Dimensionality Reduction

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

NetDB-ML, Spring 2015

Outline

Outline

Why Dimensionality Reduction?

- Given a (normalized) training dataset $\{\boldsymbol{x}^{(t)}, \boldsymbol{r}^{(t)}\}$ where $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$, we want to reduce the input dimension from d to k, $k < d$
- Why dimensionality reduction?

Why Dimensionality Reduction?

- Given a (normalized) training dataset $\{\boldsymbol{x}^{(t)}, \boldsymbol{r}^{(t)}\}$ where $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$, we want to reduce the input dimension from d to k, $k < d$
- Why dimensionality reduction?
- To reduce the time/space requirements in training, cross validation, testing, and prediction
- To save the cost of data collection
  - If an input is decided to be unnecessary, we can stop observing it in the future
- To make the classifier/regressor robust to small datasets
  - Fewer parameters (e.g., elements of Σ) implies lower model complexity, and lower variance due to particulars of a sample
- To extract knowledge
  - The k inputs can describe the whole data and may explain something

Feature Selection vs. Feature Extraction

- Basically, two ways to dimensionality reduction:
- ***Feature selection***:
  - Keeping only k of d dimensions that is most helpful to classification/regression
- ***Feature extraction***:
  - Finding a new set of k dimensions that are combinations of the original d dimensions and most helpful to classification/regression
    - The combination can be ***linear*** or ***nonlinear***
- Can be ***supervised*** or ***unsupervised*** depending on whether or not the labels $\boldsymbol{r}^{(t)}$ are used

Caution

- Is dimension reduction a preferred step in data preprocessing?

Caution

- Is dimension reduction a preferred step in data preprocessing? Generally, yes
- But if your classifier/regressor make certain assumptions over the data, you should perform dimension reduction ***only if those assumptions are still valid after the reduction***
  - Luckily, if we assume that the data are normally distributed within each class, then after linear dimension reduction, the data are still normal

Outline

Sequential Forward Selection

- Denote F the set of selected inputs
- In ***sequential forward selection***, we start with no input $F = \emptyset$, and add inputs one by one
- At each step, we add x_j to F if x_j decreases the error most
  - Specifically, we add x_j to F if

$$j = \arg_i \min err(F \cup x_i)$$

and

$$err(F \cup x_j) < err(F) - \varepsilon,$$

where $err(F)$ is the error of predictions (on the validation set) made by a classifier/regressor considering only the inputs in F and ε is a user-defined threshold

- We stop adding inputs if there is no decrease in error, or the decrease in error is too small

Sequential Backward Selection

- In ***sequential backward selection***, we start with F containing all inputs, and remove inputs one by one
- At each step, we remove x_j from F if x_j increase the error least
  - Specifically, we remove x_j from F if

$$j = \arg_i \min err(F - x_i) \text{ and } err(F - x_j) < err(F) + \varepsilon$$

- In either direction, need to perform the classification/regression $d + (d-1) + \cdots + (d-k) = O(d^2)$ times
  - But the training process in the backward direction is more costly, as F contains more inputs
  - Forward direction is preferable if k is much smaller than d

Are We Satisfied?

- The above algorithms are local search procedures and do not guarantee F to be optimal
- In some applications such as face recognition (where $\boldsymbol{x}^{(t)}$ are images and the inputs are pixels), feature selection is not a good method for dimensionality reduction
  - Individual pixels do not carry much discriminative information; it is the combinations of pixels that carry face identities

Outline

Principal Component Analysis (1/2)

- In ***Principal Component Analysis (PCA)***, we want to find k vectors $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k \in \mathbb{R}^d$, named ***principal components***, such that after projecting each instance $\boldsymbol{x}^{(t)}$ onto directions along $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ and obtaining $\boldsymbol{z}^{(t)} = \left[\begin{array}{ccc} \boldsymbol{w}_1 & , \cdots & , \boldsymbol{w}_k \end{array} \right]^\top \boldsymbol{x}^{(t)} \in \mathbb{R}^k$, the new data collection $\{\boldsymbol{z}^{(t)}\}_{t=1}^N$ helps the classifier/regressor most
- Let $k = 1$, how to find $\boldsymbol{w}_1$?

Principal Component Analysis (1/2)

- In ***Principal Component Analysis (PCA)***, we want to find k vectors $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k \in \mathbb{R}^d$, named ***principal components***, such that after projecting each instance $\boldsymbol{x}^{(t)}$ onto directions along $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ and obtaining $\boldsymbol{z}^{(t)} = \begin{bmatrix} \boldsymbol{w}_1 & , \cdots & , \boldsymbol{w}_k \end{bmatrix}^\top \boldsymbol{x}^{(t)} \in \mathbb{R}^k$, the new data collection $\{\boldsymbol{z}^{(t)}\}_{t=1}^N$ helps the classifier/regressor most
- Let $k = 1$, how to find $\boldsymbol{w}_1$?
  - Of course we won't pick $\boldsymbol{w}_1 = \boldsymbol{0}$, because it makes $z^{(t)} = \boldsymbol{w}_1^\top \boldsymbol{x}^{(t)} = 0$ for all t and undistinguishable
  - It is plausible to pick $\boldsymbol{w}_1$ such that $z^{(t)}$ is most spread out; that is, let $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}$ be i.i.d. samples drawn from an (unknown) population $\boldsymbol{x}$, we can pick $\boldsymbol{w}_1$ which maximizes $Var(\boldsymbol{z}) = Var(\boldsymbol{w}_1^\top \boldsymbol{x}) = \boldsymbol{w}_1^\top \boldsymbol{\Sigma} \boldsymbol{w}_1$

Principal Component Analysis (2/2)

- How to obtain the covairant matrix $\mathbf{\Sigma}$?

Principal Component Analysis (2/2)

- How to obtain the covairant matrix Σ?
  - Recall that $\Sigma_x = E\left[\boldsymbol{x}\boldsymbol{x}^\top\right] - \mu_x \mu_x^\top$
  - Assuming that $\boldsymbol{x}$ is centered (e.g., z-normalized), we have an estimate $\boldsymbol{S} = \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{x}^{(t)}\boldsymbol{x}^{(t)\top}$
- Given arbitrary k, PCA picks $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ such that
  - $Var(\boldsymbol{z}_i) = \boldsymbol{w}_i^\top \boldsymbol{S}\boldsymbol{w}_i$ for $i = 1, \cdots, k$ are the highest
    - $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ are orthogonal to each other so each of them helps the classifier/regressor independently
    - $\|\boldsymbol{w}_i\| = 1$ for $i = 1, \cdots, k$ so the direction of $\boldsymbol{w}_i$ is the only factor that affects $\boldsymbol{w}_i^\top \boldsymbol{S}\boldsymbol{w}_i$ (it is maximized not because we pick a large $\boldsymbol{w}_i$)

Solving $w_1, \cdots, w_k$ (1/3)

- To get $\boldsymbol{w}_1$, we need to solve the problem $\arg_{\boldsymbol{w}_1} \max \boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1$ subject to $\boldsymbol{w}_1^\top \boldsymbol{w}_1 = 1$
  - $\boldsymbol{w}_1$ is a stationary point of the Lagrangian: $\boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1 - \alpha(\boldsymbol{w}_1^\top \boldsymbol{w}_1 - 1)$
  - Taking the partial derivatives with respect to $\boldsymbol{w}_1$ and α we have $\begin{cases} 2\boldsymbol{S}\boldsymbol{w}_1 - 2\alpha\boldsymbol{w}_1 = \boldsymbol{0} \\ \boldsymbol{w}_1^\top \boldsymbol{w}_1 - 1 = 0 \end{cases}$, implying $\boldsymbol{S}\boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$ and $\boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1^\top \boldsymbol{w}_1 = \alpha$
  - The candidates of $\boldsymbol{w}_1$ and α are eigenvectors and eigenvalues of $\boldsymbol{\Sigma}$ respectively
  - Since we want to maximize $\boldsymbol{w}_1^\top \boldsymbol{S} \boldsymbol{w}_1 = \alpha$, $\boldsymbol{w}_1$ is the eigenvector corresponding to the largest eigenvalue λ_1

Solving $w_1, \cdots, w_k$ (2/3)

