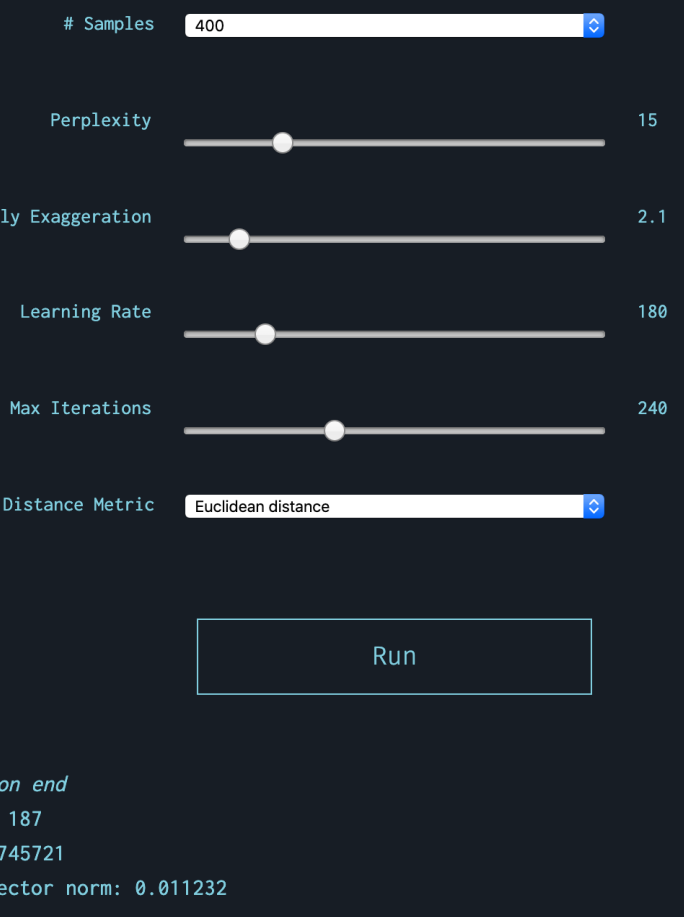
example with MNIST (raw data)



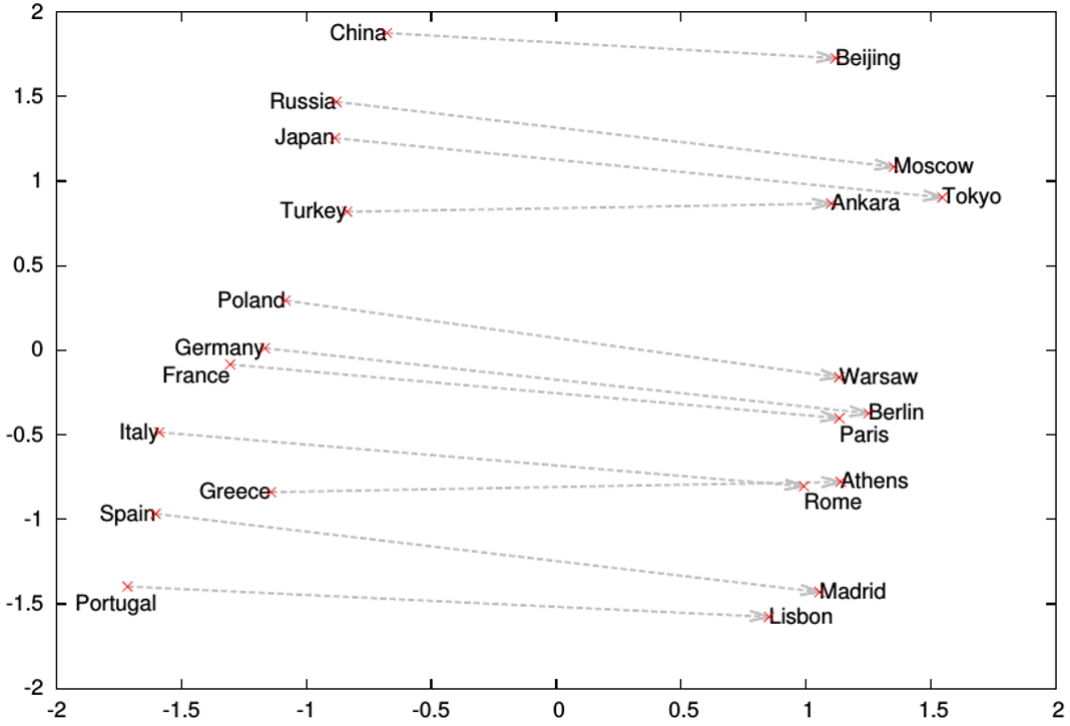
“Unsupervised” Learning:
Dimensionality Reduction

Agenda

- Dimensionality Reduction (DR) -> Embedding
- Why reduce dimensionality?
- About Data Dimensionality
- Approaches to DR
- Feature Selection
- Linear DR
- Unsupervised DR – PCA
- Non-linear DR
- Non-parametric DR – Embedding
- Summary

Dimensionality Reduction -> Embedding

- We cannot see data in high dimensional space!
- Could we see it in a 2-D/3-D space? -> Embedding
- How to embed high-dim data in low-dim space?
- Important concept: Preserving the original similarity!



Dimensionality Reduction

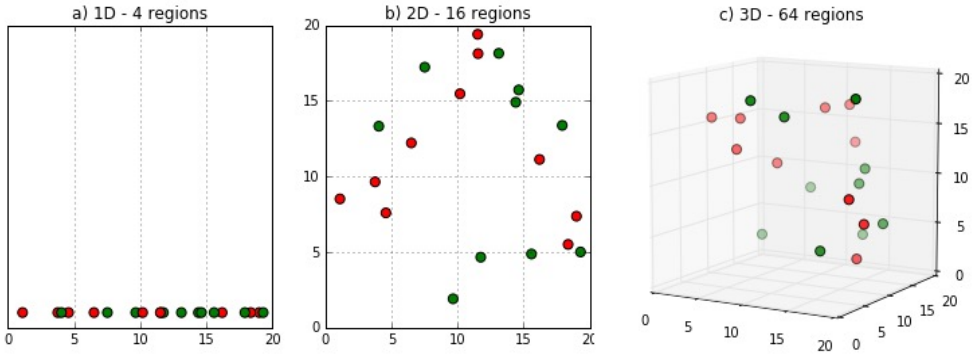
- Many modern data domains involve huge numbers of features/dimensions
  - Documents: thousands of words, millions of bigrams
  - Images: thousands to millions of pixels
  - Genomics: thousands of genes, millions of DNA polymorphisms
  - E-commerce: tens to thousands of clicks, millions of associated information
- DR is a special unsupervised learning!

Why reduce dimensions?

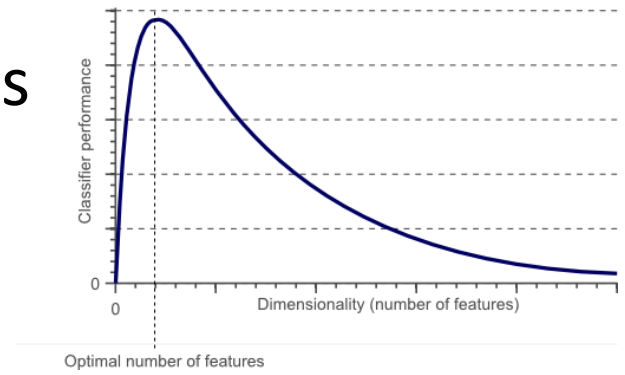
- High dimensionality has many costs
  - Redundant and irrelevant features degrade performance of some ML/analytics algorithms
  - Difficulty in interpretation and visualization
  - Computation may become infeasible
    - what if your algorithm scales as $O(n^3)$ where n is the number of features?
  - Curse of dimensionality
    - refers to various phenomena that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse

Curse of Dimensionality

- Why do we need to reduce dimensions?
- The curse of dimensionality!
  - Higher dimensions implies more empty space →Hence, more data required to maintain a certain data density

