Dimensionality reduction

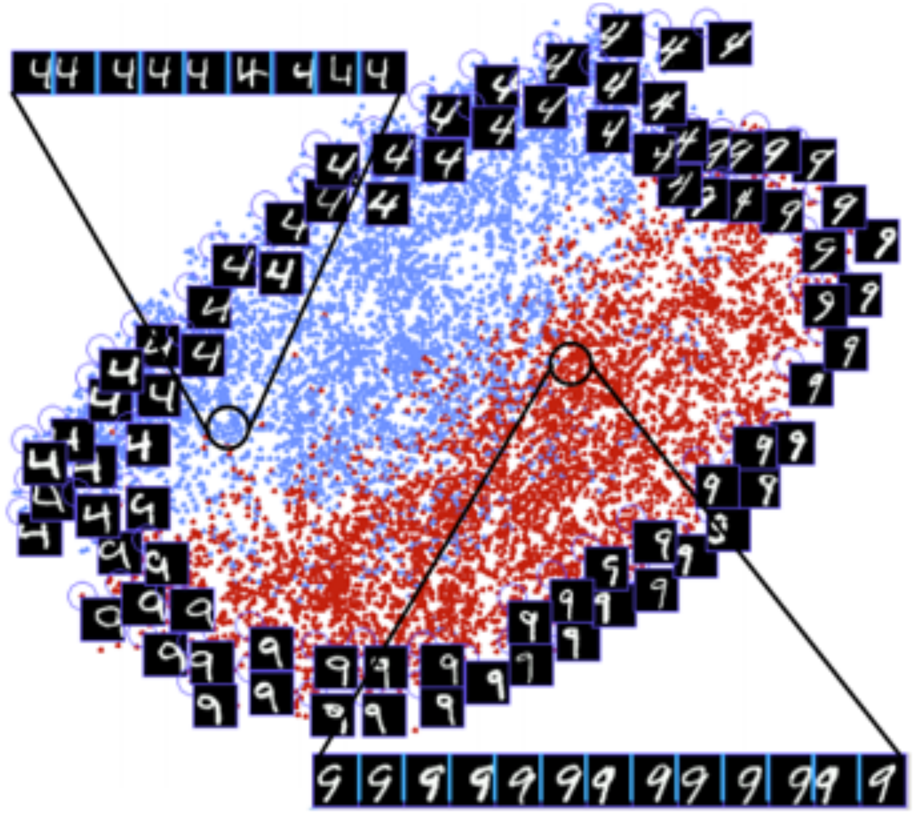
Subhransu Maji

CMPSCI 689: Machine Learning

9 April 2015

Motivation

- Data visualization
  - Hard to visualize data that lives in high dimensions — reduce it two two or three dimensions for visualization



[Hadsell et al, CVPR 06]

- Curse of dimensionality
  - Some learning methods don’t scale well with the number of features (e.g., kNN, kernel density estimators)
  - Lower memory overhead and training/testing time
  - Fewer dimensions is a form of regularization

Dimensionality reduction

- The goal is to reduce the dimension of the data in high-dimensions (say 10000) to low dimensions (say 2) while retaining the “important” characteristics of the data
- Unsupervised setting, so the notion of important characteristics is hard to define
- Closely related to clustering
  - Clustering: reduce the number of data
  - Dimensionality reduction: reduce the number of features

