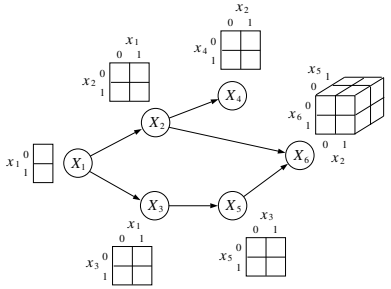
Lecture 17:

Linear Gaussian Models

Kevin Murphy
17 November 2004

Bayes nets with tabular CPDs

We have mostly focused on graphs where all latent nodes are discrete, and all CPDs/potentials are full tables.

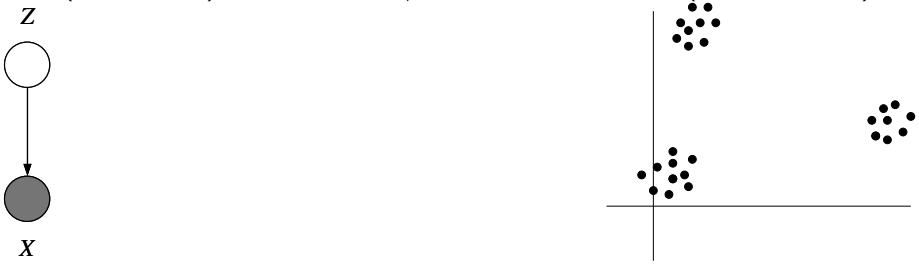


Mixtures of Gaussians

- We have also considered discrete latent variables with continuous observed variables.
- e.g., in a mixture of Gaussians:

$$P(Z = i) = \theta_i$$
$$p(X = x|Z = i) = \mathcal{N}(x; \mu_i, \Sigma_i)$$

- This can be used for classification (supervised) and clustering/ vector quantization (unsupervised).



Linear Gaussian model

- We now consider the case where all CPDs are linear-Gaussian:

$$p(Z = z) = \mathcal{N}(z; \mu_z, \Sigma_z)$$
$$p(X = x|Z = z) = \mathcal{N}(x; \mu_x + \Lambda z, \Sigma)$$

- i.e., child = linear function of parent plus gaussian noise

$$X = \Lambda Z + \text{noise}, \quad \text{noise} \sim \mathcal{N}(\mu_x, \Sigma_x)$$

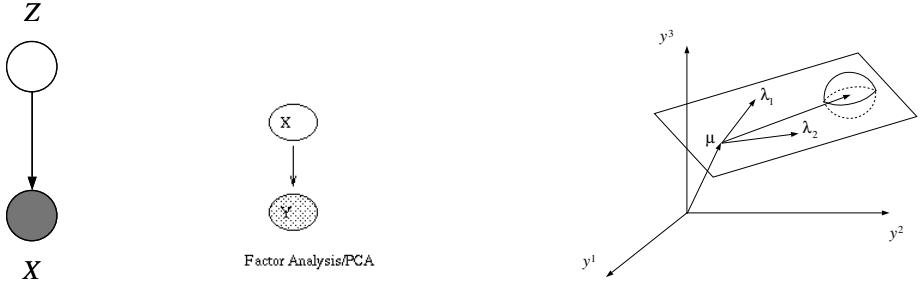
- For the supervised case, this is just linear regression:

Factor analysis (Jordan ch 14)

- Unsupervised linear regression is called factor analysis.

$$p(x) = \mathcal{N}(x; 0, I)$$
$$p(y|x) = \mathcal{N}(y; \mu + \Lambda x, \Psi)$$

where Λ is the factor loading matrix and Ψ is diagonal.



- To generate data, first generate a point within the manifold then add noise. Coordinates of point are components of latent variable.

Review: Means, Variances and Covariances

- Remember the definition of the mean and covariance of a vector random variable:

$$E[x] = \int_{\mathbf{x}} \mathbf{x} p(\mathbf{x}) d\mathbf{x} = \mathbf{m}$$

$$\mathrm{Cov}[x] = E[(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^\top] = \int_x (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^\top p(\mathbf{x}) d\mathbf{x} = \mathbf{V}$$

which is the expected value of the outer product of the variable with itself, after subtracting the mean.

- Also, the covariance between two variables:

$$\mathrm{Cov}[\mathbf{x}, \mathbf{y}] = E[(\mathbf{x} - \mathbf{m_x})(\mathbf{y} - \mathbf{m_y})^\top] = \mathbf{C}$$
$$= \int_{\mathbf{xy}} (\mathbf{x} - \mathbf{m_x})(\mathbf{y} - \mathbf{m_y})^\top p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} = \mathbf{C}$$

which is the expected value of the outer product of one variable with another, after subtracting their means.
Note: $\mathbf{C}$ is not symmetric.

