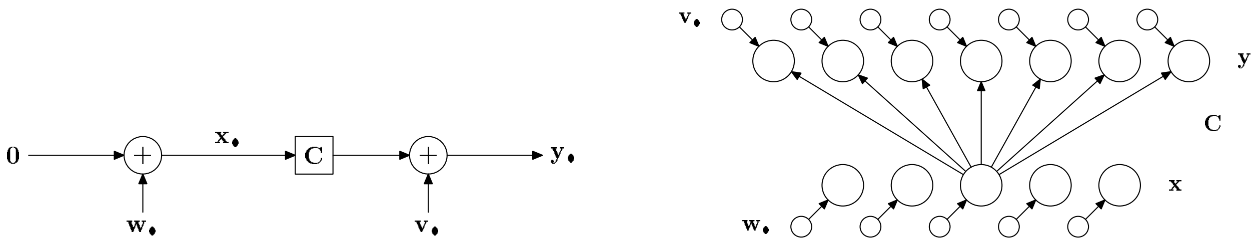


Figure 2: Static generative model (continuous state).The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model on the right, the smaller circles represent noise sources and all units are linear. Outgoing weights have only been drawn from one hidden unit. This model is equivalent to factor analysis, SPCA and PCA models depending on the output noise covariance. For factor analysis, $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R}$ is diagonal. For SPCA, $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = \alpha\mathbf{I}$. For PCA, $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon\mathbf{I}$.

restricted, learning could simply choose $\mathbf{C} = \mathbf{0}$ and then set $\mathbf{R}$ to be the sample covariance of the data thus trivially achieving the maximum likelihood model by explaining all of the structure in the data as noise. (Remember that since the model has reduced to a single Gaussian distribution for $\mathbf{y}_\bullet$ we can do no better than having the covariance of our model equal the sample covariance of our data.) Note that restricting $\mathbf{R}$, unlike making $\mathbf{Q}$ diagonal, *does* constitute some loss of generality from the original model of equations (17).

There is an intuitive spatial way to think about this static generative model: we use white noise to generate a spherical ball of density in $k$-dimensional state space. This ball is then stretched and rotated into $p$-dimensional observation space by the matrix $\mathbf{C}$ where it looks like a $k$-dimensional pancake. The pancake is then convolved with the covariance density of $\mathbf{v}_\bullet$ (described by $\mathbf{R}$) to get the final covariance model for $\mathbf{y}_\bullet$. We want the resulting ellipsoidal density to be as close as possible to the ellipsoid density which is the sample covariance of our data. If we restrict the shape of the $\mathbf{v}_\bullet$ covariance by constraining $\mathbf{R}$ we can force interesting information to appear in both $\mathbf{R}$ and $\mathbf{C}$ as a result.

Finally, observe that all varieties of filtering and smoothing reduce to the same problem in this static model because there is no time dependence. We are only seeking the posterior probability $P(\mathbf{x}_\bullet|\mathbf{y}_\bullet)$ over a single hidden state given the corresponding single observation. This inference is easily done by linear matrix projection and the resulting density is itself Gaussian:

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \frac{P(\mathbf{y}_\bullet|\mathbf{x}_\bullet)\,P(\mathbf{x}_\bullet)}{P(\mathbf{y}_\bullet)} = \frac{\mathcal{N}(\mathbf{C}\mathbf{x}_\bullet, \mathbf{R})\,|_{\mathbf{y}_\bullet}\,\mathcal{N}(\mathbf{0}, \mathbf{I})\,|_{\mathbf{x}_\bullet}}{\mathcal{N}(\mathbf{0}, \mathbf{C}\mathbf{C}^T + \mathbf{R})\,|_{\mathbf{y}_\bullet}} \tag{19a}$$

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \mathcal{N}(\boldsymbol{\beta}\mathbf{y}_\bullet, I - \boldsymbol{\beta}\mathbf{C})\,|_{\mathbf{x}_\bullet}\,, \qquad \boldsymbol{\beta} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \mathbf{R})^{-1} \tag{19b}$$

from which we obtain not only the expected value $\boldsymbol{\beta}\mathbf{y}_\bullet$ of the unknown state but also an estimate of the uncertainty in this value in the form of the covariance $\mathbf{I} - \boldsymbol{\beta}\mathbf{C}$. Computing the likelihood of a data point $\mathbf{y}_\bullet$ is merely an evaluation under the Gaussian (18). The learning problem now consists of identifying the matrices $\mathbf{C}$ and $\mathbf{R}$. There are a family of *EM* algorithms to do this for the various cases discussed below which are reviewed in detail at the end of this note.

### 5.2 Factor Analysis

If we decide to restrict the covariance matrix $\mathbf{R}$ which controls the observation noise to be *diagonal* (in other words the covariance ellipsoid of $\mathbf{v}_\bullet$ is axis aligned) and we set the state noise $\mathbf{Q}$ to be the identity matrix then we recover exactly a standard statistical model known as maximum likelihood *factor analysis*. The unknown states $\mathbf{x}$ are called the *factors* in this context; the matrix $\mathbf{C}$ is called the *factor loading matrix* and the diagonal elements of $\mathbf{R}$ are often known as the *uniquenesses*. (See Everitt (1994) for a brief and clear introduction.) The inference calculation is done exactly as in (19b) above. The learning algorithm for the loading matrix and the uniquenesses is exactly an *EM* algorithm except that we must take care to constrain $\mathbf{R}$ properly (which is fortunately as easy as taking the diagonal of the unconstrained maximum likelihood estimate – see (Rubin and Thayer, 1982; Ghahramani and Hinton, 1997)). If $\mathbf{C}$ is completely free this procedure is called *exploratory factor analysis*; if we build *a priori* zeros into $\mathbf{C}$ it is *confirmatory factor analysis*. Note that in exploratory factor analysis we are trying to model the covariance structure of our data with $p+pk-k(k-1)/2$ free parameters[8] instead of $p(p+1)/2$ free parameters in a full covariance matrix.

The diagonality of $\mathbf{R}$ is the key assumption here. Factor analysis attempts to explain the covariance structure in the observed data by putting all the variance unique to each coordinate in the matrix $\mathbf{R}$ and putting all the correlation structure into $\mathbf{C}$ (this observation was first made by Lyttkens (1966) in response to work by Wold). In essence, factor analysis considers the axis rotation in which the original data arrived to be special because observation noise (often called *sensor noise*) is independent along the coordinates in these axes. However, the original scaling of the coordinates is unimportant: if we were to change the units in which we measured some of the components of $\mathbf{y}$, factor analysis could merely rescale the corresponding entry in $\mathbf{R}$ and row in $\mathbf{C}$ and achieve a new model which assigns the rescaled data identical likelihood. On the other hand if we rotate the axes in which we measure the data we could not easily fix things since the noise $\mathbf{v}$ is constrained to have axis aligned covariance ($\mathbf{R}$ is diagonal).

### 5.3 SPCA and PCA

If instead of restricting $\mathbf{R}$ to be merely diagonal, we require it to be a multiple of the identity matrix (in other words the covariance ellipsoid of $\mathbf{v}_\bullet$ is spherical) then we have a model which we will call *sensible principal component analysis* or SPCA (Roweis, 1997a). The columns of $\mathbf{C}$ are known as the *sensible principal components* and we will call the scalar value on the diagonal of $\mathbf{R}$ the *global noise level*. Note that SPCA uses $1+pk-k(k-1)/2$ free parameters to model the covariance. Once again, inference is done with equation (19b) and learning by the *EM* algorithm (except that we now take the trace of the maximum likelihood estimate for $\mathbf{R}$ to learn the noise level – see (Roweis, 1997a)). Unlike factor analysis, SPCA considers the original axis rotation in which the data arrived to be unimportant: if the measurement coordinate system were rotated SPCA could (left) multiply $\mathbf{C}$ by the same rotation and the likelihood of the new data would not change. On the other hand, the original scaling of the coordinates *is* privileged because SPCA assumes that the observation noise has the same variance in all directions in the measurement units used for the observed data. If we were to rescale one of the components of $\mathbf{y}$ the model could not be easily corrected since $\mathbf{v}$ has spherical covariance ($\mathbf{R} = \epsilon\mathbf{I}$). The SPCA model is very similar to the independently proposed *probabilistic principal component analysis* (Tipping and Bishop, 1997).

If we go even further and take the limit $\mathbf{R} = \lim_{\epsilon\to 0} \epsilon\mathbf{I}$ (while keeping the diagonal elements of $\mathbf{Q}$ finite[9]) then we obtain the standard *principal component analysis* or PCA model. The directions

[8] The correction $k(k-1)/2$ comes in because of the degeneracy in ordering the factors – see for example (Everitt, 1984).

[9] Since isotropic scaling of the data space is arbitrary we could just as easily take the limit as the diagonal elements of $\mathbf{Q}$ became infinite while holding $\mathbf{R}$ finite or take both limits at once. The idea is that the noise variance becomes infinitesimal compared to the scale of the data.

of the columns of $\mathbf{C}$ are known as the *principal components*. Inference now reduces to simple least squares projection[10]:

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \mathcal{N}(\boldsymbol{\beta}\mathbf{y}_\bullet, I - \boldsymbol{\beta}\mathbf{C})\,|_{\mathbf{x}_\bullet}, \qquad \boldsymbol{\beta} = \lim_{\epsilon\to 0} \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \epsilon\mathbf{I})^{-1} \tag{20a}$$

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \mathcal{N}\left((\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}_\bullet, \mathbf{0}\right)|_{\mathbf{x}_\bullet} = \delta(\mathbf{x}_\bullet - (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}_\bullet) \tag{20b}$$

Since the noise has become infinitesimal, the posterior over states collapses to a single point and the covariance becomes zero. There is still an *EM* algorithm for learning (Roweis, 1997a) although it can of course only learn $\mathbf{C}$. For PCA, we could just diagonalize the sample covariance of the data and take the leading $k$ eigenvectors multiplied by their eigenvalues to be the columns of $\mathbf{C}$. This approach would give us $\mathbf{C}$ in one step but has many problems.[11] The *EM* learning algorithm amounts to an iterative procedure for finding these leading eigenvectors without explicit diagonalization and requires a minimum amount of data.

An important final comment is that (regular) PCA does not define a proper density model in the observation space. So we cannot ask directly about the likelihood assigned by the model to some data. We can however, examine a quantity which is proportional to the negative log likelihood in the limit of zero noise. This is the sum squared deviation of each data point from its projection. It is this "cost" which the learning algorithm ends up minimizing and which is the only available evaluation of how well a PCA model fits new data. This is one of the most critical failings of PCA: translating points by arbitrary amounts inside the principal subspace has no effect on the model.

