Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

PCA

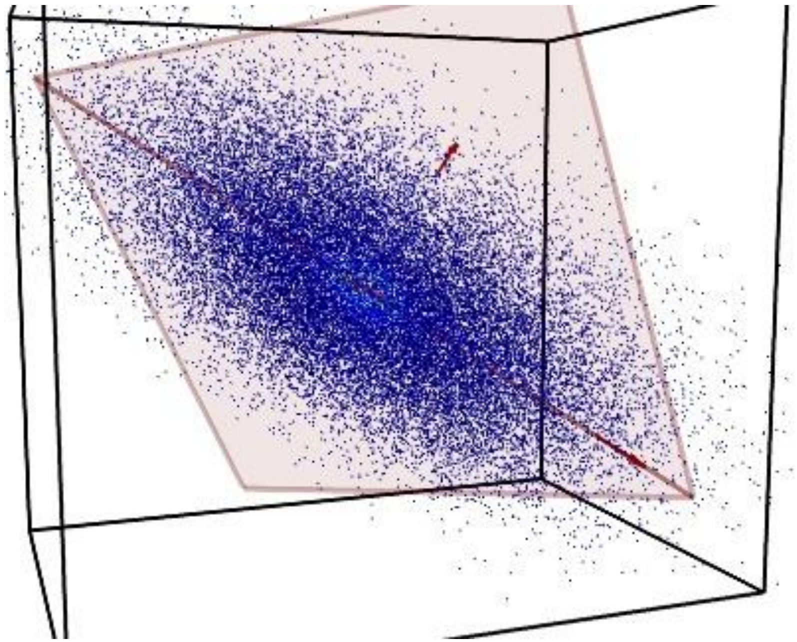
Tobias Scheffer

Overview

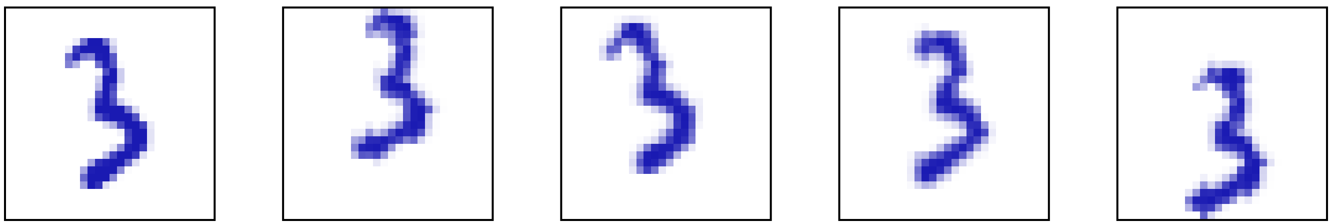
- Principal Component Analysis (PCA)
- Kernel-PCA
- Fisher Linear Discriminant Analysis
- t-SNE

PCA: Motivation

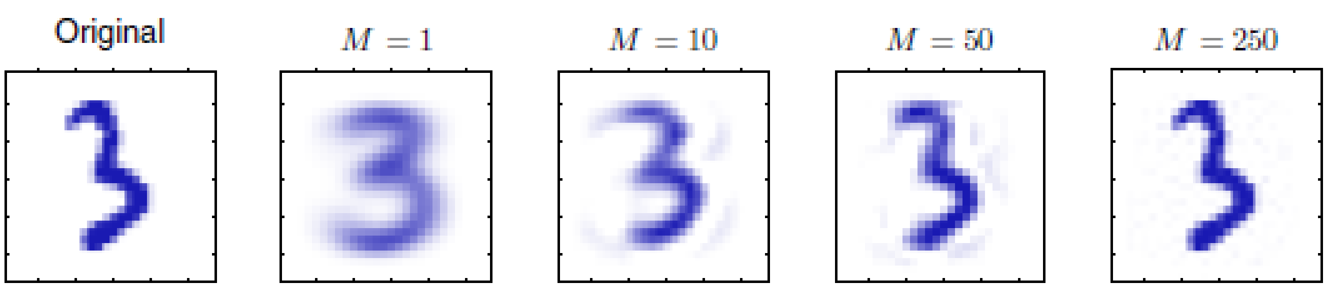
- Data compression
- Preprocessing (Feature Selection / Noisy Features)
- Data visualization



PCA: Example

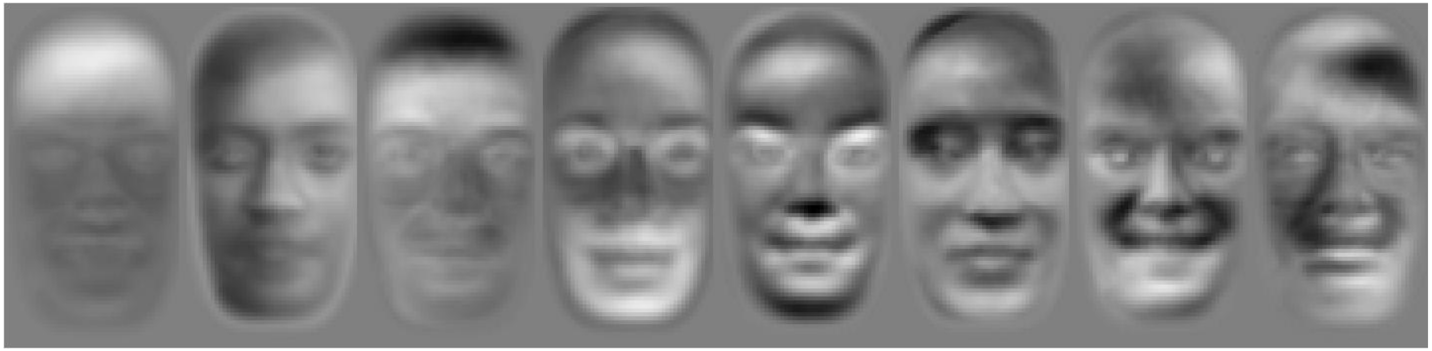


- Representation of Digits as an $m \times m$ pixel matrix
  - The actual number of degrees of freedom is significantly smaller because many features
    - Are meaningless or
    - Are composites of several pixels
- Goal: Reduce to a d-dimensional subspace



PCA: Example

- Representation of faces as an $m \times m$ pixel matrix
  - The actual number of degrees of freedom is significantly smaller because many combinations of pixels cannot occur in faces
- Reduce to a d-dimensional subspace