dim reduction: features

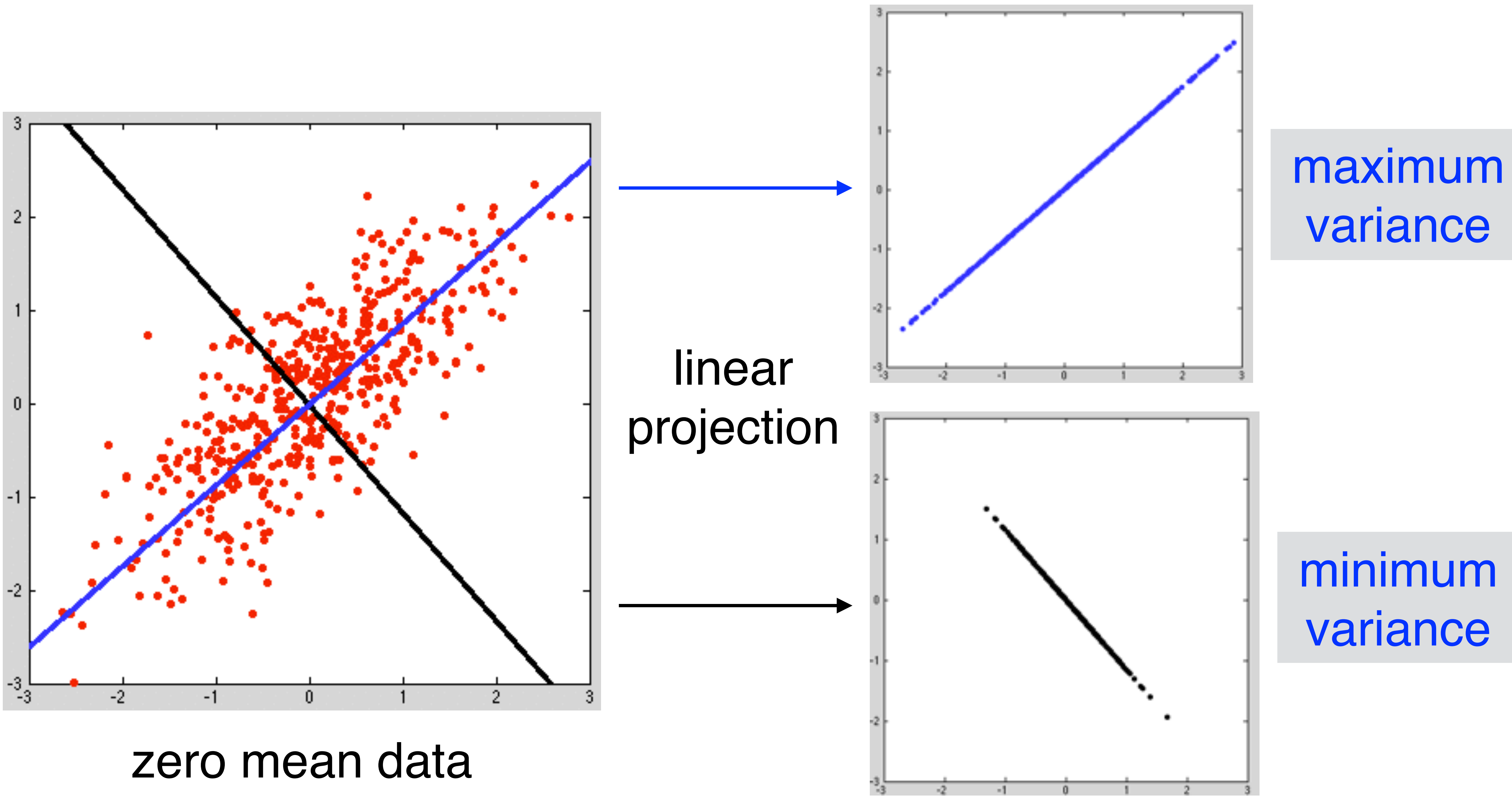
clustering: data

$$\mathbf{x}_i \in R^D, i = 1, 2, \ldots, N$$

$$\text{data matrix} = R^{N \times D}$$

Linear dimensionality reduction

- All you can do is project the data onto a vector and use the projected distances as the embeddings
- Example: projecting two dimensional data to one



Optimal linear projection

- Find a linear projection that maximizes the variance of the projection
- Assume we have data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in R^D$ of zero mean
- Let $\mathbf{u}$ be the projection vector
- Let the projections of the data $p_1, p_2, \ldots, p_N$

$$p_i \leftarrow \mathbf{x}_i^T \mathbf{u}$$

- The mean of the projections is zero

$$\sum_i p_i = \sum_i \mathbf{x_i}^T \mathbf{u} = \left(\sum_i \mathbf{x_i} \right)^T \mathbf{u} = 0$$

- Maximize the variance of the projection:

$$\max_{\mathbf{u}} \sum_i \left(\mathbf{x}_i^T \mathbf{u} \right)^2 \text{ subject to: } ||u|| = 1$$

Optimal linear projection

- Lets rewrite this in matrix notation
- Let **X** be the NxD data matrix (each row is data point)
- The projection vector **u** is a Dx1 matrix
- The vector of projections is given by **Xu**, a Nx1 matrix
- We can rewrite the optimization as:

$$\max_{\mathbf{u}} ||\mathbf{X}\mathbf{u}||^2 \text{ subject to: } \mathbf{u}^T\mathbf{u} = 1$$

- The corresponding Lagrangian is:

$$\mathcal{L}(\mathbf{u}, \lambda) = ||\mathbf{X}\mathbf{u}||^2 - \lambda(\mathbf{u}^T\mathbf{u} - 1)$$

- At maxima:

$$\Delta_u = 2\mathbf{X}^T\mathbf{X}\mathbf{u} - 2\lambda\mathbf{u}$$

$$\implies \left(\mathbf{X}^T\mathbf{X}\right)\mathbf{u} = \lambda\mathbf{u} \longleftarrow \text{eigenvalue problem}$$

Optimal linear projection

- Compute the data covariance matrix $\mathbf{X}^T\mathbf{X}$

$$[\mathbf{X}^T\mathbf{X}]_{ij} = \sum_n \mathbf{x}_{ni}\mathbf{x}_{nj}$$

  - The optimal (maximal variance) projection direction is the first eigenvector of the data covariance matrix

$$\left(\mathbf{X}^T\mathbf{X}\right)\mathbf{u} = \lambda\mathbf{u}$$

- **What about learning a second projection direction?**
  - For non-redundancy additionally require that $\mathbf{v}^T\mathbf{u} = 0$

