Statistical Methods in Artificial Intelligence
CSE471 - Monsoon 2016 : Lecture 14

Avinash Sharma

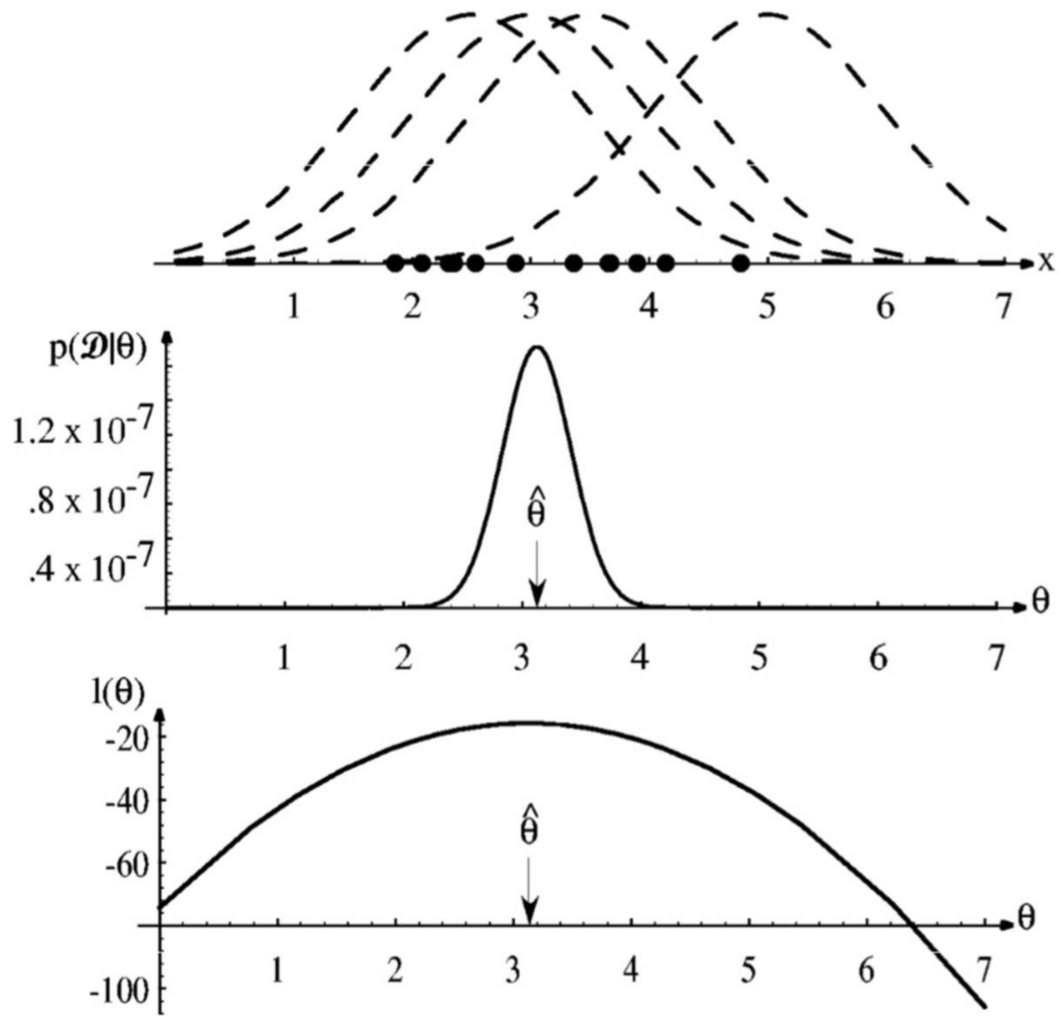
CVIT, IIIT Hyderabad

Lecture Plan

- Revision from Previous Lecture
- Principal Component Analysis (PCA)
  - Derivation
  - Practical Example
- Eigen Faces
  - Algorithm Overview
  - Eigenface Plots/Code
  - Practical Tricks
- Discriminant Analysis (Next Class)