PCA: Projection

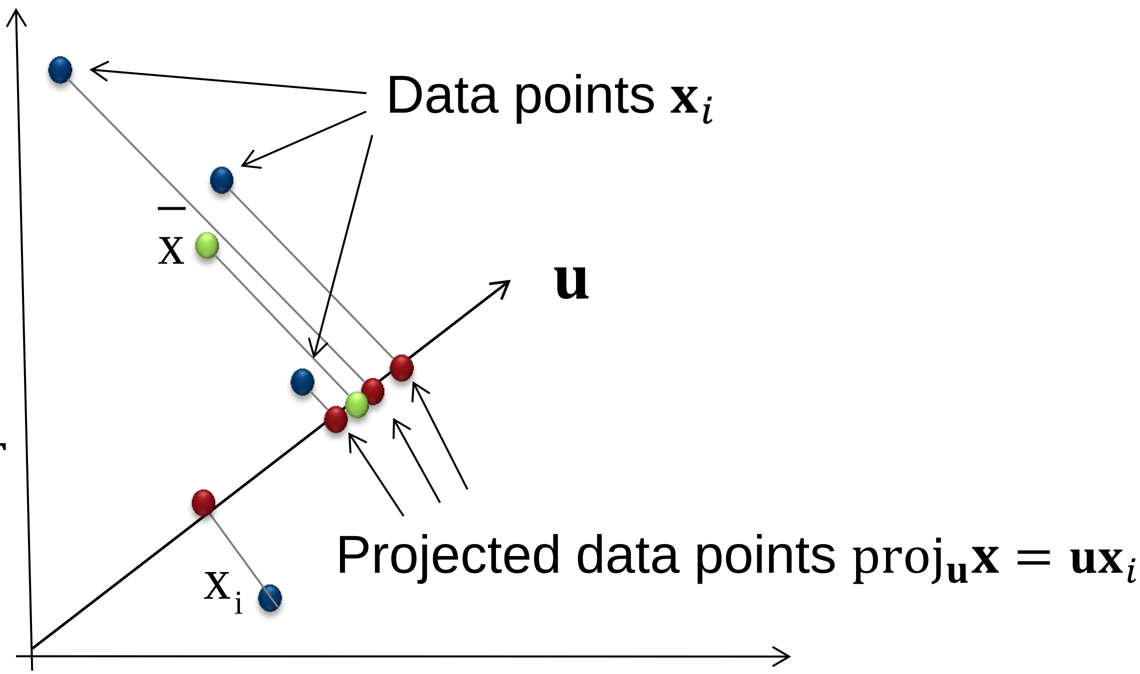
- A Projection is an idempotent linear Transformation

Center point:

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

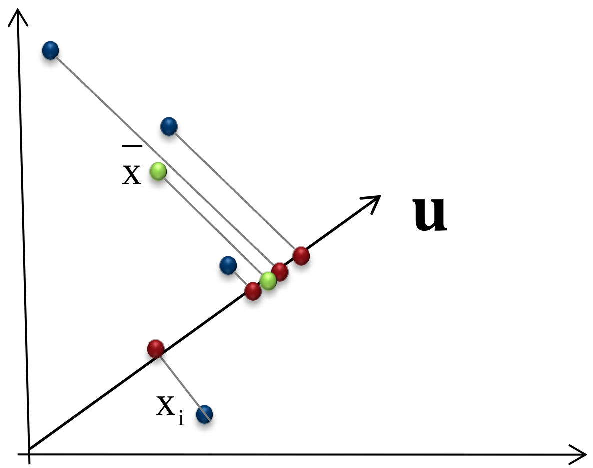
Covariance:

$$\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathrm{T}}$$



PCA: Projection

- A Projection is an idempotent linear Transformation



- Let $\mathbf{u}_1 \in \mathbb{R}^m$ with $\mathbf{u}_1{}^{\mathrm{T}}\mathbf{u}_1 = 1$
- $y_1(\mathbf{x}) = \mathbf{u}_1{}^{\mathrm{T}}\mathbf{x}$ constitutes a projection onto a one-dimensional subspace

- For data in the projection‘s space, it follows that:
  - Center (mean): $y_1(\bar{\mathbf{x}}) = \mathbf{u}_1{}^{\mathrm{T}}\bar{\mathbf{x}}$
  - Variance: $\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{u}_1{}^{\mathrm{T}}\mathbf{x}_i - \mathbf{u}_1{}^{\mathrm{T}}\bar{\mathbf{x}}\right)^2 = \mathbf{u}_1{}^{\mathrm{T}}\boldsymbol{\Sigma}\,\mathbf{u}_1$

PCA: Problem Setting

- Given: data $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ & \ddots & \\ x_{n1} & \dots & x_{n1} \end{bmatrix}$
- Find matrix $\mathbf{U} = \begin{bmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_d \\ | & & | \end{bmatrix}$ such that
  - Vectors $\mathbf{u}_i$ are orthonormal basis.
  - Vector $\mathbf{u}_1$ preserves maximal variance of data:
    $$\max_{\mathbf{u}_1 : |\mathbf{u}_1| = 1} \mathbf{u}_1^{\mathrm{T}} \frac{1}{n} \mathbf{X}\mathbf{X}^{\mathrm{T}} \mathbf{u}_1$$
  - Vector $\mathbf{u}_i$ preserves maximal residual variance.
    $$\max_{\mathbf{u}_i : |\mathbf{u}_i| = 1, \mathbf{u}_i \perp \mathbf{u}_1, \dots, \mathbf{u}_i \perp \mathbf{u}_{i-1}} \mathbf{u}_i^{\mathrm{T}} \frac{1}{n} \mathbf{X}\mathbf{X}^{\mathrm{T}} \mathbf{u}_i$$

PCA: Problem Setting

- Given: data $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ & \ddots & \\ x_{n1} & \dots & x_{n1} \end{bmatrix}$
- Find matrix $\mathbf{U} = \begin{bmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_d \\ | & & | \end{bmatrix}$ such that
- Value $\mathbf{u}_k^{\mathrm{T}}\mathbf{x}_i$ is projection of $\mathbf{x}_i$ onto dimension $\mathbf{u}_k$.
- Vector $\mathbf{U}^{\mathrm{T}}\mathbf{x}_i$ is projection of $\mathbf{x}_i$ onto coordinates $\mathbf{U}$.
- Matrix $\mathbf{Y} = \mathbf{X}\mathbf{U}$ is projection of $\mathbf{X}$ onto coordinates $\mathbf{U}$:

$$\mathbf{Y} = \mathbf{X}\mathbf{U} = \begin{bmatrix} \mathbf{x}_1\mathbf{u}_1 & \dots & \mathbf{x}_1\mathbf{u}_d \\ & \ddots & \\ \mathbf{x}_n\mathbf{u}_1 & \dots & \mathbf{x}_n\mathbf{u}_d \end{bmatrix}$$

PCA: Assumption

- To simplify the notation, assume centered data.
  - $\bar{\mathbf{x}} = 0.$
- Can be achieved by subtracting mean value
  - $\mathbf{x}_i^c = \mathbf{x}_i - \bar{\mathbf{x}}$

PCA: Direction of Maximum Variance

- Find direction $\mathbf{u}_1$ that maximizes projected variance
- Instances $\mathbf{x} \sim P_X$ (assume mean $\bar{\mathbf{x}} = 0$).
  - The projected variance onto (normalized) $\mathbf{u}_1$ is

$$\mathrm{E}\left[\left(\mathrm{proj}_{\mathbf{u}_1}\mathbf{x}\right)^2\right] = \mathrm{E}[\mathbf{u}_1{}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{u}_1] = \mathbf{u}_1{}^{\mathrm{T}}\underbrace{E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]}_{\boldsymbol{\Sigma}_{\mathbf{xx}}}\mathbf{u}_1$$

$$E[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \sum_{i=1}^{n} \begin{bmatrix} x_{i1} \\ \vdots \\ x_{im} \end{bmatrix} \begin{bmatrix} x_{i1} & \cdots & x_{im} \end{bmatrix}$$

