No. 3

Machine Learning: Data vs Feature vs Model

Hui Jiang

Department of Electrical Engineering and Computer Science
York University, Toronto, Canada

Machine Learning Framework

in-domain

the more
the better

compact
representative

generative
vs
discriminative

feature engineering

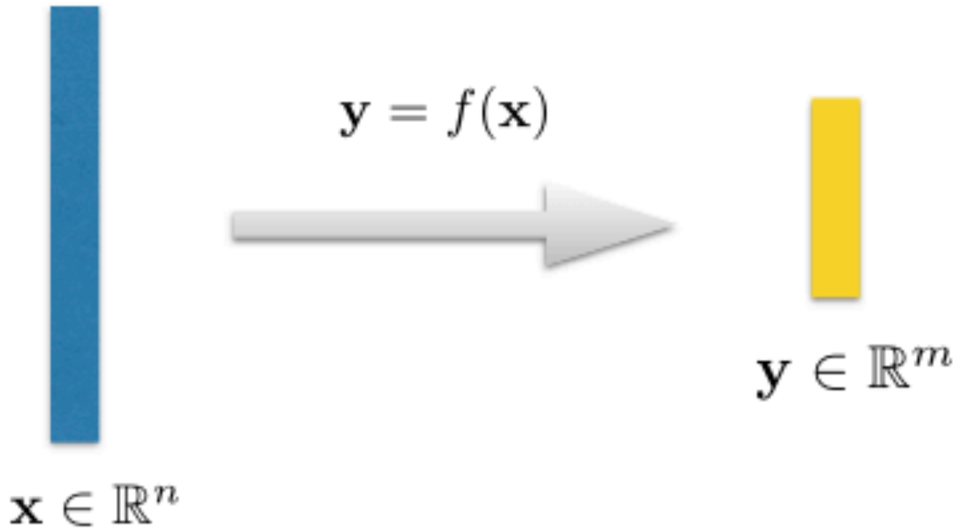
Outline

- The curse of dimensionality
- Feature Extraction
  - Linear:
    - Principal Component Analysis (PCA)
    - Linear Discriminant Analysis (LDA)
  - Nonlinear (manifold learning):
    - Multi-Dimensional Scaling (MDS)
    - Stochastic Neighbourhood Embedding (SNE)
    - Locally Linear Embedding (LLE)
    - IsoMap
    - Neural Network Bottlenecks
- Data Virtualization

The Curse of the Dimensionality

- Feature engineering ==> high-dimension feature vectors
- *"The curse of the dimensionality"*
- Highly correlated among dimensions
- Distance in high-dimension space is error-prone
- Intuitions fail in high dimensions:
  - High-D Gaussian distribution: most mass not near mean
  - Most mass of a high-D sphere is in the surface
  - Most points in high-D cube/sphere is more closer to the surface than their closest neighbours

Dimension Reduction



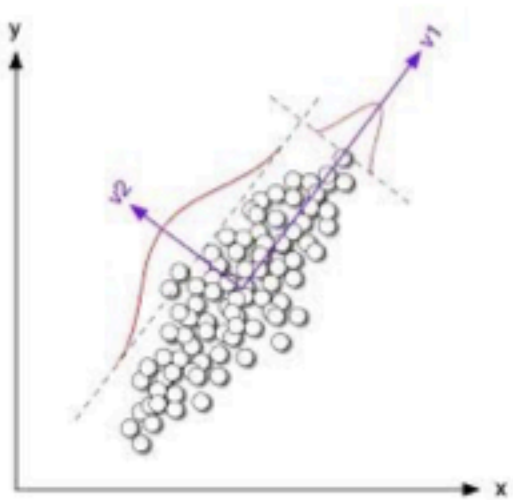
- Forms of the mapping function

  $$f(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m \quad (m \ll n)$$

  - **linear** function: $\mathbf{y} = A\mathbf{x} + \mathbf{b}$
  - **nonlinear** function: piecewise linear functions, neural networks

- Criterion to learn *f(.)*:
  - PCA; LDA
  - Manifold learning
  - Bottleneck; auto-encoder

