# Lecture 9:

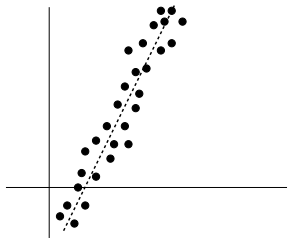
# Continuous Latent Variable Models

Sam Roweis

November 4, 2003

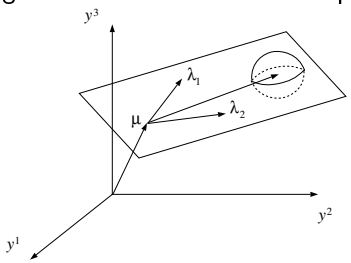
## Dimensionality Reduction vs. Clustering

- Training such "factor models" (e.g. FA, PCA, ICA) is called *dimensionality reduction*.
  You can think of this as (non)linear regression with missing inputs.
- Continuous causes can sometimes be much more efficient at representing information than discrete causes.
- For example, if there are two factors, with about 256 settings each we can describe the latent causes with two 8-bit numbers.
- If we tried to cluster we would need $2^{16} \approx 10^5$ clusters.



## Continuous Latent Variable Models

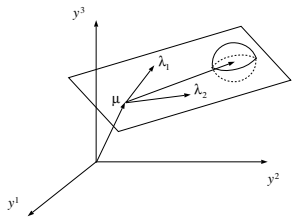
- Often there are some unknown *underlying causes* of the data.
- Mixture models use a discrete class variable: clustering.
- Sometimes, it is more appropriate to think in terms of continuous *factors* which control the data we observe. Geometrically, this is equivalent to thinking of a data *manifold* or subspace.




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

## Factor Analysis

- When we assume that the subspace is *linear* and that the underlying latent variable has a Gaussian distribution we get a model known as *factor analysis*:
  — data $\mathbf{y}$ ($p$-dim);
  — latent variable $\mathbf{z}$ ($k$-dim)




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

where $\mu$ is the mean vector, $\Lambda$ is the $p$ by $k$ *factor loading matrix*, and $\Psi$ is the *sensor noise covariance* (usually diagonal).

- Important: since the product of Gaussians is still Gaussian, the joint distribution $p(\mathbf{z}, \mathbf{y})$, the other marginal $p(\mathbf{y})$ and the conditional $p(\mathbf{z}|\mathbf{y})$ are also Gaussian.

## Marginal Data Distribution

- Just as with discrete latent variables, we can compute the marginal density $p(\mathbf{y}|\theta)$ by summing out $\mathbf{z}$. But now the sum is an integral:

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

  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.)

$$\begin{aligned} E[\mathbf{y}] &= E[\mu + \Lambda\mathbf{z} + \text{noise}] = \mu + \Lambda E[\mathbf{z}] + E[\text{noise}] \\ &= \mu + \Lambda \cdot 0 + 0 = \mu \\ \text{Cov}[\mathbf{y}] &= E[(\mathbf{y}-\mu)(\mathbf{y}-\mu)^\top] \\ &= E[(\mu + \Lambda\mathbf{z} + \text{noise} - \mu)(\mu + \Lambda\mathbf{z} + \text{noise} - \mu)^\top] \\ &= E[(\Lambda\mathbf{z} + n)(\Lambda\mathbf{z} + n)^\top] = \Lambda E(\mathbf{z}\mathbf{z}^\top)\Lambda^\top + E(nn^\top) \\ &= \Lambda\Lambda^\top + \Psi \end{aligned}$$

## Reminder: Gaussian Conditioning

- Remember the formulas for marginalizing and conditioning Gaussian probability distribution functions.
- Joint:

$$p(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix} \mid \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right)$$

- Marginals:

$$p(\mathbf{x}_1) = \mathcal{N}(\mu_1, \Sigma_{11})$$

- Conditionals:

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

## Reminder: Means, Variances and Covariances

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

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

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

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

- Also, the (cross)covariance between two variables:

$$\begin{aligned} \text{Cov}[\mathbf{x},\mathbf{y}] &= E[(\mathbf{x}-\mathbf{m}_\mathbf{x})(\mathbf{y}-\mathbf{m}_\mathbf{y})^\top] = \mathbf{C} \\ &= \int_{\mathbf{xy}} (\mathbf{x}-\mathbf{m}_\mathbf{x})(\mathbf{y}-\mathbf{m}_\mathbf{y})^\top p(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} = \mathbf{C} \end{aligned}$$

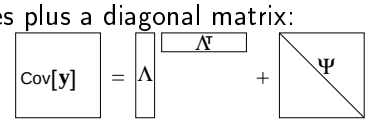
  which is the expected value of the outer product of one variable with another, after subtracting their means.
  Note: $\mathbf{C}$ is not symmetric.

## Constrained Covariance

- Marginal density for factor analysis ($\mathbf{y}$ is $p$-dim, $\mathbf{z}$ 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 $\Psi$ were not diagonal then we could model any Gaussian and it would be pointless.)
- It is easy to find $\mu$: just take the mean of the data.
  From now on assume we have done this and re-centred $\mathbf{y}$.

## EM for Factor Analysis

- We will do maximum likelihood learning using (surprise, surprise) the EM algorithm.
  **E-step**: $q^{t+1} = p(\mathbf{z}^n|\mathbf{y}^n, \theta^t)$
  **M-step**: $\theta^{t+1} = \text{argmax}_\theta \sum_n \int_{\mathbf{z}} q^{t+1}(\mathbf{z}^n|\mathbf{y}^n) \log p(\mathbf{y}^n, \mathbf{z}^n|\theta) d\mathbf{z}^n$
- For this we need the conditional distribution (inference) and the expected log of the complete data. Results:

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

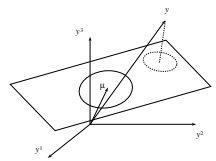
## Inference in Factor Analysis

- Apply the Gaussian conditioning formulas to the joint distribution we derived above. This gives:

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

- Now apply the matrix inversion lemma to get:

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



## Complete Data Likelihood

- Write down the joint distribution of $\mathbf{z}$ and $\mathbf{y}$:

$$p\left(\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix} \middle| \begin{bmatrix}0\\ \mu\end{bmatrix}, \begin{bmatrix}I & \Lambda^\top\\ \Lambda & \Lambda\Lambda^\top + \Psi\end{bmatrix}\right)$$

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

$$\text{Cov}[\mathbf{z}, \mathbf{y}] = E[(\mathbf{z} - 0)(\mathbf{y} - \mu)^\top] = E[\mathbf{z}(\mu + \Lambda\mathbf{z} + n - \mu)^\top]$$
$$= E[\mathbf{z}(\Lambda\mathbf{z} + n)^\top] = \Lambda^\top$$

- This gives the complete likelihood (ignoring mean):

$$\ell_c(\Lambda, \Psi) = -\frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_n \mathbf{z}^\top\mathbf{z} - \frac{1}{2}\sum_n (\mathbf{y}^n - \Lambda\mathbf{z}^n)^\top \Psi^{-1}(\mathbf{y}^n - \Lambda\mathbf{z}^n)$$
$$= -\frac{N}{2}\log|\Psi| - \frac{N}{2}\text{trace}[S\Psi^{-1}]$$
$$S = \frac{1}{N}\sum_n (\mathbf{y}^n - \Lambda\mathbf{z}^n)(\mathbf{y}^n - \Lambda\mathbf{z}^n)^\top$$

## Matrix Inversion Lemma

- There is 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*.

