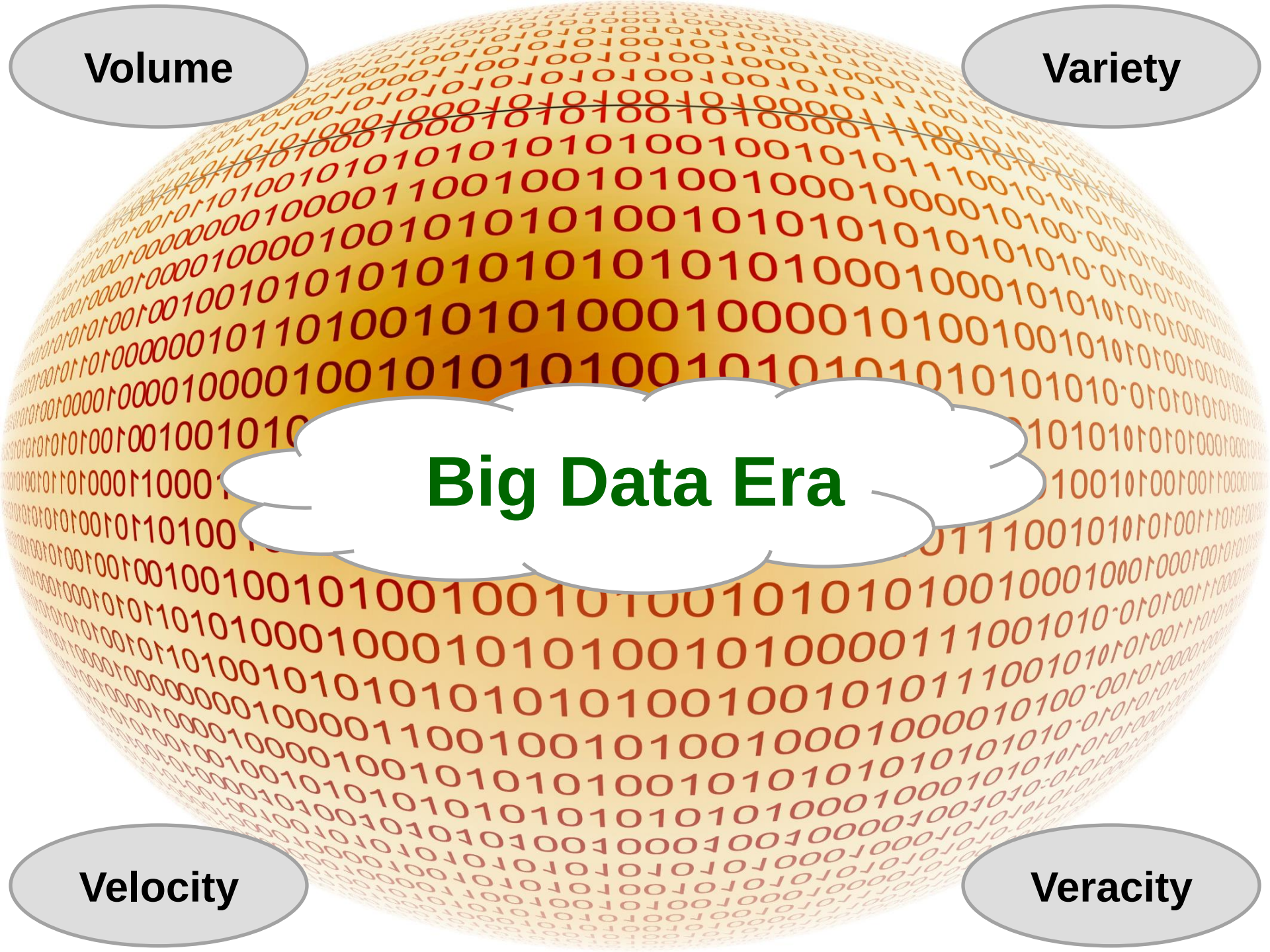
CSE 6242 / CX 4242

October 9, 2014

Dimension Reduction

Guest Lecturer: Jaegul Choo


Volume
Variety
Big Data Era
Velocity
Veracity

Big Data
are
High-Dimensional

Examples of High-Dimensional Data

Image Data

- Serialized/rasterized pixel values

Raw images

5	34	78
3	80	63
58	24	45

Pixel values

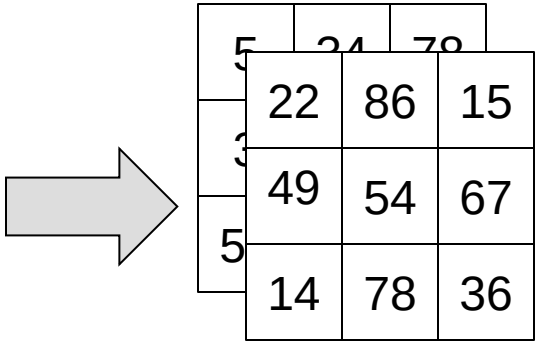
5
3
58
34
80
24
63
45
63

Serialized pixels

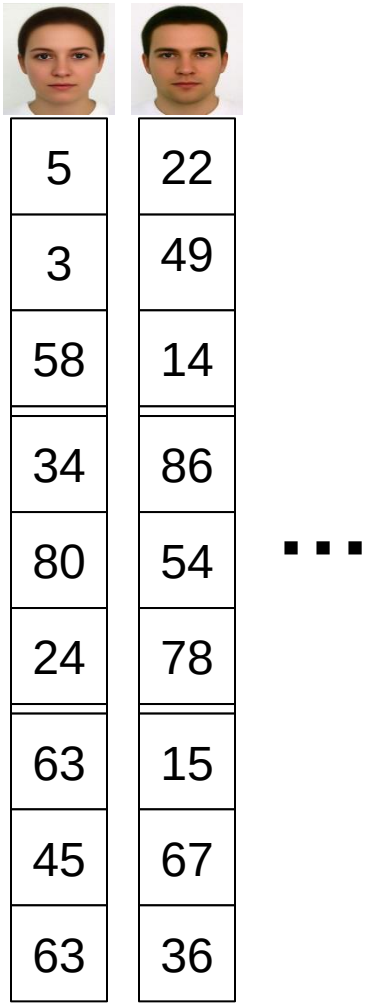
Examples of High-Dimensional Data
Facial Images