Marginal Data Distribution

- Since a marginal Gaussian times a conditional Gaussian is a joint Gaussian, we can compute the marginal density $p(\mathbf{y}|\theta)$ by integrating out $\mathbf{x}$.

$$p(\mathbf{y}|\theta) = \int_{\mathbf{x}} p(\mathbf{x}) p(\mathbf{y}|\mathbf{x}, \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y}|\mu\ ,\ \Lambda\Lambda^\top + \Psi)$$

which can be done by *completing the square* in the exponent.

- However, since the marginal is Gaussian, we can also just compute its mean and covariance. (Assume noise uncorrelated with data.)

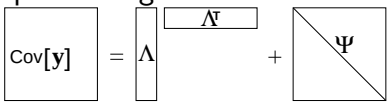
$$E[\mathbf{y}] = E[\mu + \Lambda\mathbf{x} + \text{noise}] = \mu + \Lambda E[\mathbf{x}] + E[\text{noise}]$$
$$= \mu + \Lambda \cdot 0 + 0 = \mu$$
$$\mathrm{Cov}[\mathbf{y}] = E[(\mathbf{y} - \mu)(\mathbf{y} - \mu)^\top]$$
$$= E[(\mu + \Lambda\mathbf{x} + \text{noise} - \mu)(\mu + \Lambda\mathbf{x} + \text{noise} - \mu)^\top]$$
$$= E[(\Lambda\mathbf{x} + n)(\Lambda\mathbf{x} + n)^\top] = \Lambda E(\mathbf{x}\mathbf{x}^\top)\Lambda^\top + E(nn^\top)$$
$$= \Lambda\Lambda^\top + \Psi$$

FA = Constrained Covariance Gaussian

- Marginal density for factor analysis ($\mathbf{y}$ is p-dim, $\mathbf{x}$ is k-dim):

$$p(\mathbf{y}|\theta) = \mathcal{N}(\mathbf{y}|\mu\ ,\ \Lambda\Lambda^\top + \Psi)$$

- So the effective covariance is the low-rank outer product of two long skinny matrices plus a diagonal matrix:



- In other words, factor analysis is just a constrained Gaussian model. (If Ψ were not diagonal then we could model any Gaussian and it would be pointless.)

Probabilistic Principal Component Analysis (PPCA)

- In Factor Analysis, we can write the marginal density explicitly:

$$p(\mathbf{y}|\theta) = \int_{\mathbf{x}} p(\mathbf{x})p(\mathbf{y}|\mathbf{x},\theta)d\mathbf{x} = \mathcal{N}(\mathbf{y}|\mu\ ,\ \Lambda\Lambda^\top + \Psi)$$

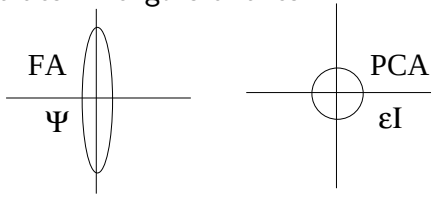
- Noise Ψ mut be restricted for model to be interesting. (Why?)
- In Factor Analysis the restriction is that Ψ is *diagonal* (axis-aligned).
- What if we further restrict $\Psi = \sigma^2 I$ (ie *spherical*)?
- We get the Probabilistic Principal Component Analysis (PPCA) model: ywhere μ is the mean vector,
  columns of Λ are the *principal components* (usually orthogonal),
  and σ^2 is the *global sensor noise*.

PCA: Zero Noise Limit

- The traditional PCA model is actually a limit as $\sigma^2 \to 0$.
- Actually PCA is the limit as the ratio of the noise variance on the output to the prior variance on the latent variables goes to zero. We can either achieve this with zero noise or with infinite variance priors.
- (P)PCA is very useful for dimensionality reduction, e.g., for visualizing high-dimensional data.

Difference between FA and PPCA

- Recall the intuition that Gaussians are hyperellipsoids, where Mean = centre of football, Eigenvectors of covariance matrix = axes of football, Eigenvalues = lengths of axes



- In FA our football is an axis aligned cigar.
  In PPCA our football is a sphere of radius σ^2.
- In FA, the variance of the noise is independent for each dimension; in PPCA, the noise is assumed to be the same.
- PCA looks for directions of large variance,
- Conversely, FA looks for directions of large correlation.

Model Invariance and Identfiability

- There is *degeneracy* in the FA model.
- Since Λ only appears as outer product $\Lambda\Lambda^\top$, the model is invariant to rotation and axis flips of the latent space.
- We can replace Λ with $\Lambda\mathbf{Q}$ for any unitary matrix $\mathbf{Q}$ and the model remains the same: $(\Lambda\mathbf{Q})(\Lambda\mathbf{Q})^\top = \Lambda(\mathbf{Q}\mathbf{Q}^\top)\Lambda^\top = \Lambda\Lambda^\top$.
- This means that there is no "one best" setting of the parameters. An infinite number of parameters all give the ML score!
- Such models are called un-identifiable since two people both fitting ML params to the identical data will not be guaranteed to identify the same parameters.

