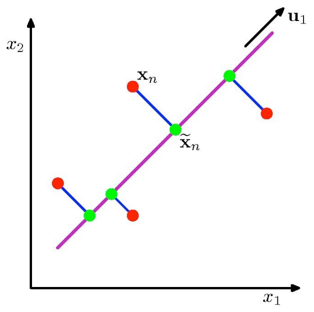
STA414/2104

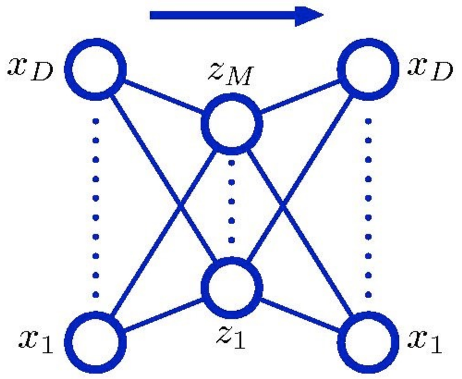
Statistical Methods for Machine Learning II

Murat A. Erdogdu

Department of Computer Science
Department of Statistical Sciences

Lecture 9





Announcements

- HW 4 is released. Due on Apr 5, 2pm.
- TA OHs will be on the next week, time TBA.

Last time

- Unsupervised learning
- Mixture models
- K-means
- EM Algorithm

Today

- PCA
- PPCA
- Autoencoders
- Recommender Systems

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.

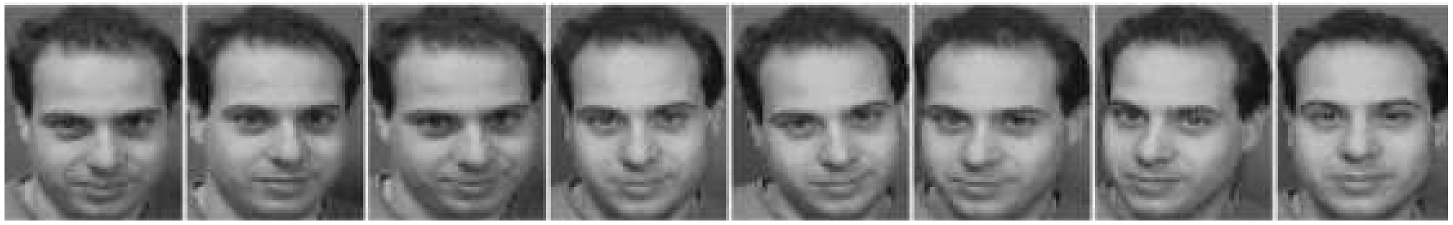
- Motivation: for many datasets, data points lie close to a manifold of much lower dimensionality compared to that of the original data space.

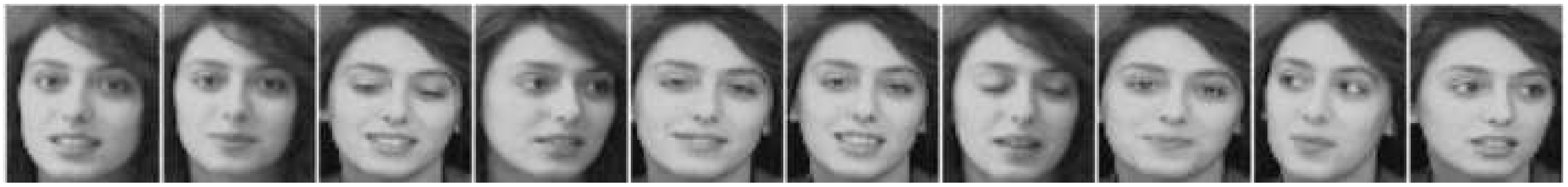
- Training continuous latent variable models often called dimensionality reduction, since there are typically many fewer latent dimensions.

- Examples: Principal Components Analysis, Factor Analysis, Independent Components Analysis

Intrinsic Latent Dimensions

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

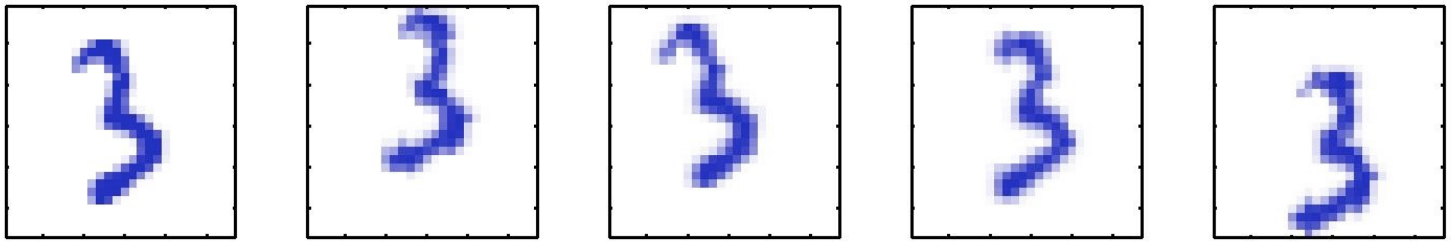




- How can we find these latent dimensions from this high-dimensional data.

Intrinsic Latent Dimensions

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



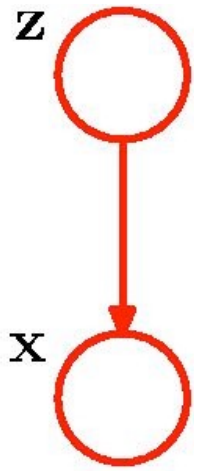
- Each image undergoes a random displacement and rotation within some larger image field.

- The resulting images have 100 x 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.



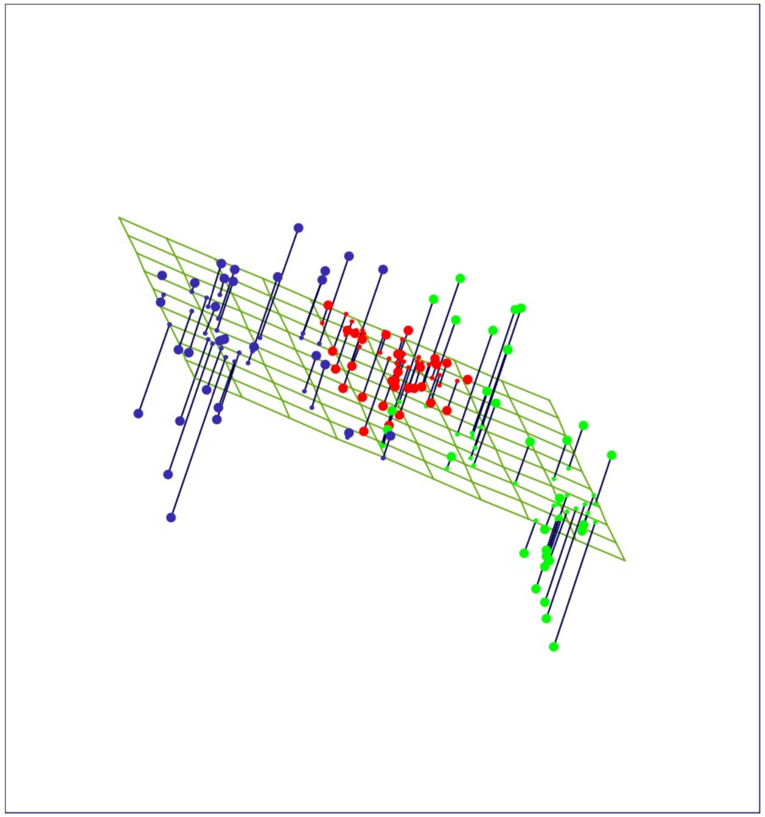
• This leads to probabilistic formulation of the Principal Component Analysis.

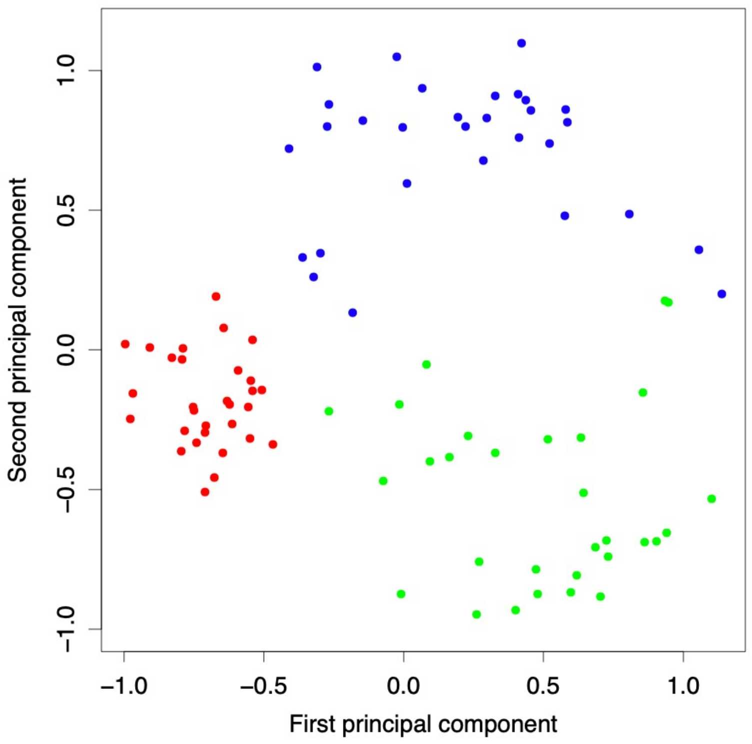
• We will first look at standard PCA, and then consider its probabilistic formation.

• Advantages of probabilistic formulation: use of EM for parameter estimation, mixture of PCAs, Bayesian PCA.

Latent dimensions

In practice, even though data is very high dimensional, important features can be accurately captured in a low dimensional subspace.