- Serialized/rasterized pixel values

Raw images



Pixel values



Serialized pixels

- Huge dimensions
  - 640x480 image size → 307,200 dimensions

Examples of High-Dimensional Data
Text Documents

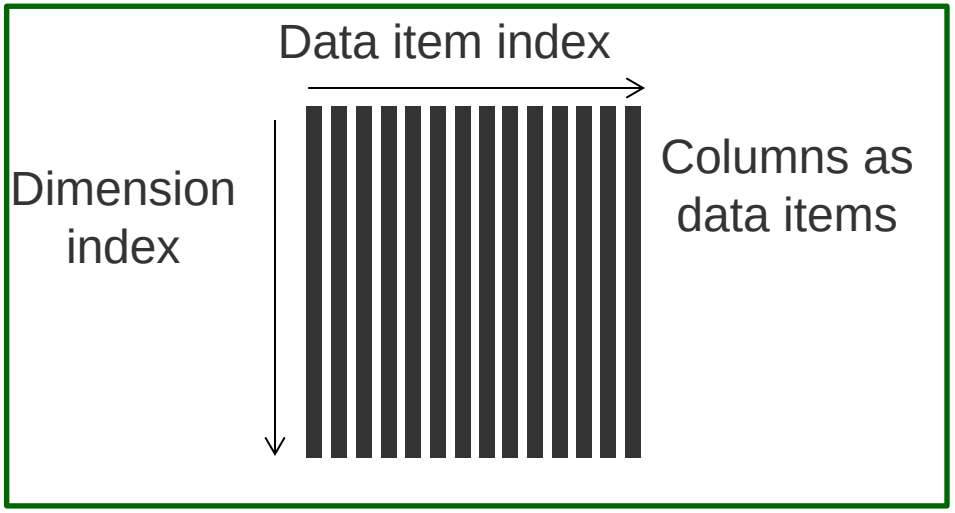
- Bag-of-words vector
  - Document 1 = “John likes movies. Mary likes too.”
  - Document 2 = “John also likes football.”

John	1	1
likes	1	1
movies	0	0
also	1	1
football	1	1
Mary	0	0
too	0	0

...

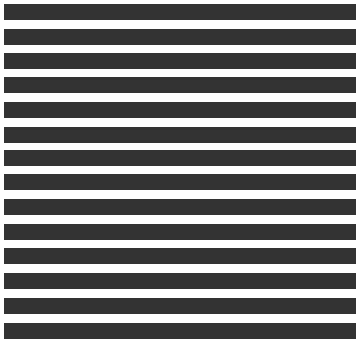
Two Axes of Data Set

- Data items
  - How many data items?
- Dimensions
  - How many dimensions representing each item?

