

# Continuous Latent Variable Models

- Often there are some **unknown underlying causes** of the data.
- So far we have looked at models with discrete latent variables, such as mixture of Gaussians.
- Sometimes, it is more appropriate to think in terms of **continuous factors** which control the data we observe.
- **Motivation:** for many datasets, data points lie close to a manifold of **much lower dimensionality** compared to that of the original data space.
- 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

# Intrinsic Latent Dimensions

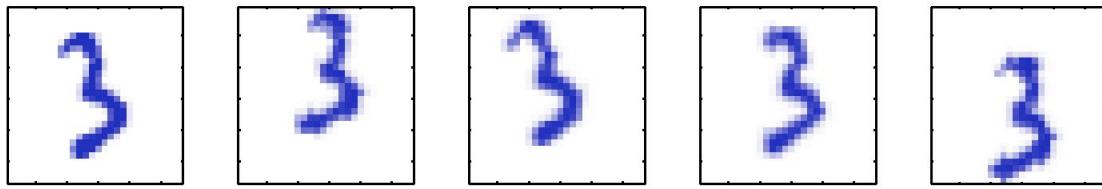
- What are the **intrinsic latent dimensions** in these two datasets?



- How can we find these latent dimensions from this high-dimensional data.

# Intrinsic Latent Dimensions

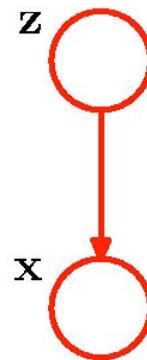
- In this dataset, there is only **3 degrees of freedom of variability**, corresponding to vertical and horizontal translations, and the rotations.



- Each image undergoes a random displacement and rotation within some larger image field.
- The resulting images have  $100 \times 100 = 10,000$  pixels.

# 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
- Simplest latent variable models: Assume Gaussian distribution for both latent and observed variables.
- This leads to probabilistic formulation of the Principal Component Analysis and Factor Analysis.
- We will first look at standard PCA, and then consider its probabilistic formulation.
- Advantages of probabilistic formulation: use of EM for parameter estimation, mixture of PCAs, Bayesian PCA.



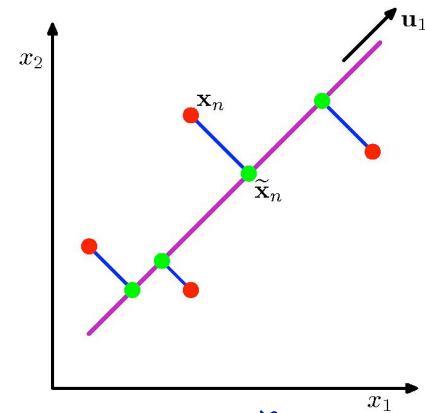
# Principal Component Analysis

- Used for data compression, visualization, feature extraction, dimensionality reduction.

投影尽可能保留更多信息

- The goal is find M principal components underlying D-dimensional data

- select the top M **eigenvectors** of  $\mathbf{S}$  (data covariance matrix):  $\{\mathbf{u}_1, \dots, \mathbf{u}_M\}$ .
- project each input vector  $\mathbf{x}$  into **this subspace**,  
e.g.  $z_{n1} = \mathbf{x}_n^T \mathbf{u}_1$ .



- Full projection into M dimensions takes form:

$$\begin{bmatrix} \mathbf{u}_1^\top \\ \vdots \\ \mathbf{u}_M^\top \end{bmatrix} [\mathbf{x}_1 \cdots \mathbf{x}_N] = [z_1 \cdots z_N]$$

投影后使方差可能大 → 信息重构

- Two views/derivations:

- **Maximize variance** (scatter of green points).
- **Minimize error** (red-green distance per data point).

中, 10/18

# Maximum Variance Formulation

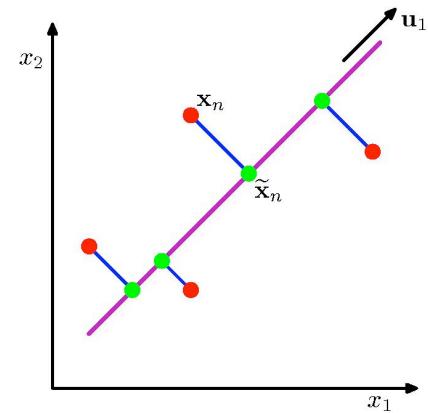
- Consider a dataset  $\{x_1, \dots, x_N\}$ ,  $x_n \in \mathbb{R}^D$ . Our goal is to project data onto a space having dimensionality  $M < D$ .
- Consider the projection into  $M=1$  dimensional space.
- Define the direction of this space using a  $D$ -dimensional unit vector  $u_1$ , so that  $u_1^T u_1 = 1$ .
- Objective:** maximize the variance of the projected data with respect to  $u_1$ .

$$\frac{1}{N} \sum_{n=1}^N \{u_1^T x_n - u_1^T \bar{x}\}^2 = u_1^T S u_1$$

where sample mean and data covariance is given by:

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

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



# Maximum Variance Formulation

- Maximize the variance of the projected data:

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

- Must constrain  $\|\mathbf{u}_1\| = 1$ . Using Langrange multiplier, maximize:

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

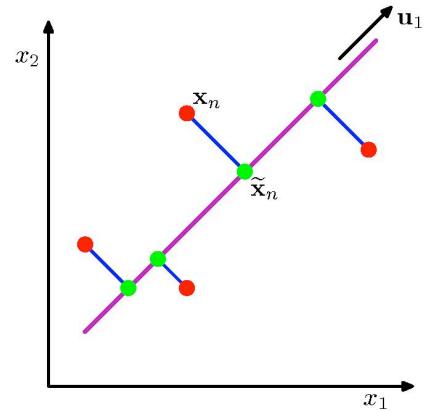
- Setting the derivative with respect to  $\mathbf{u}_1$  to zero:

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1.$$

- Hence  $\mathbf{u}_1$  must be an eigenvector of  $\mathbf{S}$ .
- The maximum variance of the projected data is given by:

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \lambda_1.$$

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



# Minimum Error Formulation

- Introduce a complete orthonormal set of D-dimensional basis vectors:  
 $\{\mathbf{u}_1, \dots, \mathbf{u}_D\}$  :

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}.$$

- Without loss of generality, we can write:

$$\mathbf{x}_n = \sum_{i=1}^D \alpha_{ni} \mathbf{u}_i, \quad \alpha_{ni} = \mathbf{x}_n^T \mathbf{u}_i.$$



Rotation of the coordinate system to a new system defined by  $\mathbf{u}_i$ .

- Our goal is to represent data points by the projection into M-dimensional subspace (plus some distortion):
- Represent M-dim linear subspace by the first M of the basis vectors:

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

# Minimum Error Formulation

- Represent M-dim linear subspace by the first M of the basis vectors:

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

where  $z_{ni}$  depend on the particular data point and  $b_i$  are constants.