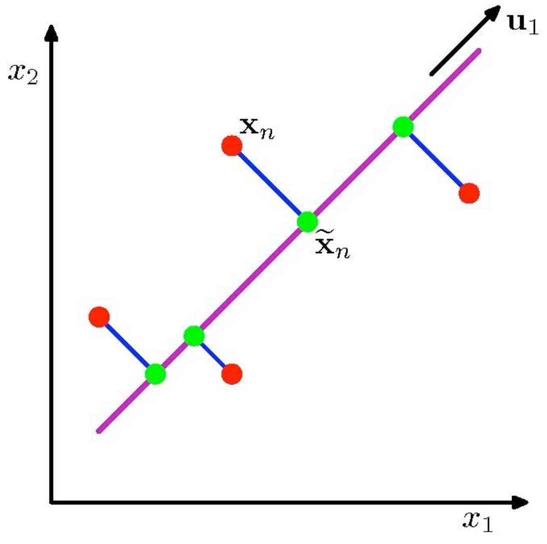
Principal Component Analysis

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

- The goal is find M principal 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.

$$z_{n1} = \mathbf{x}_n^T \mathbf{u}_1.$$

Projection of $\mathbf{x}_n$ on $\mathbf{u}_1$



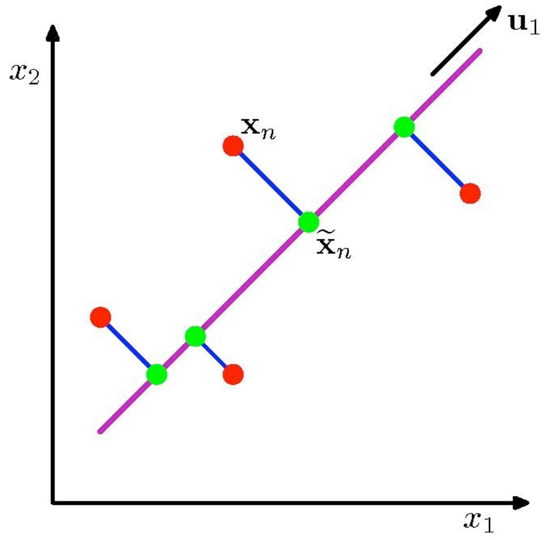
Principal Component Analysis

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

- The goal is find M principal 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.

$$z_{n1} = \mathbf{x}_n^T \mathbf{u}_1.$$

Projection of $\mathbf{x}_n$ on $\mathbf{u}_1$



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

Maximum Variance Formulation

- Consider a dataset $\{x_1,\ldots,x_N\}$, where x_n is in R^D. Our goal is to project data onto a space having dimensionality M < D.

- Consider the projection into M=1 dimensional space: $z_{n1} = \mathbf{x}_n^T \mathbf{u}_1$.

- Define the direction of this space using a D-dimensional unit vector $\mathbf{u}_1$, so that $\mathbf{u}_1^T \mathbf{u}_1 = 1$.

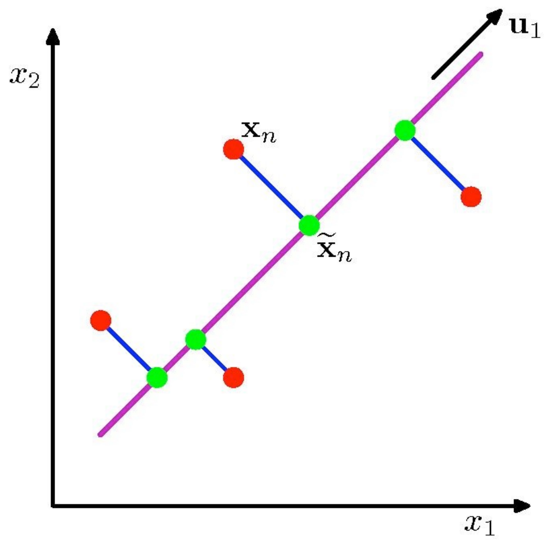
- Objective: maximize the variance of the projected data with respect to $\mathbf{u}_1$.

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

where sample mean and data covariance is given by:

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

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



Maximum Variance Formulation

- Maximize the variance of the projected data:

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

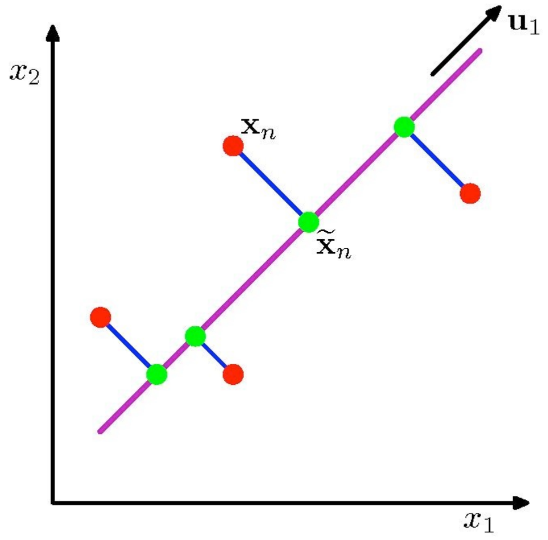
- Must constrain $\|u_1\| = 1$. Using Langrage multiplier, maximize:

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

- Setting the derivative with respect to u_1 to zero:

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

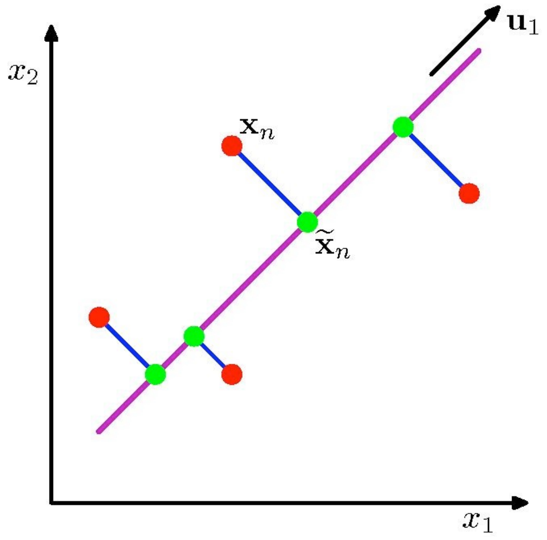
- Hence u_1 must be an eigenvector of S.



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 $\|u_1\| = 1$. Using Langrage 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 u_1 to zero:

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

- Hence u_1 must be an eigenvector of S.
- The maximum variance of the projected data is given by:

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

- Optimal u_1 is the first principal component (eigenvector with maximal eigenvalue).

Minimum Error Formulation

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

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}.$$ (1 if i=j, 0 otw)

- 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 the basis vectors $\mathbf{u}_i$.

Minimum Error Formulation

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

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}. \quad \text{(1 if i=j, 0 otw)}$$

- 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 the basis vectors $\mathbf{u}_i$.

- 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:

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

Specific to a data point

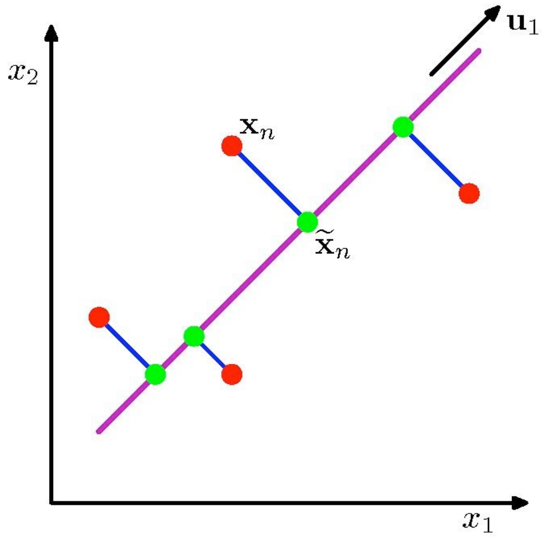
Shared across all data points

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 u_i, z_{ni}, and b_i.

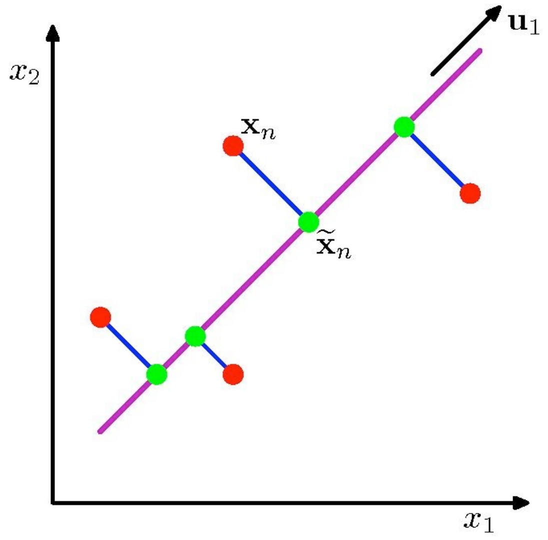
$$J = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \tilde{\mathbf{x}}_n||^2.$$

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 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 u$_i$: constraint minimization problem:

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

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

- The general solution is obtained by choosing 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 S with lowest eigenvalues → same result.

- If we use matrix notation for data: $\mathbf{X} = [\mathbf{x}_1^T; ..; \mathbf{x}_N^T]$ $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1^T; ..; \tilde{\mathbf{x}}_N^T]$

$$J = \frac{1}{N}||\mathbf{X} - \tilde{\mathbf{X}}||_F^2 \longleftarrow$$

Frobenius norm

- PCA is finding the best rank-M approximation!

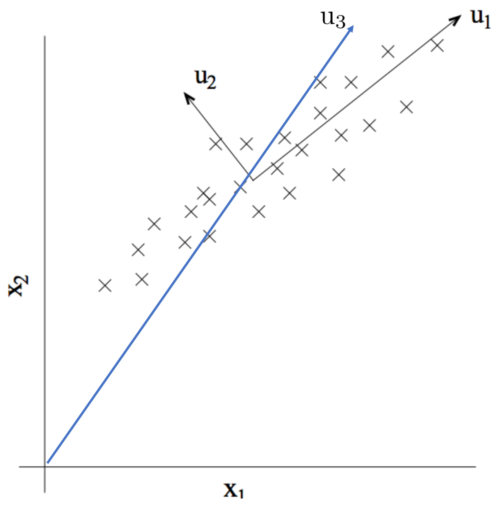
Always center data

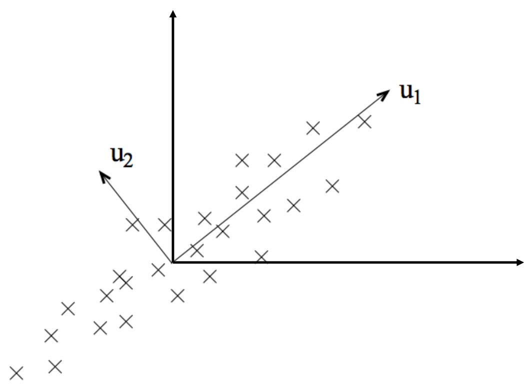
- Minimize distortion gives:

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

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

- We don't care about u_3. We want u_1.





