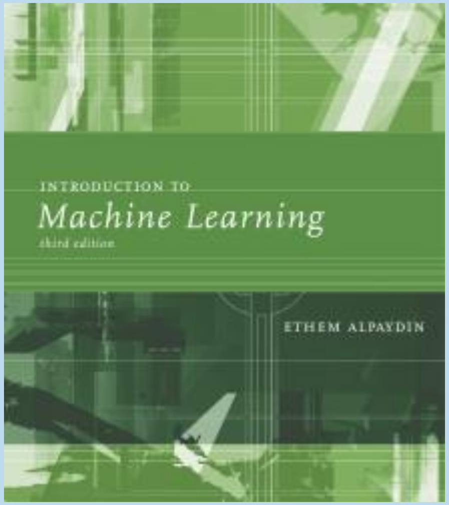


Lecture Slides for

INTRODUCTION TO MACHINE LEARNING

3RD EDITION

ETHEM ALPAYDIN

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml3e

CHAPTER 6:

DIMENSIONALITY REDUCTION

Why Reduce Dimensionality?

- Reduces time complexity: Less computation
- Reduces space complexity: Fewer parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable; simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

Feature Selection vs Extraction

- □ Feature selection: Choosing $k < d$ important features, ignoring the remaining $d - k$

  Subset selection algorithms

- □ Feature extraction: Project the original x_i , $i = 1,...,d$ dimensions to new $k < d$ dimensions, z_j , $j = 1,...,k$

  Principal components analysis (PCA), linear discriminant analysis (LDA), factor analysis (FA)

- □ Feature construction: Create new features based on old features: f=(…) with f usually being a non-linear function→support vector machines,…

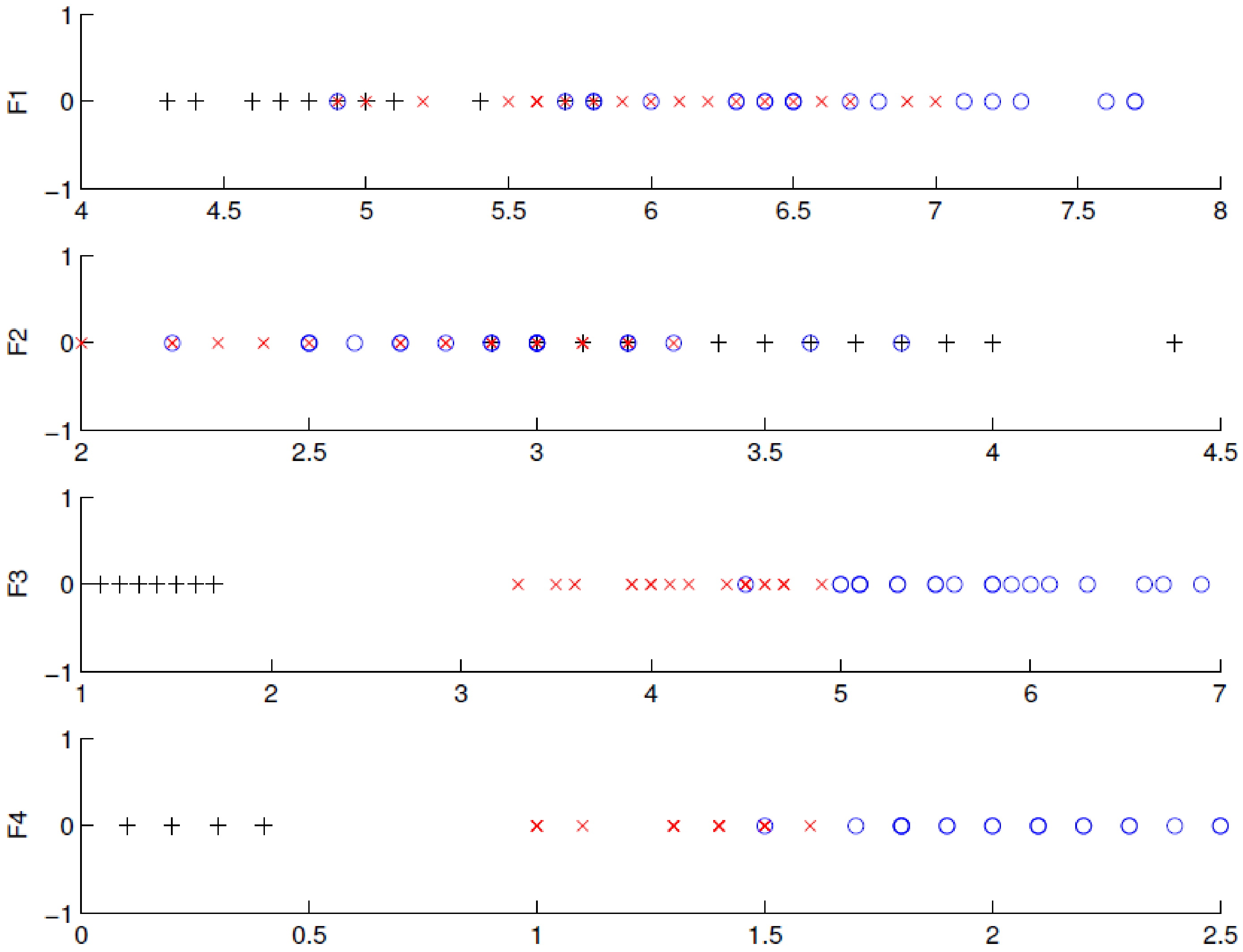
Subset Selection

- There are 2^d subsets of d features
- Forward search: Add the best feature at each step
- F is a feature set of input dimensions; $E(F)$ denotes the error incurred on the validation sample when only the inputs in F are used.
    - Set of features F initially Ø.
    - At each iteration, find the best new feature

      $j = \text{argmin}_i\, E(F \cup x_i)$

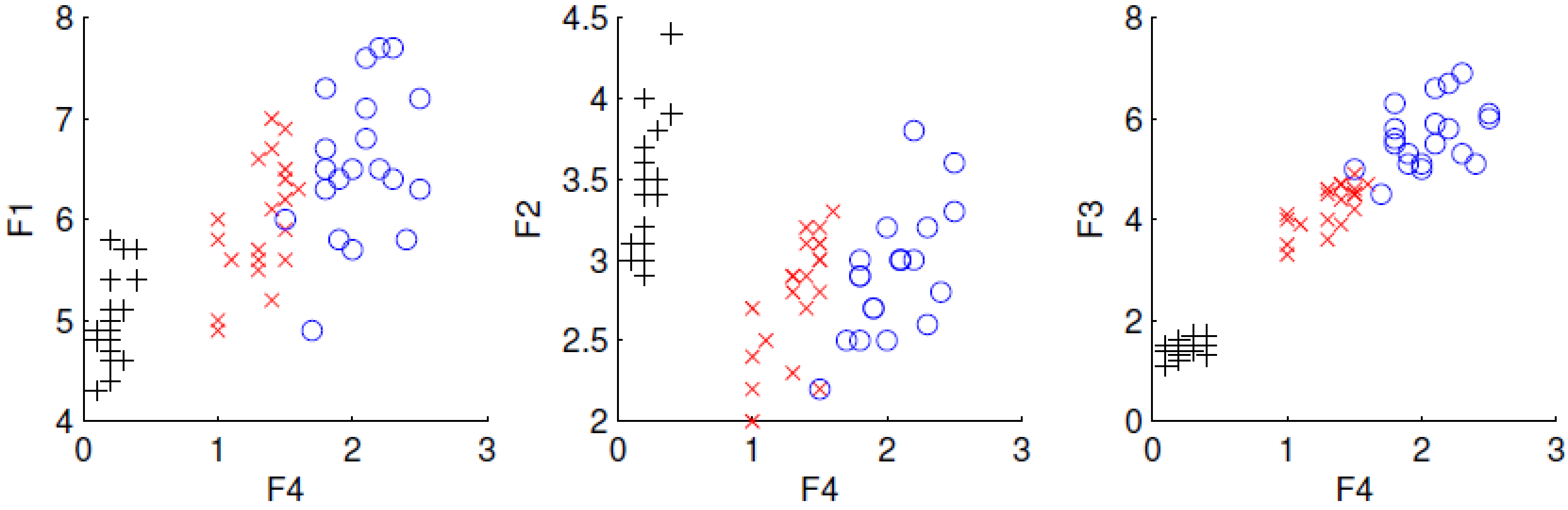
      Add x_j to F if $E(F \cup x_j) < E(F)$
    - This algorithm is also known as the wrapper approach, where the process of feature extraction is thought to “wrap” around the learner it uses as a subroutine.

Iris data (4 inputs and 3 classes): Single feature



Chosen

Iris data: Add one more feature to F4



Chosen

Subset Selection

- Hill-climbing O(d^2) algorithm
- Backward search: Start with all features and remove one at a time, if possible.
- Floating search (Add k, remove l)
- In floating search methods, the number of added features and removed features can also change at each step.
- In sequential backward selection, we start with F containing all features and do a similar process except that we remove one attribute from F as opposed to adding to it, and we remove the one that causes the least error

$$j = \operatorname{argmin}_i E\,(F - x_i)$$

$$\text{remove } x_j \text{ from } F \text{ if } E(F - x_j) < E(F)$$

Principal Components Analysis

- Find a low-dimensional space such that when $\boldsymbol{x}$ is projected there, information loss is minimized.
- The projection of $\boldsymbol{x}$ on the direction of $\boldsymbol{w}$ is: $z = \boldsymbol{w}^T\boldsymbol{x}$
- Find $\boldsymbol{w}$ such that Var(z) is maximized

$$\begin{aligned}\mathrm{Var}(z) &= \mathrm{Var}(\boldsymbol{w}^T\boldsymbol{x}) = E[(\boldsymbol{w}^T\mathbf{x} - \boldsymbol{w}^T\boldsymbol{\mu})^2] \\ &= E[(\boldsymbol{w}^T\mathbf{x} - \boldsymbol{w}^T\boldsymbol{\mu})(\boldsymbol{w}^T\mathbf{x} - \boldsymbol{w}^T\boldsymbol{\mu})] \\ &= E[\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{w}] \quad \longleftarrow \text{Note: } \boldsymbol{A}^T\boldsymbol{B}=\boldsymbol{B}^T\boldsymbol{A} \\ &= \boldsymbol{w}^{\mathrm{T}} E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]\boldsymbol{w} = \boldsymbol{w}^T \sum \boldsymbol{w}\end{aligned}$$

where $\mathrm{Var}(\boldsymbol{x}) = E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T] = \sum$

- □ Maximize Var(*z*) subject to ||***w***||=1

$$\max_{\mathbf{w}_1} \mathbf{w}_1^T \Sigma \mathbf{w}_1 - \alpha\left(\mathbf{w}_1^T \mathbf{w}_1 - 1\right)$$

$\sum \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$ that is, $\boldsymbol{w}_1$ is an eigenvector of $\sum$

Choose the one with the largest eigenvalue for Var(z) to be max; Because we want to maximize $\mathbf{w}_1^T \Sigma \mathbf{w}_1 = \alpha \mathbf{w}_1^T \mathbf{w}_1 = \alpha$

- □ Second principal component: Max Var(z_2), s.t., ||$\boldsymbol{w}_2$||=1 and orthogonal to $\boldsymbol{w}_1$

$$\max_{\mathbf{w}_2} \mathbf{w}_2^T \Sigma \mathbf{w}_2 - \alpha\left(\mathbf{w}_2^T \mathbf{w}_2 - 1\right) - \beta\left(\mathbf{w}_2^T \mathbf{w}_1 - 0\right)$$

$$\Rightarrow 2\Sigma \mathbf{w}_2 - 2\alpha \mathbf{w}_2 - \beta \mathbf{w}_1 = 0 \Rightarrow 2\mathbf{w}_1^T \Sigma \mathbf{w}_2 - 2\alpha \mathbf{w}_1^T \mathbf{w}_2 - \beta \mathbf{w}_1^T \mathbf{w}_1 = 0$$

$$\mathbf{w}_1^T \mathbf{w}_2 = 0, \mathbf{w}_1^T \Sigma \mathbf{w}_2 = \mathbf{w}_2^T \Sigma \mathbf{w}_1 = \lambda_1 \mathbf{w}_2^T \mathbf{w}_1 = 0 \Rightarrow \beta = 0 \Rightarrow$$