### 5.4 Time Series Modeling: Kalman Filter Models

We use the term *dynamic data* to refer to observation sequences in which the temporal ordering is important. For such data we do not want to ignore the state evolution dynamics, as it provides the only aspect of the model capable of capturing temporal structure. Systems described by the original dynamic generative model:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t = \mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{1a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{1b}$$

are known as *linear dynamical systems* or *Kalman filter models* and have been extensively investigated by the engineering and control communities for decades. The emphasis has traditionally been on the inference problems: the famous Discrete Kalman Filter (Kalman, 1960; Kalman and Bucy, 1961) gives an efficient recursive solution to the optimal filtering and likelihood computation problems, while the RTS recursions (Rauch, 1963; Rauch et al., 1965) solve the optimal smoothing

[10] Recall that if $\mathbf{C}$ is $p \times k$ with $p > k$ and is rank $k$ then left multiplication by $\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}$ (which appears not to be well defined because $(\mathbf{C}\mathbf{C}^T)$ is not invertible) is *exactly equivalent to* left multiplication by $(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T$. This is the same as the SVD idea of defining the "inverse" of the diagonal singular value matrix as the inverse of an element unless it is zero in which case it remains zero. The intuition is that even though $\mathbf{C}\mathbf{C}^T$ truly is not invertible, the directions along which it is not invertible are exactly those which $\mathbf{C}^T$ is about to project out.

[11] It is computationally very hard to diagonalize or invert large matrices. It also requires an enormous amount of data to make a large sample covariance matrix full rank. So if we are working with patterns in a large (thousands) number of dimensions and want to extract only a few (tens) principal components we cannot naively try to diagonalize the sample covariance of our data. Techniques like the *snap-shot method* (Sirovich, 1987) attempt to address this but still require the diagonalization of an $N$ by $N$ matrix where $N$ is the number of data points. The *EM* algorithm approach solves all of these problems, requiring no explicit diagonalization whatsoever and the inversion of only a $k$ by $k$ matrix. Empirical experiments indicate that it converges to the principal components in a few iterations.

problem. Learning of unknown model parameters was studied by Shumway and Stoffer (1982) ($\mathbf{C}$ known) and by Ghahramani and Hinton (1996a) and Digalakis *et al.* (1993) (all parameters unknown). Figure 1 illustrates this model.

We can extend our spatial intuition of the static case to this dynamic model. Just as before, any point in state space is surrounded by a ball of density (described by $\mathbf{Q}$) which is stretched (by $\mathbf{C}$) into a pancake in observation space and then convolved with the observation noise covariance (described by $\mathbf{R}$). However, unlike the static case in which we always centred our ball of density on the origin in state space, the centre of the state space ball now "flows" from time step to time step. The flow is according to the field described by the eigenvalues and eigenvectors of the matrix $\mathbf{A}$. We move to a new point according to this flow field, then we centre our ball on that point and pick a new state. From this new state we again flow to a new point and then apply noise. If $\mathbf{A}$ is the identity matrix (not the zero matrix) then the "flow" does not move us anywhere and the state just evolves according to a random walk of the noise set by $\mathbf{Q}$.

# 6 Discrete State Linear Gaussian Models

We now consider a simple modification of the basic continuous state model in which the state at any time takes on one of a finite number of discrete values. Many real world processes, especially those which have distinct modes of operation are better modeled by internal states which are not continuous. (It is also possible to construct models which have a mixed continuous and discrete state.) The state evolution is still first-order Markovian dynamics and the observation process is still linear with additive Gaussian noise. The modification involves the use of the *winner-take-all* nonlinearity $\mathbf{WTA}[\cdot]$ defined such that $\mathbf{WTA}[\mathbf{x}]$ for any vector $\mathbf{x}$ is a new vector with unity in the position of the largest coordinate of the input and zeros in all other positions. The discrete state generative model is now simply:

$$\mathbf{x}_{t+1} = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_t] = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet] \tag{21a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \tag{21b}$$

where $\mathbf{A}$ is no longer known as the *state transition matrix* (although we will see said matrix shortly). As before, the $k$-vector $\mathbf{w}$ and $p$-vector $\mathbf{v}$ are temporally white and spatially Gaussian distributed noises independent of each other and of $\mathbf{x}$ and $\mathbf{y}$. The initial state $\mathbf{x}_1$ is generated in the obvious way:

$$\mathbf{x}_1 = \mathbf{WTA}[\mathcal{N}(\boldsymbol{\mu}_1, \mathbf{Q}_1)] \tag{22}$$

(though we will soon see that without loss of generality $\mathbf{Q}_1$ can be restricted to be the identity matrix). This discrete state generative model is illustrated in figure 3.

## 6.1 Static Data Modeling: Mixtures of Gaussians and Vector Quantization

Just as in the continuous state model we can consider situations in which there is no natural ordering to our data, and so set the matrix $\mathbf{A}$ to be the zero matrix. In this discrete state case the generative model becomes:

$$\mathbf{A} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x}_\bullet = \mathbf{WTA}[\mathbf{w}_\bullet] \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}) \tag{23a}$$

$$\mathbf{y}_\bullet = \mathbf{C}\mathbf{x}_\bullet + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{23b}$$

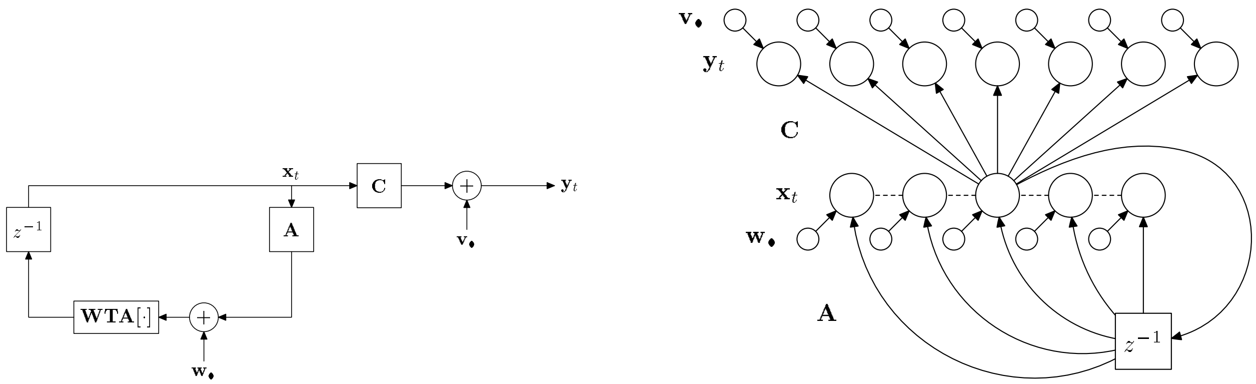


Figure 3: Discrete state generative model for dynamic data. The $\mathbf{WTA}[\cdot]$ block implements the winner-take-all nonlinearity. The $z^{-1}$ block is a unit delay. The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model on the right, the smaller circles represent noise sources and the hidden units $\mathbf{x}$ have a winner take all behaviour (indicated by dashed lines). Outgoing weights have only been drawn from one hidden unit. This model is equivalent to a hidden Markov model with tied output covariances.

Each state $\mathbf{x}_\bullet$ is generated independently[12] according to a fixed discrete probability histogram controlled by the mean and covariance of $\mathbf{w}_\bullet$. Specifically, $\boldsymbol{\pi}_j = P(\mathbf{x}_\bullet = \mathbf{e}_j)$ is the probability assigned by the Gaussian $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q})$ to the region of $k$-space in which the $j^{th}$ coordinate is larger than all others. (Here $\mathbf{e}_j$ is the unit vector along the $j^{th}$ coordinate direction.) Notice that to obtain non-uniform priors $\boldsymbol{\pi}_j$ with the $\mathbf{WTA}[\cdot]$ nonlinearity we require a non-zero mean $\boldsymbol{\mu}$ for the noise $\mathbf{w}_\bullet$. Once the state has been chosen, the corresponding output $\mathbf{y}_\bullet$ is generated from a Gaussian whose mean is the $j^{th}$ column of $\mathbf{C}$ and whose covariance is $\mathbf{R}$. This is exactly the standard mixture of Gaussian clusters model except that the covariances of all the clusters are constrained to be the same. The probabilities $\boldsymbol{\pi}_j = P(\mathbf{x}_\bullet = \mathbf{e}_j)$ correspond to the mixing coefficients of the clusters and the columns of $\mathbf{C}$ are the cluster means. Constraining $\mathbf{R}$ in various ways corresponds to constraining the shape of the covariance of the clusters. This model is illustrated in figure 4 below.

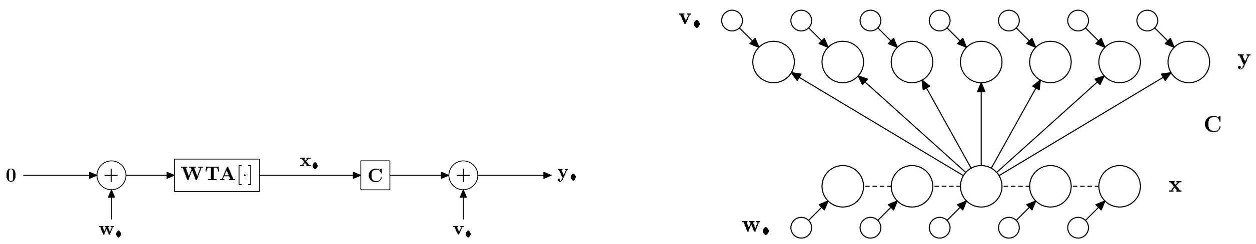


Figure 4: Static generative model (discrete state). The $\mathbf{WTA}[\cdot]$ block implements the winner-take-all nonlinearity. The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model on the right, the smaller circles represent noise sources and the hidden units $\mathbf{x}$ have a winner take all behaviour (indicated by dashed lines). Outgoing weights have only been drawn from one hidden unit. This model is equivalent to a mixture of Gaussian clusters with tied covariances $\mathbf{R}$ or to vector quantization (VQ) when $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon \mathbf{I}$.

[12] As in the continuous static case we again dispense with any special treatment of the initial state.

