CS 3750 Machine Learning
Lecture 6

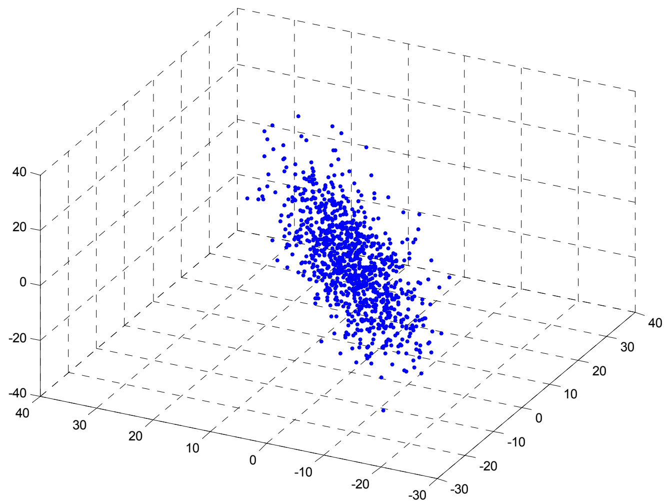
PCA + SVD

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

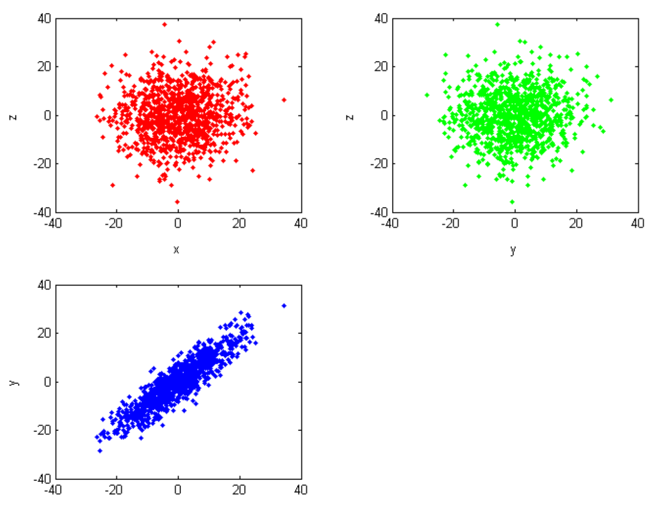
Principal component analysis (PCA)

- **Objective:** We want to replace a high dimensional input with a small subset of set of features
- **Principal component analysis (PCA):**
  - A linear transformation of *d* dimensional input *x* to M dimensional feature vector *z* such that $M < d$ under which the retained variance is maximal.
  - Equivalently it is the linear projection for which the sum of squares reconstruction cost is minimized.