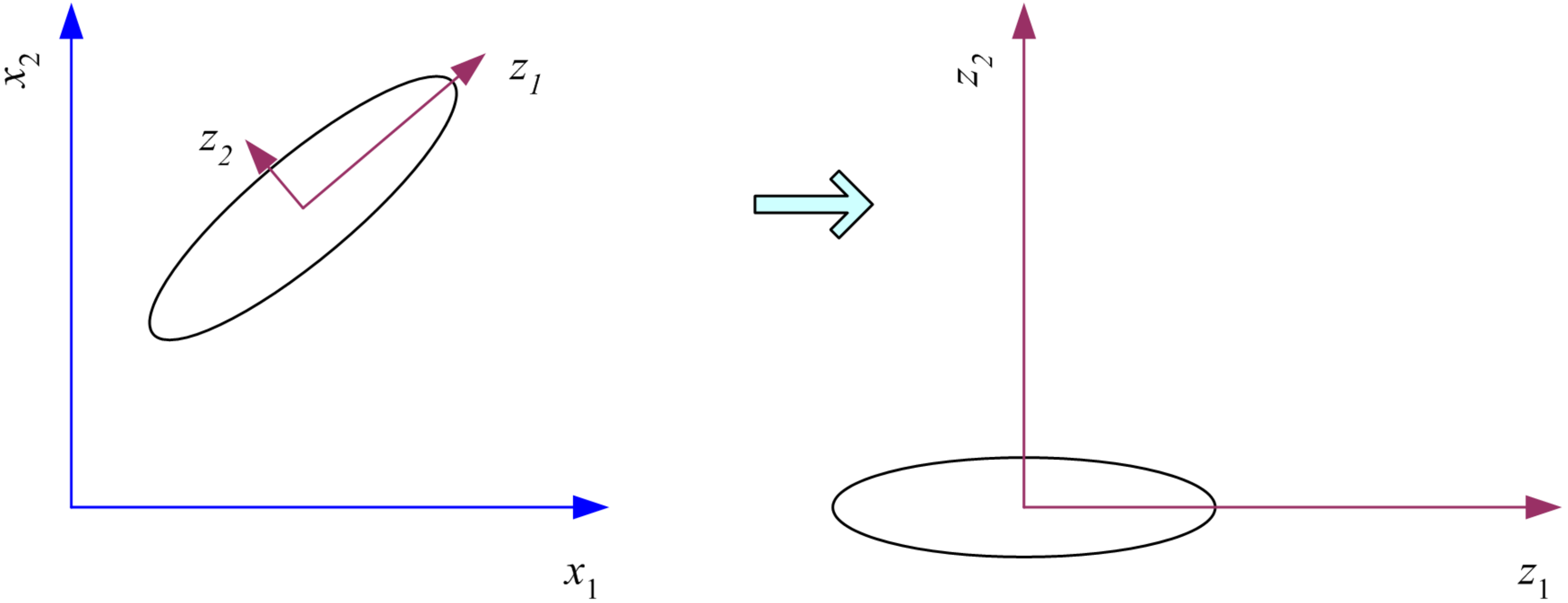
$\sum \boldsymbol{w}_2 = \alpha\, \boldsymbol{w}_2$ that is, $\boldsymbol{w}_2$ is another eigenvector of $\sum$ and so on.

What PCA does

$$\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{m})$$

where the columns of **W** are the eigenvectors of $\sum$ and ***m*** is sample mean

Centers the data at the origin and rotates the axes.



Another Derivation

- Find a matrix **W** such that when we have $\boldsymbol{z} = \mathbf{W}^T\boldsymbol{x}$ we will get Cov$(\boldsymbol{z}) = \mathbf{D}$ where **D** is any diagonal matrix;
- We form a ($d \times d$) matrix **C** whose *i*th column is the normalized eigenvector $\boldsymbol{c}_i$ of **S**, then $\mathbf{C}^T\mathbf{C} = \mathbf{I}$ and
- $\mathbf{S} = \mathbf{S}\mathbf{C}\mathbf{C}^T = \mathbf{S}(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_\mathrm{d})\mathbf{C}^T = (\mathbf{S}\boldsymbol{c}_1, \mathbf{S}\boldsymbol{c}_2, \ldots, \mathbf{S}\boldsymbol{c}_\mathrm{d})\,\mathbf{C}^T = (\lambda_1\boldsymbol{c}_1, \lambda_2\boldsymbol{c}_2, \ldots, \lambda_\mathrm{d}\boldsymbol{c}_\mathrm{d})\mathbf{C}^T = \lambda_1\boldsymbol{c}_1\boldsymbol{c}_1^T + \cdots + \lambda_d\boldsymbol{c}_d\boldsymbol{c}_\mathrm{d}^T = \mathbf{C}\mathbf{D}\mathbf{C}^T$ where **D** is a diagonal matrix whose diagonal elements are the eigenvalues, $\lambda_1, \ldots, \lambda_\mathrm{d}$.
- This is called the *spectral decomposition* of **S**.
- $\mathbf{C}^T\mathbf{S}\mathbf{C} = \mathbf{D}$; Cov$(\boldsymbol{z}) = \mathbf{W}^T\mathbf{S}\mathbf{W}$ then $\mathbf{W} = \mathbf{C}$.

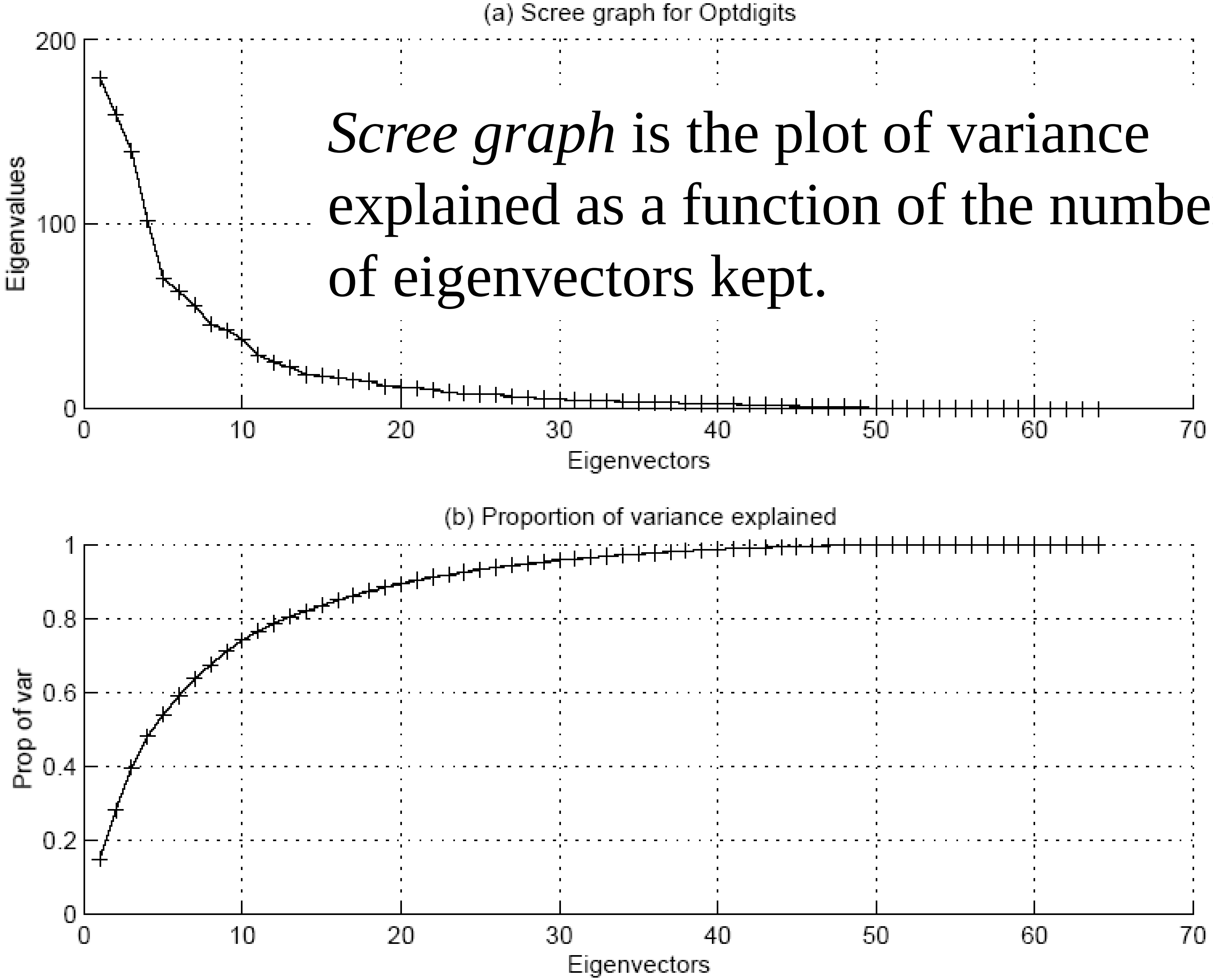
How to choose k ?

- □ Proportion of Variance (PoV) explained

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

when λ_i are sorted in descending order

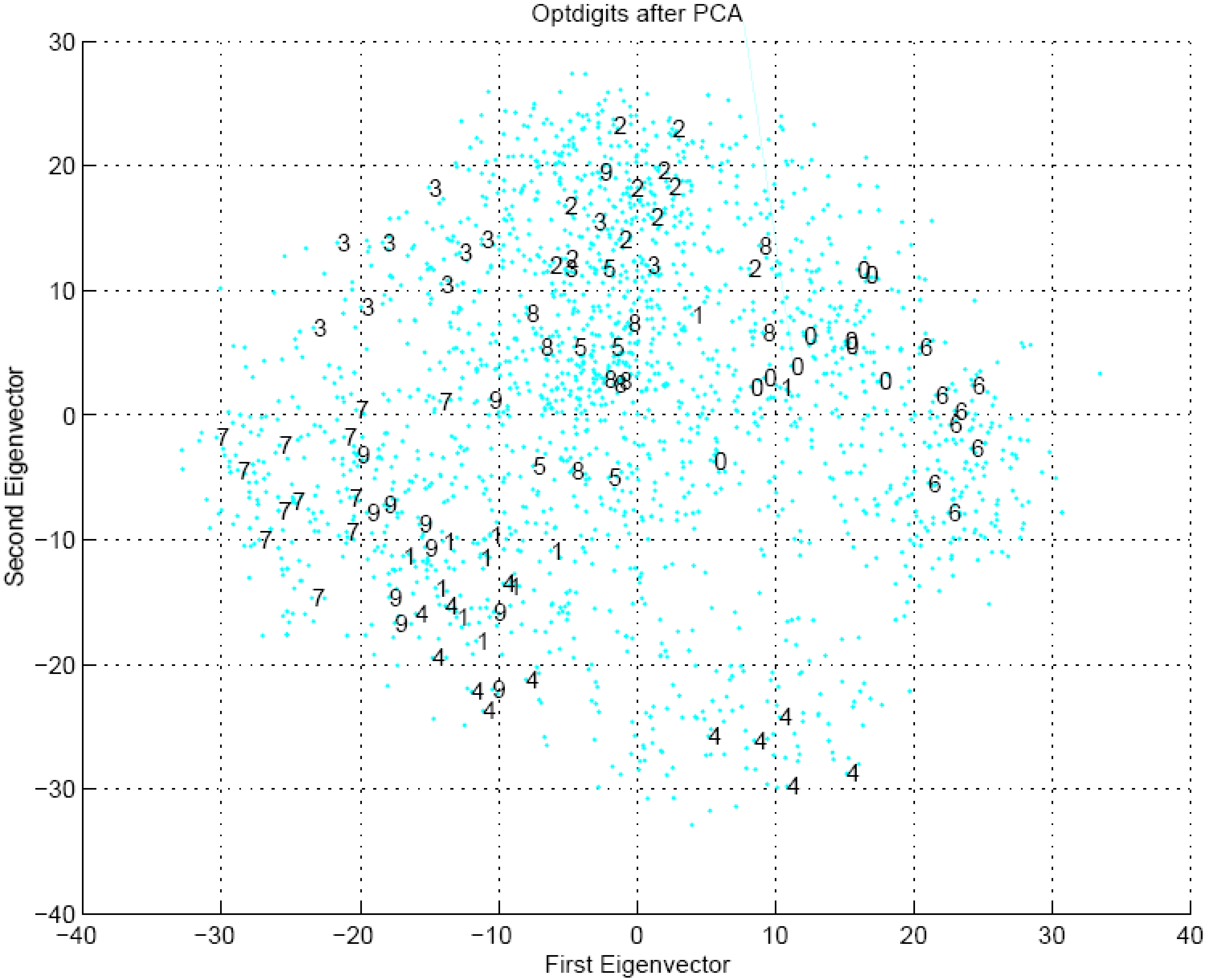
- □ Typically, stop at PoV>0.9
- □ Scree graph plots of PoV vs *k*, stop at “elbow”



(a) Scree graph for Optdigits

(b) Proportion of variance explained

Scree graph is the plot of variance explained as a function of the number of eigenvectors kept.