To compute the likelihood of a data point, we can explicitly perform the sum equivalent to the integral (6) since it contains only $k$ terms:

$$P(\mathbf{y}_\bullet) = \sum_{i=1}^{k} P(\mathbf{x}_\bullet = \mathbf{e}_j, \mathbf{y}_\bullet) = \sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\big|_{\mathbf{y}_\bullet} P(\mathbf{x}_\bullet = \mathbf{e}_i) = \sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\big|_{\mathbf{y}_\bullet} \boldsymbol{\pi}_i \tag{24}$$

where $\mathbf{C}_i$ denotes the $i^{th}$ column of $\mathbf{C}$. Again, all varieties of inference and filtering are the same and we are simply seeking the set of discrete probabilities $P(\mathbf{x}_\bullet = \mathbf{e}_j|\mathbf{y}_\bullet)$ $j = 1 \ldots k$. In other words, we need to do probabilistic classification. The problem is easily solved by computing the *responsibilities* $\hat{\mathbf{x}}_\bullet$ that each cluster has for the data point $\mathbf{y}_\bullet$:

$$(\hat{\mathbf{x}}_\bullet)_j = P(\mathbf{x}_\bullet = \mathbf{e}_j|\mathbf{y}_\bullet) = \frac{P(\mathbf{x}_\bullet = \mathbf{e}_j, \mathbf{y}_\bullet)}{P(\mathbf{y}_\bullet)} = \frac{P(\mathbf{x}_\bullet = \mathbf{e}_j, \mathbf{y}_\bullet)}{\sum_{i=1}^{k} P(\mathbf{x}_\bullet = \mathbf{e}_i, \mathbf{y}_\bullet)} \tag{25a}$$

$$(\hat{\mathbf{x}}_\bullet)_j = \frac{\mathcal{N}(\mathbf{C}_j, \mathbf{R})\big|_{\mathbf{y}_\bullet} P(\mathbf{x}_\bullet = \mathbf{e}_j)}{\sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\big|_{\mathbf{y}_\bullet} P(\mathbf{x}_\bullet = \mathbf{e}_i)} = \frac{\mathcal{N}(\mathbf{C}_j, \mathbf{R})\big|_{\mathbf{y}_\bullet} \boldsymbol{\pi}_j}{\sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\big|_{\mathbf{y}_\bullet} \boldsymbol{\pi}_i} \tag{25b}$$

The mean $\hat{\mathbf{x}}_\bullet$ of the state vector given a data point is exactly the vector of responsibilities for that data point. This quantity defines the entire posterior distribution of the discrete hidden state given the data point. As a measure of the randomness or uncertainty in the hidden state one could evaluate the entropy or *normalized entropy*[13] of the discrete distribution corresponding to $\hat{\mathbf{x}}_\bullet$. Although this may seem related to the variance of the posterior in factor analysis, this analogy is deceptive. Since $\hat{\mathbf{x}}_\bullet$ defines the entire distribution; no other "variance" measure is needed. Learning consists of finding the cluster means (columns of $\mathbf{C}$), the covariance $\mathbf{R}$, and the mixing coefficients $\boldsymbol{\pi}_j$. This is easily done with *EM* and corresponds exactly to maximum likelihood competitive learning (Duda and Hart, 1973; Nowlan, 1991) except that all the clusters share the same covariance. Later we introduce extensions to the model which remove this restriction.

As in the continuous state case, we can consider the limit as the observation noise becomes infinitessimal compared to the scale of the data. What results is the standard *vector quantization* or VQ model. The inference (classification) problem is now solved by the *one-nearest-neighbour* rule, using Euclidean distance if $\mathbf{R}$ is a multiple of the identity matrix, or Mahanalobis distance in the unscaled matrix $\mathbf{R}$ otherwise. Similarly to PCA, since the observation noise has disappeared, the posterior collapses to have all of its mass on one cluster (the closest) and the corresponding uncertainty (entropy) becomes zero. Learning with *EM* is equivalent to using a batch version of the *k-means* algorithm such as that proposed by Lloyd (1982). As with PCA, vector quantization does not define a proper density in the observation space. Once again, we examine the sum squared deviation of each point from its closest cluster centre as a quantity proportional to the likelihood in the limit of zero noise. Batch *k-means* algorithms minimize this cost in lieu of a proper likelihood.

### 6.2 Time Series Modeling: Hidden Markov Models

We return now to the fully dynamic discrete state model introduced above:

$$\mathbf{x}_{t+1} = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_t] = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet] \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{21a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{21b}$$

[13] The entropy of the distribution divided by the logarithm of $k$ so that it always lies between zero and one.

Our key observation is that the dynamics described by equation (21a) are exactly equivalent to the more traditional discrete Markov chain dynamics using a state transition matrix $\mathbf{T}$, where $\mathbf{T}_{ij} = P(\mathbf{x}_{t+1} = \mathbf{e}_j | \mathbf{x}_t = \mathbf{e}_i)$. It is easy to see how to compute the equivalent state transition matrix $\mathbf{T}$ given $\mathbf{A}$ and $\mathbf{Q}$ above: $\mathbf{T}_{ij}$ is the probability assigned by the Gaussian whose mean is the $i^{th}$ column of $\mathbf{A}$ (and whose covariance is $\mathbf{Q}$) to the region of $k$-space in which the $j^{th}$ coordinate is larger than all others. It is also true that for any transition matrix $\mathbf{T}$ (whose rows each sum to unity) there exist matrices $\mathbf{A}$ and $\mathbf{Q}$ such that the dynamics are equivalent.[14] Similarly, the initial probability mass function for $\mathbf{x}_1$ is easily computed from $\boldsymbol{\mu}_1$ and $\mathbf{Q}_1$ and for any desired histogram over the states for $\mathbf{x}_1$ there exist a $\boldsymbol{\mu}_1$ and $\mathbf{Q}_1$ which achieve it.

Similar degeneracy exists in this discrete state model as in the continuous state model except that it is now between the structure of $\mathbf{A}$ and $\mathbf{Q}$. Since for *any* noise covariance $\mathbf{Q}$ the means in the columns of $\mathbf{A}$ can be chosen to set any equivalent transition probabilities $\mathbf{T}_{ij}$ we can without loss of generality restrict $\mathbf{Q}$ to be the identity matrix and use only the means in the columns of $\mathbf{A}$ to set probabilities. Equivalently, we can restrict $\mathbf{Q}_1 = \mathbf{I}$ and use only the mean $\boldsymbol{\mu}_1$ to set the probabilities for the initial state $\mathbf{x}_1$.

Thus, this generative model is equivalent to a standard hidden Markov model except that the emission probability densities are all constrained to have the same covariance. Likelihood and filtering computations are performed with the so-called *forward* (alpha) recursions, while complete smoothing is done with the *forward-backward* (alpha-beta) recursions. The *EM* algorithm for learning is exactly the well known Baum-Welsh or forward-backward re-estimation procedure (Baum and Petrie, 1966; Baum et al., 1970).

There is an important and peculiar consequence of discretizing the state which impacts the smoothing problem. The state *sequence* formed by taking the most probable state of the posterior distribution at each time (as computed by the forward-backward recursions given the observed data and model parameters) is *not* the single state sequence most likely to have produced the observed data. In fact, the sequence of states obtained by concatenating the states which individually have maximum posterior probabilitity at each time step, may have zero probability under the posterior. This creates the need for separate inference algorithms to find the single most likely state sequence given the observations. Such algorithms for filtering and smoothing are called *Viterbi decoding* methods (Viterbi, 1967). Why was there no need for similar decoding in the continuous state case? It turns out that due to the smooth and unimodal nature of the posterior probabilities for individual states in the continuous case (all posteriors are Gaussian), the sequence of maximum a posteriori states *is exactly* the single most likely state trajectory. So the regular Kalman filter and RTS smoothing recursions suffice. It is possible (see for example (Rabiner and Juang, 1986)) to learn the discrete state model parameters based on the results of the Viterbi decoding instead of the forward-backward smoothing — in other words to maximize the joint likelihood of the observations and the single most likely state sequence rather than the total likelihood summed over all possible paths through state space.

[14] Although harder to see. Sketch of proof: Without loss of generality, always set the covariance to the identity matrix. Next, set the dot product of the mean vector with the $k$-vector having unity in all positions to be zero since moving along this direction does not change the probabilities. Now there are $(k-1)$ degrees of freedom in the mean and also in the probability model. Set the mean randomly at first (except that it has no projection along the all unity direction). Move the mean along a line defined by the constraint that all probabilities but two should remain constant until one of those two probabilities has the desired value. Repeat this until all have been set correctly.

# 7 Independent Component Analysis

There has been a great deal of recent interest in the *blind source separation* problem which attempts to recover a number of "source" signals from observations resulting from those signals, using only the knowledge that the original sources are independent. In the "square-linear" version of the problem, the observation process is characterized entirely by a *square* and *invertible* matrix $\mathbf{C}$. In other words, there are as many observation streams as sources, and there is no delay, echo or convolutional distortion. Recent experience has shown the surprising result that for non-Gaussian distributed sources this problem can often be solved even with no prior knowledge about the sources or about $\mathbf{C}$. It is widely believed (and beginning to be proven theoretically, see MacKay, 1996) that high kurtosis source distributions are most easily separated.

We will focus on a modified, but by now classic, version due to Bell and Sejnowski (1995) and Baram and Roth (1994) of the original *independent component analysis* algorithm (Comon, 1994). Although Bell and Sejnowski derived it from an information–maximization perspective, this modified algorithm can also be obtained by defining a particular prior distribution over the components of the vector $\mathbf{x}_t$ of sources and then deriving a gradient learning rule that maximizes the likelihood of the data $\mathbf{y}_t$ in the limit of zero output noise (Amari et al., 1996; Pearlmutter and Parra, 1997; MacKay, 1996). The algorithm, originally derived for unordered data, has also been extended to modeling time series (Pearlmutter and Parra, 1997).

We now show that the generative model underlying ICA can be obtained by modifying slightly the basic model we have considered thus far. The modification is to replace the $\mathbf{WTA}[\cdot]$ nonlinearity introduced above with a general nonlinearity $g[\cdot]$ which operates componentwise on its input. Our generative model (for static data) then becomes:

$$\mathbf{x}_\bullet = g[\mathbf{w}_\bullet] \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{26a}$$

$$\mathbf{y}_\bullet = \mathbf{C}\mathbf{x}_\bullet + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{26b}$$

The role of the nonlinearity is to convert the Gaussian distributed prior for $\mathbf{w}_\bullet$ into a non-Gaussian prior for $\mathbf{x}_\bullet$. Without loss of generality we can set $\mathbf{Q} = \mathbf{I}$, since any covariance structure in $\mathbf{Q}$ can be be obtained by a linear transformation of a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ random variable, and this linear transformation can be subsumed into the nonlinearity $g[\cdot]$. Assuming that the generative nonlinearity $g[\cdot]$ is invertible and differentiable, any choice of the generative nonlinearity results in a corresponding prior distribution on each source given by the probability density function:

$$p_x(x) = \frac{\mathcal{N}(0,1)\,|_{g^{-1}(x)}}{|g'(g^{-1}(x))|}. \tag{27}$$

It is important to distinguish this generative nonlinearity from the nonlinearity found in the ICA learning rule. We call this the learning rule nonlinearity, $f(\cdot)$, and clarify the distinction between the two nonlinearities below.