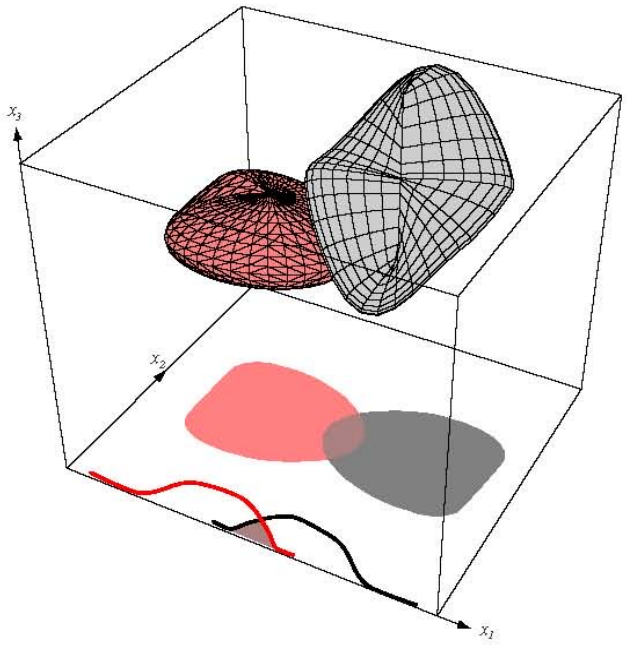


  - Models learnt in higher dimensions often overfit the data



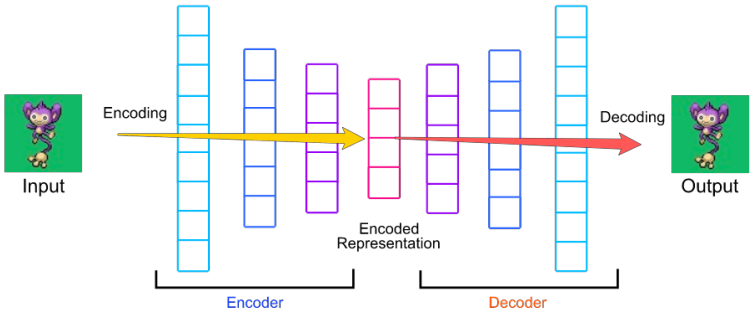
About Data Dimensionality

- From an intuitive point of view, increasing the number of features should lead to better performance (as a result of better characterization of data; cf. the thinking question of DT).
- In practice, the inclusion of more features leads to worse performance (i.e., curse of dimensionality).
- The number of training examples required increases exponentially with dimensionality.



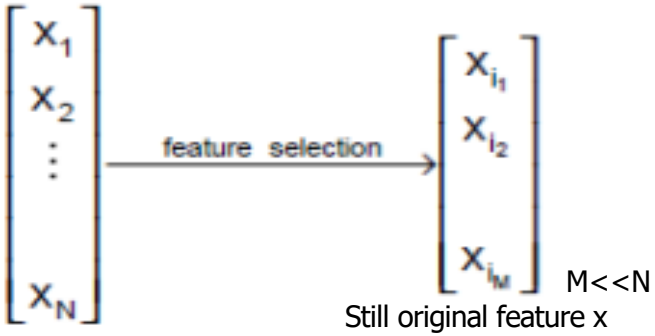
Approaches to Dimensionality Reduction

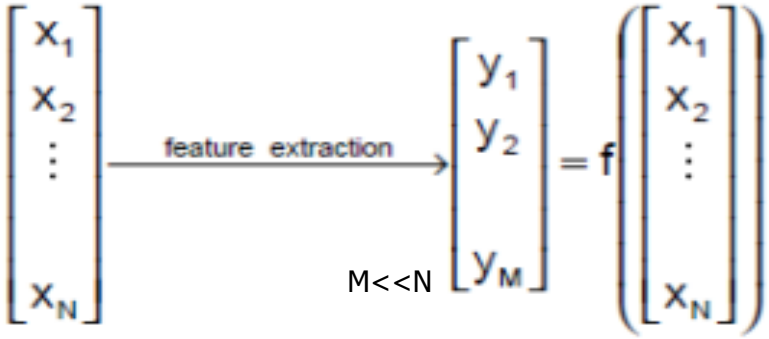
- Feature selection
  - Select subset of existing features (without modification)
- Feature extraction/transformation - Combining (mapping) existing features into smaller number of new/alternative features
  - Linear combination (projection)
  - Nonlinear combination
- Deep learning: Autoencoder



Feature Selection vs Extraction

- Feature selection: Choosing $k<d$ important features, ignoring the remaining $d - k$
  - Subset selection algorithms
- Feature extraction: Project the original x_i , i =1,...,d dimensions to new $k<d$ dimensions, z_j , j =1,...,k
  - Principal components analysis (PCA), linear discriminant analysis (LDA), factor analysis (FA)





- Feature selection can directly keep the physical meaning of features and feature extraction can be made more sophisticated but interpretation is not straightforward!

Feature Selection: Subset Selection

- There are 2^d subsets of d features
- *Forward search methods*: Add the best feature at each step
  - Set of features F initially $\emptyset$.
  - At each iteration, find the best new feature

    $j = \text{argmin}_i E (F \cup x_i)$
  - Add x_j to F if $E (F \cup x_j) < E (F)$
  - Greedy hill climbing approach

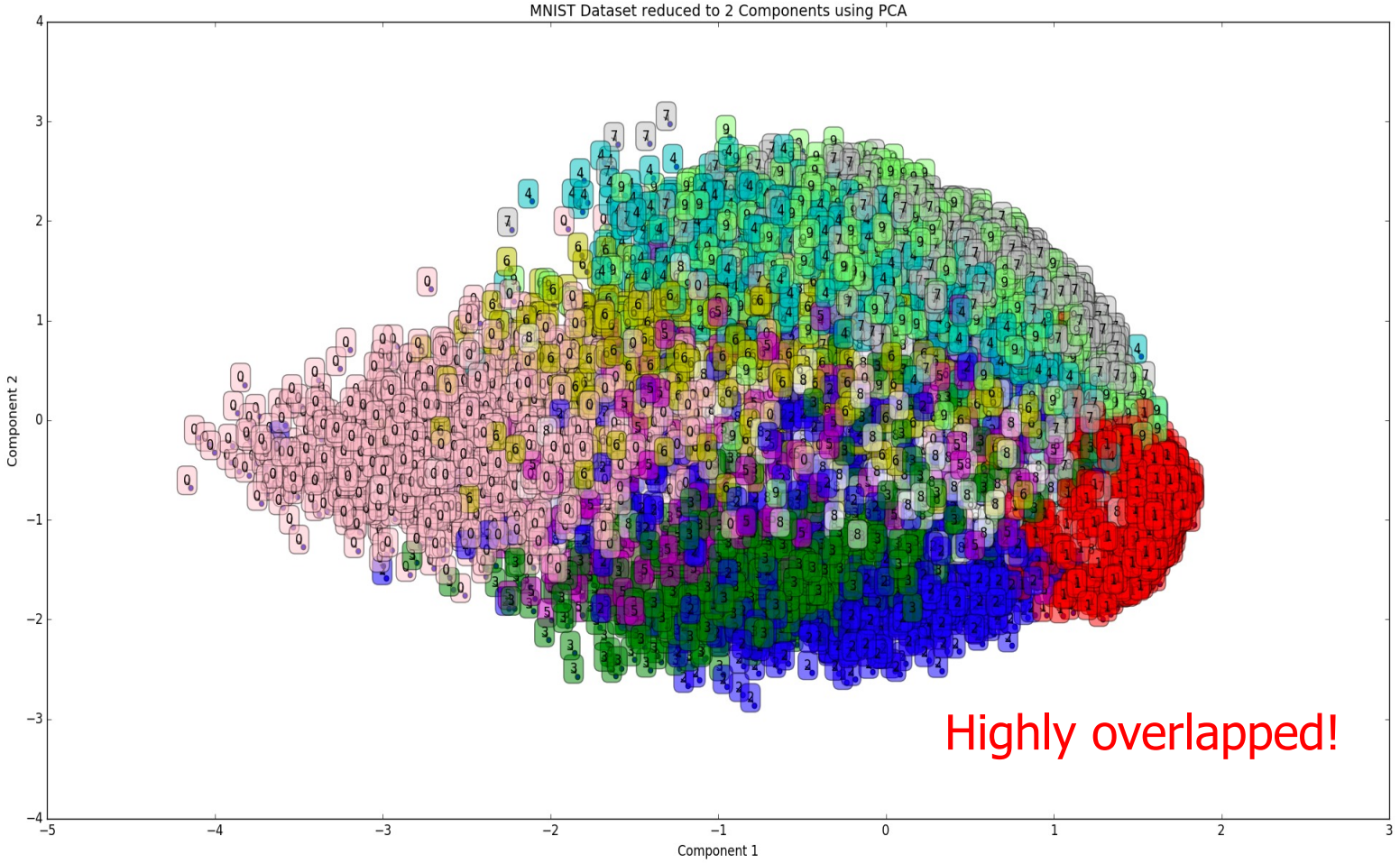
- *Backward search methods*: Start with all features and remove one at a time, if possible.
- *Floating search methods:* (Add k, remove l)