- To get $\boldsymbol{w}_2$, we solve the problem $\arg_{\boldsymbol{w}_2} \max \boldsymbol{w}_2^\top \boldsymbol{S} \boldsymbol{w}_2$ subject to $\boldsymbol{w}_2^\top \boldsymbol{w}_2 = 1$ and $\boldsymbol{w}_2^\top \boldsymbol{w}_1 = 0$
  - $\boldsymbol{w}_2$ is a stationary point of the Lagrangian: $\boldsymbol{w}_2^\top \boldsymbol{S} \boldsymbol{w}_2 - \alpha(\boldsymbol{w}_2^\top \boldsymbol{w}_2 - 1) - \beta \boldsymbol{w}_2^\top \boldsymbol{w}_1$
  - Taking the partial derivatives with respect to $\boldsymbol{w}_2$, α, and β we have
    $$\begin{cases} 2\boldsymbol{S}\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_2 - \beta\boldsymbol{w}_1 = \boldsymbol{0} \\ \boldsymbol{w}_2^\top \boldsymbol{w}_2 - 1 = 0 \\ \boldsymbol{w}_2^\top \boldsymbol{w}_1 = 0 \end{cases}$$
    - This implies $\boldsymbol{S}\boldsymbol{w}_2 = \alpha \boldsymbol{w}_2$, as $0 = \boldsymbol{w}_1^\top \boldsymbol{0} = 2\boldsymbol{w}_1^\top \boldsymbol{S}\boldsymbol{w}_2 - 2\alpha \boldsymbol{w}_1^\top \boldsymbol{w}_2 - \beta \boldsymbol{w}_1^\top \boldsymbol{w}_1 = 2\boldsymbol{w}_2^\top \boldsymbol{S}\boldsymbol{w}_1 - \beta = 2\lambda_1 \boldsymbol{w}_2^\top \boldsymbol{w}_1 - \beta = -\beta$, hence $2\boldsymbol{S}\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_2 = \boldsymbol{0}$
    - Additionally, $\boldsymbol{w}_2^\top \boldsymbol{S}\boldsymbol{w}_2 = \alpha$, as $0 = \boldsymbol{w}_2^\top \boldsymbol{0} = 2\boldsymbol{w}_2^\top \boldsymbol{S}\boldsymbol{w}_2 - 2\alpha \boldsymbol{w}_2^\top \boldsymbol{w}_2 - \beta \boldsymbol{w}_2^\top \boldsymbol{w}_1 = 2\boldsymbol{w}_2^\top \boldsymbol{S}\boldsymbol{w}_2 - 2\alpha$
  - The candidates of $\boldsymbol{w}_2$ are eigenvectors of $\boldsymbol{S}$ orthogonal to $\boldsymbol{w}_1$
  - Since we want to maximize $\boldsymbol{w}_2^\top \boldsymbol{S}\boldsymbol{w}_2 = \alpha$, $\boldsymbol{w}_2$ is the eigenvector corresponding to the second largest eigenvalue λ_2
    - Recall that the eigenvectors of a symmetric matrix ($\boldsymbol{S}$) are orthogonal

Solving $w_1, \cdots, w_k$ (3/3)

- From the above, we can see that $\boldsymbol{w}_i$ are the ith largest eigenvector of $\boldsymbol{S}$

  We can reach the same conclusion by Rayleigh's quotient [Proof]
- Define $z = \boldsymbol{W}^\top \boldsymbol{x}$, where $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k]$
  - If $\boldsymbol{x}$ are not normalized, we can center $\boldsymbol{z}$ around the origin by letting $\boldsymbol{z} = \boldsymbol{W}^\top (\boldsymbol{x} - \boldsymbol{m})$, where $\boldsymbol{m}$ is the sample mean of $\boldsymbol{x}$
  - In addition, we can make z_i have the unit variance by dividing itself by $\sqrt{\lambda_i}$
- $z^{(t)}$ are i.i.d. samples of z

Effects of PCA

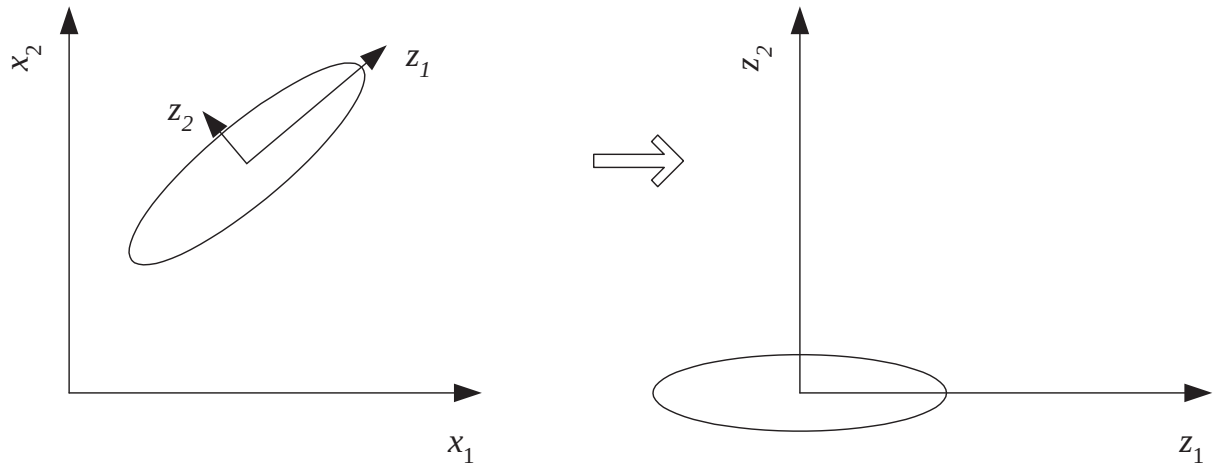
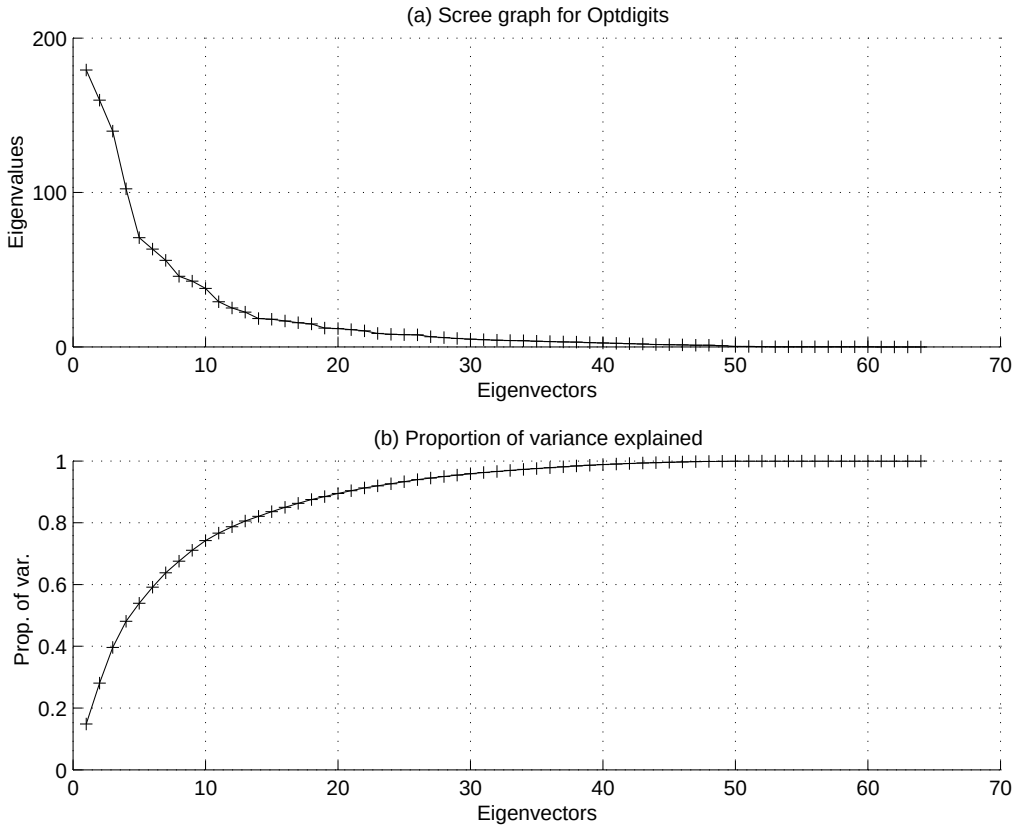


Figure : PCA centers the instances and rotates the axes to line up with the directions of the highest variance. With these new axes, the covariance matrix $\boldsymbol{\Sigma_z} = \boldsymbol{W}^\top \boldsymbol{S} \boldsymbol{W} \in \mathbb{R}^{k \times k}$ is always diagonal, making the naive Bayes' classifiers feasible.

The Scree Graph

- How do we decide a proper k?
  - Generally, we want to pick k such that the ***proportion of variance*** $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$ is more than 90%
- We seek for the "elbow" in the ***scree graph***:



Are We Satisfied? (1/3)

- In image processing applications (where $\boldsymbol{x}^{(t)}$ are images), $\boldsymbol{w}_i$ themselves can also be displayed as images and be seen as templates for important features
  - In these cases, $\boldsymbol{w}_i$ have a special name ***eigenfaces***
- When should we use PCA?