Covariance matrix

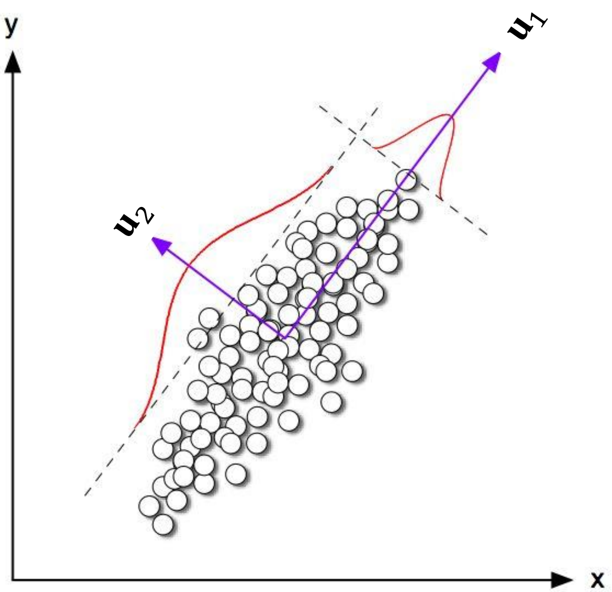
$$= \sum_{i=1}^{n} \begin{bmatrix} (x_{i1} - \bar{x}_{i1})^2 & & (x_{i1} - \bar{x}_{i1})(x_{im} - \bar{x}_{im}) \\ & \ddots & \\ (x_{im} - \bar{x}_{im})(x_{i1} - \bar{x}_{i1}) & & (x_{im} - \bar{x}_{im})^2 \end{bmatrix}$$

PCA: Direction of Maximum Variance

- Find direction $\mathbf{w}$ that maximizes projected variance
- Instances $\mathbf{x} \sim P_X$ (assume mean $\bar{\mathbf{x}} = 0$).
  - The projected variance onto (normalized) $\mathbf{u}_1$ is

$$\mathrm{E}\left[\left(\mathrm{proj}_{\mathbf{u}_1}\mathbf{x}\right)^2\right] = \mathrm{E}[\mathbf{u}_1{}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{u}_1] = \mathbf{u}_1{}^{\mathrm{T}}\underbrace{E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]}_{\boldsymbol{\Sigma}_{\mathbf{xx}}}\mathbf{u}_1$$

- The empirical covariance matrix (of centered data) is
$$\widehat{\boldsymbol{\Sigma}}_{xx} = \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}}$$
- How can we find direction $\mathbf{u}_1$ to maximize $\mathbf{u}_1{}^{\mathrm{T}}\widehat{\boldsymbol{\Sigma}}_{xx}\mathbf{u}_1$?
- How can we kernelize it?



PCA: Optimization Problem

- Solution for $\mathbf{u}_1$: max variance of the projected data:

$$\max_{\mathbf{u}_1} \mathbf{u}_1{}^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{xx}\, \mathbf{u}_1, \text{such that}$$
$$\mathbf{u}_1{}^{\mathrm{T}} \mathbf{u}_1 = 1$$

- Lagrangian: $\mathbf{u}_1{}^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{xx}\, \mathbf{u}_1 + \lambda_1 \left(1 - \mathbf{u}_1{}^{\mathrm{T}} \mathbf{u}_1\right)$

PCA: Optimization Problem

- Solution for $\mathbf{u}_1$: max variance of the projected data:

$$\max_{\mathbf{u}_1} {\mathbf{u}_1}^{\mathrm{T}} \hat{\mathbf{\Sigma}}_{xx}\, \mathbf{u}_1, \text{such that}$$
$${\mathbf{u}_1}^{\mathrm{T}} \mathbf{u}_1 = 1$$

- Lagrangian: ${\mathbf{u}_1}^{\mathrm{T}} \hat{\mathbf{\Sigma}}_{xx}\, \mathbf{u}_1 + \lambda_1\left(1 - {\mathbf{u}_1}^{\mathrm{T}} \mathbf{u}_1\right)$
- Taking its derivative & setting it to 0: $\hat{\mathbf{\Sigma}}_{xx} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$
  - The solution $\mathbf{u}_1$ must be an eigenvector of $\hat{\mathbf{\Sigma}}_{xx}$
- Variance of the projected data:

$$\mathbf{u}_1^{\mathrm{T}} \hat{\mathbf{\Sigma}}_{xx} \mathbf{u}_1 = \mathbf{u}_1^{\mathrm{T}} \lambda_1 \mathbf{u}_1 = \lambda_1$$

- The solution is the eigenvector $\mathbf{u}_1$ of $\hat{\mathbf{\Sigma}}_{xx}$ with greatest eigenvalue λ_1, called the 1st principal component

PCA: Optimization Problem

- Solution for $\mathbf{u}_i$: max variance of the projected data:

$$\max_{\mathbf{u}_i} \mathbf{u}_i{}^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{xx} \mathbf{u}_i, \text{such that}$$
$$\mathbf{u}_i{}^{\mathrm{T}} \mathbf{u}_i = 1$$
$$\mathbf{u}_i \perp \mathbf{u}_1, \dots, \mathbf{u}_i \perp \mathbf{u}_{i-1}$$

- Lagrangian: $\mathbf{u}_i{}^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{xx} \mathbf{u}_i + \lambda_i \left(1 - \mathbf{u}_i{}^{\mathrm{T}} \mathbf{u}_i\right)$
- Taking its derivative & setting it to 0: $\widehat{\boldsymbol{\Sigma}}_{xx} \mathbf{u}_i = \lambda_i \mathbf{u}_i$
  - The solution $\mathbf{u}_i$ must be an eigenvector of $\widehat{\boldsymbol{\Sigma}}_{xx}$
- To maximize variance of the projected data:

$$\mathbf{u}_i^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{xx} \mathbf{u}_i = \mathbf{u}_i^{\mathrm{T}} \lambda_i \mathbf{u}_i = \lambda_i$$

- And to assure that $\mathbf{u}_i$ are orthogonal:
  - $\mathbf{u}_i$ is eigenvector with next-best eigenvalue $\lambda_i < \lambda_{i-1}$.

PCA: Optimization Problem

- Eigenvector decomposition implies:
$$\widehat{\boldsymbol{\Sigma}}_{xx} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}$$
- With $\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_m \end{bmatrix}$
- However, if $\mathbf{U}_{1:d}$ contains the first d eigenvectors, then $\mathbf{Y} = \mathbf{X}\mathbf{U}_{1:d}$ has only a fraction of the variance:
$$\frac{\sum_{i=1}^{d} \lambda_i}{tr(\widehat{\boldsymbol{\Sigma}}_{xx})}$$
- Choose d smaller than m but large enough to cover most of the variance.

