CSC2515 Fall 2007
Introduction to Machine Learning

Lecture 9: Continuous Latent Variable Models

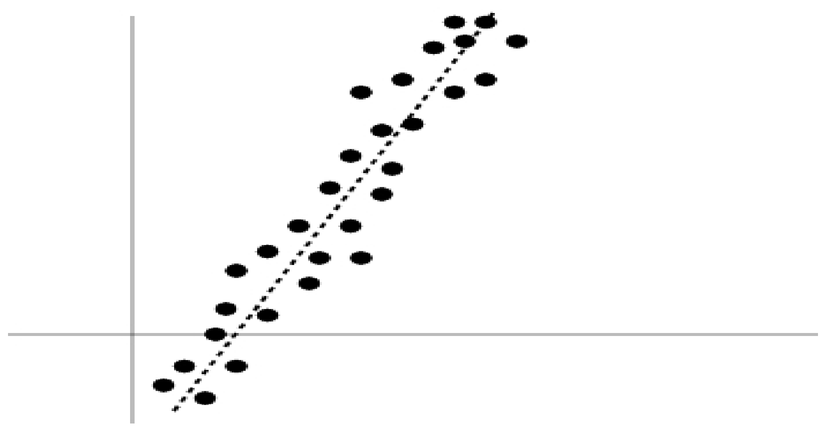
Example: continuous underlying variables

- What are the intrinsic latent dimensions in these two datasets?

- How can we find these dimensions from the data?

Dimensionality Reduction vs. Clustering

- Training continuous latent variable models often called *dimensionality reduction*, since there are typically many fewer latent dimensions
- Examples: Principal Components Analysis, Factor Analysis, Independent Components Analysis
- Continuous causes often more efficient at representing information than discrete
- 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 try to cluster the data, we need $2^{16} \sim= 10^5$ numbers

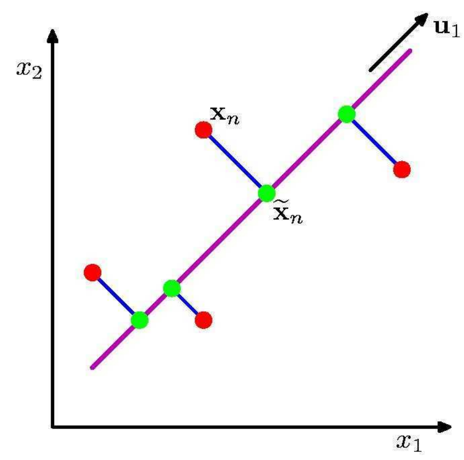


Generative View

- Each data example generated by first selecting a point from a distribution in the latent space, then generating a point from the conditional distribution in the input space
- Simple models: Gaussian distributions in both latent and data space, linear relationship betwixt
- This view underlies Probabilistic PCA, Factor Analysis
- Will return to this later

Standard PCA

- Used for data compression, visualization, feature extraction, dimensionality reduction
- Algorithm: to find *M* components underlying *D*-dimensional data
  - select the top *M* eigenvectors of **S** (data covariance matrix): $\{\mathbf{u}_1, ..., \mathbf{u}_M\}$
  - project each input vector **x** into this subspace, e.g., $z_{n1} = \mathbf{u}_1^T \mathbf{x}_n$



- Full projection onto M dimensions:

$$\begin{bmatrix} \mathbf{u}_1^\top \\ \cdots \\ \mathbf{u}_M^\top \end{bmatrix} [\mathbf{x}_1 \cdots \mathbf{x}_N] = [\mathbf{z}_1 \cdots \mathbf{z}_N]$$

- Two views/derivations:
  - Maximize variance (scatter of green points)
  - Minimize error (red-green distance per datapoint)

