

Dimensionality Reduction

- Olive slides: Alpaydin
- Numbered blue slides: Haykin, *Neural Networks: A Comprehensive Foundation*, Second edition, Prentice-Hall, Upper Saddle River:NJ, 1999.

[Alp] 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

[Alp] Feature Selection vs. Extraction

- Feature selection: Choosing $k < d$ important features, ignoring the remaining $d - k$
 - Subset selection algorithms
- Feature extraction: Project the original $x_i, i = 1, \dots, d$ dimensions to new $k < d$ dimensions, $z_i, i = 1, \dots, k$

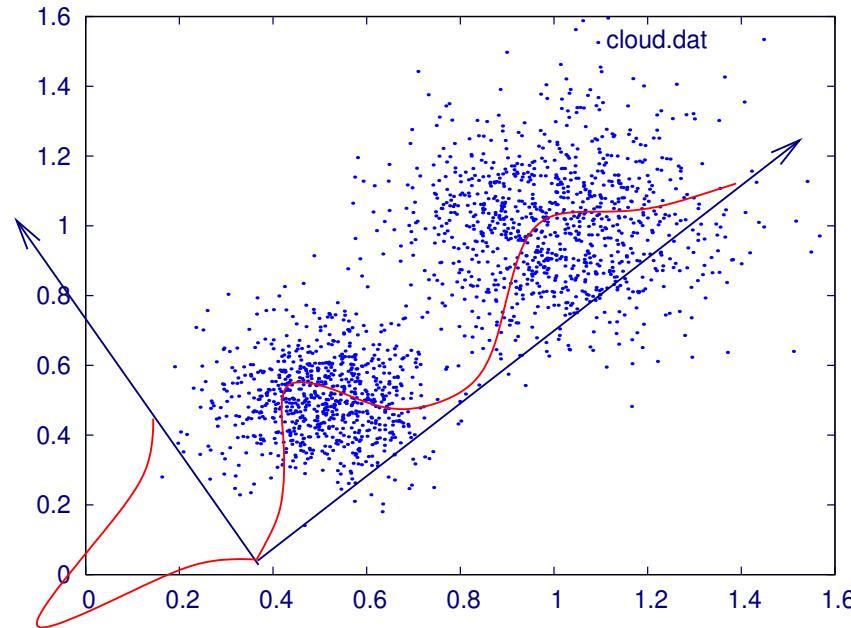
[Alp] Subset Selection

- There are 2^d subsets of d features
- Forward search: Add the best feature at each step
 - ▣ Set of features F initially \emptyset .
 - ▣ At each iteration, find the best new feature
$$j = \operatorname{argmin}_i E(F \cup x_i)$$
 - ▣ Add x_j to F if $E(F \cup x_j) < E(F)$
- 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)

Principal Components Analysis (PCA)

Note: \mathbf{Q} means eigenvector matrix of the covariance matrix, in Haykin slides. In Alpaydin slides, Σ is the covariance matrix.

Motivation



- How can we project the given data so that the variance in the projected points is maximized?

Eigenvalues/Eigenvectors

- For a square matrix \mathbf{A} , if a vector \mathbf{x} and a scalar value λ exists so that

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$$

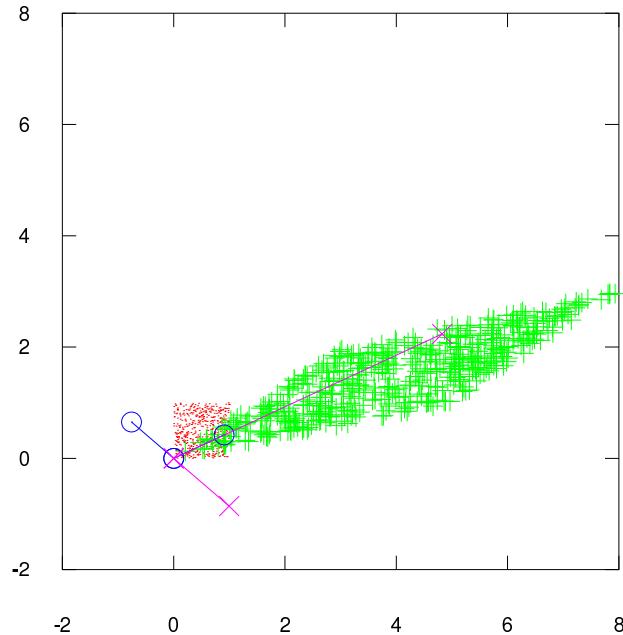
then \mathbf{x} is called an **eigenvector** of \mathbf{A} and λ an **eigenvalue**.

- Note, the above is simply

$$\mathbf{Ax} = \lambda\mathbf{x}$$

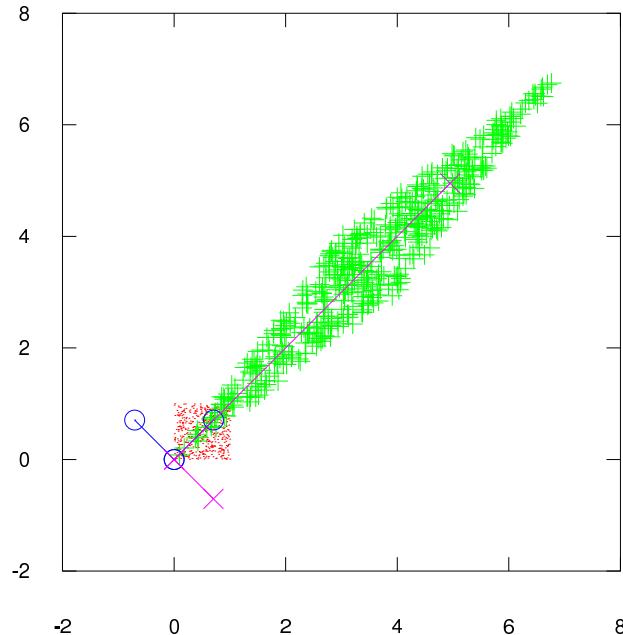
- An intuitive meaning is: \mathbf{x} is the direction in which applying the linear transformation \mathbf{A} only changes the magnitude of \mathbf{x} (by λ) but not the angle.
- There can be as many as n eigenvector/eigenvalue for an $n \times n$ matrix.

Eigenvector/Eigenvalue Example



- Red: original data \mathbf{x}
- Green: projected data using $A = \begin{bmatrix} 3 & 5 \\ 2 & 1 \end{bmatrix}$.
- Blue: Eigenvectors $\mathbf{v}_1 = (0.91, 0.42)$, $\mathbf{v}_2 = (-0.76, 0.65)$,
 $\lambda_1 = 5.3$, $\lambda_2 = -1.3$. Octave/Matlab code:
`[V, Lambda] = eig (A)`
- Magenta: A times eigenvectors.

Eigenvector/Eigenvalue Example 2



- Red: original data \mathbf{x}
- Green: projected data using $A = \begin{bmatrix} 3 & 4 \\ 4 & 3 \end{bmatrix}$.
- Blue: Eigenvectors; Magenta: A times eigenvectors.
- A is a symmetric matrix, so eigenvectors are orthogonal.