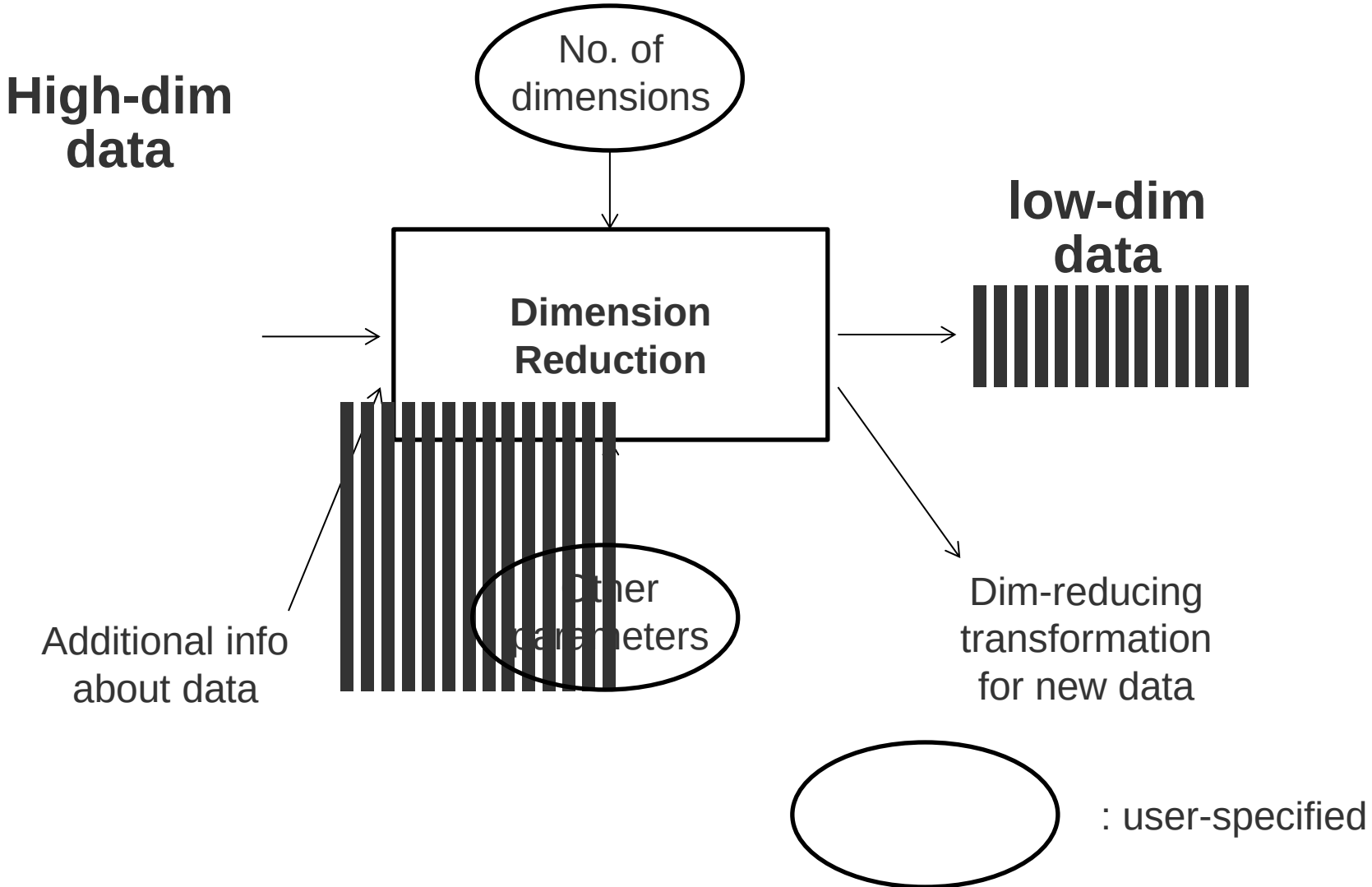


We will use this during lecture

vs. Rows as data items



Overview of Dimension Reduction
Let's Reduce Data (along Dimension Axis)



Benefits of Dimension Reduction

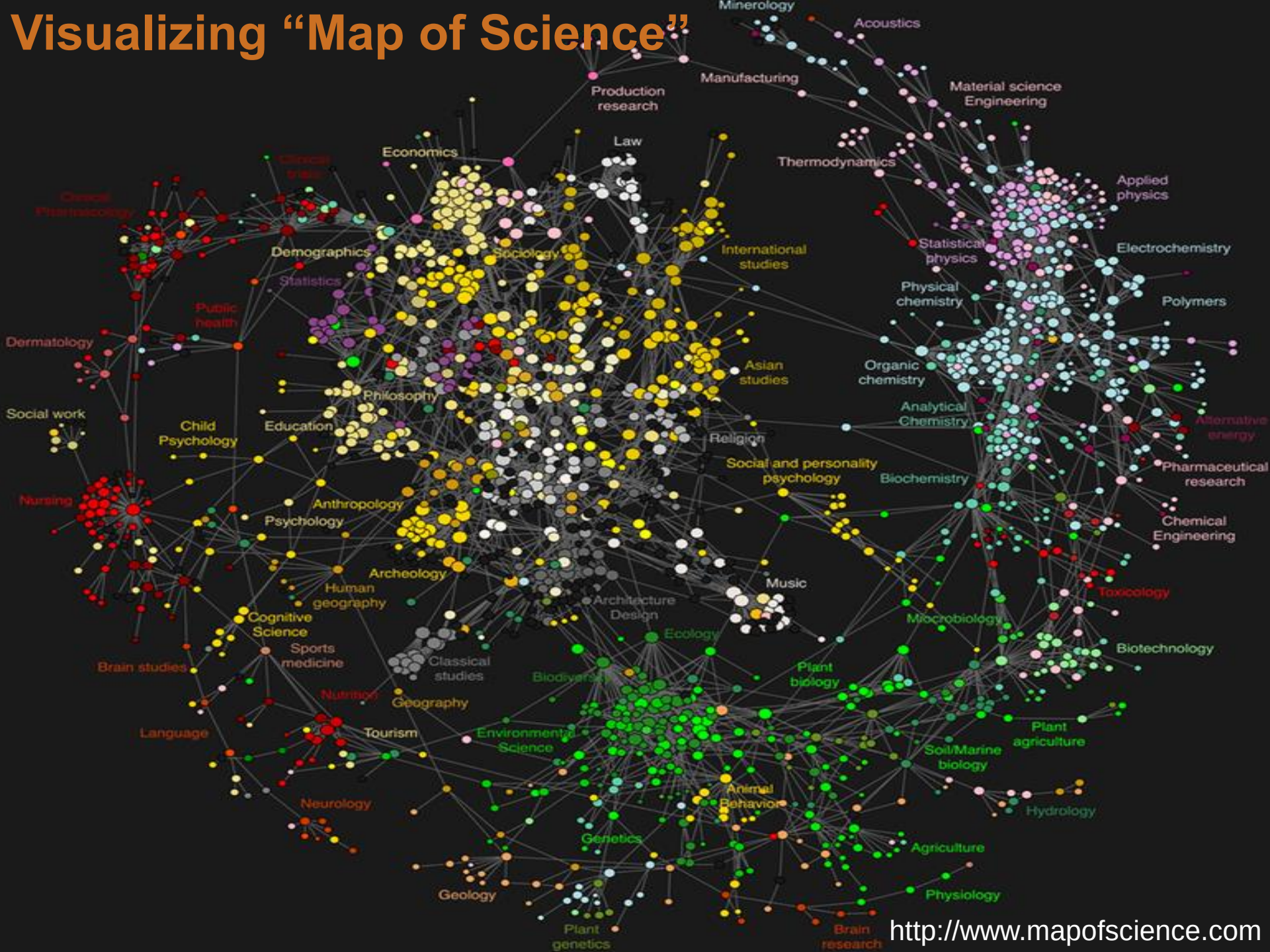
Obviously,

- Less storage
- Faster computation
  - Computing distances: 100,000-dim vs. 10-dim vectors

More importantly,

- Noise removal (improving data quality)
  - Works as **pre-processing** for **better performance**
  - e.g., microarray data analysis, information retrieval, face recognition, protein disorder prediction, network intrusion detection, document categorization, speech recognition
- **2D/3D representation**
  - **Interactive visual exploration**

Visualizing “Map of Science”



http://www.mapofscience.com

Two Main Techniques

1. Feature **selection**

- Selects a subset of the original variables as reduced dimensions
- e.g., the number of genes responsible for a particular disease may be small

2. Feature **extraction**

- Each reduced dimension combines multiple original dimensions
- Active research area

Feature = Variable = Dimension

Feature Selection

What are the optimal subset of ***m*** features to maximize a given criterion?

