Statistical Pattern Recognition

Feature Extraction

Hamid R. Rabiee
Jafar Muhammadi, Alireza Ghasemi, Payam Siyari

Spring 2013
http://ce.sharif.edu/courses/91-92/2/ce725-1/
Agenda

- Dimensionality Reduction
- Feature Extraction
  - Feature Extraction Approaches
- Linear Methods
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
    - Multiple Discriminant Analysis (MDA)
  - PCA vs LDA
  - Linear Methods Drawbacks
- Nonlinear Dimensionality Reduction
  - ISOMAP
  - Local Linear Embedding (LLE)
  - ISOMAP vs. LLE
Dimensionality Reduction

- Feature Selection (discussed previous time)
  - Select the best subset from a given feature set
- Feature Extraction (will be discussed today)
  - Create new features based on the original feature set
  - Transforms are usually involved
**Feature Extraction**

\[ X_i = x_{i1}, x_{i2}, \ldots, x_{id} \]

\[ Y_i = f(X_i) = y_{i1}, y_{i2}, \ldots, y_{im}^T \]

\[ m \leq d, \text{ usually} \]

- For example:

\[ X = x_1 \ x_2 \ x_3 \ x_4^\top \Rightarrow Y = \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \end{bmatrix} \]
Feature Extraction Approaches

✧ The best f(x) is most likely a non-linear function, but linear functions are easier to find though

✧ Linear Approaches

✧ Principal Component Analysis (PCA) → will be discussed
  ✧ or Karhunen-Loeve Expansion (KLE)
✧ Linear Discriminant Analysis (LDA) → will be discussed
✧ Multiple Discriminant Analysis (MDA) → will be discussed
✧ Independent Component Analysis (ICA)
✧ Project Pursuit
✧ Factor Analysis
✧ Multidimensional Scaling (MDS)
**Feature Extraction Approaches**

✧ **Non-linear approach**
 ✧ Kernel PCA
 ✧ ISOMAP
 ✧ Locally Linear Embedding (LLE)

✧ **Neural Networks**
 ✧ Feed-Forward Neural Networks
    ✧ High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors.

✧ **Self-Organizing Map**
 ✧ A Clustering Approach to Dimensionality Reduction
 ✧ Transform data to lower dimensional lattice
Feature Extraction Approaches

✧ Another view

✧ Unsupervised approaches
  ✧ PCA
  ✧ LLE
  ✧ Self organized map

✧ Supervised approaches
  ✧ LDA
  ✧ MDA
**Principal Component Analysis (PCA)**

✧ **Main idea:**
  
  ✧ seek most accurate data representation in a lower dimensional space

✧ **Example in 2-D**

  ✧ Project data to 1-D subspace (a line) which minimize the projection error

  ✧ Notice that the good line to use for projection lies in the direction of largest variance

Diagram:

- Large projection error, bad line to project to
- Small projection errors, good line to project to
Principal Component Analysis (PCA)

- Preserves largest variances in the data
  - What is the direction of largest variance in data?
    - Hint: If $x$ has multivariate Gaussian distribution $N(\mu, \Sigma)$, direction of largest variance is given by eigenvector corresponding to the largest eigenvalue of $\Sigma$. 
Principal Component Analysis (PCA)

We can derive following algorithm (will be discussed in next slides)

PCA algorithm:

- X ← input \( n \times d \) data matrix (each row a d-dimensional sample)
- X ← subtract mean of X, from each row of X
  - The new data has zero mean (normalized data)
- \( \Sigma \) ← covariance matrix of X
- Find eigenvectors and eigenvalues of \( \Sigma \)
- C ← the M eigenvectors with largest eigenvalues, each in a column (a \( d \times M \) matrix) - value of eigenvalues gives importance of each component
- Y (transformed data) ← transform X using C \( (Y = X \times C) \)
  - The number of new dimensional is M \( (M<<d) \)
- Q: How much is the data energy loss?
Principal Component Analysis (PCA)

- Illustration:

Original axes

First principal component

Second principal component

Original axes
Principal Component Analysis (PCA)

✧ Example: Consider 2 classes below:

✧ $C1 = \{(0,-1),(1,0),(2,1)\}$

✧ $C2 = \{(1,1),(-1,1),(-1,-1),(-2,-1)\}$

Consider: $X = C1 \cup C2$

Then:

$\mu_x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$S_x = XX^T = \begin{bmatrix} 12 & 5 \\ 5 & 6 \end{bmatrix}$

$\Rightarrow \det(S_x - \lambda I) = \begin{vmatrix} 12 - \lambda & 5 \\ 5 & 6 - \lambda \end{vmatrix} = (12 - \lambda)(6 - \lambda) - 25 = 0$

$\Rightarrow \lambda \approx \begin{bmatrix} 15 \\ 3 \end{bmatrix}$

$\lambda_1 > \lambda_2$
Principal Component Analysis (PCA)

✧ Example:

\[ \begin{align*}
\lambda_1 &= 15 \\
\lambda_2 &= 3
\end{align*} \]

First principal component:

\[ S_X x = \lambda_1 x \Rightarrow (S_X - \lambda_1 I)x = 0 \Rightarrow x = \begin{bmatrix} 5 \\ 3 \end{bmatrix} \] (The corresponding eigenvector of \( \lambda_1 \))
Principal Component Analysis (PCA)

✧ Example:
Principal Component Analysis (PCA)

Example:

Projected Data Points in the new space:

\[ C1 = \{-0.5, 0.9, 2.3\} \]
\[ C2 = \{1.4, -0.4, -1.4, -2.3\} \]
Principal Component Analysis (PCA)

✧ **Drawbacks**

✧ **PCA was designed for accurate data representation, not for data classification**
  
  ✧ Preserves as much variance in data as possible
  
  ✧ If directions of maximum variance is important for classification, will work (give an example?)

✧ **However the direction of maximum variance may be useless for classification**
PCA Derivation

✧ Can be considered in many viewpoints:
  ✧ Minimum Error of Projection
  ✧ Maximum Information gain
  ✧ Or by Neural Nets

✧ The result would be the same!
PCA Derivation

- We want to find the most accurate representation of d-dimensional data $D=\{x_1, x_2, \ldots, x_n\}$ in some subspace $W$ which has dimension $k < d$

- Let $\{e_1, e_2, \ldots, e_k\}$ be the orthonormal basis for $W$. Any vector in $W$ can be written as
  \[ \sum_{i=1}^{k} \alpha_i e_i \]
  $e_i$s are d-dimensional vectors in original space.

- Thus $x_1$ will be represented by some vectors in $W$: $x_1 \approx \sum_{i=1}^{k} \alpha_i e_i$

- Error of this representation is $\text{error} = \left\| x_1 - \sum_{i=1}^{k} \alpha_i e_i \right\|^2$

- Then, the total error is:
  \[ J = \sum_{j=1}^{n} \left\| x_j - \sum_{i=1}^{k} \alpha_{ji} e_i \right\|^2 \]
  \[ = \sum_{j=1}^{n} \left\| x_j \right\|^2 - 2 \sum_{j=1}^{n} x_j^t \sum_{i=1}^{k} \alpha_{ji} e_i + \sum_{j=1}^{n} \sum_{i=1}^{k} \alpha_{ji}^2 \]
  \[ = \sum_{j=1}^{n} \left\| x_j \right\|^2 - 2 \sum_{j=1}^{n} \sum_{i=1}^{k} \alpha_{ji} x_j^t e_i + \sum_{j=1}^{n} \sum_{i=1}^{k} \alpha_{ji}^2 \]
PCA Derivation

To minimize $J$, need to take partial derivatives and also enforce constraint that $\{e_1, e_2, \ldots, e_k\}$ are orthogonal.

\[
J(e_1, \ldots, e_k, \alpha_{i1}, \ldots, \alpha_{nk}) = \sum_{j=1}^{n} \left\| x_j \right\|^2 - 2 \sum_{j=1}^{n} \sum_{i=1}^{k} \alpha_{ji} x_j^t e_i + \sum_{j=1}^{n} \sum_{i=1}^{k} \alpha_{ji}^2
\]

First take partial derivatives with respect to $\alpha_{ml}$

\[
\frac{\partial}{\partial \alpha_{ml}} J(e_1, \ldots, e_k, \alpha_{i1}, \ldots, \alpha_{nk}) = -2x_m^t e_l + 2\alpha_{ml}
\]