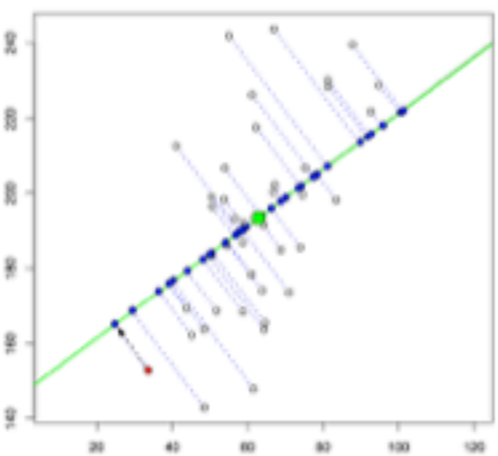
Principal Component Analysis (PCA)



- Two equivalent explanations:

  1. Maximum variance formulation

  2. Minimum-error formulation



Principal Component Analysis (PCA)

- A little math: maximize variance in linear projection

the variance of the projected data is given by

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n - \mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}\right\}^2 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$$

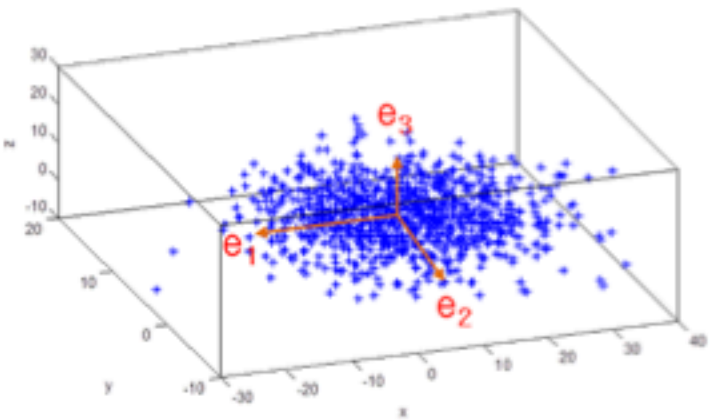
$\mathbf{S}$ is the data covariance matrix defined by

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}.$$

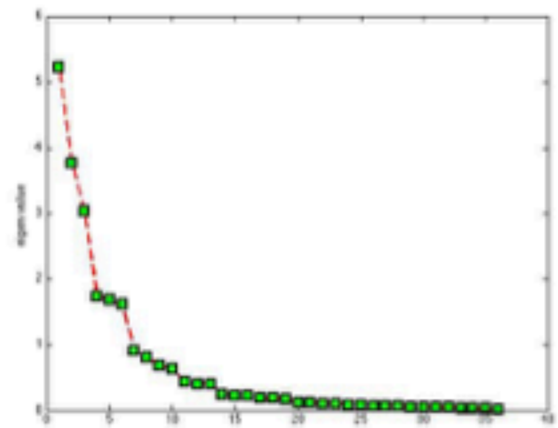
Principal Component Analysis (PCA)