Are We Satisfied? (1/3)

- In image processing applications (where $\boldsymbol{x}^{(t)}$ are images), $\boldsymbol{w}_i$ themselves can also be displayed as images and be seen as templates for important features
  - In these cases, $\boldsymbol{w}_i$ have a special name ***eigenfaces***
- When should we use PCA?
  - PCA is helpful only when we have a small number of large eigenvalues
  - Or, when the original inputs of $\boldsymbol{x}$ are highly correlated (so the contours are stretched)
- Note the variance of each input of $\boldsymbol{x}$ may be very high that affects the directions of principal components more than the correlations between inputs do
  - Generally, we perform z-normalization before applying PCA

Are We Satisfied? (2/3)

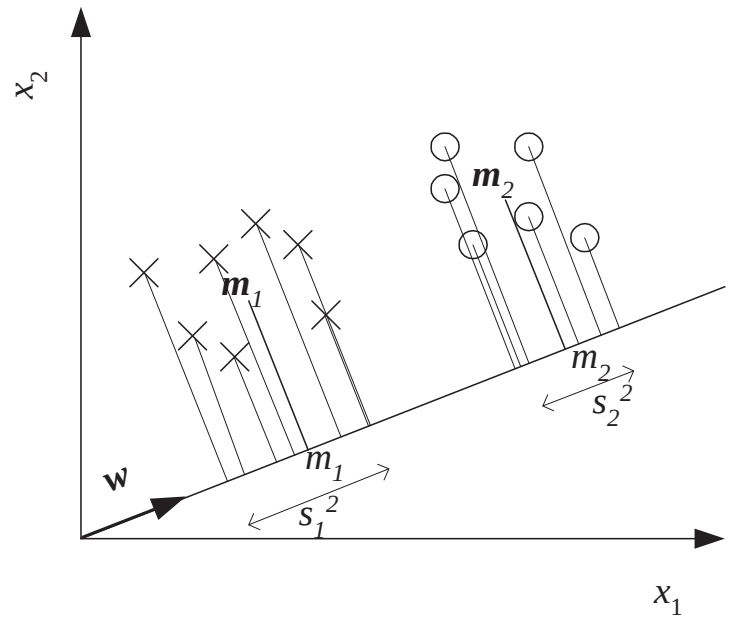
- PCA is a one-group procedure
  - Fine with the regression
  - But in the case of classification, there are multiple groups
- The Karhunen-Loeve expansion:
  - Instead of using the covariance matrix of the whole examples, we can estimate separate covariance matrices for individual classes
  - Take their average (weighted by the estimated priors) as the covariance matrix, and use its eigenvectors

Outline

Linear Discriminant Analysis (1/3)

- Given a training set $\mathcal{X} = \{\boldsymbol{x}^{(t)}, \boldsymbol{r}^{(t)}\}_{t=1}^{N}$, in ***Linear Discriminant Analysis (LDA)***, we want to find k vectors $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k \in \mathbb{R}^d$, such that after projecting each instance $\boldsymbol{x}^{(t)}$ onto directions along $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ and obtaining $\boldsymbol{z}^{(t)} = \boldsymbol{W}^\top \boldsymbol{x}^{(t)} \in \mathbb{R}^k$, where $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k]$, the new data collection $\{\boldsymbol{z}^{(t)}\}_{t=1}^{N}$ has the properties:

- Examples in different classes are as separate as possible
- Examples of the same class are as close as possible



Linear Discriminant Analysis (2/3)

- Let $\boldsymbol{m}_i = \frac{1}{N_i} \sum_{t=1}^{N} r_i^{(t)} \boldsymbol{x}^{(t)}$ and $\boldsymbol{S}_i = \frac{1}{N_i - 1} \sum_{t=1}^{N} r_i^{(t)} (\boldsymbol{x}^{(t)} - \boldsymbol{m}_i)(\boldsymbol{x}^{(t)} - \boldsymbol{m}_i)^\top$, where $N_i = \sum_{t=1}^{N} r_i^{(t)}$, be the estimated mean and variance of examples in class i respectively
- We can measure the total within-class separation by $\boldsymbol{S}_{within} = \sum_{i=1}^{K} N_i \boldsymbol{S}_i$
- Denote $\boldsymbol{m} = \frac{1}{K} \sum_{i=1}^{K} \boldsymbol{m}_i$, we can also measure the separation between classes by $\boldsymbol{S}_{between} = \sum_{i=1}^{K} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^\top$
- For each $\boldsymbol{w}_j$, the mean and variance of the projections $z^{(t)}$ are $m_i = \frac{1}{N_i} \sum_{t=1}^{N} r_i^{(t)} z_j^{(t)} = \frac{1}{N_i} \sum_{t=1}^{N} r_i^{(t)} \boldsymbol{w}_j^\top \boldsymbol{x}^{(t)} = \boldsymbol{w}_j^\top \left(\frac{1}{N_i} \sum_{t=1}^{N} r_i^{(t)} \boldsymbol{x}^{(t)} \right) = \boldsymbol{w}_j^\top \boldsymbol{m}_i$ and $s_i^2 = \frac{1}{N_i - 1} \sum_{t=1}^{N} r_i^{(t)} (z_j^{(t)} - \boldsymbol{w}_j^\top \boldsymbol{m}_i)(z_j^{(t)} - \boldsymbol{w}_j^\top \boldsymbol{m}_i)^\top = \frac{1}{N_i - 1} \sum_{t=1}^{N} r_i^{(t)} (\boldsymbol{w}_j^\top \boldsymbol{x}^{(t)} - \boldsymbol{w}_j^\top \boldsymbol{m}_i)(\boldsymbol{w}_j^\top \boldsymbol{x}^{(t)} - \boldsymbol{w}_j^\top \boldsymbol{m}_i)^\top = \frac{1}{N_i - 1} \sum_{t=1}^{N} r_i^{(t)} \boldsymbol{w}_j^\top (\boldsymbol{x}^{(t)} - \boldsymbol{m}_i)(\boldsymbol{x}^{(t)} - \boldsymbol{m}_i)^\top \boldsymbol{w}_j = \boldsymbol{w}_j^\top \boldsymbol{S}_i \boldsymbol{w}_j$ respectively
- The within- and between-class separation after projection becomes $\boldsymbol{w}_j^\top \boldsymbol{S}_{within} \boldsymbol{w}_j$ and $\boldsymbol{w}_j^\top \boldsymbol{S}_{between} \boldsymbol{w}_j$ respectively

Linear Discriminant Analysis (3/3)

- LDA picks $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ such that
  - The ***Fisher's linear discriminant*** $J(\boldsymbol{w}_j) = \frac{\boldsymbol{w}_j^\top \boldsymbol{S}_{between} \boldsymbol{w}_j}{\boldsymbol{w}_j^\top \boldsymbol{S}_{within} \boldsymbol{w}_j}$ are the highest for $j = 1, \cdots, k$
  - $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ are orthogonal to each other
- We don't care $\|\boldsymbol{w}_j\|$ now since we are maximizing the ratio
- Unfortunately, the solution to the above objective is not unique since if $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_k$ are the solution, so do $c_1 \boldsymbol{w}_1, \cdots, c_k \boldsymbol{w}_k, \forall c_j \in \mathbb{R}$
- We can instead maximize $\boldsymbol{w}_j^\top \boldsymbol{S}_{between} \boldsymbol{w}_j$ subject to an additional constrain $\boldsymbol{w}_j^\top \boldsymbol{S}_{within} \boldsymbol{w}_j = 1$

Solving $w_1, \cdots, w_k$

- Lagrangian for $\boldsymbol{w}_1$: $\arg_{\boldsymbol{w}_1} \max \boldsymbol{w}_1^\top \boldsymbol{S}_{between} \boldsymbol{w}_1 - \alpha(\boldsymbol{w}_1^\top \boldsymbol{S}_{within} \boldsymbol{w}_1 - 1)$
  - Taking the partial derivatives with respect to $\boldsymbol{w}_1$ and α and setting them to zero we have $\begin{cases} 2\boldsymbol{S}_{between}\boldsymbol{w}_1 - 2\alpha \boldsymbol{S}_{within}\boldsymbol{w}_1 = \boldsymbol{0} \\ \boldsymbol{w}_1^\top \boldsymbol{S}_{within}\boldsymbol{w}_1 - 1 = 0 \end{cases}$, leading to $(\boldsymbol{S}_{within})^{-1}\boldsymbol{S}_{between}\boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$ and $\boldsymbol{w}_1^\top \boldsymbol{S}_{between}\boldsymbol{w}_1 = \alpha$
  - The candidates of $\boldsymbol{w}_1$ and α are eigenvectors and eigenvalues of $(\boldsymbol{S}_{within})^{-1}\boldsymbol{S}_{between}$ respectively
  - Since we want to maximize $\boldsymbol{w}_1^\top \boldsymbol{S}_{between}\boldsymbol{w}_1 = \alpha$, $\boldsymbol{w}_1$ is the eigenvector corresponding to the largest eigenvalue λ_1
- Similarly, $\boldsymbol{w}_j$ is the jth largest eigenvector of $(\boldsymbol{S}_{within})^{-1}\boldsymbol{S}_{between}$
- We can reach the same conclusion using the Rayleigh's quotient [Proof: Observe that $\frac{\boldsymbol{w}_j^\top \boldsymbol{S}_{between}\boldsymbol{w}_j}{\boldsymbol{w}_j^\top \boldsymbol{S}_{within}\boldsymbol{w}_j} = \frac{\boldsymbol{w}_j^\top \boldsymbol{S}_{between}\boldsymbol{w}_j}{(\boldsymbol{U}^\top \boldsymbol{w}_j)^\top \boldsymbol{U}^\top \boldsymbol{w}_j} = \frac{(\boldsymbol{U}^\top \boldsymbol{w}_j)^\top \boldsymbol{U}^{-1}\boldsymbol{S}_{between}(\boldsymbol{U}^\top)^{-1}(\boldsymbol{U}^\top \boldsymbol{w}_j)}{(\boldsymbol{U}^\top \boldsymbol{w}_j)^\top \boldsymbol{U}^\top \boldsymbol{w}_j}$, where $\boldsymbol{S}_{within} = \boldsymbol{U}\boldsymbol{U}^\top$]