Latent Covariance in Factor Analysis and PCA

- What if we allow the latent variable $\mathbf{x}$ to have a covariance matrix of its own: $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, \mathbf{P})$?
- We can still compute the marginal probability:

$$p(\mathbf{y}|\theta) = \int_{\mathbf{x}} p(\mathbf{x})p(\mathbf{y}|\mathbf{x}, \theta)d\mathbf{x} = \mathcal{N}(\mathbf{y}|\mu\ ,\ \Lambda\mathbf{P}\Lambda^\top + \Psi)$$

- We can always absorb $\mathbf{P}$ into the loading matrix Λ by diagonalizing it: $P = EDE^\top$ and setting $\Lambda = \Lambda E D^{1/2}$.
- Thus, there is another degeneracy in FA, between $\mathbf{P}$ and Λ: we can set $\mathbf{P}$ to be the identity, to be diagonal, whatever we want.
- Traditionally we break this degeneracy by either:
  - set the covariance $\mathbf{P}$ of the latent variable to be I (FA) or
  - force the columns of Λ to be orthonormal (PCA)

FA joint distribution

- Model

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, I)$$
$$p(\mathbf{y}|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{y}|\mu + \Lambda\mathbf{x}, \Psi)$$

- Hence the joint distribtion of $\mathbf{x}$ and $\mathbf{y}$:

$$p(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}) = \mathcal{N}(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \mid \begin{bmatrix}0\\ \mu\end{bmatrix}, \begin{bmatrix}I & \Lambda^\top\\ \Lambda & \Lambda\Lambda^\top + \Psi\end{bmatrix})$$

where the corner elements $\Lambda^\top, \Lambda$ come from $\mathrm{Cov}[\mathbf{x}, \mathbf{y}]$:

$$\begin{aligned}\mathrm{Cov}[\mathbf{x}, \mathbf{y}] &= E[(\mathbf{x} - 0)(\mathbf{y} - \mu)^\top] = E[\mathbf{x}(\mu + \Lambda\mathbf{x} + noise - \mu)^\top]\\ &= E[\mathbf{x}(\Lambda\mathbf{x} + noise)^\top] = \Lambda^\top\end{aligned}$$

- Assume noise is uncorrelated with data or latent variables.

Review: Gaussian Conditioning (Jordan ch 13)

- Remember the formulas for conditional Gaussian distributions:

$$p(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}) = \mathcal{N}(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix})$$
$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\mathbf{m}_{1|2}, \mathbf{V}_{1|2})$$
$$\mathbf{m}_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2)$$
$$\mathbf{V}_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

Inference in Factor Analysis

- Apply the Gaussian conditioning formulas to the joint distribution we derived above, where

$$\begin{aligned}\Sigma_{11} &= I\\ \Sigma_{12} &= \Lambda^T\\ \Sigma_{22}^{-1} &= (\Lambda\Lambda^T + \Psi)^{-1}\end{aligned}$$

gives

$$\begin{aligned}V_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\\ &= I - \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}\Lambda\\ m_{1|2} &= m_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - m_2)\\ &= \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}(\mathbf{y} - \mu)\end{aligned}$$

Inference in Factor Analysis

- The previous formula requires inverting a matrix of size $|y| \times |y|$.

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{m}, \mathbf{V})$$
$$\mathbf{V} = I - \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}\Lambda$$
$$\mathbf{m} = \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}(\mathbf{y} - \mu)$$

- We now use a good trick for inverting matrices when they can be decomposed into the sum of an easily inverted matrix (D) and a low rank outer product. It is called the *matrix inversion lemma*.

$$(D - AB^{-1}A^\top)^{-1} = D^{-1} + D^{-1}A(B - A^\top D^{-1}A)^{-1}A^\top D^{-1}$$

- Apply the matrix inversion lemma we get:

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{m}, \mathbf{V})$$
$$\mathbf{V} = (I + \Lambda^\top\Psi^{-1}\Lambda)^{-1}$$
$$\mathbf{m} = \mathbf{V}\Lambda^\top\Psi^{-1}(\mathbf{y} - \mu)$$

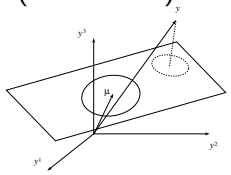
which inverts a matrix of size $|x| \times |x|$.

