Chapter 4
Feature Extraction

supplementary slides to
Machine Learning Fundamentals
©Hui Jiang 2020
published by Cambridge University Press

August 2020

Outline

Feature Extraction

- feature engineering
  - use domain knowledge to manually extract features from raw data
  - e.g. bag-of-words for text, MFC features for speech/audio, SIFT features for image/video
- feature normalization:
  - normalize each dimension towards zero-mean and unit-variance
- feature selection
  - select a subset of the most informative features and discard the rest
  - e.g. filter, wrapper or embedded method
- dimensionality reduction

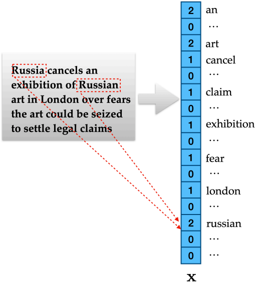
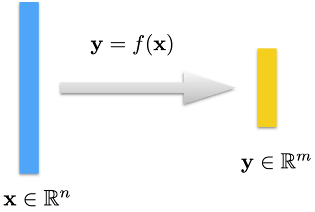


Figure: represent a text document as a fixed-size bag-of-words feature vector

Dimensionality Reduction

- utilize a mapping function to convert high-dimensional feature vectors to lower-dimensional ones while retaining the information as much as possible
- the function $f(\cdot)$ maps any point in an n-dimensional space to a point in an m-dimensional space, where $m \ll n$
- choices for $f(\cdot)$:
  - linear transformation
  - piece-wise linear functions
  - nonlinear functions
  - neural networks
- the learning criterion: what to retain



Linear Dimension Reduction

- use a linear mapping function

$$\mathbf{y} = f(\mathbf{x}) = \mathbf{A}\,\mathbf{x}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ denotes all parameters to be estimated

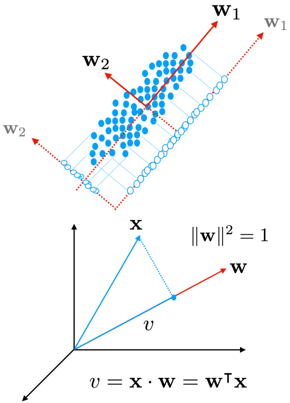
- depending on the learning criterion:
  - principal component analysis (PCA)
  - linear discriminant analysis (LDA)

Principal Component Analysis (PCA)

- more information $\Longleftrightarrow$ larger variance
- PCA aims to search for some orthogonal projection directions to achieve the maximum variances

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \xrightarrow{v=\mathbf{w}^\intercal \mathbf{x}} \{v_1, v_2, \cdots, v_N\}$$

$$\begin{aligned}\sigma^2 &= \frac{1}{N}\sum_{i=1}^{N}(v_i - \bar{v})(v_i - \bar{v}) \\ &= \mathbf{w}^\intercal \underbrace{\left[\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\intercal\right]}_{\mathbf{S}:\,\text{sample covariance matrix}} \mathbf{w}\end{aligned}$$



Principal Component Analysis (PCA)

- the principal components can be derived:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \; \mathbf{w}^\intercal \mathbf{S}\, \mathbf{w}$$

  subject to

$$\mathbf{w}^\intercal \mathbf{w} = 1$$

- the method of Lagrange multipliers leads to a closed-form solution:

$$\mathbf{S}\, \hat{\mathbf{w}} = \lambda\, \hat{\mathbf{w}}$$

  where each principal component $\hat{\mathbf{w}}$ is an eigenvector of $\mathbf{S}$
- the projection variance equals to the corresponding eigenvalue:

$$\sigma^2 = \hat{\mathbf{w}}^\intercal \mathbf{S} \hat{\mathbf{w}} = \hat{\mathbf{w}}^\intercal \lambda\, \hat{\mathbf{w}} = \lambda \cdot \|\hat{\mathbf{w}}\|^2 = \lambda$$

PCA Procedure

1. compute the sample covariance matrix $\mathbf{S}$ from training data
2. calculate the top m eigenvectors of $\mathbf{S}$
3. form $\mathbf{A} \in \mathbb{R}^{m \times n}$ with an eigenvector in a row

$$\mathbf{A} = \begin{bmatrix} - & \hat{\mathbf{w}}_1^\intercal & - \\ - & \hat{\mathbf{w}}_2^\intercal & - \\ & \vdots & \\ - & \hat{\mathbf{w}}_m^\intercal & - \end{bmatrix}_{m \times n}$$

4. for any $\mathbf{x} \in \mathbb{R}^n$, map it to $\mathbf{y} \in \mathbb{R}^m$ as $\mathbf{y} = \mathbf{A}\mathbf{x}$.