Linear dimensionality reduction

- Linearly project *n*-dimensional data onto a *k*-dimensional space
  - $k < n$, often $k << n$
  - Example: project 10^4-D space of words into 3 dimensions
  - Example: project MNIST 28x28 (784-D) image pixels into 2 dimensions.
- There are infinitely many *k*-dimensional subspaces we can project the data onto.
- Which one should we choose?

A MNIST example

10 Handwritten digits

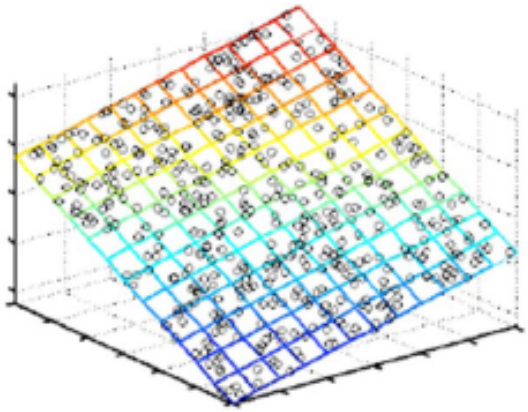


Linear dimensionality reduction for both unsupervised and supervised tasks

- Best *k*-dimensional subspace for projection depends on task
  - Unsupervised: retain as much data variance as possible
    - Example: principal component analysis (PCA)
  - Classification: maximize separation among classes (like SVM)
    - Example: linear discriminant analysis (LDA)
    - DR is not limited to unsupervised learning!
  - Regression: maximize correlation between projected data and response variable
    - Example: partial least squares (PLS)

Unsupervised dimensionality reduction

- Consider data without class labels
- Try to find a more compact representation of the data



$3d \Rightarrow 2d$

- Assume that the high dimensional data actually resides in a inherent low-dimensional space
- Additional dimensions are just random noise
- Goal is to recover these inherent dimensions and discard noise dimensions

Unsupervised dimensionality reduction

- Idea: represent data in terms of basis vectors in a lower dimensional space which is embedded within the original space.

(1) Higher-dimensional space representation:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$v_1, v_2, ..., v_N$ is a basis of the N-dimensional space

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_N \end{bmatrix}$$

(2) Lower-dimensional sub-space representation:

$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

$u_1, u_2, ..., u_K$ is a basis of the K-dimensional space

$$y = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_K \end{bmatrix}$$

$K<<N$!!!

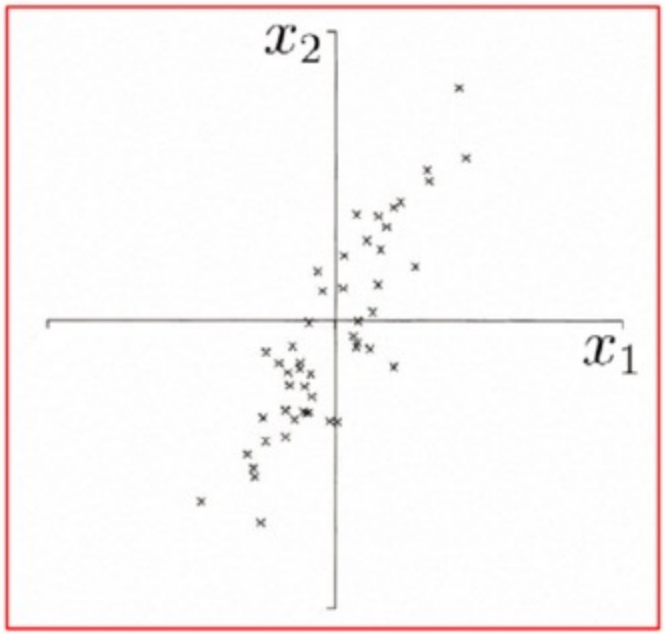
Principal Component Analysis (PCA)

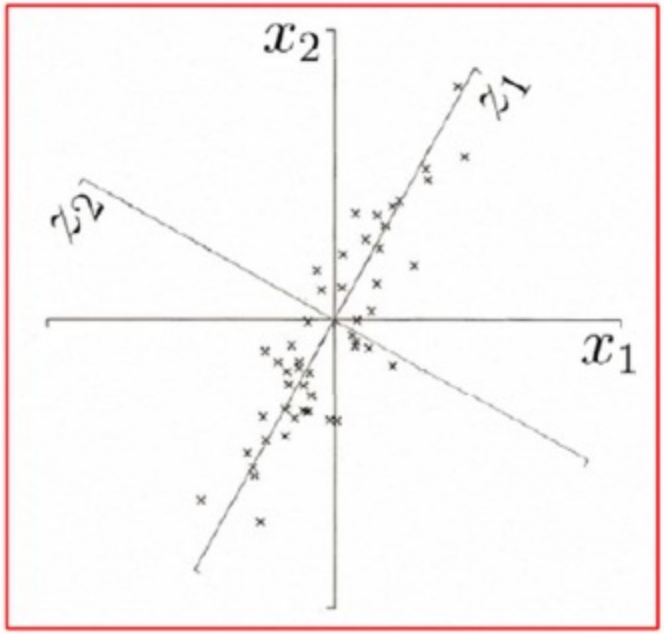
- Widely used method for unsupervised, linear dimensionality reduction

- GOAL: account for variance of data in as few dimensions as possible (using linear projection)

Geometric picture of principal components (PCs)

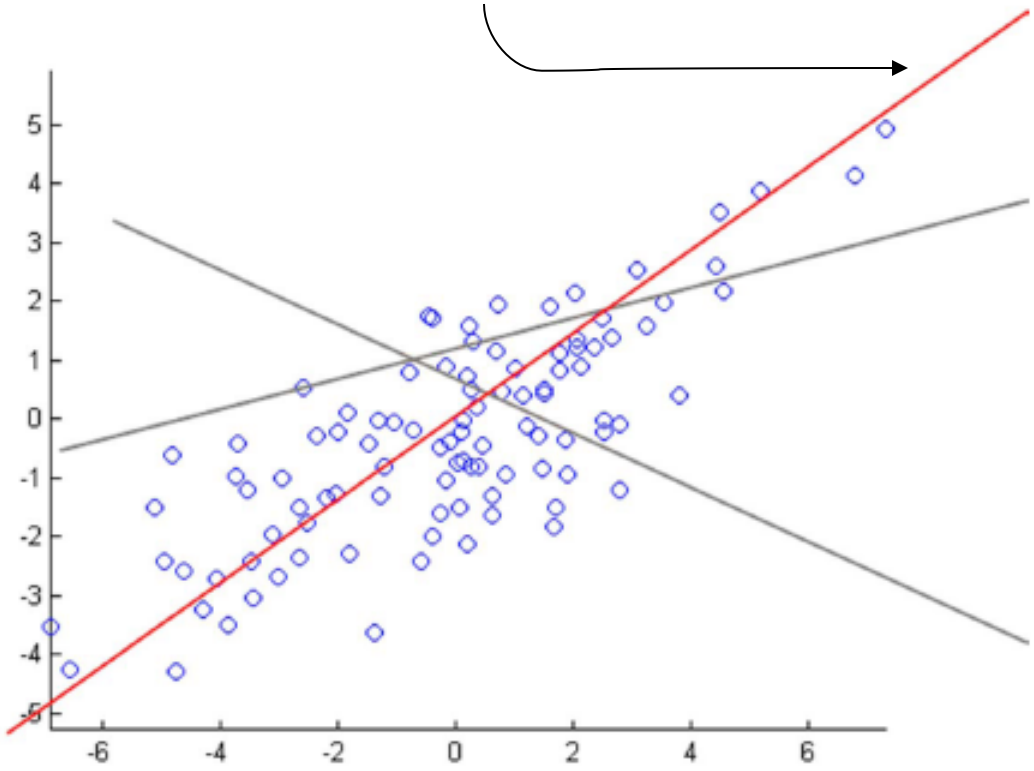
- First PC is the projection direction that maximizes the variance of the projected data
- Second PC is the projection direction that is orthogonal to the first PC and maximizes variance of the projected data