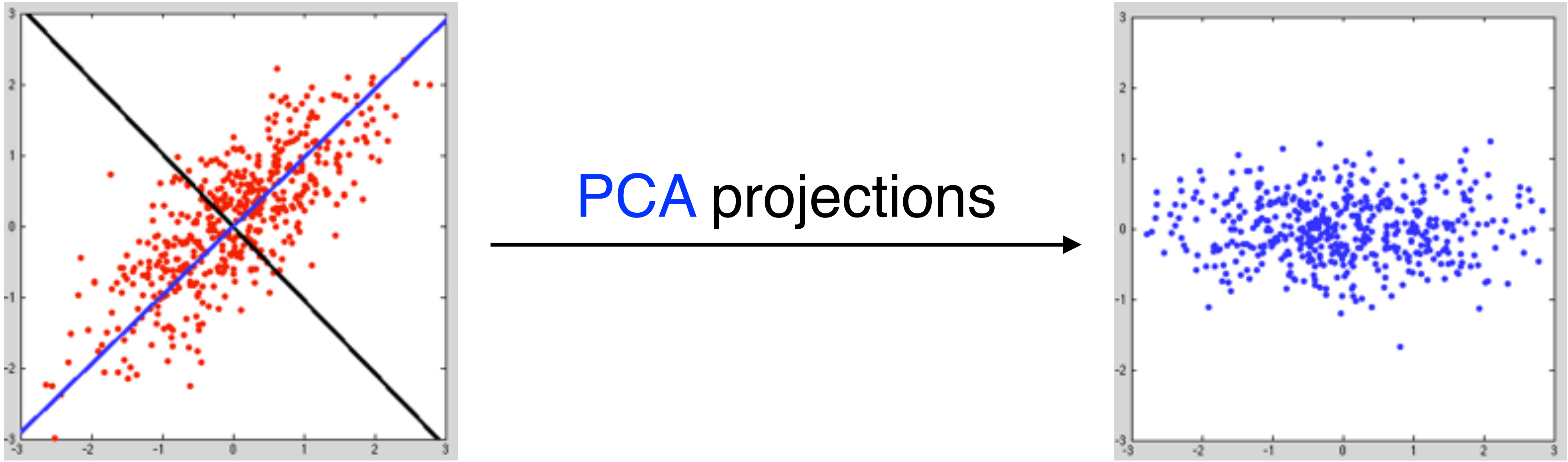
$$\max_{\mathbf{v}} ||\mathbf{X}\mathbf{v}||^2 \text{ subject to: } \mathbf{v}^T\mathbf{v} = 1, \mathbf{v}^T\mathbf{u} = 0$$

  - This is the second eigenvector of the data covariance matrix

Principal component analysis (PCA)

- Generalizing this argument leads to principal component analysis
- The eigenvectors give you the projection directions — to compute the embeddings you have to multiply the data by the projections
- For completeness here is the Matlab code:

```
function [E, U, lambda] = PCA(X, K)
mu = mean(X);                     % Data mean
N = size(X,1);                    % Number of data
X = X - ones(N,1)*mu;             % Center the data
covX = X'*X;                      % Data covaraince
[U,lambda] = eigs(covX, K);  % Compute top K eigenvalues
E = X*U;                          % Compute embeddings
```