Standard PCA: Variance maximization

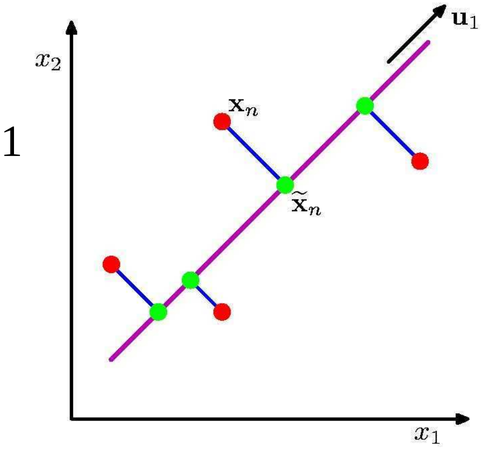
- One dimensional example
- Objective: maximize projected variance w.r.t. $\mathbf{U}_1$

$$\frac{1}{N}\sum_{n=1}^{N}\{\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}}\}^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

  – where sample mean and data covariance are:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$



- Must constrain $\|\mathbf{u}_1\|$: via Lagrange multiplier, maximize w.r.t $\mathbf{u}_1$

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda(1 - \mathbf{u}_1^T\mathbf{u}_1)$$

- Optimal $\mathbf{u}_1$ is principal component (eigenvector with maximal eigenvalue)

Standard PCA: Extending to higher dimensions

- Extend solution to additional latent components: find variance-maximizing directions orthogonal to previous ones
- Equivalent to Gaussian approximation to data
- Think of Gaussian as football (hyperellipsoid)
  - Mean is center of football
  - Eigenvectors of covariance matrix are axes of football
  - Eigenvalues are lengths of axes
- PCA can be thought of as fitting the football to the data: maximize volume of data projections in M-dimensional subspace
- Alternative formulation: minimize error, equivalent to minimizing average distance from datapoint to its projection in subspace

Standard PCA: Error minimization

- Data points represented by projection onto *M*-dimensional subspace, plus some distortion:
- Objective: minimize distortion w.r.t. $\mathbf{U}_1$ (*reconstruction error* of $\mathbf{x}_n$)

$$J = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \tilde{\mathbf{x}}_n^T||^2$$

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i$$

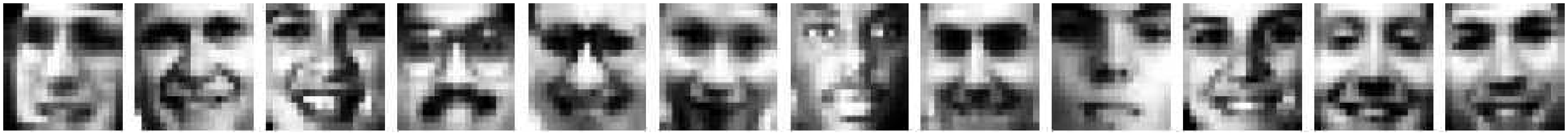
$$\begin{aligned} z_{nj} &= \mathbf{x}_n^T\mathbf{u}_j \\ b_j &= \bar{\mathbf{x}}^T\mathbf{u}_j \end{aligned}$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D} b_i(\mathbf{x}_n^T\mathbf{u}_i - \bar{\mathbf{x}}^T\mathbf{u}_i)^2 = \sum_{i=M+1}^{D}\mathbf{u}_i^T\mathbf{S}\mathbf{u}_i$$

- The objective is minimized when the *D*-*M* components are the eigenvectors of **S** with *lowest* eigenvalues $\rightarrow$ same result

Applying PCA to faces

- Need to first reduce dimensionality of inputs (will see in tutorial how to handle high-dimensional inputs) – down-sample images
- Run PCA on 2429 19x19 grayscale images (CBCL database)