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

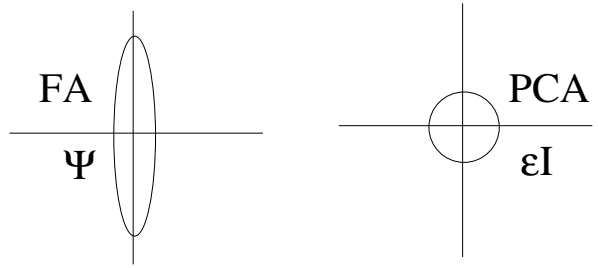
## Derivatives

- You need these tricks to compute the M-step derivatives:

$$\frac{\partial}{\partial A}\log|A| = A^{-\top}$$
$$\frac{\partial}{\partial A}\mathrm{trace}[B^\top A] = B$$
$$\frac{\partial}{\partial A}\mathrm{trace}[BA^\top CA] = 2CAB$$

## Gaussians are Footballs in High-D

- Recall the intuition that Gaussians are hyperellipsoids.
- 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 PCA our football is a sphere of radius $\sigma^2$.




## Principal Component Analysis

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

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

- Noise $\Psi$ mut be restricted for model to be interesting. (Why?)
- In Factor Analysis the restriction is that $\Psi$ is *diagonal* (axis-aligned).
- What if we further restrict $\Psi = \sigma^2 I$ (ie *spherical*)?
- We get the Principal Component Analysis (PCA) model:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, I)$$
$$p(\mathbf{y}|\mathbf{z},\theta) = \mathcal{N}(\mathbf{y}|\mu + \Lambda\mathbf{z}, \sigma^2 I)$$

  where $\mu$ is the mean vector,
  columns of $\Lambda$ are the *principal components* (usually orthogonal),
  and $\sigma^2$ is the *global sensor noise*.

## Likelihood Functions

- For both FA and PCA, the data model is Gaussian.
  Thus, the likelihood function is simple:

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

  $\mathbf{V}$ 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 $\sum_n(\mathbf{y}^n - \mu)(\mathbf{y}^n - \mu)^\top$.

## Fitting the PCA model

- The standard EM algorithm applies to PCA also:
  **E-step**: $q^{t+1} = p(\mathbf{z}^n|\mathbf{y}^n, \theta^t)$
  **M-step**: $\theta^{t+1} = \mathrm{argmax}_\theta \sum_n \int_{\mathbf{z}} q^{t+1}(\mathbf{z}^n|\mathbf{y}^n) \log p(\mathbf{y}^n, \mathbf{z}^n|\theta) d\mathbf{z}^n$
- For this we need the conditional distribution (inference) and the expected log of the complete data. Results:

$$\begin{aligned}
\mathbf{E-step}: q^{t+1} &= p(\mathbf{z}|\mathbf{y}, \theta^t) = \mathcal{N}(\mathbf{z}^n|\mathbf{m}^n, \mathbf{V}^n) \\
\mathbf{V}^n &= (I + \sigma^{-2}\Lambda^\top\Lambda)^{-1} \\
\mathbf{m}^n &= \sigma^{-2}\mathbf{V}^n\Lambda^\top(\mathbf{y} - \mu) \\
\mathbf{M-step}: \Lambda^{t+1} &= \left(\sum_n \mathbf{y}^n\mathbf{m}^{n\top}\right)\left(\sum_n \mathbf{V}^n\right)^{-1} \\
\sigma^{2\,t+1} &= \frac{1}{ND}\sum_i \left[\sum_n \mathbf{y}^n\mathbf{y}^{n\top} + \Lambda^{t+1}\sum_n \mathbf{m}^n\mathbf{y}^{n\top}\right]_{ii}
\end{aligned}$$

## Inference is Linear

- Recall the inference formulas for FA:

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

- Note: inference of the posterior mean is just a linear operation!

$$\mathbf{m} = \beta(\mathbf{y} - \mu)$$

  where $\beta$ can be computed beforehand given the model parameters.
- Also: posterior covariance does not depend on observed data!