[Alp] Principal Component Analysis

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

$$\begin{aligned}\text{Var}(z) &= \text{Var}(\mathbf{w}^T \mathbf{x}) = E[(\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \boldsymbol{\mu})^2] \\ &= E[(\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \boldsymbol{\mu})(\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \boldsymbol{\mu})] \\ &= E[\mathbf{w}^T (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{w}] \\ &= \mathbf{w}^T E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] \mathbf{w} = \mathbf{w}^T \sum \mathbf{w}\end{aligned}$$

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

[Alp] PCA: Derivation

- Maximize $\text{Var}(z)$ subject to $\|w\| = 1$

$$\max_{w_1} w_1^T \Sigma w_1 - \alpha(w_1^T w_1 - 1)$$

$\sum w_1 = \alpha w_1$ that is, w_1 is an eigenvector of \sum

Choose the one with the largest eigenvalue for $\text{Var}(z)$ to be
max

- Second principal component: Max $\text{Var}(z_2)$, s.t.,
 $\|w_2\| = 1$ and orthogonal to w_1

$$\max_{w_2} w_2^T \Sigma w_2 - \alpha(w_2^T w_2 - 1) - \beta(w_2^T w_1 - 0)$$

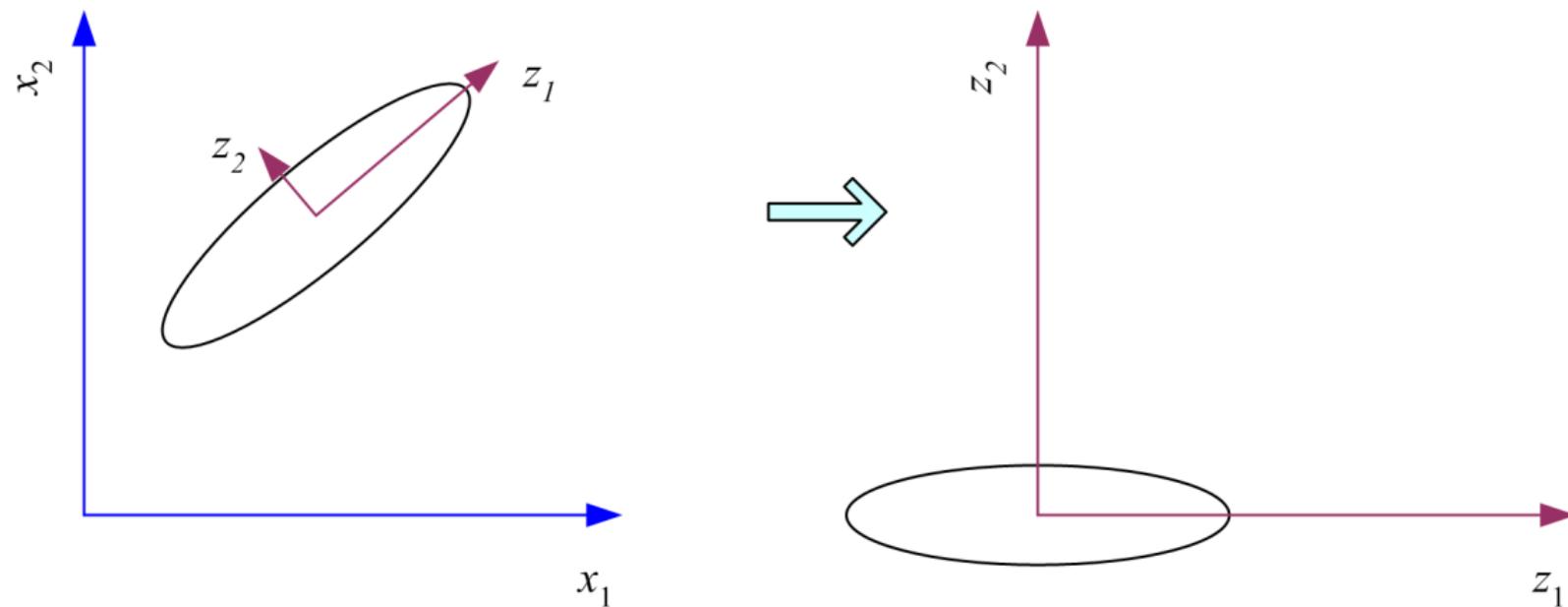
$\sum w_2 = \alpha w_2$ that is, w_2 is another eigenvector of \sum
and so on.

[Alp] What PCA Does

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

where the columns of \mathbf{W} are the eigenvectors of Σ
and \mathbf{m} is sample mean

Centers the data at the origin and rotates the axes



[Alp] PCA: How to Choose k

- Proportion of Variance (PoV) explained

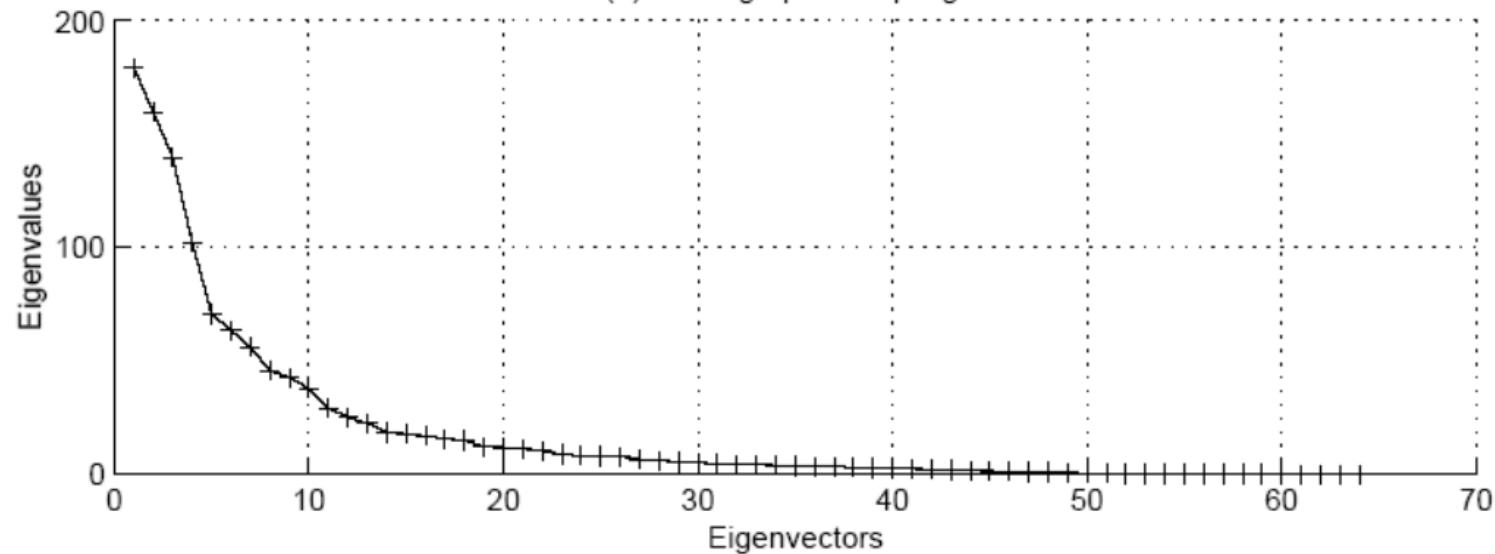
$$\frac{\lambda_1 + \lambda_2 + \dots + \lambda_k}{\lambda_1 + \lambda_2 + \dots + \lambda_k + \dots + \lambda_d}$$