Classic ICA is defined for square and invertible $\mathbf{C}$ in the limit of vanishing noise, $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon \mathbf{I}$. Under these conditions, the posterior density of $\mathbf{x}_\bullet$ given $\mathbf{y}_\bullet$ is a delta function at $\mathbf{x}_\bullet = \mathbf{C}^{-1}\mathbf{y}_\bullet$, and the ICA algorithm can be defined in terms of learning the recognition (or unmixing) weights $\mathbf{W} = \mathbf{C}^{-1}$, rather than the generative (mixing) weights $\mathbf{C}$. The gradient learning rule to increase the likelihood is:

$$\Delta \mathbf{W} \propto \mathbf{W}^{-T} + f(\mathbf{W}\mathbf{y}_\bullet)\mathbf{y}_\bullet^T \tag{28}$$

where the learning rule nonlinearity $f(\cdot)$ is the derivative of the implicit log prior: $f(x) = \frac{d \log p_x(x)}{dx}$ (MacKay, 1996). Therefore, any generative nonlinearity $g(\cdot)$ results in a non-Gaussian prior $p_x(\cdot)$, which in turn results in a nonlinearity $f(\cdot)$ in the maximum likelihood learning rule. Somewhat frustratingly from the generative models perspective, ICA is often discussed in terms of the learning rule nonlinearity without any reference to the implicit prior over the sources.

A popular choice for the ICA learning rule nonlinearity $f(\cdot)$ is the tanh function, which corresponds a heavy tailed prior over the sources (MacKay, 1996):

$$p_x(x) = \frac{1}{\pi \cosh(x)}. \tag{29}$$

From (27) we obtain a general relationship between the cumulative distribution function of the prior on the sources, $\mathrm{cdf}_x(x)$, and of the zero-mean, unit variance noise $w$:

$$\mathrm{cdf}_x(g(w)) = \mathrm{cdf}_w(w) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}(w/\sqrt{2}). \tag{30}$$

for monotonic $g$, where $\mathrm{erf}(z)$ is the error function $2/\sqrt{\pi} \int_0^z e^{-u^2} du$. This relationship can often be solved to obtain an expression for $g$. For example, if $p_x(x) = \frac{1}{\pi \cosh(x)}$ we find that setting:

$$g(w) = \ln \left( \tan \left( \frac{\pi}{4} \left( 1 + \mathrm{erf}(w/\sqrt{2}) \right) \right) \right) \tag{31}$$

causes the generative model of (26) to generate vectors $\mathbf{x}$ in which each component is distributed exactly according to $1/(\pi \cosh(x))$. This nonlinearity is shown in figure 5.

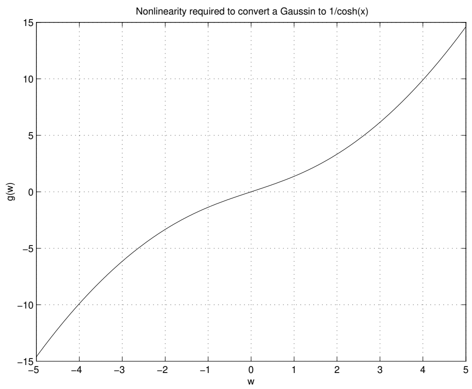


Figure 5: The nonlinearity $g(\cdot)$ from equation (31) which converts a Gaussian distributed source $w \sim \mathcal{N}(0, 1)$ into one distributed as $x = g(w) \sim 1/(\pi \cosh(x))$.

So, ICA can be seen either as a linear generative model with non-Gaussian priors for the hidden variables, or as a nonlinear generative model with Gaussian priors for the hidden variables. It is therefore possible to derive an *EM* algorithm for ICA, even when the observation noise $\mathbf{R}$ is nonzero and there are fewer sources than observations. The only complication is that the posterior distribution of $\mathbf{x}_\bullet$ given $\mathbf{y}_\bullet$ will be the product of a non-Gaussian prior and a Gaussian likelihood term, which can be difficult to evaluate. Given this posterior, the M step then consists of maximizing the expected log of the joint probability as a function of $\mathbf{C}$ and $\mathbf{R}$. The M step for $\mathbf{C}$ is

$$\mathbf{C} \leftarrow \underset{\mathbf{C}}{\arg\max} \sum_i \langle \log P(\mathbf{x}) + \log P(\mathbf{y}_i|\mathbf{x}, \mathbf{C}, \mathbf{R}) \rangle_i \tag{32}$$

where $i$ indexes the data points and $\langle \cdot \rangle_i$ denotes expectation with respect to the posterior distribution of $\mathbf{x}$ given $\mathbf{y}_i$, $P(\mathbf{x}|\mathbf{y}_i, \mathbf{C}, \mathbf{R})$. The first term does not depend on $\mathbf{C}$ and the second term is a quadratic in $\mathbf{C}$, so taking derivatives with respect to $\mathbf{C}$ we obtain a linear system of equations which can be solved in the usual manner:

$$\mathbf{C} \leftarrow \left( \sum_i \mathbf{y}_i \langle \mathbf{x}^T \rangle_i \right) \left( \sum_i \langle \mathbf{x}\mathbf{x}^T \rangle_i \right)^{-1}. \tag{33}$$

A similar M step can be derived for $\mathbf{R}$. Since, given $\mathbf{x}_\bullet$, the generative model is linear, the M step only requires evaluating *the first and second moments* of the posterior distribution of $\mathbf{x}$, $\langle \mathbf{x} \rangle_i$ and $\langle \mathbf{x}\mathbf{x}^T \rangle_i$. It is not necessary to know anything else about the posterior if its first two moments can be computed. These may be computed using Gibbs sampling or, for certain priors, using table lookup or closed form computation.[15]

Alternatively, we can compute the posterior distribution of $\mathbf{w}_\bullet$ given $\mathbf{y}_\bullet$, which is the product of a Gaussian prior and a non-Gaussian likelihood. Again, this may not be easy and we may wish to resort to Gibbs sampling. On the other hand, we can employ a trick recently introduced by Bishop et al (1997) in the context of a model called the Generative Topographic Mapping (GTM), which is a probabilistic version of Kohonen's (1982) self-organized topographic maps. We approximate the Gaussian prior via a finite number ($N$) of fixed samples (this is the trick). In other words,

$$P(\mathbf{w}_\bullet) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \approx \tilde{P}(\mathbf{w}_\bullet) = \sum_{j=1}^{N} \delta(\mathbf{w}_\bullet - \mathbf{w}_j) \tag{34}$$

where the $\mathbf{w}_j$'s are a finite sample from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The generative model then takes these $N$ points, maps them through a fixed nonlinearity $g$, an adaptable linear mapping $\mathbf{C}$, and adds Gaussian noise with covariance $\mathbf{R}$ to produce $\mathbf{y}_\bullet$. The generative model is therefore a *constrained mixture* of $N$ Gaussians, where the constraint comes from the fact that the only way the centers can move is by varying $\mathbf{C}$. Then, computing the posterior over $\mathbf{w}_\bullet$ amounts to computing the responsibility under each of the $N$ Gaussians for the data point. Given these responsibilities, the problem is again linear in $\mathbf{C}$ which means that it can be solved using (33). For the traditional zero-noise limit of ICA, the responsibilities will just select the center closest to the data point in exactly the same manner as standard vector quantization. Therefore, ICA could potentially be implemented using *EM* for Generative Topographic Mappings in the limit of zero output noise.

# 8 Network interpretations and regularization

Since early in the modern history of neural networks it was realized that principal component analysis could be implemented using a linear autoencoder network (Baldi and Hornik, 1989). The data is fed both as the input and target of the network, and the network parameters are learned using the squared error cost function. In this section we show how factor analysis and mixture of Gaussian clusters can also be implemented in this manner, albeit with a different cost function.

To understand how a probabilistic model can be learned using an autoencoder it is very useful to make a recognition/generation decomposition of the autoencoder (Hinton and Zemel, 1994; Hinton

[15] In the limit of zero noise, $\mathbf{R} = \mathbf{0}$, the *EM* updates derived in this manner degenerate to $\mathbf{C} \leftarrow \mathbf{C}$ and $\mathbf{R} \leftarrow \mathbf{R}$. Since this does not decrease the likelihood, it does not contradict the convergence proof for the *EM* algorithm. However, it also does not increase the likelihood, which might explain why no one uses *EM* to fit the standard zero-noise ICA model!

et al., 1997). An autoencoder takes an input $\mathbf{y}_\bullet$, produces some internal representation in the hidden layer $\hat{\mathbf{x}}_\bullet$ and generates at its output a reconstruction of the input $\hat{\mathbf{y}}_\bullet$ (figure 6). We call the mapping from hidden to output layers the *generative* part of the network since it generates the data from a (usually more compact) representation under some noise model. Conversely, we call the mapping from input to hidden units the *recognition* part of the network since it produces some representation in the hidden variables given the input. Since autoencoders are usually assumed to be deterministic, we will think of the recognition network as computing the posterior mean of the hidden variables given the input.

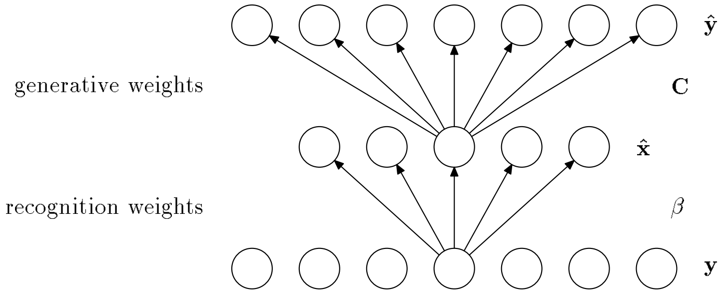


Figure 6: A network for state inference and for learning parameters of a static data model. The input $\mathbf{y}_\bullet$ is clamped to the input units (bottom), and the mean $\hat{\mathbf{x}}_\bullet$ of the posterior of the estimated state appears on the hidden units above. The covariance of the state posterior is constant at $\mathbf{I}-\boldsymbol{\beta}\mathbf{C}$ which is easily computed if the weights $\boldsymbol{\beta}$ are known. The inference computation is a trivial linear projection, but learning the weights of the inference network is difficult. The input to hidden weights are always constrained to be a function of the hidden to output weights, and the network is trained as an autoencoder using self-supervised learning. Outgoing weights have only been drawn from one input unit and one hidden unit.