$$\mathrm{cov}[\mathbf{z}|\mathbf{y}] = \mathbf{V} = (I + \Lambda^\top\Psi^{-1}\Lambda)^{-1}$$

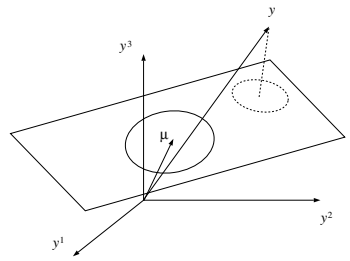
## Direct Fitting

- 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 PCA, the ML params can be solved for directly:
  The $k^{th}$ column of $\Lambda$ is the $k^{th}$ largest eigenvalue of the sample covariance $\mathbf{S}$ times the associated eigenvector.
- The global sensor noise $\sigma^2$ is the sum of all the eigenvalues smaller than the $k^{th}$ one.
- This technique is good for initializing FA also.
- We can't make the sensor noise unconstrained, or else we would always get a perfect fit!

## Zero Noise Limit

- The traditional PCA model is actually a limit as $\sigma^2 \to 0$.
  The model we saw is actually called "probabilistic PCA".
- However, the ML parameters $\Lambda^*$ are the same.
  The only difference is the global sensor noise $\sigma^2$.
- In the zero noise limit inference is easier: orthogonal projection.

$$\lim_{\sigma^2 \to 0} \Lambda^\top(\Lambda\Lambda^\top + \sigma^2 I)^{-1} = (\Lambda^\top\Lambda)^{-1}\Lambda^\top$$

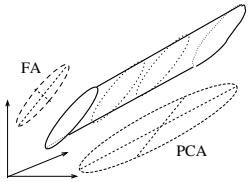


## Scale Invariance in Factor Analysis

- In FA the *scale* of the data is unimportant: we can multiply $y_i$ by $\alpha_i$ without changing anything:

$$\begin{aligned} \mu_i &\leftarrow \alpha_i \mu_i \\ \Lambda_{ij} &\leftarrow \alpha_i \Lambda_{ij} \quad \forall j \\ \Psi_i &\leftarrow \alpha_i^2 \Psi_i \end{aligned}$$

- However, the *rotation* of the data *is* important.
- FA looks for directions of large correlation in the data, so it is not fooled by large variance noise.




## Model Invariance and Identifiability

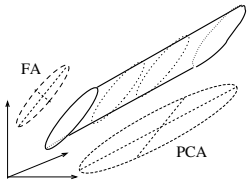
- There is *degeneracy* in the FA model.
- Since $\Lambda$ only appears as outer product $\Lambda\Lambda^\top$, the model is invariant to rotation and axis flips of the latent space.
- We can replace $\Lambda$ with $\Lambda\mathbf{Q}$ for any unitary matrix $\mathbf{Q}$ and the model remains the same: $(\Lambda\mathbf{Q})(\Lambda\mathbf{Q})^\top = \Lambda(\mathbf{Q}\mathbf{Q}^\top)\Lambda^\top = \Lambda\Lambda^\top$.
- 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.

## Rotational Invariance in PCA

- In PCA the *rotation* of the data is unimportant: we can multiply the data $\mathbf{y}$ by and rotation $\mathbf{Q}$ without changing anything:

$$\begin{aligned} \mu &\leftarrow \mathbf{Q}\mu \\ \Lambda &\leftarrow \mathbf{Q}\Lambda \\ \Psi &\leftarrow \text{unchanged} \end{aligned}$$

- However, the *scale* of the data *is* important.
- PCA looks for directions of large variance, so it will chase big noise directions.