Variance (energy) distribution among principal components

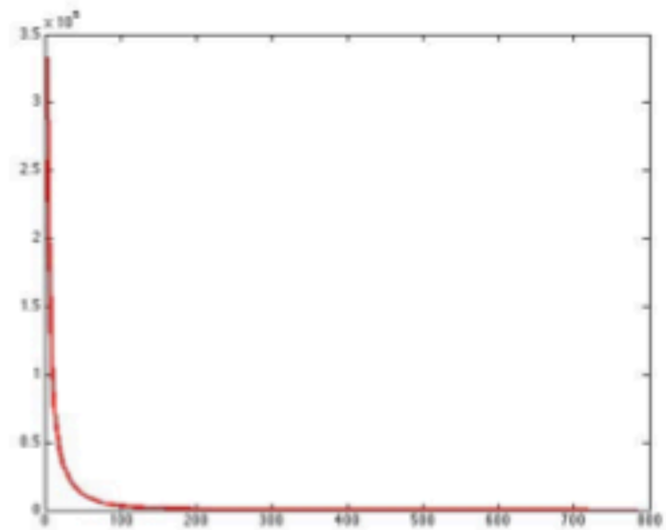
high-dimension data



variance (energy) along dimensions after PCA



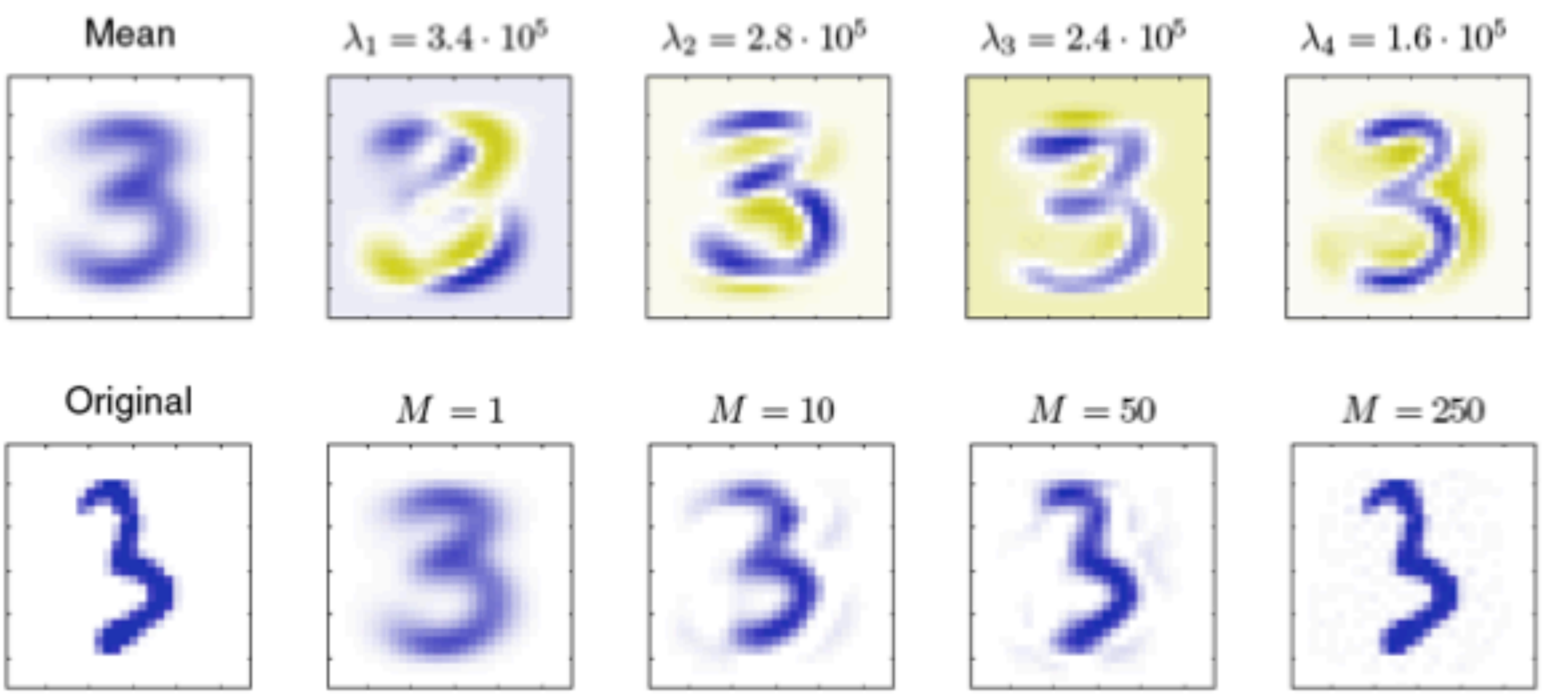
MNIST



Applications of PCA

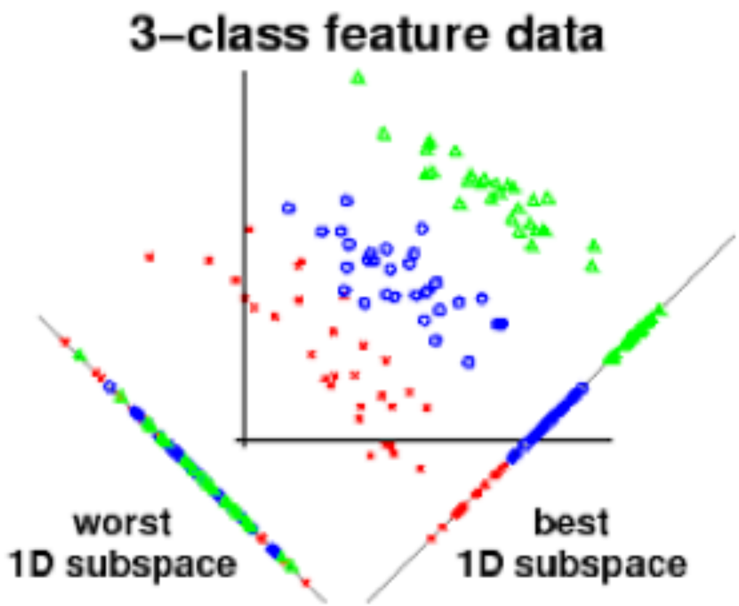
- Dimensionality reduction
- Reconstruct high-dimension data from the lower-dimension PCA features