Optdigits after PCA
Second Eigenvector
First Eigenvector
30
20
10
0
−10
−20
−30
−40
−40
−30
−20
−10
0
10
20
30
40

PCA numerical example

1. Consider a set of 2D points $P_i = (x_i, y_i)$
2. Subtract the mean from each of the data dimensions. All the x values have $\bar{x}$ subtracted and y values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero. Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.

original data

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

zero-mean data

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

and calculate the covariance matrix

$$C = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

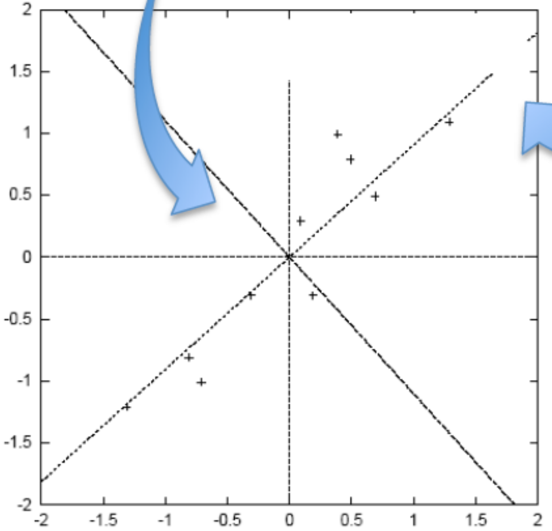
3. Calculate the eigenvalues and eigenvectors of the covariance matrix

$$\boxed{\text{Det } (C - \lambda I) = 0}$$

$$\boxed{\text{Det } (C - \lambda I)x = 0}$$

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \left(\begin{array}{c:c} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{array}\right)$$



Eigenvectors are plotted as diagonal dotted lines on the plot.

- they are perpendicular to each other.
- one of the eigenvectors is like a line of best fit.
- the second eigenvector gives the less important, pattern in the data: all the points follow the main line, but are off by some amount.

4. To reduce dimensionality it must be formed a *feature vector.*
The eigenvector with the *highest* eigenvalue is the *principle component* of the data set.
Once eigenvectors are found from the covariance matrix, they must be ordered by eigenvalue, from the highest to the lowest. This gives the components in order of significance. The components of lesser significance can be ignored. If the eigenvalues are small, only little is lost.

Feature Vector = (e_1 e_2 e_3 … e_n)
we can either form a feature vector with both of the eigenvectors:

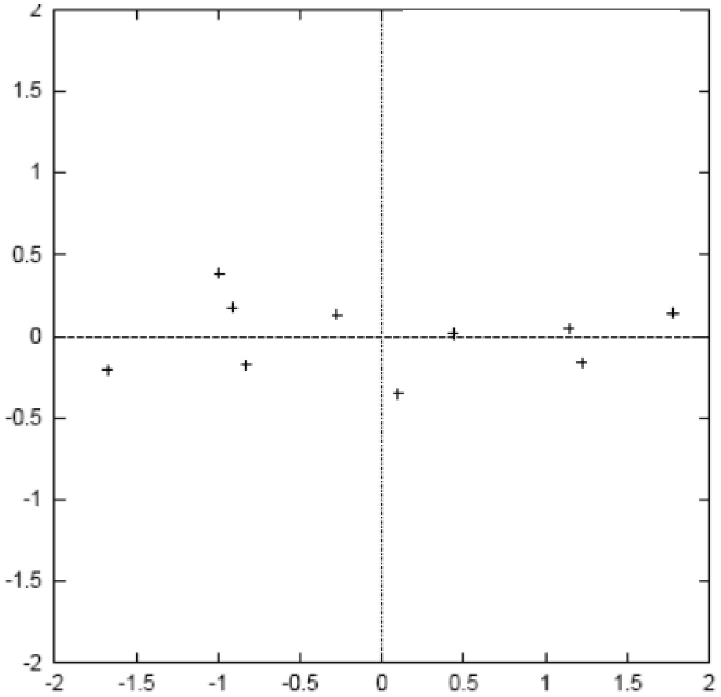
$$\left(\begin{array}{c:c} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{array}\right)$$

or, choose to leave out the less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

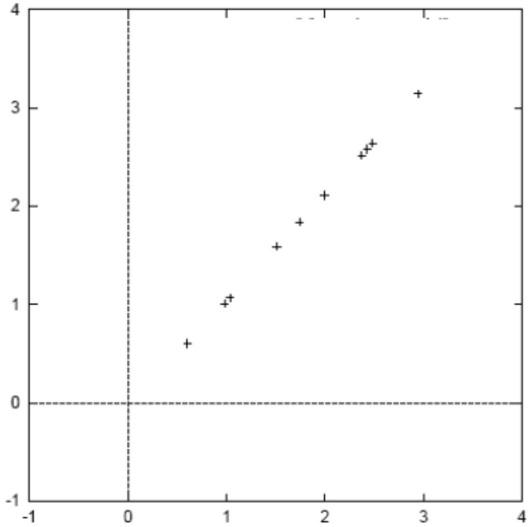
5. Considering both eigenvectors, the new data is obtained as:

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

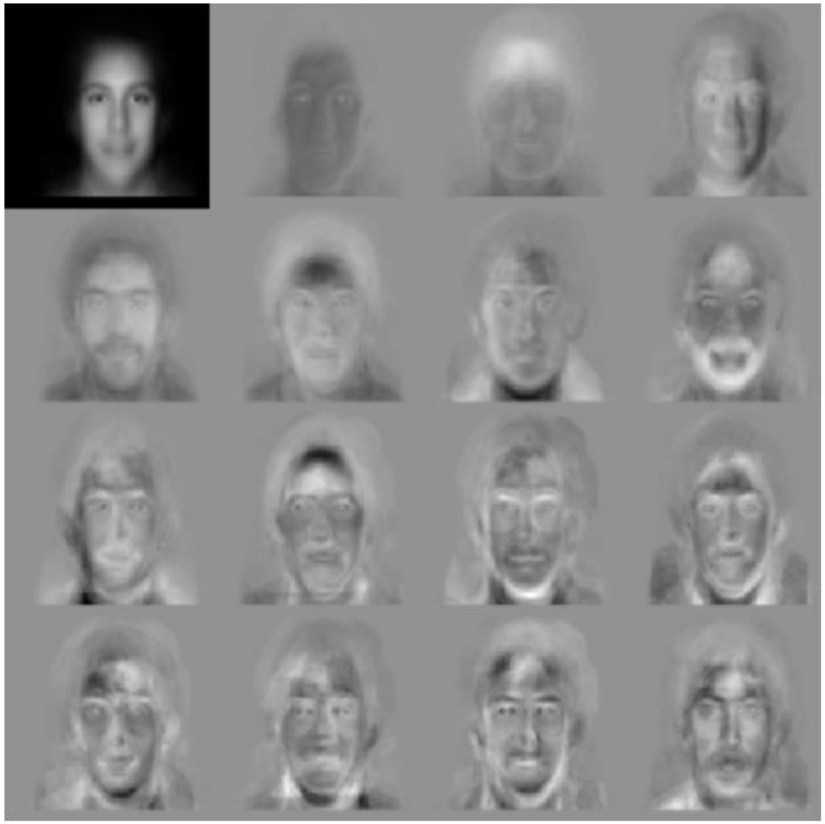


6. If we reduce the dimensionality, when reconstructing the data those dimensions we chose to discard are lost. If the y component is discarded and only the x dimension is retained…

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056



- Given a training set of faces represented as N^2x1 vectors, PCA extracts the eigenvectors of the matrix A built from this set of vectors. Each eigenvector has the same dimensionality as the original images, and can be regarded as an image.
- They are referred to as eigenfaces. Eigenfaces can be considered a set of "standardized face ingredients", derived from statistical analysis of many pictures of faces. They are the directions in which the images differ from the mean image.

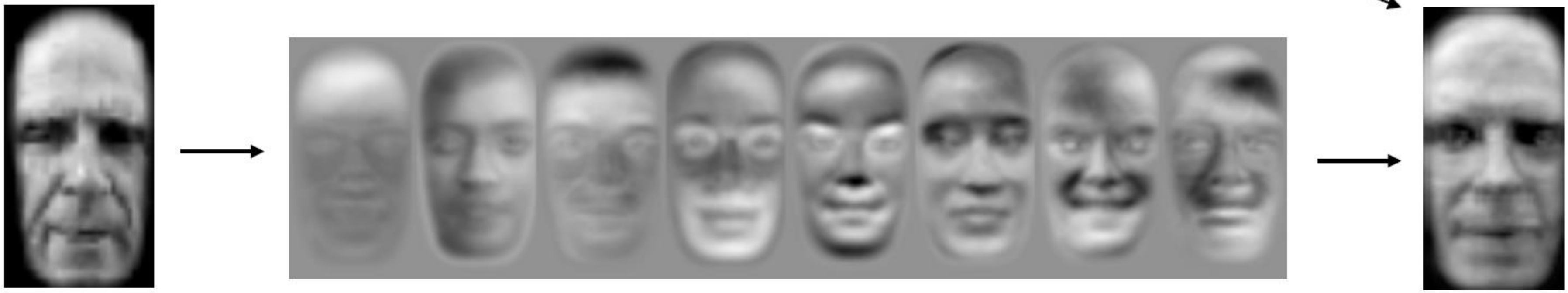


- The eigenvectors (eigenfaces) with largest associated eigenvalue are kept.

- These eigenfaces can now be used to represent both existing and new faces by projecting a new (mean-subtracted) image on the eigenfaces and recording how that new face differs from the mean face.

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot \mathbf{v}_1}_{a_1},\ \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot \mathbf{v}_2}_{a_2}, \ldots,\ \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot \mathbf{v}_\mathbf{K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_K\mathbf{v}_\mathbf{K}$$



Computation of the covariance matrix is simplified (suppose 300 images of 100x100 pixels, that yelds a 10000x10000 covariance matrix. Eigenvalues are instead extracted from 300 x 300 covariance matrix).

In practical applications, most faces can be identified using a projection on between 100 and 150 eigenfaces, so that most of the eigenvectors can be discarded.