A few top eigenvalues usually dominate the total variance in PCA

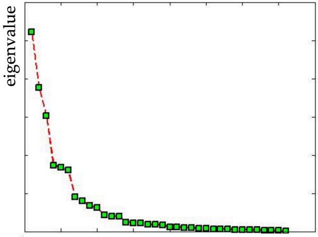


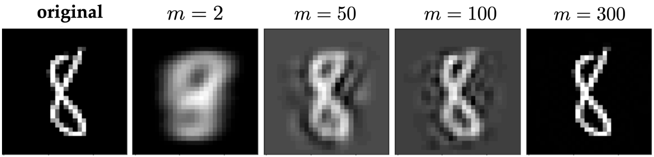
Figure: the distribution of all eigenvalues in PCA

Inverse PCA Transformation

- full PCA without truncation ($m = n$):

$$\tilde{\mathbf{x}} = \mathbf{A}^\intercal \mathbf{y} = \underbrace{\mathbf{A}^\intercal \mathbf{A}}_{\mathbf{I}} \mathbf{x} = \mathbf{x}$$

- truncated PCA ($m < n$):
  1. method 1: $\tilde{\mathbf{x}} = \mathbf{A}^\intercal \mathbf{y} \neq \mathbf{x}$
  2. method 2: $\tilde{\mathbf{x}} = \mathbf{A}^\intercal \mathbf{y} + \big(\mathbf{I} - \mathbf{A}^\intercal \mathbf{A}\big)\bar{\mathbf{x}} \neq \mathbf{x}$



Linear Discriminant Analysis (I)

- class labels are known
- how to linearly project data in order to maximize class separation
- Fisher's linear discriminant analysis (LDA) aims to maximize the ratio:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\intercal \mathbf{S}_b \mathbf{w}}{\mathbf{w}^\intercal \mathbf{S}_w \mathbf{w}}$$

  - *between-class scatter matrix* $\mathbf{S}_b = \sum_{k=1}^{K} \left|C_k\right| \left(\boldsymbol{\mu}_k - \boldsymbol{\mu}\right)\left(\boldsymbol{\mu}_k - \boldsymbol{\mu}\right)^\intercal$
  - *within-class scatter matrix* $\mathbf{S}_w = \sum_{k=1}^{K} \mathbf{S}_k$

PCA does not always maximize class separation

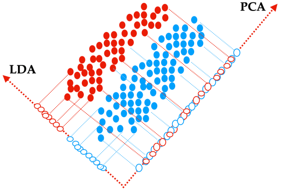


Figure: LDA vs. PCA

Linear Discriminant Analysis (II)

- Fisher's LDA is equivalent to the following constrained optimization:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ \mathbf{w}^\intercal \mathbf{S}_b \mathbf{w}$$

subject to

$$\mathbf{w}^\intercal \mathbf{S}_w \mathbf{w} = 1$$

- LDA projections correspond to the eigenvectors of $\mathbf{S}_w^{-1}\mathbf{S}_b$
- LDA has at most $K-1$ projection directions

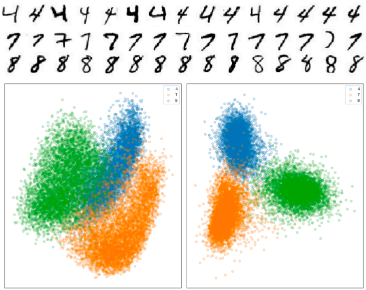
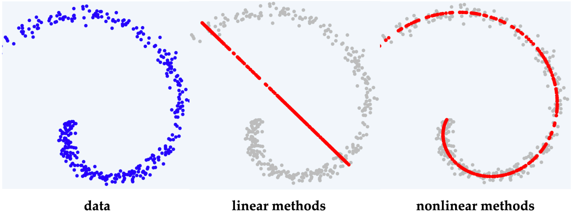


Figure: PCA vs. LDA projections in 2-dimensional spaces

Nonlinear Dimension Reduction (I): Manifold Learning



- **manifolds**: nonlinear topological structures in a lower-dimensional space
- **manifold learning**: identify low-dimensional manifolds using some non-parametric approaches
  - locally linear embedding (LLE)
  - multidimensional scaling (MDS)
  - stochastic neighborhood embedding (SNE)

Locally Linear Embedding (LLE)

- **assumption 1**: a locally linear structure in the high-D space $\mathbf{x}_i \approx \sum_{j \in N_i} w_{ij}\mathbf{x}_j$
- all pair-wise weights can be derived:

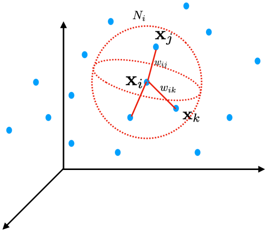
$$\{\hat{w}_{ij}\} = \arg\min_{\{w_{ij}\}} \sum_i \Big\|\mathbf{x}_i - \sum_{j \in N_i} w_{ij}\mathbf{x}_j\Big\|^2$$

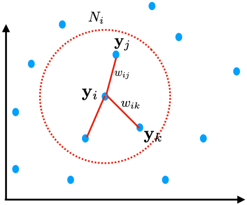
subject to $\sum_j w_{ij} = 1 \quad (\forall i)$