PCA summary

- Observe D-dimensional N data points: $\mathbf{X} \in \mathbb{R}^{N \times D}$

- Compute the centered empirical covariance matrix:

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

- The optimal PCA subspace spanned by the top eigenvectors of the matrix S.

$$\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$$

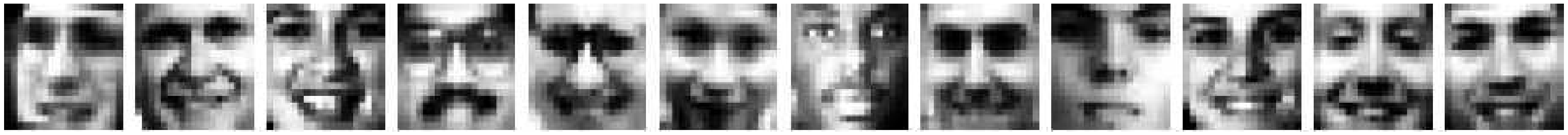
- More precisely, for the eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_D$

- **We compute,**

$$\begin{aligned} z_{nj} &= \mathbf{x}_n^T \mathbf{u}_j \qquad j = 1, ..., M \\ b_j &= \bar{\mathbf{x}}^T \mathbf{u}_j \qquad j = M+1, ..., D \end{aligned}$$

Applications of PCA

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



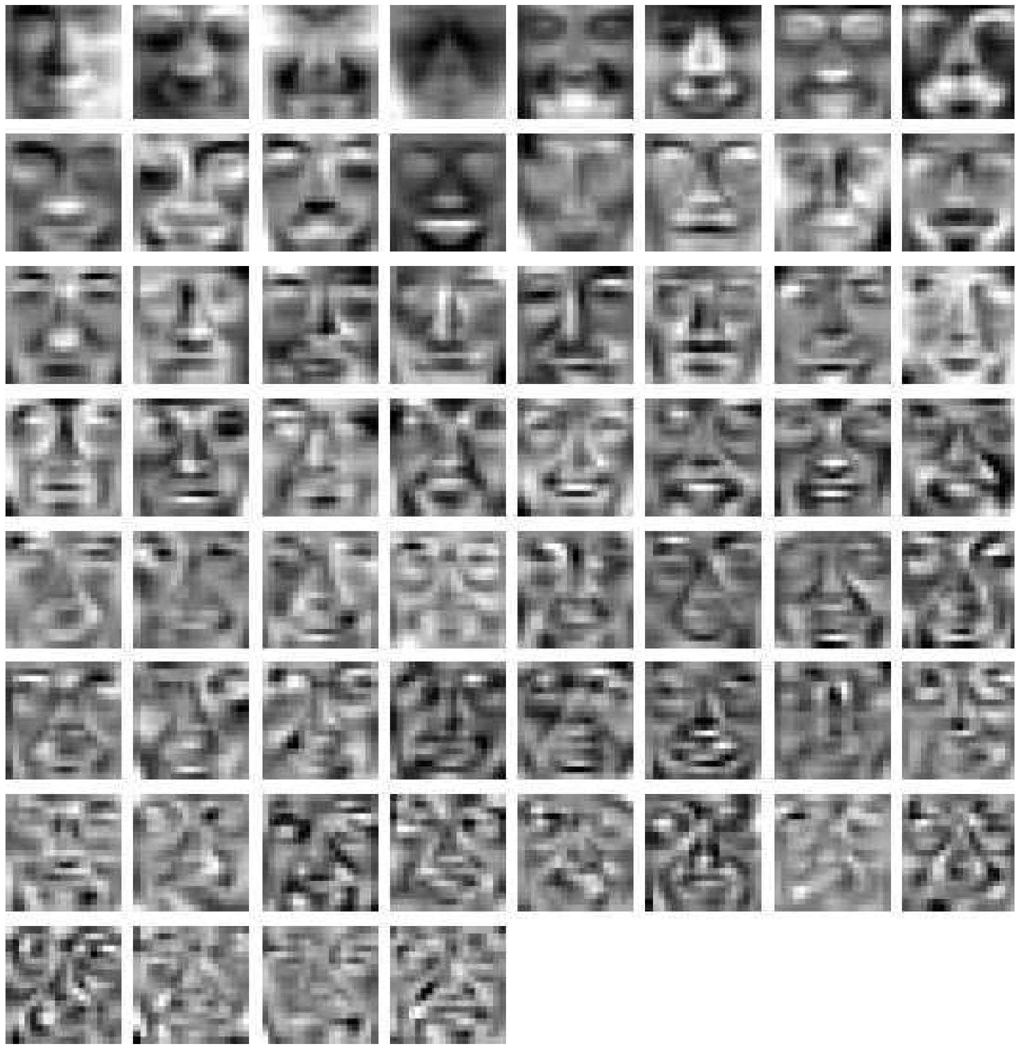
- Data compression: We can get good reconstructions with only 3 components.

- 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.

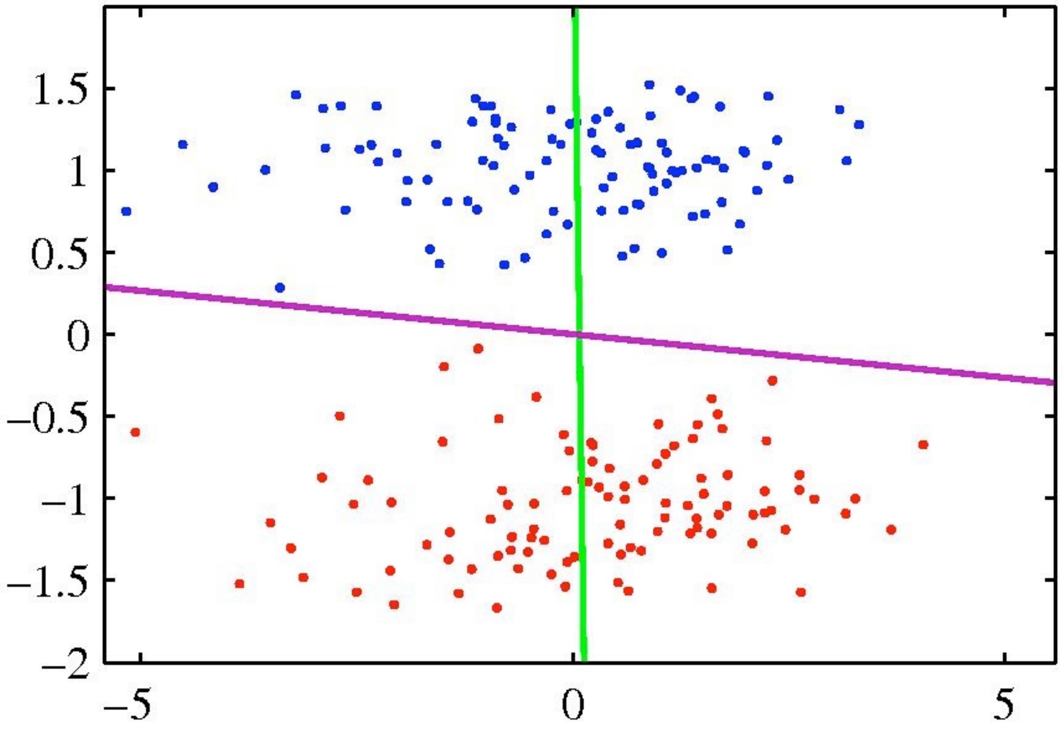
Learned Basis

- 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 x D data covariance matrix S, which scales as $O(D^3)$.
- Direct application of PCA will often be computationally infeasible.
- Solution: Let X be the N x 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 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 x N matrix
- It has the same N-1 eigenvalues as the original data covariance matrix 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 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 $X^T v_i$ is an eigenvector of S with eigenvalue λ_i .

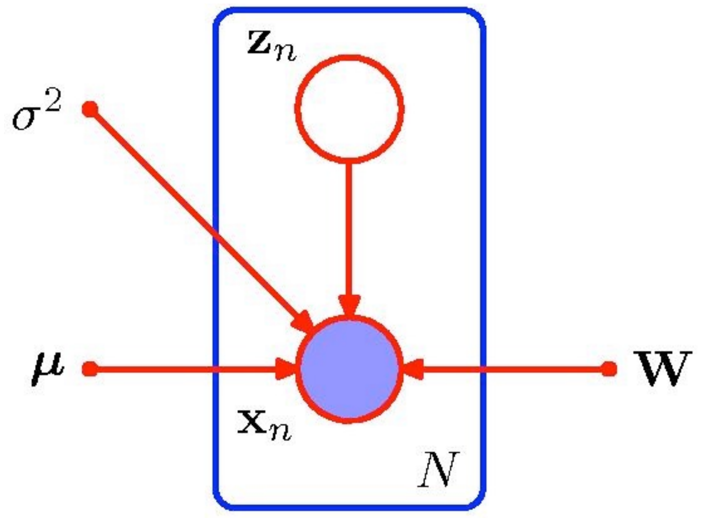
Probabilistic PCA

- Probabilistic, generative view of data.
- Key properties of the 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 z has a Gaussian distribution (This is z_{nj} in previous formulation)
  - linear relationship between M-dim latent z and D-dim observed x variables.
  - isotropic Gaussian noise in observed dimensions