The generative model for factor analysis assumes that both the hidden states and the observables are normally distributed, from which we get the posterior probabilities for the hidden states in equation (19b). If we assume that the generative weight matrix from the hidden units to the outputs is $\mathbf{C}$ and the noise model at the output is Gaussian with covariance $\mathbf{R}$, then, the posterior mean of the hidden variables is $\hat{\mathbf{x}}_\bullet = \boldsymbol{\beta}\mathbf{y}_\bullet$, where $\boldsymbol{\beta} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \mathbf{R})^{-1}$. Therefore, the hidden units can compute the posterior mean exactly if they are linear and the weight matrix from input to hidden units is $\boldsymbol{\beta}$. Notice that $\boldsymbol{\beta}$ is tied to $\mathbf{C}$ and $\mathbf{R}$, so we only need to estimate $\mathbf{C}$ and $\mathbf{R}$ during learning. We denote expectations under the posterior state distribution by $\langle\cdot\rangle$, e.g.

$$\langle \mathbf{x}_\bullet \rangle = \int \mathbf{x}_\bullet P(\mathbf{x}_\bullet | \mathbf{y}_\bullet)\, d\mathbf{x}_\bullet = \hat{\mathbf{x}}_\bullet.$$

From the theory of the *EM* algorithm (see section 4.2), we know that one way to maximize the likelihood is to maximize the expected value of the log of the joint probability under the posterior distribution of the hidden variables:

$$\langle \log P(\mathbf{x}_\bullet, \mathbf{y}_\bullet | \mathbf{C}, \mathbf{R}) \rangle$$

Changing signs and ignoring constants we can equivalently minimize the following cost function:

$$\begin{aligned}
\mathcal{C} &= \left\langle (\mathbf{y}_\bullet - \mathbf{C}\mathbf{x}_\bullet)^T \mathbf{R}^{-1} (\mathbf{y}_\bullet - \mathbf{C}\mathbf{x}_\bullet) \right\rangle + \log|\mathbf{R}| && \text{(35a)} \\
&= \mathbf{y}_\bullet^T \mathbf{R}^{-1}\mathbf{y}_\bullet - 2\mathbf{y}_\bullet^T \mathbf{R}^{-1}\mathbf{C}\langle \mathbf{x}_\bullet \rangle + \langle \mathbf{x}_\bullet^T \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C}\mathbf{x}_\bullet \rangle + \log|\mathbf{R}| && \text{(35b)} \\
&= (\mathbf{y}_\bullet - \mathbf{C}\hat{\mathbf{x}}_\bullet)^T \mathbf{R}^{-1} (\mathbf{y}_\bullet - \mathbf{C}\hat{\mathbf{x}}_\bullet) + \log|\mathbf{R}| + \mathrm{trace}[\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\boldsymbol{\Sigma}]. && \text{(35c)}
\end{aligned}$$

Here we have defined $\mathbf{\Sigma}$ to be the posterior covariance of $\mathbf{x}_\bullet$,

$$\mathbf{\Sigma} \equiv \langle \mathbf{x}_\bullet \mathbf{x}_\bullet^T \rangle - \langle \mathbf{x}_\bullet \rangle \langle \mathbf{x}_\bullet \rangle^T = \mathbf{I} - \boldsymbol{\beta}\mathbf{C}$$

and in the last step we have reorganized terms and made use of the fact that $\langle \mathbf{x}_\bullet^T \mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} \mathbf{x}_\bullet \rangle = \text{trace}[\mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} \langle \mathbf{x}_\bullet \mathbf{x}_\bullet^T \rangle]$.

The first two terms of cost function (35c) are just a squared error cost function evaluated with respect to a Gaussian noise model with covariance $\mathbf{R}$—they are exactly the terms minimized when fitting a standard neural network with this Gaussian noise model. The last term is a regularization term that accounts for the posterior variance in the hidden states given the inputs.[16] When we take derivatives of this cost function, we do not differentiate $\hat{\mathbf{x}}$ and $\mathbf{\Sigma}$ with respect to $\mathbf{C}$ and $\mathbf{R}$. As is usual for the *EM* algorithm we differentiate the cost *given* the posterior distribution of the hidden variables. Taking derivatives with respect to $\mathbf{C}$ and premultipling by $\mathbf{R}$ we obtain a weight change rule

$$\Delta \mathbf{C} \propto (\mathbf{y}_\bullet - \mathbf{C}\mathbf{x}_\bullet)\mathbf{x}_\bullet^T - \mathbf{C}\mathbf{\Sigma}. \tag{36}$$

The first term is the usual delta rule. The second term is simply a weight-decay term decaying the columns of $\mathbf{C}$ with respect to the posterior covariance of the hidden variables. Intuitively, the higher the uncertainty in a hidden unit, the more its outgoing weight vector is shrunk towards zero. To summarize, factor analysis can be implemented in an autoassociator by tying the recognition weights to the generative weights and using a particular regularizer in addition to squared reconstruction error during learning.

We now analyze the mixture of Gaussians model in the same manner. The recognition network is meant to produce the mean of the hidden variable given the inputs. Since we assume that the discrete hidden variable is represented as a unit vector, its mean is just the vector of probabilities of being in each of its $k$ settings given the inputs, i.e. the responsibilities. Assuming equal mixing coefficients, $P(\mathbf{x}_\bullet = \mathbf{e}_j) = P(\mathbf{x}_\bullet = \mathbf{e}_i) \forall ij$, the responsibilities defined in (25b) are

$$\hat{\mathbf{x}}_\bullet = P(\mathbf{x}_\bullet = \mathbf{e}_j | \mathbf{y}_\bullet) = \frac{\exp\{-\frac{1}{2}(\mathbf{y}_\bullet - \mathbf{C}_j)^T \mathbf{R}^{-1} (\mathbf{y}_\bullet - \mathbf{C}_j)\}}{\sum_{i=1}^k \exp\{-\frac{1}{2}(\mathbf{y}_\bullet - \mathbf{C}_i)^T \mathbf{R}^{-1} (\mathbf{y}_\bullet - \mathbf{C}_i)\}} \tag{37a}$$

$$= \frac{\exp\{\boldsymbol{\beta}_j \mathbf{y}_\bullet - \boldsymbol{\alpha}_j\}}{\sum_{i=1}^k \exp\{\boldsymbol{\beta}_i \mathbf{y}_\bullet - \boldsymbol{\alpha}_i\}} \tag{37b}$$

where we have defined $\boldsymbol{\beta}_j = \mathbf{C}_j \mathbf{R}^{-1}$ and $\boldsymbol{\alpha}_j = \frac{1}{2}\mathbf{C}_j^T \mathbf{R}^{-1} \mathbf{C}_j$. Equation (37b) describes a recognition model which is linear followed by the softmax nonlinearity, $\Phi$, written in full matrix notation: $\hat{\mathbf{x}}_\bullet = \Phi(\boldsymbol{\beta}\mathbf{y}_\bullet - \boldsymbol{\alpha})$. In other words, a simple network could do exact inference with linear input to hidden weights $\boldsymbol{\beta}$ and softmax hidden units.

Appealing again to the *EM* algorithm we obtain a cost function which when minimized by an autoassociator will implement the mixture of Gaussians.[17] The log probability of the data given the hidden variables can be written as

$$-2 \log P(\mathbf{y}_\bullet | \mathbf{x}_\bullet, \mathbf{C}, \mathbf{R}) = (\mathbf{y}_\bullet - \mathbf{C}\mathbf{x}_\bullet)^T \mathbf{R}^{-1} (\mathbf{y}_\bullet - \mathbf{C}\mathbf{x}_\bullet) + \log |\mathbf{R}| + const.$$

[16] PCA assumes infinitesimal noise and therefore the posterior "distribution" over the hidden states has zero variance ($\mathbf{\Sigma} \to \mathbf{0}$) and the regularizer vanishes ($\mathbf{C}^T \mathbf{R}^{-1} \mathbf{C} \mathbf{\Sigma} \to \mathbf{0}$).

[17] Our derivation assumes tied covariance and equal mixing coefficients. Slightly more complex equations result for the general case.

Using this and the previous derivation, we obtain the cost function:

$$\mathcal{C} = (\mathbf{y}_\bullet - \mathbf{C}\hat{\mathbf{x}}_\bullet)^T \mathbf{R}^{-1} (\mathbf{y}_\bullet - \mathbf{C}\hat{\mathbf{x}}_\bullet) + \log|\mathbf{R}| + \mathrm{trace}[\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\boldsymbol{\Sigma}] \tag{38}$$

where $\boldsymbol{\Sigma} = \langle \mathbf{x}_\bullet \mathbf{x}_\bullet^T \rangle - \langle \mathbf{x}_\bullet \rangle \langle \mathbf{x}_\bullet \rangle^T$. The second order term, $\langle \mathbf{x}_\bullet \mathbf{x}_\bullet^T \rangle$, evaluates to a matrix with $\hat{\mathbf{x}}_\bullet$ along its diagonal and zero elsewhere. Notice that, unlike in factor analysis, $\boldsymbol{\Sigma}$ now depends on the input.

To summarize, the mixture of Gaussians model can also be implemented using an autoassociator. The recognition part of the network is linear, followed by a softmax nonlinearity. The cost function is the usual squared error penalized by a regularizer of exactly the same form as in factor analysis. Similar network interpretations can be obtained for the other probabilistic models.

# 9 Comments and Extensions

There are several advantages, both theoretical and practical, to a unified treatment of the unsupervised methods reviewed in this note. From a theoretical viewpoint the treatment emphasizes that all of the techniques for inference in the various models are essentially the same and just correspond to probability propagation in the particular model variation. Similarly, all the learning procedures are nothing more than an application of the *EM* algorithm to iteratively increase the total likelihood of the observed data. Furthermore, the origin of zero noise limit algorithms such as VQ and PCA are easy to see. A unified treatment also highlights the relationship between similar questions across the different models. For example, picking the number of clusters in a mixture model or state dimension in a dynamical system or the number of factors or principal components in a covariance model or the number of states in a hidden Markov model are all really the same question.

From a practical standpoint, a unified view of these models allows us to apply well known solutions to hard problems in one area to similar problems in another. For example, in this framework it is obvious how to deal properly with missing data in solving both the learning and inference problems. This topic has been well understood for many static models (Little and Rubin, 1987; Tresp et al., 1994; Ghahramani and Jordan, 1994) but is typically not well addressed in the linear dynamical systems literature. As another example, it is easy to design and work with models having a mixed continuous and discrete state vector (such as for example Hidden Filter HMMs (Fraser and Dimitriadis, 1993)) which is something not directly addressed by the individual literatures on discrete or continuous models.