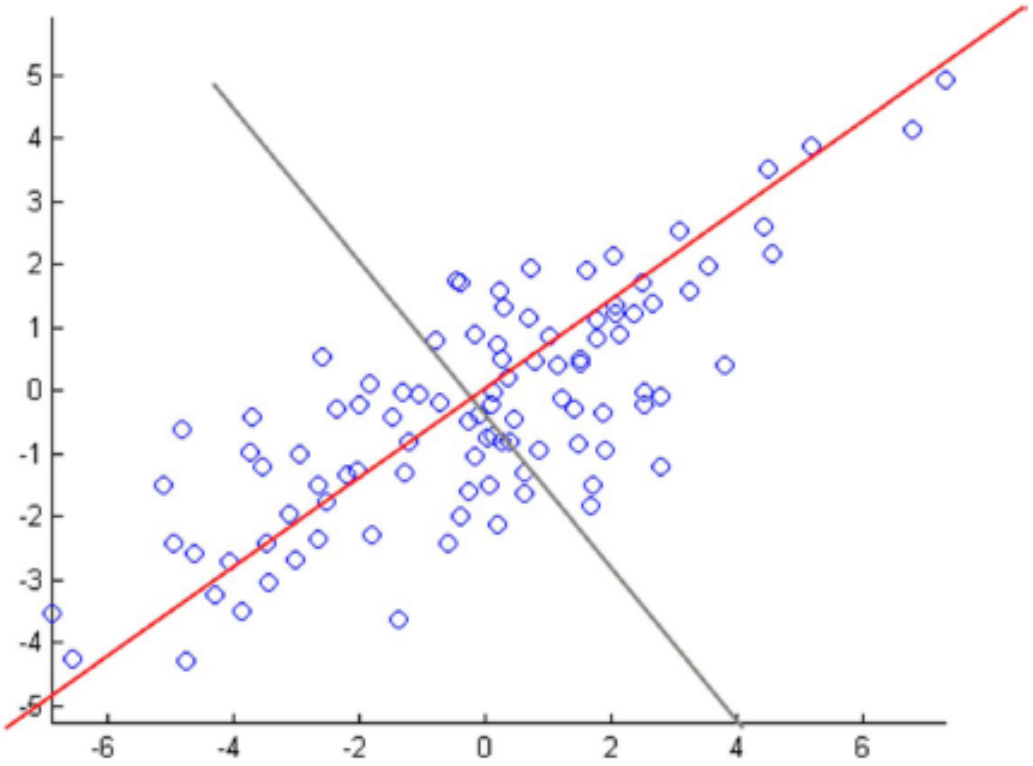
PCA: Conceptual algorithm

- Find a line, such that when the data is projected onto that line, and it has the maximum variance.



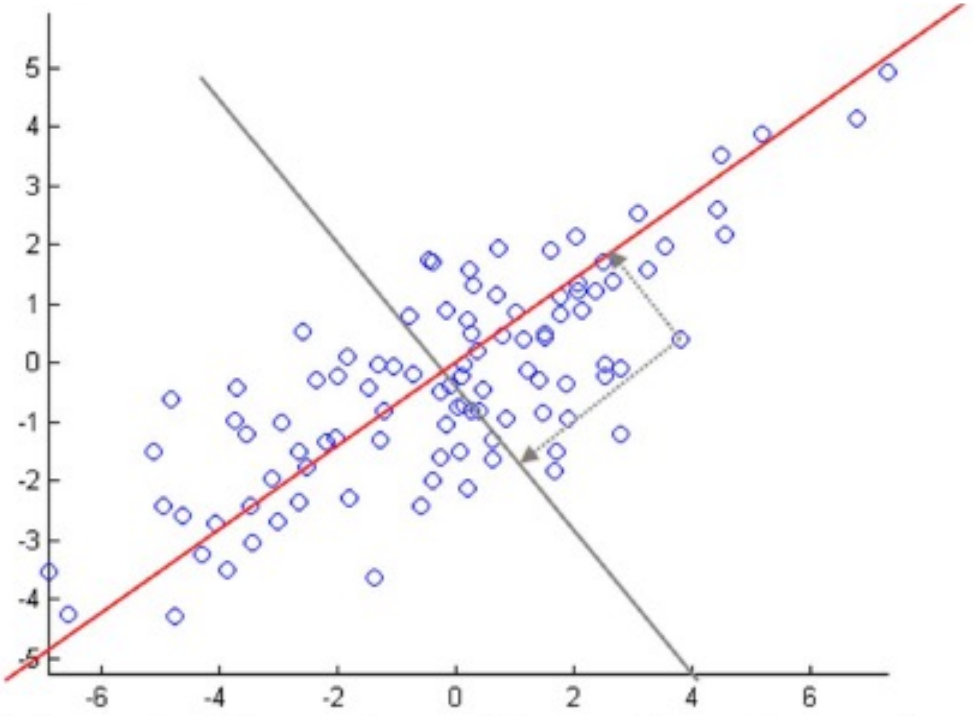
PCA: Conceptual algorithm

- Find a second line, orthogonal to the first, that has maximum projected variance.



PCA: Conceptual algorithm

- Repeat until having *k* orthogonal lines
- The projected position of a point on these lines gives the coordinates in the *k*-dimensional reduced space.



Steps in principal component analysis

- Mean center the data
- Compute covariance matrix Σ

See covariance-matrix

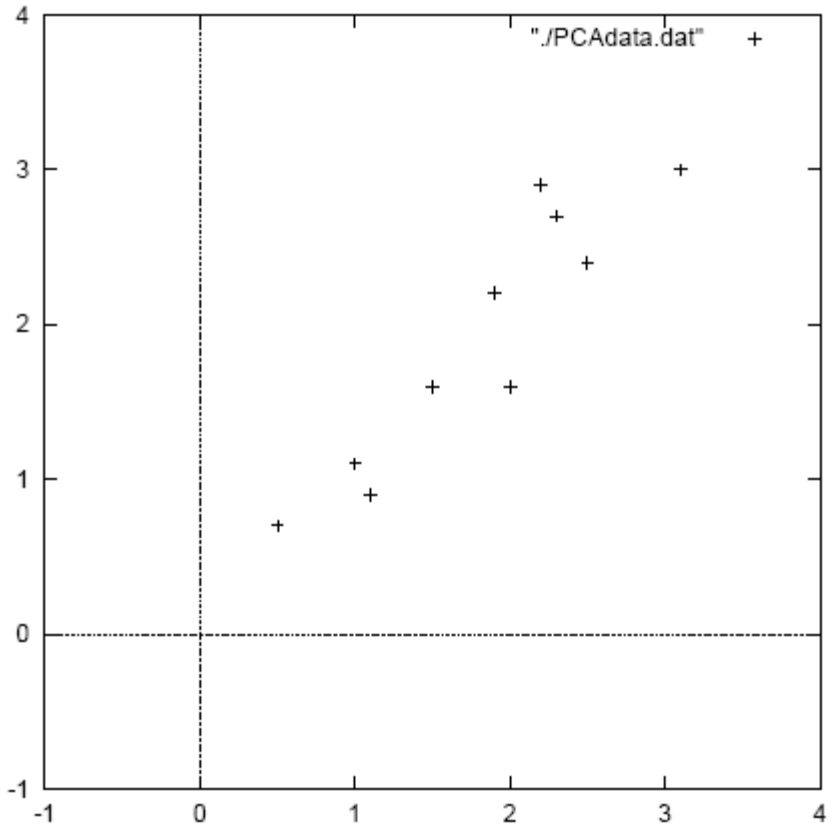
- Calculate eigenvalues and eigenvectors of Σ
  - Eigenvector with largest eigenvalue λ_1 is 1st principal component (PC)
  - Eigenvector with k^{th} largest eigenvalue λ_k is k^{th} PC
  - $\frac{\lambda_k}{\sum_i \lambda_i}$ = proportion of variance captured by k^{th} PC