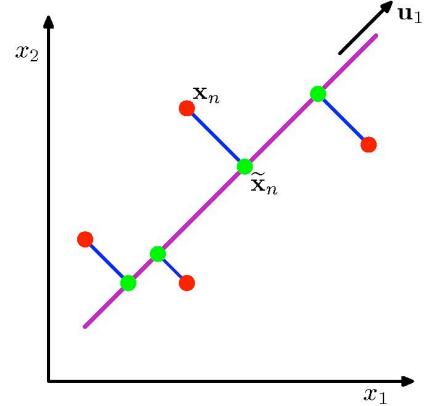
- Objective:** minimize distortion with respect to  $\mathbf{u}_i$ ,  $z_{ni}$ , and  $b_i$ .

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

- Minimizing with respect to  $z_{nj}$ ,  $b_j$ :
- $$\begin{aligned} z_{nj} &= \mathbf{x}_n^T \mathbf{u}_j \\ b_j &= \bar{\mathbf{x}}^T \mathbf{u}_j \end{aligned}$$

- Hence, the objective reduces to:

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



# Minimum Error Formulation

- Minimize distortion with respect to  $\mathbf{u}_i$ : constraint minimization problem:

$$J = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \sum_{i=M+1}^D \mathbf{u}_i^T \mathbf{S} \mathbf{u}_i.$$

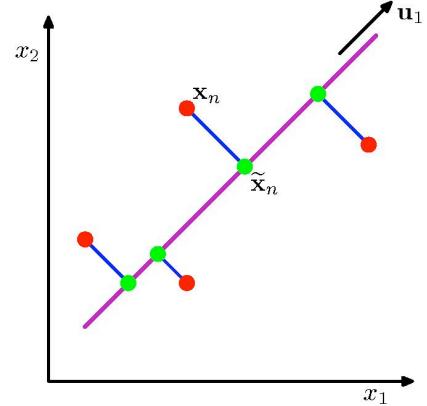
- The general solution is obtained by choosing  $\mathbf{u}_i$  to be eigenvectors of the covariance matrix:

$$\mathbf{S} \mathbf{u}_i = \lambda_i \mathbf{u}_i.$$

- The distortion is then given by:  $J = \sum_{i=M+1}^D \lambda_i.$

- The objective is minimized when the remaining D-M components are the eigenvectors of  $\mathbf{S}$  with **lowest eigenvalues** → same result.

- We will later see a generalization: deep autoencoders.



# Applications of PCA

- Run PCA on 2429 19x19 grayscale images (CBCL database)



- **Data compression:** We can get good reconstructions with only 3 components.
- **Pre-processing:** We can apply a **standard classifier to latent representation** -- PCA with 3 components obtains 79% accuracy on face/non-face discrimination in test data vs. 76.8% for mixture of Gaussians with 84 components.
- **Data visualization:** by projecting the data onto the first two principal components.

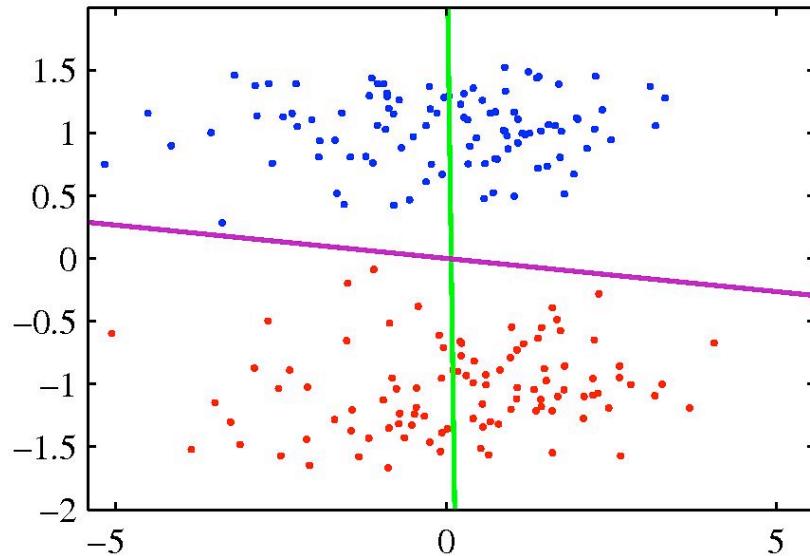
# Learned Basis

- Run PCA on 2429 19x19 grayscale images (CBCL database)



# PCA vs. Fisher's LDA

- A comparison of PCA with Fisher's LDA for **linear dimensionality reduction**.



- PCA chooses **direction of maximum variance** (magenta curve) leading to strong class overlap (unsupervised).
- LDA takes into account the **class labels** (supervised), leading to a projection into the green curve.

# PCA for High-Dimensional Data

- In some applications of PCA, the number of data points is smaller than the dimensionality of the data space, i.e.  $N < D$ .
- In so far, we need to find the eigenvectors of the  $D \times D$  data covariance matrix  $\mathbf{S}$ , which scales as  $O(D^3)$ .
- Direct application of PCA will often be computationally infeasible.
- Solution: Let  $\mathbf{X}$  be the  $N \times D$  centered data matrix. The corresponding eigenvector equation becomes:

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \mathbf{u}_i = \lambda_i \mathbf{u}_i.$$

- Pre-multiply by  $\mathbf{X}$ :

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{u}_i) = \lambda_i (\mathbf{X} \mathbf{u}_i).$$

# PCA for High-Dimensional Data

- Define  $v_i = Xu_i$ , and hence we have:

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T \mathbf{v}_i = \lambda_i \mathbf{v}_i.$$

- This is an **eigenvector equation** for the  $N \times N$  matrix
- It has the same  $N-1$  eigenvalues as the original data covariance matrix  $\mathbf{S}$  (which itself has **an additional  $D-N+1$  zero eigenvalues**).
- Computational cost scales as  $O(N^3)$  rather than  $O(D^3)$ .
- To determine eigenvectors, we multiply by  $\mathbf{X}^T$ :

$$\left( \frac{1}{N} \mathbf{X}^T \mathbf{X} \right) (\mathbf{X}^T \mathbf{v}_i) = \lambda_i \mathbf{X}^T \mathbf{v}_i.$$

- Hence  $\mathbf{X}^T \mathbf{v}_i$  is **an eigenvector of  $\mathbf{S}$  with eigenvalue  $\lambda_i$** .
- These eigenvectors may not be normalized.

# Probabilistic PCA

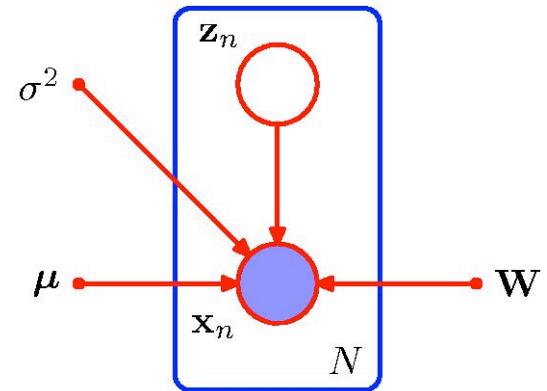
- Probabilistic, generative view of data.
- Key advantages of probabilistic PCA (PPCA):
  - It represents a **constrained form** of the Gaussian distribution.
  - We can **derive EM algorithm** for PCA which is computationally efficient.
  - PPCA allows us to deal with **missing values** in the data set.
  - We can formulate **mixture of PPCAs** in a principled way.
  - PPCA forms the basis for a **Bayesian PCA**, in which the dimensionality of the principal subspace can be determined from the data.
  - The **existence of a likelihood function** allows direct comparisons with other probabilistic density models
  - PPCA can be used to model class conditional densities and hence it can be applied to **classification problems**.