Another practical advantage is the ease with which natural extensions to the basic models can be developed. For example, using the hierarchical mixture of experts formalism developed by Jordan and Jacobs (1994) we can consider global mixtures of any of the model variants discussed. In fact, most of these mixtures have already been considered: mixtures of linear dynamical systems are known as *switching state-space models* (see (Shumway and Stoffer, 1991; Ghahramani and Hinton, 1996b) and references therein); mixtures of factor analyzers (Ghahramani and Hinton, 1997) and of pancakes (PCA) (Hinton et al., 1995); mixtures of hidden Markov models (Smyth, 1997). A mixture of $m$ of our constrained mixtures of Gaussians each with $k$ clusters gives a mixture model with $mk$ components in which there are only $m$ possible covariance matrices. This "tied covariance" approach is popular in speech modeling to reduce the number of free parameters. (For $k = 1$ this corresponds to a full "unconstrained" mixture of Gaussians model with $m$ clusters.)

It is also possible to consider "local mixtures" in which the conditional probability $P(\mathbf{y}_t|\mathbf{x}_t)$ is no longer a single Gaussian but a more complicated density such as a mixture of Gaussians. For our (constrained) mixture of Gaussians model this is another way to get a "full" mixture. For hidden Markov models this is a well known extension and is usually the standard approach for emission density modeling (Rabiner and Juang, 1986). However we are not aware of any work which considers this variation in the continuous state cases, either for static or dynamic data.

Another important natural extension is *spatially adaptive observation noise*. The idea here is that the observation noise $\mathbf{v}$ can have different statistics in different parts of (state or observation) space rather than being described by a single matrix $\mathbf{R}$. For discrete mixture models, this idea is well known and it achieved by giving each mixture component a private noise model. However, for continuous state models this idea is unexplored, and is an interesting area for further investigation. The crux of the problem is how to parameterize a positive definite matrix over some space. We propose some simple ways to achieve this. One possibility is replacing the single covariance shape $\mathbf{Q}$ for the observation noise with a *conic*[18] linear blending of $k$ "basis" covariance shapes. In the case of linear dynamical systems (Roweis, 1997b) or factor analysis this amounts to a novel type of model in which the local covariance matrix $\mathbf{R}$ is computed as a conic linear combination of several "canonical" covariance matrices through a tensor product between the current state vector $\mathbf{x}$ (or equivalently the "noiseless" observation $\mathbf{Cx}$) and a master noise tensor $\mathcal{R}$[19]. Another approach would be to drop the conic restriction (allow general linear combinations) and then add a multiple of the identity matrix to the resulting noise matrix in order to make it positive definite. A third approach is to represent the covariance shape as the compression of an elastic sphere due to a spatially varying force field. This representation is easier to work with because the parameterization of the field is unconstrained, but is hard learn the local field from measurements of the effective covariance shape.

It is also interesting to consider what happens to the *dynamic* models when the output noise tends to zero. In other words, what are the dynamic analogues of PCA and VQ? For both linear dynamical systems and hidden Markov models this causes the state to no longer be hidden. In linear dynamical systems the optimal observation matrix is then found by performing PCA on the data and using the principal components as the columns of $\mathbf{C}$; for hidden Markov models $\mathbf{C}$ is found by vector quantization of the data (using the codebook vectors as the columns of $\mathbf{C}$). Given these observation matrices the state is no longer hidden. All that remains is to identify a first-order Markov dynamics in state space: this is a simple AR(1) model in the continuous case or a first order Markov chain in the discrete case. Such zero-noise limits are not only interesting models in their own right, but are also valuable as good choices for initialization of learning in linear dynamical systems and hidden Markov models.


## Acknowledgments

We thank Carlos Brody, Sanjoy Mahajan and Erik Winfree for many fruitful discussions in the early stages, and Geoffrey Hinton and John Hopfield for providing outstanding intellectual environments and guidance. SR was supported in part by the Center for Neuromorphic Systems Engineering as a part of the National Science Foundation Engineering Research Center Program under grant EEC-9402726 and by the Natural Sciences and Engineering Research Council of Canada under an NSERC 1967 Award. ZG was supported by the Ontario Information Technology Research Centre.


[18] A conic linear combination is one in which all the coefficients are positive.

[19] For mixtures of Gaussians or hidden Markov models, this kind of linear "blending" merely selects the $j^{th}$ submatrix of the tensor if the discrete state is $\mathbf{e}_j$. This is yet another way to recover the conventional "full" or unconstrained mixture of Gaussians or hidden Markov model emission density in which each cluster or state has its own private covariance shape for observation noise.

# A Algorithms

In this appendix we review in detail the inference and learning algorithms for each of the models. The goal is to enable the reader to actually implement these algorithms from the pseudo-code provided. For each class of model, we first present the solution to the inference problem, and then the *EM* algorithm for learning the model parameters. **NB**: for this appendix only, we adopt the notation that the transpose of a vector or matrix is written as $\mathbf{x}'$ not $\mathbf{x}^T$. We also define the binary operator $\odot$ to be the element-wise product of two equal-size vectors or matrices.

## A.1 Factor Analysis, SPCA and PCA

### A.1.1 Inference

For FA and related models, the posterior probability of the hidden state given the observations, $P(\mathbf{x}_\bullet|\mathbf{y}_\bullet)$, is Gaussian. The inference problem therefore consists of computing the mean and covariance of this Gaussian, $\hat{\mathbf{x}}_\bullet$ and $\mathbf{V} = \mathrm{Cov}[\mathbf{x}_\bullet]$:

FactorAnalysisInference($\mathbf{y}_\bullet$,$\mathbf{C}$,$\mathbf{R}$)
  $\boldsymbol{\beta} \leftarrow \mathbf{C}'(\mathbf{C}\mathbf{C}' + \mathbf{R})^{-1}$
  $\hat{\mathbf{x}}_\bullet \leftarrow \boldsymbol{\beta}\mathbf{y}_\bullet$
  $\mathbf{V} \leftarrow \mathbf{I} - \boldsymbol{\beta}\mathbf{C}$
  return $\hat{\mathbf{x}}_\bullet$, $\mathbf{V}$

Since the observation noise matrix $\mathbf{R}$ is assumed to be diagonal and $\mathbf{x}_\bullet$ is of smaller dimension than $\mathbf{y}_\bullet$, $\boldsymbol{\beta}$ can be more efficiently computed using the matrix inversion lemma:

$$\boldsymbol{\beta} = \mathbf{C}'\mathbf{R}^{-1}\left(\mathbf{I} - \mathbf{C}(\mathbf{I} + \mathbf{C}'\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{C}'\mathbf{R}^{-1}\right).$$

The sensible PCA algorithm (SPCA) is a special case of factor analysis in which the observation noise is assumed to be spherically symmetric: $\mathbf{R} = \alpha\mathbf{I}$. Inference in SPCA is therefore identical to inference for factor analysis.

The traditional PCA algorithm can be obtained as a limiting case of factor analysis: $\mathbf{R} = \lim_{\epsilon\to 0}\epsilon\mathbf{I}$. The inverse used for computing $\boldsymbol{\beta}$ in factor analysis is no longer well-defined. However, the limit of $\boldsymbol{\beta}$ is well-defined: $\boldsymbol{\beta} = (\mathbf{C}'\mathbf{C})^{-1}\mathbf{C}'$. Also, the posterior collapses to a single point, so $\mathbf{V} = \mathrm{Cov}[\mathbf{x}_\bullet] = \mathbf{I} - \boldsymbol{\beta}\mathbf{C} = \mathbf{0}$.

PCAInference($\mathbf{y}_\bullet$,$\mathbf{C}$) *% Projection onto principal components*
  $\boldsymbol{\beta} \leftarrow (\mathbf{C}'\mathbf{C})^{-1}\mathbf{C}'$
  $\hat{\mathbf{x}}_\bullet \leftarrow \boldsymbol{\beta}\mathbf{y}_\bullet$
  return $\hat{\mathbf{x}}_\bullet$

### A.1.2 Learning

The *EM* algorithm for learning the parameters of a factor analyzer with $k$ factors from a zero-mean data set $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ is

```
FactorAnalysisLearn(Y,k,ε)
  initialize C, R
  compute sample covariance S of Y
  while change in log likelihood > ε
      % E step
      X̂, V ← FactorAnalysisInference(Y,C,R)
      δ ← YX̂'
      γ ← X̂X̂' + nV
      % M step
      C ← δγ^{-1}
      set diagonal elements of R to R_pp ← (S − Cδ'/n)_pp
  end
  return C, R
```

Here FactorAnalysisInference($\mathbf{Y}$,$\mathbf{C}$,$\mathbf{R}$) has the obvious interpretation of the inference function applied to the entire matrix of observations—since $\boldsymbol{\beta}$ and $\mathbf{V}$ are do not depend on $\mathbf{Y}$, this can be computed efficiently in matrix form. Note that since the data appears only in outer products, we can run factor analysis learning with just the sample covariance. Note also that the log likelihood is computed as $-\frac{1}{2}\mathbf{y}'(\mathbf{C}\mathbf{C}'+\mathbf{R})^{-1}\mathbf{y}+\frac{n}{2}\log|\mathbf{C}\mathbf{C}'+\mathbf{R}|+const.$

The *EM* algorithm for SPCA is identical to the *EM* algorithm for factor analysis, except that since the observation noise covariance is spherically symmetrical, the M step for $\mathbf{R}$ is changed to $\mathbf{R}\leftarrow\alpha\mathbf{I}$, where $\alpha\leftarrow\sum_{j=1}^{p}(\mathbf{S}-\mathbf{C}\boldsymbol{\delta}')_{jj}/p$.

The *EM* algorithm for PCA can be obtained in a similar manner:

```
PCALearn(Y,k,ε)
  initialize C
  while change in squared reconstruction error > ε
      % E step
      X̂ ← PCAInference(Y,C)
      δ ← YX̂'
      γ ← X̂X̂'
      % M step
      C ← δγ^{-1}
  end
  return C
```

Note that since PCA is not a probability model (i.e. it assumes zero noise) the likelihood is undefined, so convergence is assessed by monitoring the squared reconstruction error.