PCA

- Projection of $\mathbf{x}$ to the eigenspace:

$$y_1(\mathbf{x}) = \mathbf{u}_1{}^{\mathrm{T}}\mathbf{x} \Rightarrow y(\mathbf{x}) = \mathbf{U}^{\mathrm{T}}\mathbf{x} \text{ with } \mathbf{U} = \begin{pmatrix} | & & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_d \\ | & & | \end{pmatrix}$$

$$\mathbf{Y} = \mathbf{X}\mathbf{U}$$

- Largest eigenvector is 1st principal component
- The remaining principal components are orthogonal directions which maximize the residual variance
  - d principal components → vectors of the d largest eigenvalues

PCA: Reverse Projection

- Observation: $\{\mathbf{u}_j\}$ form a basis for $\mathbb{R}^m$ & $\langle y_j(\mathbf{x})\rangle$ are the coordinates of $\mathbf{x}$ in that basis
  - Data $\mathbf{x}_i$ can thus be reconstructed in that basis:

$$\mathbf{x}_i = \sum_{j=1}^{m} \left(\mathbf{x}_i{}^{\mathrm{T}}\mathbf{u}_j\right)\mathbf{u}_j \quad \text{or} \quad \mathbf{X} = \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}$$

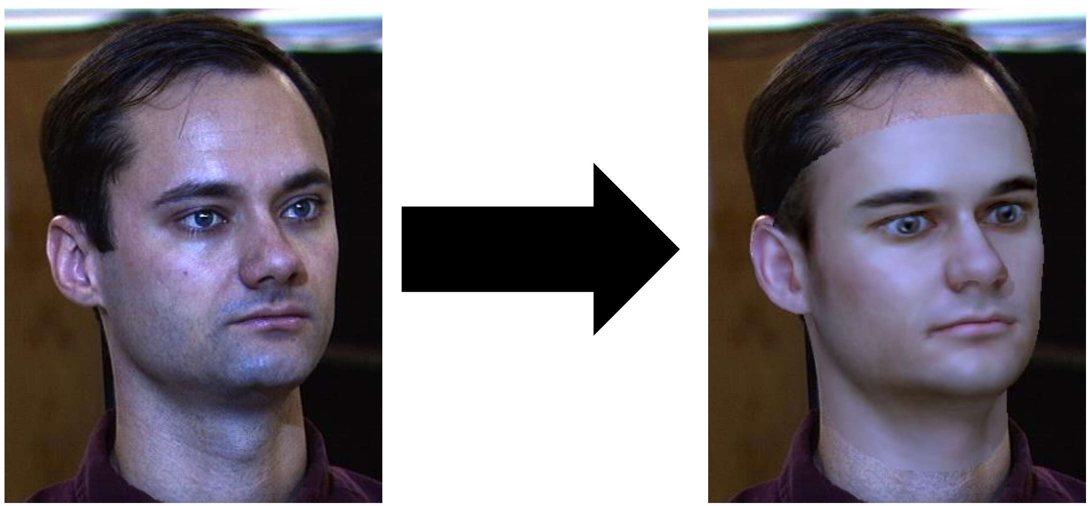
- If data lies (mostly) in d-dimensional principal subspace, we can also reconstruct the data there:

$$\tilde{\mathbf{x}}_i = \sum_{j=1}^{d} \left(\mathbf{x}_i{}^{\mathrm{T}}\mathbf{u}_j\right)\mathbf{u}_j \quad \text{or} \quad \widetilde{\mathbf{X}} = \mathbf{U}_{1:d}\mathbf{U}_{1:d}{}^{\mathrm{T}}\mathbf{X}$$

  - where $\mathbf{U}_{1:d}$ is the matrix of 1st d eigenvectors

Reverse Projection: Example

- Morphace (Universität Basel)
  - 3D face model of 200 persons (150,000 features)
  - PCA with 199 principal components.



PCA: Algorithm

- PCA finds dataset's principal components, which maximize the projected variance
- Algorithm:
  1. Compute data's mean: $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$
  2. Compute data's covariance: $\hat{\boldsymbol{\Sigma}}_{xx} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathrm{T}}$
  3. Find principal axes: $\mathbf{U} = \mathrm{eigenvektors}\left(\hat{\boldsymbol{\Sigma}}_{xx}\right)$
  4. Project data onto 1st d eigenvectors

$$\tilde{\mathbf{x}}_i \leftarrow \mathbf{U}_{1:d}{}^{\mathrm{T}}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})$$

Difference Between PCA and Autoencoder

- PCA: Linear mapping $y(\mathbf{x}) = \mathbf{U}^{\mathrm{T}}\mathbf{x}$ from $\mathbf{x}$ to $\mathbf{y}$.
- Autoencoder: Linear mapping from $\mathbf{x}$ to $\mathbf{h}$, then nonlinear activation function $\mathbf{y} = \boldsymbol{\sigma}(\mathbf{x})$.
- Autoencoder with squared loss and linear activation function = PCA.
- Stacked autoencoder: more nonlinearity, more complex mappings.
- Kernel-PCA: linear mapping in feature space $\boldsymbol{\Phi}$.

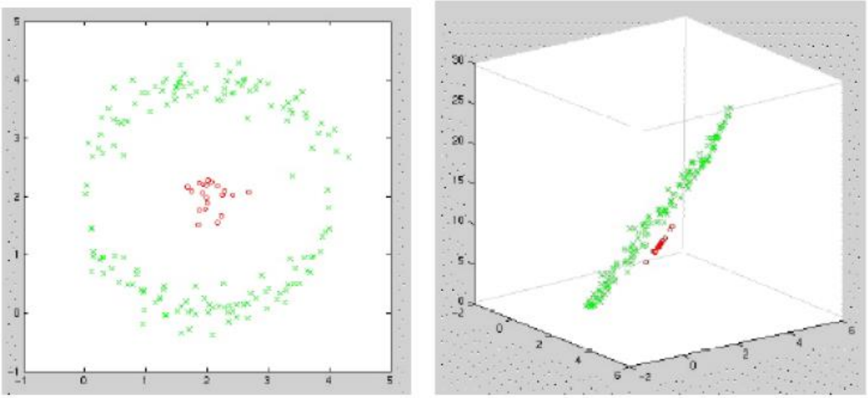
Overview

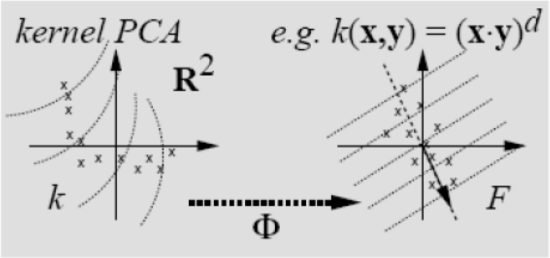
- Principal Component Analysis (PCA)
- Kernel-PCA
- Fisher Linear Discriminant Analysis
- t-SNE

Kernel PCA

- PCA can only capture linear subspaces
  - More complex features can capture non-linearity
  - Want to use PCA in high-dimensional spaces