Feature Embedding (FE)

- When $\boldsymbol{X}$ is the $N \times d$ data matrix,

  $\boldsymbol{X}^T\boldsymbol{X}$ is the $d \times d$ matrix (cov. of features, if mean-centered)

  $\boldsymbol{X}\boldsymbol{X}^T$ is the $N \times N$ matrix (pairwise similarities of instances)
- PCA uses the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ which are d-dim and can be used for projection
- FE uses the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$ which are N-dim and which give directly the coordinates after projection
- Sometimes, we can define pairwise similarities (or distances) between instances, then we can use feature embedding without needing to represent instances as vectors.
- $(\mathbf{X}^T\mathbf{X})\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i \rightarrow \mathbf{X}\,(\mathbf{X}^T\mathbf{X})\boldsymbol{w}_i = \mathbf{X}\,\lambda_i \boldsymbol{w}_i \rightarrow (\mathbf{X}\mathbf{X}^T)\mathbf{X}\boldsymbol{w}_i = \lambda_i \mathbf{X}\boldsymbol{w}_i$
- $\mathbf{X}\boldsymbol{w}_i$ must be the eigenvectors of $\mathbf{X}\mathbf{X}^T$ with the same eigenvalues

Factor Analysis

- Find a small number of factors **z**, which when combined generate ***x***:

$$x_i - \mu_i = v_{i1}z_1 + v_{i2}z_2 + \ldots + v_{ik}z_k + \varepsilon_i$$

where z_j, $j = 1,\ldots,k$ are the latent factors with

$$E[\ z_j\ ]=0,\ \mathrm{Var}(z_j)=1,\ \mathrm{Cov}(z_i\ ,\ z_j)=0,\ i \neq j\ ,$$

ε_i are the noise sources

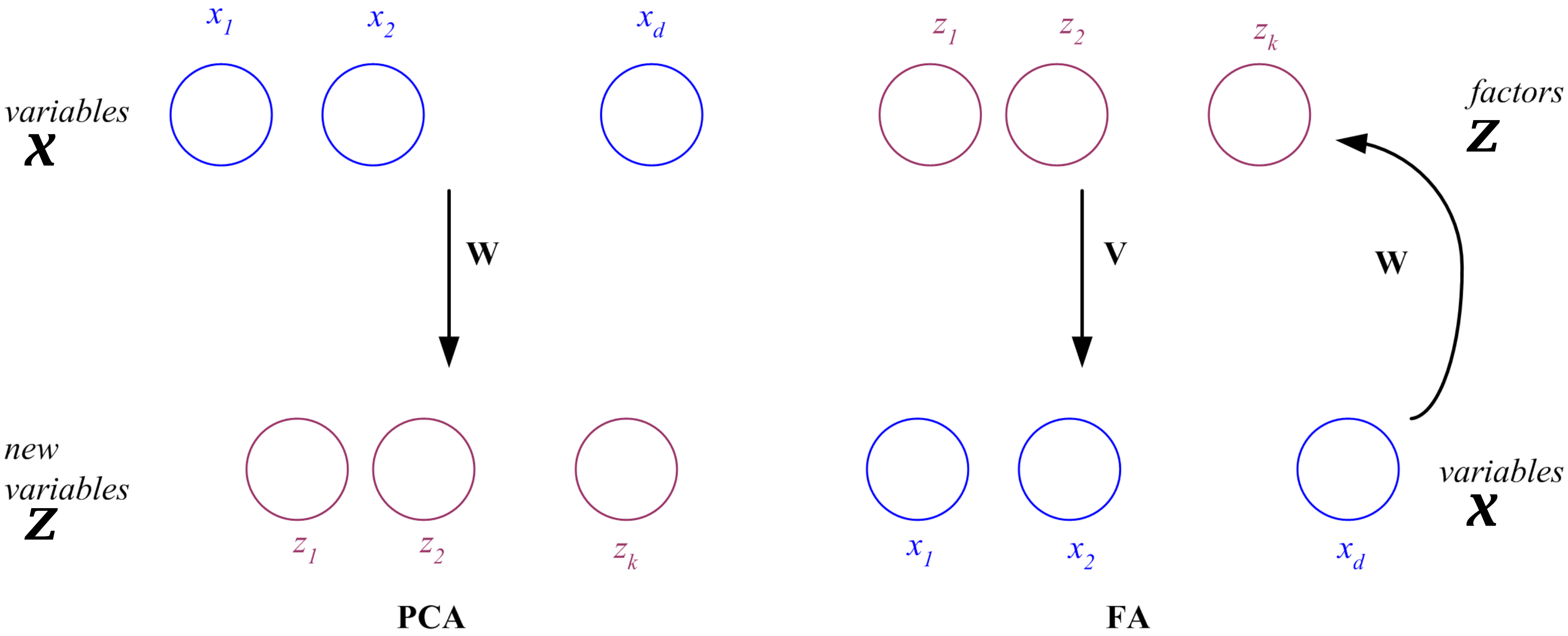
$E[\ \varepsilon_i\ ]= 0$, $\mathrm{Var}(\ \varepsilon_i\ )= \psi_i$, $\mathrm{Cov}(\varepsilon_i\ ,\ \varepsilon_j) =0,\ i \neq j$,

$\mathrm{Cov}(\varepsilon_i\ ,\ z_j) =0$,

and v_{ij} are the factor loadings

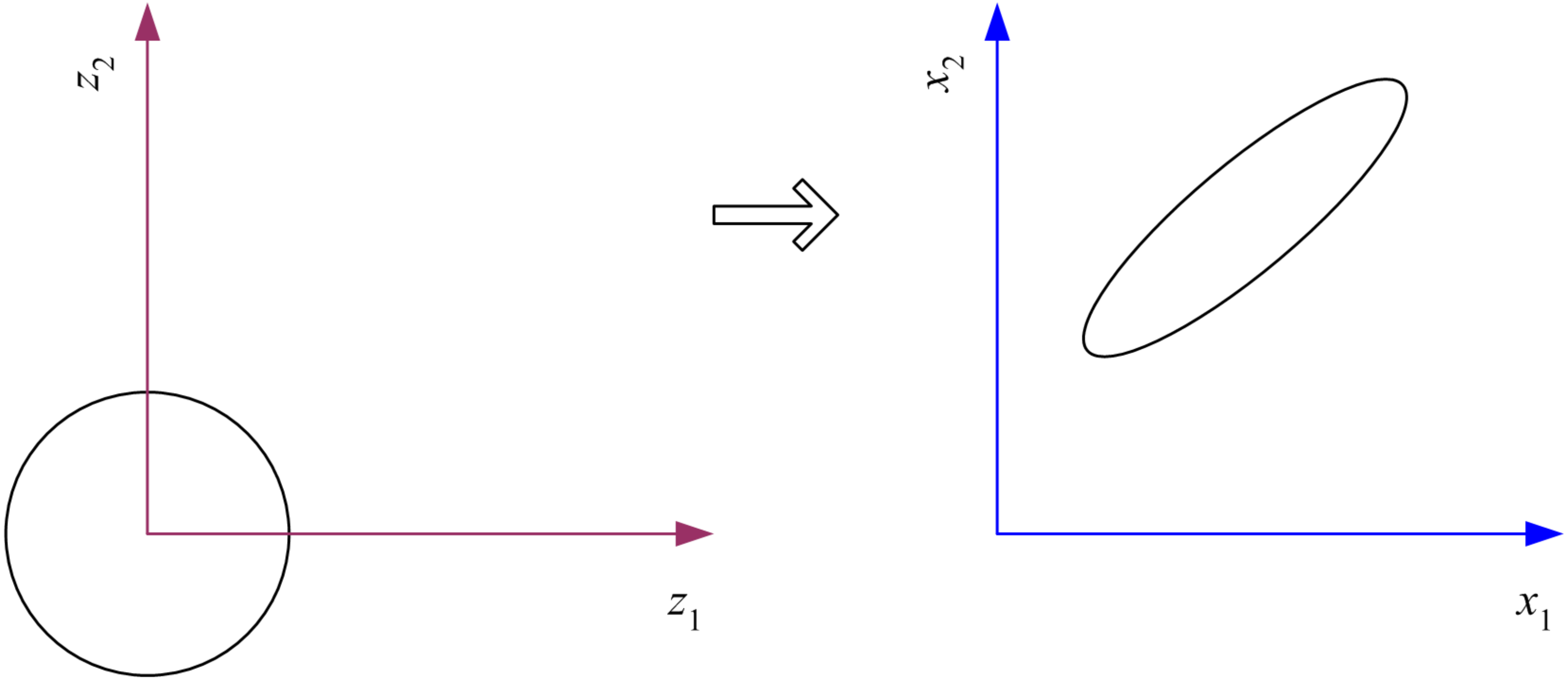
PCA vs FA

- □ PCA   From $\boldsymbol{x}$ to $\boldsymbol{z}$ $\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{\mu})$
- □ FA   From $\boldsymbol{z}$ to $\boldsymbol{x}$ $\boldsymbol{x} - \boldsymbol{\mu} = \mathbf{V}\mathbf{z} + \boldsymbol{\varepsilon}$



Factor Analysis

- In FA, factors z_j are stretched, rotated and translated to generate $\boldsymbol{x}$



$$\begin{aligned}\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{x}) &= \mathrm{Cov}(\mathbf{Vz} + \boldsymbol{\epsilon}) \\ &= \mathrm{Cov}(\mathbf{Vz}) + \mathrm{Cov}(\boldsymbol{\epsilon}) \\ &= \mathbf{V}\mathrm{Cov}(\mathbf{z})\mathbf{V}^T + \boldsymbol{\Psi} \\ &= \mathbf{VV}^T + \boldsymbol{\Psi}\end{aligned}$$

a diagonal matrix

The estimator of $\boldsymbol{\Sigma} = \mathbf{S} = \mathbf{VV}^T + \boldsymbol{\Psi}$

V= Covariances or factor loadings,

Ψ= Variances

V is not Unique.

$$\mathbf{S} = (\mathbf{VT})(\mathbf{VT})^T = \mathbf{VTT}^T\mathbf{V}^T = \mathbf{VIV}^T = \mathbf{VV}^T$$

□ If **T** is an orthogonal matrix, the distance to the origin does not change. If **z** = **Tx**, then

$$\boldsymbol{z}^T\boldsymbol{z} = (\mathbf{T}\boldsymbol{x})^T(\mathbf{T}\boldsymbol{x}) = \boldsymbol{x}^T\mathbf{T}^T\mathbf{T}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x}$$

Dimensionality Reduction

- Finding the factor scores, z_j, from x_i . Finding the loadings w_{ji} such that

$$z_j = \sum_{i=1}^{d} w_{ji} x_i + \epsilon_j, j = 1, \ldots, k$$

$$\mathbf{z}^t = \mathbf{W}^T \boldsymbol{x}^t + \boldsymbol{\epsilon}, \forall t = 1, \ldots, N$$

$$(\mathbf{z}^t)^T = (\boldsymbol{x}^t)^T \mathbf{W} + \boldsymbol{\epsilon}^T, \forall t = 1, \ldots, N$$

$$\mathbf{Z} = \mathbf{X}\mathbf{W} + \boldsymbol{\Xi} \qquad \mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Z}$$

$$\mathbf{W} = (N-1)(\mathbf{X}^T\mathbf{X})^{-1}\frac{\mathbf{X}^T\mathbf{Z}}{N-1} = \left(\frac{\mathbf{X}^T\mathbf{X}}{N-1}\right)^{-1}\frac{\mathbf{X}^T\mathbf{Z}}{N-1} = \mathbf{S}^{-1}\mathbf{V}$$

$$\mathbf{Z} = \mathbf{X}\mathbf{W} = \mathbf{X}\mathbf{S}^{-1}\mathbf{V}$$

- There are two uses of factor analysis:
- It can be used for knowledge extraction when we find the loadings and try to express the variables using fewer factors.
- It can also be used for dimensionality reduction when $k < d$.
- For dimensionality reduction, FA offers no advantage over PCA except the interpretability of factors allowing the identification of common causes, a simple explanation, and knowledge extraction.