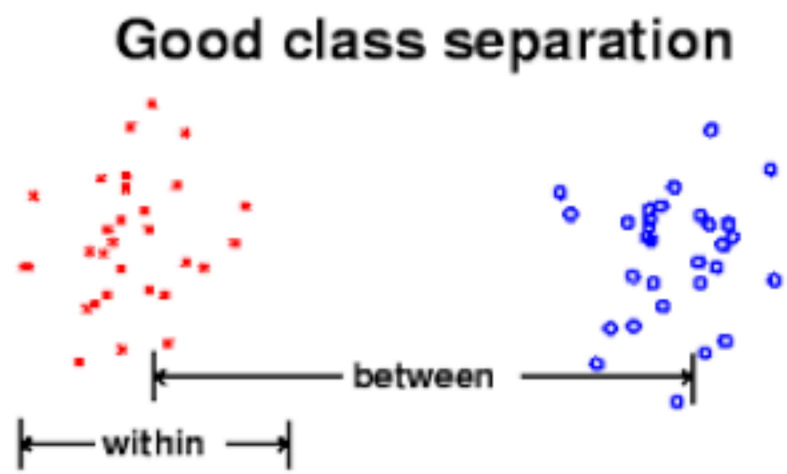
$$\begin{aligned}\widetilde{\mathbf{x}}_n &= \sum_{i=1}^{M}(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i)\mathbf{u}_i + \sum_{i=M+1}^{D}(\overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i)\mathbf{u}_i \\ &= \overline{\mathbf{x}} + \sum_{i=1}^{M}\left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i\end{aligned}$$



Linear Discriminant Analysis (LDA)

- Fisher's linear discriminant: maximize the class separation
- Supervised dimensionality reduction: needs class labels

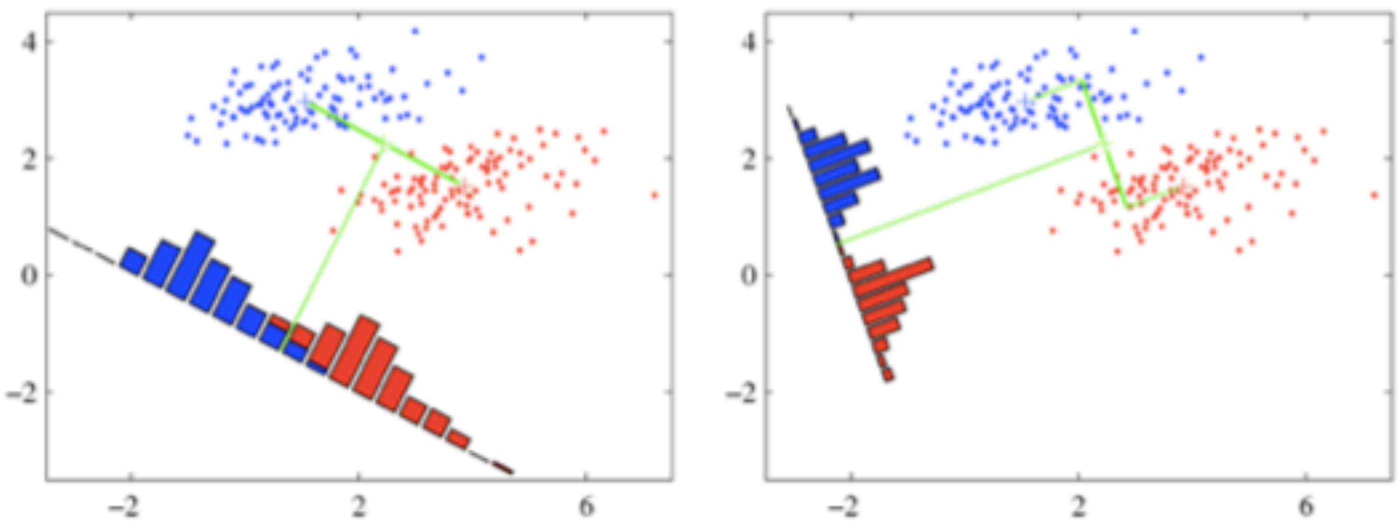




Linear Discriminant Analysis (LDA)