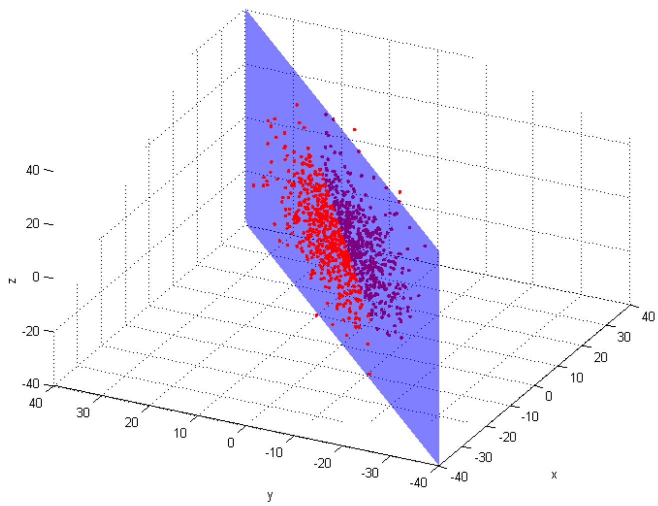
PCA



PCA

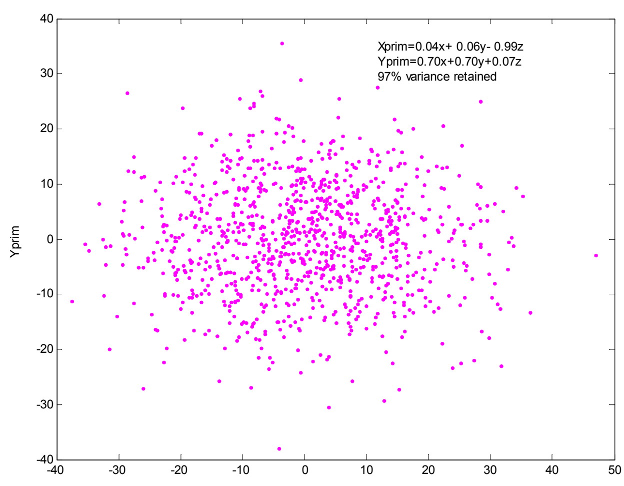


PCA



z
y
x

PCA



Xprim=0.04x+ 0.06y- 0.99z
Yprim=0.70x+0.70y+0.07z
97% variance retained
Yprim
Xprim

Principal component analysis (PCA)

- **PCA:**
  - linear transformation of d dimensional input x to M dimensional feature vector z such that $M < d$ under which the retained variance is maximal.
  - Task independent
- **Fact:**
  - A vector x can be represented using a set of **orthonormal** vectors u

$$\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i$$

  - Leads to transformation of coordinates (from x to z using u's)

$$z_i = \mathbf{u}_i^T \mathbf{x}$$

PCA

- **Idea:** replace d coordinates with M of z_i coordinates to represent x. We want to find the subset M of basis vectors.

$$\tilde{\mathbf{x}} = \sum_{i=1}^{M} z_i \mathbf{u}_i + \sum_{i=M+1}^{d} b_i \mathbf{u}_i$$

b_i - constant and fixed

- **How to choose the best set of basis vectors?**
  - We want the subset that gives the best approximation of data x in the dataset on average (we use least squares fit)

Error for a data entry $\mathbf{x}^n$ $\quad \mathbf{x}^n - \tilde{\mathbf{x}}^n = \sum_{i=M+1}^{d} (z_i^n - b_i)\mathbf{u}_i$

$$E_M = \frac{1}{2}\sum_{n=1}^{N} \left\| \mathbf{x}^n - \tilde{\mathbf{x}}^n \right\|_{L2} = \frac{1}{2}\sum_{n=1}^{N} \sum_{i=M+1}^{d} (z_i^n - b_i)^2$$

PCA

- **Differentiate the error function** with regard to all b_i and set equal to 0 we get:

$$b_i = \frac{1}{N}\sum_{n=1}^{N} z_i^n = \mathbf{u}_i^T \overline{\mathbf{x}} \qquad \overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^n$$

- Then we can rewrite:

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d} \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{u}_i \qquad \mathbf{\Sigma} = \sum_{n=1}^{N} (\mathbf{x}^n - \overline{\mathbf{x}})(\mathbf{x}^n - \overline{\mathbf{x}})^T$$

- We want to optimize the error function over basis vectors $\mathbf{u}_i$:

$$E_M = \min_{\text{subsets of } u_i} \frac{1}{2}\sum_{i=M+1}^{d} \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{u}_i$$

PCA

- The error function

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d} \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{u}_i \qquad \mathbf{\Sigma} = \sum_{n=1}^{N} (\mathbf{x}^n - \overline{\mathbf{x}})(\mathbf{x}^n - \overline{\mathbf{x}})^T$$

is optimized in terms of basis vectors $\mathbf{u}_i$ when they satisfy:

$$\mathbf{\Sigma}\mathbf{u}_i = \lambda_i \mathbf{u}_i \Rightarrow \lambda_i \mathbf{u}_i^T \mathbf{u}_i = \lambda_i 1 \Rightarrow E_M = \frac{1}{2}\sum_{i=M+1}^{d} \lambda_i$$

Vectors $\mathbf{u}_i$ that satisfy:

- **correspond to eigenvectors of $\mathbf{\Sigma}$**

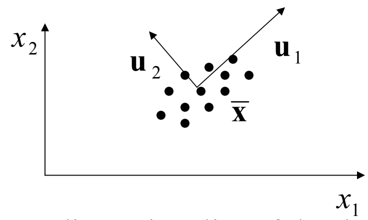
The best *M* basis vectors that lead to the optimal error:

discard vectors with *d-M* smallest eigenvalues (or keep vectors with M largest eigenvalues)