when λ_i are sorted in descending order

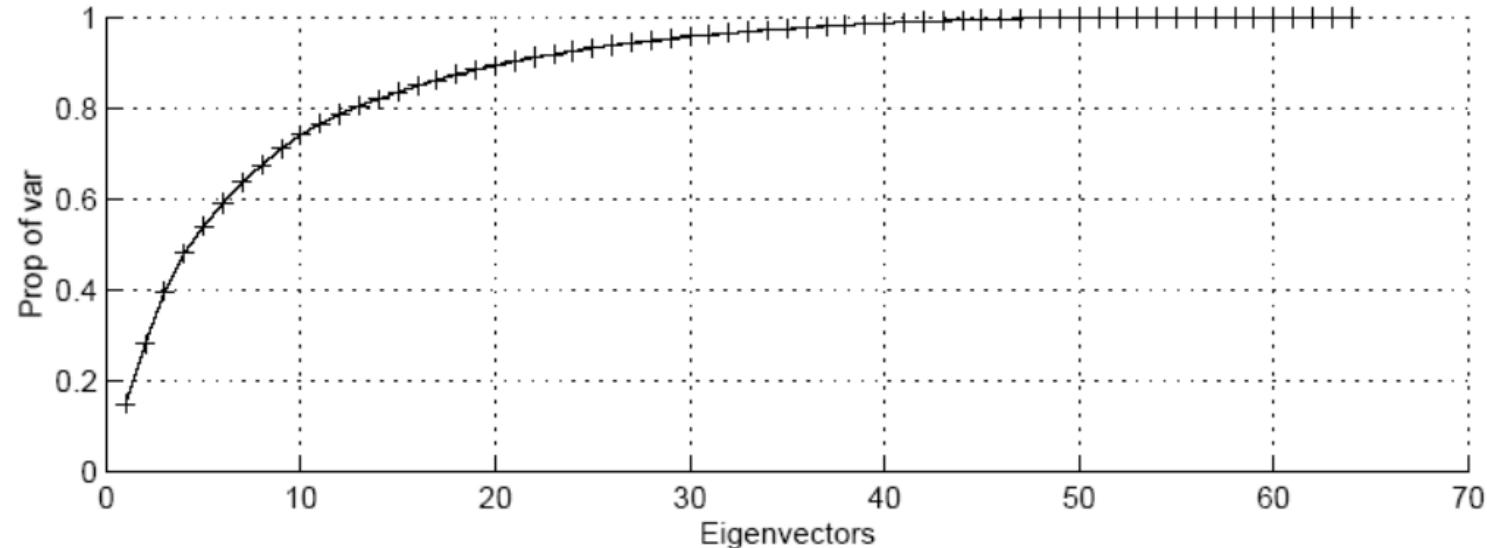
- Typically, stop at PoV > 0.9
- Scree graph plots of PoV vs k , stop at “elbow”

[Alp] PCA: Eigenvalues vs. Variance

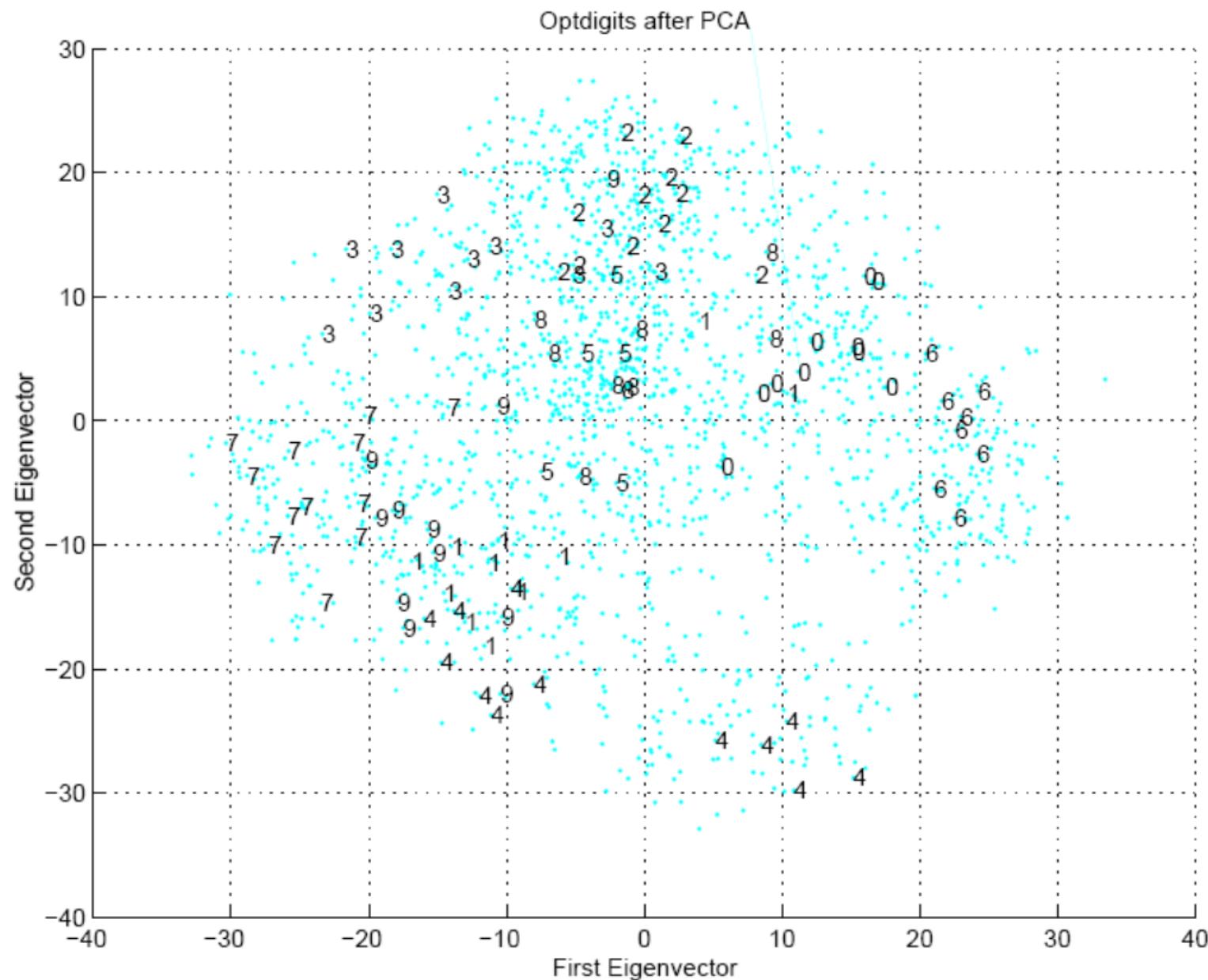
(a) Scree graph for Optdigits



(b) Proportion of variance explained



[Alp] PCA: Projection



PCA: Usage

- Project input \mathbf{x} to the principal directions:

$$\mathbf{a} = \mathbf{Q}^T \mathbf{x}.$$

- We can also recover the input from the projected point \mathbf{a} :

$$\mathbf{x} = (\mathbf{Q}^T)^{-1} \mathbf{a} = \mathbf{Q}\mathbf{a}.$$

- Note that we don't need all m principal directions, depending on how much variance is captured in the first few eigenvalues: We can do dimensionality reduction.