Maximum Likelihood Estimation

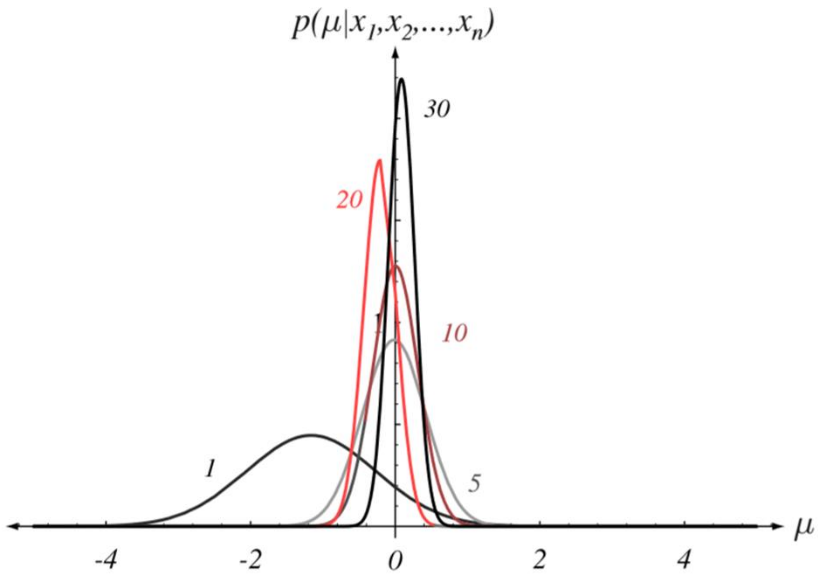
$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta})$$

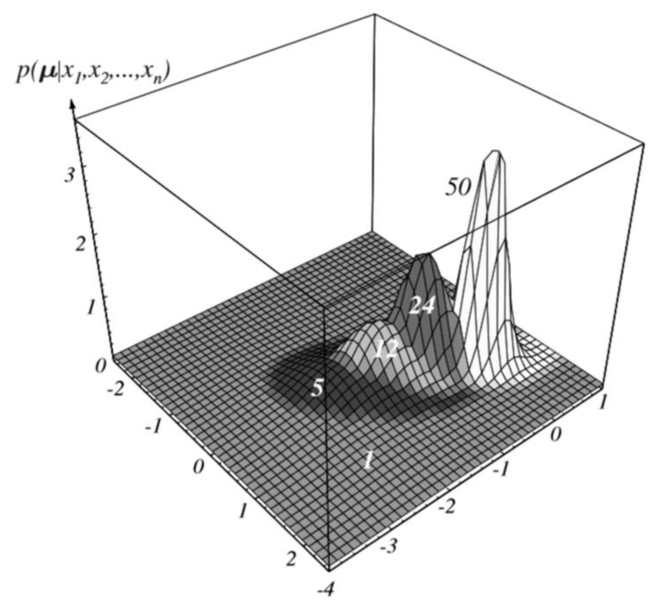


BPE: General Theory

$$p(\mathbf{x}|\mathcal{D}) = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\ d\boldsymbol{\theta} \qquad p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\ d\boldsymbol{\theta}},$$

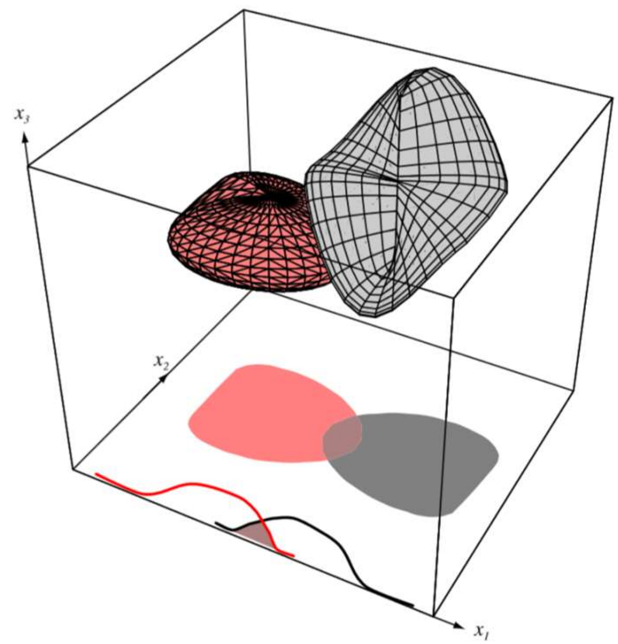
$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta}).$$