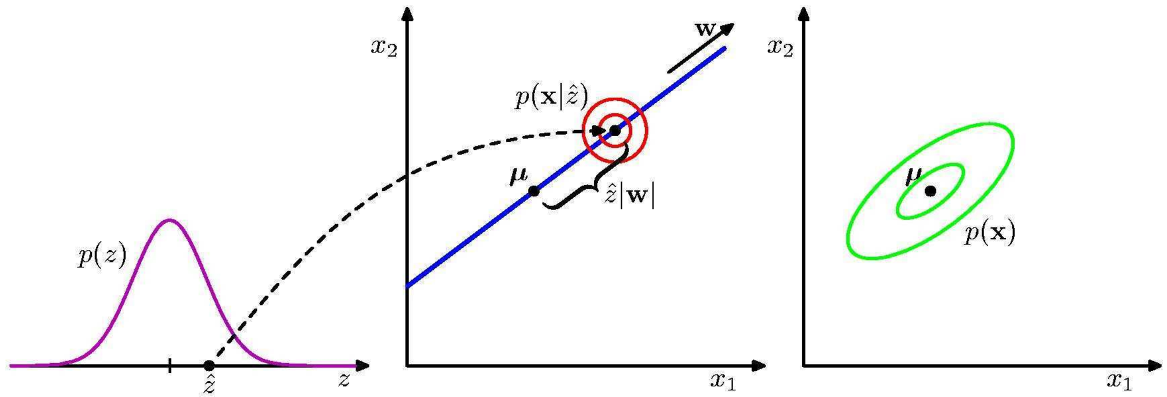
- Compresses the data: can get good reconstructions with only 3 components
- Pre-processing: can apply classifier to latent representation -- PPCA w/ 3 components obtains 79% accuracy on face/non-face discrimination in test data vs. 76.8% for m.o.G with 84 states
- Can be good for visualization

Applying PCA to faces: Learned basis

Probabilistic PCA

- Probabilistic, generative view of data
- Assumptions:
  - underlying latent variable has a Gaussian distribution
  - linear relationship between latent and observed variables
  - isotropic Gaussian noise in observed dimensions

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I}) \\
p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z}+\mu,\sigma^2\mathbf{I})
\end{aligned}
$$



$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon$$

Probabilistic PCA: Marginal data density

- Columns of **W** are the *principal components*, σ^2 is *sensor noise*
- Product of Gaussians is Gaussian: the joint p(**z**,**x**), the marginal data distribution p(**x**) and the posterior p(**z**|**x**) are also Gaussian
- Marginal data density (predictive distribution):

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})d\mathbf{z} = \mathcal{N}(\mathbf{x}|\mu, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})$$

- Can derive by completing square in exponent, or by just computing mean and covariance given that it is Gaussian:

$$\begin{aligned} E[\mathbf{x}] &= E[\mu + \mathbf{W}\mathbf{z} + \epsilon] = \mu + \mathbf{W}E[\mathbf{z}] + E[\epsilon] \\ &= \mu + \mathbf{W}0 + 0 = \mu \\ \mathbf{C} &= Cov[\mathbf{x}] = E[(\mathbf{z} - \mu)(\mathbf{z} - \mu)^T] \\ &= E[(\mu + \mathbf{W}\mathbf{z} + \epsilon - \mu)(\mu + \mathbf{W}\mathbf{z} + \epsilon - \mu)^T] \\ &= E[(\mathbf{W}\mathbf{z} + \epsilon)(\mathbf{W}\mathbf{z} + \epsilon)^T] \\ &= \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \end{aligned}$$

Probabilistic PCA: Joint distribution

- Joint density for PPCA (**x** is D-dim., **z** is M-dim):

$$p\left(\begin{bmatrix}\mathbf{z}\\ \mathbf{x}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{z}\\ \mathbf{x}\end{bmatrix} \mid \begin{bmatrix}0\\ \mu\end{bmatrix}, \begin{bmatrix} I & \mathbf{W}^\top \\ \mathbf{W} & \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}\end{bmatrix}\right)$$