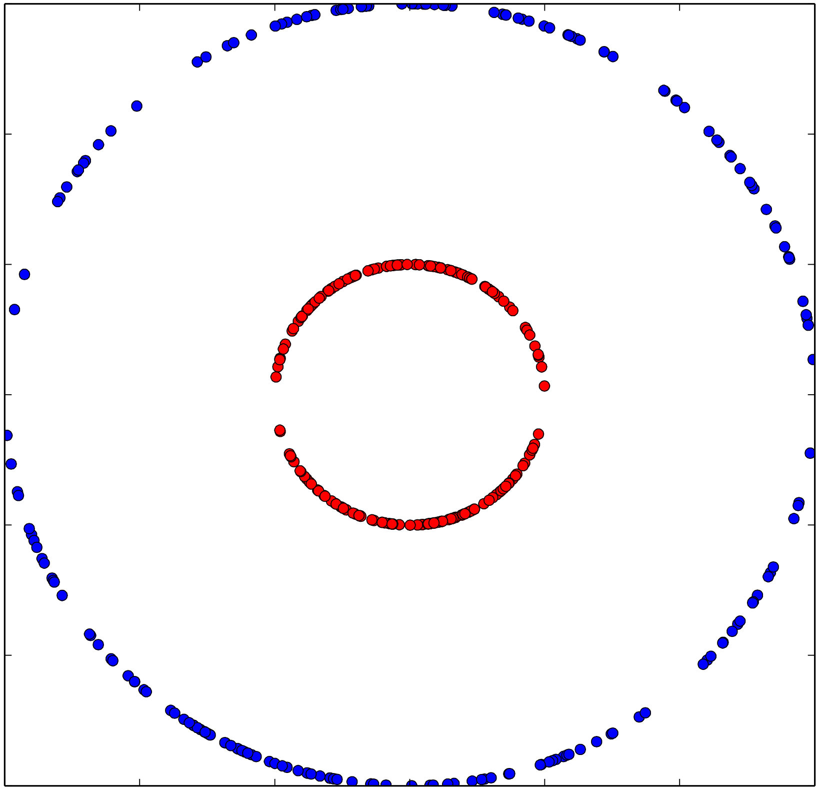
$$\Phi : \mathcal{X} = \mathbb{R}^2 \to \mathcal{H} = \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$





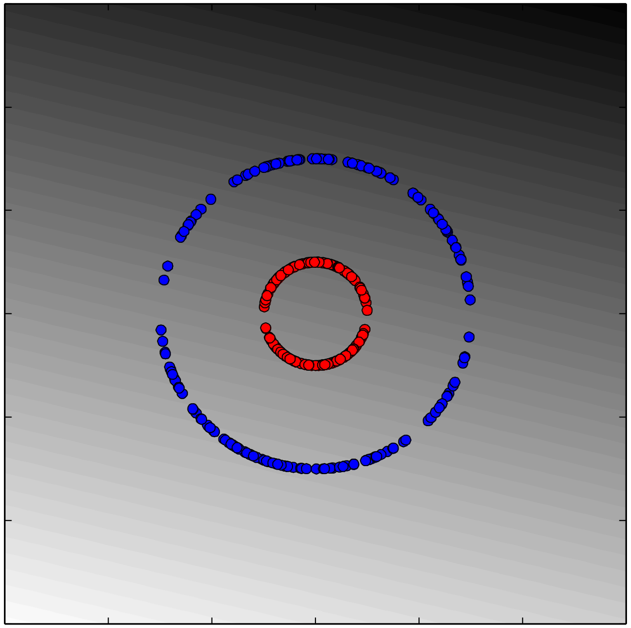
- Requirements: Data only interact through inner product

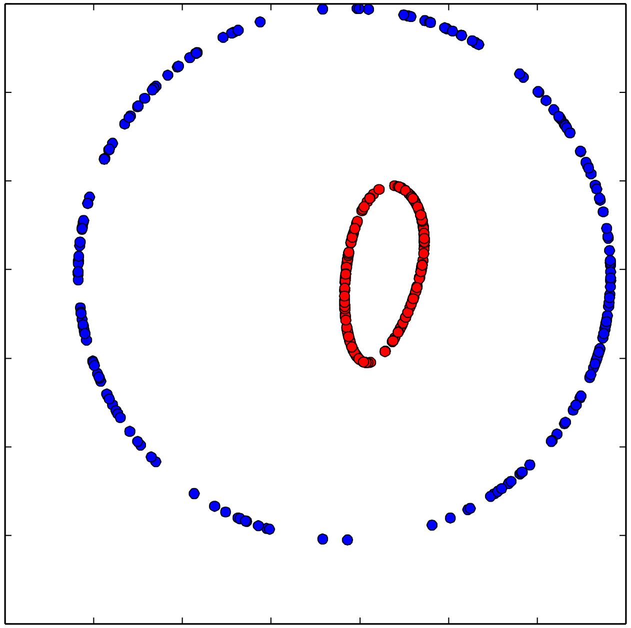
Kernel PCA: Example



Kernel PCA: Example

- PCA fails to capture the data's two ring structure—rings are not separated in the first 2 components.





Kernel PCA: Kernel Recap

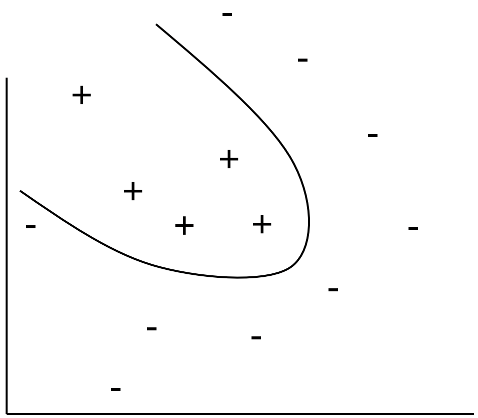
- Linear classifiers:
  - Often adequate, but not always.
- Idea: data implicitly mapped to another space, in which they are linearly classifiable
- Image mapping:

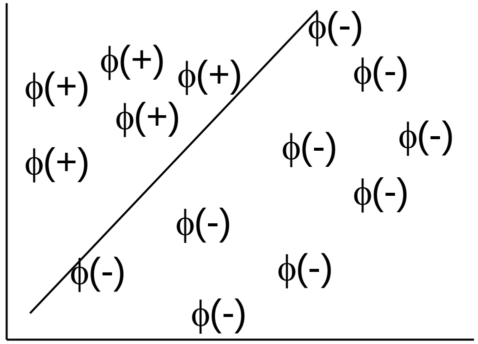
$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

- Associated kernel:

$$\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi\left(\mathbf{x}_j\right)$$

- Kernel = inner product = similarity of Examples.