Inference is Linear projection

- The posterior is

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\mathbf{m}, \mathbf{V})$$
$$\mathbf{V} = I - \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}\Lambda = (I + \Lambda^\top\Psi^{-1}\Lambda)^{-1}$$
$$\mathbf{m} = \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}(\mathbf{y} - \mu) = \mathbf{V}\Lambda^\top\Psi^{-1}(\mathbf{y} - \mu)$$

- Posterior covariance does not depend on observed data y! (Can be precomputed.)
- Also, computing the posterior mean is just a linear operation: $\mathbf{m} = \beta(\mathbf{y} - \mu)$ where β (hat matrix) can be precomputed.



Incomplete data log likelihood Function

- Using the marginal density of $p(y)$:

$$\ell(\theta; \mathcal{D}) = -\frac{N}{2}\log|\Lambda\Lambda^\top + \Psi| - \frac{1}{2}\sum_n (\mathbf{y}^n - \mu)^\top(\Lambda\Lambda^\top + \Psi)^{-1}(\mathbf{y}^n - \mu)$$
$$= -\frac{N}{2}\log|\mathbf{V}| - \frac{1}{2}\text{trace}\left[\mathbf{V}^{-1}\sum_n (\mathbf{y}^n - \mu)(\mathbf{y}^n - \mu)^\top\right]$$
$$= -\frac{N}{2}\log|\mathbf{V}| - \frac{1}{2}\text{trace}\left[\mathbf{V}^{-1}\mathbf{S}\right]$$

$\mathbf{V}$ is model covariance; $\mathbf{S}$ is sample data covariance.

- In other words, we are trying to make the constrained model covariance as close as possible to the observed covariance, where "close" means the trace of the ratio.
- Thus, the sufficient statistics are the same as for the Gaussian: mean $\sum_n \mathbf{y}^n$ and covariance $\sum_n (\mathbf{y}^n - \mu)(\mathbf{y}^n - \mu)^\top$.

EM for Factor Analysis

- Parameters are coupled nonlinearly in log-likelihood:

$$\ell(\theta; \mathcal{D}) = -\frac{N}{2}\log|\Lambda\Lambda^\top + \Psi| - \frac{1}{2}\text{trace}\left[(\Lambda\Lambda^\top + \Psi)^{-1}\mathbf{S}\right]$$

- We could use conjugate gradient methods.
- But EM is simpler.
- For E-step we do inference (linear projection)
  For M-step we do linear regression on the expected values.
  More precisely, we maximize the expected complete log likelihood.

$$\mathbf{E-step}: q_n^{t+1} = p(\mathbf{x}^n|\mathbf{y}^n, \theta^t) = \mathcal{N}(\mathbf{x}^n|\mathbf{m}^n, \mathbf{V}^n)$$
$$\mathbf{M-step}: \Lambda^{t+1} = \text{argmax}_\Lambda \sum_n \langle \ell_c(\mathbf{x}^n, \mathbf{y}^n)\rangle_{q_n^{t+1}}$$
$$\Psi^{t+1} = \text{argmax}_\Psi \sum_n \langle \ell_c(\mathbf{x}^n, \mathbf{y}^n)\rangle_{q_n^{t+1}}$$

Complete Data Likelihood

- We know the optimal μ is the data mean.
  Assume the mean has been subtracted off $\mathbf{y}$ from now on.
- The complete likelihood (ignoring mean):

$$\begin{aligned}
\ell_c(\Lambda, \Psi) &= \sum_n \log p(\mathbf{x}^n, \mathbf{y}^n) \\
&= \sum_n \log p(\mathbf{x}^n) + \log p(\mathbf{y}^n|\mathbf{x}^n) \\
&= -\frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_n \mathbf{x}^\top\mathbf{x} - \frac{1}{2}\sum_n (\mathbf{y}^n - \Lambda\mathbf{x}^n)^\top\Psi^{-1}(\mathbf{y}^n - \Lambda\mathbf{x}^n) \\
&= -\frac{N}{2}\log|\Psi| - \frac{N}{2}\text{trace}[S\Psi^{-1}] \\
S &= \frac{1}{N}\sum_n (\mathbf{y}^n - \Lambda\mathbf{x}^n)(\mathbf{y}^n - \Lambda\mathbf{x}^n)^\top
\end{aligned}$$

M-step: Optimize Parameters (Jordan ch 13)

- Take the derivates of the complete log likelihood wrt. parameters.
- We will need thse identities.

$$\begin{aligned}
\frac{\partial}{\partial A}\log|A| &= (A^{-1})^\top \\
\frac{\partial}{\partial A}\text{trace}[B^\top A] &= B \\
\frac{\partial}{\partial A}\text{trace}[BA^\top CA] &= 2CAB
\end{aligned}$$