## A.2 Mixtures of Gaussians and Vector Quantization

### A.2.1 Inference

We begin by discussing the inference problem for mixtures of Gaussians and then discuss the inference problem in vector quantization (VQ) as a limiting case. The hidden variable in a mixture of Gaussians is a discrete variable that can take on one of $k$ values. We represent this variable using a vector $\mathbf{x}_\bullet$ of length $k$, where each setting of the hidden variable corresponds to $\mathbf{x}_\bullet$ taking a value of unity in one dimension and zero elsewhere. The probability distribution of the discrete hidden variable, which has $k-1$ degrees of freedom (since it must sum to one), is fully described by the mean of $\mathbf{x}_\bullet$. Therefore, the inference problem is limited to computing the posterior mean of $\mathbf{x}_\bullet$ given a data point $\mathbf{y}_\bullet$ and the model parameters, which are $\boldsymbol{\pi}$ (the prior mean of $\mathbf{x}_\bullet$), $\mathbf{C}$ (the matrix whose $k$ columns are the means of $\mathbf{y}_\bullet$ given each of the $k$ settings of $\mathbf{x}_\bullet$) and $\mathbf{R}$ (the observation noise covariance matrix).

`MixtureofGaussiansInference(`$\mathbf{y}_\bullet$`,`$\mathbf{C}$`,`$\mathbf{R}$`,`$\boldsymbol{\pi}$`)`
  $\alpha \leftarrow 0$
  `for` $i = 1$ `to` $k$
    $\boldsymbol{\delta}_i \leftarrow (\mathbf{y}_\bullet - \mathbf{C}_i)'\mathbf{R}^{-1}(\mathbf{y}_\bullet - \mathbf{C}_i)$
    $\gamma_i \leftarrow \boldsymbol{\pi}_i \exp\left\{-\frac{1}{2}\boldsymbol{\delta}_i\right\}$
    $\alpha \leftarrow \alpha + \gamma_i$
  `end`
  $\hat{\mathbf{x}}_\bullet \leftarrow \gamma/\alpha$
  `return` $\hat{\mathbf{x}}_\bullet$

A measure of the randomness of the hidden state can be obtained by evaluating the entropy of the discrete distribution corresponding to $\hat{\mathbf{x}}_\bullet$.

Standard VQ corresponds to the limiting case $\mathbf{R} = \lim_{\epsilon\to 0}\epsilon\mathbf{I}$ and equal priors $\boldsymbol{\pi}_i = 1/k$. Inference in this case is performed by the well known (1-)nearest-neighbour rule.

`VQInference(`$\mathbf{y}_\bullet$`,`$\mathbf{C}$`)` *% 1-nearest-neighbour*
  `for` $i = 1$ `to` $k$
    $\boldsymbol{\delta}_i \leftarrow (\mathbf{y}_\bullet - \mathbf{C}_i)'(\mathbf{y}_\bullet - \mathbf{C}_i)$
  `end`
  $\hat{\mathbf{x}}_\bullet \leftarrow \mathbf{e}_j$ `if` $j = \arg\min \boldsymbol{\delta}$
  `return` $\hat{\mathbf{x}}_\bullet$

As before, $\mathbf{e}_j$ is the unit vector along the $j^{th}$ coordinate direction. The squared distances $\boldsymbol{\delta}_i$ can be generalized to some Mahalanobis metric with respect to some matrix $\mathbf{R}$, and nonuniform priors $\boldsymbol{\pi}_i$ can easily be incorporated. As was the case with PCA, the posterior distribution has zero entropy.

### A.2.2 Learning

The *EM* algorithm for learning the parameters of a mixture of Gaussian is

`MixtureofGaussiansLearn(`$\mathbf{Y}$`,`$k$`,`$\epsilon$`)`
  `initialize` $\mathbf{C}$, $\mathbf{R}$, $\boldsymbol{\pi}$
  `while change in log likelihood` $> \epsilon$
    `initialize` $\boldsymbol{\delta} \leftarrow \mathbf{0}$, $\gamma \leftarrow \mathbf{0}$, $\boldsymbol{\alpha} \leftarrow \mathbf{0}$
    *% E step*
    `for` $i$ `=` `1` `to` $n$

```
      x_i ← MixtureofGaussiansInference(y_i,C,R,π)
      δ ← δ + y_i x_i'
      γ ← γ + x_i
    end
    % M step
    for j = 1 to k
      C_j ← δ_j/γ_j
      for i = 1 to n
        α ← α + x_ij (y_i − C_j)(y_i − C_j)'
      end
    end
    R ← α/n
    π ← γ/n
  end
  return C, R, π
```

We have assumed a common covariance matrix $\mathbf{R}$ for all the Gaussians; the extension to different covariances for each Gaussian is straightforward.

The *k-means* vector quantization learning algorithm results when we take the appropriate limit of the above algorithm.

```
VQLearn(Y,k,ε) % k-means
  initialize C
  while change in squared reconstruction error > ε
    % E step
    initialize δ ← 0, γ ← 0
    for i = 1 to n
      x_i ← VQInference(y_i,C)
      δ ← δ + y_i x_i'
      γ ← γ + x_i
    end
    % M step
    for j = 1 to k
      C_j ← δ_j/γ_j
    end
  end
  return C
```

## A.3 Linear Dynamical Systems

### A.3.1 Inference

Inference in linear dynamical system involves computing the posterior distributions of the hidden state variables given the sequence of observations. Like in factor analysis, all the hidden state variables are assumed Gaussian, and are therefore fully described by their means and covariance matrices. The algorithms for computing the posterior means and covariances consists of two parts: a forward recursion which uses the observations from $\mathbf{y}_1$ to $\mathbf{y}_t$, known as the *Kalman filter* (Kalman, 1960), and a backward recursion which uses the observations from $\mathbf{y}_\tau$ to $\mathbf{y}_{t+1}$ (Rauch, 1963). The combined forward and backward recursions are known as the Kalman or Rauch-Tung-Streibel (RTS) smoother.

To describe the smoothing algorithm it will be useful to define the following quantities: $\mathbf{x}_t^s$ and $\mathbf{V}_t^s$ are respectively the mean and covariance matrix of $\mathbf{x}_t$ given observations $\{\mathbf{y}_1, \ldots \mathbf{y}_s\}$; $\hat{\mathbf{x}}_t \equiv \mathbf{x}_t^\tau$ and $\hat{\mathbf{V}}_t \equiv \mathbf{V}_t^\tau$ are the "full smoother" estimates. To learn the $\mathbf{A}$ matrix using *EM* it is also necessary to compute the covariance across time between $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$.

`LDSInference(`$\mathbf{Y}$,$\mathbf{A}$,$\mathbf{C}$,$\mathbf{Q}$,$\mathbf{R}$,$\mathbf{x}_1^0$,$\mathbf{V}_1^0$`)` *% Kalman smoother*
  `for` $t = 1$ `to` $T$ *% Kalman filter (forward pass)*
    $\mathbf{x}_t^{t-1} \leftarrow \mathbf{A}\mathbf{x}_{t-1}^{t-1}$ `if` $t > 1$
    $\mathbf{V}_t^{t-1} \leftarrow \mathbf{A}\mathbf{V}_{t-1}^{t-1}\mathbf{A}' + \mathbf{Q}$ `if` $t > 1$
    $\mathbf{K}_t \leftarrow \mathbf{V}_t^{t-1}\mathbf{C}'(\mathbf{C}\mathbf{V}_t^{t-1}\mathbf{C}' + \mathbf{R})^{-1}$
    $\mathbf{x}_t^t \leftarrow \mathbf{x}_t^{t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t^{t-1})$
    $\mathbf{V}_t^t \leftarrow \mathbf{V}_t^{t-1} - \mathbf{K}_t\mathbf{C}\mathbf{V}_t^{t-1}$
  `end`
  `initialize` $\hat{\mathbf{V}}_{T,T-1} = (\mathbf{I} - \mathbf{K}_\tau\mathbf{C})\mathbf{A}\mathbf{V}_{T-1}^{T-1}$
  `for` $t = T$ `to 2` *% Rauch recursions (backward pass)*
    $\mathbf{J}_{t-1} \leftarrow \mathbf{V}_{t-1}^{t-1}\mathbf{A}'(\mathbf{V}_t^{t-1})^{-1}$
    $\hat{\mathbf{x}}_{t-1} \leftarrow \mathbf{x}_{t-1}^{t-1} + \mathbf{J}_{t-1}(\hat{\mathbf{x}}_t - \mathbf{A}\mathbf{x}_{t-1}^{t-1})$
    $\hat{\mathbf{V}}_{t-1} \leftarrow \mathbf{V}_{t-1}^{t-1} + \mathbf{J}_{t-1}(\hat{\mathbf{V}}_t - \mathbf{V}_t^{t-1})\mathbf{J}_{t-1}'$
    $\hat{\mathbf{V}}_{t,t-1} \leftarrow \mathbf{V}_t^t\mathbf{J}_{t-1}' + \mathbf{J}_t(\hat{\mathbf{V}}_{t+1,t} - \mathbf{A}\mathbf{V}_t^t)\mathbf{J}_{t-1}'$ `if` $t < \tau$
  `end`
  `return` $\hat{\mathbf{x}}_t$, $\hat{\mathbf{V}}_t$, $\hat{\mathbf{V}}_{t,t-1}$ `for all` $t$

### A.3.2 Learning

The *EM* algorithm for learning a linear dynamical system is (Shumway and Stoffer, 1982; Ghahramani and Hinton, 1996a), assuming for simplicity that we only have a single sequence of observations:

`LDSLearn(`$\mathbf{Y}$,$k$,$\epsilon$`)`
  `initialize` $\mathbf{A}$, $\mathbf{C}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{x}_1^0$, $\mathbf{V}_1^0$
  `set` $\boldsymbol{\alpha} \leftarrow \sum_t \mathbf{y}_t\mathbf{y}_t'$
  `while change in log likelihood` $> \epsilon$
    `LDSInference(`$\mathbf{Y}$,$\mathbf{A}$,$\mathbf{C}$,$\mathbf{Q}$,$\mathbf{R}$,$\mathbf{x}_1^0$,$\mathbf{V}_1^0$`)` *% E step*
    `initialize` $\boldsymbol{\delta} \leftarrow \mathbf{0}$, $\gamma \leftarrow \mathbf{0}$, $\boldsymbol{\beta} \leftarrow \mathbf{0}$
    `for` $t = 1$ `to` $\tau$
      $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + \mathbf{y}_t\hat{\mathbf{x}}_t'$
      $\gamma \leftarrow \gamma + \hat{\mathbf{x}}_t\hat{\mathbf{x}}_t' + \hat{\mathbf{V}}_t$