- **assumption 2**: the same locally linear structure is applied to the low-D space

$$\{\hat{\mathbf{y}}_i\} = \arg\min_{\{\mathbf{y}_i\}} \sum_i \Big\|\mathbf{y}_i - \sum_{j \in N_i} \hat{w}_{ij}\mathbf{y}_j\Big\|^2$$

- closed-form solutions exist for LLE





Multidimensional Scaling (MDS)

- preserve all pair-wise distances when projecting from high-D to low-D
- all pair-wise distances in high-D:
  $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \quad (\forall i, j)$
- MDS computes all low-D projections:

$$\{\hat{\mathbf{y}}_i\} \;=\; \arg\min_{\{\mathbf{y}_i\}} \sum_i \sum_{j>i} \left(\frac{\|\mathbf{y}_i - \mathbf{y}_j\| - d_{ij}}{d_{ij}} \right)^2$$

- isometric feature mapping (Isomap)
  - only compute pairwise distances for nearby vectors in high-D
  - form a sparse graph in the high-D space

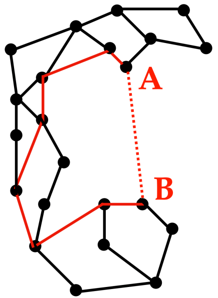


Figure: Isomap uses the shortest path in the weighted graph for d_{ij}

Stochastic Neighborhood Embedding (SNE)

- define a conditional probability distribution in high-D:

$$p_{ij} = \frac{\exp\left(-\gamma_i \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)}{\sum_k \exp\left(-\gamma_i \|\mathbf{x}_i - \mathbf{x}_k\|^2\right)} \qquad (\forall i, j \;\; i \neq j)$$

- similarly define a conditional probability distribution in low-D:
  - SNE (stochastic neighbor embedding):

$$q_{ij} = \frac{\exp\left(-\|\mathbf{y}_i - \mathbf{y}_j\|^2\right)}{\sum_k \exp\left(-\|\mathbf{y}_i - \mathbf{y}_k\|^2\right)} \qquad (\forall i, j)$$

  - t-SNE (t-distributed stochastic neighbor embedding):

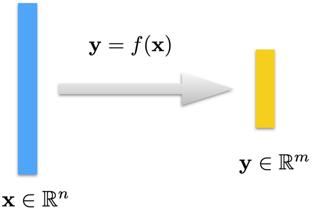
$$q_{ij} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2\right)^{-1}} \qquad (\forall i, j)$$

- low-D projections are derived by

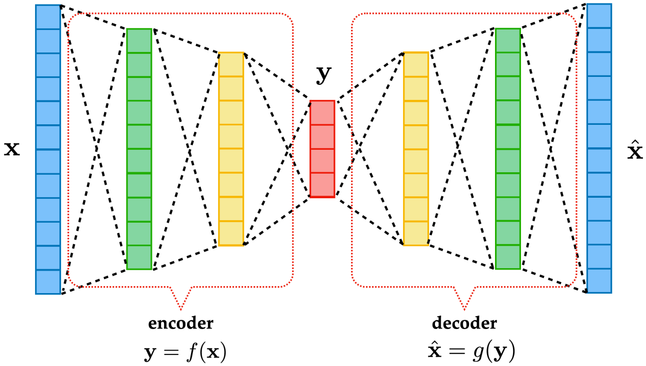
$$\{\hat{\mathbf{y}}_i\} = \arg\min_{\{\mathbf{y}_i\}} \sum_i \mathsf{KL}\left(P_i \,\|\, Q_i\right) = \arg\min_{\{\mathbf{y}_i\}} \sum_i \sum_j p_{ij} \ln \frac{p_{ij}}{q_{ij}}$$

Nonlinear Dimension Reduction (II): Neural Networks

- use neural networks as parametric models for the nonlinear mapping function $\mathbf{y} = f(\mathbf{x})$ in dimensionality reduction
- **autoencoder**
  - unsupervised: does not require class labels
  - nonlinear extension of PCA
- **bottleneck features**
  - supervised: requires class labels
  - nonlinear extension of LDA

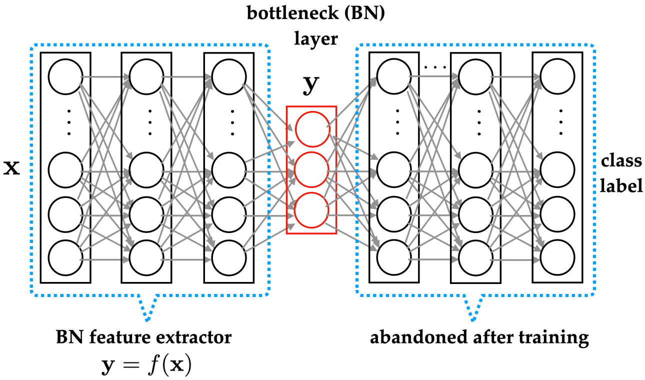


Autoencoder



neural networks are learned to minimize the difference between inputs and outputs: $\|\hat{\mathbf{x}} - \mathbf{x}\|^2$

Bottleneck Features



neural networks are learned to project inputs $\mathbf{x}$ to any given class labels