- Fisher's linear discriminant: maximize the class separation using within-class and between-class covariance matrices

- maximizing a ratio defined as:

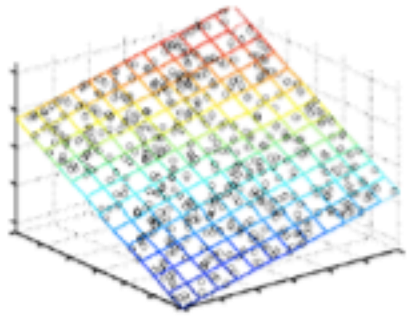


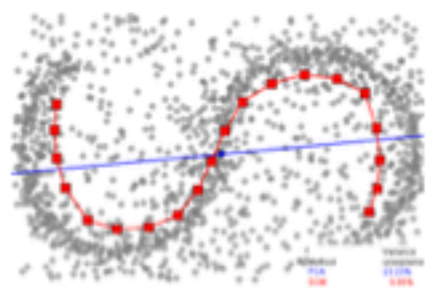
$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}}$$

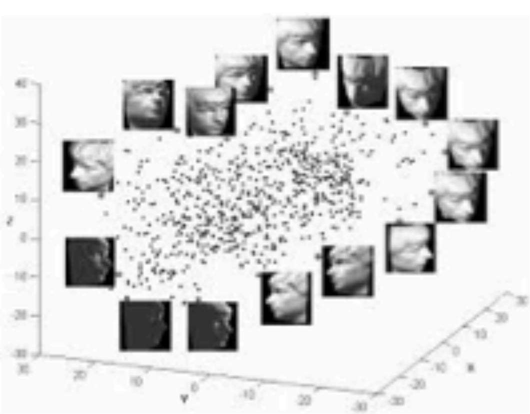
Related Work

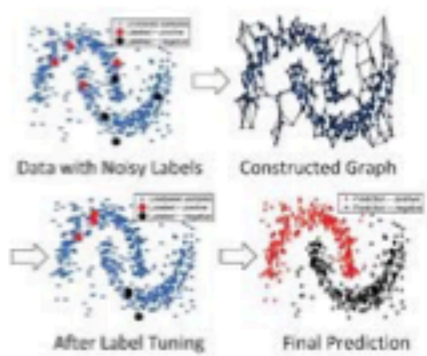
- Probabilistic PCA (PPCA) *(Tipping & Bishop, 1999a)*
- Bayesian PCA, Kernel PCA, Sparse PCA
- Mixture of PPCA *(Tipping & Bishop, 1999b)*
- Factor Analysis
- Heteroscedastic LDA (HLDA/HDA) *(Kumar & Andreous, 1998)*
- Independent Component Analysis (ICA) *(Hyvarinen & Oja, 2000)*
- Projection Pursuit (Friedman & Tukey, 1974)

Manifold Learning: nonlinear dimensionality reduction

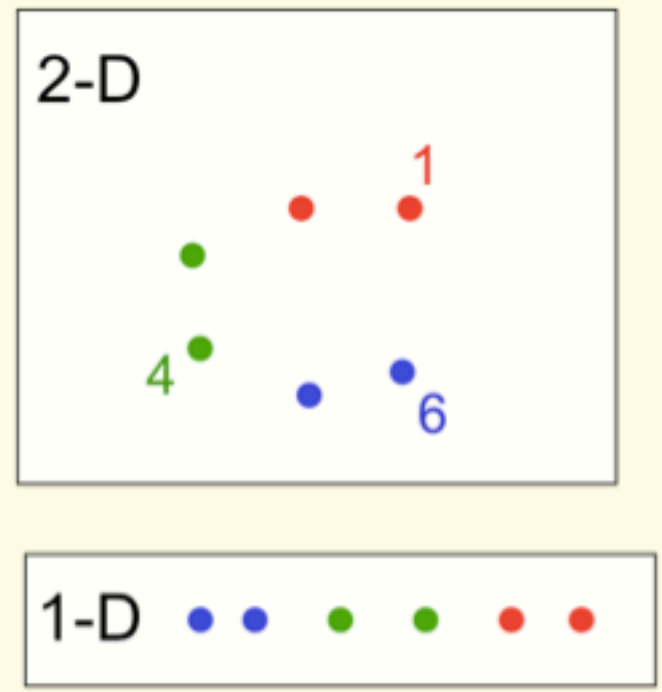








If we measure distances along the manifold, d(1,6) > d(1,4)