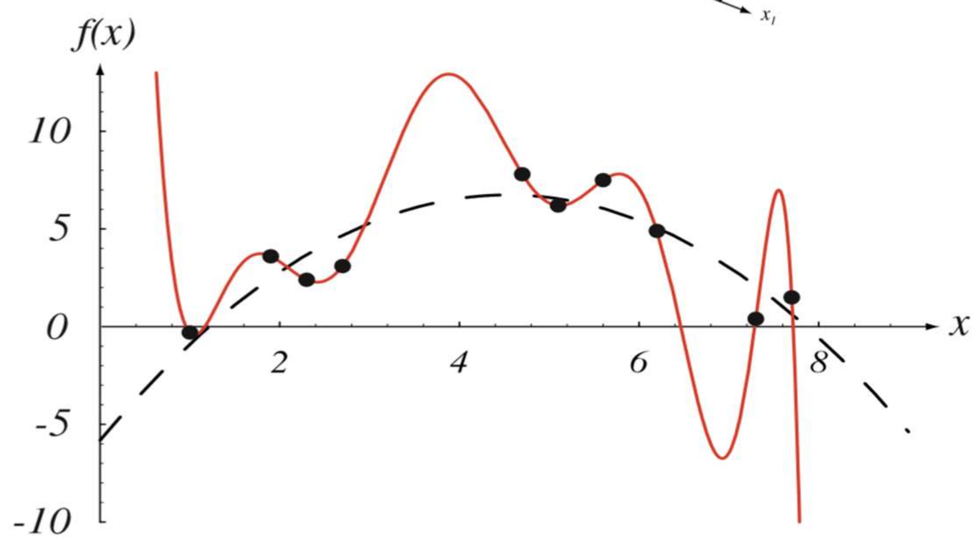




Problems of Dimensionality

- Dimensions
  - More the Merrier (?)
    - Wrong Model Choice
    - Small Training Sample Size
- Computational Complexity
  - Order of operations
- Overfitting





Principal Component Analysis (PCA)

- "Curse of Dimensionality" !
  - Lack of enough data samples
  - Computational Intractability
- Large number of features might be redundant i.e., only few have relevant information for classification task.
- Feature selection is not always the best option as many feature dimensions are highly correlated.
- PCA achieves dimensionality reduction by linearly combining multiple features (though with some information loss).

Principal Component Analysis (PCA)

- PCA assumes that the information is carried in the variance of the features, i.e., higher variance of features implies more information (more Entropy).
- The idea is to find a lower dimensional projection of data while "*preserving the* ***most of the variance*** *of data*".
- The variance is preserved in the ***least square*** sense, i.e., sum of least square error between original data points and their projection is minimized.

Principal Component Analysis (PCA)

- Let data matrix $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n\}$ and $\mathbf{x}_i \in \mathbb{R}^d$.
- 1-dimensional representation: Let $\mathbf{x} = \mathbf{x}_0$

$$J_0(\mathbf{x}_0) = \sum_{i=1}^{n} \|\mathbf{x}_0 - \mathbf{x}_i\|^2, \qquad \mathbf{m} = \arg\min_{\mathbf{x}_0} J_0(\mathbf{x}_0) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

- 2-dimensional representation: Let $\mathbf{x} = \mathbf{m} + a\mathbf{e}$

$$J_1(a_1, \dots, a_n, \mathbf{e}) = \sum_{i=1}^{n} \|(\mathbf{m} + a_{\boldsymbol{i}}\mathbf{e}) - \mathbf{x}_i\|^2, \qquad a_i = \mathbf{e}^{\boldsymbol{T}}(\mathbf{x}_i - \mathbf{m})$$

$$J_1(\mathbf{e}) = -\mathbf{e}^{\boldsymbol{T}}\mathbf{S}\mathbf{e}^{\boldsymbol{T}} + \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{m}\|^2, \qquad \mathbf{v} = \arg\min_{\mathbf{e}} J_1(\mathbf{e})$$