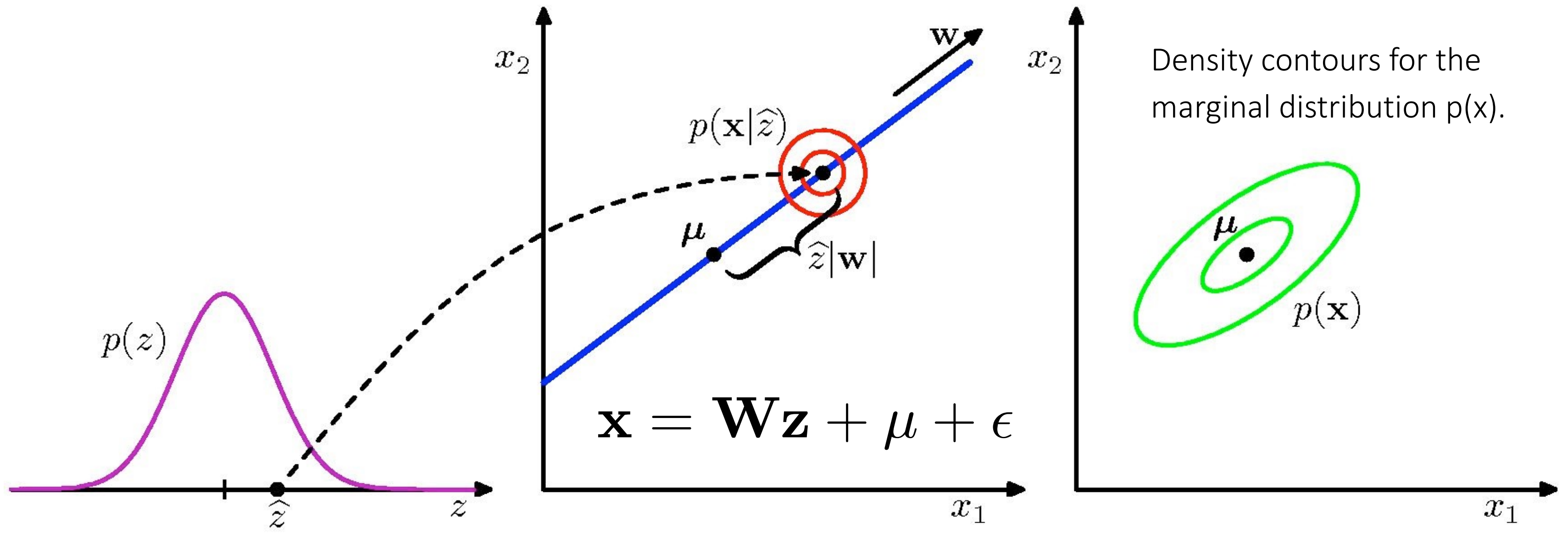


$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I}) \\ p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z}+\mu,\sigma^2\mathbf{I}) \end{aligned}$$

- Hence the mean of x is a linear function of z governed by the D x M matrix W and the D-dim vector μ .
- We will see that the columns of W span the principal subspace of the data space (Columns of W are the principal components, σ^2 is sensor noise).

Generative View of PPCA

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


$p(z)$
$\widehat{z}$
z
x_2
x_1
$\mathbf{w}$
$p(\mathbf{x}|\widehat{z})$
$\boldsymbol{\mu}$
$\widehat{z}|\mathbf{w}|$
$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon$
x_2
Density contours for the marginal distribution p(x).
$\boldsymbol{\mu}$
$p(\mathbf{x})$
x_1

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

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

- Draw a value for x from from an isotropic Gaussian distribution:

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

Marginal Data Density

- The joint p(z,x), the marginal data distribution p(x) and the posterior distribution p(z|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 this result directly by computing mean and covariance given that it is Gaussian:

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

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

Redundancy in Parameterization

- The marginal distribution is governed by parameters W, μ , σ^2 :

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

- Redundancy in parameterization: rotation of the latent space coordinates.

- Let R be an orthogonal matrix, then define a new matrix:

$$\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}, \qquad \mathbf{R}\mathbf{R}^T = \mathbf{I}.$$

- Then

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

- 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 x is D-dim and z is M-dim is given:

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

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 x D matrix **C**, which can be expensive.
- Reduce $O(D^3)$ to $O(M^3)$ by applying matrix inversion lemma (Sherman–Morrison formula – check wikipedia):

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

Posterior Distribution for PPCA

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

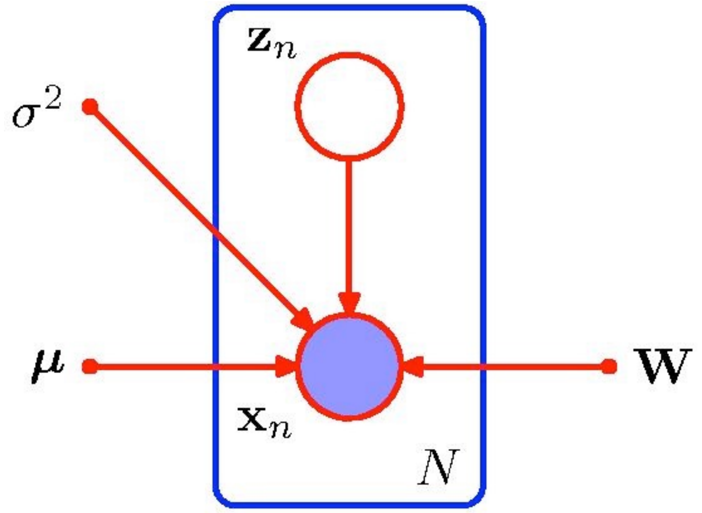
$$p(\mathbf{z}|\mathbf{x}) \;=\; \mathcal{N}(\mathbf{z}|\mathbf{m}, \mathbf{V})$$

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

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

(Exercise)

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



- Mean of inferred z is a linear map of centered x.

- Posterior variance does not depend on the input x at all.

- Remember:

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

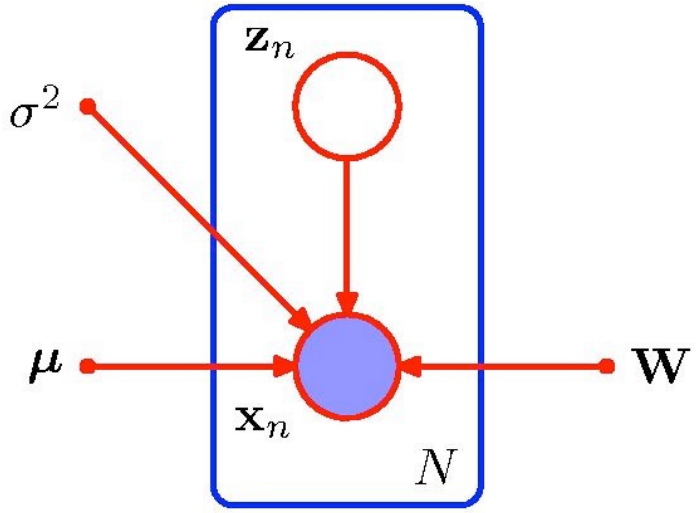
$$\mathbf{C}^{-1} = \sigma^{-1}\mathbf{I} - \sigma^{-2}\mathbf{W}\overbrace{(\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})}^{\text{M matrix}}{}^{-1}\mathbf{W}^T$$

Constrained Covariance

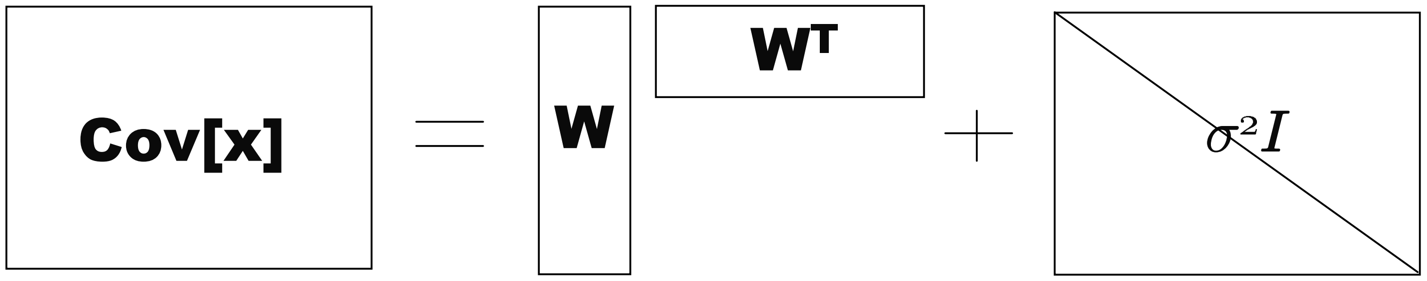
- Marginal density for PPCA has the following form:

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

where θ = (W, μ, σ^2).



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



- 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):

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

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

- We then have:

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

- Maximizing with respect to W and σ^2 can be solved directly.

Maximum Likelihood

- Objective:

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

- C is model covariance; 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.

Maximum Likelihood

- Objective:

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

- Maximizing with respect to W (derivation skipped in class):

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

where

- U_M is a D x M matrix whose columns are given by the M principal eigenvectors of the data covariance matrix S.
- L_M is the M x M diagonal matrix containing M largest eigenvalues of S.
- R is an arbitrary M x M orthogonal matrix.

- If the eigenvectors have been arranged in the order of decreasing values of the corresponding eigenvalues, then the columns of W define the principal subspace of standard PCA.

Maximum Likelihood

- Objective:

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

- Maximizing with respect to : σ^2

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

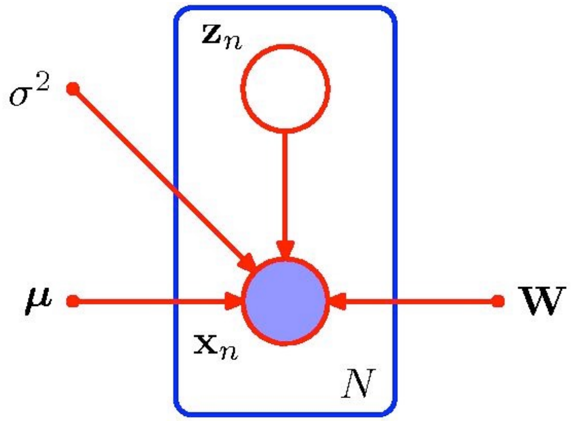
which is the average variance associated with the discarded dimensions.