Multi-Dimensional Scaling (MDS)

- Preserve between-object distances as much as possible

$$Cost = \sum_{i<j} (d_{ij} - \hat{d}_{ij})^2$$

$$d_{ij} = \| x_i - x_j \|^2$$

$$\hat{d}_{ij} = \| y_i - y_j \|^2$$

Sammon Mapping

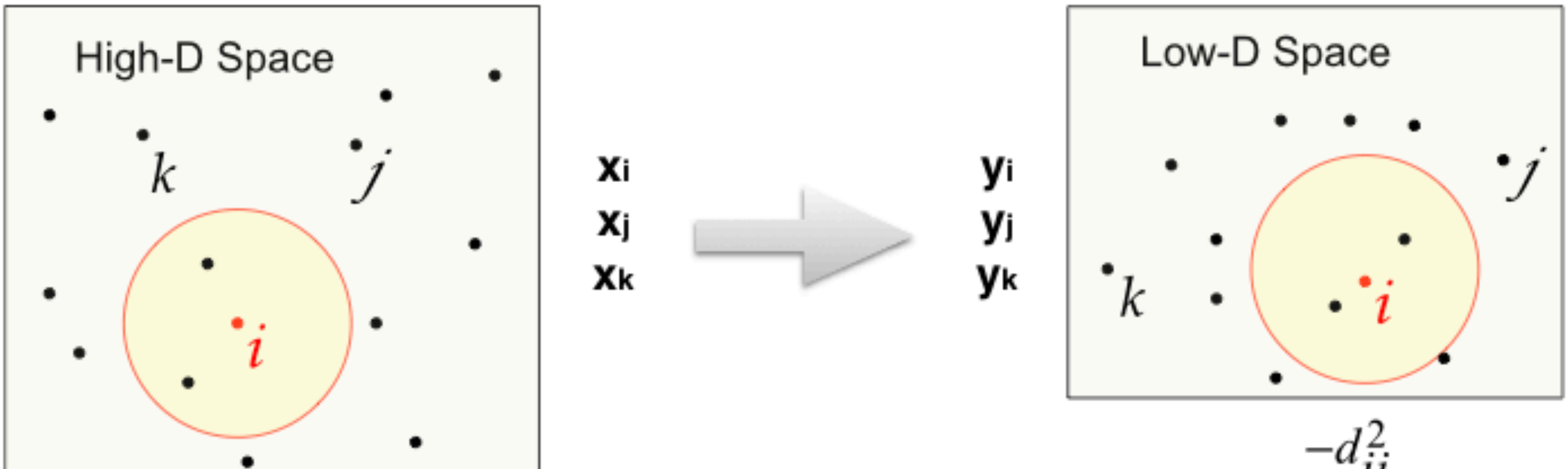
high-D distance

low-D distance

$$Cost = \sum_{ij} \left(\frac{\| \mathbf{x}_i - \mathbf{x}_j \| - \| \mathbf{y}_i - \mathbf{y}_j \|}{\| \mathbf{x}_i - \mathbf{x}_j \|} \right)^2$$

Stochastic Neighbourhood Embedding (SNE)

- A probabilistic local mapping method



$$p_{j|i} = \frac{e^{-d_{ij}^2 / 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 / 2\sigma_i^2}}$$