# Probabilistic PCA

- Key assumptions:

- underlying latent M-dim variable  $\mathbf{z}$  has a Gaussian distribution.
- linear relationship between M-dim latent  $\mathbf{z}$  and D-dim observed  $\mathbf{x}$  variables.
- isotropic Gaussian noise in observed dimensions

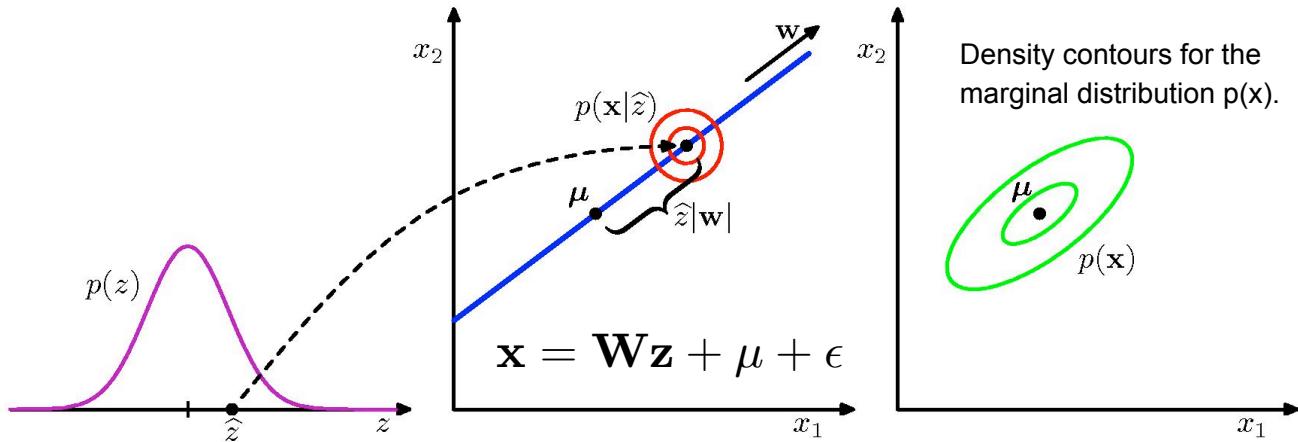
$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \\ p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \end{aligned}$$



- Hence the mean of  $\mathbf{x}$  is a linear function of  $\mathbf{z}$  governed by the  $D \times M$  matrix  $\mathbf{W}$  and the  $D$ -dim vector  $\boldsymbol{\mu}$ .
- We will see that the columns of  $\mathbf{W}$  span the principal subspace of the data space (Columns of  $\mathbf{W}$  are the *principal components*,  $\sigma^2$  is *sensor noise*).

# Generative View of PPCA

- Generative view of the PPCA for a 2-d data space and 1-d latent space:



- Draw a value of the latent variable from its prior distribution:

$$\hat{z} \sim p(z)$$

- Draw a value for  $\mathbf{x}$  from an isotropic Gaussian distribution:

$$\hat{x} \sim p(\mathbf{x}|\hat{z}) = \mathcal{N}(\mathbf{x}|\mathbf{w}\hat{z} + \mu, \sigma^2 I).$$

# Marginal Data Density

- The joint  $p(\mathbf{z}, \mathbf{x})$ , the marginal data distribution  $p(\mathbf{x})$  and the posterior distribution  $p(\mathbf{z}|\mathbf{x})$  are also Gaussian.
- **Marginal data density** (also known as 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 this result **directly by computing mean and covariance** given that it is Gaussian:

$$\begin{aligned} E[\mathbf{x}] &= E[\mu + \mathbf{W}\mathbf{z} + \epsilon] = \mu + \mathbf{W}E[\mathbf{z}] + E[\epsilon] \\ &= \mu + \mathbf{W}\mathbf{0} + \mathbf{0} = \mu \end{aligned}$$

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

# Redundancy in Parameterization

- The marginal distribution is governed by parameters  $\mathbf{W}$ ,  $\mu$ ,  $\sigma^2$ :

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

- Redundancy in parameterization: rotation of the latent space coordinates.
- Let  $\mathbf{R}$  be an orthogonal matrix, then define a new matrix:

$$\tilde{\mathbf{W}} = \mathbf{WR}, \quad \mathbf{RR}^T = \mathbf{I}.$$

- Then

$$\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T = \mathbf{WR}\mathbf{R}^T\mathbf{W}^T = \mathbf{WW}^T.$$

- There is a whole family of matrices all of which give rise to the same marginal distribution.
- Rotations within the latent space.

# Joint Density for PPCA

- Joint density for PPCA, where  $\mathbf{x}$  is D-dim and  $\mathbf{z}$  is M-dim is given:

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

where cross covariance term forms:

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

- When evaluating marginal distribution, we need to invert a  $D \times D$  matrix  $\mathbf{C}$ , which can be expensive.
- Reduce  $O(D^3)$  to  $O(M^3)$  by applying *matrix inversion lemma*:

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

# Posterior Distribution for PPCA

- Inference in PPCA amounts to computing posterior distribution over latent variables:

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{m}, \mathbf{V})$$

$$\mathbf{m} = \mathbf{M}^{-1} \mathbf{W}^T (\mathbf{x} - \boldsymbol{\mu}),$$

$$\mathbf{V} = \sigma^2 \mathbf{M}^{-1},$$

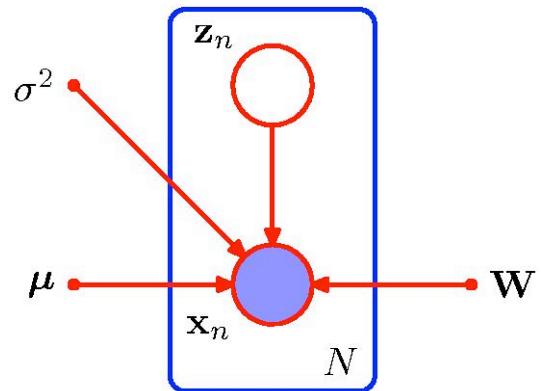
$$\mathbf{M} = \mathbf{W}^T \mathbf{W} + \sigma^2 \mathbf{I}.$$

- Mean of inferred  $\mathbf{z}$  is projection of centered  $\mathbf{x}$ :  
linear operation.

- Posterior variance does not depend on the input  $\mathbf{x}$  at all.