Applying principal component analysis

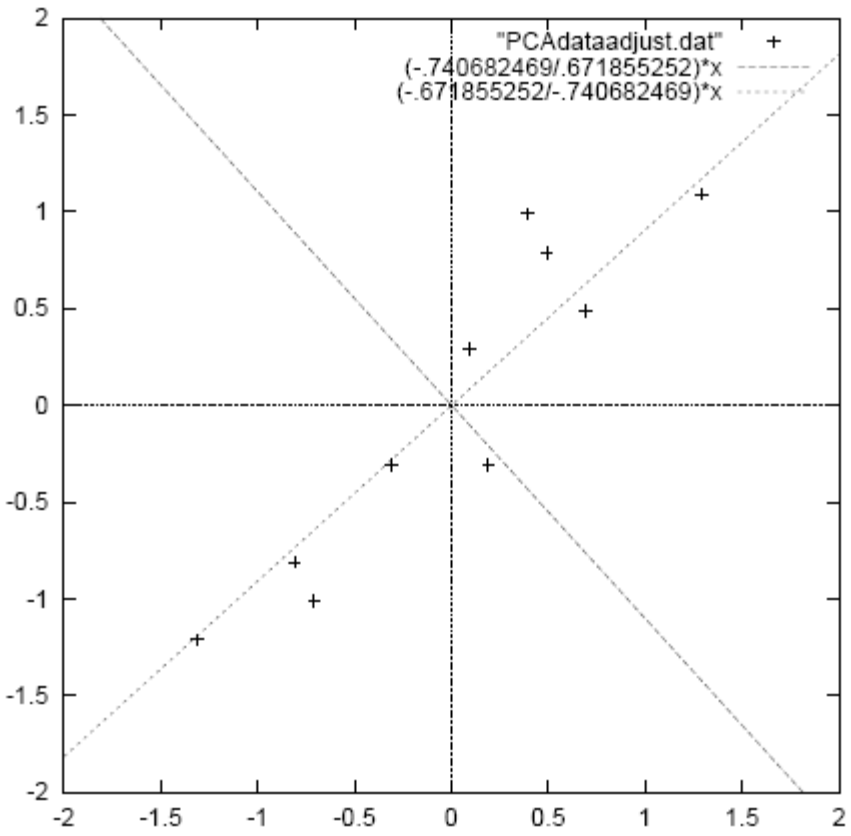
- Full set of PCs comprise a new orthogonal basis for feature space, whose axes are aligned with the maximum variances of original data.
- Projection of original data onto first *k* PCs gives a reduced dimensionality representation of the data.
- Transforming reduced dimensionality projection back into original space gives a reduced dimensionality *reconstruction* of the original data.
- Reconstruction will have some error, but it can be small and often is acceptable given the other benefits of dimensionality reduction.

PCA example (1)

original data



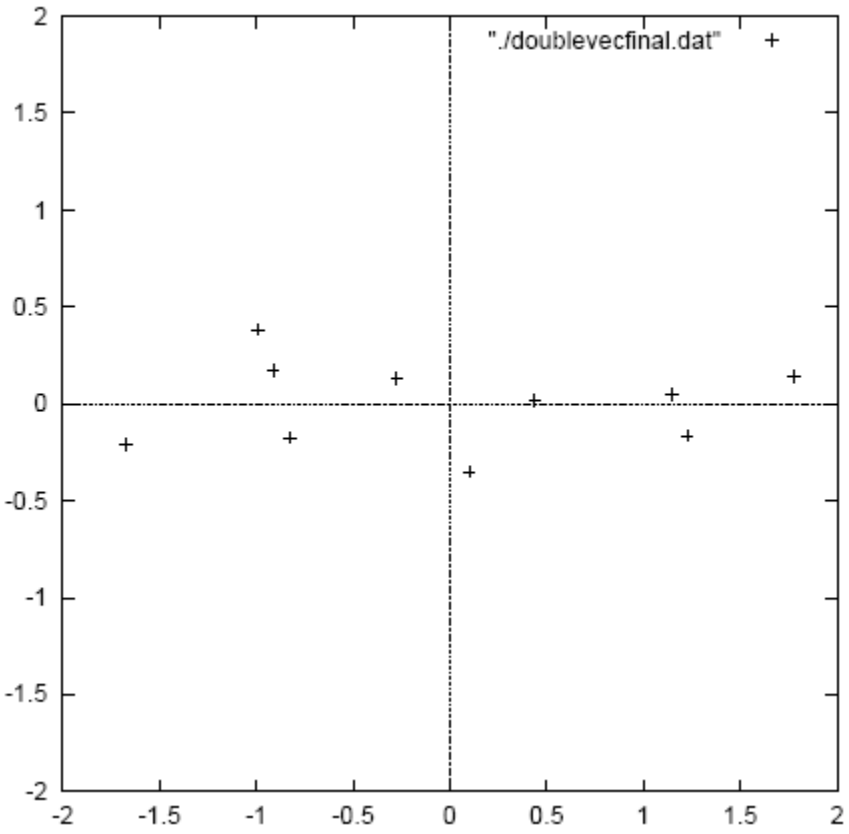
mean centered data with PCs overlayed



Original feature space

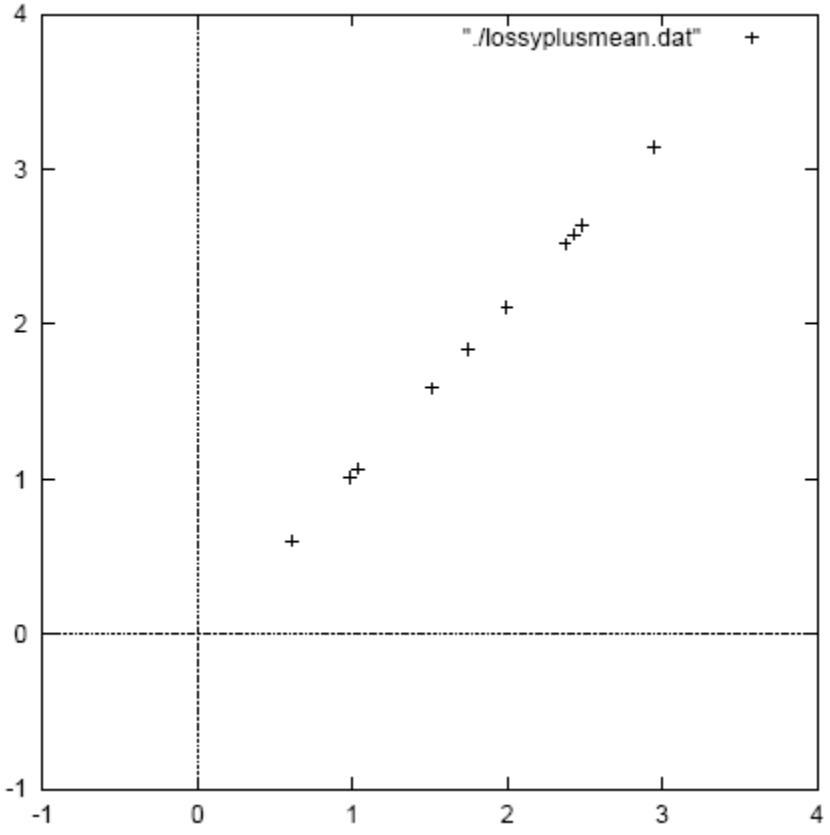
PCA example (1)