- Widely-used criteria
  - Information gain, correlation, …
- Typically combinatorial optimization problems
- Therefore, greedy methods are popular
  - Forward selection: **Empty** set → **Add** one variable at a time
  - Backward elimination: **Entire** set → **Remove** one variable at a time

Feature Extraction
(Our main topic from now on)

Aspects of Dimension Reduction

- Linear vs. Nonlinear
- Unsupervised vs. Supervised
- Global vs. Local
- Feature vectors vs. Similarity (as an input)

Aspects of Dimension Reduction
Linear vs. Nonlinear

Linear

- Represents each reduced dimension as a linear combination of original dimensions
  - e.g., Y1 = 3*X1 – 4*X2 + 0.3*X3 – 1.5*X4,
    Y2 = 2*X1 + 3.2*X2 – X3 + 2*X4
- Naturally capable of mapping new data to the same space

	D1	D2
X1	1	1
X2	1	0
X3	0	2
X4	1	1

	D1	D2
Y1	1.75	-0.27
Y2	-0.21	0.58

Aspects of Dimension Reduction
Linear vs. Nonlinear

Linear

- Represents each reduced dimension as a linear combination of original dimensions
  - e.g., Y1 = 3*X1 – 4*X2 + 0.3*X3 – 1.5*X4,
    Y2 = 2*X1 + 3.2*X2 – X3 + 2*X4
- Naturally capable of mapping new data to the same space