Singular Value Decomposition and Matrix Factorization

- Singular value decomposition: $\mathbf{X} = \mathbf{VAW}^T$

  $\mathbf{V}$ is $N \times N$ and contains the eigenvectors of $\mathbf{XX}^T$

  $\mathbf{W}$ is $d \times d$ and contains the eigenvectors of $\mathbf{X}^T\mathbf{X}$

  and $\mathbf{A}$ is $N \times d$ and contains singular values on its first $k = min(N, d)$ diagonal.

  $$\mathbf{XX}^T=(\mathbf{VAW}^T)(\mathbf{VAW}^T)^T=\mathbf{VAW}^T\mathbf{WA}^T\mathbf{V}^T=\mathbf{VEV}^T$$

  $$\mathbf{X}^T\mathbf{X}=(\mathbf{VAW}^T)^T(\mathbf{VAW}^T)=\mathbf{WA}^T\mathbf{V}^T\mathbf{VAW}^T=\mathbf{WDW}^T$$

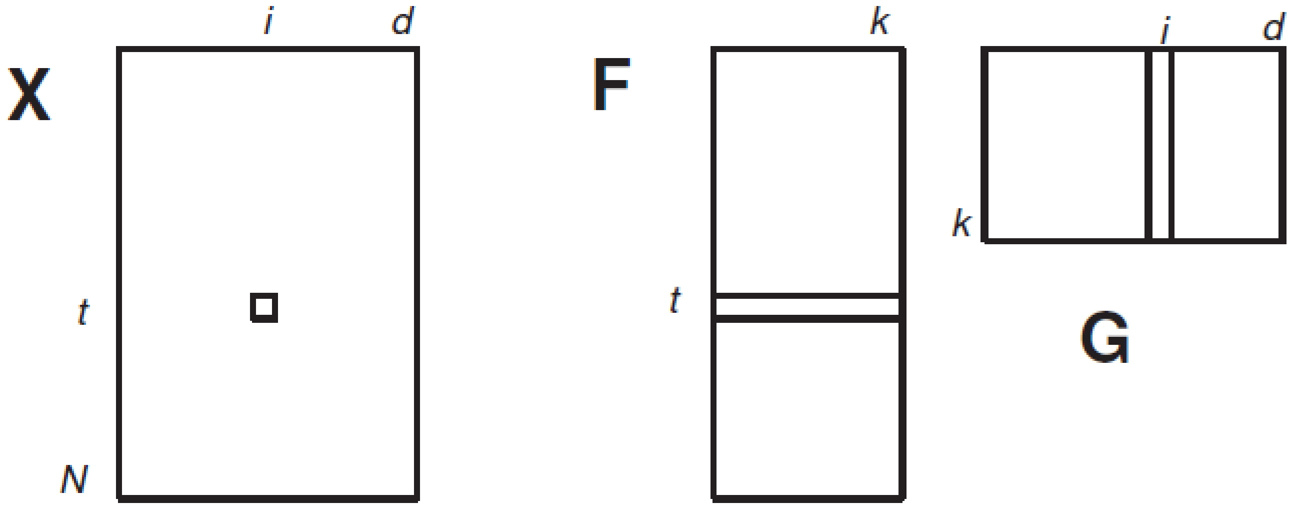
  Where $\mathbf{E}=\mathbf{AA}^T$, $\mathbf{D}=\mathbf{A}^T\mathbf{A}$. They are of different sizes but are both square and contain a_i^2, $i = 1,\ldots,k$ on their diagonal and zero elsewhere.

- $\mathbf{X}=\boldsymbol{u}_1 a_1 \boldsymbol{v}_1^T+\ldots+\boldsymbol{u}_k a_k \boldsymbol{v}_k^T$ where k is the rank of $\mathbf{X}$.

Matrix Factorization

- Matrix factorization: **X=FG**

  F is $N \times k$ and **G** is $k \times d$



$$\mathbf{X}_{ti} = \mathbf{F}_t^T \mathbf{G}_i = \sum_{j=1}^{k} \mathbf{F}_{tj} \mathbf{G}_{ji}$$

Latent semantic indexing

- **G** defines factors in terms of the original attributes and **F** defines data instances in terms of these factors.
- Exp: **X** is the consumer data. We have *N* customers and we sell *d* different products. $\mathbf{X}_{ti}$ corresponds to the amount of product *i* customer *N* has purchased.
- Purchases depend on a number of factors, for example, household size and composition, income level, taste, and so on.

Multidimensional Scaling

- Given pairwise distances between N points,

  $d_{ij}, i, j = 1, \ldots, N$

  place on a low-dim map s.t. distances are preserved (by feature embedding)

- $\boldsymbol{z} = \boldsymbol{g}\,(\mathbf{x} \mid \boldsymbol{\theta}\,)$

where $\boldsymbol{z} \in R^k$, $\boldsymbol{x} \in R^d$, and $\boldsymbol{g}\,(\boldsymbol{x} \mid \boldsymbol{\theta}\,)$ is the mapping function from d to k dimensions defined up to a set of parameters $\boldsymbol{\theta}$.

- Classical MDS - linear transformation $\boldsymbol{z} = \boldsymbol{g}\,(\boldsymbol{x} \mid \mathbf{W}\,) = \mathbf{W}^T\boldsymbol{x}$

- Find $\boldsymbol{\theta}$ that min Sammon stress (normalized error in mapping) and is defined as:

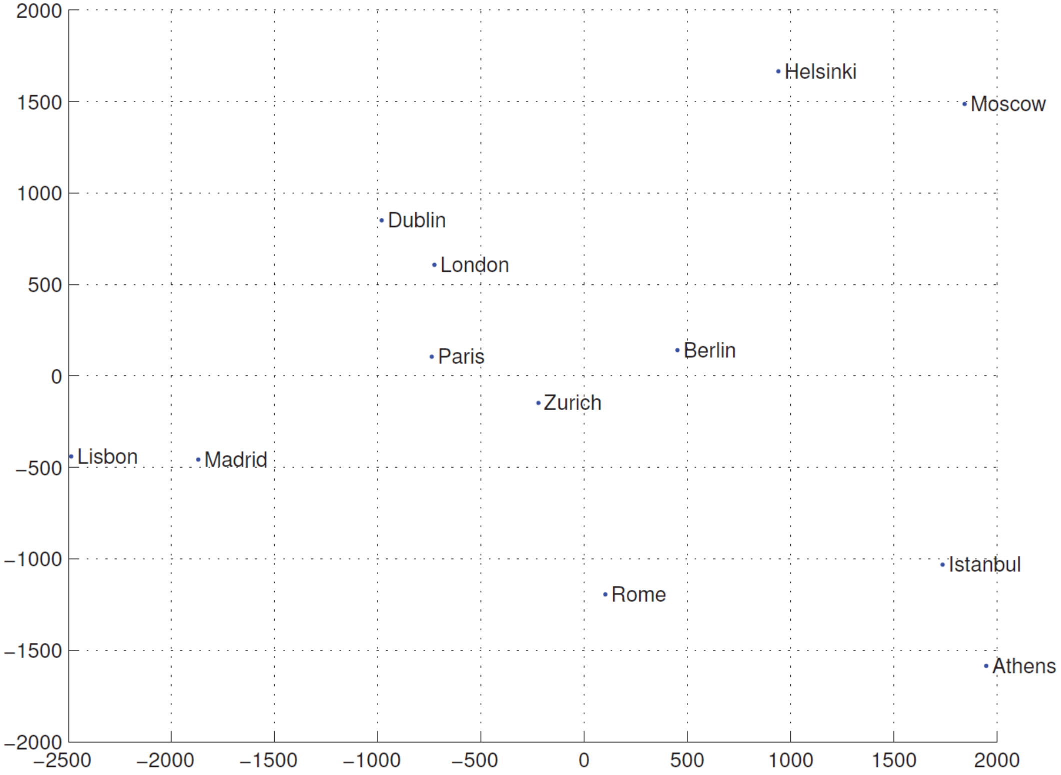
$$E\left(\boldsymbol{\theta}|\mathrm{X}\right)=\sum_{r,s}\frac{\left(\left\|\mathbf{z}^{r}-\mathbf{z}^{s}\right\|-\left\|\mathbf{x}^{r}-\mathbf{x}^{s}\right\|\right)^{2}}{\left\|\mathbf{x}^{r}-\mathbf{x}^{s}\right\|^{2}}$$

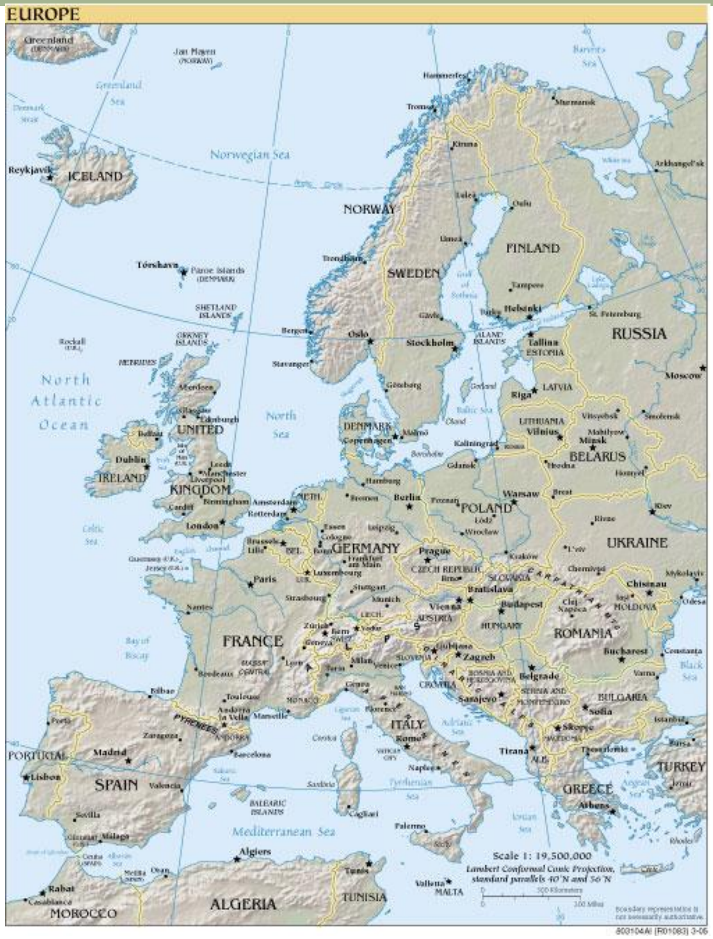
$$=\sum_{r,s}\frac{\left(\left\|\mathbf{g}\left(\mathbf{x}^{r}|\boldsymbol{\theta}\right)-\mathbf{g}\left(\mathbf{x}^{s}|\boldsymbol{\theta}\right)\right\|-\left\|\mathbf{x}^{r}-\mathbf{x}^{s}\right\|\right)^{2}}{\left\|\mathbf{x}^{r}-\mathbf{x}^{s}\right\|^{2}}$$

For two points ***r*** and ***s***

- Any regression method can be used to estimate ***θ*** to minimize the stress on the training data **X**.

Map of Europe by MDS



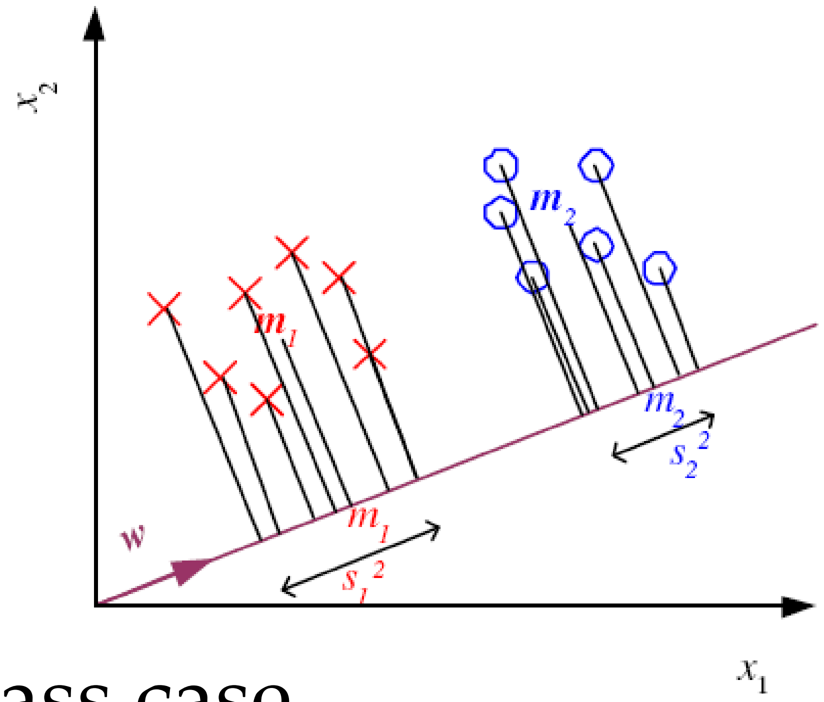


Map of Europe drawn by MDS. Pairwise road travel distances bw. these cities are given as input, and MDS places them in two dims. such that these distances are preserved as well as possible.