Deciding k

- Note $\boldsymbol{S}_{between}$ is the sum of K matrices, namely $(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^\top$, of rank 1
  - $(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^\top \overset{c}{\backsim} [\boldsymbol{m}_i - \boldsymbol{m}, \boldsymbol{0}, \cdots, \boldsymbol{0}]$
- Given K columns $\{\boldsymbol{m}_1 - \boldsymbol{m}, \cdots, \boldsymbol{m}_K - \boldsymbol{m}\}$, only $K-1$ of them are linearly independent
  - Since $\boldsymbol{m} = \frac{1}{K}\sum_{i=1}^{K} \boldsymbol{m}_i$, we can express $\boldsymbol{m}_K - \boldsymbol{m}$ by the linear combination of $\boldsymbol{m}_1 - \boldsymbol{m}, \cdots, \boldsymbol{m}_{K-1} - \boldsymbol{m}$
- The maximum rank of $\boldsymbol{S}_{between}$ is $K-1$ and we can pick at most $K-1$ eigenvectors (i.e., $k = K-1$)

Are We Satisfied?

- Any classifier can be used after LDA
- To be able to apply LDA, $\boldsymbol{S}_{within}$ must be invertible
  - If not, we can apply PCA first to get rid of the singularity
- When should we use LDA?

Are We Satisfied?

- Any classifier can be used after LDA
- To be able to apply LDA, $\boldsymbol{S}_{within}$ must be invertible
  - If not, we can apply PCA first to get rid of the singularity
- When should we use LDA?
  - LDA works best if instances in different class are distributed in groups (e.g., normal)

PCA vs. LDA

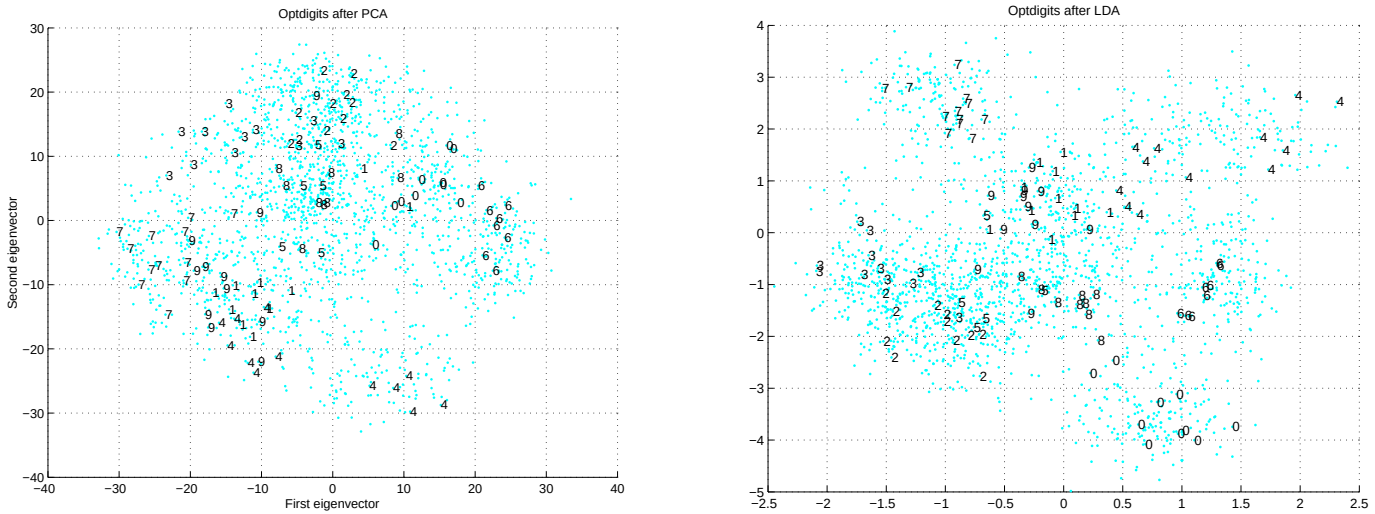


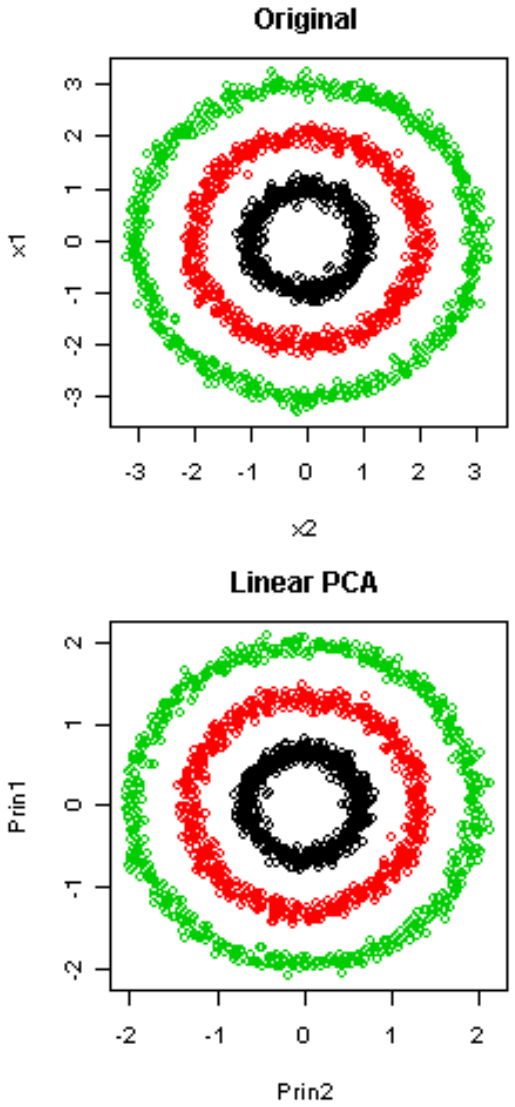
Figure : The distribution of $z^{(t)}$ found by PCA (left) and LDA (right) respectively. Instances are plotted in the space of the first two attributes (out of nine). LDA, as expected, leads to a better separation between classes.

- Identify situations where PCA and LDA will find the same and totally different (orthogonal) directions [Homework]

Outline

Why Nonlinear Mappings?

- Both PCA and LDA map examples $\boldsymbol{x}^{(t)}$ to a low dimensional space (through projection $\boldsymbol{W}$)
  - We assume that the d-dimensional input space has a linear relationship with the low dimensional space
- Sometimes, this linear mapping does not help



Kernel PCA: Basic Idea

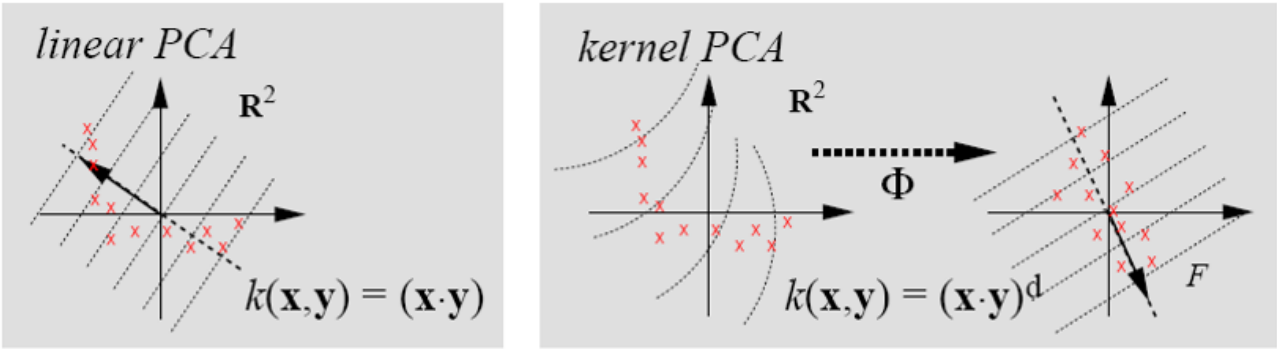


Fig. 1. Basic idea of kernel PCA: by using a nonlinear kernel function k instead of the standard dot product, we implicitly perform PCA in a possibly high–dimensional space F which is nonlinearly related to input space. The dotted lines are contour lines of constant feature value.