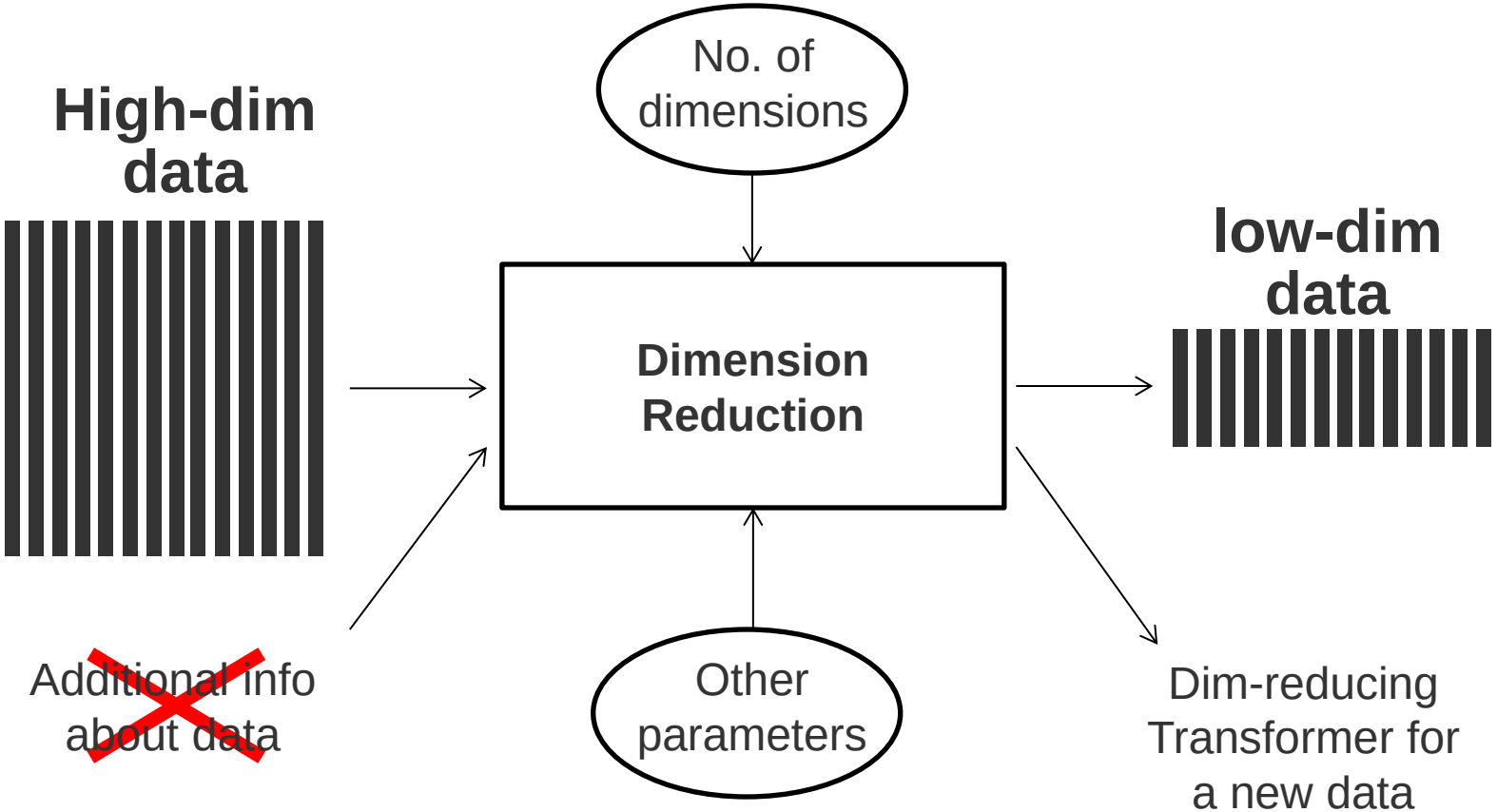
Nonlinear

- More complicated, but generally more powerful
- Recently popular topics

Aspects of Dimension Reduction
Unsupervised vs. Supervised

Unsupervised

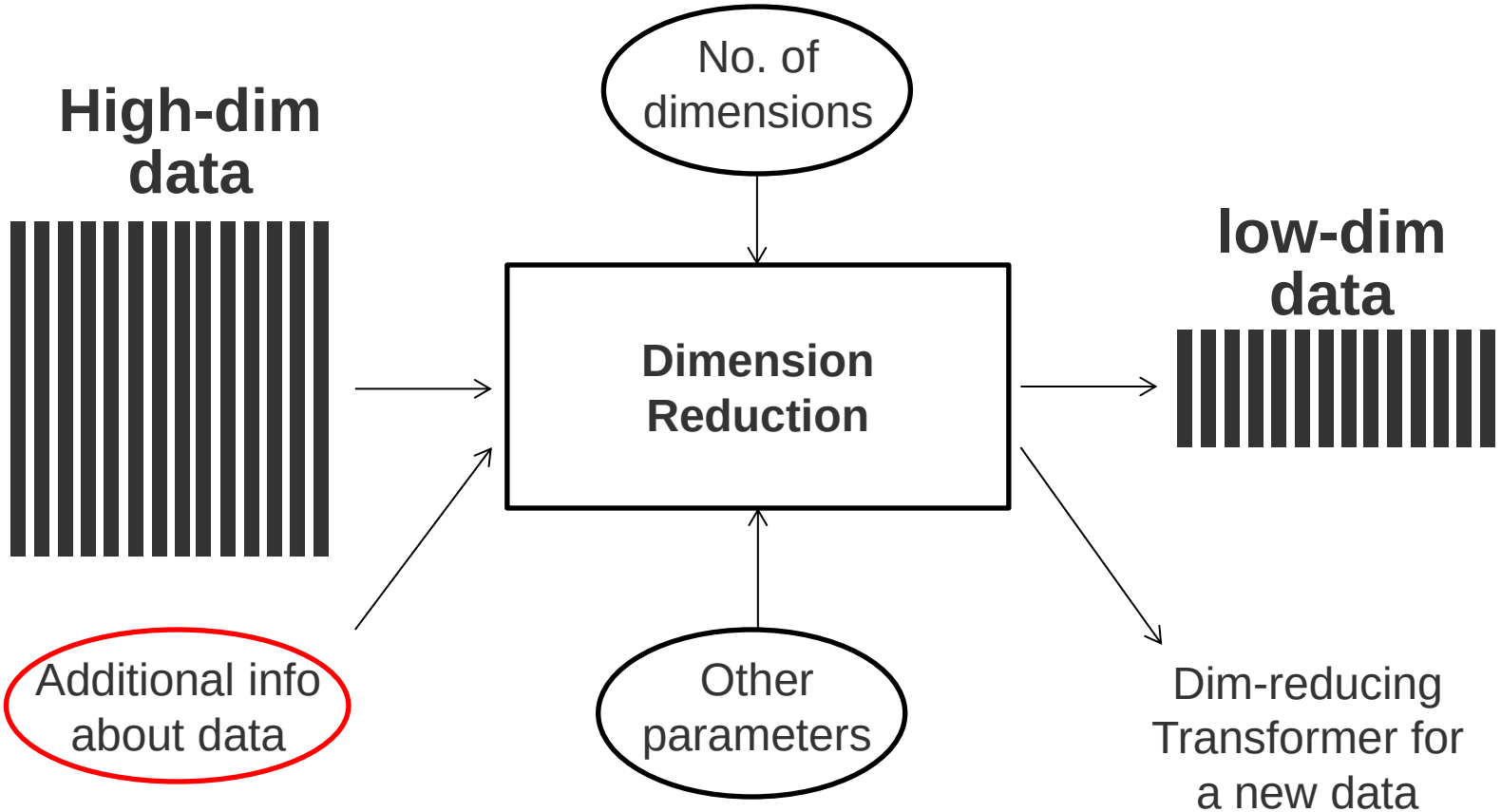
- Uses only the input data



Aspects of Dimension Reduction
Unsupervised vs. Supervised

Supervised

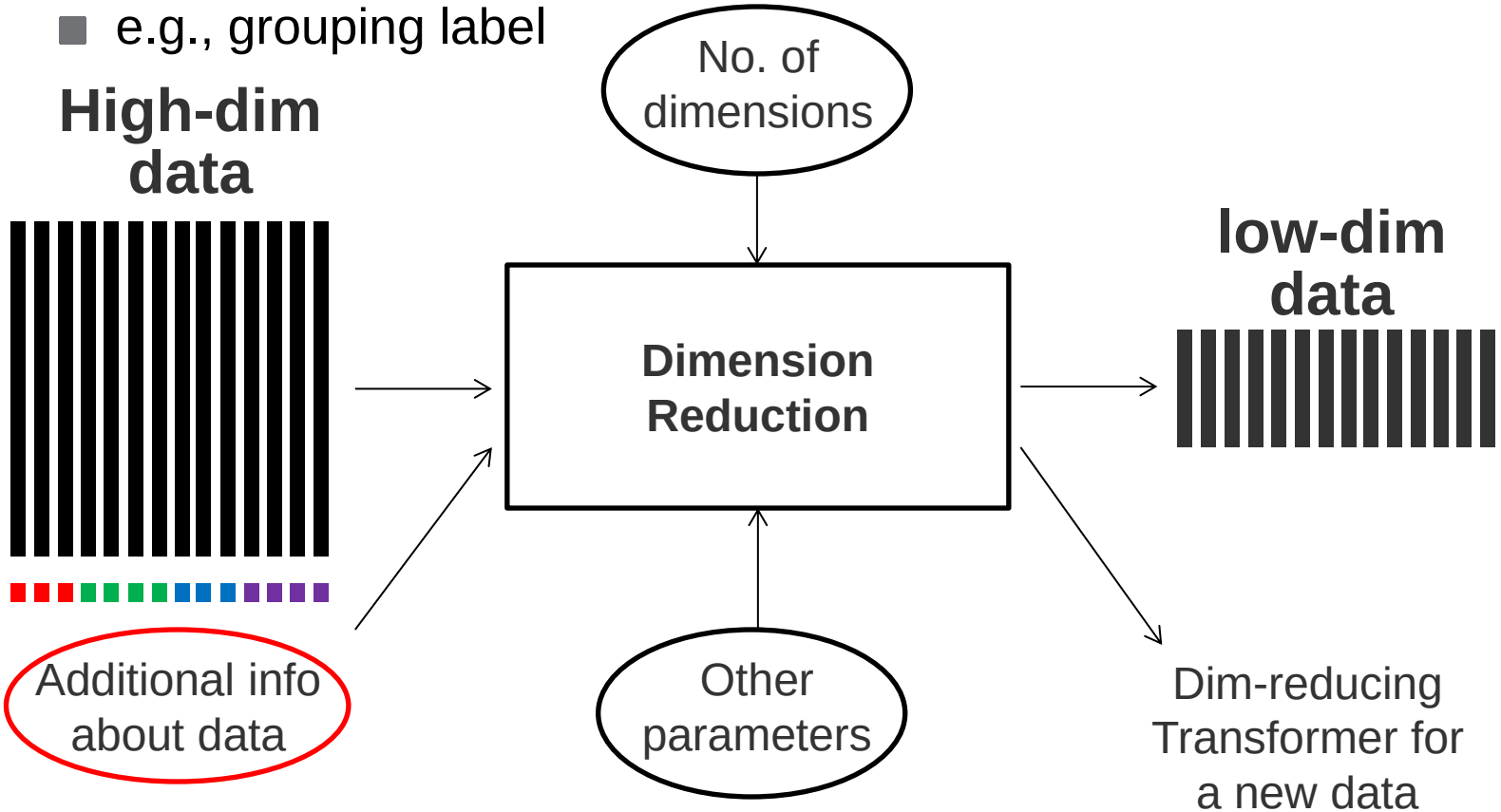
- Uses the input data + additional info



Aspects of Dimension Reduction
Unsupervised vs. Supervised

Supervised