## Latent Covariance in Factor Analysis and PCA

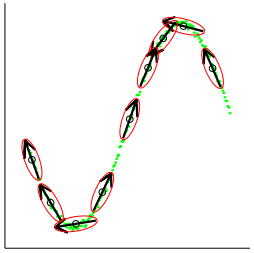
- What if we allow the latent variable $\mathbf{z}$ to have a covariance matrix of its own: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{P})$?
- We can still compute the marginal probability:

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

- We can always absorb $\mathbf{P}$ into the loading matrix $\Lambda$ by diagonalizing it: $P = EDE^\top$ and setting $\Lambda = \Lambda ED^{1/2}$.
- Thus, there is another degeneracy in FA, between $\mathbf{P}$ and $\Lambda$: 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 $\Lambda$ to be orthonormal (PCA)

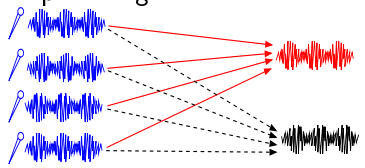
## Mixtures of Dimensionality Reducers

- What's the next logical step?
- Try a model that has two kinds latent variables: one discrete cluster, and one vector of continuous causes.
- Such models simultaneously do clustering, and within each cluster, dimensionality reduction. Great idea!



## Independent Components Analysis (ICA)

- ICA is another continuous latent variable model, like FA, but it has a *non-Gaussian* and *factorized* prior on the latent variables.
- This is good in situations where most of the factors are very small most of the time and they do not interact with each other. Example: mixtures of speech signals.



- The learning problem is the same: find the weights from the factors to the outputs and infer the unknown factor values. In the case of ICA the factors are sometimes called "sources", and the learning is sometimes called "unmixing".

## Mixtures of Factor Analyzers

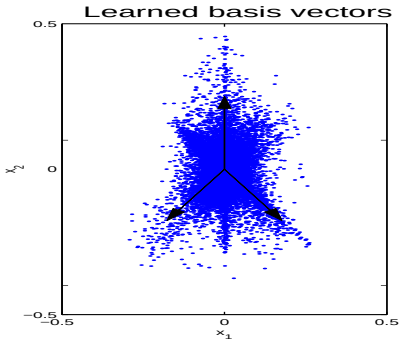
- The simplest version of this is the *mixture of factor analyzers*.

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|0, I) \qquad p(k) = \alpha_k \\
p(\mathbf{y}|\mathbf{z}, k, \theta) &= \mathcal{N}(\mathbf{y}|\mu_k + \Lambda_k \mathbf{z}, \Psi) \\
p(\mathbf{y}|\theta) &= \sum_k \int_{\mathbf{z}} p(k) p(\mathbf{z}) p(\mathbf{y}|\mathbf{z}, k, \theta) d\mathbf{z} \\
&= \sum_k \mathcal{N}(\mathbf{y}|\mu_k \ , \ \Lambda_k \Lambda_k^\top + \Psi)
\end{aligned}
$$

- Which is a *constrained mixture of Gaussians*.
- This is like a mixture of linear experts, using a logistic regression gate, eith missing inputs.
- Fitting procedure? EM, of course!
- see `ftp.cs.toronto.edu/pub/zoubin/tr-96-1.ps.gz`

## Geometric Intuition

- Since the latent variables are assumed to be independent, we are trying to find a linear transformation of the data that recovers these independent causes.
- Often we use *heavy tailed* source priors, e.g. $p(z_i) \propto 1/\cosh(z_i)$.
- Geometric intuition: finding spikes in histograms.



## ICA Model

- The simplest form of ICA has as many outputs as sources (square) and no sensor noise on the outputs:

$$p(\mathbf{z}) = \prod_k p(z_k)$$
$$\mathbf{y} = \mathbf{V}\mathbf{z}$$

- Learning in this case can be done with gradient descent (plus some "covariant" tricks to make the updates faster and more stable).
- If you keep the square $\mathbf{V}$ and use isotropic Gaussian noise on the outputs there is a simple EM algorithm, derived by Max Welling and Markus Weber.
- Much more complex cases have been studied also: nonsquare, convolutional, time delays in mixing, etc.
- But for that, we need to know about time-series...