PCA Revisited

- Find the eigenvectors $\boldsymbol{w}^{(1)}, \cdots, \boldsymbol{w}^{(k)}$ of the matrix $\boldsymbol{S} = \frac{1}{N} \sum_{t=1}^{N} \boldsymbol{x}^{(t)} \boldsymbol{x}^{(t)\top}$ corresponding to the largest eigenvalues
- Each eigenvector $\boldsymbol{w}^{(i)}$ lies in the span of $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}$
  - As $\boldsymbol{S}\boldsymbol{w}^{(i)} = \lambda^{(i)} \boldsymbol{w}^{(i)} \Rightarrow \boldsymbol{w}^{(i)} = \frac{1}{\lambda^{(i)}} \sum_{t=1}^{N} \boldsymbol{x}^{(t)} (\boldsymbol{x}^{(t)\top} \boldsymbol{w}^{(i)}) = \sum_{t=1}^{N} \alpha_t^{(i)} \boldsymbol{x}^{(t)}$
- This holds after the lifting: $\boldsymbol{w}^{(i)} = \sum_{t=1}^{N} \alpha_t^{(i)} \Phi(\boldsymbol{x}^{(t)})$
  - We can instead solve N variables in $\boldsymbol{\alpha}^{(i)} \in \mathbb{R}^N$ for each $\boldsymbol{w}^{(i)}$
- We don't need to compute $\boldsymbol{w}^{(i)}$ explicitly to obtain $\boldsymbol{z}^{(t)} = \left[\boldsymbol{w}^{(1)\top} \Phi(\boldsymbol{x}^{(t)}), \cdots, \boldsymbol{w}^{(k)\top} \Phi(\boldsymbol{x}^{(t)})\right]^\top$
  - $z_i^{(t)} = \boldsymbol{w}^{(i)\top} \Phi(\boldsymbol{x}^{(t)}) = \sum_{s=1}^{N} \alpha_s^{(i)} k(\boldsymbol{x}^{(s)}, \boldsymbol{x}^{(t)})$

Solving $\alpha^{(1)}, \cdots, \alpha^{(k)}$ (1/3)

- Let $\boldsymbol{S} = \frac{1}{N} \sum_{t=1}^{N} \Phi(\boldsymbol{x}^{(t)}) \Phi(\boldsymbol{x}^{(t)})^{\top}$
- Then we can write $\boldsymbol{w}^{(i)}$ as $\boldsymbol{w}^{(i)} = \sum_{t=1}^{N} \alpha_t^{(i)} \Phi(\boldsymbol{x}^{(t)})$
- Also we have $\boldsymbol{S}\boldsymbol{w}^{(i)} = \lambda^{(i)} \boldsymbol{w}^{(i)}$, implying that

$$\frac{1}{N} \sum_{t=1}^{N} \Phi(\boldsymbol{x}^{(t)}) \Phi(\boldsymbol{x}^{(t)})^{\top} \sum_{s=1}^{N} \alpha_s^{(i)} \Phi(\boldsymbol{x}^{(s)}) = \lambda^{(i)} \sum_{s=1}^{N} \alpha_s^{(i)} \Phi(\boldsymbol{x}^{(s)})$$

Solving $\alpha^{(1)}, \cdots, \alpha^{(k)}$ (2/3)

$$\frac{1}{N}\sum_{t=1}^{N}\Phi(\boldsymbol{x}^{(t)})\Phi(\boldsymbol{x}^{(t)})^{\top}\sum_{s=1}^{N}\alpha_s^{(i)}\Phi(\boldsymbol{x}^{(s)}) = \lambda^{(i)}\sum_{s=1}^{N}\alpha_s^{(i)}\Phi(\boldsymbol{x}^{(s)})$$

$$\Rightarrow \sum_{t=1}^{N}\sum_{s=1}^{N}\Phi(\boldsymbol{x}^{(t)})(\Phi(\boldsymbol{x}^{(t)})^{\top}\Phi(\boldsymbol{x}^{(i)}))\alpha_s^{(i)} = N\lambda^{(i)}\sum_{s=1}^{N}\alpha_s^{(i)}\Phi(\boldsymbol{x}^{(s)})$$

$$\Rightarrow \sum_{t=1}^{N}\sum_{s=1}^{N}\Phi(\boldsymbol{x}^{(l)})^{\top}\Phi(\boldsymbol{x}^{(t)})(\Phi(\boldsymbol{x}^{(t)})^{\top}\Phi(\boldsymbol{x}^{(i)}))\alpha_s^{(i)}$$

$$= N\lambda^{(i)}\sum_{s=1}^{N}\alpha_s^{(i)}\Phi(\boldsymbol{x}^{(l)})^{\top}\Phi(\boldsymbol{x}^{(i)}), \text{ for } l = 1, \cdots, N$$

$$\Rightarrow \boldsymbol{K}^2\boldsymbol{\alpha}^{(i)} = N\lambda^{(i)}\boldsymbol{K}\boldsymbol{\alpha}^{(i)}$$

$$\Rightarrow \boldsymbol{K}\boldsymbol{\alpha}^{(i)} = N\lambda^{(i)}\boldsymbol{\alpha}^{(i)} = \widetilde{\lambda}^{(i)}\boldsymbol{\alpha}^{(i)}$$

- Maximizing $\lambda^{(1)}, \cdots, \lambda^{(k)}$ (eigenvalues of $\boldsymbol{S}$) amounts to maximizing $\widetilde{\lambda}^{(1)}, \cdots, \widetilde{\lambda}^{(k)}$ (eigenvalues of $\boldsymbol{K}$)
- $\boldsymbol{\alpha}^{(i)}$'s are the eigenvectors corresponding the maximal eigenvalues of $\boldsymbol{K}$

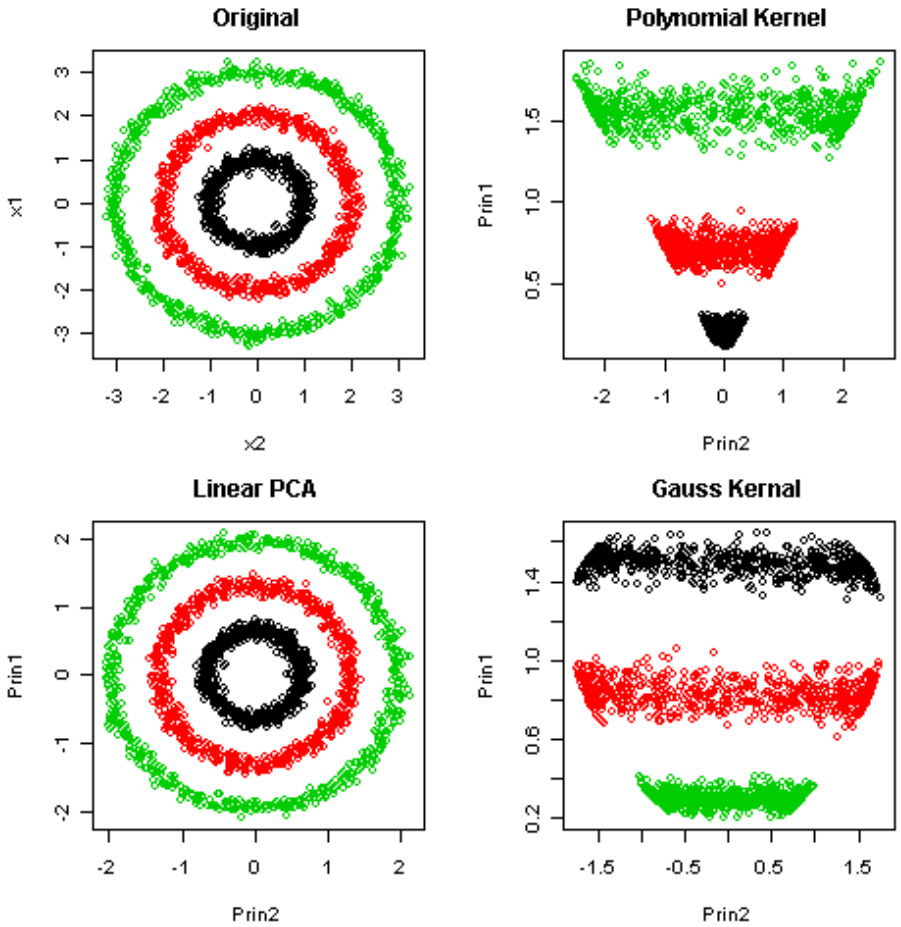
Solving $\alpha^{(1)}, \cdots, \alpha^{(k)}$ (3/3)