- Hence

$$\begin{aligned}
\partial\ell_c(\Lambda, \Psi)/\partial\Lambda &= -\Psi^{-1}\sum_n \mathbf{y}_n\mathbf{x}_n^\top + \Psi^{-1}\Lambda\sum_n \mathbf{x}_n\mathbf{x}_n^\top \\
\partial\ell_c(\Lambda, \Psi)/\partial\Psi^{-1} &= +(N/2)\Psi - (N/2)S
\end{aligned}$$

M-step: Optimize Parameters

- Derivatives from previous slide:

$$\begin{aligned}
\partial\ell_c(\Lambda, \Psi)/\partial\Lambda &= -\Psi^{-1}\sum_n \mathbf{y}_n\mathbf{x}_n^\top + \Psi^{-1}\Lambda\sum_n \mathbf{x}_n\mathbf{x}_n^\top \\
\partial\ell_c(\Lambda, \Psi)/\partial\Psi^{-1} &= +(N/2)\Psi - (N/2)S
\end{aligned}$$

- Take the expectation of gradient with respect to q^t from E-step:

$$\begin{aligned}
< \ell'_\Lambda > &= -\Psi^{-1}\sum_n \mathbf{y}_n\mathbf{m}_n^\top + \Psi^{-1}\Lambda\sum_n \mathbf{V}_n \\
< \ell'_{\Psi^{-1}} > &= +(N/2)\Psi - (N/2) < S >
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{m}_n &= E[X_n|y_n] \\
\mathbf{V}_n &= Var(X_n|y_n) + E(X_n|y_n)E(X_n|y_n)^\top \\
< S > &= \frac{1}{N}\sum_n y_n y_n^\top - y_n m_n^\top\Lambda^\top - \Lambda m_n y_n^\top + \Lambda V_n\Lambda^\top
\end{aligned}$$

M-step: Optimize Parameters

- Take the expectation of gradient with respect to q^t from E-step:

$$\begin{aligned}
< \ell'_\Lambda > &= -\Psi^{-1}\sum_n \mathbf{y}_n\mathbf{m}_n^\top + \Psi^{-1}\Lambda\sum_n \mathbf{V}_n \\
< \ell'_{\Psi^{-1}} > &= +(N/2)\Psi - (N/2) < S >
\end{aligned}$$

- Finally, set the derivatives to zero to solve for optimal parameters:

$$\begin{aligned}
\Lambda^{t+1} &= \left(\sum_n \mathbf{y}^n\mathbf{m}_n{}^\top\right)\left(\sum_n \mathbf{V}_n\right)^{-1} \\
\Psi^{t+1} &= \frac{1}{N}\text{diag}\left[\sum_n \mathbf{y}_n\mathbf{y}_n{}^\top - \Lambda^{t+1}\sum_n \mathbf{m}_n\mathbf{y}_n{}^\top\right]
\end{aligned}$$

Final Algorithm: EM for Factor Analysis

- First, set μ equal to the sample mean $(1/N)\sum_n \mathbf{y}_n$, and subtract this mean from all the data.
- Now run the following iterations:

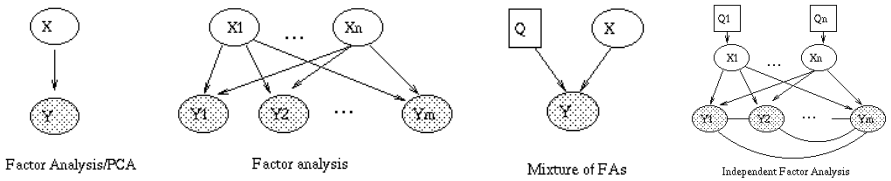
$$\mathbf{E-step}: q^{t+1} = p(\mathbf{x}|\mathbf{y},\theta^t) = \mathcal{N}(\mathbf{x}^n|\mathbf{m}^n,\mathbf{V}^n)$$
$$\mathbf{V}^n = (I + \Lambda^\top \Psi^{-1}\Lambda)^{-1}$$
$$\mathbf{m}^n = \mathbf{V}^n \Lambda^\top \Psi^{-1}(\mathbf{y}-\mu)$$
$$\mathbf{M-step}: \Lambda^{t+1} = \left(\sum_n \mathbf{y}^n \mathbf{m}^{n\top}\right)\left(\sum_n \mathbf{V}^n\right)^{-1}$$
$$\Psi^{t+1} = \frac{1}{N}\mathrm{diag}\left[\sum_n \mathbf{y}^n\mathbf{y}^{n\top} - \Lambda^{t+1}\sum_n \mathbf{m}^n\mathbf{y}^{n\top}\right]$$

Learning for PPCA

- For FA the parameters are coupled in a way that makes it impossible to solve for the ML params directly.
  We must use EM or other nonlinear optimization techniques.
- But for (P)PCA, the ML params can be solved for directly:
  The k^{th} column of Λ is the k^{th} largest eigenvalue of the sample covariance $\mathbf{S}$ times the associated eigenvector.
- The global sensor noise σ^2 is the sum of all the eigenvalues smaller than the k^{th} one.
- This technique is good for initializing FA also.
- Note: EM can be used as a fast way to compute eigenvectors!