- Uses the input data + additional info
  - e.g., grouping label



Aspects of Dimension Reduction
Global vs. Local

Dimension reduction typically tries to preserve all the relationships/distances in data

- Information loss is unavoidable!

Then, what should we emphasize more?

Global

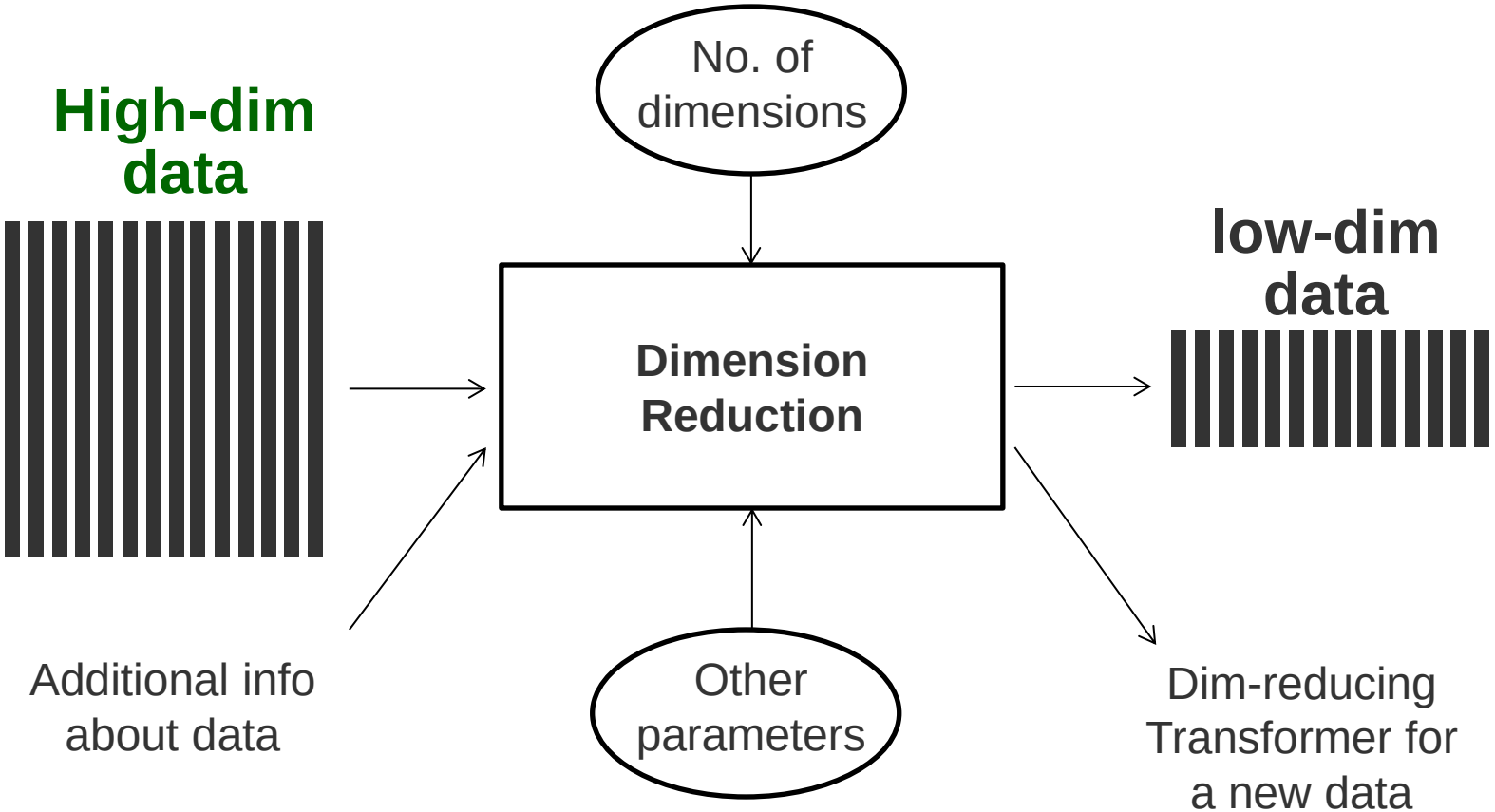
- Treats all pairwise distances equally important
  - Focuses on preserving large distances

Local

- Focuses on small distances, neighborhood relationships
- Active research area, e.g., manifold learning