Thus the optimal value for $\alpha_{ml}$ is $-2x_m^t e_l + 2\alpha_{ml} = 0 \Rightarrow \alpha_{ml} = x_m^t e_l$

Plug the optimal value for $\alpha_{ml}$ back into $J$

\[
J(e_1, \ldots, e_k) = \sum_{j=1}^{n} \left\| x_j \right\|^2 - 2 \sum_{j=1}^{n} \sum_{i=1}^{k} (x_j^t e_i) x_j^t e_i + \sum_{j=1}^{n} \sum_{i=1}^{k} (x_j^t e_i)^2
\]

\[
= \sum_{j=1}^{n} \left\| x_j \right\|^2 - \sum_{j=1}^{n} \sum_{i=1}^{k} (x_j^t e_i)^2
\]

\[
= \sum_{j=1}^{n} \left\| x_j \right\|^2 - \sum_{i=1}^{k} e_i^t \sum_{j=1}^{n} (x_j x_j^t) e_i
\]

\[
= \sum_{j=1}^{n} \left\| x_j \right\|^2 - \sum_{i=1}^{k} e_i^t S e_i; \quad S = \sum_{j=1}^{n} x_j x_j^t
\]
PCA Derivation

✧ The $J$ is: $J(e_1, ..., e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} e_i^t S e_i$

✧ Minimizing $J$ is equivalent to maximizing $J' = \sum_{i=1}^{k} e_i^t S e_i$

✧ Then, the new problem is maximizing $J'$ with enforce constraints $e_i^t e_i = 1$ for all $i$

✧ Use the method of Lagrange multipliers, incorporate the constraints with undetermined $\lambda_1$, ..., $\lambda_k$. Need to maximize new function $u$

\[
u(e_1, ..., e_k) = \sum_{i=1}^{k} e_i^t S e_i - \sum_{j=1}^{k} \lambda_j e_j^t e_j - 1
\]

✧ Compute the partial derivatives with respect to $e_m$

\[
\frac{\partial}{\partial e_m} u(e_1, ..., e_k) = 2S e_m - 2\lambda_m e_m = 0 \Rightarrow S e_m = \lambda_m e_m
\]

✧ Thus $\lambda_m$ and $e_m$ are eigenvalues and eigenvectors of scatter matrix $S$
PCA Derivation

Let’s plug $e_m$ back into $J$ and use $S e_m = \lambda_m e_m$

$$J(e_1, ..., e_k) = \sum_{i=1}^{n} \|x_i\|^2 - \sum_{i=1}^{k} e_i^t S e_i$$

$$= \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} \lambda_i \|e_i\|^2 = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} \lambda_i$$

The first part of this equation is constant, Thus, to minimize $J$ take for the basis of $W$ the $k$ eigenvectors of $S$ corresponding to the $k$ largest eigenvalues

- The larger the eigenvalue of $S$, the larger is the variance in the direction of corresponding eigenvector
- This result is exactly what we expected: project $x$ into subspace of dimension $k$ which has the largest variance
- This is very intuitive: restrict attention to directions where the scatter is the greatest
- Thus PCA can be thought of as finding new orthogonal basis by rotating the old axis until the directions of maximum variance are found
Kernel PCA

- Assumption behind PCA is that the data points $x$ are multivariate Gaussian
  - Often this assumption does not hold
- However, it may still be possible that a transformation $\phi(x)$ is still Gaussian, then we can perform PCA in the space of $\phi(x)$
- Kernel PCA performs this PCA; however, because of “kernel trick,” it never computes the mapping $\phi(x)$ explicitly!