Independent Components Analysis (ICA)

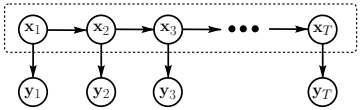
- ICA is similar to FA, except it assumes the latent source has *non-Gaussian density*.
- Hence ICA can extract higher order moments (not just second order).
- It is commonly used to solve blind source separation (cocktail party problem).
- Independent Factor Analysis (IFA) is an approximation to ICA where we model the source using a mixture of Gaussians.



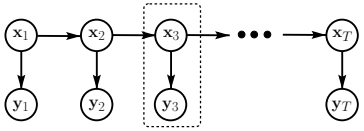
FA FA Mixture of FA IFA

Hidden Markov Models

- You can think of an HMM as:
  A Markov chain with stochastic measurements.



or a mixture of Gaussians, with latent variables changing over time (lily pad model)



State space models (SSMs)

- An SSM is like an HMM, except the hidden state is continuous valued.
- In general, we can write

$$x_t = f(x_{t-1}) + w_t$$
$$y_t = g(x_t) + v_t$$

  where f is the dynamical (process) model, g is the observation model.
- w_t is the process noise, embedded inside $P(X_t|X_{t-1})$.
- v_t is the observation noise embedded inside $P(Y_t|X_t)$.
- For a controlled Markov chain, we condition all quantities on the observed input signals u_t.

Linear dynamic systems (LDS) (Jordan ch 15)

- An LDS is a special case of a SSM where f and g are linear functions and the noise terms are Gaussian.

$$x_t = Ax_{t-1} + w_t$$
$$y_t = Bx_t + v_t$$
$$w_t \sim \mathcal{N}(0, Q)$$
$$v_t \sim \mathcal{N}(0, R)$$

- An LDS is like factor analysis through time.

Kalman filtering and smoothing

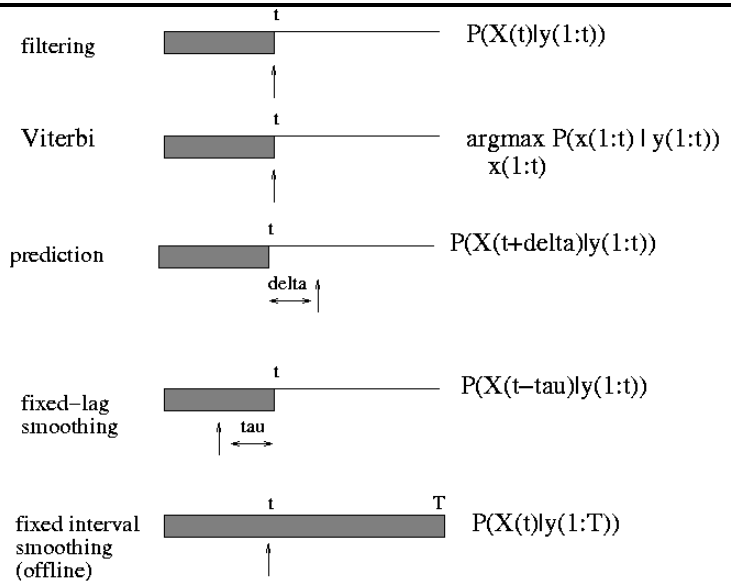
- The Kalman filter is a way to perform exact online inference (sequential Bayesian updating) in an LDS. It is the Gaussian analog of the forwards algorithm for HMMs:

$$P(X_t = i|y_{1:t}) = \alpha_t(i) \propto p(y_t|X_t = i) \sum_j P(X_t = i|X_{t-1} = j)\alpha_{t-1}(j)$$

- The Rauch-Tung-Strievel smoother is a way to perform exact offline inference in an LDS. It is the Gaussian analog of the forwards-backwards algorithm:

$$P(X_t = i|y_{1:T}) \propto \alpha_t(i)\beta_t(i)$$

Online vs offline inference



LDS FOR 2D TRACKING

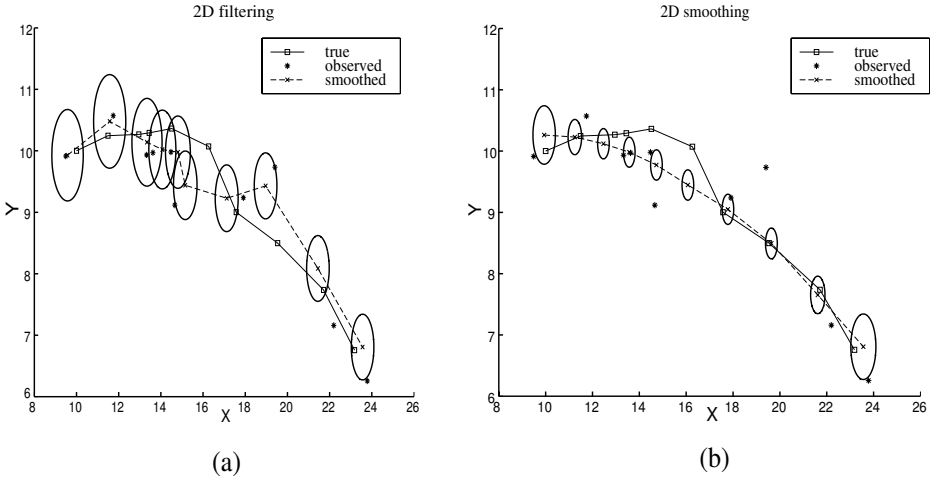
- Dynamics: new position = old position + Δ * velocity + noise (constant velocity model, Gaussian acceleration)

$$\begin{pmatrix} x_t^1 \\ x_t^2 \\ \dot{x}_t^1 \\ \dot{x}_t^2 \end{pmatrix} = \Delta \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{t-1}^1 \\ x_{t-2}^2 \\ \dot{x}_{t-3}^1 \\ \dot{x}_{t-4}^2 \end{pmatrix} + \text{noise}$$

- Observation: project out first two components (we observe Cartesian position of object - linear!)

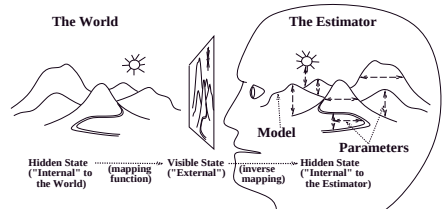
$$\begin{pmatrix} y_t^1 \\ y_t^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ \dot{x}_t^1 \\ \dot{x}_t^2 \end{pmatrix} + \text{noise}$$

2D TRACKING

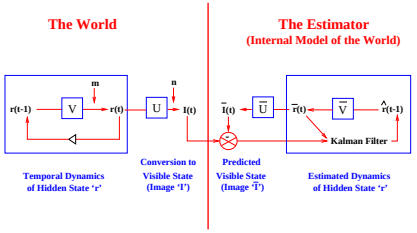


(a) (b)

KALMAN FILTERING IN THE BRAIN?



(a)



(b)

KALMAN FILTERING DERIVATION

- Since all CPDs are linear Gaussian, the system defines a large multivariate Gaussian.
- Hence all marginals are Gaussian.
- Hence we can represent the belief state $P(X_t|y_{1:t})$ as a Gaussian with mean $\hat{x}_{t|t}$ and covariance $P_{t|t}$.
- It is common to work with the inverse covariance (precision) matrix $P_{t|t}^{-1}$; this is called *information form*.
- Kalman filtering is a recursive procedure to update the belief state:
  - Predict step: compute $P(X_{t+1}|y_{1:t})$ from prior belief $P(X_t|y_{1:t})$ and dynamical model $P(X_{t+1}|X_t)$.
  - Update step: compute new belief $P(X_{t+1}|y_{1:t+1})$ from prediction $P(X_{t+1}|y_{1:t})$, observation y_{t+1} and observation model $P(y_t|X_t)$.

Predict step

- Dynamical Model: $x_{t+1} = Ax_t + Gw_t, \;\; w_t \sim \mathcal{N}(0, Q)$.
- One step ahead prediction of state:

$$\begin{aligned} \hat{x}_{t+1|t} &= E[X_{t+1}|y_{1:t}] = A\hat{x}_{t|t} \\ P_{t+1|t} &= E[(X_{t+1} - \hat{x}_{t+1|t})(X_{t+1} - \hat{x}_{t+1|t})^T|y_{1:t}] \\ &= AP_{t|t}A^T + GQG^T \end{aligned}$$

- Observation Model: $y_t = Cy_t + v_t, \;\; v_t \sim \mathcal{N}(0, R)$.
- One step ahead prediction of observation:

$$\begin{aligned} E[Y_{t+1}|y_{1:t}] = E[CX_{t+1} + v_{t+1}|y_{1:t}] &= C\hat{x}_{t+1|t} \\ E[(Y_{t+1} - \hat{y}_{t+1|t})(Y_{t+1} - \hat{y}_{t+1|t})^T|y_{1:t}] &= CP_{t+1|t}C^T + R \\ E[(Y_{t+1} - \hat{y}_{t+1|t})(X_{t+1} - \hat{x}_{t+1|t})^T|y_{1:t}] &= CP_{t+1|t} \end{aligned}$$