- Note that $\boldsymbol{w}^{(i)}$ is normalized, i.e., $\|\boldsymbol{w}^{(i)}\| = 1$
  - Each $\boldsymbol{\alpha}^{(i)}$ needs to be scaled properly
- $\|\boldsymbol{w}^{(i)}\|^2 = (\sum_{t=1}^{N} \alpha_t^{(i)} \Phi(\boldsymbol{x}^{(t)}))^\top (\sum_{t=1}^{N} \alpha_t^{(i)} \Phi(\boldsymbol{x}^{(t)})) = \boldsymbol{\alpha}^{(i)\top} \boldsymbol{K} \boldsymbol{\alpha}^{(i)} = 1 \Rightarrow \boldsymbol{\alpha}^{(i)\top} \boldsymbol{\alpha}^{(i)} = 1/N\lambda^{(i)} = 1/\widetilde{\lambda}^{(i)}$
  - $\boldsymbol{\alpha}^{(i)} \leftarrow \frac{\boldsymbol{\alpha}^{(i)}}{\sqrt{\widetilde{\lambda}^{(i)}} \|\boldsymbol{\alpha}^{(i)}\|}$
- To project the $z^{(t)} = \boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})$
- Similarly, $\boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_k$ are the (scaled) eigenvectors corresponding to the largest eigenvalues of $\boldsymbol{K}$

Centering $\Phi(\boldsymbol{x}^{(t)})$'s

- Note that by letting $\boldsymbol{S} = \frac{1}{N}\sum_{t=1}^{N}\Phi(\boldsymbol{x}^{(t)})\Phi(\boldsymbol{x}^{(t)})^{\top}$, we assume that $\Phi(\boldsymbol{x}^{(t)})$'s are centered
  - Recall that $\boldsymbol{\Sigma}_{\Phi(x)} = E\left[\Phi(\boldsymbol{x})\Phi(\boldsymbol{x})^{\top}\right] - \boldsymbol{\mu}_{\Phi(x)}\boldsymbol{\mu}_{\Phi(x)}^{\top}$
  - In linear PCA, we can simply center $\Phi(\boldsymbol{x}^{(t)})$'s by a preprocessing step
- Given an arbitrary kernel function $k(\cdot)$, there is ***no*** guarantee that $\Phi(\boldsymbol{x}^{(t)})$'s will be centered in the lifted space
  - Preprocessing is infeasible
  - The model itself needs to be extended to accept uncentered instances in the lifted space. How? [Homework]

PCA vs. Kernel PCA



Outline

Manifold-Preserved Nonlinear Mapping

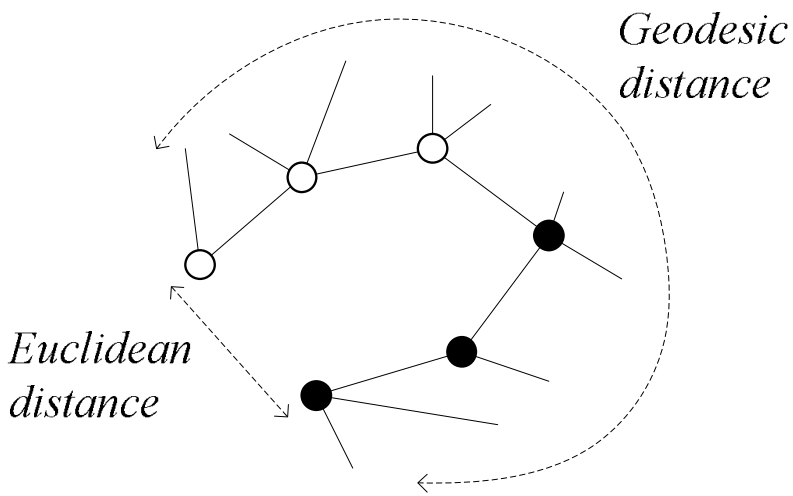
- Kernel PCA does not assumed a particular nonlinear mapping
  - Performance largely depends on the selection of kernel function
- In some applications, assuming some particular form of nonlinear mapping will be more helpful
- Consider an example where $\boldsymbol{x}^{(t)}$ are face photos of size 100×100 pixels
  - A series of one's photos token from different angles forms a trajectory in the 10000-dimensional space
  - The collection of people's photos defines a ***manifold*** in the 10000-dimensional space
  - If we map the 10000-dimensional space linearly to a low dimensional space, the structure of the manifold may not be preserved

The Basic Idea

- Nonlinear methods rebuild the manifold in a low dimensional space
  1. Treat the manifold as a space
  2. Measure the relations/constrains between $\boldsymbol{x}^{(t)}$ in that space (which reflect the structure of the manifold)
  3. Find instances $\boldsymbol{z}^{(t)}$ in a low dimensional space that obey the constrains most (so preserve the structure)
- The final $\boldsymbol{z}^{(t)}$ may not relate to $\boldsymbol{x}^{(t)}$ linearly
- Two popular algorithms:
  - Isometric feature mapping (Isomap)
  - Locally linear embedding (LLE)
- It might be a good idea to review the topology now

Isomap

- The ***Isometric feature mapping (Isomap)*** measures the ***geodesic distances*** between examples in step 2; and then in step 3, find instances $z^{(t)}$ in a low dimensional space with mutual distances as close as the geodesic distances as possible
  - Geodesic distance is the distance along the manifold that the data lies in, as opposed to the Euclidean distance in the input space



Measuring the Geodesic Distances

- Recall that a manifold resembles a Euclidean space at a small enough scale
  - One topological property of the Euclidean space is that points are connected
- A point $\boldsymbol{x}^{(r)}$ and another $\boldsymbol{x}^{(s)}$ are directly connected if $\boldsymbol{x}^{(s)}$ lies in the same neighborhood with $\boldsymbol{x}^{(r)}$, and their geodesic distance can simply be the Euclidean distance
  - Any other choice?

Measuring the Geodesic Distances

- Recall that a manifold resembles a Euclidean space at a small enough scale
  - One topological property of the Euclidean space is that points are connected
- A point $\boldsymbol{x}^{(r)}$ and another $\boldsymbol{x}^{(s)}$ are directly connected if $\boldsymbol{x}^{(s)}$ lies in the same neighborhood with $\boldsymbol{x}^{(r)}$, and their geodesic distance can simply be the Euclidean distance
  - Any other choice? Mahalanobis distance, at the cost of computing the covariance matrix $\boldsymbol{\Sigma}$
- How about points not in the same neighborhood?

Measuring the Geodesic Distances

- Recall that a manifold resembles a Euclidean space at a small enough scale
  - One topological property of the Euclidean space is that points are connected
- A point $\boldsymbol{x}^{(r)}$ and another $\boldsymbol{x}^{(s)}$ are directly connected if $\boldsymbol{x}^{(s)}$ lies in the same neighborhood with $\boldsymbol{x}^{(r)}$, and their geodesic distance can simply be the Euclidean distance
  - Any other choice? Mahalanobis distance, at the cost of computing the covariance matrix $\boldsymbol{\Sigma}$
- How about points not in the same neighborhood?
  - In an atlas, neighborhoods are overlapped
  - The geodesic distance between two points that are not in the same neighborhood can be calculated by the length of their shortest path

Identifying the Neighborhood

- How identify to the neighboring points of $\boldsymbol{x}^{(r)}$?

Identifying the Neighborhood

- How identify to the neighboring points of $\boldsymbol{x}^{(r)}$?
- We assume that $\boldsymbol{x}^{(r)}$ and $\boldsymbol{x}^{(s)}$ lie in the same neighborhood if:
  - $\left\|\boldsymbol{x}^{(r)} - \boldsymbol{x}^{(s)}\right\| < \varepsilon$; or
  - $\boldsymbol{x}^{(s)}$ is one of the n-nearest neighbors of $\boldsymbol{x}^{(r)}$
- User-specific parameters ε and n are usually small, but must be chosen carefully to make sure that the network is still connected

Finding $\{z^{(t)}\}_{t=1}^{N}$ (1/3)

- Let $d_{r,s}$ be the geodesic distance between $x^{(r)}$ and $x^{(s)}$, $1 \leqslant r, s \leqslant N$, now we want to find $z^{(r)}$ and $z^{(s)}$ in a k-dimensional space such that their mutual Euclidean distances are as close to $d_{r,s}$ as possible
- Exact solution (without error) may exist only when k is ***larger*** than d
  - There may be no way to "straighten" the geodesic distances between points in a space of dimension d (or lower)
- The solution $\{z^{(t)}\}_t$ is not unique, as we can shift all $z^{(t)}$ together to get another solution
  - To constrain the solution, we assume that $\sum_{t=1}^{N} z_i^{(t)} = 0$ for $i = 1, \cdots, k$

Finding $\{z^{(t)}\}_{t=1}^{N}$ (2/3)