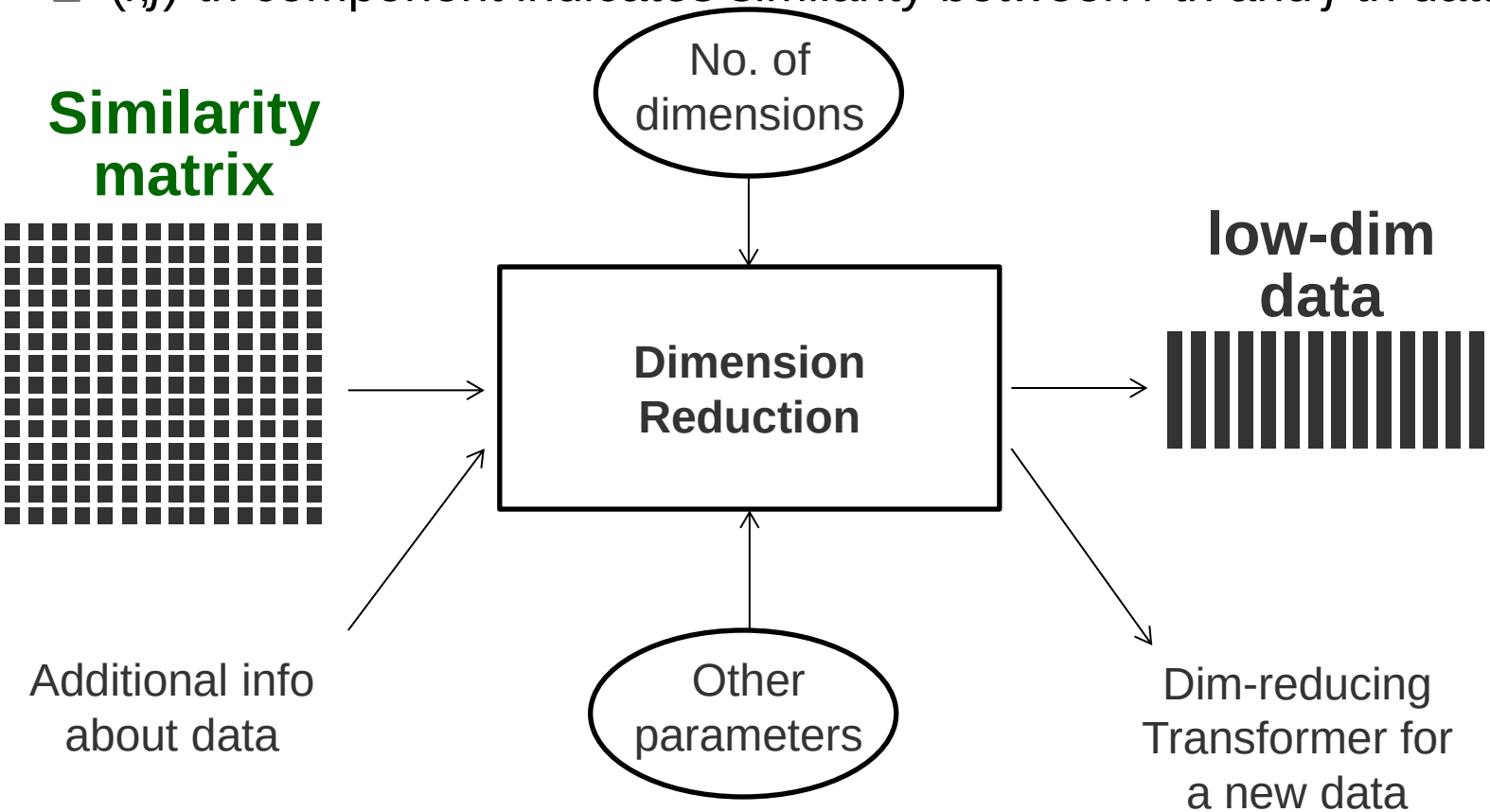
Aspects of Dimension Reduction
Feature vectors vs. Similarity (as an input)

- Typical setup (feature vectors as an input)



Aspects of Dimension Reduction
Feature vectors vs. Similarity (as an input)

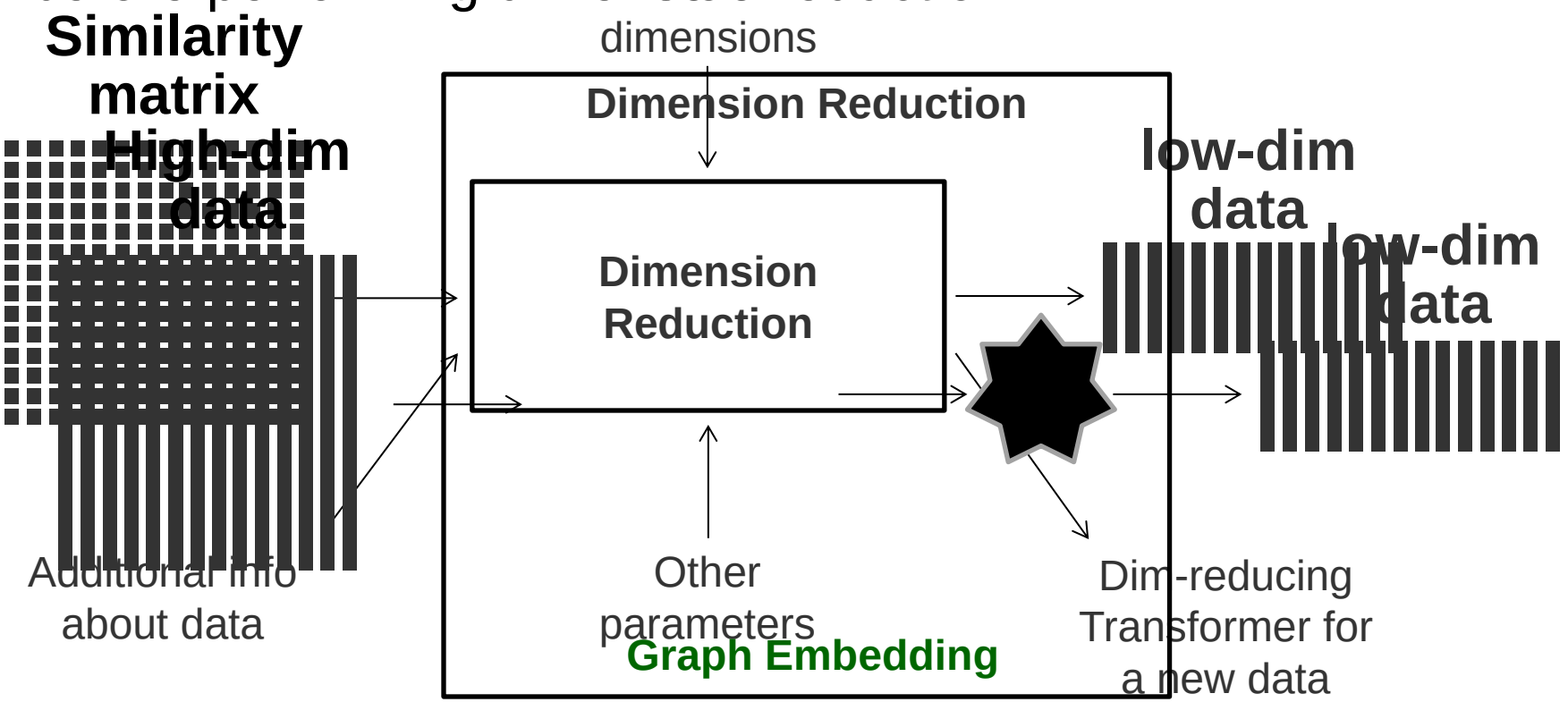
- Typical setup (feature vectors as an input)
- Alternatively, takes similarity matrix instead
  - (*i*,*j*)-th component indicates similarity between *i*-th and *j*-th data