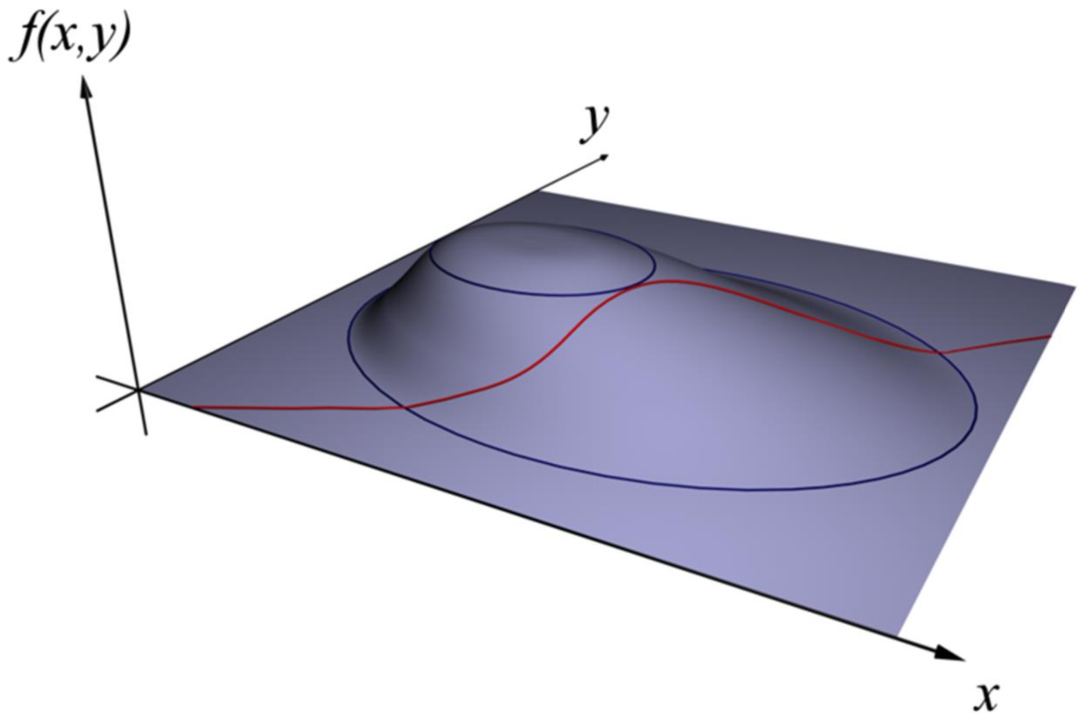
$$\mathbf{S}\mathbf{v} = \lambda\mathbf{v}, \qquad \|\mathbf{v}\| = 1 \text{ where } \mathbf{S} = \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\boldsymbol{T}}$$

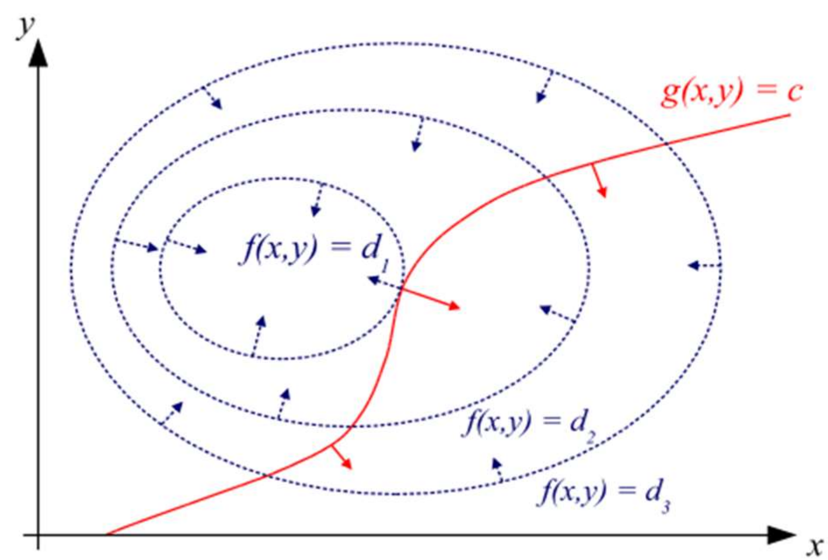
Lagrangian Multiplier

$$\max\ f(x,y),\ \ \text{Subject to}\ \ g(x,y) = 0$$

$$\mathcal{L}(x,y,\lambda) = f(x,y) - \lambda g(x,y)$$

$$\nabla_{x,y} f(x,y) = \lambda\, \nabla_{x,y} g(x,y)$$





Principal Component Analysis (PCA)