- Let $\boldsymbol{Z} = \begin{bmatrix} z^{(1)} \\ \vdots \\ z^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times k}$, we want to find the relationship between $\boldsymbol{Z}$ and $d_{r,s}$
- For any $z^{(r)}$ and $z^{(s)}$, we have
  $d_{r,s}^2 = \left\| z^{(r)} - z^{(s)} \right\|^2 = b_{r,r} + b_{s,s} - 2b_{r,s}$, where
  $b_{r,s} = \sum_{i=1}^{k} z_i^{(r)} z_i^{(s)} = \left(z^{(r)} \right)^\top z^{(s)}$
- Therefore, $\boldsymbol{B} = \begin{bmatrix} b_{1,1} & \cdots & b_{1,N} \\ \vdots & \ddots & \vdots \\ b_{N,1} & \cdots & b_{N,N} \end{bmatrix} = \boldsymbol{Z}\boldsymbol{Z}^\top$

Finding $\{z^{(t)}\}_{t=1}^{N}$ (3/3)

- On the other hand, let $T = \sum_{t=1}^{N} b_{t,t} = \sum_t \sum_i (z_i^{(t)})^2$, we get
  - $\sum_r d_{r,s}^2 = T + Nb_{s,s} - 2\sum_r \sum_i z_i^{(r)} z_i^{(s)} = T + Nb_{s,s} - 2\sum_i \left(\sum_r z_i^{(r)}\right) z_i^{(s)} = T + Nb_{s,s}$
  - $\sum_s d_{r,s}^2 = Nb_{r,r} + T$
  - $\sum_r \sum_s d_{r,s}^2 = 2NT$
- Each element in $\boldsymbol{B}$ can be expressed by the geodesic distance by $b_{i,j} = \frac{1}{2}\left(b_{i,i} + b_{j,j} - d_{i,j}^2\right) = \frac{1}{2}\left(\frac{1}{N}\sum_s d_{i,s}^2 + \frac{1}{N}\sum_r d_{r,j}^2 - \frac{1}{N^2}\sum_r \sum_s d_{r,s}^2 - d_{i,j}^2\right)$
- Note $\boldsymbol{B}$ is symmetric and can be written as $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\top$, where the columns of $\boldsymbol{U}$ are eigenvectors
- We find $Z = \boldsymbol{U}\boldsymbol{D}^{1/2}$

Deciding k

- Given $\boldsymbol{Z} \in \mathbb{R}^{N \times k}$, the rank of $\boldsymbol{Z}\boldsymbol{Z}^\top$ is at most k
- In the case that $\boldsymbol{B} \in \mathbb{R}^{N \times N}$ has full rank, we need $k = N$ to obtain the exact solution
- However, for dimension reduction, we want $k < d$ (and N)
- This leads to a ***low rank approximation problem***: given a small k, $k < N$, we want to find a matrix $\widetilde{\boldsymbol{B}}$ such that $\left\| \boldsymbol{B} - \widetilde{\boldsymbol{B}} \right\|_F$ is minimized, subject to $rank(\widetilde{\boldsymbol{B}}) = k$
  - Why Frobenius norm?

Deciding k

- Given $\boldsymbol{Z} \in \mathbb{R}^{N \times k}$, the rank of $\boldsymbol{Z}\boldsymbol{Z}^\top$ is at most k
- In the case that $\boldsymbol{B} \in \mathbb{R}^{N \times N}$ has full rank, we need $k = N$ to obtain the exact solution
- However, for dimension reduction, we want $k < d$ (and N)
- This leads to a ***low rank approximation problem***: given a small k, $k < N$, we want to find a matrix $\widetilde{\boldsymbol{B}}$ such that $\left\|\boldsymbol{B} - \widetilde{\boldsymbol{B}}\right\|_F$ is minimized, subject to $rank(\widetilde{\boldsymbol{B}}) = k$
  - Why Frobenius norm? To minimize the error of the approximated distances
  - Recall that the best approximation is given by $\widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{D}}\widetilde{\boldsymbol{U}}^\top$, where $\widetilde{\boldsymbol{D}} \in \mathbb{R}^{k \times k}$ contains only the k largest eigenvalues and $\widetilde{\boldsymbol{U}} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k] \in \mathbb{R}^{N \times k}$ contains the corresponding eigenvectors
- Finally, we let $Z = \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{D}}^{1/2}$

Are We Satisfied? (1/2)

- Isomap is also a one-group process
  - This is less severe since the "structure" between examples is preserved
  - We can also let $d'_{r,s} = (1-\alpha)d_{r,s} + \alpha c_{r,s}$, where $c_{r,s}$ is the distance between classes $\boldsymbol{x}^{(r)}$ and $\boldsymbol{x}^{(s)}$ belong to, and the parameter α can be tuned using the cross validation
- The major problem of Isomap is that it does not learn a mapping between $\boldsymbol{x}^{(t)}$ and $\boldsymbol{z}^{(t)}$
  - $Z = \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{D}}^{1/2}$ implies that $z_j^{(t)} = \lambda_j u_j^{(t)}$, where $u_j^{(t)}$ is the tth component of the eigenvector $\boldsymbol{u}_j \in \mathbb{R}^{N\times 1}$ in $\widetilde{\boldsymbol{U}}$
  - Given a new instance $\boldsymbol{x}'$, we need to rerun the whole algorithm using the $N+1$ instances to get $\boldsymbol{z}'$

Are We Satisfied? (2/2)

- Solution?

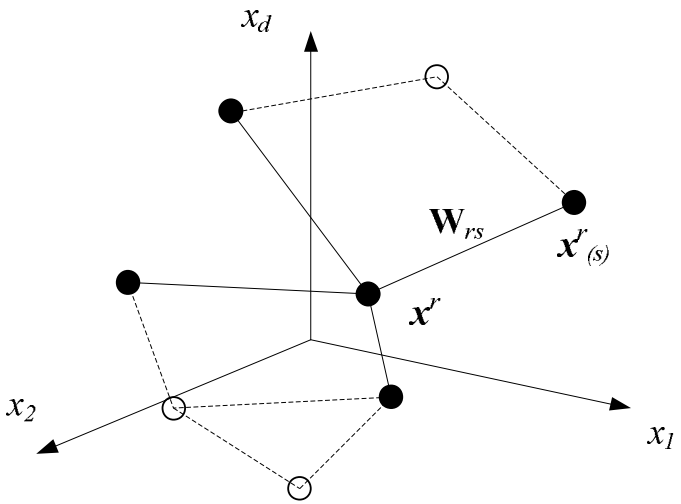
Are We Satisfied? (2/2)

- Solution?
  - Taking the advantage that a manifold is locally linear, we can identify examples $\boldsymbol{x}^{(s)}$ in the same neighborhood as $\boldsymbol{x}$ and calculate $\boldsymbol{w}$ such that $\left\|\boldsymbol{x}-\sum_s w_s \boldsymbol{x}^{(s)}\right\|^2$ is minimized
  - Calculate $\boldsymbol{z}$ by interpolation: $\boldsymbol{z}=\sum_s w_s \boldsymbol{z}^{(s)}$
- The cost is that we need to store the whole set of $\left\{\boldsymbol{x}^{(t)}, \boldsymbol{z}^{(t)}\right\}_{t=1}^{N}$

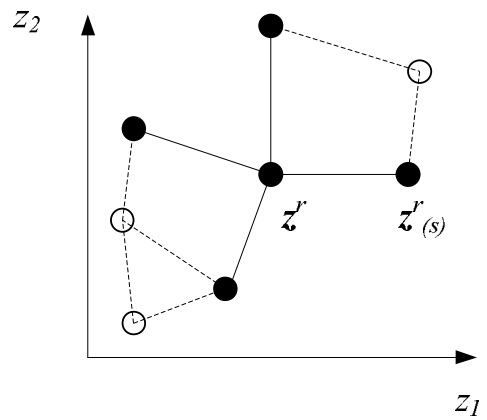
Outline

Locally Linear Embedding (1/2)

- Based on the similar idea above, the ***Locally Linear Embedding (LLE)*** represents each example as a linear combination of its nearby points in step 2; and then in step 3, find instances $z^{(t)}$ in a low dimensional space which preserve the combinations most



x space



z space

Locally Linear Embedding (2/2)