Eigenvector $\mathbf{u}_i$ – is called a **principal component**

PCA

- Once eigenvectors $\mathbf{u}_i$ with largest eigenvalues are identified, they are used to transform the original *d*-dimensional data to *M* dimensions



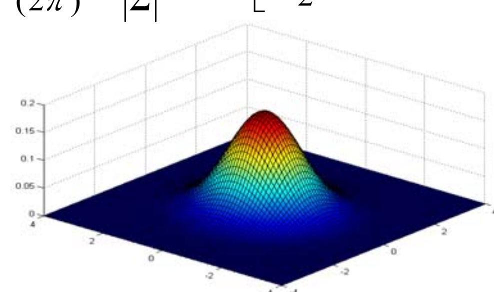
- To find the "true" dimensionality of the data *d'* we can just look at eigenvalues that contribute the most (small eigenvalues are disregarded)
- **Problem:** PCA is a linear method. The "true" dimensionality can be overestimated. There can be non-linear correlations.

PCA and its relations

- **Multivariate normal:** $\mathbf{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- **Parameters:** $\boldsymbol{\mu}$ - mean
  $\boldsymbol{\Sigma}$ - covariance matrix
- **Density function:**

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}) \right]$$

- **Example:**



Parameter estimates

- **Loglikelihood** $l(D,\boldsymbol{\mu},\boldsymbol{\Sigma}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i \mid \boldsymbol{\mu},\boldsymbol{\Sigma})$
- **ML estimates of the mean and covariances:**

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$$

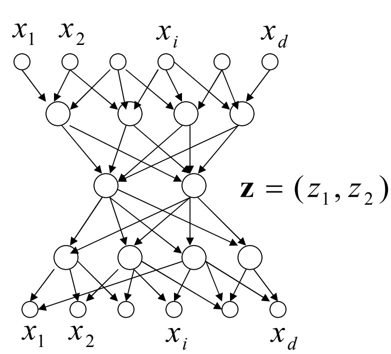
- **Unbiased estimate:** $\hat{\boldsymbol{\Sigma}} = \frac{1}{n-1}\sum_{i=1}^{n} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$
- **Notice that in PCA we want to optimize (minimize):**

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d} \mathbf{u}_i^T \boldsymbol{\Sigma} \mathbf{u}_i \qquad \boldsymbol{\Sigma} = \sum_{n=1}^{N} (\mathbf{x}^n - \overline{\mathbf{x}})(\mathbf{x}^n - \overline{\mathbf{x}})^T$$

  – Differs by a scalar from the covariance matrix

Dimensionality reduction with neural nets

- **PCA** is limited to linear dimensionality reduction
- To do non-linear reductions we can use neural nets
- **Auto-associative network:** a neural network with the same inputs and outputs ($\boldsymbol{x}$)



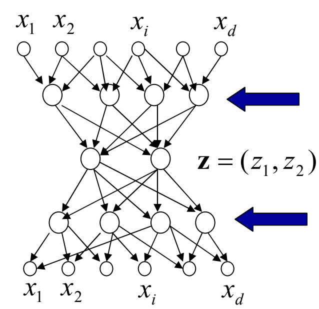
- The middle layer corresponds to the reduced dimensions

Dimensionality reduction with neural nets

- **Error criterion:**

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{d}\left(y_i(x^n) - x^n\right)^2$$

- Error measure tries to recover the original data through limited number of dimensions in the middle layer
- **Non-linearities** modeled through intermediate layers between the middle layer and input/output
- If no intermediate layers are used the model replicates PCA optimization through learning



Networks with radial basis functions

- An alternative to **multilayer NN for non-linearities**
- Radial basis functions: $f(x) = w_0 + \sum_{j=1}^{k} w_j \phi_j(\mathbf{x})$
  - Based on interpolations of prototype points (**mean**s)
  - Affected by the distance between the **x** and the **mean**
  - Fit the outputs of basis functions through the linear model
- Choice of basis functions:

Gaussian $\quad \phi_j(x) = \exp\left\{\frac{\left\|x - \mu_j\right\|^2}{2\sigma_j^2}\right\}$

- **Learning:**
  - In practice seem to work OK for up to 10 dimensions
  - For higher dimensions (ridge functions – logistic) combining multiple learners seem to do better job