- k -dimensional representation:

Let $\mathbf{x} = \mathbf{m} + \sum_{i=1}^{k} a_i \mathbf{e}_i$

$$\mathbf{v}_1, \ldots, \mathbf{v}_k = arg \max_{\mathbf{e}_{1,} \ldots, \mathbf{e}_k} J_k = \sum_{i=1}^{n} \left\| \left(\mathbf{m} + \sum_{j=1}^{k} a_j \mathbf{e}_j \right) - \mathbf{x}_i \right\|^2 ,$$

$$\text{for } k \ll d$$

where,

$$\mathbf{S}\mathbf{v}_i = \lambda_i \mathbf{v}_i \, ,$$

$$\mathbf{v}_i \perp \mathbf{v}_j \, , \|\mathbf{v}_i\| = 1 \; \forall \; i, j \; \in \{1, \ldots, k\}$$

PCA: Practical Example

- Let $\mathbf{X} = \begin{bmatrix} 1\ 3\ 3\ 5\ 5\ 6\ 8\ 9 \\ 2\ 3\ 5\ 4\ 6\ 5\ 7\ 8 \end{bmatrix}$
- $\mathbf{m} = \frac{1}{8}\begin{bmatrix} 40 \\ 40 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$
- $\mathbf{S} = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$
- $|\mathbf{S} - \lambda\mathbf{I}| = 0,$

  $$\begin{vmatrix} 6.25 - \lambda & 4.25 \\ 4.25 & 3.5 - \lambda \end{vmatrix} = 0$$

  $\lambda_1 = 0.41, \lambda_2 = 9.34$
- $\mathbf{S}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2]$