  – where cross-covariance terms from:

$$\begin{aligned} Cov[\mathbf{z},\mathbf{x}] &= E[(\mathbf{z}-0)(\mathbf{x}-\mu)^T] = E[\mathbf{z}(\mu + \mathbf{W}\mathbf{z} + \epsilon - \mu)^T] \\ &= E[\mathbf{z}(\mathbf{W}\mathbf{z}+\epsilon)^T] = \mathbf{W}^T \end{aligned}$$

- Note that evaluating predictive distribution involves inverting **C**: reduce O(D^3) to O(M^3) by applying *matrix inversion lemma*:

$$\mathbf{C}^{-1} = \sigma^{-1}\mathbf{I} - \sigma^{-2}\mathbf{W}(\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^T$$

Probabilistic PCA: Posterior distribution

- Inference in PPCA produces posterior distribution over latent **z**
- Derive by applying Gaussian conditioning formulas (see 2.3 in book) to joint distribution

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

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

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\mathbf{m}_{1|2}, \mathbf{V}_{1|2})$$

$$\mathbf{m}_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2)$$

$$\mathbf{V}_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

$$\begin{aligned} p(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}|\mathbf{m}, \mathbf{V}) \\ \mathbf{m} &= \mathbf{W}^T(\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})^{-1}(\mathbf{x} - \mu) \\ \mathbf{V} &= \mathbf{I} - \mathbf{W}^T(\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})^{-1}\mathbf{W} \end{aligned}$$

- Mean of inferred **z** is projection of centered **x** **–** linear operation
- Posterior variance does not depend on the input **x** at all!

Standard PCA: Zero-noise limit of PPCA

- Can derive standard PCA as limit of Probabilistic PCA (PPCA) as $\sigma^2 \rightarrow 0$.
- ML parameters $\mathbf{W}^*$ are the same
- Inference is easier: orthogonal projection

$$\lim_{\sigma^2 \rightarrow 0} \mathbf{W}^T(\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{W}^T)^{-1} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T$$

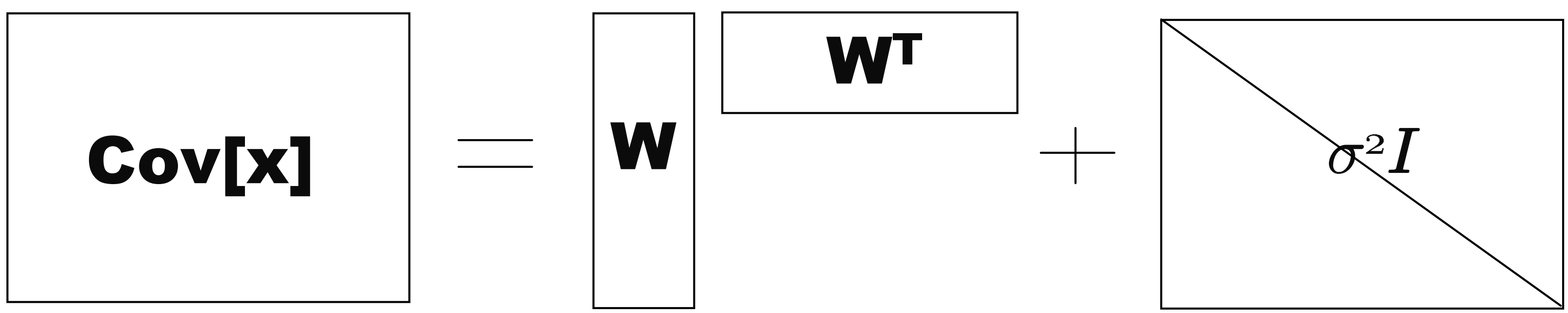
- Posterior covariance is zero

Probabilistic PCA: Constrained covariance

- Marginal density for PPCA (**x** is D-dim., **z** is M-dim):

$$p(\mathbf{x}|\theta) = \mathcal{N}(\mathbf{x}|\mu, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})$$

  - where $\theta = \mathbf{W}, \mu, \sigma$
- Effective covariance is low-rank outer product of two long skinny matrices plus a constant diagonal matrix