Update step

- Summarizing results from previous slide, we have $P(X_{t+1}, Y_{t+1}|y_{1:t}) \sim \mathcal{N}(m_{t+1}, V_{t+1})$, where

$$m_{t+1} = \begin{pmatrix} \hat{x}_{t+1|t} \\ C\hat{x}_{t+1|t} \end{pmatrix}, \;\; V_{t+1} = \begin{pmatrix} P_{t+1|t} & P_{t+1|t}C^T \\ CP_{t+1|t} & CP_{t+1|t}C^T + R \end{pmatrix}$$

- Remember the formulas for conditional Gaussian distributions:

$$\begin{aligned} p(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}) &= \mathcal{N}(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} | \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}) \\ p(\mathbf{x}_1|\mathbf{x}_2) &= \mathcal{N}(\mathbf{x}_1|\mathbf{m}_{1|2}, \mathbf{V}_{1|2}) \\ \mathbf{m}_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2) \\ \mathbf{V}_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned}$$

Kalman filter equations

- Using the conditioning formulas, we get:

$$\begin{aligned} \hat{x}_{t+1|t} &= A\hat{x}_{t|t} \\ P_{t+1|t} &= AP_{t|t}A^T + GQG^T \\ \hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + K_{t+1}(y_{t+1} - C\hat{x}_{t+1|t}) \\ P_{t+1|t+1} &= P_{t+1|t} - K_{t+1}CP_{t+1|t} \end{aligned}$$

  where K_{t+1} is the *Kalman gain matrix*

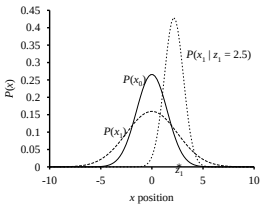
$$K_{t+1} = P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1}$$

- K_t can be precomputed (since it is independent of the data). This rapidly converges to a constant matrix (Ricatti equation).

Example of KF in 1D (Russell & Norvig 2e p554)

- Consider noisy observations of a 1D particle doing a random walk: $x_t = x_{t-1} + \mathcal{N}(0, \sigma_x), \;\; y_t = x_t + \mathcal{N}(0, \sigma_y)$
- Hence

$$\begin{aligned} P_{t+1|t} &= AP_{t|t}A^T + GQG^T = \sigma_t + \sigma_x \\ K_{t+1} &= P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1} = (\sigma_t + \sigma_x)(\sigma_t + \sigma_x + \sigma_z)^{-1} \\ \hat{x}_{t+1|t+1} &= \hat{x}_{t+1|t} + K_{t+1}(y_{t+1} - C\hat{x}_{t+1|t}) = \frac{(\sigma_t + \sigma_x)z_t + \sigma_z\mu_t}{\sigma_t + \sigma_x + \sigma_z} \\ P_{t+1|t+1} &= P_{t+1|t} - K_{t+1}CP_{t+1|t} = \frac{(\sigma_t + \sigma_x)\sigma_z}{\sigma_t + \sigma_x + \sigma_z} \end{aligned}$$



KF: intuition

- The KF update of the mean is

$$\hat{x}_{t+1|t+1} = A\hat{x}_{t|t} + K_{t+1}(y_{t+1} - CA\hat{x}_{t|t}) = \frac{(\sigma_t + \sigma_x)z_t + \sigma_z \mu_t}{\sigma_t + \sigma_x + \sigma_z}$$

- The term $(y_{t+1} - C\hat{x}_{t+1|t})$ is called the *innovation*.
- New belief is convex combination of updates from prior and observation, weighted by Kalman Gain matrix:

$$K_{t+1} = P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1}$$

- If the observation is unreliable, σ_z is large so K_{t+1} is small, so we pay more attention to the prediction.
- If the old prior is unreliable (large σ_t) or the process is very unpredictable (large σ_x), we pay more attention to the observation.

KF, RLS and LMS

- The KF update of the mean is

$$\hat{x}_{t+1|t+1} = A\hat{x}_{t|t} + K_{t+1}(y_{t+1} - CA\hat{x}_{t|t})$$

- Consider the special case where the hidden state is a constant, $x_t = \theta$, but the "observation matrix" C is a time-varying vector, $C = x_t^T$.
- Hence $y_t = x_t^T\theta + v_t$ as in linear regression.
- We can estimate θ recursively using the Kalman filter:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + P_{t+1}R^{-1}(y_{t+1} - x_t^T\hat{\theta}_t)x_t$$

  This is called the *recursive least squares* (RLS) algorithm.
- We can approximate $P_{t+1}R^{-1} \approx \eta_{t+1}$ by a scalar constant. This is called the *least mean squares* (LMS) algorithm.
- We can adapt η_t online using stochastic approximation theory.