Kernel PCA

- Covariance of centered data:

$$\hat{\boldsymbol{\Sigma}}_{xx} = \frac{1}{n}\sum_i \mathbf{x}_i \mathbf{x}_i^{\mathrm{T}} = \sum_i \begin{bmatrix} x_{i1} \\ \vdots \\ x_{im} \end{bmatrix} \begin{bmatrix} x_{i1} & \dots & x_{im} \end{bmatrix}$$

- Eigenvectors: $\hat{\boldsymbol{\Sigma}}_{xx}\mathbf{u} = \boldsymbol{\lambda}\mathbf{u}.$
- In feature space, centered:

$$\hat{\boldsymbol{\Sigma}}_{\phi(x)\phi(x)} = \sum_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^{\mathrm{T}}$$

- Eigenvectors:

$$\hat{\boldsymbol{\Sigma}}_{\phi(x)\phi(x)}\mathbf{u} = \boldsymbol{\lambda}\mathbf{u}$$

$$\sum_i \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^{\mathrm{T}}\,\mathbf{u} = \boldsymbol{\lambda}\mathbf{u}$$

- All solutions live in the span of $\phi(\mathbf{x}_1), \dots \phi(\mathbf{x}_n)$

Kernel PCA

- All solutions live in the span of $\phi(\mathbf{x}_1), \ldots \phi(\mathbf{x}_n)$
- Hence, all eigenvectors $\mathbf{u}$ must be linear combination of $\phi(\mathbf{x}_1), \ldots \phi(\mathbf{x}_n)$:

$$\exists \alpha_k : \mathbf{u} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i)$$

- Hence, $\widehat{\boldsymbol{\Sigma}}_{\phi(x)\phi(x)} \mathbf{u} = \boldsymbol{\lambda} \mathbf{u}$ is satisfied if n projected equations are satisfied:

$$\forall i : \phi(x_i)^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{\phi(x)\phi(x)} \mathbf{u} = \lambda \phi(x_i) \mathbf{u}$$

$$\Rightarrow \phi(x_i)^{\mathrm{T}} \sum_j \phi(\mathbf{x}_j) \phi(\mathbf{x}_j)^{\mathrm{T}} \sum_{k=1}^{n} \alpha_k \phi(\boldsymbol{x}_k)$$

$$= \lambda \phi(\mathbf{x}_j)^{\mathrm{T}} \sum_{k=1}^{n} \alpha_k \phi(\boldsymbol{x}_k)$$

Kernel PCA

- n projected equations:

$$\forall i: \phi(\mathbf{x}_i)^{\mathrm{T}} \widehat{\boldsymbol{\Sigma}}_{\phi(x)\phi(x)} \mathbf{u} = \lambda \phi(x_i) \mathbf{u}$$

$$\Rightarrow \phi(\mathbf{x}_i)^{\mathrm{T}} \frac{1}{n} \sum_j \phi(\mathbf{x}_j) \phi(\mathbf{x}_j)^{\mathrm{T}} \sum_{k=1}^{n} \alpha_k \phi(\mathbf{x}_k)$$

$$= \lambda \phi(\mathbf{x}_j)^{\mathrm{T}} \sum_{k=1}^{n} \alpha_k \phi(\mathbf{x}_k)$$

$$\Rightarrow \frac{1}{n} \sum_{j,k} \alpha_k [\phi(\mathbf{x}_i)^{\mathrm{T}} \phi(\mathbf{x}_j)][\phi(\mathbf{x}_j)^{\mathrm{T}} \phi(\mathbf{x}_k)]$$

$$= \lambda \sum_{k=1}^{n} \alpha_k [\phi(\mathbf{x}_i)^{\mathrm{T}} \phi(\mathbf{x}_k)]$$

$$\Leftrightarrow \mathbf{K}^2 \boldsymbol{\alpha} = n\lambda \mathbf{K} \boldsymbol{\alpha}$$

$$\Leftarrow \mathbf{K} \boldsymbol{\alpha} = n\lambda \boldsymbol{\alpha}$$

$$\mathbf{K} = \phi(\mathbf{X}) \phi(\mathbf{X})^{\mathrm{T}}$$

Kernel PCA

- Results in eigenvalue problem:

$$\mathbf{K}\boldsymbol{\alpha} = n\lambda\boldsymbol{\alpha}$$

- Centering data in feature space:

$$\mathbf{K}_{ij}^{c} = \left(\phi(\mathbf{x}_i) - \frac{1}{n}\sum_k \phi(\mathbf{x}_k)\right)\left(\phi(\mathbf{x}_j) - \frac{1}{n}\sum_k \phi(\mathbf{x}_k)\right)$$

$$= \mathbf{K}_{ij} - \mathbf{k}_i \mathbf{1}_j^{\mathrm{T}} - \mathbf{1}_i \mathbf{k}_j^{\mathrm{T}} + \mathbf{k}\mathbf{1}_i \mathbf{1}_j^{\mathrm{T}}$$

$$\mathbf{k}_i = \frac{1}{n}\sum_k \mathbf{K}_{ik} \qquad \mathbf{k} = \frac{1}{n^2}\sum_{j,k} \mathbf{K}_{jk}$$

Intelligent Data Analysis II

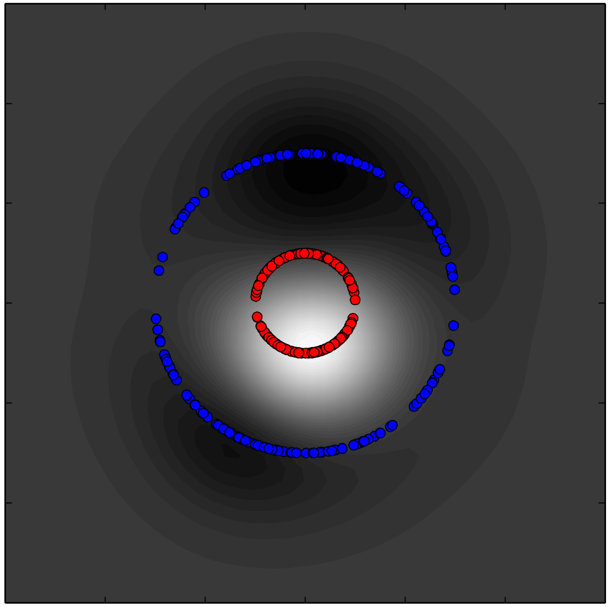
Kernel-PCA

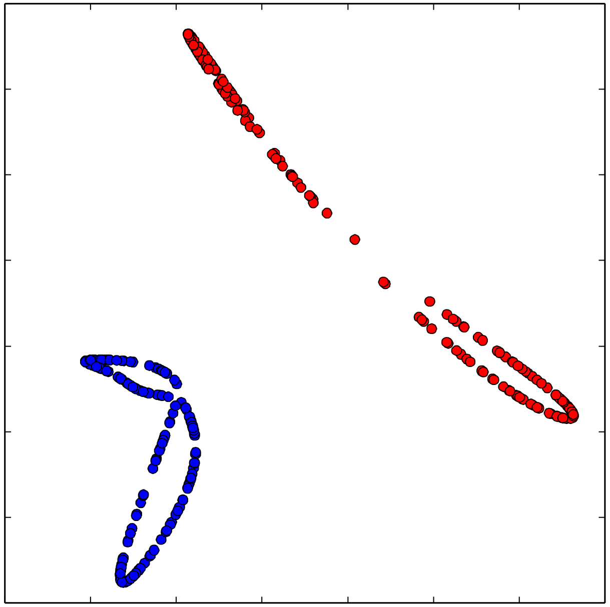
Algorithm

- Kernel-PCA finds dataset's principal components in an implicitly defined feature space
- Algorithm:
  1. Compute kernel matrix $\mathbf{K}$: $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$
  2. Center the kernel matrix:
  $$\bar{\mathbf{K}} = \mathbf{K} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{K} - \frac{1}{n}\mathbf{K}\mathbf{1}\mathbf{1}^{\mathrm{T}} + \frac{\mathbf{1}^{\mathrm{T}}\mathbf{K}\mathbf{1}}{n^{2}}\mathbf{1}\mathbf{1}^{\mathrm{T}}$$
  3. Find its eigenvectors: $\mathbf{U}, \mathbf{V} = \mathrm{eig}(\bar{\mathbf{K}})$
  4. Find the dual vectors: $\boldsymbol{\alpha}_k = \lambda_k^{-1/2}\mathbf{u}_k$
  5. Project the data onto the subspace:
  $$\tilde{\mathbf{x}}_j \leftarrow \left\langle \sum_{i=1}^{n} \alpha_{k,i}\bar{K}_{ij} \right\rangle_{k=1}^{d} = \left\langle \boldsymbol{\alpha}_k{}^{\mathrm{T}}\bar{\mathbf{K}}_{*,j} \right\rangle_{k=1}^{d}$$