- So PPCA is just a constrained Gaussian model:
  - Standard Gaussian has D + D(D+1)/2 effective parameters
  - Diagonal-covariance Gaussian has D+D, but cannot capture correlations
  - PPCA: DM + 1 – M(M-1)/2, can represent M most significant correlations

Probabilistic PCA: Maximizing likelihood

$$
\begin{aligned}
L(\theta;\mathbf{X}) &= \log p(\mathbf{X}|\theta) = \sum_n \log p(\mathbf{x}_n|\theta) \\
&= -\frac{N}{2}\log|\mathbf{C}| - \frac{1}{2}\sum_n (\mathbf{x}_n - \mu)\mathbf{C}^{-1}(\mathbf{x}_n - \mu)^T \\
&= -\frac{N}{2}\log|\mathbf{C}| - \frac{1}{2}Tr[\mathbf{C}^{-1}\sum_n (\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^T] \\
&= -\frac{N}{2}\log|\mathbf{C}| - \frac{1}{2}Tr[\mathbf{C}^{-1}\mathbf{S}]
\end{aligned}
$$

- Fit parameters (θ = **W**, μ, σ) to max likelihood: make model covariance match observed covariance; distance is trace of ratio
- Sufficient statistics: mean $\mu = (1/N)\sum_n \mathbf{x}_n$ and sample covariance **S**
- Can solve for ML params directly: k^{th} column of **W** is the M^{th} largest eigenvalue of **S** times the associated eigenvector; σ^2 is the sum of all eigenvalues less than M^{th} one

Probabilistic PCA: EM

- Rather than solving directly, can apply EM
- Need complete-data log likelihood

$$\log p(\mathbf{X}, \mathbf{Z}|\mu, \mathbf{W}, \sigma^2) = \sum_n [\log p(\mathbf{x}_n|\mathbf{z}_n) + \log p(\mathbf{z}_n)]$$

- E step: compute expectation of complete log likelihood with respect to posterior of latent variables z, using current parameters – can derive $E[\mathbf{z}_n]$ and $E[\mathbf{z}_n \mathbf{z}_n^T]$ from posterior $p(\mathbf{z}|\mathbf{x})$
- M step: maximize with respect to parameters W and σ^2
- Iterative solution, updating parameters given current expectations, expectations give current parameters
- Nice property – avoids direct $O(ND^2)$ construction of covariance matrix, instead involves sums over data cases: $O(NDM)$; can be implemented online, without storing data

Probabilistic PCA: Why bother?

- Seems like a lot of formulas, algebra to get to similar model to standard PCA, but…
- Leads to understanding of underlying data model, assumptions (e.g., vs. standard Gaussian, other constrained forms)
- Derive EM version of inference/learning: more efficient
- Can understand other models as generalizations, modifications
- More readily extend to mixtures of PPCA models
- Principled method of handling missing values in data
- Can generate samples from data distribution

Factor Analysis

- Can be viewed as generalization of PPCA
- Historical aside – controversial method, based on attempts to interpret factors: e.g., analysis of IQ data identified factors related to race
- Assumptions:
  - underlying latent variable has a Gaussian distribution
  - linear relationship between latent and observed variables
  - *diagonal* Gaussian noise in data dimensions

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \\
p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \mathbf{\Psi})
\end{aligned}
$$

- W: *factor loading matrix* (*D* x *M*)
- Ψ : data covariance (diagonal, or axis-aligned; vs. PCA's spherical)

Factor Analysis: Distributions

- As in PPCA, the joint p(**z**,**x**), the marginal data distribution p(**x**) and the posterior p(**z**|**x**) are also Gaussian
- Marginal data density (predictive distribution):

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})d\mathbf{z} = \mathcal{N}(\mathbf{x}|\mu, \mathbf{W}\mathbf{W}^T + \mathbf{\Psi})$$

- Joint density:

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

- Posterior, derived via Gaussian conditioning

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

Factor Analysis: Optimization

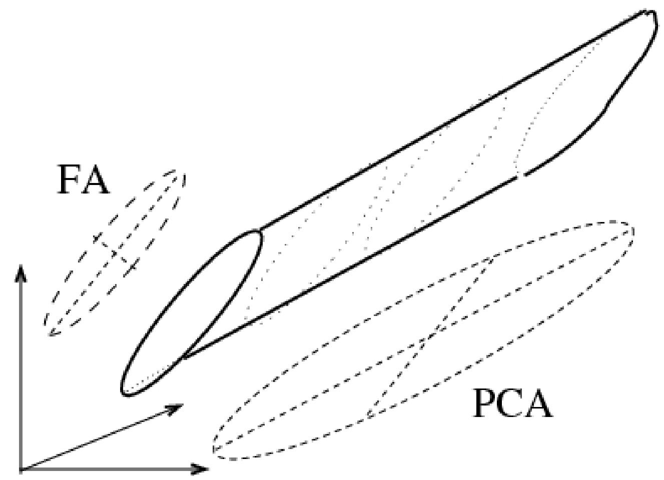
- Parameters are coupled, making it impossible to solve for ML parameters directly, unlike PCA
- Must use EM, or other nonlinear optimization
- E step: compute posterior $p(\mathbf{z}|\mathbf{x})$ – use matrix inversion to convert D x D matrix inversions to M x M
- M step: take derivatives of expected complete log likelihood with respect to parameters

Factor Analysis vs. PCA: Rotations

- In PPCA, the data can be rotated without changing anything: multiply data by matrix **Q,** obtain same fit to data

$$
\begin{aligned}
\mu &\leftarrow \mathbf{Q}\mu \\
\mathbf{W} &\leftarrow \mathbf{QW} \\
\Psi &\leftarrow \Psi
\end{aligned}
$$

- But the scale is important
- PCA looks for directions of large variance, so it will grab large noise directions

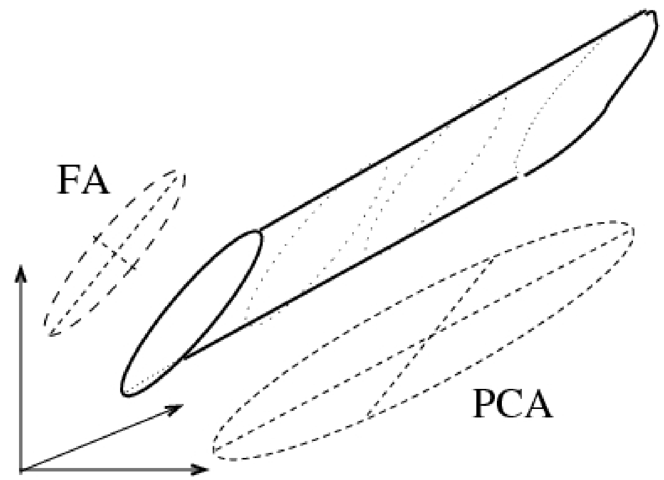


Factor Analysis vs. PCA: Scale

- In FA, the data can be re-scaled without changing anything
- Multiply x_i by α_i:

$$
\begin{aligned}
\mu_i &\leftarrow \alpha_i \mu_i \\
\mathbf{W}_{ij} &\leftarrow \alpha_i \mathbf{W}_{ij} \\
\Psi_i &\leftarrow \alpha_i^2 \Psi_i
\end{aligned}
$$

- But rotation in data space is important
- FA looks for directions of large correlation in the data, so it will not model large variance noise