- Kernel methods will be discussed later!
**Linear Discriminant Analysis (LDA)**

- LDA, also known as Fisher Discriminant Analysis (FLD)
- The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible
Linear Discriminant Analysis (LDA)

✧ Main idea:
  ✧ find projection to a line so that samples from different classes are well separated

✧ Example in 2-D
  ✧ Project data to 1-D subspace (a line) which minimize the separation error

bad line to project to, classes are mixed up

good line to project to, classes are well separated
Linear Discriminant Analysis (LDA)

✧ We can derive following algorithm (will be discussed in next slides)

✧ LDA algorithm:

✧ $X_1, X_2 \leftarrow$ input $n_{1xd}$ and $n_{2xd}$ data matrices belong to class 1 and class 2

✧ $\mu_1, \mu_2 \leftarrow$ the means of $X_1$ and $X_2$

✧ $S_1, S_2 \leftarrow$ scatter matrices of $X_1$ and $X_2$ ($\text{Scatter} = n \ast \Sigma$ ; $n$: size of data)

✧ $S_w \leftarrow$ within class scatter matrix ($S_w = S_1 + S_2$)

✧ $V \leftarrow$ The direction of $V$ (the new 1-D space) obtains from $V = S_w^{-1}(\mu_1-\mu_2)$

✧ The border would be a point the new space, and a hyperplane in the original space (Why?).

✧ $Y$ (transformed data) $\leftarrow$ Project the old data onto new line
**LDA Derivation**

- Suppose we have 2 classes and d-dimensional samples $x_1, \ldots, x_n$ where
  - $n_1$ samples come from the first class
  - $n_2$ samples come from the second class

- The projection of sample $x_i$ onto a line in direction $v$ is given by $v^t x_i$

- How to measure separation between projections of different classes?
  - If $\mu'_1$ and $\mu'_2$ be the means of projections of classes 1 and 2, then $|\mu'_1 - \mu'_2|$ seems like a good measure

- The problem with this measure is that it does not consider the variances of the classes
  - we need to normalize that by a factor which is proportional to variance
  - we use the scatter (S) of the data
LDA Derivation

✧ The means and scatters of data are (for feature vectors $x_i$s):

\[
\mu_1 = \frac{1}{n_1} \sum_{x_i \in C_1} x_i \\
\mu_2 = \frac{1}{n_2} \sum_{x_i \in C_2} x_i
\]

\[
S_1^2 = \sum_{x_i \in C_1} (x_i - \mu_1)(x_i - \mu_1)^T \\
S_2^2 = \sum_{x_i \in C_2} (x_i - \mu_2)(x_i - \mu_2)^T
\]

✧ The means and scatters of projected data are: (why?)

\[
\mu_1' = \mathbf{v}^T \mu_1 \\
\mu_2' = \mathbf{v}^T \mu_1
\]

\[
S_1'^2 = \sum_{x_i \in C_1} (\mathbf{v}^T x_i - \mu_1')(\mathbf{v}^T x_i - \mu_1')^T \\
S_2'^2 = \sum_{x_i \in C_2} (\mathbf{v}^T x_i - \mu_2')(\mathbf{v}^T x_i - \mu_2')^T
\]

✧ Then we must maximize the following objective function.

\[
J(\mathbf{v}) = \frac{\mu_1' - \mu_2'}{S_1'^2 + S_2'^2}
\]

✧ We can consider another objective functions, too.
LDA Derivation

- All we need to do now is to express $J$ explicitly as a function of $v$ and maximize it.
- It is straightforward to see that $S'_1^2 = v'S_1v$ and $S'_2^2 = v'S_2v$.
- Therefore $S'_1^2 + S'_2^2 = v'S_wv$, where $S_w = S_1 + S_2$.
- Also it is straightforward to see that $(\mu'_1 - \mu'_2)^2 = v'S_Bv$, where $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t$.
- Then

$$J(v) = \frac{\mu'_1 - \mu'_2}{S'_1^2 + S'_2^2} = \frac{v'S_Bv}{v'S_wv}$$

- Maximize $J(v)$ by taking the derivative w.r.t. $v$ and setting it to 0.

$$\frac{\partial}{\partial v} J(v) = \frac{\partial}{\partial v} \left( v'S_Bv \right) v'S_wv - \left( \frac{\partial}{\partial v} v'S_wv \right) v'S_Bv = \frac{2S_Bv}{v'S_wv^2} v'S_wv - \frac{2S_Wv}{v'S_wv^2} v'S_Bv$$
LDA Derivation

✧ Need to solve $v^t S_W v (S_B v) - v^t S_B v (S_W v) = 0$, Then

\[
\frac{v^t S_w v (S_B v)}{v^t S_w v} - \frac{v^t S_B v (S_W v)}{v^t S_w v} = 0
\]

\[
\Rightarrow S_B v - \alpha (S_W v) = 0; \quad \alpha = J(v) = \frac{v^t S_B v}{v^t S_w v}
\]

\[
\Rightarrow S_B v = \alpha S_W v
\]