PCA: Dimensionality Reduction

- **Encoding:** We can use the first l eigenvectors to encode \mathbf{x} .

$$[a_1, a_2, \dots, a_l]^T = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l]^T \mathbf{x}.$$

- Note that we only need to calculate l projections a_1, a_2, \dots, a_l , where $l \leq m$.
- **Decoding:** Once $[a_1, a_2, \dots, a_l]^T$ is obtained, we want to reconstruct the full $[x_1, x_2, \dots, x_l, \dots, x_m]^T$.

$$\mathbf{x} = \mathbf{Q}\mathbf{a} \approx [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l][a_1, a_2, \dots, a_l]^T = \hat{\mathbf{x}}.$$

Or, alternatively

$$\hat{\mathbf{x}} = \mathbf{Q}[a_1, a_2, \dots, a_l, \underbrace{0, 0, \dots, 0}_{m-l \text{ zeros}}]^T.$$

PCA: Total Variance

- The total variance of the m components of the data vector is

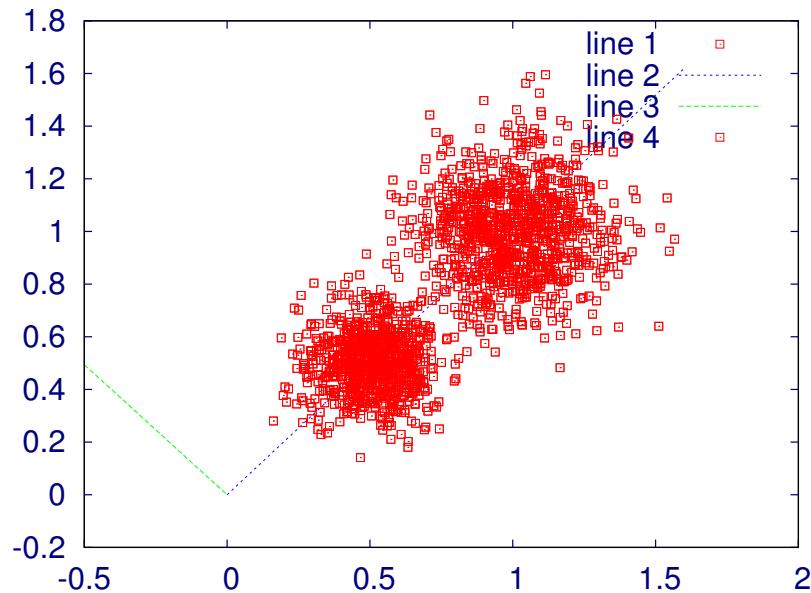
$$\sum_{j=1}^m \sigma_j^2 = \sum_{j=1}^m \lambda_j.$$

- The truncated version with the first l components have variance

$$\sum_{j=1}^l \sigma_j^2 = \sum_{j=1}^l \lambda_j.$$

- The larger the variance in the truncated version, i.e., the smaller the variance in the remaining components, the more accurate the dimensionality reduction.

PCA Example



```
inp=[randn(800,2)/9+0.5;randn(1000,2)/6+ones(1000,2)];
```

$$\mathbf{Q} = \begin{bmatrix} 0.70285 & -0.71134 \\ 0.71134 & 0.70285 \end{bmatrix}$$

$$\lambda = \begin{bmatrix} 0.14425 & 0.00000 \\ 0.00000 & 0.02161 \end{bmatrix}$$

[Alp] Factor Analysis

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

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

where $z_i, i = 1, \dots, k$ are the latent factors with

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

ε_i are the noise sources

$$E[\varepsilon_i] = \Psi_i, \text{Cov}(\varepsilon_i, \varepsilon_j) = 0, i \neq j, \text{Cov}(\varepsilon_i, z_j) = 0,$$