  $\mathbf{v}_1 = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$

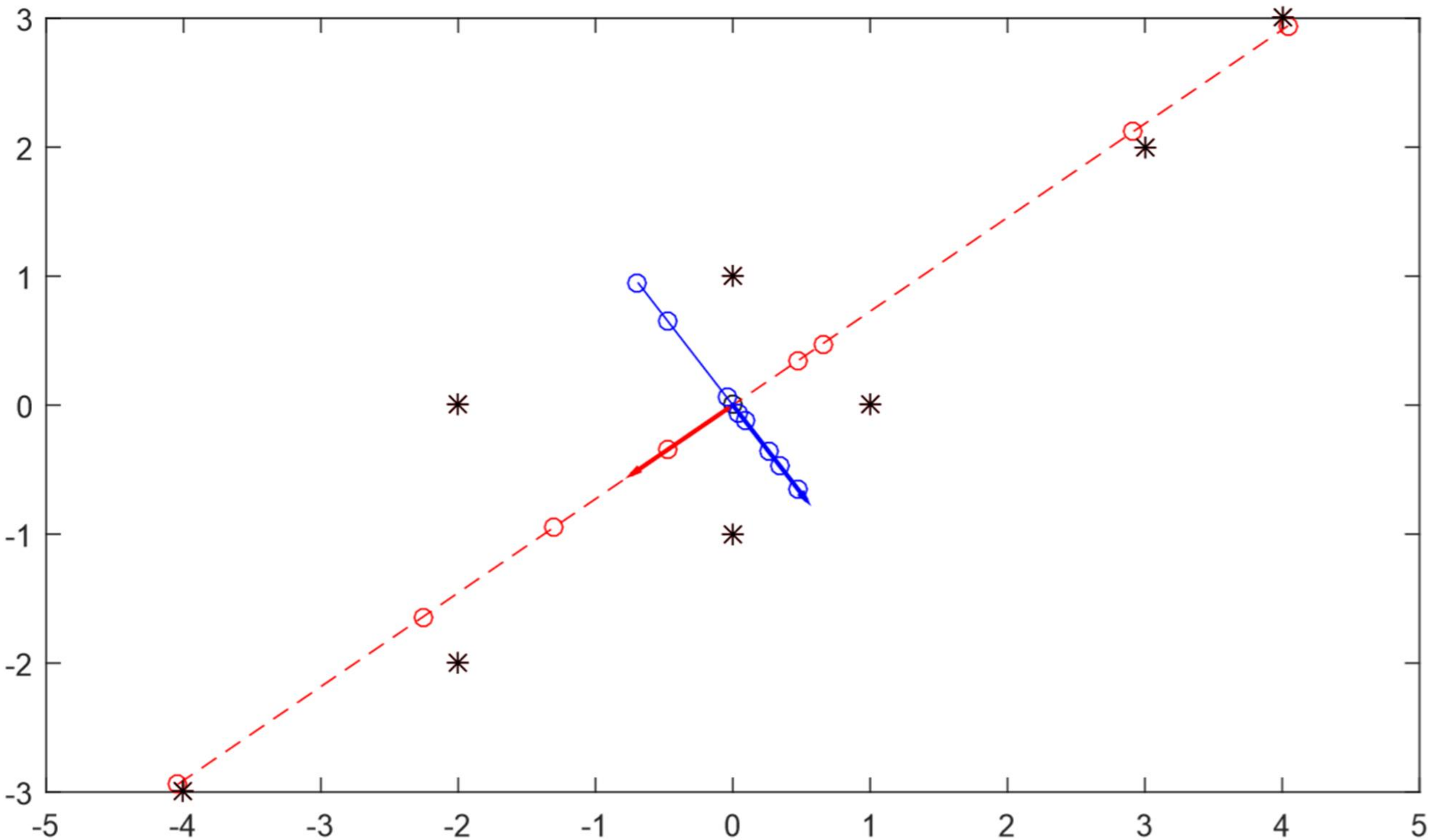
```
X=[1 3 3 5 5 6 8 9;2 3 5 4 6 5 7 8]
m=mean(X')
temp=X-repmat(m',1,8);

S=zeros(2,2);
for i=1:size(X,2)
    t=temp(:,i) ;
    S=S+t*t';
end
S=S/size(X,2); % Optional

[V D]=eig(S);

v1=V(:,1);
v2=V(:,2); % direction of max
variance
```