✧ $S_B v$ for any vector $v$, points in the same direction as $(\mu_1 - \mu_2)$

✧ $S_B v = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t v = (\mu_1 - \mu_2)((\mu_1 - \mu_2)^t v) = \beta (\mu_1 - \mu_2)$

✧ Then, $\beta (\mu_1 - \mu_2) = \alpha S_w v$

✧ If $S_W$ has full rank (the inverse exists), then:

\[
v = \gamma S_W^{-1} (\mu_1 - \mu_2)
\]
Linear Discriminant Analysis (LDA)

✧ Example: Consider 2 classes below:

✧ $C1 = \{(0,-1),(1,0),(2,1)\}$

✧ $C2 = \{(1,1),(-1,1),(-1,-1),(-2,-1)\}$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

$\mu_1 = (1,0)^T, \mu_2 = (-.75,0)^T$

$S_1 = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}, S_2 = \begin{bmatrix} 4.75 & 3 \\ 3 & 4 \end{bmatrix}$

$S_w = S_1 + S_2 = \begin{bmatrix} 6.75 & 5 \\ 5 & 6 \end{bmatrix}$

$\Rightarrow S_w^{-1} = \frac{1}{15.5} \begin{bmatrix} 6 & -5 \\ -5 & 6.75 \end{bmatrix} \approx \begin{bmatrix} .4 & -.33 \\ -.33 & .43 \end{bmatrix}$

$v = S_w^{-1}(\mu_1 - \mu_2) = \begin{bmatrix} .4 & -.33 \\ -.33 & .43 \end{bmatrix} \begin{bmatrix} 1.75 \\ 0 \end{bmatrix} = \begin{bmatrix} .7 \\ -.6 \end{bmatrix}$

$v = \frac{v}{\|v\|} = \begin{bmatrix} .75 \\ -.65 \end{bmatrix}$
Example: Consider 2 classes below:

- $C_1 = \{(0,-1),(1,0),(2,1)\}$
- $C_2 = \{(1,1),(-1,1),(-1,-1),(-2,-1)\}$

Projected $C_1$ data points = $v^T C_1 = \{.63, .77, .9\}$
Projected $C_2$ data points = $v^T C_2 = \{.13, -1.4, -.13, -.9\}$
Linear Discriminant Analysis (LDA)

✧ LDA Drawbacks

✧ Reduces dimension only to $k = c - 1$ (unlike PCA) ($c$ is the number of classes - why?)

✧ For complex data, projection to even the best hyperplane may result in inseparable projected samples

✧ Will fail:

✧ If $J(v)$ is always 0: happens if $\mu_1 = \mu_2$ (discriminatory information is not in the mean but rather in the variance of the data)

✧ If classes have large overlap when projected to any line
Multiple Discriminant Analysis (MDA)

- Can generalize LDA to multiple classes (how?)
  - Refer to the persian notes on the course page.
- In case of c classes, can reduce dimensionality to 1, 2, 3,…, c-1 dimensions (how and why?).
PCA vs LDA

- PCA (unsupervised)
  - Uses Total Scatter Matrix

- LDA (supervised)
  - Uses $|\text{between-class scatter matrix}| / |\text{within-class scatter matrix}|$

- PCA might outperform LDA when the number of samples per class is small or when the training data non-uniformly sample the underlying distribution
  - With few data, number of samples per class will be too low to have a reasonable estimation for covariance matrix, however the total number of samples may be still sufficient.

- Never knows in advance the underlying distributions for the different classes
Linear Methods Drawbacks

✧ Nonlinear Manifolds

✧ PCA uses the Euclidean distance

✧ Sometimes Euclidean distance is not proper:

*manifold* is a topological space which is locally Euclidean

What is important is the geodesic distance

Unroll the manifold
Deficiencies of Linear Methods

- Data may not be best summarized by linear combination of features
  - Example: PCA cannot discover 1D structure of a helix
  - Question: Does a nonlinear method can discover a perfect 1D structure for helix? (how?)
  - Did you realize what the nonlinear dimensionality reduction means?
Nonlinear Dimensionality Reduction

✧ Many data sets contain essential nonlinear structures that invisible to PCA and LDA.