and v_{ij} are the factor loadings

[Alp] PCA vs. FA

□ PCA

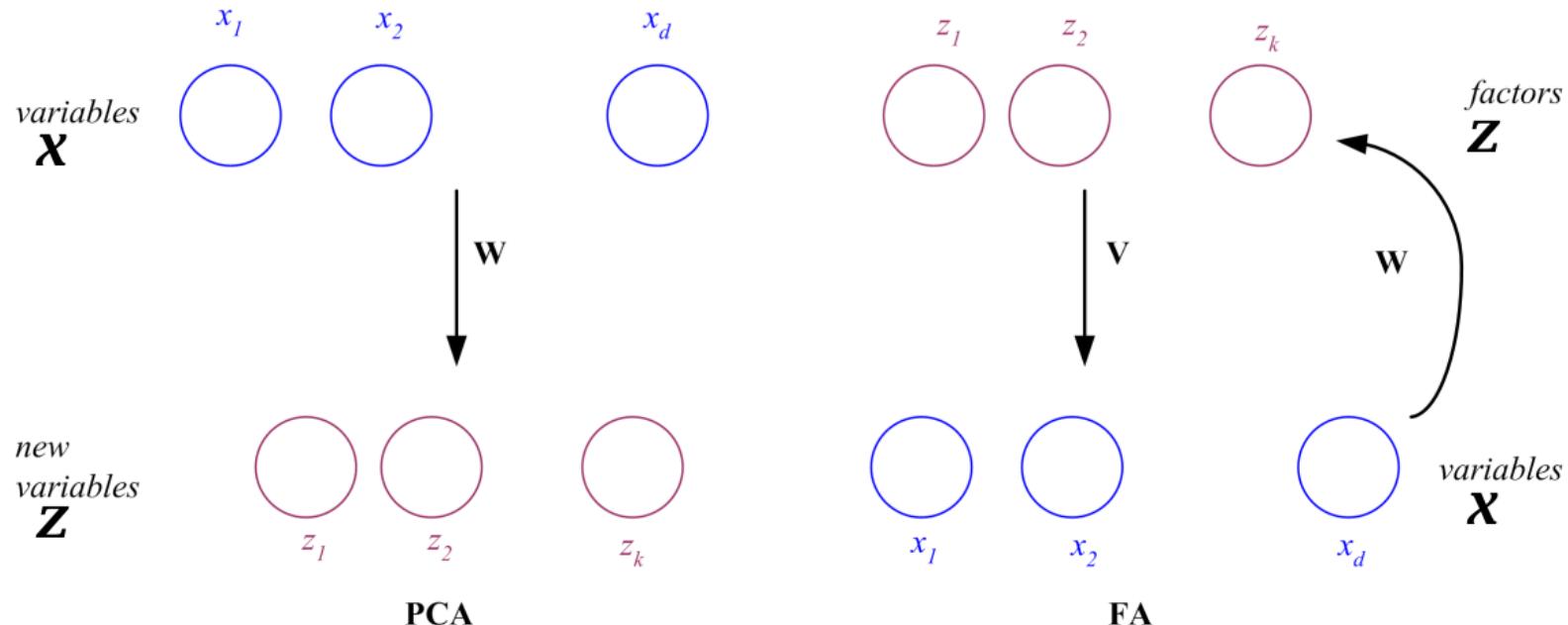
From \mathbf{x} to \mathbf{z}

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

□ FA

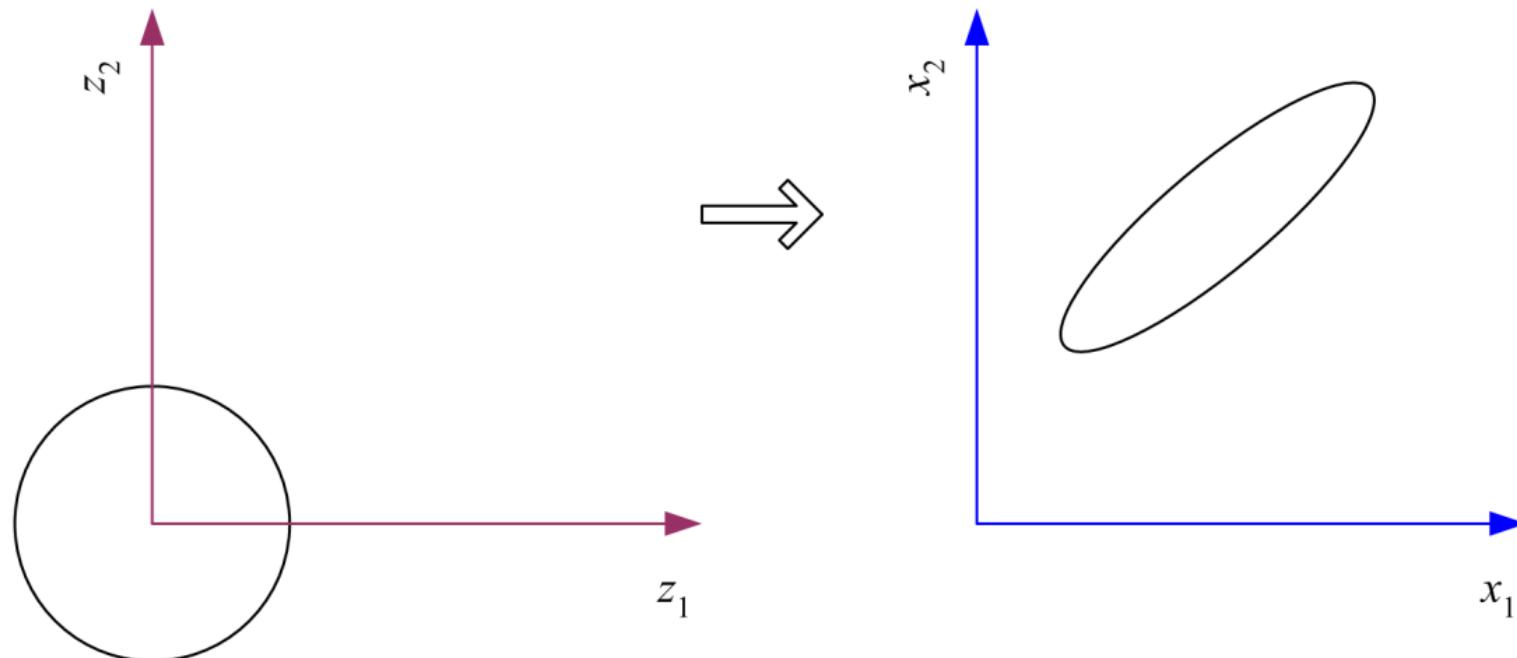
From \mathbf{z} to \mathbf{x}

$$\mathbf{x} - \boldsymbol{\mu} = \mathbf{V}\mathbf{z} + \boldsymbol{\varepsilon}$$



[Alp] Factor Analysis

- In FA, factors z_i are stretched, rotated and translated to generate x



[Alp] Singular Value Decomposition (SVD)

- Singular value decomposition: $\mathbf{X} = \mathbf{V} \mathbf{A} \mathbf{W}^T$
 \mathbf{V} is $N \times N$ and contains the eigenvectors of $\mathbf{X} \mathbf{X}^T$
 \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 diagonal
- $\mathbf{X} = \mathbf{U}_1 \sigma_1 \mathbf{V}_1^T + \dots + \mathbf{U}_k \sigma_k \mathbf{V}_k^T$ where k is the rank of \mathbf{X}

SVD Applications

$$\mathbf{V} : N \times N; \mathbf{A} : N \times d; \mathbf{W} : d \times d$$

$$\mathbf{X} = \mathbf{V}\mathbf{A}\mathbf{W}^T$$

$$\mathbf{X}^\dagger = (\mathbf{V}\mathbf{A}\mathbf{W}^T)^{-1} = \mathbf{W}\mathbf{A}^*\mathbf{V}^T$$

Note: \mathbf{A}^* : Transpose, then element-wise reciprocal.

Property: $\mathbf{X}^\dagger \mathbf{X} = \mathbf{I}$

Usages: Pseudo inverse (data set D with input vectors in \mathbf{X} and target values in \mathbf{y} ; figure out weight vector \mathbf{w}).

$$\mathbf{X}\mathbf{w} = \mathbf{y} \tag{1}$$

$$\mathbf{X}^\dagger \mathbf{X}\mathbf{w} = \mathbf{X}^\dagger \mathbf{y}$$

$$\mathbf{I}\mathbf{w} = \mathbf{X}^\dagger \mathbf{y}$$

$$\mathbf{w} = \mathbf{X}^\dagger \mathbf{y}$$

[Alp] Multidimensional Scaling (MDS)

- Given pairwise distances between N points,

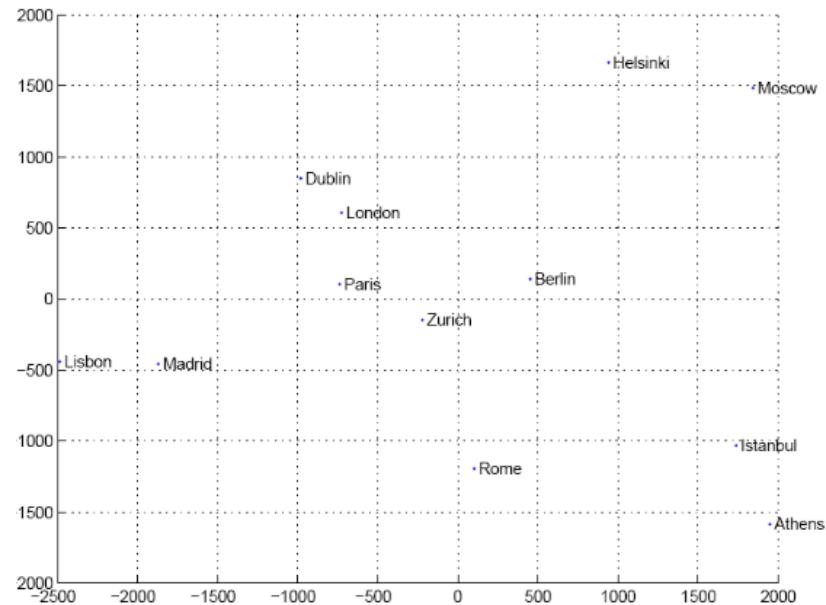
$$d_{ij}, i,j = 1, \dots, N$$

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

- $\mathbf{z} = \mathbf{g}(\mathbf{x} \mid \theta)$ Find θ that min Sammon stress

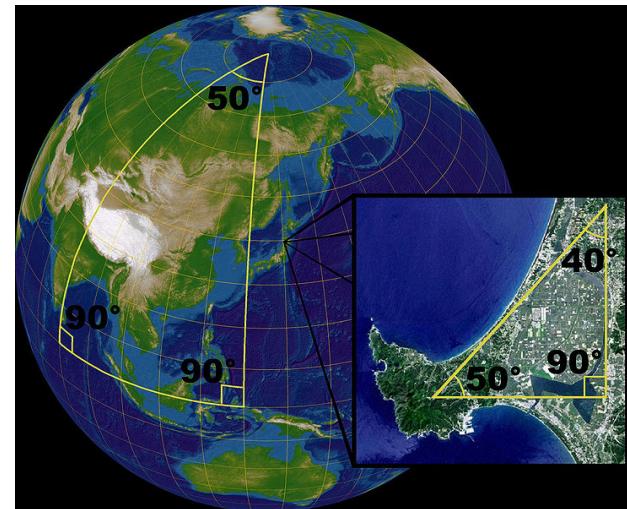
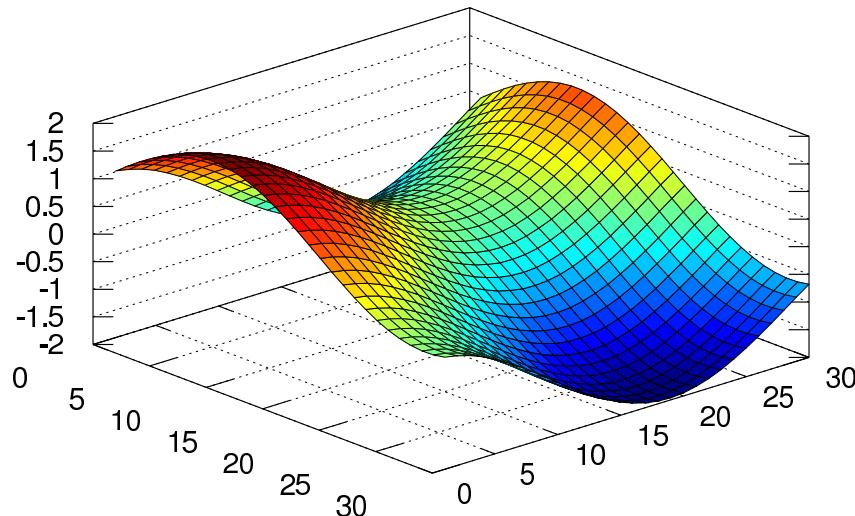
$$\begin{aligned} E(\theta \mid \mathcal{X}) &= \sum_{r,s} \frac{(\|\mathbf{z}^r - \mathbf{z}^s\| - \|\mathbf{x}^r - \mathbf{x}^s\|)^2}{\|\mathbf{x}^r - \mathbf{x}^s\|^2} \\ &= \sum_{r,s} \frac{(\|\mathbf{g}(\mathbf{x}^r \mid \theta) - \mathbf{g}(\mathbf{x}^s \mid \theta)\| - \|\mathbf{x}^r - \mathbf{x}^s\|)^2}{\|\mathbf{x}^r - \mathbf{x}^s\|^2} \end{aligned}$$