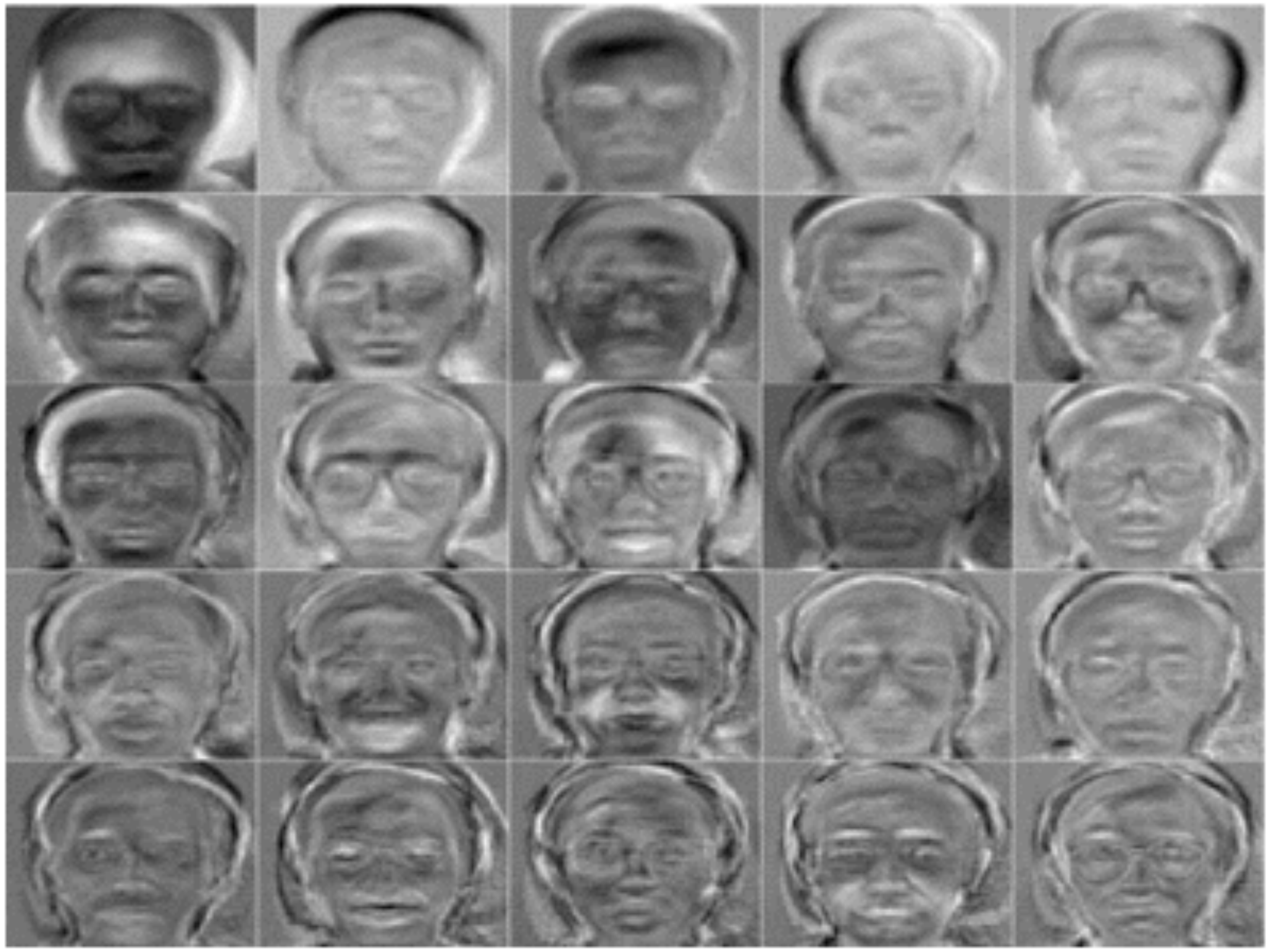


Application: Eigenfaces

- Eigenfaces — a linear basis for face images [Turk, Pentland '91]
- Each face is a weighted linear combination of eigenfaces
- Compare faces by comparing the weights