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 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(z | X).
- M-step: maximize with respect to parameters W and σ^2.



- Appealing property: EM avoids direct O(ND^2) construction of covariance matrix!
- Instead EM involves sums over data cases: O(NDM).

Zero Noise Limit

- We can derive standard PCA as a limit of probabilistic PCA as the noise term goes to zero: $\sigma^2 \to 0.$

- MLE parameters are the same.

- Inferring the distribution over latent variables is easier: The posterior mean reduces to:

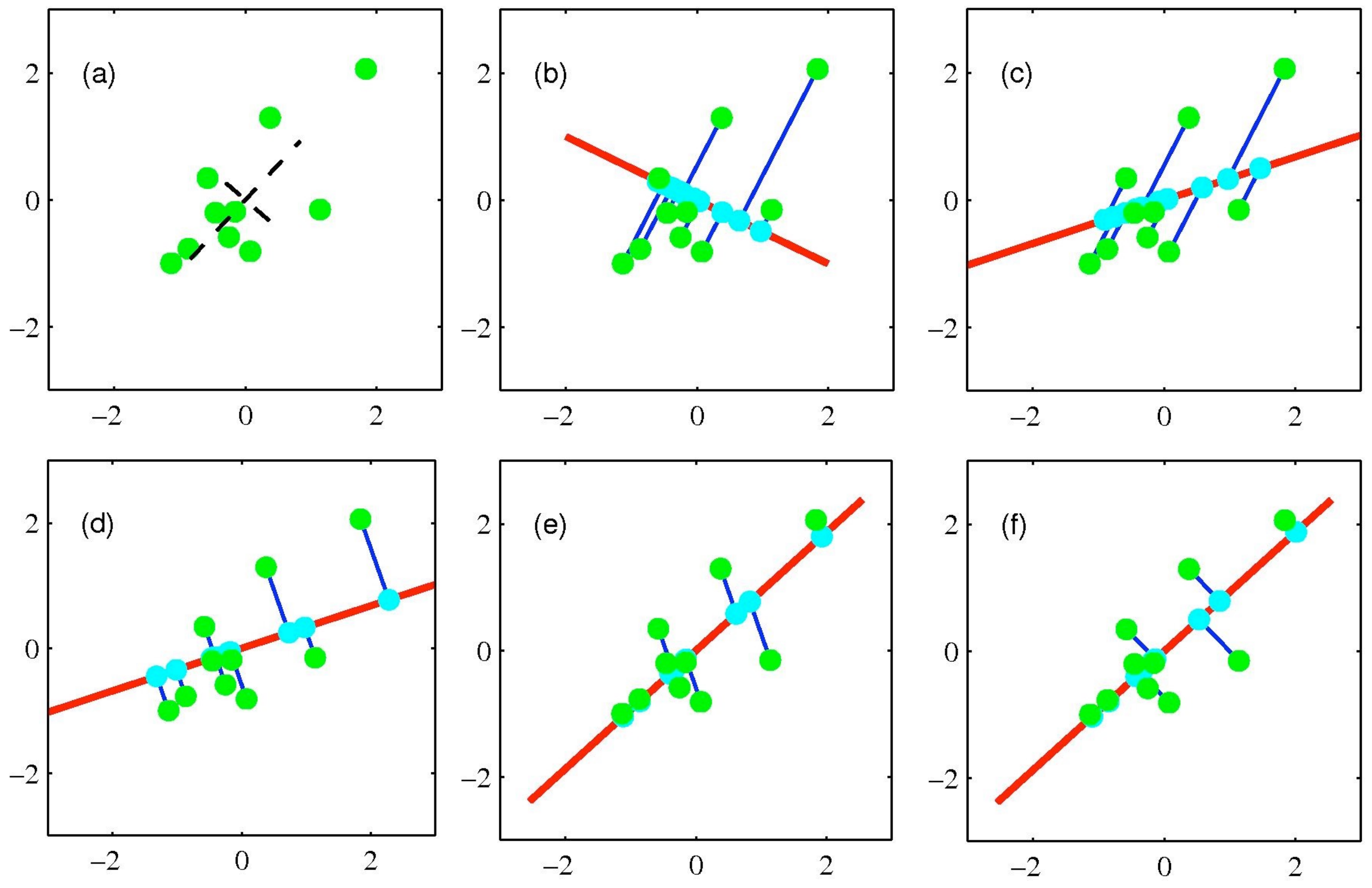
$$\lim_{\sigma^2 \to 0} (\mathbf{W}^T\mathbf{W} + \sigma\mathbf{I})^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}) = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}),$$

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

- Posterior covariance goes to zero!

EM for PPCA

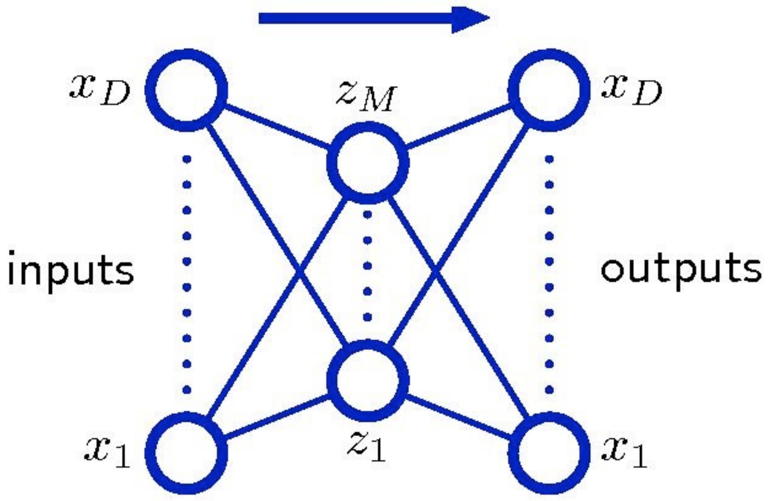
- EM algorithm for PCA.



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 (linear activation) this is very similar to Principal Components Analysis.



Autoencoders and PCA

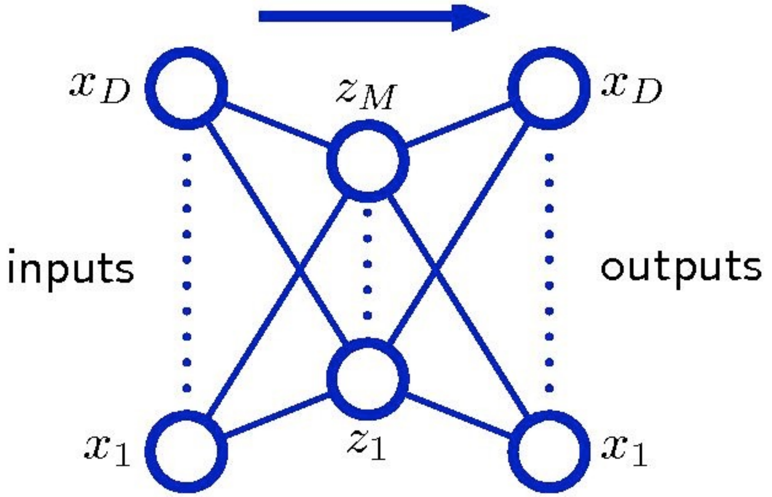
- Given an input x, its corresponding reconstruction is given by:

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

- We can determine the network parameters w by minimizing the reconstruction error:

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

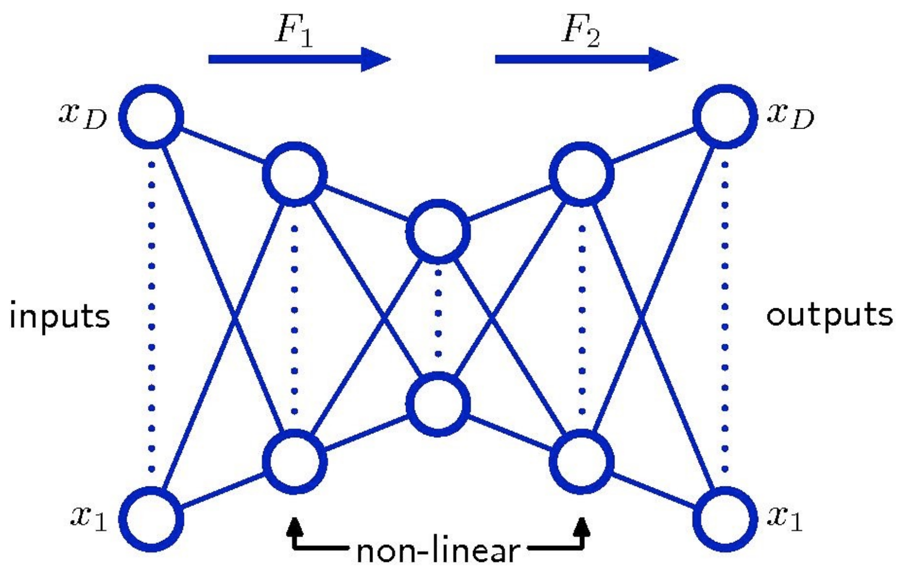
- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.



- The M hidden units will span the same space as the first m principal components. The weight vectors may not be orthogonal.