Map from CIA – The World Factbook: http://www.cia.gov/

Linear Discriminant Analysis

- Find a low-dimensional space such that when ***x*** is projected, classes are well-separated.
- Find ***w*** that maximizes



$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

2 class case

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t} \qquad s_1^2 = \sum_t \left(\mathbf{w}^T \mathbf{x}^t - m_1\right)^2 r^t$$

- Between-class scatter:

$$\begin{aligned}(m_1 - m_2)^2 &= \left(\mathbf{w}^T\mathbf{m}_1 - \mathbf{w}^T\mathbf{m}_2\right)^2 \\ &= \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\mathbf{w} \\ &= \mathbf{w}^T\mathbf{S}_B\mathbf{w} \quad \text{where } \mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T\end{aligned}$$

- Within-class scatter:

$$\begin{aligned}s_1^2 &= \sum_t \left(\mathbf{w}^T\mathbf{x}^t - m_1\right)^2 r^t \\ &= \sum_t \mathbf{w}^T\left(\mathbf{x}^t - \mathbf{m}_1\right)\left(\mathbf{x}^t - \mathbf{m}_1\right)^T\mathbf{w}r^t = \mathbf{w}^T\mathbf{S}_1\mathbf{w}\end{aligned}$$

$$\text{where } \mathbf{S}_1 = \sum_t \left(\mathbf{x}^t - \mathbf{m}_1\right)\left(\mathbf{x}^t - \mathbf{m}_1\right)^T r^t$$

$$s_1^2 + s_2^2 = \mathbf{w}^T\mathbf{S}_W\mathbf{w} \quad \text{where } \mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

Fisher’s Linear Discriminant

- □ Find **w** that max

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \frac{\left|\mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)\right|^2}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- □ LDA soln:

$$\mathbf{w} = c \cdot \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

- □ Parametric soln:

$$\mathbf{w} = \mathbf{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

$$\text{when } p(\mathbf{x}|C_i) \sim N(\boldsymbol{\mu}_i, \mathbf{\Sigma})$$

K > 2 Classes

- Within-class scatter:

$$\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i \qquad \mathbf{S}_i = \sum_t r_i^t \left(\mathbf{x}^t - \mathbf{m}_i\right)\left(\mathbf{x}^t - \mathbf{m}_i\right)^T$$

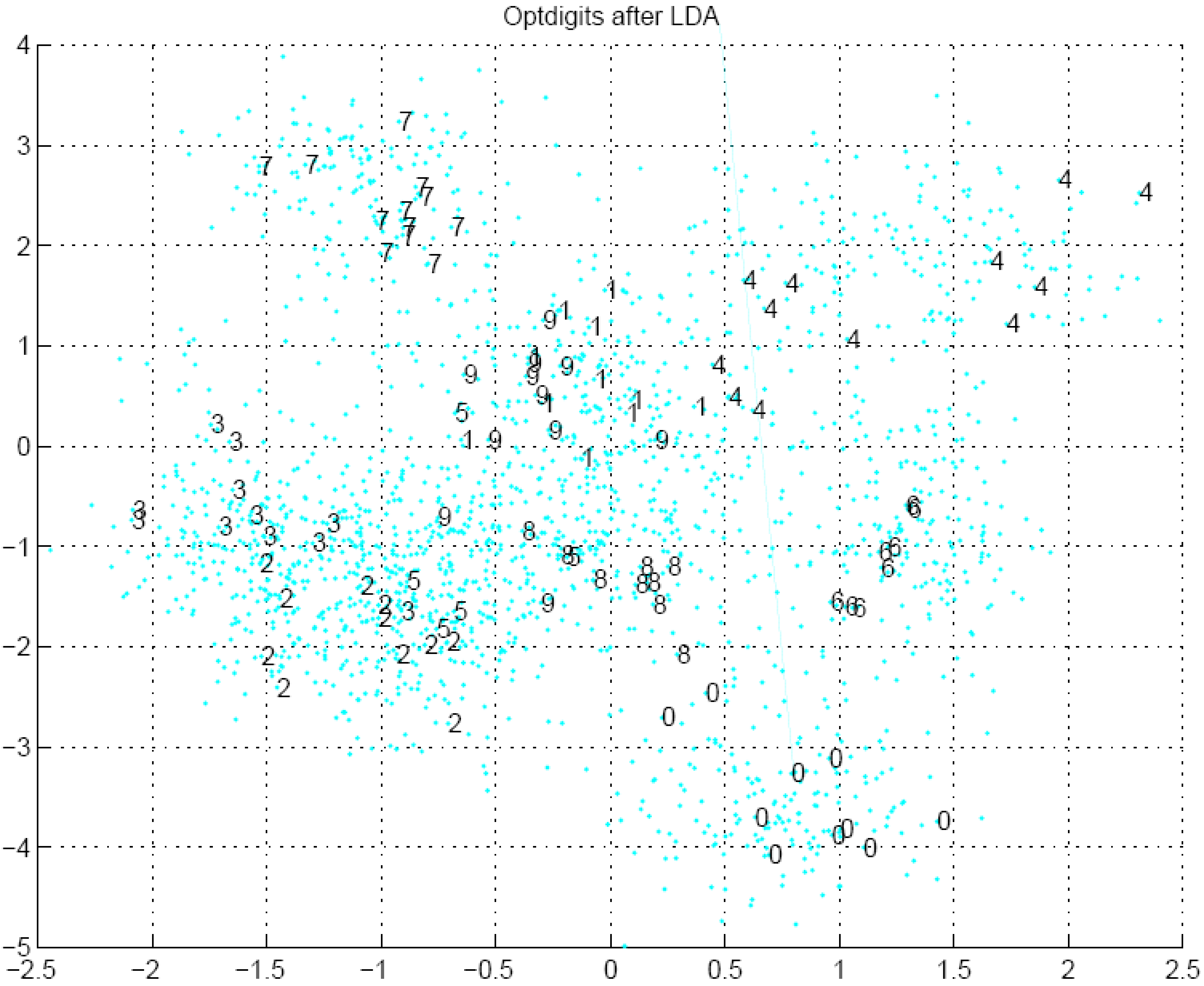
- Between-class scatter:

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i \left(\mathbf{m}_i - \mathbf{m}\right)\left(\mathbf{m}_i - \mathbf{m}\right)^T \qquad \mathbf{m} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{m}_i$$

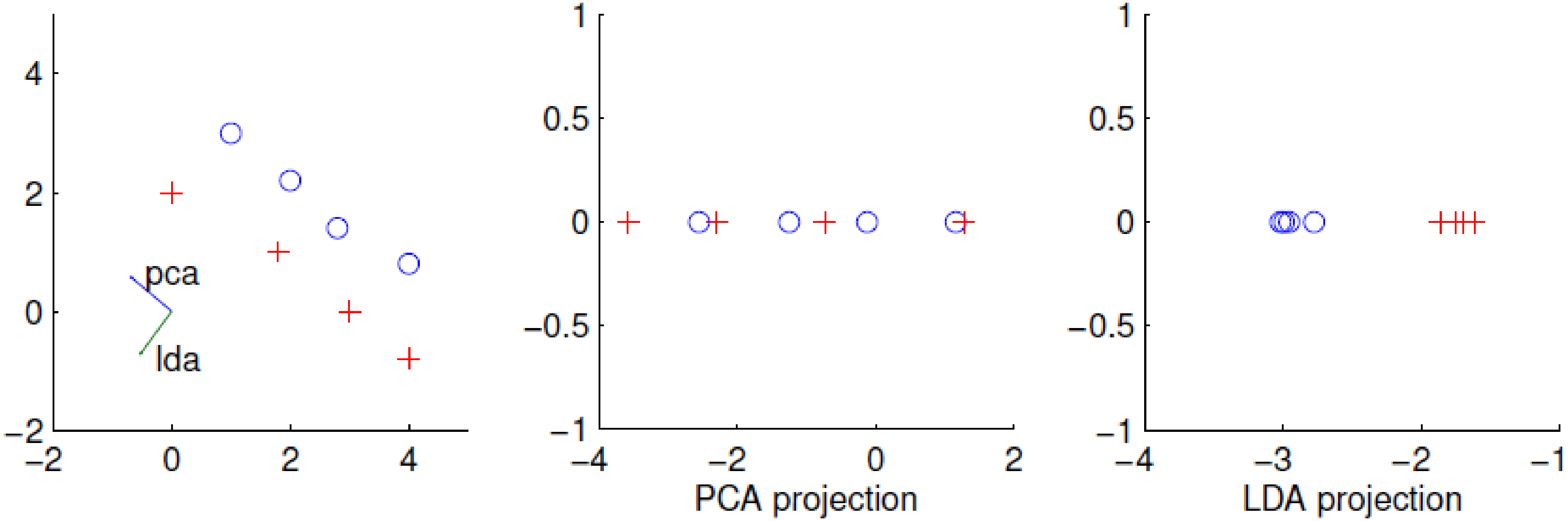
- Find **W** that max

$$J\left(\mathbf{W}\right) = \frac{\left|\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right|}$$

The largest eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_{B;}$ maximum rank of K-1


Optdigits after LDA
4
3
2
1
0
−1
−2
−3
−4
−5
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5

PCA vs LDA


pca
lda
4
2
0
−2
−2
0
2
4
1
0.5
0
−0.5
−1
−4
−2
0
2
PCA projection
1
0.5
0
−0.5
−1
−4
−3
−2
−1
LDA projection

Canonical Correlation Analysis

- **X**={$\boldsymbol{x}^t$,$\mathbf{y}^t$}$_t$; two sets of variables $\boldsymbol{x} \in R^d$ and $\boldsymbol{y} \in R^e$
- We want to find two projections ***w*** and ***v*** such that when ***x*** is projected along ***w*** and ***y*** is projected along ***v***, the correlation is maximized:

$$\rho = \mathrm{Corr}(\boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{v}^T\boldsymbol{y}) = \frac{\mathrm{Cov}(\boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{v}^T\boldsymbol{y})}{\sqrt{\mathrm{Var}(\boldsymbol{w}^T\boldsymbol{x})}\sqrt{\mathrm{Var}(\boldsymbol{v}^T\boldsymbol{y})}}$$

$$= \frac{\boldsymbol{w}^T\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{v}}{\sqrt{\boldsymbol{w}^T\mathrm{Var}(\boldsymbol{x})\boldsymbol{w}}\sqrt{\boldsymbol{v}^T\mathrm{Var}(\boldsymbol{y})\boldsymbol{v}}} = \frac{\boldsymbol{w}^T\mathbf{S}_{xy}\boldsymbol{v}}{\sqrt{\boldsymbol{w}^T\mathbf{S}_{xx}\boldsymbol{w}}\sqrt{\boldsymbol{v}^T\mathbf{S}_{yy}\boldsymbol{v}}}$$