- Remember:

$$\mathbf{C} = \mathbf{W} \mathbf{W}^T + \sigma^2 \mathbf{I}. \quad \underbrace{\mathbf{W} \mathbf{W}^T}_{\text{M matrix}}$$
$$\mathbf{C}^{-1} = \sigma^{-2} \mathbf{I} - \sigma^{-2} \mathbf{W} (\mathbf{W}^T \mathbf{W} + \sigma^2 \mathbf{I})^{-1} \mathbf{W}^T$$

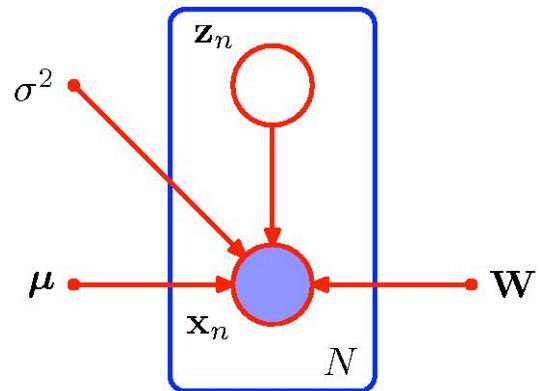


# Constrained Covariance

- Marginal density for PPCA has the following form:

$$p(\mathbf{x}|\theta) = \mathcal{N}(\mathbf{x}|\mu, \underbrace{\mathbf{WW}^T + \sigma^2 \mathbf{I}}_{\text{Covariance } \mathbf{C}})$$

where  $\theta = \{\mathbf{W}, \mu, \sigma^2\}$ .



- The covariance is low-rank outer product of two long skinny matrices plus a constant diagonal matrix:

$$\mathbf{Cov}[\mathbf{x}] = \mathbf{W} \mathbf{W}^T + \sigma^2 \mathbf{I}$$

- Hence PPCA is a constrained Gaussian model.
- We can fit model parameters using maximum likelihood.

# Maximum Likelihood

- Model parameters can be determined using maximum likelihood (by integrating our latent variables):

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

- Maximizing with respect to the mean:  $\boldsymbol{\mu}_{ML} = \bar{\mathbf{x}}$ .

- We then have:

$$\log p(\mathbf{X}|\theta) = -\frac{N}{2} \log |\mathbf{C}| - \frac{1}{2} \text{Tr}[\mathbf{C}^{-1} \mathbf{S}] + \text{const.}$$

- Maximizing with respect to  $\mathbf{W}$  and  $\sigma^2$  can be solved directly.

# Maximum Likelihood

- Objective:

$$\log p(\mathbf{X}|\theta) = -\frac{N}{2} \log |\mathbf{C}| - \frac{1}{2} \text{Tr} [\mathbf{C}^{-1} \mathbf{S}] + \text{const.}$$

- $\mathbf{C}$  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.
- Sufficient statistics: mean  $\bar{\mathbf{x}} = \frac{1}{N} \sum_n \mathbf{x}_n$  and sample covariance  $\mathbf{S}$ .

# Maximum Likelihood

- Objective:

$$\log p(\mathbf{X}|\theta) = -\frac{N}{2} \log |\mathbf{C}| - \frac{1}{2} \text{Tr} [\mathbf{C}^{-1} \mathbf{S}] + \text{const.}$$

- Maximizing with respect to  $\mathbf{W}$ :

$$\mathbf{W}_{ML} = \mathbf{U}_M (\mathbf{L}_M - \sigma^2 \mathbf{I})^{1/2} \mathbf{R},$$

where

- $\mathbf{U}_M$  is a  $D \times M$  matrix whose columns are given by the **M principal eigenvectors** of the data covariance matrix  $\mathbf{S}$ .
- $\mathbf{L}_M$  is the  $M \times M$  diagonal matrix containing **M largest eigenvalues**.
- $\mathbf{R}$  is an arbitrary  $M \times M$  **orthogonal matrix**.
- If the eigenvectors have been arranged in the order of decreasing values of the corresponding eigenvalues, then the columns of  $\mathbf{W}$  **define the principal subspace of standard PCA**.

# Maximum Likelihood

- Objective:

$$\log p(\mathbf{X}|\theta) = -\frac{N}{2} \log |\mathbf{C}| - \frac{1}{2} \text{Tr}[\mathbf{C}^{-1} \mathbf{S}] + \text{const.}$$

- Maximizing with respect to  $\sigma^2$ :

$$\sigma_{ML}^2 = \frac{1}{D-M} \sum_{i=M+1}^D \lambda_i,$$

which is the average variance associated with the discarded dimensions.

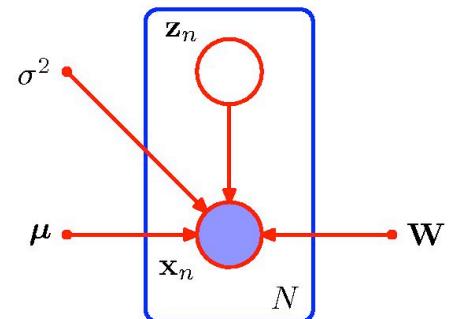
# EM for PPCA

- Instead of solving directly, we can use EM. The EM can be scaled to very large high-dimensional datasets.
- The complete-data log-likelihood takes form:

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

- **E-step:** compute expectation of complete log likelihood with respect to posterior of latent variables  $\mathbf{z}$ , using current parameters.

- We need to derive  $\mathbb{E}[\mathbf{z}_n]$ ,  $\mathbb{E}[\mathbf{z}_n \mathbf{z}_n^T]$  with respect to the true posterior:  $p(\mathbf{z} | \mathbf{X})$ .



- **M-step:** maximize with respect to parameters  $\mathbf{W}$  and  $\sigma^2$ .

- Appealing property: EM avoids direct  $O(ND^2)$  construction of covariance matrix!
- Instead EM involves sums over data cases:  $O(NDM)$ . It can also be implemented online, without storing data.

# Zero Noise Limit

- We can derive standard PCA as a **limit of probabilistic PCA** as the noise term goes to zero:  $\sigma^2 \rightarrow 0$ .
- ML parameters are the same.
- Inferring the distribution over latent variables is easier: The posterior mean reduces to:

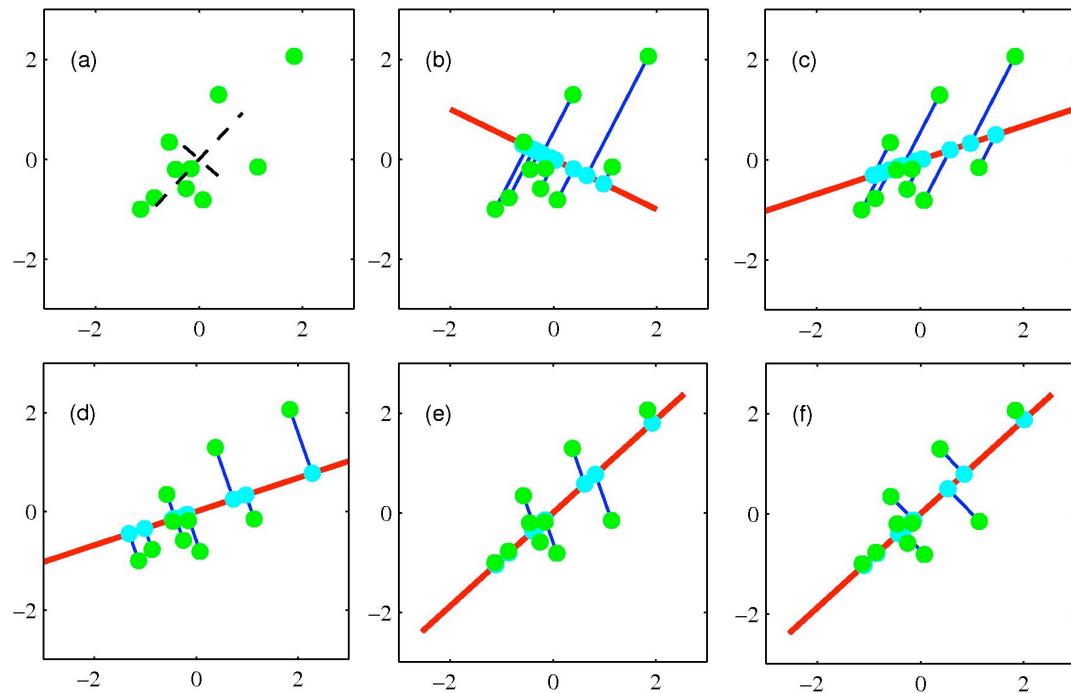
$$\lim_{\sigma^2 \rightarrow 0} (\mathbf{W}^T \mathbf{W} + \sigma \mathbf{I})^{-1} \mathbf{W}^T (\mathbf{x} - \boldsymbol{\mu}) = (\mathbf{W}^T \mathbf{W})^{-1} \mathbf{W}^T (\mathbf{x} - \boldsymbol{\mu}),$$

which represents an orthogonal projection of the data point onto the latent space – standard PCA.

- Posterior covariance goes to zero:

# EM for PPCA

- EM algorithm for PCA.



# Bayesian PCA

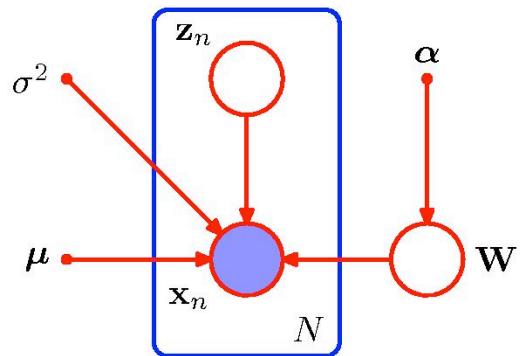
- It is easy to take a **Bayesian viewpoint** and place priors over model parameters.
- One option is to employ the **evidence approximation** (empirical Bayes) framework.
- We can define an independent Gaussian prior over each column of  $\mathbf{W}$ .
- Each such Gaussian has an **independent variance**:

$$p(\mathbf{W}|\alpha) = \prod_{i=1}^M \left( \frac{\alpha_i}{2\pi} \right) \exp \left[ -\frac{1}{2} \alpha_i \mathbf{w}_i^T \mathbf{w}_i \right],$$

where  $\mathbf{w}_i$  is the  $i^{\text{th}}$  column of  $\mathbf{W}$ .

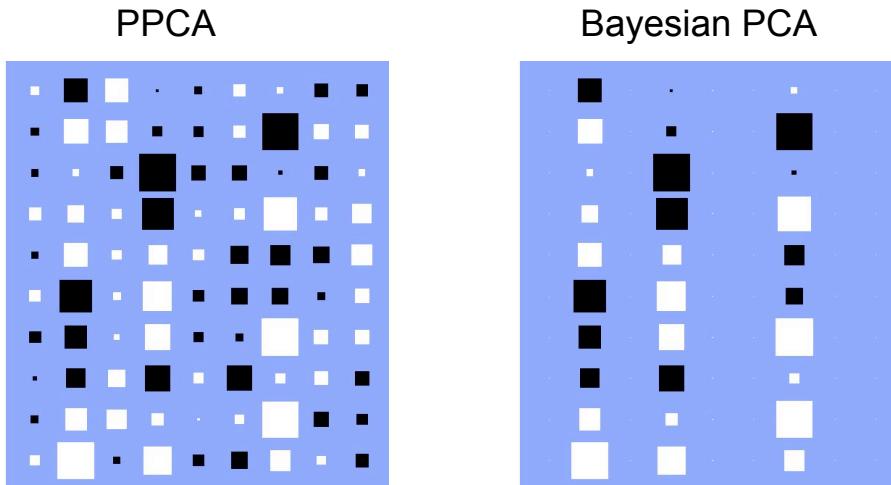
- The values of  $\alpha_i$  are re-estimated during training by **maximizing the marginal likelihood**:

$$p(\mathbf{X}|\alpha, \mu, \sigma^2) = \int p(\mathbf{X}|\mathbf{W}, \mu, \sigma^2) p(\mathbf{W}|\alpha) d\mathbf{W}.$$



# Example of Bayesian PCA

- Hinton diagram of the matrix  $W$ : each element of  $W$  is depicted as a square (white for positive and black for negative).



- The synthetic dataset contains 300 points in  $D=10$  space with the **intrinsic dimensionality** set to  $D=3$ .
- Bayesian PCA discovers appropriate dimensionality.

# Factor Analysis

- Linear Gaussian latent variable model that is closely related to PPCA.
- Key assumptions:
  - underlying latent M-dim variable  $\mathbf{z}$  has a Gaussian distribution
  - linear relationship between M-dim latent  $\mathbf{z}$  and D-dim observed  $\mathbf{x}$  variables.
  - diagonal Gaussian noise in observed dimensions.

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

- $\mathbf{W}$  is a  $D \times M$  factor loading matrix.
- $\boldsymbol{\Psi}$  is a  $M \times M$  diagonal matrix (or axis-aligned).
- The only difference between PPCA and FA is that in Factor Analysis the conditional distribution of the observed variable  $\mathbf{x}$  has diagonal rather than isotropic covariance.