[Alp] Map of Europe by MDS



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

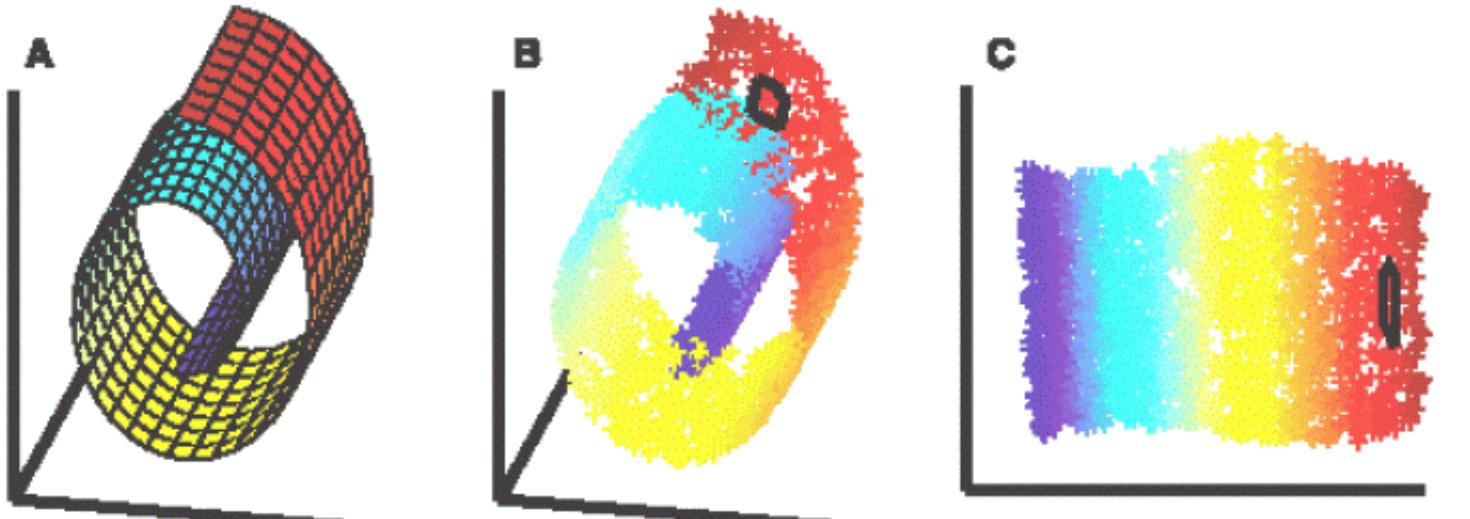
Manifolds



Lars H. Rohwedder, Wikimedia Commons

- A topological space that is locally Euclidean (flat, not curved).
- Dimensionality of the manifold = dimensionality of the Euclidean space it resembles, locally.
 - Straight line, wiggly curves, etc. are 1D manifolds.
 - Flat plane, surface of sphere, etc. are 2D manifolds.
- Detecting curvature of space: sum of internal angles of triangle = 180° ?

Manifold Learning

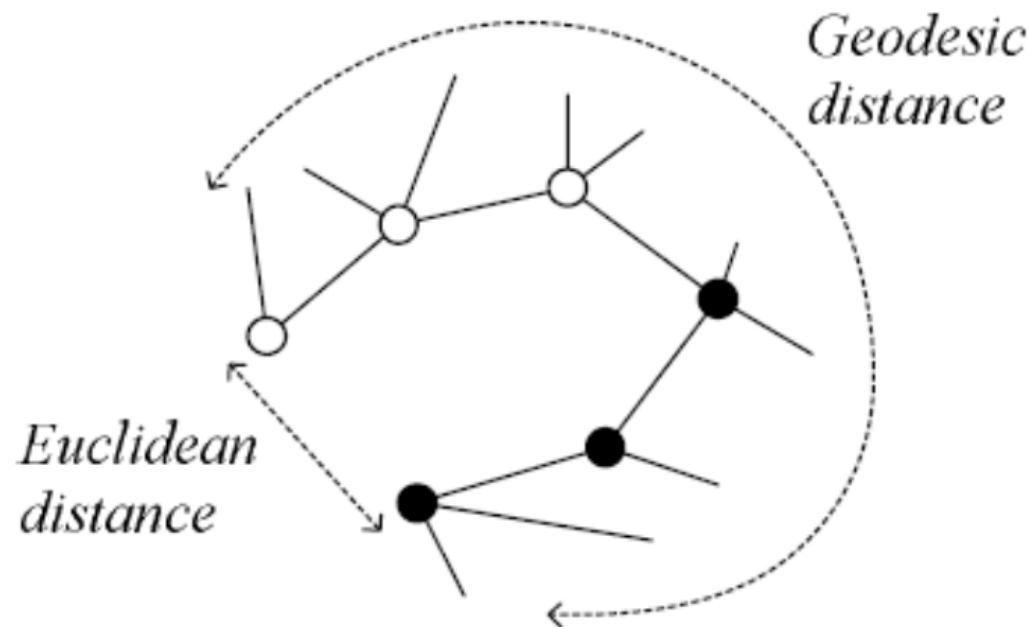


Choi, et. al
J. Pattern Recognition (2007)