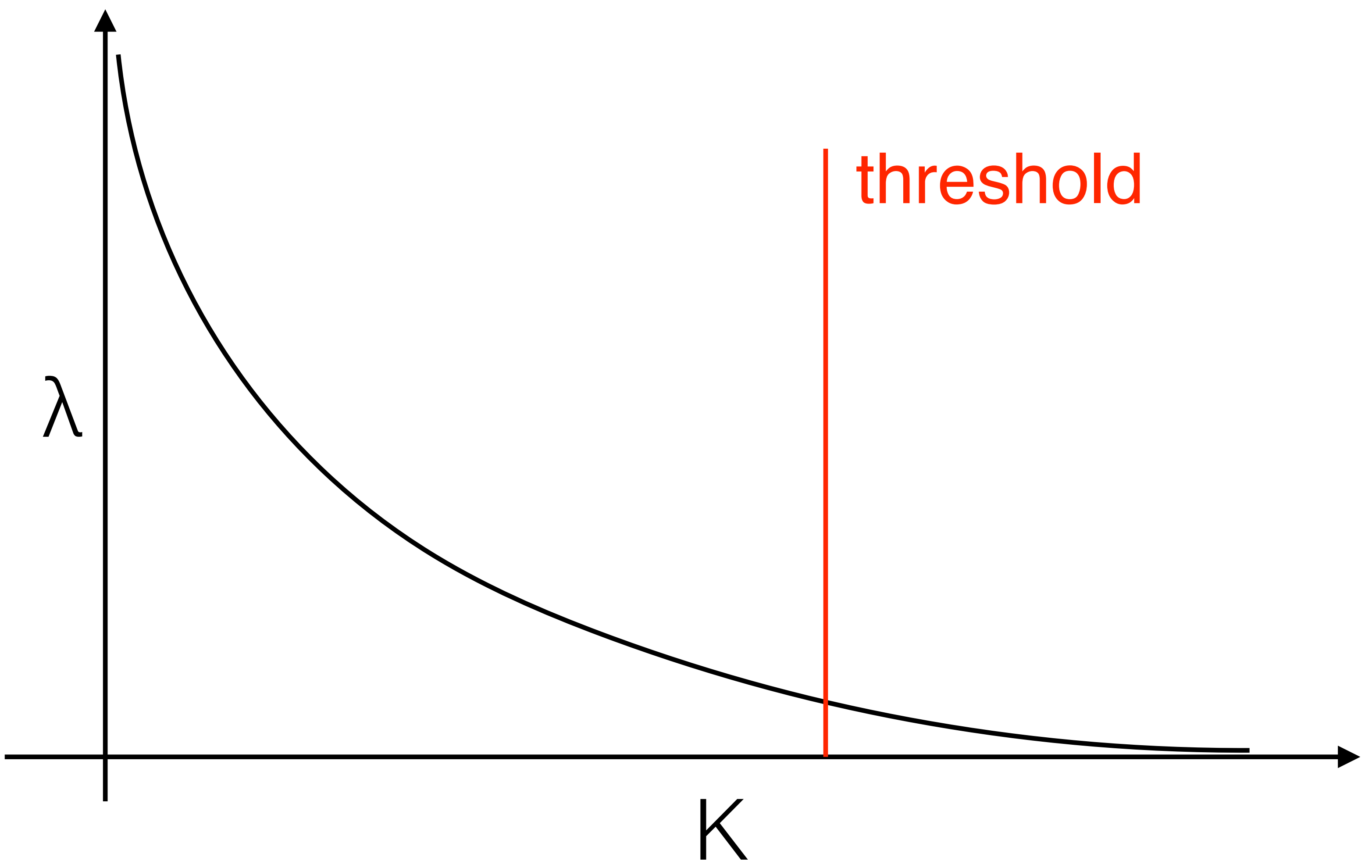
Input images

Principal components



What should K be?

- For visualization K=2 or 3
- For dimensionality reduction it depends on the problem
  - Option: ignore projections that correspond to small eigenvalues
  - Option: based on computational and memory constraints



Kernel PCA

- We can use the kernel trick to learn linear projections in feature space
- PCA representer theorem: the projection direction is a linear combination of the data points

$$\mathbf{u} = \sum_i \alpha_i \phi(\mathbf{x}_i)$$

- PCA using only dot products between the data

$$\max_{\mathbf{u}} \sum_i \left(\phi(\mathbf{x}_i)^T \mathbf{u}\right)^2 \quad \text{subject to: } ||u|| = 1$$

$$\sum_i \left(\phi(\mathbf{x}_i)^T \left(\sum_j \alpha_j \phi(\mathbf{x}_j) \right) \right)^2$$

$$= \sum_i \left(\sum_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \alpha_j \right)^2$$

$$= \sum_i \left(\sum_j \mathbf{K}_{ij} \alpha_j \right)^2 = ||\mathbf{K}\alpha||^2$$

$$\left(\sum_i \phi(\mathbf{x}_i) \alpha_i \right)^T \left(\sum_j \phi(\mathbf{x}_j) \alpha_j \right)$$

$$= \sum_i \sum_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \alpha_i \alpha_j$$

$$= \alpha^T \mathbf{K} \alpha$$

Kernel PCA

- Formulation using kernels:

$$\max_{\alpha} ||\mathbf{K}\alpha||^2 \text{ subject to: } \alpha^T\mathbf{K}\alpha = 1$$

- The corresponding Lagrangian is:

$$\mathcal{L}(\alpha, \lambda) = ||\mathbf{K}\alpha||^2 - \lambda(\alpha^T\mathbf{K}\alpha - 1)$$

- At optimality:

$$\Delta_\alpha \mathcal{L}(\alpha, \lambda) = 2\mathbf{K}^T\mathbf{K}\alpha - 2\lambda\mathbf{K}\alpha$$

$$\implies \mathbf{K}\alpha = \lambda\alpha \text{ , since: } \mathbf{K}^T = \mathbf{K}$$

- Hence α is a eigenvector of the kernel matrix **K**
- Different eigenvectors correspond to different projections

Kernel PCA

- How do we center the data in kernel space?
  - Recall that PCA requires zero mean data
- Centering be written in terms of kernels as well:

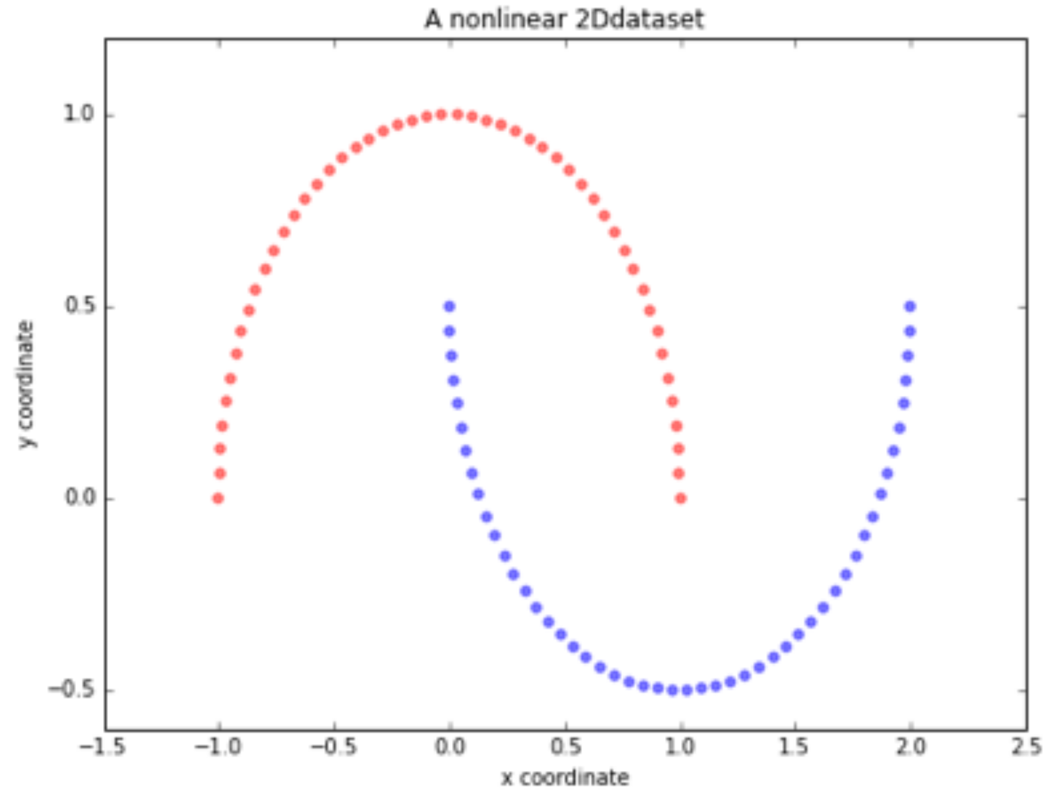
$$
\begin{aligned}
\text{dot product} &= (\phi(\mathbf{x}_i) - \mu)^T (\phi(\mathbf{x}_j) - \mu) \\
&= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) - \mu\phi(\mathbf{x}_i)^T - \mu^T\phi(\mathbf{x}_j) + \mu^T\mu \\
&\implies \mathbf{K}' = \mathbf{K} - \mathbf{1K} - \mathbf{K1} + \mathbf{1K1}
\end{aligned}
$$

- Where the matrix **1** is defined as:

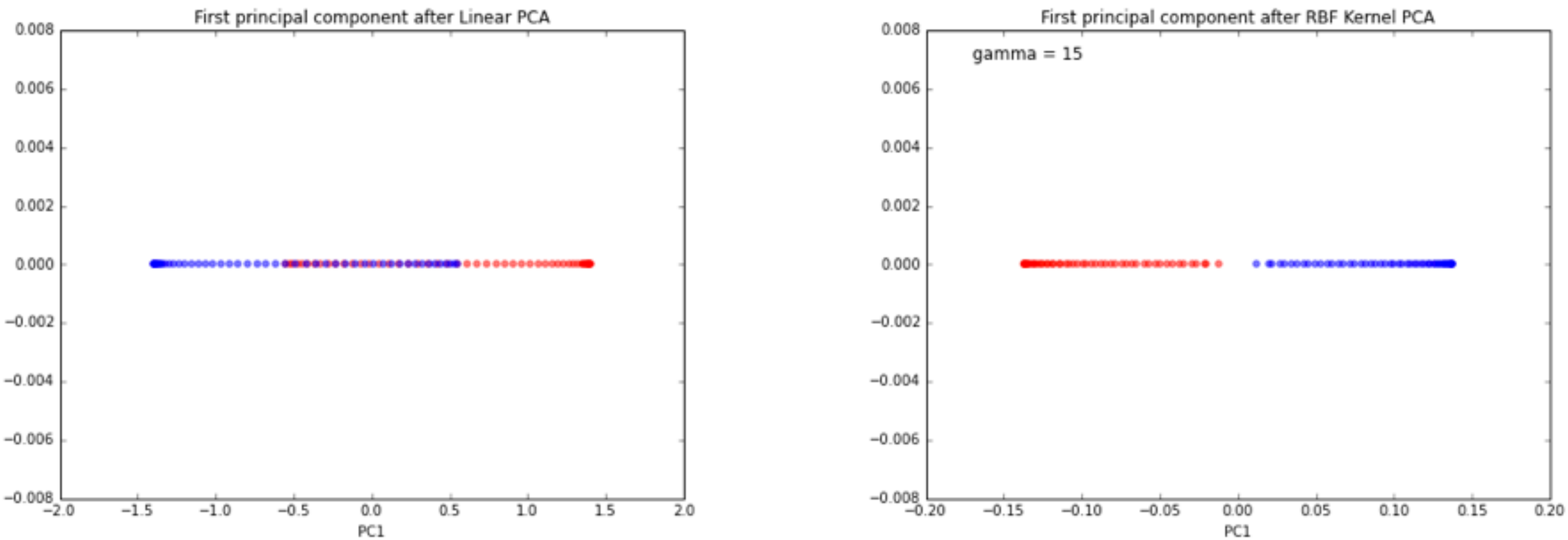
$$\mathbf{1}_{ij} = \frac{1}{N}$$

- Perform PCA on the **K**' matrix and compute the eigenvectors α
- Projections of the data are **K**'α