# Factor Analysis: Distributions

- As in PPCA, the joint  $p(\mathbf{z}, \mathbf{x})$ , the **marginal data distribution**  $p(\mathbf{x})$  and the posterior  $p(\mathbf{z}|\mathbf{x})$  are also Gaussian.
- **Marginal distribution** (predictive distribution):

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

- The **joint distribution**:

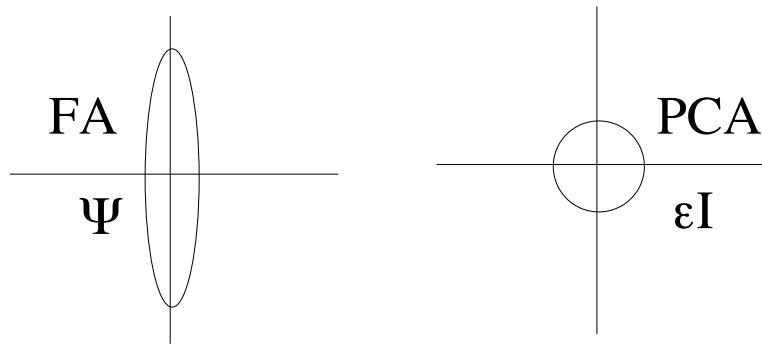
$$p(\begin{bmatrix} \mathbf{z} \\ \mathbf{x} \end{bmatrix}) = \mathcal{N}(\begin{bmatrix} \mathbf{z} \\ \mathbf{x} \end{bmatrix} | \begin{bmatrix} 0 \\ \mu \end{bmatrix}, \begin{bmatrix} I & \mathbf{W}^\top \\ \mathbf{W} & \mathbf{W}\mathbf{W}^\top + \Psi \end{bmatrix})$$

# Factor Analysis: Optimization

- Parameters are coupled, which makes it impossible to solve for ML parameters directly, unlike in probabilistic PCA.
- Because FA is a latent variable model, we can use EM, or other nonlinear optimization
- E-step: compute posterior  $p(\mathbf{z}|\mathbf{x})$ : Use matrix inversion to convert  $D \times D$  matrix inversions to  $M \times M$ .
- M-step: take derivatives of the expected complete log likelihood with respect to parameters.
- Bayesian treatment of the factor analysis can be obtained by a straightforward extension of standard FA (as we did for PPCA).

# FA vs. PCA

- intuition: **Gaussians** are hyperellipsoids.
- Mean == center 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  $\sigma^2$** .

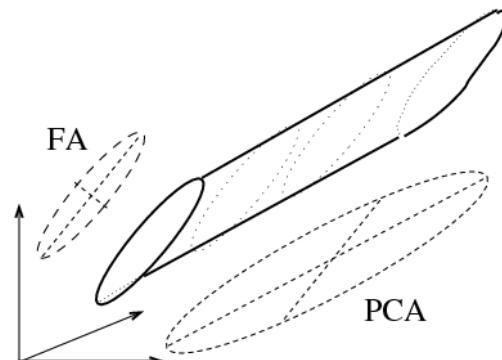


# Rotation Invariance in PCA

- In PPCA the **rotation of the data is unimportant**: we can multiply the data  $\mathbf{x}$  by a rotation matrix  $\mathbf{Q}$  without changing anything:

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

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



# Scale Invariance in FA

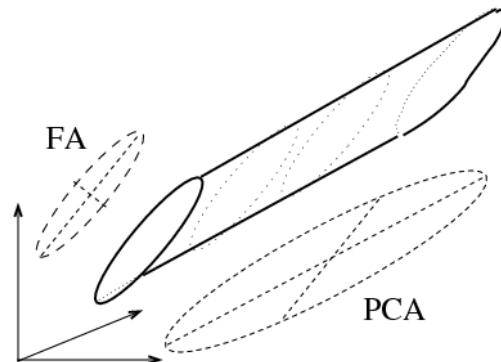
- In FA, the data can be re-scaled without changing anything.
- Multiply  $x_i$  by  $\alpha_i$ :

$$\mu_i \leftarrow \alpha_i \mu_i$$

$$\mathbf{W}_{ij} \leftarrow \alpha_i \mathbf{W}_{ij}$$

$$\Psi_i \leftarrow \alpha_i^2 \Psi_i$$

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



# Model 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  $\mathbf{WW}^T$

$$(\mathbf{WQ})(\mathbf{WQ})^T = \mathbf{WW}^T.$$

- This means that there is **no “one best” setting of the parameters**. An infinite number of parameters all give the ML score.
- Degeneracy **makes unique interpretation of learned factors impossible**.

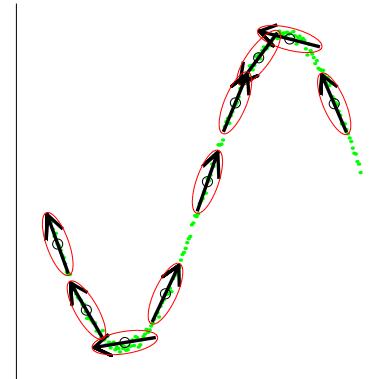
# Mixture of Dimensionality Reducers

- The next logical step is to consider 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**.
- Example: Mixture of Factor Analyzers:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}), \quad p(k) = \pi_k,$$

$$p(\mathbf{x}|\mathbf{z}, k, \theta) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k + W_k \mathbf{z}, \Psi),$$

$$\begin{aligned} p(\mathbf{x}|\theta) &= \sum_k \int_{\mathbf{z}} p(k)p(\mathbf{z})p(\mathbf{x}|\mathbf{z}, k, \theta)d\mathbf{z} \\ &= \sum_k \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, W_k W_k^T + \Psi). \end{aligned}$$

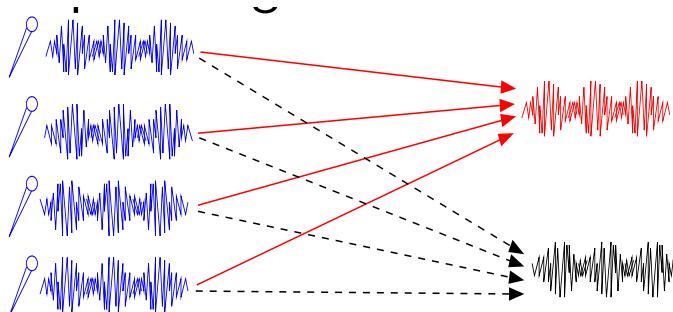


which is **constrained mixture of Gaussians**.

- Fitting is done via EM algorithm.

# Independent Components Analysis

- 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 small most of the time**, and do not interact with each other.
- Example: **Mixture of speech signals**.



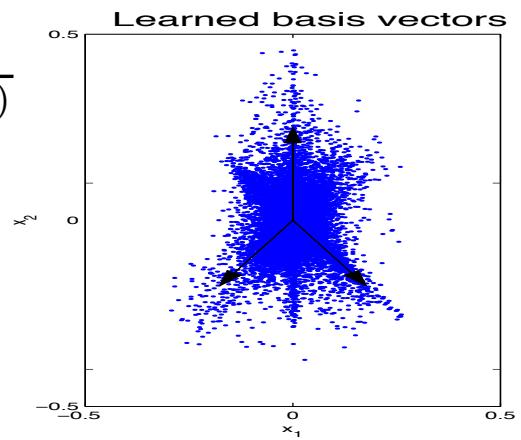
- The learning problem is the same: **find the weights from the factors to the outputs and infer the unknown factor values**.
- ICA: the factors are sometimes called “**sources**”, and the learning is sometimes called “**unmixing**”.