$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \hat{\mathbf{x}}_t \hat{\mathbf{x}}'_{t-1} + \hat{\mathbf{V}}_{t,t-1}$ `if` $t > 1$
`end`
$\gamma_1 \leftarrow \boldsymbol{\gamma} - \hat{\mathbf{x}}_\tau \hat{\mathbf{x}}'_\tau - \hat{\mathbf{V}}_\tau$
$\gamma_2 \leftarrow \boldsymbol{\gamma} - \hat{\mathbf{x}}_1 \hat{\mathbf{x}}'_1 - \hat{\mathbf{V}}_1$
*% M step*
$\mathbf{C} \leftarrow \boldsymbol{\delta}\boldsymbol{\gamma}^{-1}$
$\mathbf{R} \leftarrow (\boldsymbol{\alpha} - \mathbf{C}\boldsymbol{\delta}')/T$
$\mathbf{A} \leftarrow \boldsymbol{\beta}\gamma_1^{-1}$
$\mathbf{Q} \leftarrow (\gamma_2 - \mathbf{A}\boldsymbol{\beta}')/(T-1)$
$\mathbf{x}_1^0 \leftarrow \hat{\mathbf{x}}_1$
$\mathbf{V}_1^0 \leftarrow \hat{\mathbf{V}}_1$
`end`
`return` $\mathbf{A}$, $\mathbf{C}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{x}_1^0$, $\mathbf{V}_1^0$

## A.4 Hidden Markov Models

### A.4.1 Inference

The *forward–backward algorithm* computes the posterior probabilities of the hidden states in an HMM, and therefore forms the basis of the the inference required for *EM*. We present the algorithm using the following standard definitions:

$$\boldsymbol{\alpha}_t = P(\mathbf{x}_t, \mathbf{y}_1, \ldots, \mathbf{y}_t) \tag{39}$$
$$\boldsymbol{\beta}_t = P(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_\tau | \mathbf{x}_t), \tag{40}$$

where both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of the same length as $\mathbf{x}$. We present the algorithm for the case where the observations $\mathbf{y}_t$ are real-valued $p$-dimensional vectors and the probability of the observation given the data is assumed to be Gaussian with mean $\mathbf{C}\mathbf{x}_t$ and covariance $\mathbf{R}$. The parameters of the model are therefore: a $k \times k$ transition matrix $\mathbf{T}$, initial state prior probability vector $\boldsymbol{\pi}$, an observation mean matrix $\mathbf{C}$, and an observation noise matrix $\mathbf{R}$ which is tied across states.

`HMMInference(`$\mathbf{Y}$`,`$\mathbf{T}$`,`$\boldsymbol{\pi}$`,`$\mathbf{C}$`,`$\mathbf{R}$`)` *% forward–backward algorithm*
`for` $t = 1$ `to` $T$ *% forward pass*
`for` $i = 1$ `to` $k$
$\mathbf{b}_{t,i} \leftarrow \exp\left\{-\frac{1}{2}(\mathbf{y}_t - \mathbf{C}_i)'\mathbf{R}^{-1}(\mathbf{y}_t - \mathbf{C}_i)\right\} \; |R|^{-1/2}(2\pi)^{-p/2}$
`end`
`if` $t = 1$
$\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\pi} \odot \mathbf{b}_1$
`else`
$\boldsymbol{\alpha}_t \leftarrow [\mathbf{T}'\boldsymbol{\alpha}_{t-1}] \odot \mathbf{b}_t$
`end`
$\rho_t \leftarrow \sum_i \boldsymbol{\alpha}_{t,i}$
$\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_t / \rho_t$
`end`

```
β_T ← 1/ρ_T
for t = T − 1 to 1   % backward pass
     β_t ← T [β_{t+1} ⊙ b_{t+1}]/ρ_t
end
γ_t ← α_t ⊙ β_t/(α'_t β_t)
ξ_t ← α_t[β_{t+1} ⊙ b_{t+1}]'
ξ_t ← ξ_t/(∑_{ij} ξ_{t,i,j})
return γ_t, ξ_t, ρ_t for all t
```

The definitions of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in equations (39) and (40) correspond to running the above algorithm *without* the scaling factors $\rho_t$. These factors, however, are essential to the numerical stability of the algorithm; otherwise, for long sequences both $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ become vanishingly small. Furthermore, from the $\rho$'s we can compute the log likelihood of the sequence

$$\log P(\mathbf{y}_1, \ldots, \mathbf{y}_\tau) = \sum_{t=1}^{\tau} \log \rho_t$$

which is why it is useful for the above function to return them.

### A.4.2 Learning

Again, we assume for simplicity that we have a single sequence of observations from which we wish to learn the parameters of a hidden Markov model. The *EM* algorithm for learning these parameters, known as the Baum-Welsh algorithm, is:

```
HMMLearn(Y,k,ε)  % Baum-Welch
  initialize T, π, C, R
  while change in log likelihood > ε
      HMMInference(Y,T,π,C,R)  % E step
      % M step
      π ← γ_1
      T ← ∑_t ξ_t
      T_ij ← T_ij/∑_ℓ T_iℓ for all i, j
      C_j ← ∑_t γ_{t,j} y_t/∑_t γ_{t,j} for all j
      R ← ∑_{t,j} γ_{t,j}(y_t − C_j)(y_t − C_j)'/n
  return T, π, C, R
```

## References


Amari, S., Cichocki, A., and Yang, H. 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*, volume 8, pages 757–763. The MIT Press.

Baldi, P. and Hornik, K. (1989). Neural networks and principal components analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58.

Baram, Y. and Roth, Z. (1994). Density shaping by neural networks with application to classification, estimation and forecasting. Technical Report TR-CIS-94-20, Center for Intelligent Systems, Technion, Israel Institute for Technology, Haifa, Israel.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8.

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of American Mathematical Society*, 73:360–363.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 41.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1):164–171.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159.

Bishop, C. M., Svensen, M., and Williams, C. K. I. (In Press). GTM: A principled alternative to the self-organizing map. *Neural Computation*.

Comon, P. (1994). Independent component analysis: A new concept. *Signal Processing*, 36:287–314.

Delyon, B. (1993). Remarks on filtering of semi-Markov data. Technical Report 733, Institute de Recherche en Informatique et Systems Aleatiores, Campus de Beaulieu – 35042 Rennes Cedex – France.

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

Digalakis, V., Rohlicek, J. R., and Ostendorf, M. (1993). ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley, New York.

Elliott, R. J., Aggoun, L., and Moore, J. B. (1995). *Hidden Markov Models: Estimation and Control*, volume 29 of *Applications of Mathematics*. Springer-Verlag, New York Berlin Heidelberg.

Everitt, B. S. (1984). *An Introducction to Latent Variable Models*. Chapman and Hill, London and New York.

Fraser, A. M. and Dimitriadis, A. (1993). Forecasting probability densities by using hidden markov models with mixed states. In Weigend, A. S. and Gershenfeld, N. A., editors, *Time series prediction: Forecasting the future and understanding the past*, pages 265–282. Addison Wesley, Reading, MA.

Ghahramani, Z. and Hinton, G. (1996a). Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, Dept. of Computer Science, University of Toronto.

Ghahramani, Z. and Hinton, G. (1996b). Switching state-space models. Technical Report CRG-TR-96-3 DRAFT, Dept. of Computer Science, University of Toronto.

Ghahramani, Z. and Hinton, G. (1996 (revised Feb. 1997)). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Dept. of Computer Science, University of Toronto.

Ghahramani, Z. and Jordan, M. I. (1994). Supervised learning from incomplete data via an EM approach. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc.

Goodwin, G. and Sin, K. (1984). *Adaptive filtering prediction and control*. Prentice-Hall.

Hinton, G. and Ghahramani, Z. ((to appear 1997)). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society*, B.

Hinton, G. E., Dayan, P., and Revow, M. (1997). Modeling the manifolds of Images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74.

Hinton, G. E., Revow, M., and Dayan, P. (1995). Recognizing handwritten digits using mixtures of linear models. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 1015–1022. The MIT Press.

Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 3–10. Morgan Kaufmann Publishers, San Francisco, CA.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME, Series D, Journal of Basis Engineering*, (8):35–45.

Kalman, R. E. and Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Trans. ASME, Series D, Journal of Basis Engineering*, (83):95–108.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Little, R. J. A. and Rubin, D. B. (1987). *Statistical Analysis with Missing Data.* Wiley, New York.

Ljung, L. and Söderström, T. (1983). *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, MA.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:128–137.

Lyttkens, E. (1966). On the fixpoint property of wold's iterative estimation method for principal components. In Krishnaiah, P., editor, *Paper in Multivariate Analysis.* Academic Press, New York.

MacKay, D. J. C. (1996). Maximum likelihood and covariant algorithms for independent component analysis. Technical Report Draft 3.7, Cavendish Laboratory, University of Cambridge, Madingley Road, Cambridge CB3 0HE.

Neal, R. M. and Hinton, G. E. (1993). A new view of the EM algorithm that justifies incremental and other variants. *Submitted.*

Nowlan, S. J. (1991). Maximum likelihood competitive learning. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems*, volume 3, pages 574–582. Morgan Kaufmann, San Mateo, CA.

Pearlmutter, B. A. and Parra, L. C. (1997). Maximum likelihood blind source separation: A context-sensitive generalization of ica. In Mozer, M., Jordan, M., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9, pages 613–619. The MIT Press.

Rabiner, L. R. and Juang, B. H. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16.

Rauch, H. E. (1963). Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*, 8:371–372.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450.

Roweis, S. (1997a). EM algorithms for PCA and sensible PCA [to appear in NIPS10]. Technical Report CNS-TR-97-02, (April 1997), Computation and Neural Systems, California Institute of Technology.

Roweis, S. (1997b). Learning spatially adaptive observation noise in linear dynamical systems. Technical Report CNS-TR-97-03, Computation and Neural Systems, California Institute of Technology.

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

Shumway, R. H. and Stoffer, D. S. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264.

Shumway, R. H. and Stoffer, D. S. (1991). Dynamic linear models with switching. *Journal of the American Statistical Association*, 86(415):763–769.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. *Quarterly Applied Mathematics*, 45(3):561–590.

Smyth, P. (1997). Clustering sequences with hidden Markov models. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 9. The MIT Press.

Tipping, M. E. and Bishop, C. M. (1997). Mixtures of probabilistic principal component analysers. Technical Report TR-NCRG/97/003, Neural Computing Research Group, Aston University, Birmingham B4 7ET.

Tresp, V., Ahmad, S., and Neuneier, R. (1994). Training neural networks with deficient data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, San Francisco, CA. Morgan Kaufman Publishers.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269.