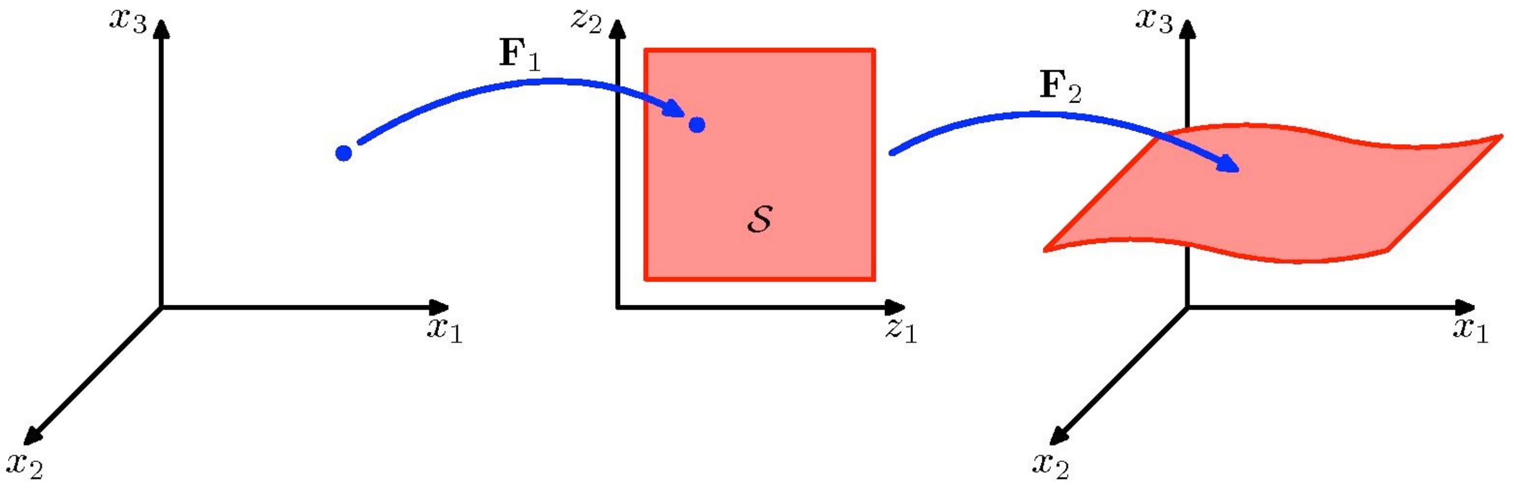
Deep Autoencoders

- 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.

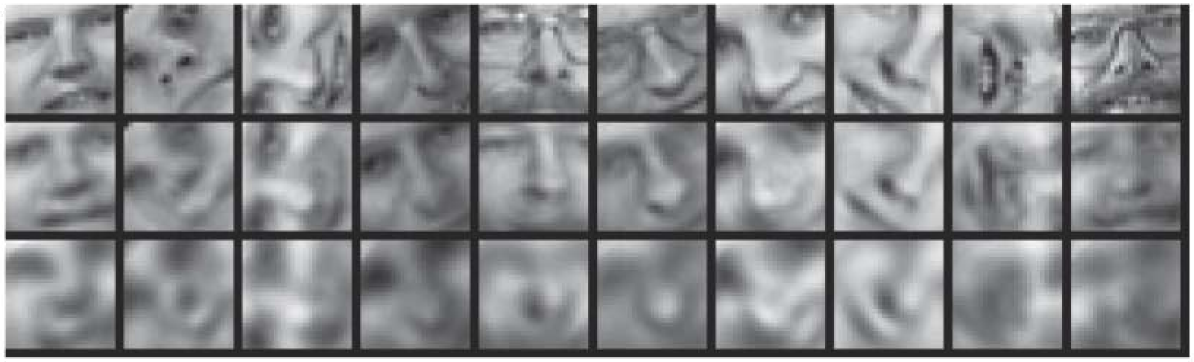


- The mapping F_1 defines a nonlinear projection of points in the original D-space into the M-dimensional subspace.

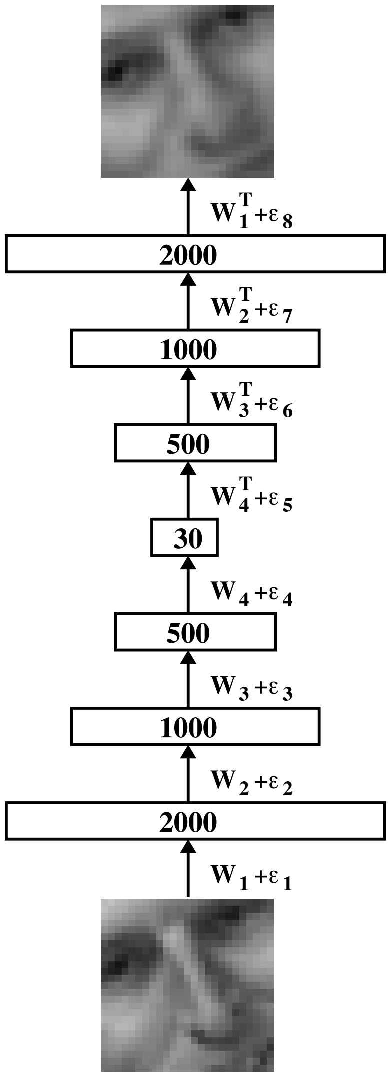
- The mapping F_2 maps from an M-dimensional space into D-dimensional space .

Deep Autoencoders

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

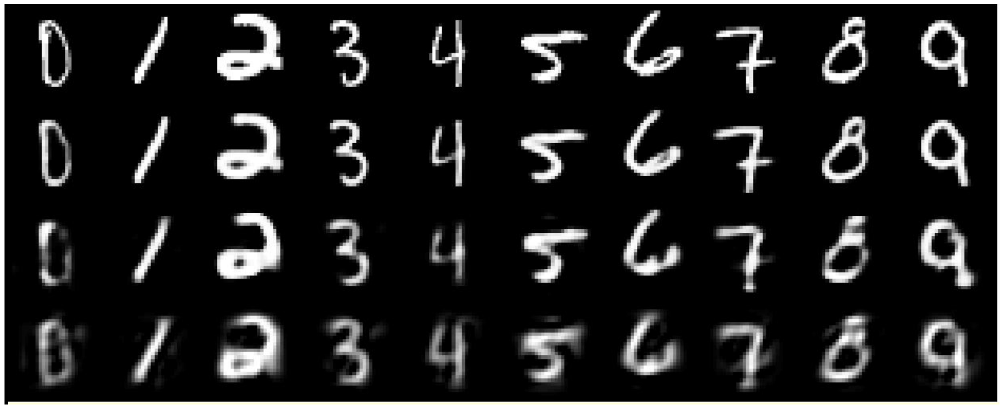


- By row: Real data, Deep autoencoder with a bottleneck of 30 linear units, and 30-d PCA.



Deep Autoencoders

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



- Deep auto produces much better reconstructions.

Class Structure of the Data

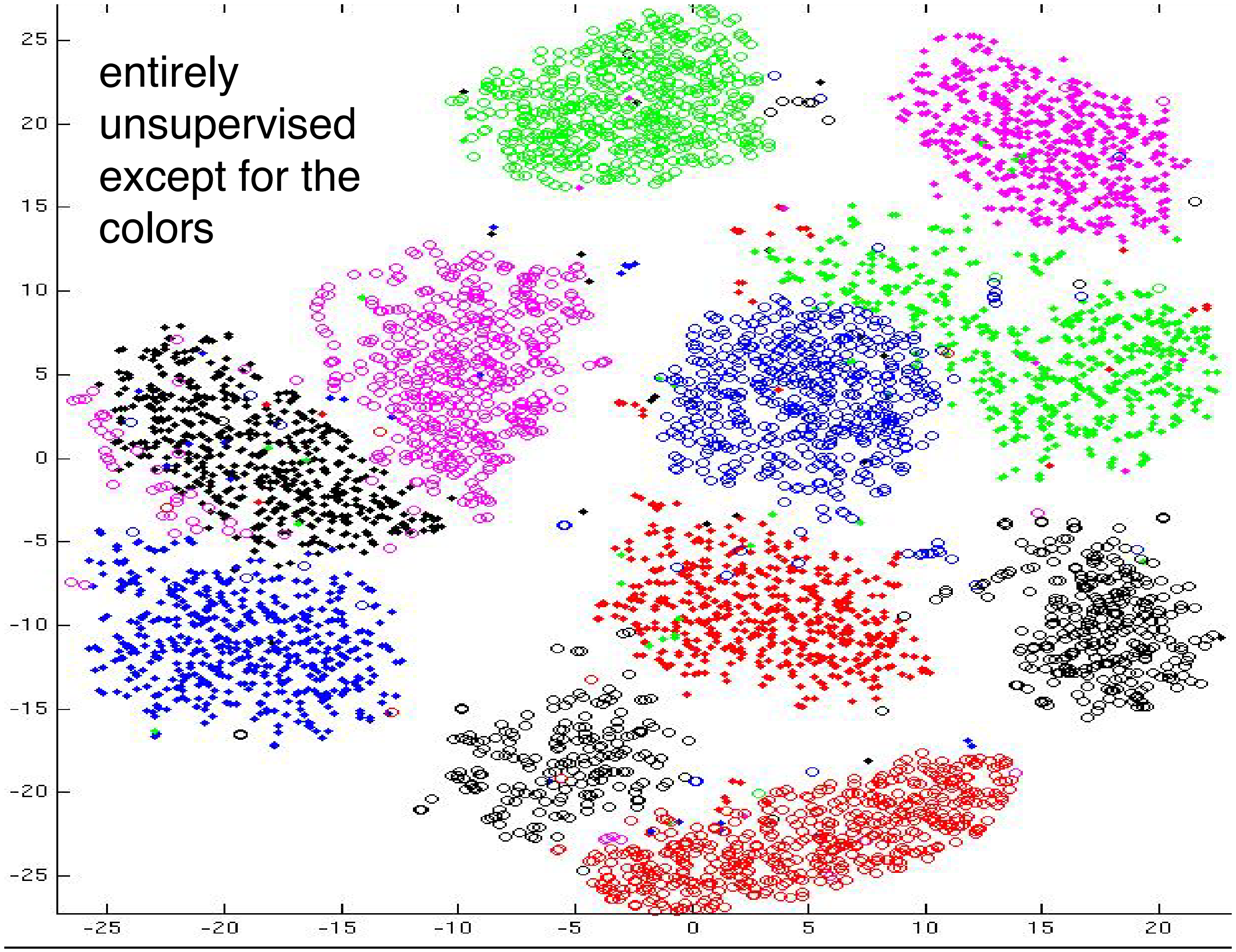
- Do the 30-D codes found by the deep autoencoder preserve the class structure of the data?

- 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?