# Geometric Intuition

- Since latent variables are assumed to be independent, we are trying to find **linear transformation of data that recovers independent causes**.
- Avoid degeneracies in Gaussian latent variable models: **Assume non-Gaussian prior distribution** for latent variables (sources).
- Recall that in PPCA (and FA) the model cannot distinguish between two different choices for the latent variables: These differ simply by a rotation in latent space!
- Often we use **heavy-tailed** source priors, e.g.:

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

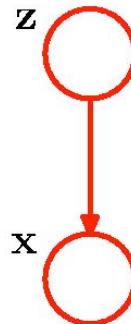
- Geometric intuition: **finding spikes in histograms**.



# ICA Model

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

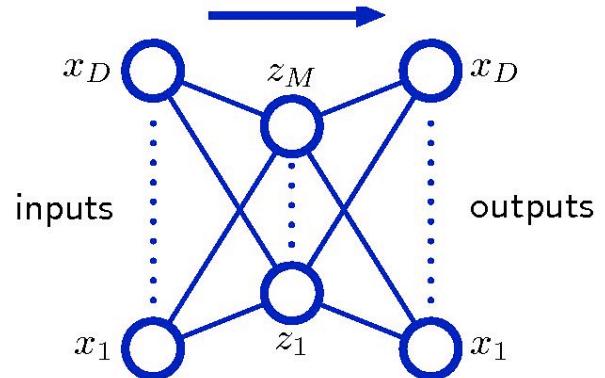
$$p(\mathbf{z}) = \prod_k p(z_k)$$
$$\mathbf{x} = \mathbf{V}\mathbf{z}$$



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

# Autoencoders

- Neural networks can also be used for nonlinear dimensionality reduction.
- This is achieved by having the same number of outputs as inputs. These models are called **autoencoders**.
- Consider a multilayer perceptron that has D inputs, D outputs, and M hidden units, with  $M < D$ .
- It is useful if we can squeeze the information **through some kind of bottleneck**.
  - If we use a **linear network** this is very similar to Principal Components Analysis.



# Autoencoders and PCA

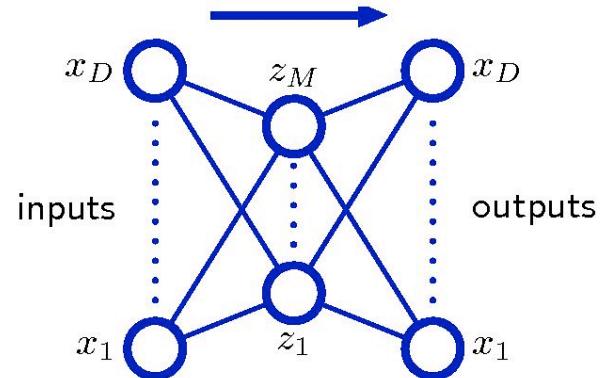
- Given an input  $\mathbf{x}$ , its corresponding reconstruction is given by:

$$y_k(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^M w_{kj}^{(2)} \sigma \left( \sum_{i=1}^D w_{ji}^{(1)} x_i \right), \quad k = 1, \dots, D.$$

- We can determine the network parameters  $\mathbf{w}$  by minimizing the reconstruction error:

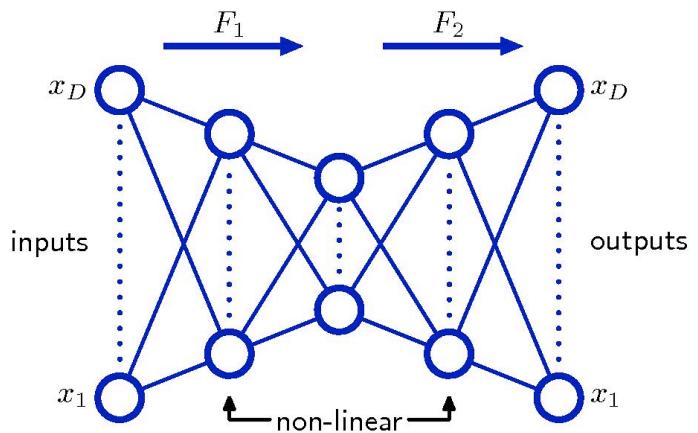
$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|y(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n\|^2.$$

- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.
- The  $M$  hidden units will span the same space as the first  $m$  principal components. The weight vectors may not be orthogonal.



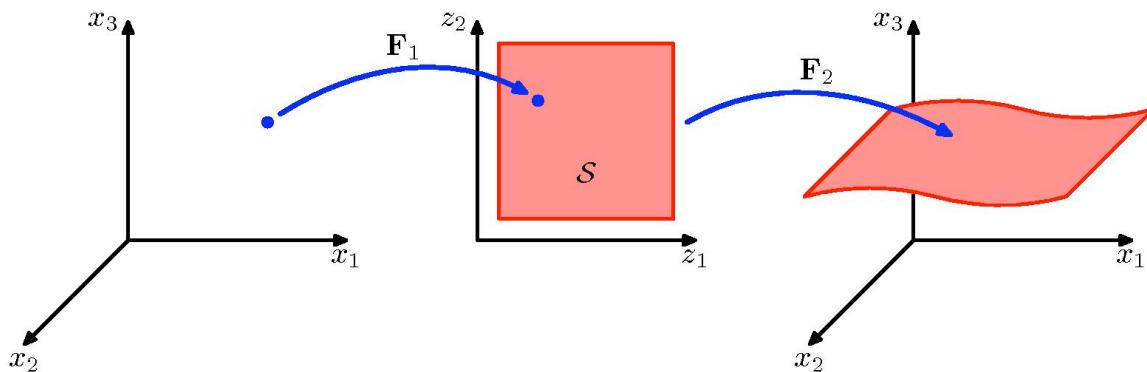
# Deep Autoencoders

- We can put **extra nonlinear hidden layers** between the input and the bottleneck and between the bottleneck and the output.
- This gives **nonlinear generalization** of PCA.
- It should be very good for non-linear dimensionality reduction.
- The network can be trained by the **minimization of the reconstruction error function**.
- Much harder to train.



# Geometrical Interpretation

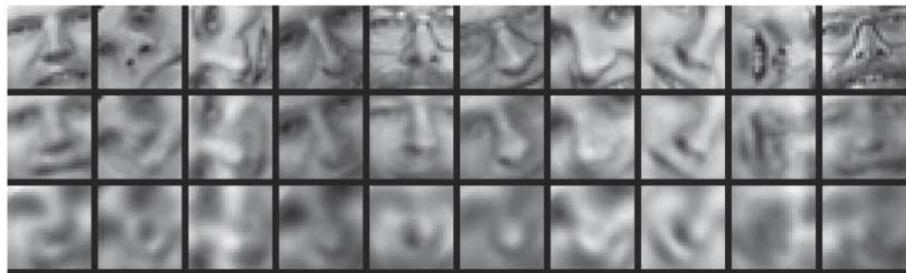
- Geometrical interpretation of the mappings performed by the network with 2 hidden layers for the case of  $D=3$  and  $M=2$  units in the middle layer.