- Step 2: For each example $\boldsymbol{x}^{(r)}$ and all its nearby points $\boldsymbol{x}^{(s)}$, $s \neq r$, find weights $w_{r,s}$ such that $\left\|\boldsymbol{x}^{(r)} - \sum_s w_{r,s}\boldsymbol{x}^{(s)}\right\|^2$ is minimized, subject to $\sum_s w_{r,s} = 1$
  - The constraint ensures that after translating all the points together by some vector $\boldsymbol{c}$, the combination is still valid; i.e., $\left\|(\boldsymbol{x}^{(r)} + \boldsymbol{c}) - \sum_s w_{r,s}(\boldsymbol{x}^{(s)} + \boldsymbol{c})\right\|^2 = \left\|\boldsymbol{x}^{(r)} - \sum_s w_{r,s}\boldsymbol{x}^{(s)}\right\|^2$
- Step 3: Find $\{\boldsymbol{z}^{(t)}\}_{t=1}^N$ such that $\sum_r \left\|\boldsymbol{z}^{(r)} - \sum_s w_{r,s}\boldsymbol{z}^{(s)}\right\|^2$ is minimized, subject to $\frac{1}{N}\sum_t \boldsymbol{z}^{(t)} = \boldsymbol{0}$ $(E[\boldsymbol{z}] = \boldsymbol{0})$ and $\frac{1}{N-1}\sum_t (\boldsymbol{z}^{(t)} - \boldsymbol{0})(\boldsymbol{z}^{(t)} - \boldsymbol{0})^\top = \boldsymbol{I}$ $(Cov(\boldsymbol{z}) = \boldsymbol{I})$
  - The first constraint is similar to that of Isomap and ensures an unique solution
  - The second guarantees that attributes of $\boldsymbol{z}$ a) are uncorrelated; ***b)*** have the same (unit) variance (this is stronger)

Solving $w_{r,s}$

- $w_{r,s}$ can be solved by considering one example $\boldsymbol{x}^{(r)}$ by one
- Let $\boldsymbol{w}^{(r)} = [w_{r,1}, \cdots, w_{r,n}]^\top$, we can rewrite the objective in step 2 as $\left\|\boldsymbol{x}^{(r)} - \sum_s w_{r,s}\boldsymbol{x}^{(s)}\right\|^2 = \left\|\sum_s w_{r,s}\left(\boldsymbol{x}^{(r)} - \boldsymbol{x}^{(s)}\right)\right\|^2 = \boldsymbol{w}^{(r)\top}\boldsymbol{G}^{(r)\top}\boldsymbol{G}^{(r)}\boldsymbol{w}^{(r)}$
  - $\boldsymbol{G}^{(r)} = [\boldsymbol{x}^{(r)} - \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(r)} - \boldsymbol{x}^{(n)}] \in \mathbb{R}^{d\times n}$
  - Subject to $\mathbf{1}^\top \boldsymbol{w}^{(r)} = 1$
- Taking the partial derivatives of the Lagrangian $\boldsymbol{w}^{(r)\top}\boldsymbol{G}^{(r)\top}\boldsymbol{G}^{(r)}\boldsymbol{w}^{(r)} - \alpha(\mathbf{1}^\top \boldsymbol{w}^{(r)} - 1)$ with respect to $\boldsymbol{w}^{(r)}$ and α and setting them to zero, we have $\begin{cases} 2\boldsymbol{G}^{(r)\top}\boldsymbol{G}^{(r)}\boldsymbol{w}^{(r)} - \alpha\mathbf{1} = 0 \\ \mathbf{1}^\top \boldsymbol{w}^{(r)} = 1 \end{cases}$
  - $\boldsymbol{w}^{(r)} = \frac{(\boldsymbol{G}^{(r)\top}\boldsymbol{G}^{(r)})^{-1}\mathbf{1}}{\mathbf{1}^\top(\boldsymbol{G}^{(r)\top}\boldsymbol{G}^{(r)})^{-1}\mathbf{1}}$ can be solve analytically [Proof]
  - Note for $\boldsymbol{G}^{(r)\top}\boldsymbol{G}^{(r)}$ to be invertible, we need make sure that $n \leqslant d$

Solving $\{z^{(t)}\}_{t=1}^{N}$ (1/3)

- Now we are given $\boldsymbol{W} \in \mathbb{R}^{N \times N}$, a matrix with nonzero elements corresponding to the weights $w_{r,s}$ found in step 2
  - $\boldsymbol{W}\mathbf{1} = \mathbf{1}$
- Let $\boldsymbol{Z} = \begin{bmatrix} z^{(1)} \\ \vdots \\ z^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times k}$. Since the attributes of z are uncorrelated, the columns $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_k \in \mathbb{R}^{N \times 1}$ of $\boldsymbol{Z}$ are orthogonal to each other
- To get $\boldsymbol{c}_1$, we consider only the first attribute of $z^{(t)}$ and rewrite the objective in step 3 as

$$\sum_r \left\| z_1^{(r)} - \sum_s w_{r,s} z_1^{(s)} \right\|^2 = \sum_r z_1^{(r)2} - \sum_r z_1^{(r)} \left(\sum_s w_{r,s} z_1^{(s)} \right) -$$
$$\sum_r \left(\sum_s w_{r,s} z_1^{(s)} \right) z_1^{(r)} + \sum_r \left(\sum_s w_{r,s} z_1^{(s)} \right)^2 =$$
$$\boldsymbol{c}_1^\top \boldsymbol{c}_1 - \boldsymbol{c}_1^\top (\boldsymbol{W}\boldsymbol{c}_1) - (\boldsymbol{W}\boldsymbol{c}_1)^\top \boldsymbol{c}_1 + (\boldsymbol{W}\boldsymbol{c}_1)^\top (\boldsymbol{W}\boldsymbol{c}_1) =$$
$$((\boldsymbol{I} - \boldsymbol{W})\boldsymbol{c}_1)^\top ((\boldsymbol{I} - \boldsymbol{W})\boldsymbol{c}_1) = \boldsymbol{c}_1^\top \boldsymbol{M} \boldsymbol{c}_1, \text{ where } \boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{W})^\top (\boldsymbol{I} - \boldsymbol{W})$$

  - Subject to $\frac{1}{N-1} \boldsymbol{c}_1^\top \boldsymbol{c}_1 = 1$

Solving $\{z^{(t)}\}_{t=1}^{N}$ (2/3)

- Taking the partial derivatives of the Lagrangian $\boldsymbol{c}_1^\top \boldsymbol{M}\boldsymbol{c}_1 - \alpha(\boldsymbol{c}_1^\top \boldsymbol{c}_1 - N + 1)$ with respect to $\boldsymbol{c}_1$ and α and setting them to zero, we have $\begin{cases} 2\boldsymbol{M}\boldsymbol{c}_1 - 2\alpha\boldsymbol{c}_1 = \boldsymbol{0} \\ \boldsymbol{c}_1^\top \boldsymbol{c}_1 - N + 1 = 0 \end{cases}$, implying $\boldsymbol{M}\boldsymbol{c}_1 = \alpha\boldsymbol{c}_1$ and $\boldsymbol{c}_1^\top \boldsymbol{M}\boldsymbol{c}_1 = (N-1)\alpha$
  - $\boldsymbol{c}_1$ is the eigenvector of $\boldsymbol{M}$ corresponding to the smallest eigenvalue
- Similarly, $\boldsymbol{c}_j$ which is orthogonal to $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_{j-1}$ is the eigenvector corresponding to the jth smallest eigenvalue
  - $\boldsymbol{M}$ is symmetric and has orthogonal eigenvectors

Solving $\{z^{(t)}\}_{t=1}^{N}$ (3/3)

- Note we did not enforce the constraint $\frac{1}{N}\sum_t z^{(t)} = \frac{1}{N}\boldsymbol{Z}^\top \mathbf{1} = \mathbf{0}$ (or $\frac{1}{N}\boldsymbol{c}_j^\top \mathbf{1} = 0$ for all $1 \leqslant j \leqslant k$)
- Notice that the first eigenvector is always $\mathbf{1}$
  - Since $\sum_s w_{r,s} = 1$, we have $\boldsymbol{M}\mathbf{1} = (\boldsymbol{I} - \boldsymbol{W})^\top (\boldsymbol{I} - \boldsymbol{W})\mathbf{1} = (\boldsymbol{I} - \boldsymbol{W})^\top (\mathbf{1} - \mathbf{1}) = \mathbf{0}$
  - $\boldsymbol{M}$ is positive semidefinite and 0 must be the smallest eigenvalue [Proof]
- To be orthogonal to $\mathbf{1}$, all other eigenvectors must have components summed to 0, by virtue of orthogonality
- We can simply discard $\mathbf{1}$ and let $\boldsymbol{c}_j$ correspond to the $(j+1)$th eigenvector to enforce $\boldsymbol{c}_j^\top \mathbf{1} = 0$
- Finally, $z_j^{(t)}$ equals to the tth component of the eigenvector $\boldsymbol{c}_j \in \mathbb{R}^{N \times 1}$ of $\boldsymbol{M}$

Summary of Nonlinear Methods

- LLE has a similar problem with Isomap in that there is no mapping between $\boldsymbol{x}^{(t)}$ and $\boldsymbol{z}^{(t)}$
  - The interpolation technique applies to LLE
- It can be shown that LLE is equivalent to kernel PCA with the "LLE kernel"
- Both Isomap and LLE reconstruct the manifold in a k-dimensional space by patching the overlapping neighborhoods
  - In Isomap, geodesic distances are calculated hop by hop and preserved in the low dimensional space
  - In LLE, weights of combination are preserved hop by hop in the low dimensional space