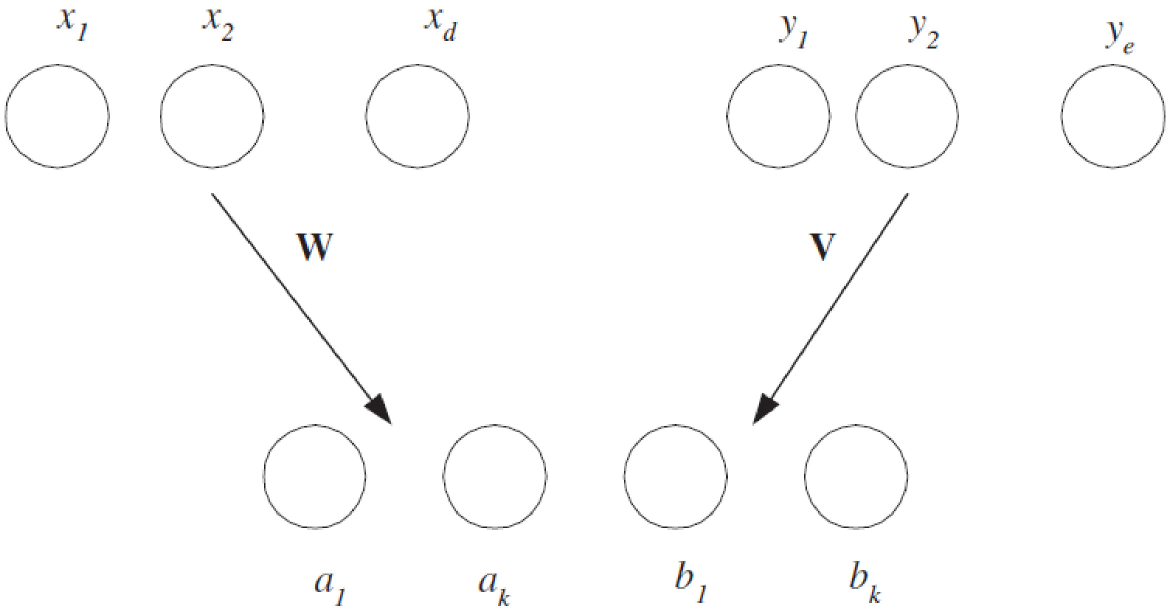
Canonical Correlation Analysis

- Maximize $\boldsymbol{w}^T\mathbf{S}_{xy}\boldsymbol{v}$ subject to $\boldsymbol{w}^T\mathbf{S}_{xx}\boldsymbol{w}=1$ and $\boldsymbol{v}^T\mathbf{S}_{yy}\boldsymbol{v}=1$.
- Writing these as Lagrangian terms …
- ***w*** should be an eigenvector of $\mathbf{S}_{xx}^{-1}\mathbf{S}_{xy}\mathbf{S}_{yy}^{-1}\mathbf{S}_{yx}$ and similarly ***v*** should be an eigenvector of $\mathbf{S}_{yy}^{-1}\mathbf{S}_{yx}\mathbf{S}_{xx}^{-1}\mathbf{S}_{xy}$
- Choose *k* as the dimensionality then we get the canonical variates by projecting the training instances along them

$$a_i^t = \mathbf{w}_i^T \mathbf{x}^t, b_i^t = \mathbf{v}_i^T \mathbf{y}^t, i = 1, \ldots, k \;\Rightarrow\; \boldsymbol{a}^t = \mathbf{W}^T \boldsymbol{x}^t, \boldsymbol{b}^t = \mathbf{V}^T \boldsymbol{y}^t$$

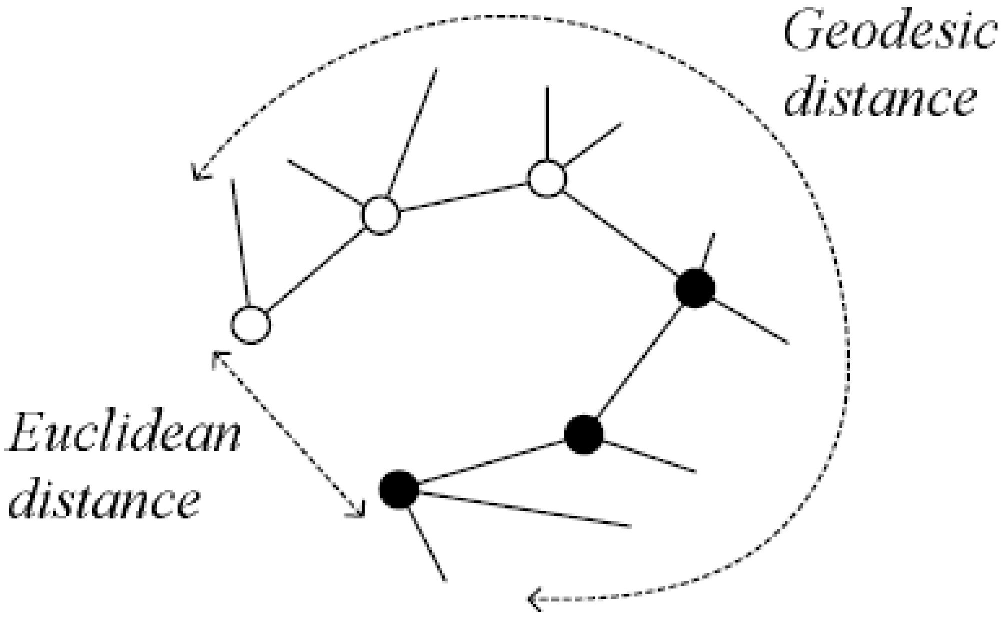
Canonical Correlation Analysis

- ***x*** and ***y*** may be two different views or modalities; e.g., image and word tags, and CCA does a joint mapping



(Isometric feature mapping) Isomap

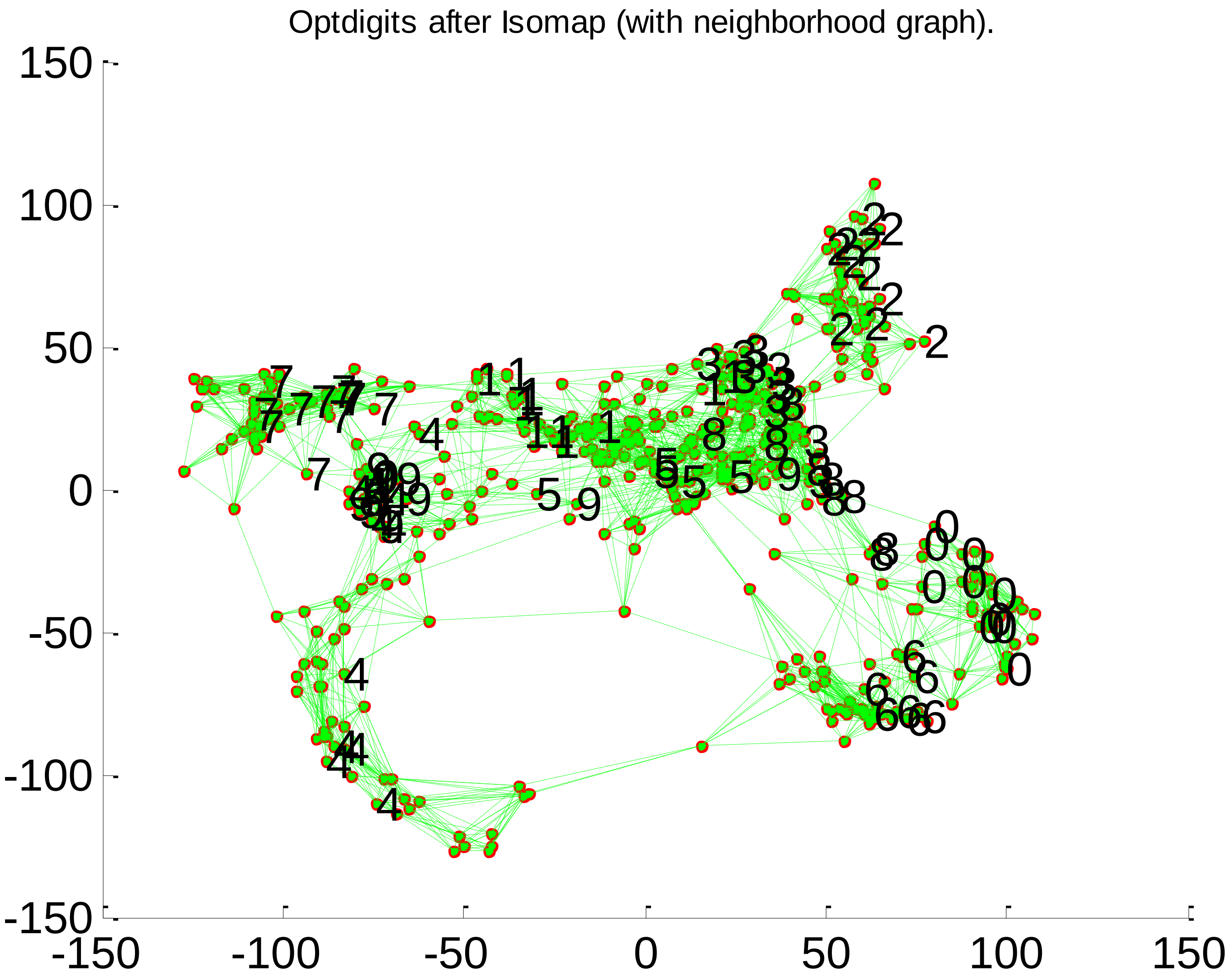
- Geodesic distance is the distance along the manifold that the data lies in, as opposed to the Euclidean distance in the input space



Isomap

- Instances *r* and *s* are connected in the graph if $\|\boldsymbol{x}^r-\boldsymbol{x}^s\|<\varepsilon$ or if $\boldsymbol{x}^s$ is one of the *k* neighbors of $\boldsymbol{x}^r$

  The edge length is $\|\boldsymbol{x}^r-\boldsymbol{x}^s\|$.
- For two nodes *r* and *s* not connected, the distance is equal to the shortest path between them.
- Once the $N\times N$ distance matrix is formed, use MDS to find a lower-dimensional mapping.
- This will have the effect of placing *r* and *s* that are far apart in the geodesic space also far apart in the new *k*-dim space even if they are close in terms of Euclidean distance in the original *d*-dim space.


Optdigits after Isomap (with neighborhood graph).

Matlab source from http://web.mit.edu/cocosci/isomap/isomap.html

Locally Linear Embedding

1. LLE recovers global nonlinear structure from locally linear fits.
2. Given $\boldsymbol{x}^r$ find its neighbors $\boldsymbol{x}^s{}_{(r)}$;
3. Find $\mathbf{W}_{rs}$ that minimize (using least squares subject to $\mathbf{W}_{rr} = 0, \forall r$ and $\Sigma_s \mathbf{W}_{rs} = 1$.)

$$E^w(\mathbf{W} \mid X) = \sum_r \left\| \mathbf{x}^r - \sum_s \mathbf{W}_{rs} \mathbf{x}^s_{(r)} \right\|^2$$

3. Find the new coordinates $\boldsymbol{z}^r$ that minimize

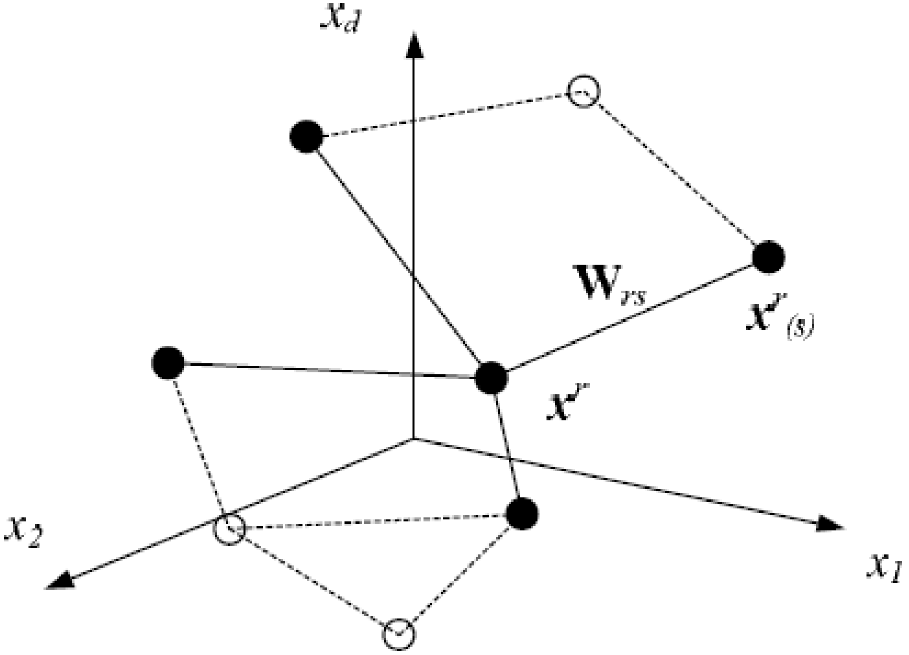
$$E^z(\mathbf{z} \mid \mathbf{W}) = \sum_r \left\| z^r - \sum_s \mathbf{W}_{rs} z^s_{(r)} \right\|^2$$

$$E^z(\mathbf{z} \mid \mathbf{W}) = \sum_{r,s} \mathbf{M}_{rs} (\mathbf{z}^r)^T \mathbf{z}^s$$