PCA: Practical Example



Eigen Faces: PCA of Face Data

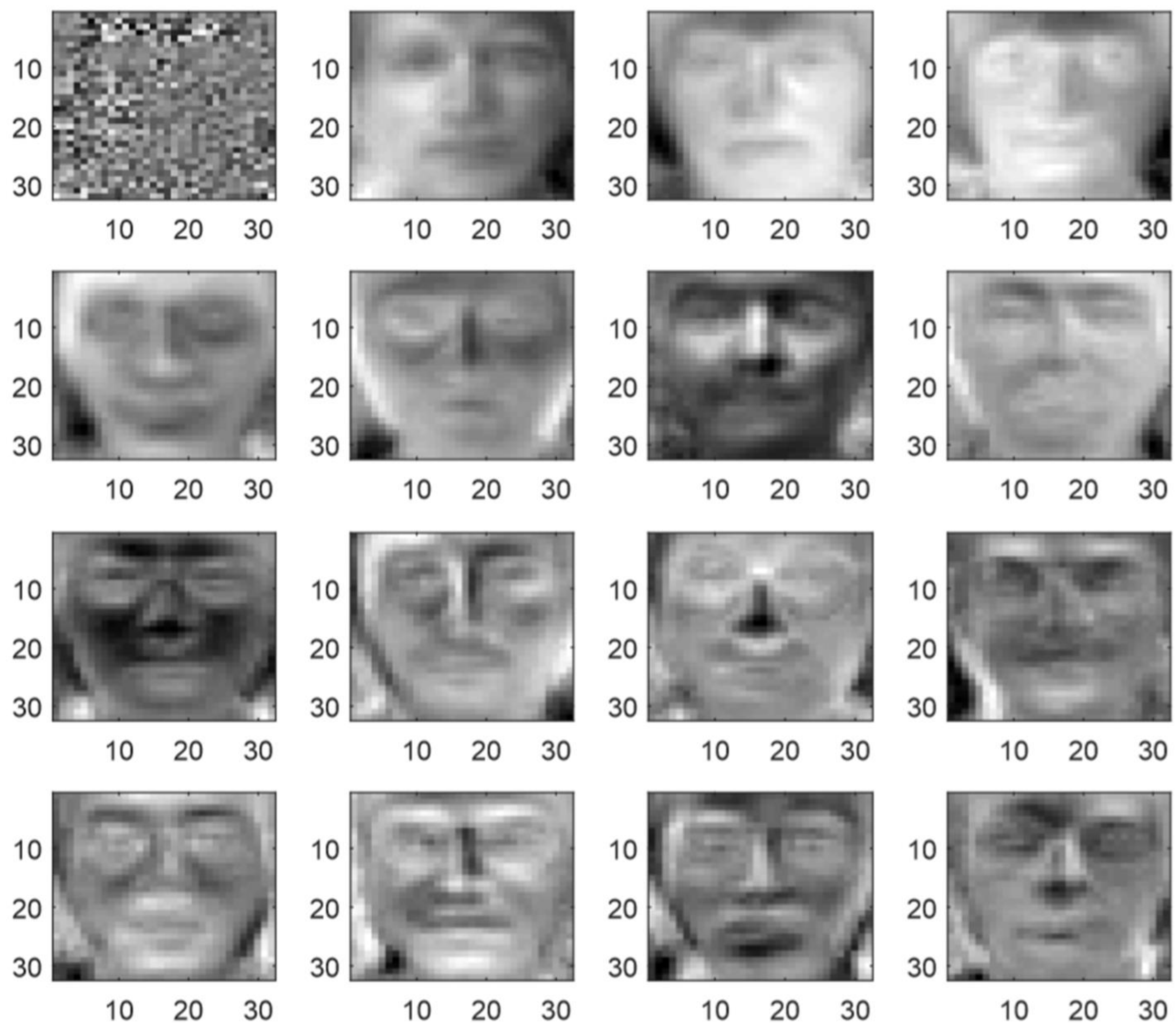
Eigen Faces: PCA of Face Data

Eigenfaces for Recognition,
M. Turk, A. Pentland,
Journal of Cognitive Neuroscience, 3(1), pp. 71-86, 1991.

Algorithm to compute eigenfaces:

1. Vectorize grey scale images to $d \times 1$ vector (from w $\times$ h matrix)
2. Computer Mean Vector
3. Compute Scatter Matrix ($d \times d$)using the Mean Vector and all data points
4. Compute eigen-values and eigen-vectors of Scatter Matrix
5. Plot eigenVectors corresponding to largest eigen-values after reshaping them to $w \times h$ matrix
6. Project each training & test shape to k dimensional subspace by multiplying $d \times 1$ vector by matrix of k largest eigenvectors of size $k \times d$

Eigen Faces: PCA of Face Data



Eigen Faces: PCA of Face Data

```
load Yale_32x32
h=32; w=32;
d = h*w;
% vectorize images
%x = reshape(yalefaces,[d n]);
n=165;
x=fea';
x = double(x);
%subtract mean
x=bsxfun(@minus, x', mean(x'))';
% calculate covariance
s = cov(x');
% obtain eigenvalue & eigenvector
[V,D] = eig(s);
eigval = diag(D);
% sort eigenvalues in descending order
eigval = eigval(end:-1:1);
V = fliplr(V);
```

```
% show 0th through 15th principal
%eigenvectors
eig0 = reshape(mean(x,2), [h,w]);
figure,subplot(4,4,1)
imagesc(eig0)
colormap gray
for i = 1:15
subplot(4,4,i+1)
imagesc(reshape(V(:,i),h,w))
end
```

Practical Tricks

- How to choose k?
  - Choose k such that following ratio has value 0.8 or 0.95 (corresponds to % of energy left after dropping $\mathrm{d}-k$ eigenvectors)

  Sum of k largest eigenvalues/sum of all eigenvalues

- How to handle if $\mathrm{d} \gg n$?
  - Instead of computing eigenvector of $d \times d$ Scatter matrix ($\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^T$), compute the $k < n$ eigenvectors of $n \times n$ matrix $\boldsymbol{X}^T\boldsymbol{X}$ (Self Exercise)