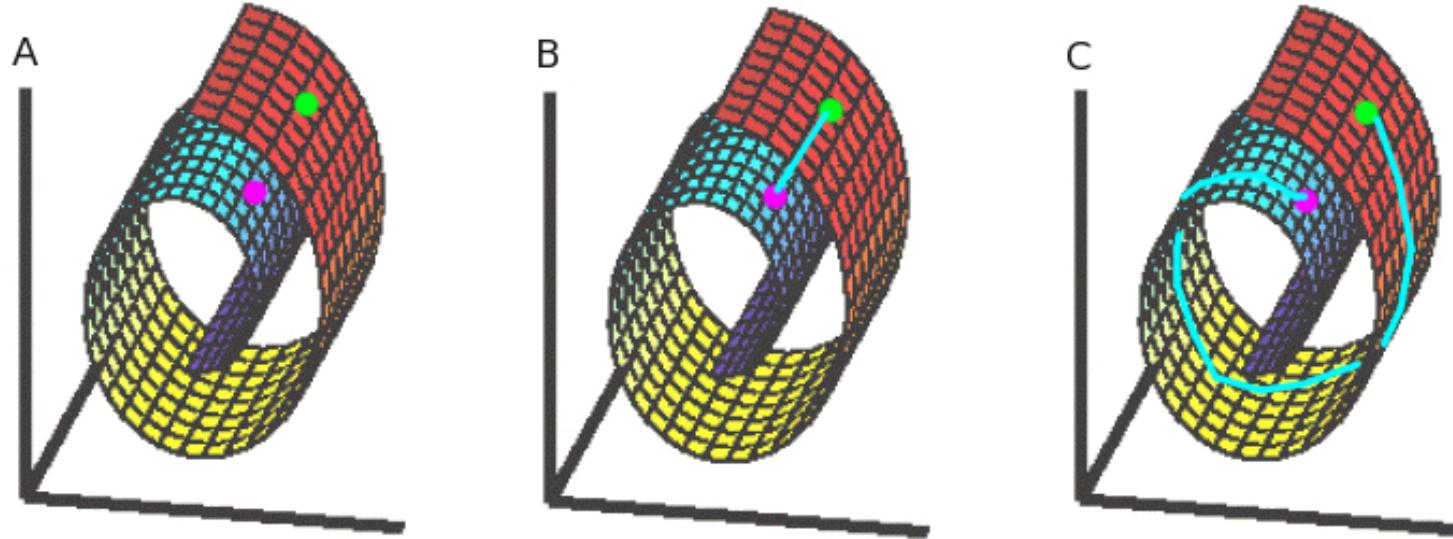
- A: 2D manifold embedded in 3D embedding space.
- B: Data points extraced from A.
- C: Recovered 2D structure.
- Task: recover C from B, without knowledge of A.

[Alp] Isomap

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



Geodesic Distance



Geodesic distance = Shortest path.

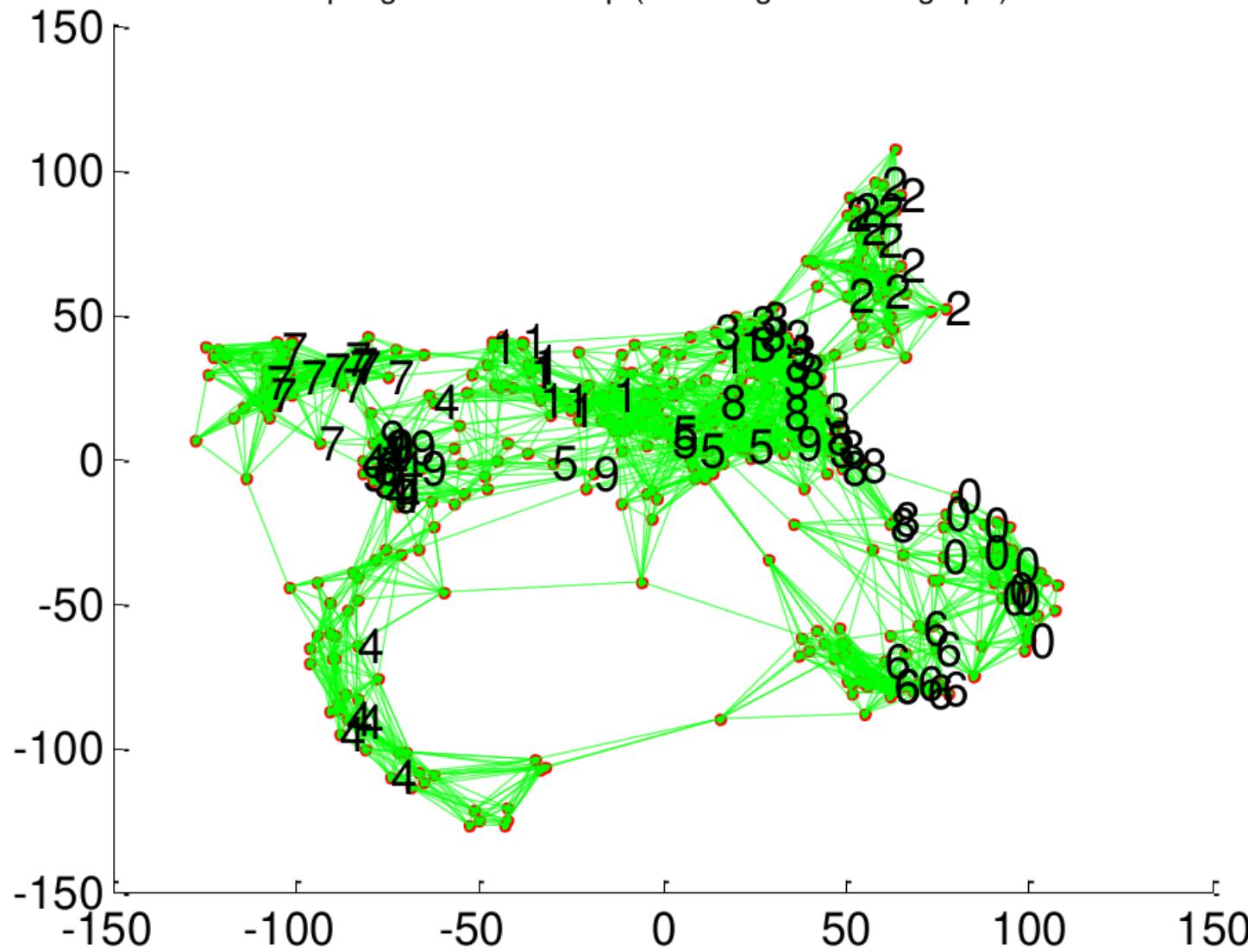
- A: Manifold with two points.
- B: Euclidean distance between the two points.
- C: Geodesic distance between the two points.

[Alp] Isomap Algorithm

- Instances r and s are connected in the graph if
 $\|x^r - x^s\| < \varepsilon$ or if x^s is one of the k neighbors of x^r
The edge length is $\|x^r - x^s\|$
- 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 thus formed, use MDS to find a lower-dimensional mapping

[Alp] Isomap Projection

Optdigits after Isomap (with neighborhood graph).



[Alp] Locally Linear Embedding (LLE)