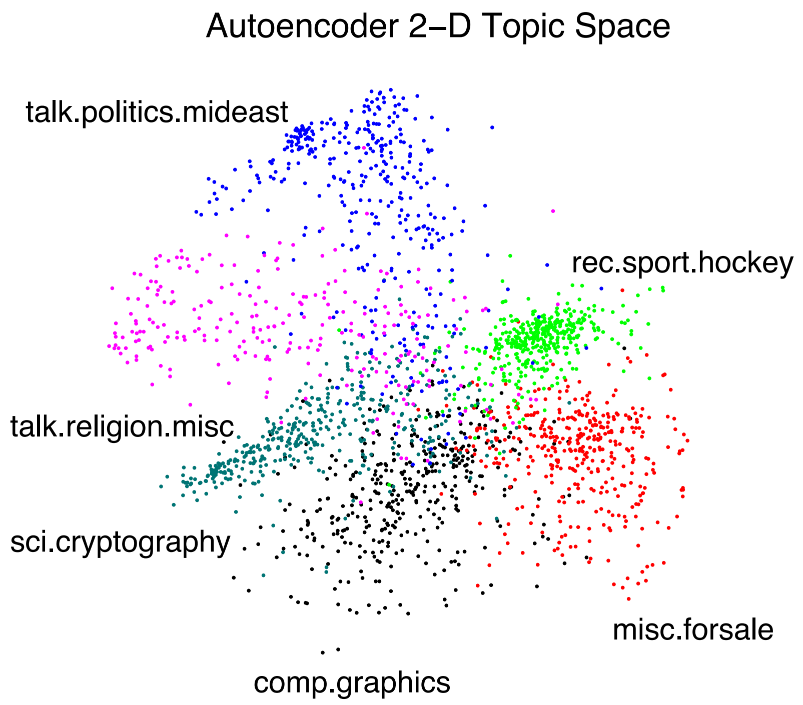


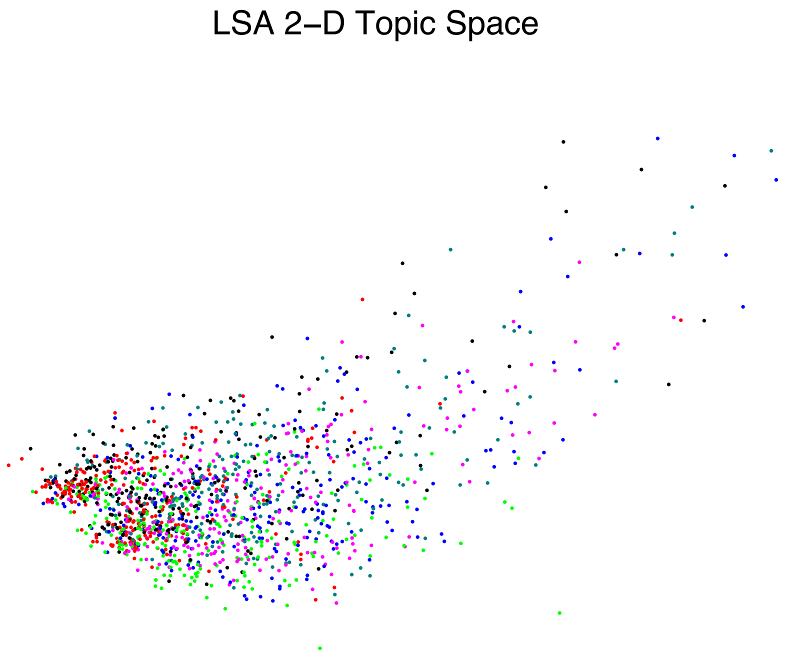
Learning 2-D topic Space

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

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

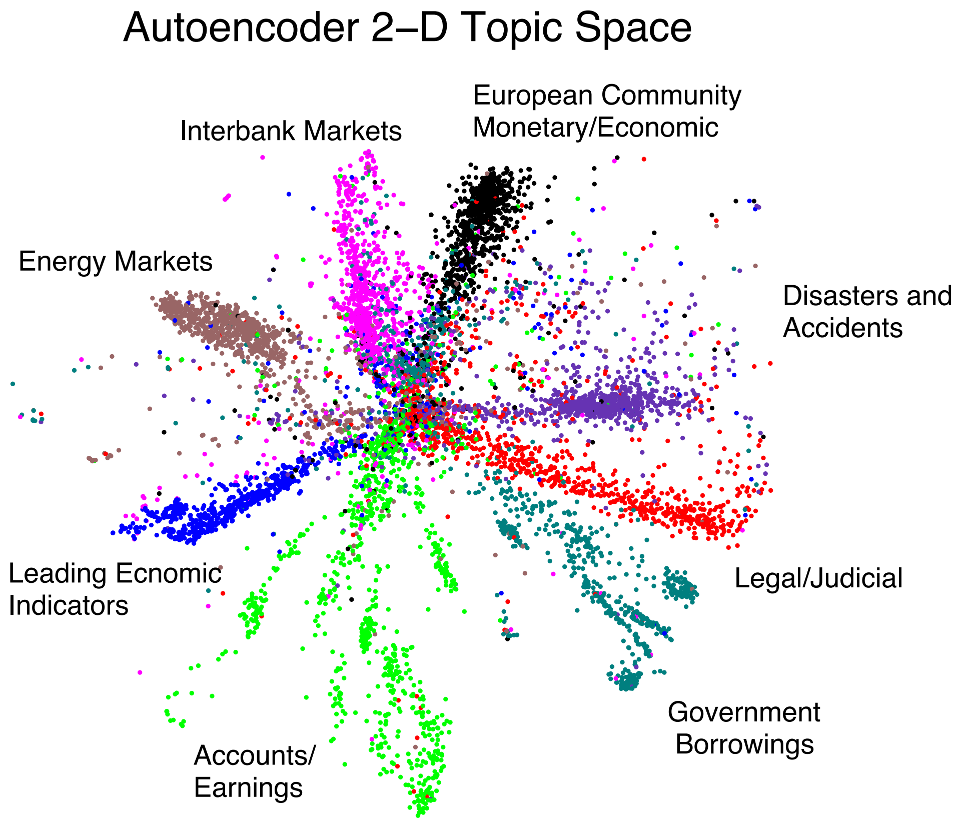
$$U = |doc| \times d, S = d \times d, V = d \times |w|.$$

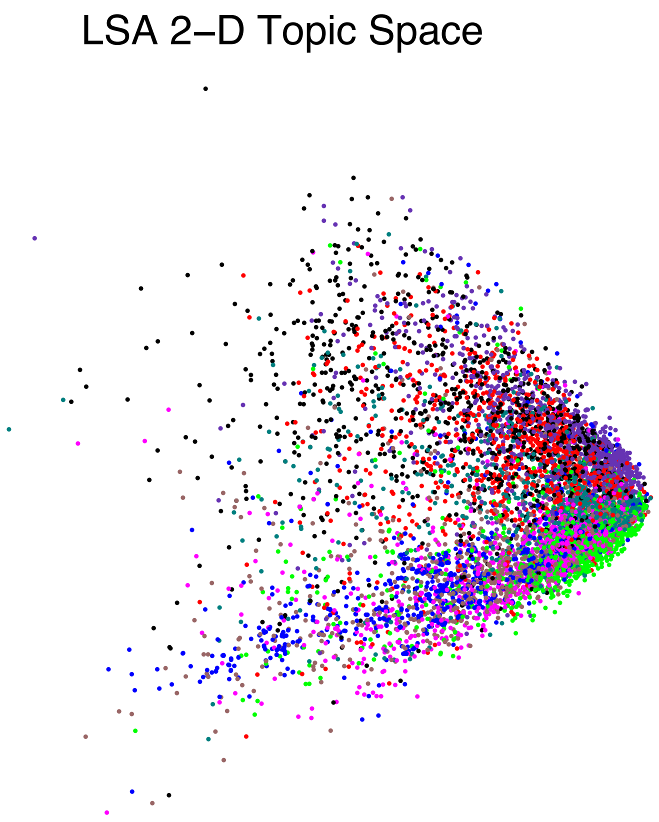




Reuters dataset

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





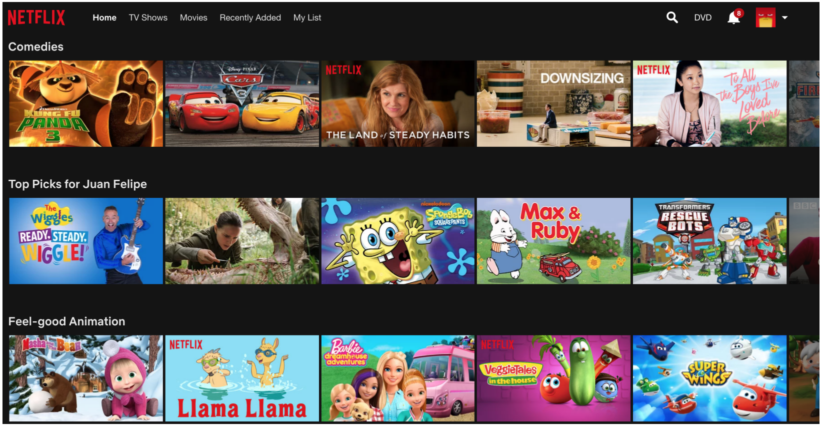
Recommender systems: Why?

- YouTube CA 400 hours of video are uploaded to YouTube every minute
- amazon.ca 353 million products and 310 million users
- Spotify 83 million paying subscribers and streams about 35 million songs

Who cares about all these videos, products and songs? People may care only about a few → Personalization: Connect users with content they may use/enjoy.

Recommender systems suggest items of interest and enjoyment to people based on their preferences

Some recommender systems in action



The Netflix problem

Movie recommendation: Users watch movies and rate them out of 5★.

User	Movie	Rating
🥷	Thor	★ ☆ ☆ ☆ ☆
🥷	Chained	★ ★ ☆ ☆ ☆
🥷	Frozen	★ ★ ★ ☆ ☆
🐱	Chained	★ ★ ★ ★ ☆
🐱	Bambi	★ ★ ★ ★ ★
😇	Titanic	★ ★ ★ ☆ ☆
😇	Goodfellas	★ ★ ★ ★ ★
😇	Dumbo	★ ★ ★ ★ ★
🙂	Twilight	★ ★ ☆ ☆ ☆
😋	Frozen	★ ★ ★ ★ ★
😐	Tangled	★ ☆ ☆ ☆ ☆

Because users only rate a few items, one would like to infer their preference for unrated items

PCA as a matrix factorization

Ultimately, PCA with M principal components finds the *optimal* rank-M approximation of $\mathrm{X} \in \mathbb{R}^{N \times D}$, in terms of error $||\mathrm{X}^\top - \mathrm{U}\mathrm{Z}^\top||_F^2$.