$$q_{j|i} = \frac{e^{-d_{ij}^2}}{\sum_k e^{-d_{ik}^2}}$$

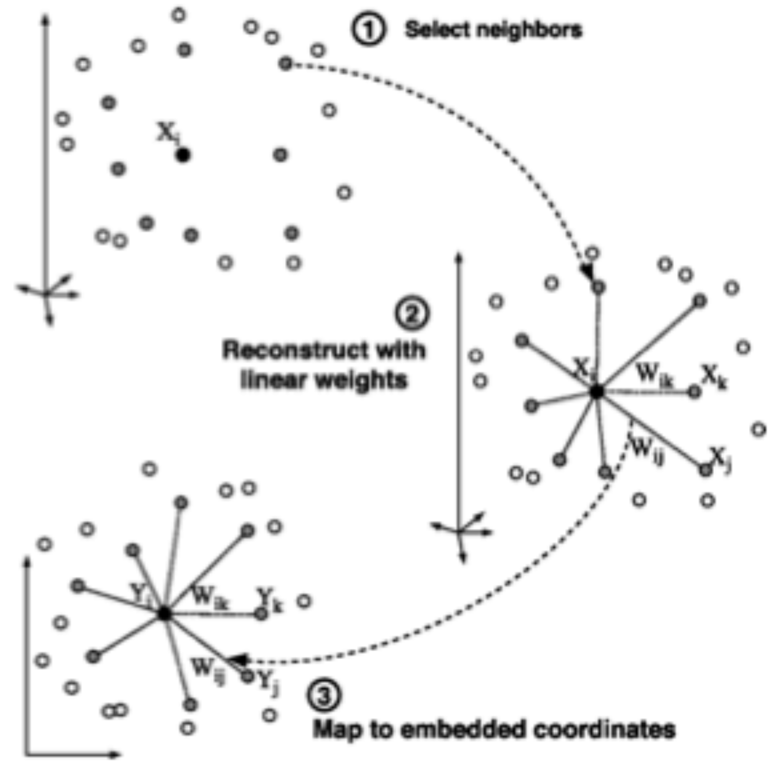
$$Cost = \sum_i KL(P_i \,||\, Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Locally Linear Embedding (LLE)

- **Maps that preserve local geometry**: local configurations of points in the low-dimensional space resemble the local configurations in the high-dimensional space.
- Represent a point as a weighted average of nearby points, the weights describe the local configuration:

$$\mathbf{x}_i \approx \sum_j w_{ij} \mathbf{x}_j$$

- Use the data points in high-dimension to determine the local weights, then try to re-construct them from its neighbours in low-dimension.



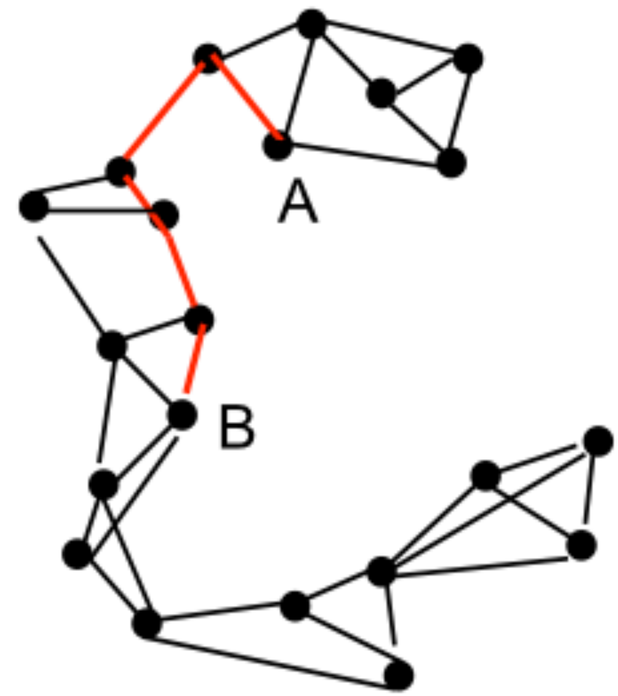
$$Cost = \sum_i \| \mathbf{x}_i - \sum_{j \varepsilon N(i)} w_{ij} \mathbf{x}_j \|^2, \qquad \sum_{j \varepsilon N(i)} w_{ij} = 1$$

fixed weights

$$Cost = \sum_i \| \mathbf{y}_i - \sum_{j \varepsilon N(i)} w_{ij} \mathbf{y}_j \|^2$$

IsoMap: Local MDS without local optima

- Connect each datapoint to its K nearest neighbours in the high-dimensional space.
- Put the true Euclidean distance on each of these links.
- Then approximate the manifold distance between any pair of points as the shortest path in this "neighbour graph".



Data Virtualization

- Project data into 2-D or 3D space for virtualization
- Popular approaches:
  - **t-SNE**: https://lvdmaaten.github.io/tsne/
  - **Isomap**: http://isomap.stanford.edu/