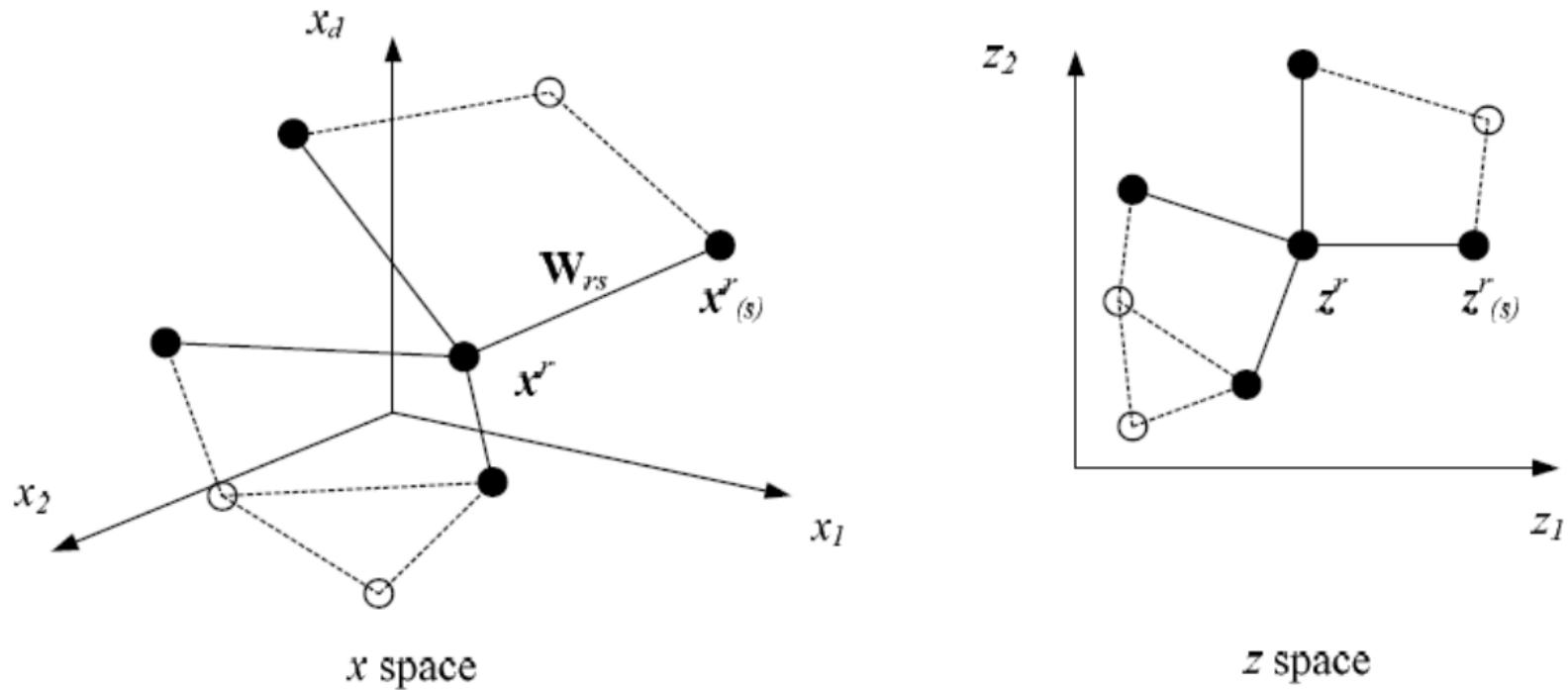
1. Given \mathbf{x}^r find its neighbors $\mathbf{x}_{(r)}^s$
2. Find \mathbf{W}_{rs} that minimize

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

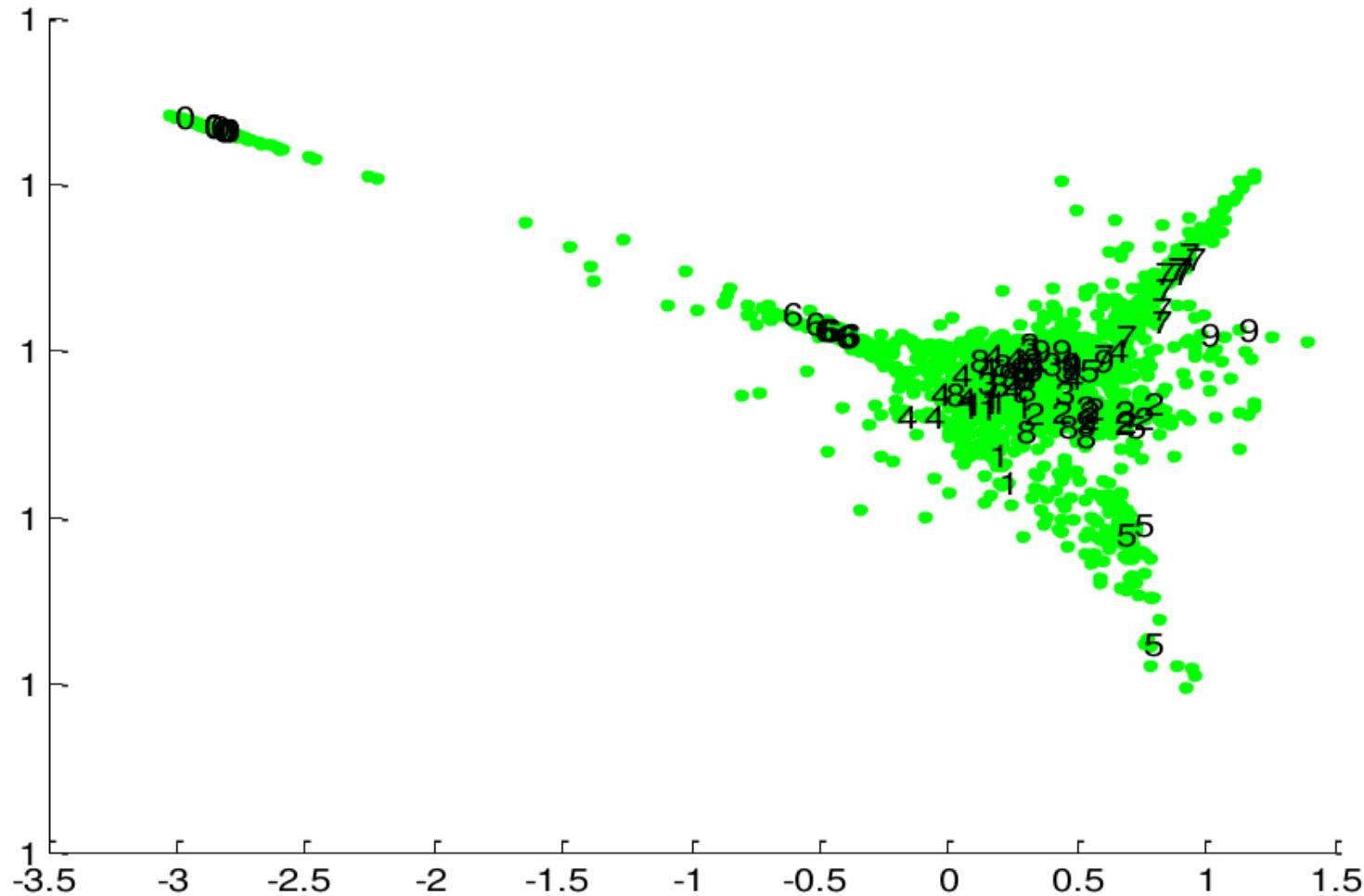
3. Find the new coordinates \mathbf{z}^r that minimize

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

[Alp] LLE Principle

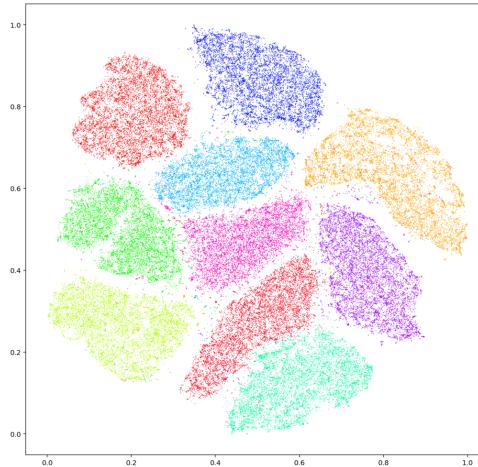


[Alp] LLE Projection



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

tSNE: t-distributed Stochastic Neighbor Embedding



tSNE project of the MNIST data : from wikipedia

- Step 1: Construct probability distribution P in high-D space over a pair of data points. High prob for similar objects, low prob for dissimilar objects.
- Step 2: Construct probability distribution Q in low-D space so that KL divergence between P and Q are minimized.