original data projected into full PC space



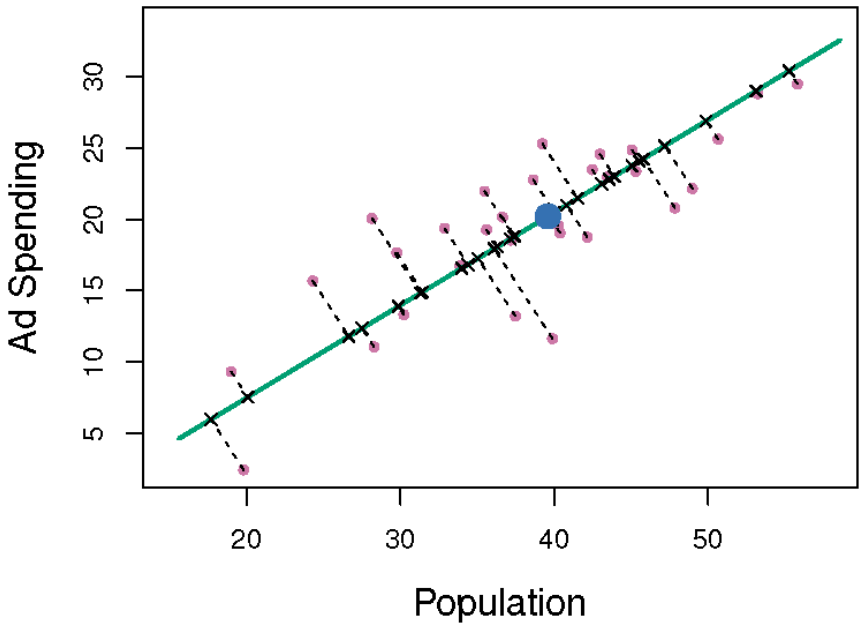
Projected/Transformed PC space

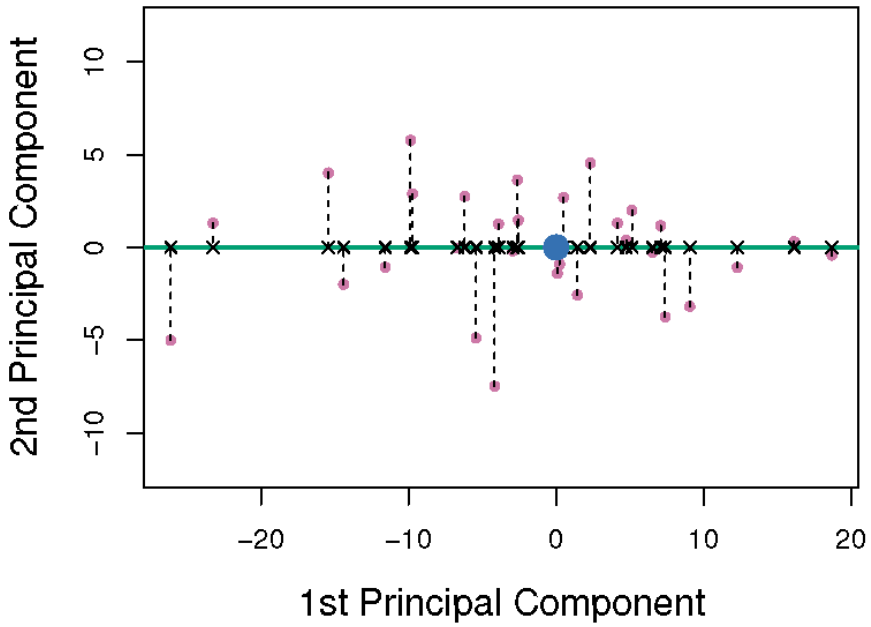
original data reconstructed using only a single PC



Back to Original feature space

Another PCA example



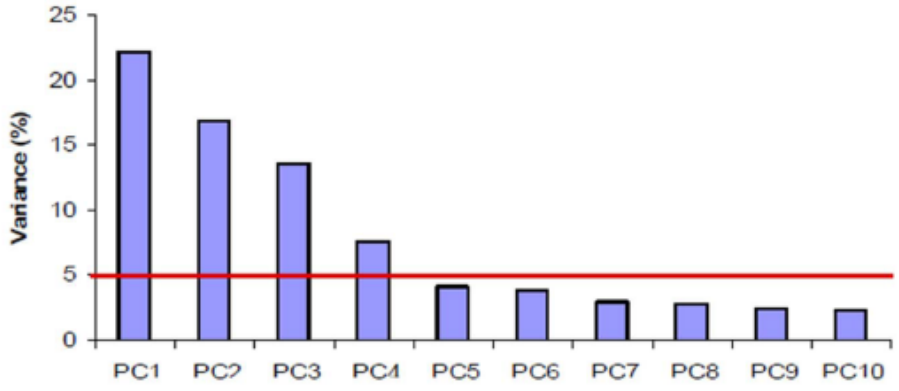


Original feature space

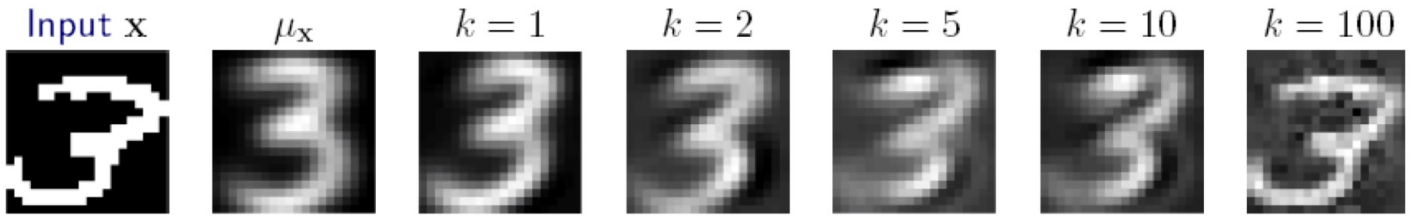
Projected/Transformed PC space

PCA: Choosing the dimension *k*

- Calculate the covariance matrix of the data S
- Calculate the eigen-vectors/eigen-values of S
- Rank the eigen-values in decreasing order
- Select eigen-vectors that retain a fixed percentage of the variance, (e.g., 80%, the smallest d such that $\frac{\sum_{i=1}^{d} \lambda_i}{\sum_i \lambda_i} \geq 80\%$)



You might loose some info. But if the eigen-values are small, not much is lost.



The more PC you use, the better the reconstruction!

PCA: A useful preprocessing step

- Helps reduce computational complexity.
- Can help supervised learning.
  - Reduced dimension $\Rightarrow$ simpler hypothesis space.
  - Smaller VC (Vapnik–Chervonenkis) dimension $\Rightarrow$ less risk of overfitting.
- PCA can also be seen as noise reduction.
- Caveats (Notes):
  - Fails when data consists of multiple separate clusters (mixture of densities).
  - Directions of greatest variance may not be most informative (i.e. greatest classification power).

Small VC -> Less complicated for classification

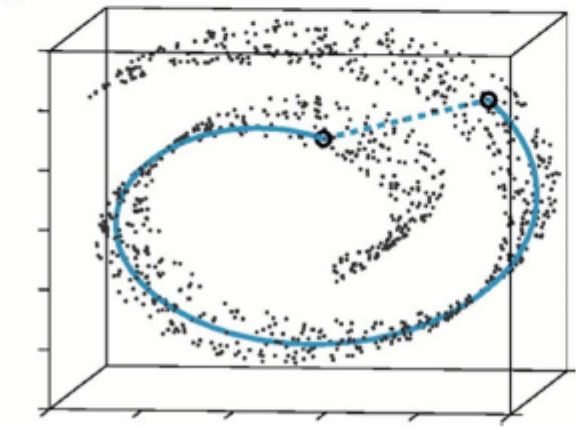
Scaling up PCA

- Practical issue: covariance matrix is n x n.
  - E.g. for image data Σ = 32768 x 32768 (over billions elements).
  - Finding eigenvectors of such a matrix is slow.