✧ Resorts to some nonlinear dimensionality reduction approaches.
  ✧ Kernel methods (like kernel PCA)
    ✧ Depend on the kernels
    ✧ Most kernels are not data dependent
  ✧ Manifold based methods
    ✧ ISOMAP ← Will be discussed here!
    ✧ Locally Linear Embedding (LLE) ← Will be discussed here!
ISOMAP

✧ A non-linear approach for manifold learning (dimensionality reduction)
  ✧ Estimate the geodesic distance between points, by finding shortest paths in a graph with edges connecting neighboring data points

✧ Looking for new data points in a low dimensional space (d-dimensional) that preserve the geodesic distances.
**ISOMAP**

- **Construct neighborhood graph G**
  - In neighborhood graph, each sample is connected to $K$ nearest neighbors.
  - Steps to form neighborhood graph matrix ($D_G$)
    - Create binary $N \times N$ adjacency matrix so that each sample be connected to $K$ nearest neighbors
    - Compute all-pairs shortest path in $D_G$
      - Now $D_G$ is $N \times N$ geodesic distance matrix of two arbitrary points along the manifold
  - Use $D_G$ as distance matrix in MDS.
Multi Dimensional Scaling (MDS)

- MDS attempts to find an embedding from the K objects in $\mathbb{R}^N$, such that distances are preserved.

$$\min_{x_1, \ldots, x_K} \sum_{i<j} (\|x_i - x_j\| - D_{ij})^2$$

- Use D as distance matrix in MDS

  - The result of MDS is a N-dimensional Euclidean space $X$ that minimizes the cost function

An example of multi dimensional scaling

![Diagram of multi dimensional scaling](image-url)
**ISOMAP**

- Multi Dimensional Scaling (MDS)
  - The top $d$ eigenvectors of the dissimilarity matrix, represent the coordinates in the new $d$-dimensional Euclidean space.

- For more information visit the ISOMAP home page: [http://isomap.stanford.edu/](http://isomap.stanford.edu/)
**ISOMAP**

- **Advantages**
  - Nonlinear
  - Globally optimal
  - Still produces globally optimal low-dimensional Euclidean representation even though input space is highly folded, twisted, or curved.
  - Guarantee asymptotically to recover the true dimensionality.

- **Disadvantages**
  - May not be stable, dependent on topology of data
    - Sensitive to noise (short circuits)
  - Guaranteed asymptotically to recover geometric structure of nonlinear manifolds
    - As $N$ increases, pair wise distances provide better approximations to geodesics, but cost more computation
    - If $N$ is small, geodesic distances will be very inaccurate.
Local Linear Embedding (LLE)

- ISOMAP is a global approach
  - It uses geodesic distances and needs a graph traversal to compute them
  - Can we have the same functionality with a local approach?

- Local Linear Embedding (LLE)
  - A local approach to dimensionality reduction
  - LLE doesn’t use geodesic distances.
Local Linear Embedding (LLE)

✧ Main idea:

✧ Finding a nonlinear manifold by stitching together small linear neighborhoods.

✧ Assumption: manifold is approximately “linear” when viewed locally, that is, in a small neighborhood

✧ ISOMAP does this by doing a graph traversal.
Local Linear Embedding (LLE)

✧ LLE procedure

1) Compute the k nearest neighbors for each sample
2) Reconstruct each point using a linear combination of its neighbors
3) Find a low dimensional embedding which minimizes reconstruction loss
Local Linear Embedding (LLE)

✧ Each data point is constructed by its K neighbors (Step 2):

\[ \hat{X}_i = \sum_{j=1}^{K} W_{ij} \tilde{X}_j ; \quad \sum_{j=1}^{K} W_{ij} = 1 \]

✧ \( W_{ij} \) summarizes the contribution of the \( j \)-th data point to the \( i \)-th data reconstruction

✧ To obtain the weights we must solve the following optimization problem:

\[ \varepsilon(W) = \min_W || X_i - \sum_{j=1}^{K} W_{ij} X_j ||^2 ; \quad \sum_{j=1}^{K} W_{ij} = 1 \]

✧ Find a low dimensional embedding which minimizes reconstruction loss (Step 3):

\[ \phi(Y) = \min_Y || Y_i - \sum_{j=1}^{K} W_{ij} Y_j ||^2 \]
Local Linear Embedding (LLE)

✧ The weights that minimize the reconstruction errors are invariant to rotation, rescaling and translation of the data points.
  ✧ Invariance to translation is enforced by adding the constraint that the weights sum to one (Why?).
  ✧ The weights characterize the intrinsic geometric properties of each neighborhood.