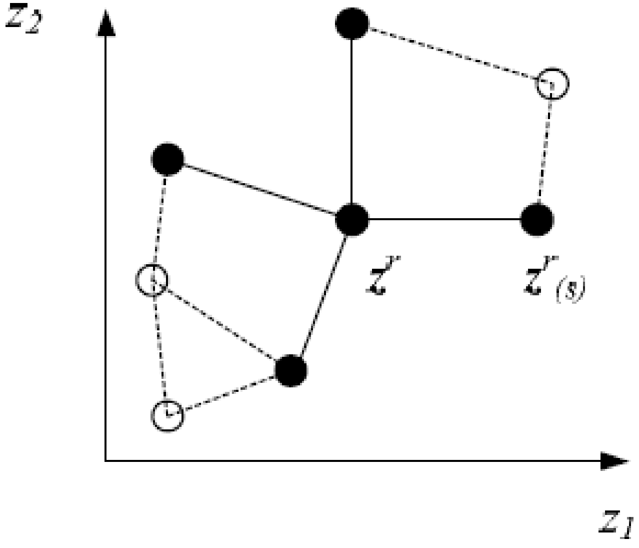
$$E[\mathbf{z}] = 0$$

$$Cov(\mathbf{z}) = \mathbf{I}$$

$$\mathbf{M}_{rs} = \delta_{rs} - W_{rs} - W_{sr} + \sum_i W_{ir} W_{is}$$

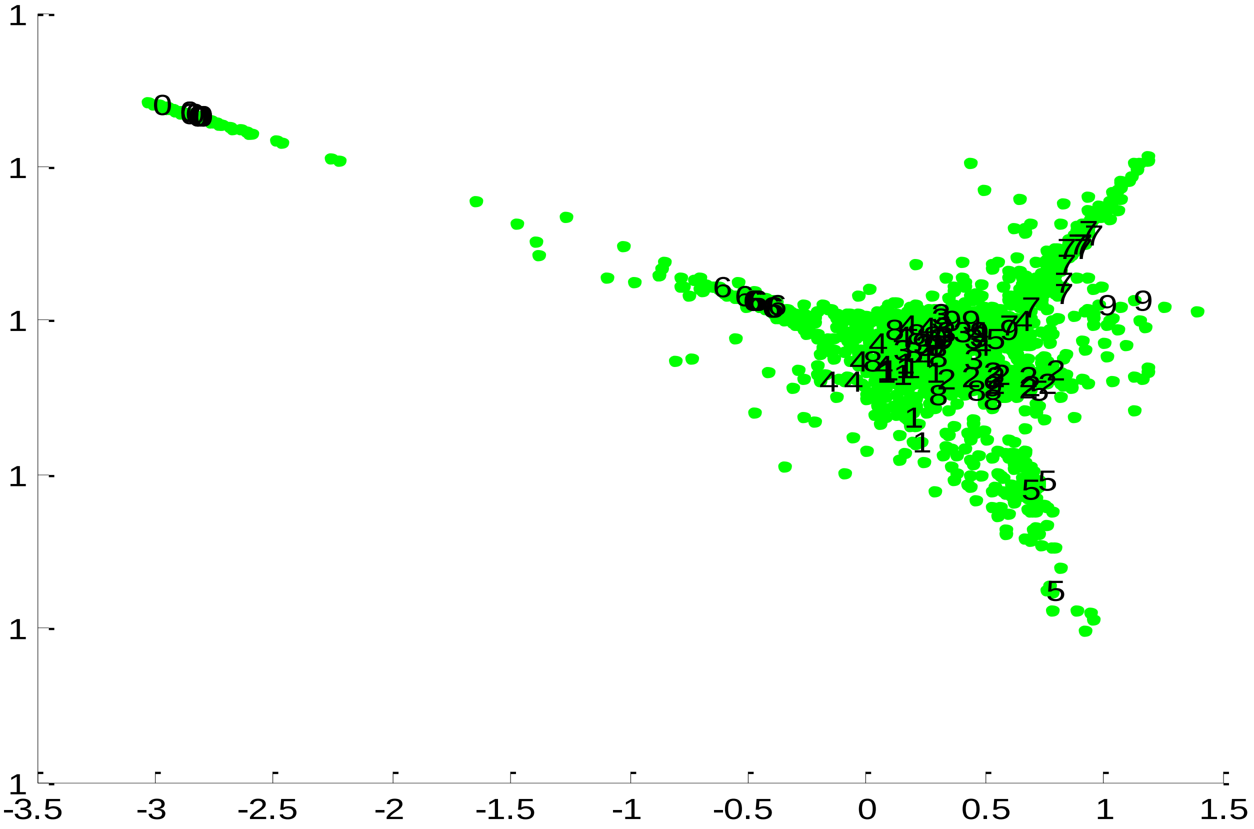


x space



z space

LLE on Optdigits



Matlab source from http://www.cs.toronto.edu/~roweis/lle/code.html

Laplacian Eigenmaps

- Let *r* and *s* be two instances and B_{rs} is their similarity, we want to find $\boldsymbol{z}^r$ and $\boldsymbol{z}^s$ that

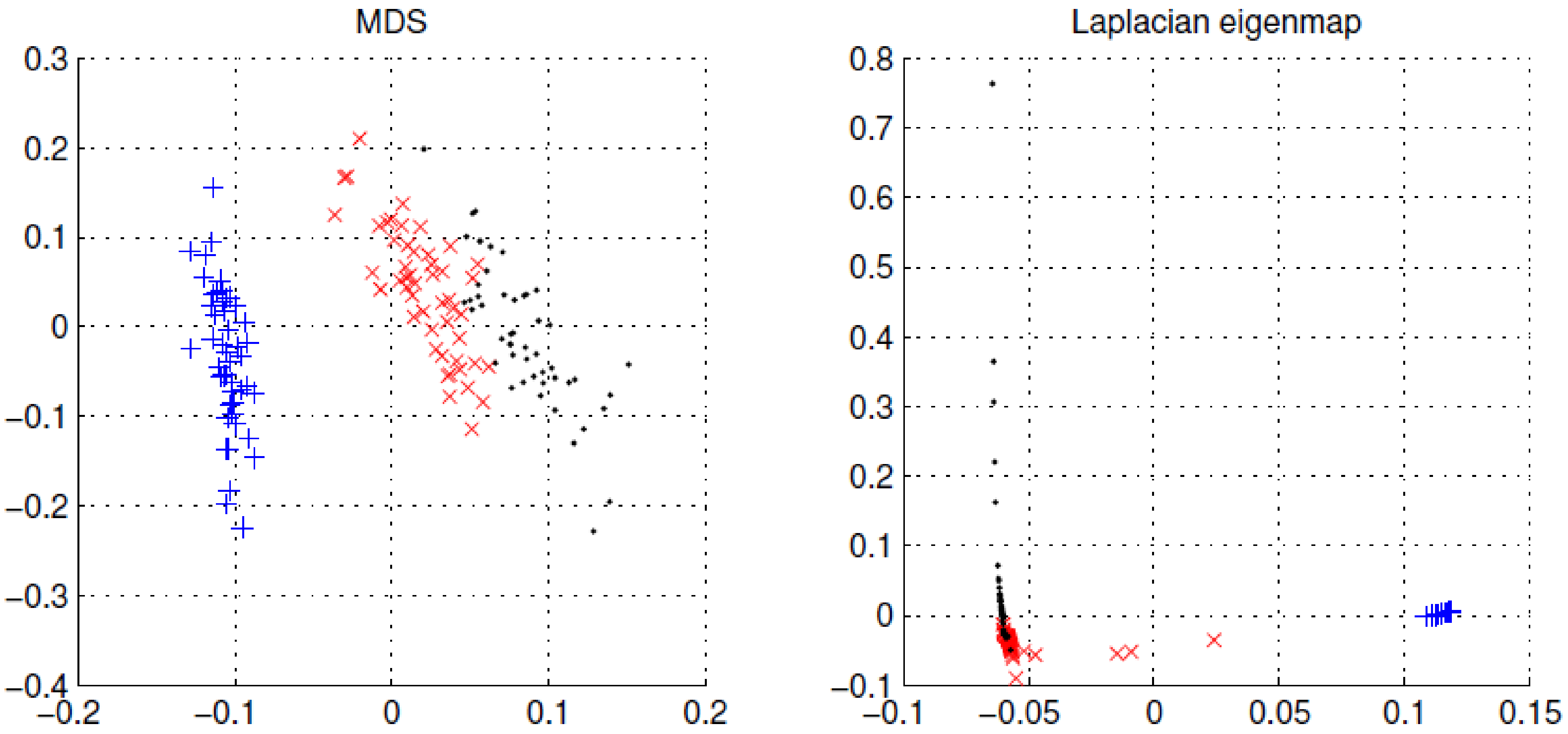
$$\min \sum_{r,s} \|\mathbf{z}^r - \mathbf{z}^s\|^2 B_{rs}$$

- B_{rs} can be defined in terms of similarity in an original space: 0 if $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$ are too far, otherwise

$$B_{rs} = \exp\left[-\frac{\|\mathbf{x}^r - \mathbf{x}^s\|^2}{2\sigma^2}\right]$$

- Defines a graph Laplacian, and feature embedding returns $\boldsymbol{z}^r$

Laplacian Eigenmaps on Iris



Spectral clustering (chapter 7)

Iris data reduced to two dimensions using multidimensional scaling and Laplacian eigenmaps. The latter leads to a more dense placement of similar instances.

Notes

- The forward and backward search procedures we discussed are local search procedures.
- You can use a stochastic procedure like simulated annealing or genetic algorithms to search more widely in the search space.
- There are also filtering algorithms for feature selection where heuristic measures are used to calculate the "relevance" of a feature in a preprocessing stage without actually using the learner.
- Projection methods work with numeric inputs, and discrete variables should be represented by 0/1 dummy variables, whereas subset selection can use discrete inputs directly.

Notes

- The projection methods we discussed are batch procedures in that they require that the whole sample be given before the projection directions are found. Mao and Jain (1995) discuss online procedures.
- Laplacian eigenmaps use the idea of feature embedding such that given pairwise similarities are preserved; the same idea is also used in kernel machines where pairwise similarities are given by a kernel function.
- Matrix decomposition methods are quite popular in various big data applications because they allow us to explain a large data matrix using smaller matrices.
- One example application is recommendation systems where we may have millions of movies and millions of customers and entries are customer ratings.

Notes

- There is a trade-off between feature extraction and decision making.
- If the feature extractor is good, the task of the classifier (or regressor) becomes trivial.
- If the classifier is good enough, then there is no need for feature extraction; it does its automatic feature selection or combination internally.
- There exist algorithms that do some feature selection internally, though in a limited way. Decision trees (Ch 9) do feature selection while generating the decision tree, and multilayer perceptrons (Ch 11) do nonlinear feature extraction in the hidden nodes.