- Singular value decomposition (SVD) to the rescue!
  - Can be used to compute principal components.
  - Efficient implementations available, e.g. MATLAB svd.

Nonlinear dimensionality reduction

- Data often lies on or near a nonlinear low-dimensional surface
- Such low-dimensional surfaces are called *manifolds*.



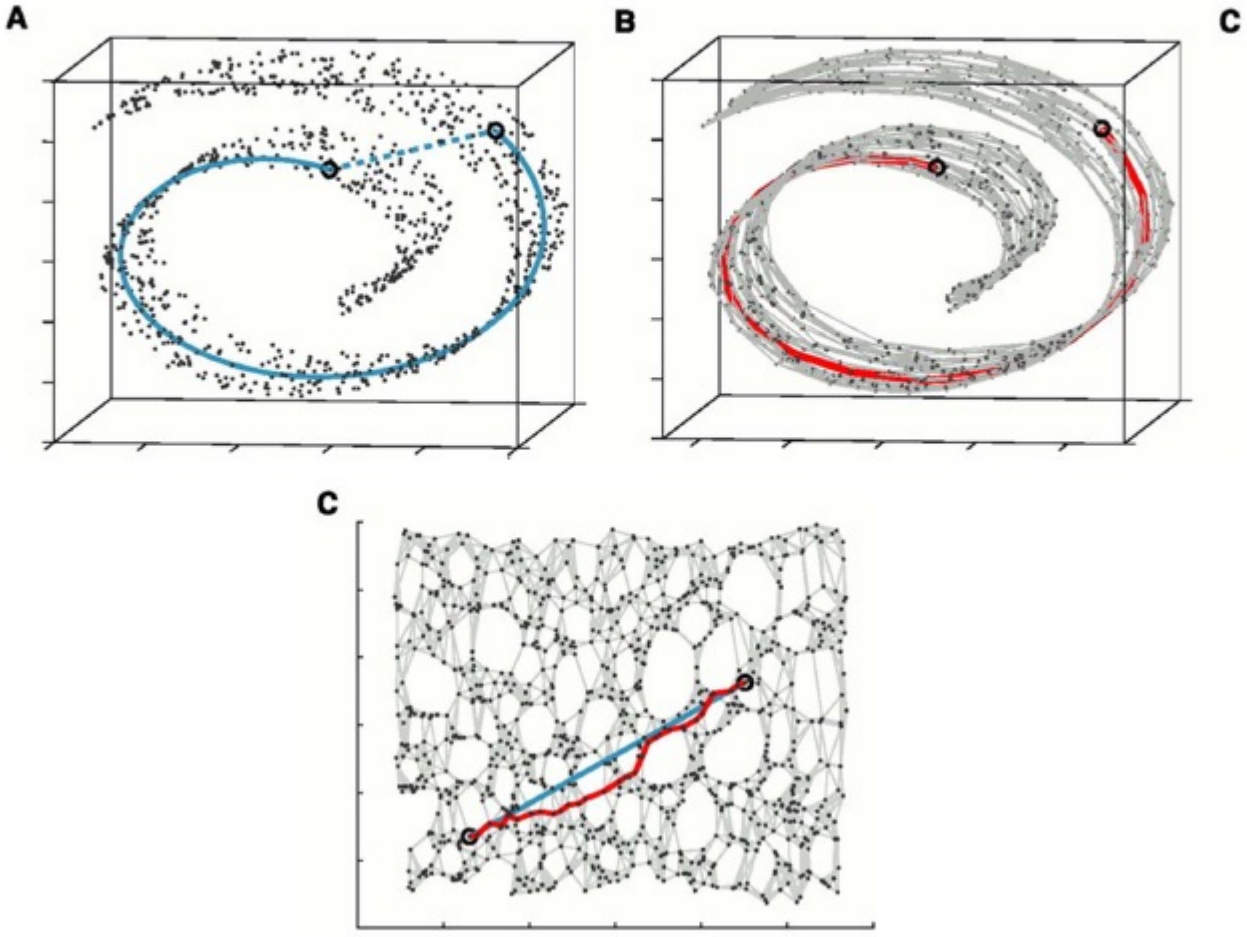
Swiss roll data

ISOMAP: Isometric Feature Mapping

(Tenenbaum et al. 2000)

- A nonlinear method for dimensionality reduction
- Preserves the global, nonlinear geometry of the data by preserving the geodesic distances
- Geodesic: originally geodesic means the shortest route between two points on the surface of the manifold

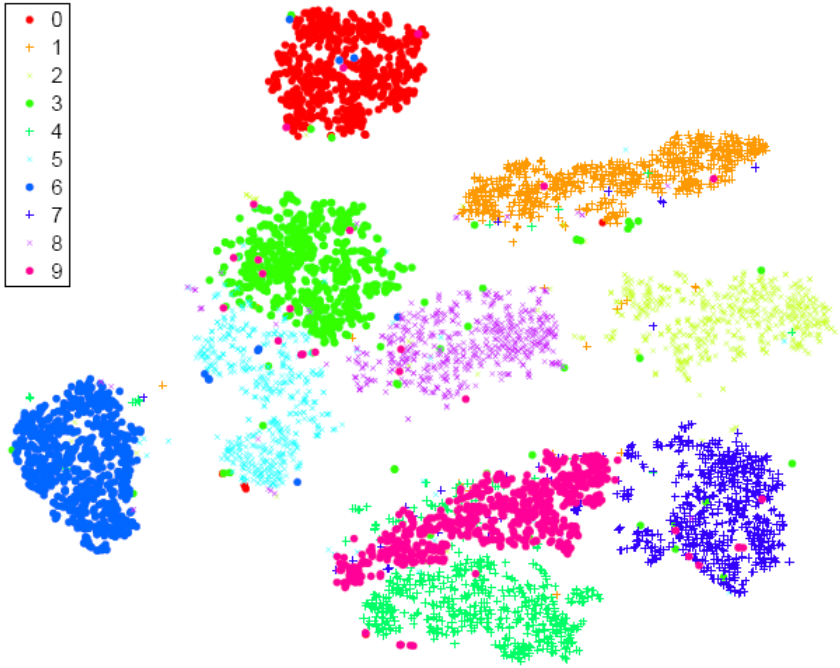
ISOMAP on Swiss Roll Data



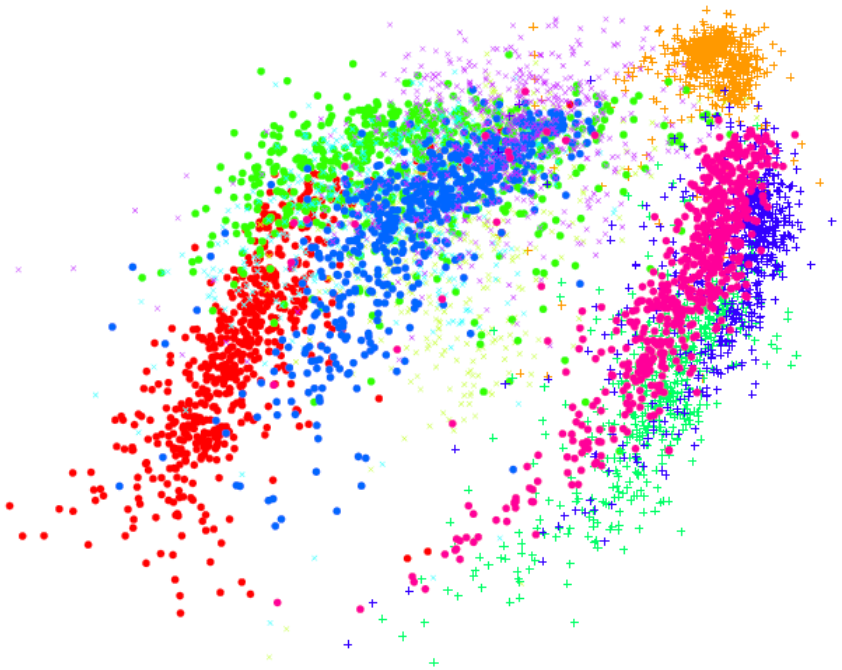
t-Stochastic Neighbor Embedding (t-SNE)

- Visualizing high-D (big) data using t-SNE, published by Laurens van der Maaten and Geoffrey Hinton in 2008
- Visualizes high-dimensional data in a 2- or 3-dimensional map.
- Better than existing techniques at creating a single map that reveals structure at many different scales.
- Particularly good for high-dimensional data that lie on several different, but related, low-dimensional manifolds.
  - Example: images of objects from multiple classes seen from multiple viewpoints.