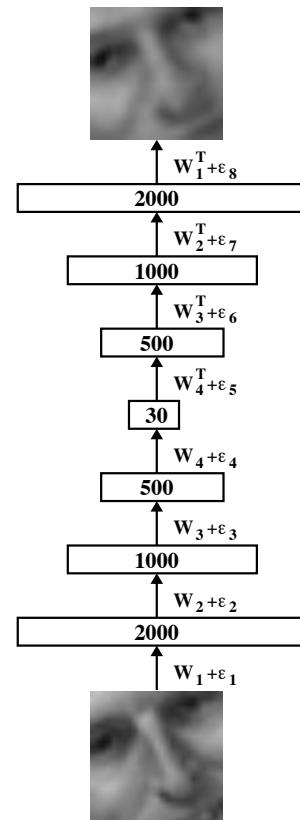
- The mapping  $F_1$  defines a nonlinear projection of points in the original  $D$ -space into the  $M$ -dimensional subspace.
- The mapping  $F_2$  maps from an  $M$ -dimensional space into  $D$ -dimensional space .

# Deep Autoencoders

- We can consider very deep autoencoders.
- There is an efficient way to learn these deep autoencoders



- By row: Real data, Deep autoencoder with a bottleneck of 30 linear units, and 30-d PCA.



# Deep Autoencoders

- We can consider very deep autoencoders.
- Similar model for MNIST handwritten digits:



Real data

30-d deep autoencoder

30-d logistic PCA

30-d PCA

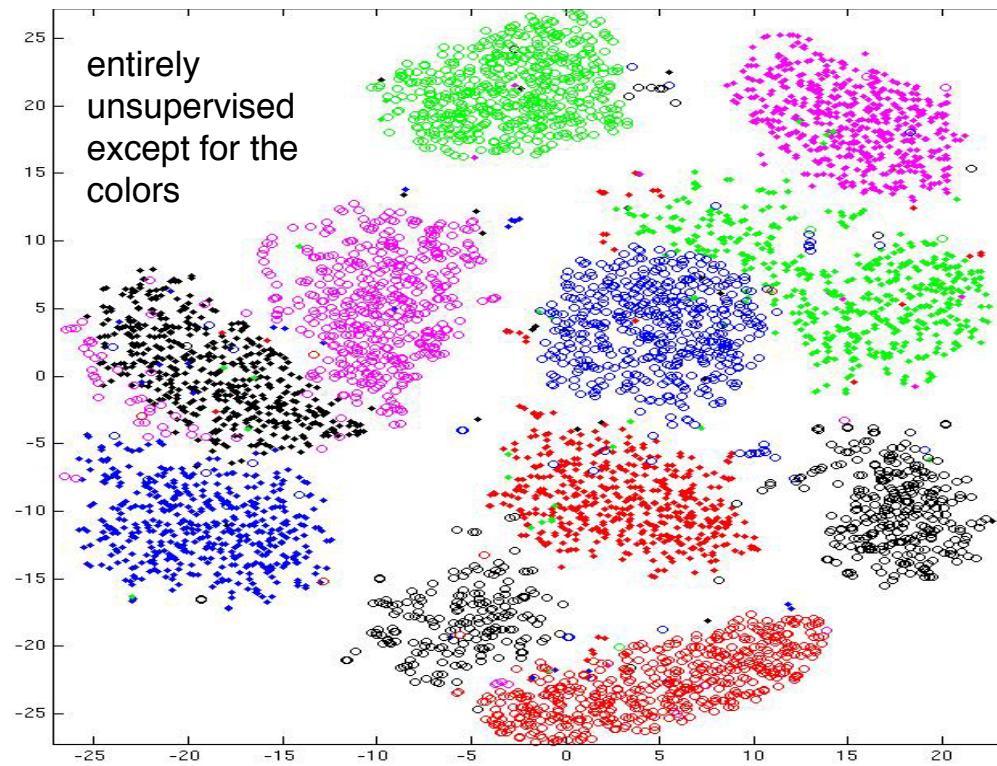
- Deep auto produces much better reconstructions.

# Class Structure of the Data

- Do the 30-D codes found by the deep autoencoder preserve the class structure of the data?
- Take the 30-D activity patterns in the code layer and display them in 2-D using a new form of **non-linear multi-dimensional scaling** (UNI-SNE).
- Will the learning find the natural classes?

# Class Structure of the Data

- Do the 30-D codes found by the deep autoencoder preserve the class structure of the data?

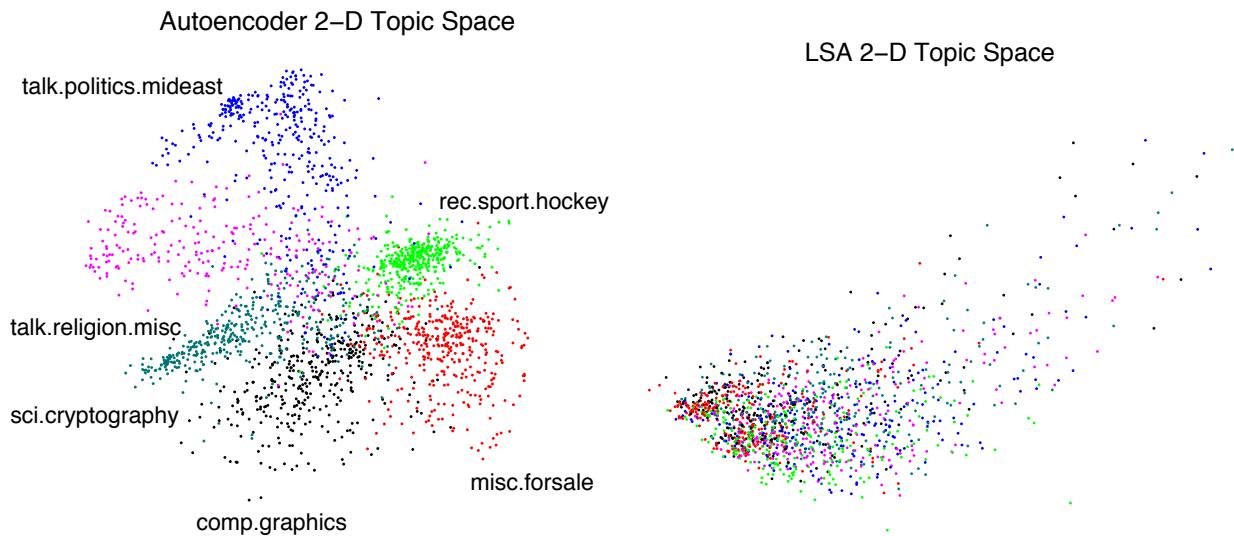


# Learning 2-D topic Space

- Latent Semantic Analysis (LSA) uses SVD to get a **low-rank approximation** of the log of term-frequency matrix:

$$\log(1 + M(\text{doc}, w)) \sim USV$$

$$U = |\text{doc}| \times d, S = d \times d, V = d \times |w|.$$



# Reuters dataset

- Autoencoder: 2000-500-250-125-2