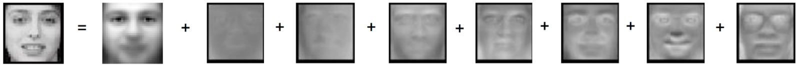
$$\min ||\mathrm{X}^\top - \mathrm{U}\mathrm{Z}^\top||_F^2 \;\; \text{over} \;\; \mathrm{Z} \in \mathbb{R}^{N \times M}, \mathrm{U} \in \mathbb{R}^{D \times M}.$$

Note that the case $M = D$ corresponds to the original matrix X.

- Can we do something similar for recommender systems?

PCA as matrix factorization of X

Recall



$$\bar{\mathbf{x}} = \mu + z_1\mathbf{u_1} + z_2\mathbf{u_2} + z_3\mathbf{u_3} + \ldots$$

where the vectors $\mathbf{u_i}$ are the principal components of the data matrix X (the latent factors).
We can do the same for our ratings matrix R. Rating of movie

$\bar{🥷}$ = **average user**$+z_1$**comedy user**$+z_2$**drama user**$+z_3$**action user**$+\ldots$

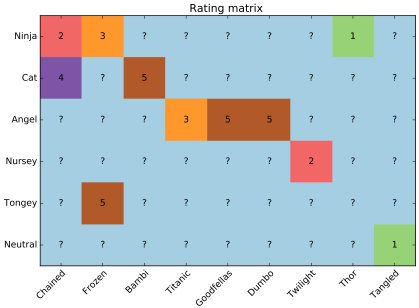
These latent factors are idealized, the real latent factors do not necessarily reveal these semantic concepts so clearly.

Matrix Completion

- We just saw that PCA gives the optimal low-rank matrix factorization.
- Two ways to generalize this:
  - 1) Consider when $\mathbf{X}$ is only partially observed.
    - A sparse $\mathbf{1000 \times 1000}$ matrix with 50,000 observations (only 5% observed).
    - A rank 5 approximation requires only 10,000 parameters, so it's reasonable to fit this.
    - Unfortunately, no closed form solution.
  - 2) Impose structure on the factors. We can get lots of interesting models this way.

Matrix completion problem

Matrix completion problem: Transform the table into a N users by M movies matrix R



Rating matrix

	Chained	Frozen	Bambi	Titanic	Goodfellas	Dumbo	Twilight	Thor	Tangled
Ninja	2	3	?	?	?	?	?	1	?
Cat	4	?	5	?	?	?	?	?	?
Angel	?	?	?	3	5	5	?	?	?
Nursey	?	?	?	?	?	?	2	?	?
Tongey	?	5	?	?	?	?	?	?	?
Neutral	?	?	?	?	?	?	?	?	1

- Data: Users rate some movies. $R_{user,movie}$. Very sparse
- Task: Finding missing data, e.g. for recommending new movies to users. Fill in the question marks
- Algorithms: Alternating Least Square method, Gradient Descent, Non-negative Matrix Factorization, low rank matrix Completion, etc.

Latent factor models

- In our current setting, latent factor models attempt to explain the ratings by characterizing both movies and users on a number of factors K inferred from the ratings patterns.
- That is, we seek representations for movies and users as vectors in $\mathbb{R}^K$ that can ultimately be translated to ratings.
- For simplicity, we can associate these factors (i.e. the dimensions of the vectors) with idealized concepts like
  - comedy
  - drama
  - action
  - But also uninterpretable dimensions

Can we use the sparse ratings matrix R to find these latent factors automatically?

Alternating least squares

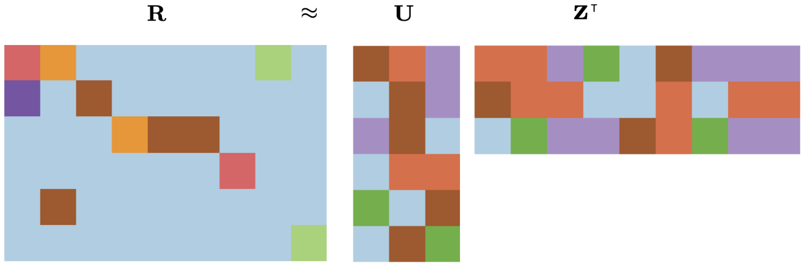
- Let the representation of user n in the K-dimensional space be $\mathbf{u}_n$ and the representation of movie m be $\mathbf{z}_m$
- Assume the rating user n gives to movie m is given by a dot product: $R_{nm} \approx \mathbf{u}_n^T \mathbf{z}_m$
- In matrix form, if:

$$\mathbf{U} = \begin{bmatrix} - & \mathbf{u}_1^\top & - \\ & \vdots & \\ - & \mathbf{u}_N^\top & - \end{bmatrix} \text{ and } \mathbf{Z}^\top = \begin{bmatrix} | & & | \\ \mathbf{z}_1 & \dots & \mathbf{z}_M \\ | & & | \end{bmatrix}$$

  then: $\mathbf{R} \approx \mathbf{U}\mathbf{Z}^\top$

- This is a matrix factorization problem!

Approach: Matrix factorization methods



Cost for Matrix Factorization for Recommender Systems

- Recall PCA: To enforce $\mathsf{X}^\top \approx \mathsf{U}\mathsf{Z}^\top$, we minimized

$$\min_{\mathsf{U},\mathsf{Z}} \|\mathsf{X}^\top - \mathsf{U}\mathsf{Z}^\top\|_F^2 = \sum_{i,j} (x_{ji} - \mathsf{u}_i^\top \mathsf{z}_j)^2$$

  where u_i and z_i are the i-th rows of matrices U and Z, respectively.

- How do we enforce $\mathsf{R} \approx \mathsf{U}\mathsf{Z}^\top$
  - Try

$$\min_{\mathsf{U},\mathsf{Z}} \sum_{i,j} (R_{ij} - \mathsf{u}_i^\top \mathsf{z}_j)^2$$

  - Most entries of R are missing!

Alternating least squares

- Let $O = \{(n, m) : \text{ entry } (n, m) \text{ of matrix } \mathbf{R} \text{ is observed}\}$
- Using the squared error loss, a matrix factorization corresponds to solving

$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(n,m)\in O} \left(R_{nm} - \mathbf{u}_n^\top \mathbf{z}_m\right)^2$$

- The objective is non-convex in $\mathbf{U}$ and $\mathbf{Z}$ and in fact it's generally NP-hard to minimize the above cost function.
- As a function of either $\mathbf{U}$ or $\mathbf{Z}$ individually, the problem is convex and easy to optimize. We can use coordinate descent, just like with K-means and mixture models!

Alternating Least Squares (ALS): fix $\mathbf{Z}$ and optimize $\mathbf{U}$, followed by fix $\mathbf{U}$ and optimize $\mathbf{Z}$, and so on until convergence.

Alternating least squares

ALS for Matrix Completion algorithm

```
1. Initialize U and Z randomly
2. repeat until convergence
3.     for $n = 1, .., N$ do
4.         $\mathrm{u}_n = \left(\sum_{m:(n,m)\in O} \mathrm{z}_m \mathrm{z}_m^\top\right)^{-1} \sum_{m:(n,m)\in O} R_{nm}\mathrm{z}_m$
5.     for $m = 1, .., M$ do
6.         $\mathrm{z}_m = \left(\sum_{n:(n,m)\in O} \mathrm{u}_n \mathrm{u}_n^\top\right)^{-1} \sum_{n:(n,m)\in O} R_{nm}\mathrm{u}_n$
```

Gradient descent method

- We can also do full gradient descent for matrix completion.
- Minimize $f(\mathsf{U}, \mathsf{Z})$ with GD. Both U, Z are variables. Gradient descent step:

$$\begin{bmatrix} \mathsf{U} \\ \mathsf{Z} \end{bmatrix} \leftarrow \begin{bmatrix} \mathsf{U} \\ \mathsf{Z} \end{bmatrix} - \alpha \nabla f(\mathsf{U}, \mathsf{Z}) \tag{1}$$

- Computation of the gradient term per iteration is expensive if all the index pairs in the ratings matrix are considered and R is large (e.g. Netflix).

Stochastic gradient descent method

Stochastic gradient descent for matrix completion (recall SGD from lecture 5). Attempt to minimize $f(\mathrm{U}, \mathrm{Z}) = \frac{1}{2}\sum_{(n,m)\in O}\left(R_{nm} - \mathrm{u}_n^\top \mathrm{z}_m\right)^2$.
For a randomly chosen observed pair (n, m) in R, the SGD update:

$$\begin{bmatrix} \mathrm{u}_n \\ \mathrm{z}_m \end{bmatrix} \leftarrow \begin{bmatrix} \mathrm{u}_n \\ \mathrm{z}_m \end{bmatrix} - \alpha \begin{bmatrix} \left(R_{nm} - \mathrm{u}_n^\top \mathrm{z}_m\right)\mathrm{z}_m \\ \left(R_{nm} - \mathrm{u}_n^\top \mathrm{z}_m\right)\mathrm{u}_n \end{bmatrix} \tag{2}$$

Algorithm:

```
1. Initialize $\mathrm{U}$ and $\mathrm{Z}$
2. repeat until “convergence”
3.     Randomly select a pair $(n, m) \in O$ among observed elements of $\mathrm{R}$
4.     $\mathrm{u}_n \leftarrow \mathrm{u}_n - \alpha\left(R_{nm} - \mathrm{u}_n^\top \mathrm{z}_m\right)\mathrm{z}_m$
5.     $\mathrm{z}_m \leftarrow \mathrm{z}_m - \alpha\left(R_{nm} - \mathrm{u}_n^\top \mathrm{z}_m\right)\mathrm{u}_n$
```

Netflix Prize