Visualization of classes in MNIST data



t-SNE

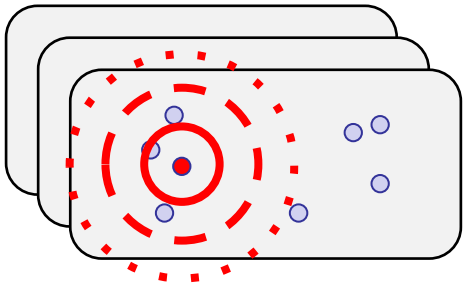


ISOMAP

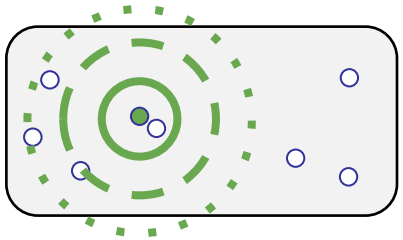
A tsne Demo

Some Assumptions

- High-dimensional data often lies on or near a much lower dimensional, curved manifold.
- A good way to represent data points is by their low-dimensional coordinates.
- The low-dimensional representation of the data should capture information about high-dimensional pairwise distances.



High-Dimensional Space



Low-Dimensional Space

Similarity Computation → Embedding Optimization

$X = \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$

$p_{ij} =$ Similarity between *i* and *j in H-Dim*

$P =$ Probability distribution encoding similarities

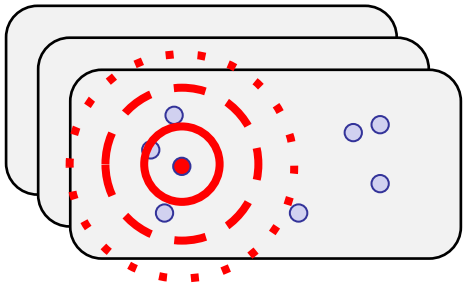
$Y = \{\mathbf{y}_1 \ldots \mathbf{y}_N\}$

$q_{ij} =$ Similarity between *i* and *j in L-Dim*

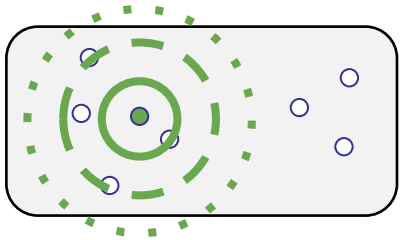
$Q =$ Probability distribution encoding similarities

Entropy-based cost function

$$C(P,Q) = KL(P||Q)$$



High-Dimensional Space



Low-Dimensional Space

Similarity Computation → Embedding Optimization

$X = \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$

$p_{ij} =$ Similarity between *i* and *j in H-Dim*

$P =$ Probability distribution encoding similarities

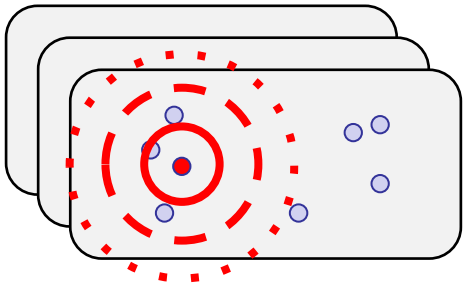
$Y = \{\mathbf{y}_1 \ldots \mathbf{y}_N\}$

$q_{ij} =$ Similarity between *i* and *j in L-Dim*

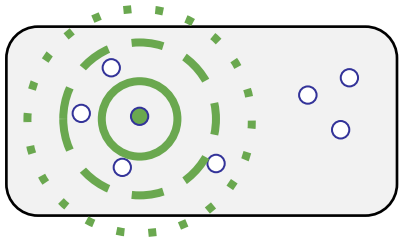
$Q =$ Probability distribution encoding similarities

Entropy-based cost function

$$C(P,Q) = KL(P||Q)$$



High-Dimensional Space



Low-Dimensional Space

Similarity Computation → Embedding Optimization

$X = \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$

$p_{ij} =$ Similarity between *i* and *j in H-Dim*

$P =$ Probability distribution encoding similarities

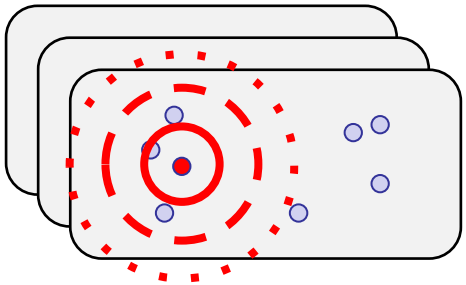
$Y = \{\mathbf{y}_1 \ldots \mathbf{y}_N\}$

$q_{ij} =$ Similarity between *i* and *j in L-Dim*

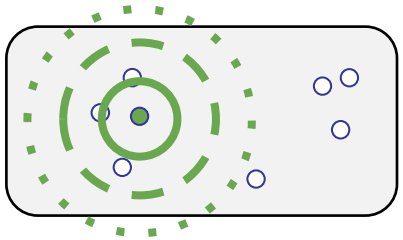
$Q =$ Probability distribution encoding similarities

Entropy-based cost function

$$C(P,Q) = KL(P||Q)$$



High-Dimensional Space



Low-Dimensional Space

Similarity Computation → Embedding Optimization

$X = \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$

$p_{ij} =$ Similarity between *i* and *j in H-Dim*

$P =$ Probability distribution encoding similarities

$Y = \{\mathbf{y}_1 \ldots \mathbf{y}_N\}$

$q_{ij} =$ Similarity between *i* and *j in L-Dim*

$Q =$ Probability distribution encoding similarities

Entropy-based cost function

$$C(P,Q) = KL(P||Q)$$

Basic idea of non-parametric dimensionality reduction

- Represent each data-point by a point in a lower dimensional space.
- Choose the low-dimensional points so that they optimally represent some property of the data-points (e.g. the pairwise distances).
  - Many different properties have been tried.
- Do not insist on learning a parametric "encoding" function that maps each individual data-point to its low-dimensional representative.
- Do not insist on learning a parametric "decoding" function that reconstructs a data-point from its low dimensional representative.

Non-parametric approach!

Summary

- Gain insight into data by:
  - Classical dimensionality reduction techniques
  - Dimensionality reduction techniques that preserve certain properties of the original dataset
- Dimensionality reduction is nearly a must for machine learning
- Dimensionality reduction can be carried out w.r.t. different tasks, classification, regression, clustering, etc. Typically, the method itself is considered as unsupervised learning.
- Many methods have been developed but still an active area of research, particularly with respect to different applications like bioinformatics (reducing from multi-million features) and time series data (with dimensionality dynamically increasing)

Acknowledgement

Slides from

- J. Jeffry Howbert, U of Washington
- Xiaoli Fern, Oregon State University
- Geoffrey Hinton, University of Toronto
- Nicola Pezzotti@VIS2019
- E. Alpaydin, Introduction to Machine Learning. 2nd Ed. MIT Press, 2010.

Photos from Internet