Linear and kernel PCA



$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\gamma \, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \right)$$



Spectral clustering revisited

- Normalized cuts objective:

$$\min_{\mathbf{x}} \mathrm{NCut}(\mathbf{x}) = \min_{\mathbf{y}} \frac{\mathbf{y}^T(D-W)\mathbf{y}}{\mathbf{y}^T D\mathbf{y}}$$

$$\text{subject to: } \mathbf{y}^T D\mathbf{1} = 0$$

$$\mathbf{y}(i) \in \{1, -b\}$$

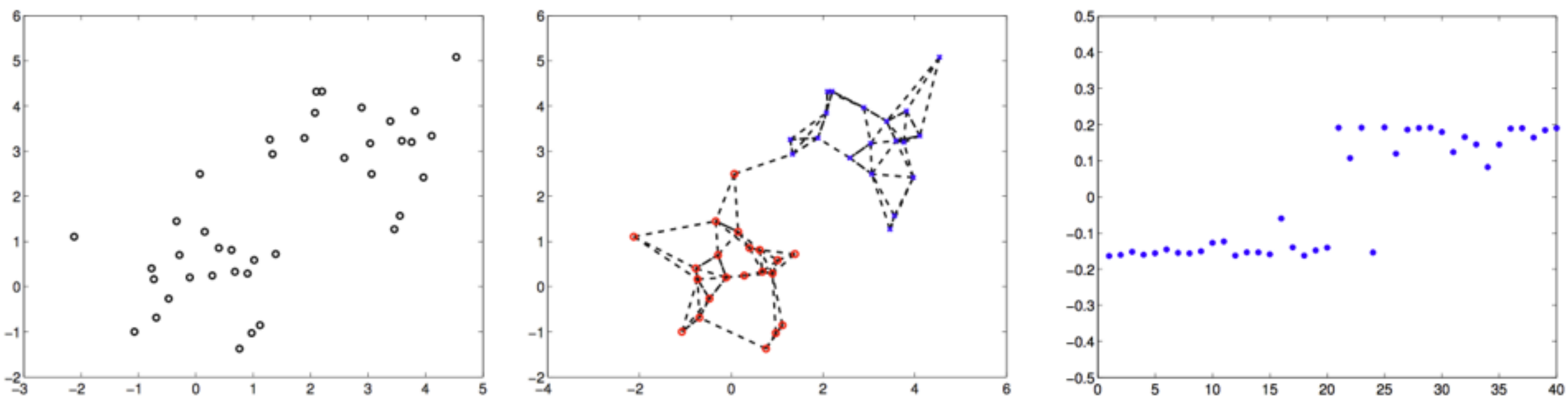
- Relax the integer constraint on **y**:

$$\min_{\mathbf{y}} \mathbf{y}^T(D-W)\mathbf{y}; \text{ subject to: } \mathbf{y}^T D\mathbf{y} = 1$$

- Same as: $(D-W)\mathbf{y} = \lambda D\mathbf{y}$ (generalized eigenvalue problem)
- Note that $(D-W)\mathbf{1} = 0$, so the first eigenvector is $\mathbf{y}_1 = \mathbf{1}$, with the corresponding eigenvalue of 0
- The eigenvector corresponding to the second smallest eigenvalue is the solution to the relaxed problem

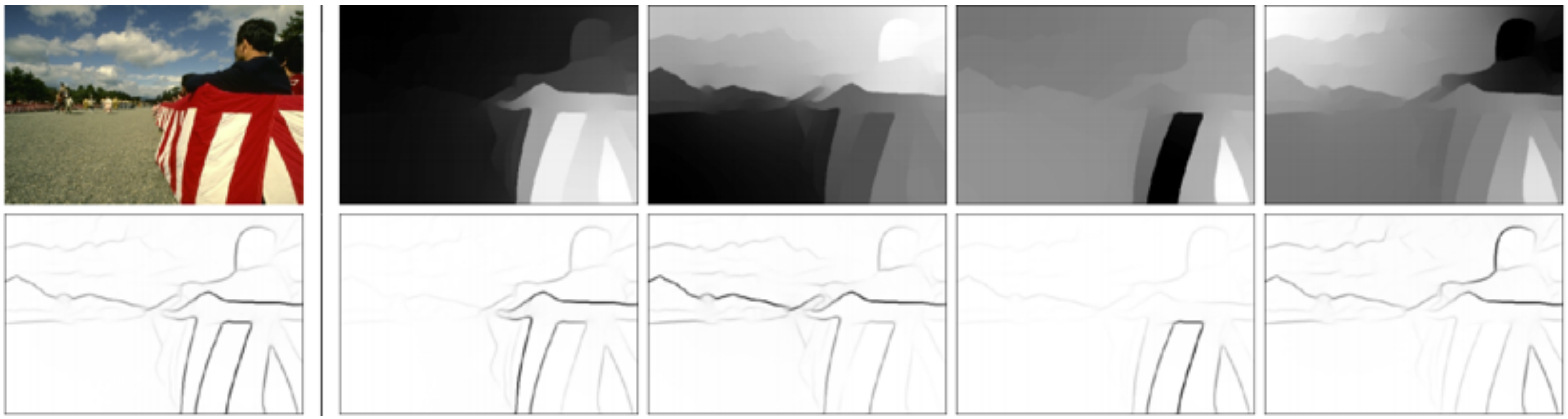
Spectral clustering example

- Spectral clustering = spectral embedding + thresholding (or k-means)
- Recall the earlier example
  - Gaussian weighted edges connected to 3 nearest neighbors
  - Below are the components of the eigenvector corresponding to the second smallest eigenvalue