✧ The same weights that reconstruct the data points in D dimensions should reconstruct it in the manifold in d dimensions (d<D).
  ✧ Local geometry is preserved
Local Linear Embedding (LLE)

✧ Meaning of W: a linear representation of every data point by its neighbors
  ✧ This is an intrinsic geometrical property of the manifold
  ✧ A good projection should preserve this geometric property as much as possible

✧ In LLE, we must solve two optimization problems:
  ✧ First optimization problem: finding W
    ✧ It is a “Constrained Least Square” problem
    ✧ It is also a convex optimization problem
  ✧ Second optimization problem: finding vectors Y
    ✧ It is a “Least Square” problem
    ✧ It is also a convex optimization problem, too.
Local Linear Embedding (LLE)

- Optimization problem 1: Obtaining $W$
  - Compute the optimal weight for each point individually:

$$
e = |x_i - \sum_{j \in \text{Neighbors of } x_i} w_{ij} x_j|^2 = \sum_j w_{ij} (x_i - x_j)^2 = \sum_j \sum_{p,q} w_{ij} w_{jk} C_{jk}
$$

$$C_{jk} = (x_j - x_k)^T (x_j - x_k)$$

- This error can be minimized in closed form, using Lagrange multipliers to enforce the constraint that $\sum_j w_{ij} = 1$ in terms of the $C$, the optimal weights are given by:
  - $W_{ij}$ is Zero for all non-neighbors of $x$

$$w_{ij} = \frac{\sum_k C_{jk}^{-1}}{\sum_p \sum_q C_{pq}^{-1}}$$
Local Linear Embedding (LLE)

✧ Optimization problem 2: Obtaining Y

✧ The following is a more direct and simpler derivation for Y:

\[ \Phi(Y) = \sum_i \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2 = \sum_i \left\| Y_i - [Y_1;Y_2;\ldots;Y_n]W_i^T \right\|^2 \]

\[ = \left\| [Y_1;Y_2;\ldots;Y_n] - [Y_1;Y_2;\ldots;Y_n][W_1^T;W_2^T;\ldots;W_n^T] \right\|^2 \]

\[ = \left\| Y - YW^T \right\|_F^2 = \left\| Y(I - W)^T(I - W)Y^T \right\|_F^2 = \text{trace}(Y(I - W)^T(I - W)Y^T) \]

\[ = \text{trace}(YM^T), \text{ where } Y = [Y_1;Y_2;\ldots;Y_n], \quad M = (I - W)^T(I - W) \]

Which \( \| \cdot \|_F \) indicates the Frobenius norm, i.e. \( \| A \|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\text{trace}(AA^T)} \)

✧ Y is given by the eigenvectors of the lowest \( d \) non-zero eigenvalues of the matrix

\[ M = (I - W)^T(I - W) \]

✧ For more information visit the LLE home page: http://cs.nyu.edu/~roweis/lle/
Local Linear Embedding (LLE)

✧ Some Limitations of LLE

✧ require dense data points on the manifold for good estimation
✧ A good neighborhood seems essential to their success
  ✧ How to choose k?
    ✧ Too few neighbors: Result in rank deficient tangent space and lead to over-fitting
    ✧ Too many neighbors: Tangent space will not match local geometry well
**ISOMAP vs. LLE**

✧ ISOMAP preserves the neighborhoods and their geometric relation better than LLE.

✧ LLE requires massive input data sets and it must have same weight dimension.

✧ Merit of ISOMAP is fast processing time with Dijkstra’s algorithm.

✧ ISOMAP is more practical than LLE.
Any Question?

End of Lecture 4

Thank you!

Spring 2013

http://ce.sharif.edu/courses/91-92/2/ce725-1/