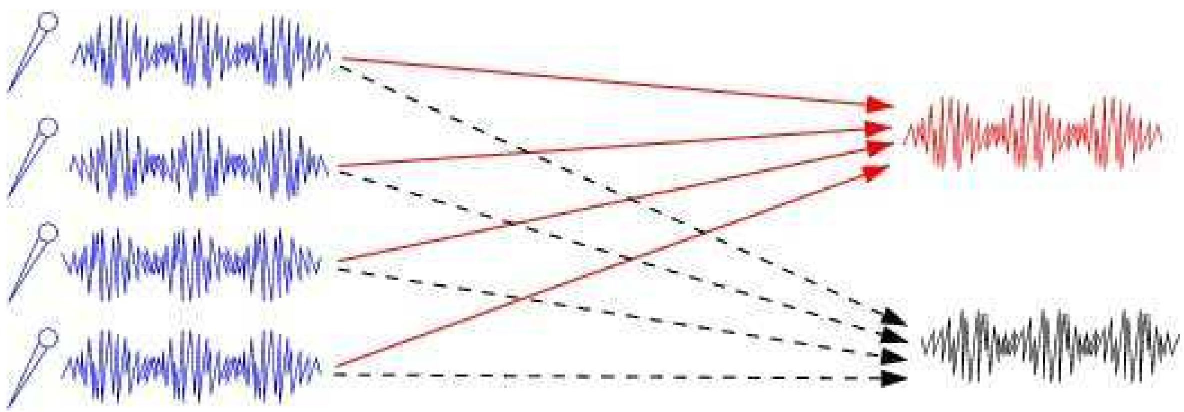


Factor Analysis : Identifiability

- Factors in FA are *non-identifiable:* not guaranteed to find same set of parameters – not just local minimum but invariance
- Rotate $\mathbf{W}$ by any unitary $\mathbf{Q}$ and model stays the same – $\mathbf{W}$ only appears in model as outer product $\mathbf{W}\mathbf{W}^T$
- Replace $\mathbf{W}$ with $\mathbf{WQ}$: $(\mathbf{WQ})(\mathbf{WQ})^T = \mathbf{W}(\mathbf{Q}\,\mathbf{Q}^T)\,\mathbf{W}^T = \mathbf{W}\mathbf{W}^T$
- So no single best setting of parameters
- Degeneracy makes unique interpretation of learned factors impossible

Independent Components Analysis (ICA)

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



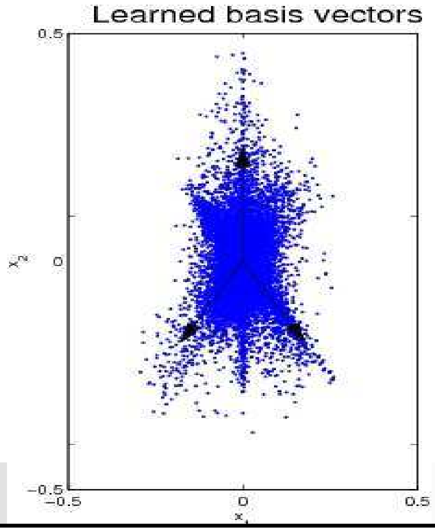
- Learning problem same as before: find weights from factors to observations, infer the unknown factor values for given input
- ICA: factors are called "sources", learning is "unmixing"

ICA Intuition

- Since latent variables assumed to be independent, trying to find linear transformation of data that recovers independent causes
- Avoid degeneracies in Gaussian latent variable models: assume non-Gaussian prior distribution for latents (sources)
- Often we use *heavy-tailed* source priors, e.g.,

$$p(z_j) = \frac{1}{\pi \cosh(z_j)} = \frac{1}{\pi(\exp(z_j)+\exp(-z_j))}$$

- Geometric intuition: find spikes in histogram



ICA Details

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

$$
\begin{aligned}
p(\mathbf{z}) &= \prod_k p(z_k) \\
\mathbf{x} &= \mathbf{V}\mathbf{z}
\end{aligned}
$$

- Learning in this case can be done with gradient descent (plus some "covariant" tricks to make updates faster and more stable)
- If keep **V** square, and assume isotropic Gaussian noise on the outputs, there is a simple EM algorithm
- Much more complex cases have been studied also: non-square, time delays, etc.