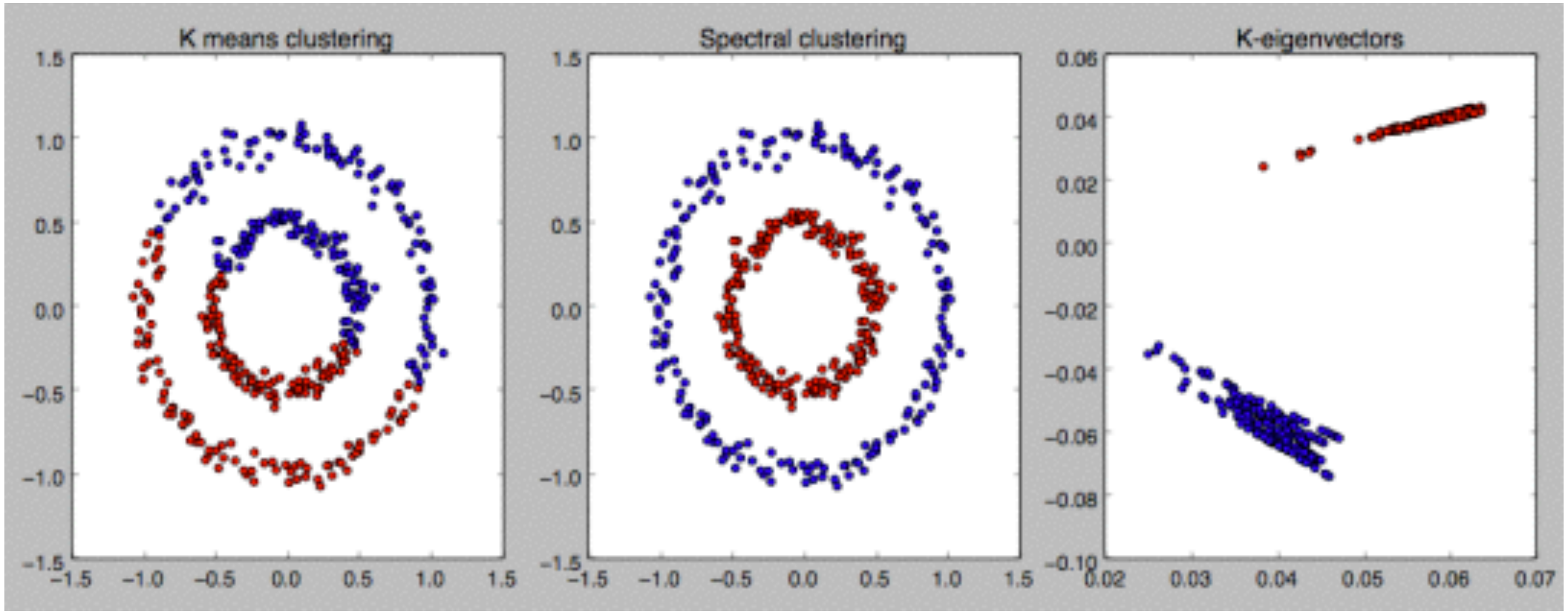
Spectral embedding examples

- Image segmentation: multiple eigenvalues reshaped into an image



http://ttic.uchicago.edu/~mmaire/papers/pdf/amfm_tpami2011.pdf

- Toy dataset



Summary

- Dimensionality reduction for visualization or preprocessing
- Linear methods
  - PCA — linear projections of data
    - ➡ solve $(\mathbf{X}^T\mathbf{X})x = \lambda x$ — eigenvectors of covariance matrix
- Non-linear methods
  - kernel PCA — linear projections in kernel space
    - ➡ solve $\mathbf{K}x = \lambda x$ — eigenvectors of the kernel matrix
  - Spectral embedding — graph partitions
    - ➡ solve $(\mathbf{D} - \mathbf{W})x = \lambda \mathbf{D}x$ — eigenvectors of the Graph laplacian
- There are several methods that we didn't discuss
  - ISOMAP, locally linear embedding, tSNE, etc

Slides credit

- Some of the slides are based on CIML book by Hal Daume III
- Linear and kernel PCA notes: http://pca.narod.ru/scholkopf_kernel.pdf
- The example for kernel PCA is from: http://sebastianraschka.com/Articles/2014_kernel_pca.html