Kernel PCA
Ring Data Example

- Kernel PCA (RBF) does capture the data's structure & resulting projections separate the 2 rings



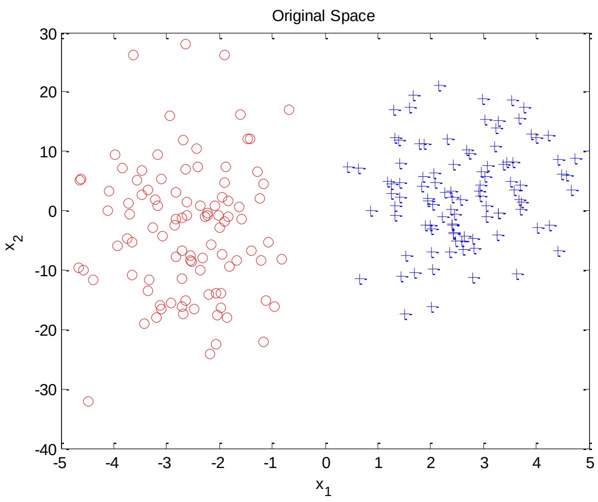


Overview

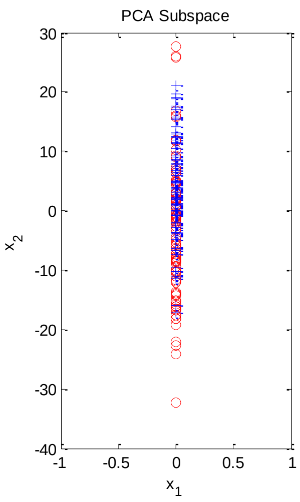
- Principal Component Analysis (PCA)
- Kernel-PCA
- Fisher Linear Discriminant Analysis
- t-SNE

Fisher-Discriminant Analysis (FDA)

- The subspace induced by PCA maximally captures variance from *all* data
  - Not the correct criterion for classification…



$$\mathbf{X}^{\mathrm{T}}\mathbf{u}_{PCA}$$



$$\boldsymbol{\Sigma}\mathbf{u}_{PCA} = \lambda_{PCA}\mathbf{u}_{PCA}$$

Fisher-Discriminant Analysis (FDA)

- Optimization criterion of PCA:
  - Maximize the data's variance in the subspace.

$$\max_{\mathbf{u}_i} \mathbf{u}^{\mathrm{T}} \mathbf{\Sigma} \mathbf{u}, \text{ where } \mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = 1, \mathbf{u}_i \perp \mathbf{u}_j$$

- Optimization criterion of FDA:
  - Maximize between-class variance and minimize within-class variance within the subspace.

Variance per class

$$\max_{\mathbf{u}} \frac{\mathbf{u}^{\mathrm{T}} \mathbf{\Sigma}_b \mathbf{u}}{\mathbf{u}^{\mathrm{T}} \mathbf{\Sigma}_w \mathbf{u}}, \text{ where } \begin{array}{c} \mathbf{\Sigma}_w = \mathbf{\Sigma}_{+1} + \mathbf{\Sigma}_{-1} \\ \mathbf{\Sigma}_b = (\bar{\mathbf{x}}_{+1} - \bar{\mathbf{x}}_{-1})(\bar{\mathbf{x}}_{+1} - \bar{\mathbf{x}}_{-1})^{\mathrm{T}} \end{array}$$

Fisher-Discriminant Analysis (FDA)

- Optimization criterion of FDA for k classes:
  - Maximize between-class variance and minimize within-class variance within the subspace.

$$\max_{\mathbf{u}} \frac{\mathbf{u}^{\mathrm{T}}\boldsymbol{\Sigma}_b\mathbf{u}}{\mathbf{u}^{\mathrm{T}}\boldsymbol{\Sigma}_w\mathbf{u}}, \text{ where } \begin{aligned} \boldsymbol{\Sigma}_w &= \boldsymbol{\Sigma}_1 + \cdots + \boldsymbol{\Sigma}_k \\ \boldsymbol{\Sigma}_b &= \sum_{i=1}^{k} n_i(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

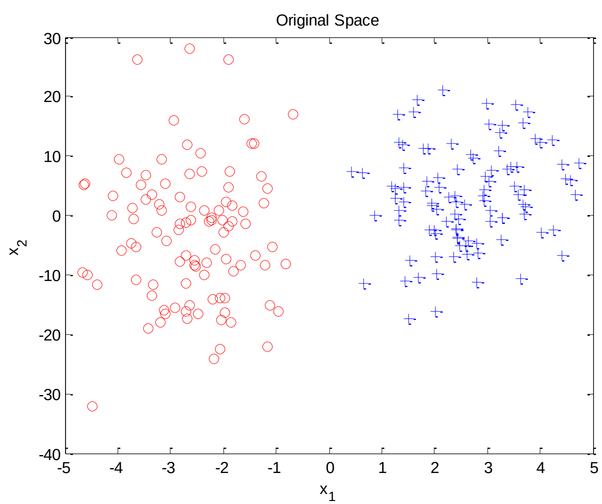
Leads to the generalized eigenvalue problem $\boldsymbol{\Sigma}_b\mathbf{u} = \lambda\boldsymbol{\Sigma}_w\mathbf{u}$

Number of instances per class

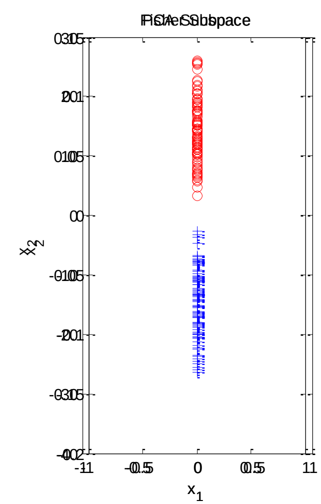
- Generalized eigenvalue problem has $k - 1$ different solutions

Fisher-Discriminant Analysis (FDA)

- The subspace induced by PCA maximally captures variance from *all* data
  - Not the correct criterion for classification…



$$\mathbf{x}^{\mathrm{T}}\mathbf{u}_{FDA}$$



$$\mathbf{\Sigma}_b \mathbf{u}_{FIS} = \lambda_{FIS} \mathbf{\Sigma}_w \mathbf{u}_{FIS}$$

Overview

- Principal Component Analysis (PCA)
- Kernel-PCA
- Fisher Linear Discriminant Analysis
- t-SNE

t-SNE

- Impossible to preserve all distances when projecting data into lower-dimensional space.
- PCA: Preserve maximum variance.
  - Variance is squared distance.
  - Sum is dominated by instances that are far apart.
  - → Instances that are far apart from each other shall remain as far apart in the projected space.

- Idea of t-SNE: Preserve local neighborhood.
  - Instances that are close to each other shall remain close in the projected space.
  - Instances that are far apart may be moved further apart by the projection.

2D PCA for MNIST Handwritten Digits

- PAC is poor at preserving closeness between similar bitmaps.

Local Neighborhood in Original Space

- Probability that $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as neighbor if neighbors were picked by Gaussian distribution centered at $\mathbf{x}_i$:

$$p_{j|i} = \frac{\exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma_i^2}\right)}{\sum_{j\neq i}\exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma_i^2}\right)}$$

- Set each σ_i such that conditional has fixed entropy.

Distance in Projected Space

- Probability that $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as neighbor if neighbors were picked by Student‘s t-distribution centered at $\mathbf{x}_i$:

$$q_{j|i} = \frac{\left(1 + \left|\left|\mathbf{y}_i - \mathbf{y}_j\right|\right|^2\right)^{-1}}{\sum_{j \neq i} \left(1 + \left|\left|\mathbf{y}_i - \mathbf{y}_j\right|\right|^2\right)^{-1}}$$

- Student‘s t-distribution has heavier tails: very large distances are more likely than under Gaussian.
- Moving far instances further apart incurs less penalty.

t-SNE: Optimization Criterion

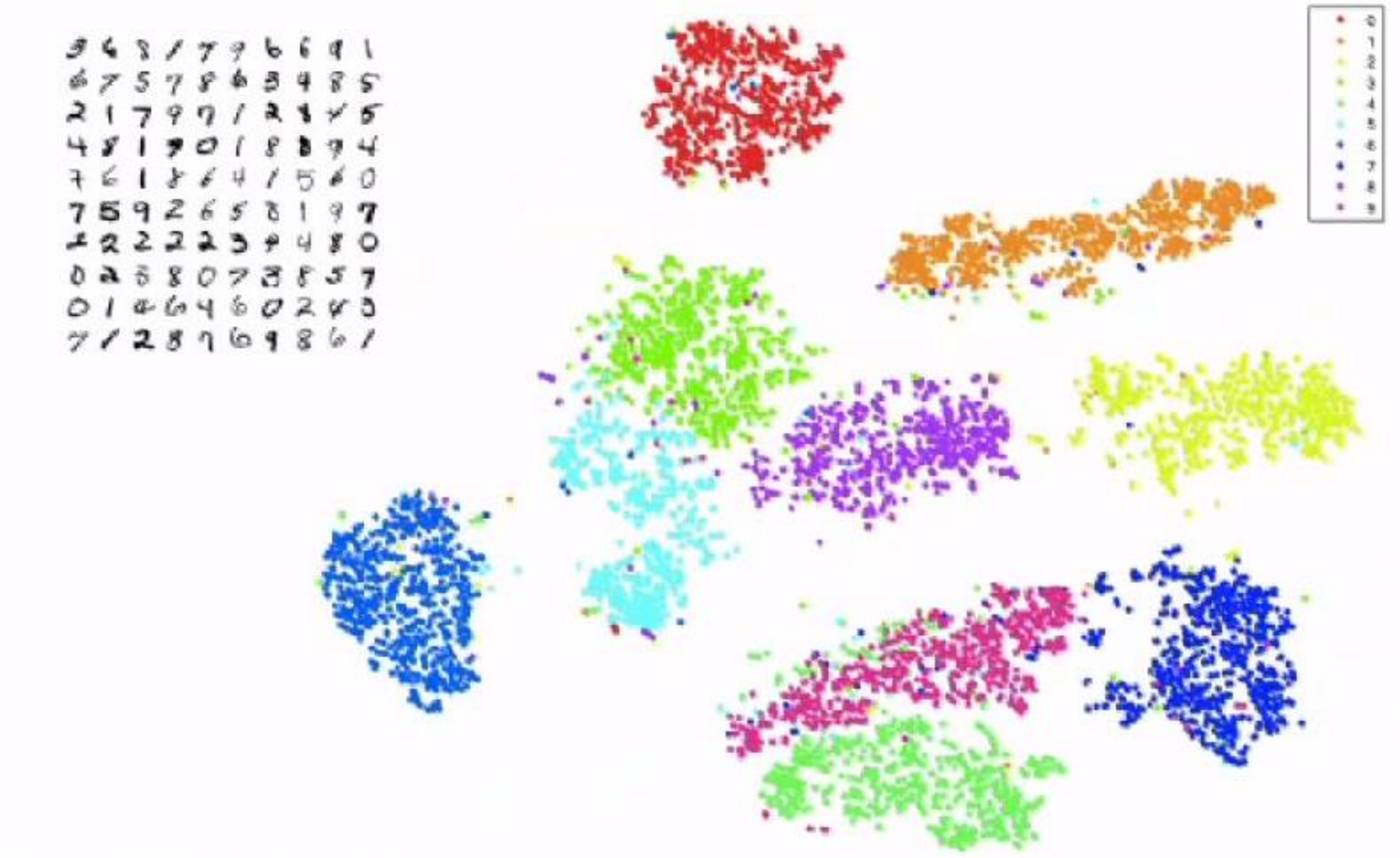
- Move instances around in projected space to minimize Kullback-Leibler divergence:
$$KL(p||q) = \sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$
- If $p_{j|i}$ is large but $q_{j|i}$ is small: large penalty.
- If $q_{j|i}$ is large but $p_{j|i}$ is small: smaller penalty.
- Hence, preserves local neighborhood structure of the data.

t-SNE: Optimization

- Move instances around in projected space to minimize Kullback-Leibler divergence
- Gradient for projected instance $\mathbf{y}_i$:
$$\frac{\partial KL(p||q)}{\partial \mathbf{y}_i} = 4\sum_{j \neq i}(p_{j|i} - q_{j|i})\left(1 + \left|\left|\mathbf{y}_i - \mathbf{y}_j\right|\right|^2\right)^{-1}(\mathbf{y}_i - \mathbf{y}_j)$$
- Implementation for large samples:
  - Build quadtree over data
  - Approximate $p_{j|i}$ and $q_{j|i}$ of instances in distinct branches by distances between centers of mass.

2D t-SNE for MNIST Handwritten Digits

- Local similarities are preserved better.



Summary

- PCA constructs lower-dimensional space that preserves most of the variance.
- Kernel PCA works on the kernel matrix; good when there are fewer instances than there are features.
- Fisher linear discriminant analysis maximizes between-class and minimizes within-class variance
- t-SNE finds a projection that preserves local neighborhood relations.