Aspects of Dimension Reduction

Feature vectors vs. Similarity (as an input)

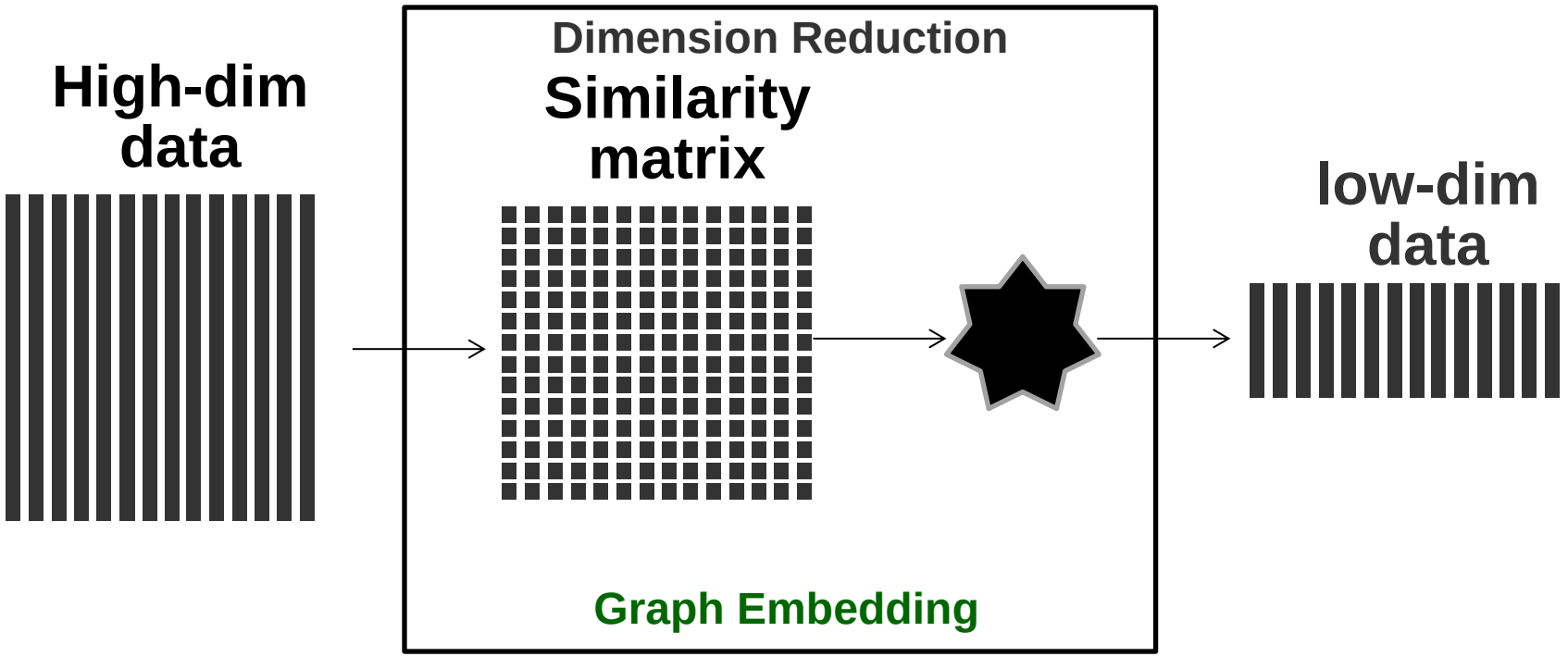
- Typical setup (feature vectors as an input)
- Alternatively, takes similarity matrix instead
- Internally, converts feature vectors to similarity matrix before performing dimension reduction



Aspects of Dimension Reduction
Feature vectors vs. Similarity (as an input)

Why called graph embedding?

- Similarity matrix can be viewed as a **graph** where similarity represents edge weight



Methods

- Traditional
  - Principal component analysis (PCA)
  - Multidimensional scaling (MDS)
  - Linear discriminant analysis (LDA)

- Advanced (nonlinear, kernelized, manifold learning)
  - Isometric feature mapping (Isomap)
  - t-distributed stochastic neighborhood embedding (t-SNE)

* Matlab codes are available at
http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

Principal Component Analysis

- Finds the axis showing the greatest variation, and project all points into this axis
- Reduced dimensions are orthogonal
- Algorithm: Eigen-decomposition
- Pros: Fast
- Cons: Limited performances

Linear
Unsupervised
Global
Feature vectors

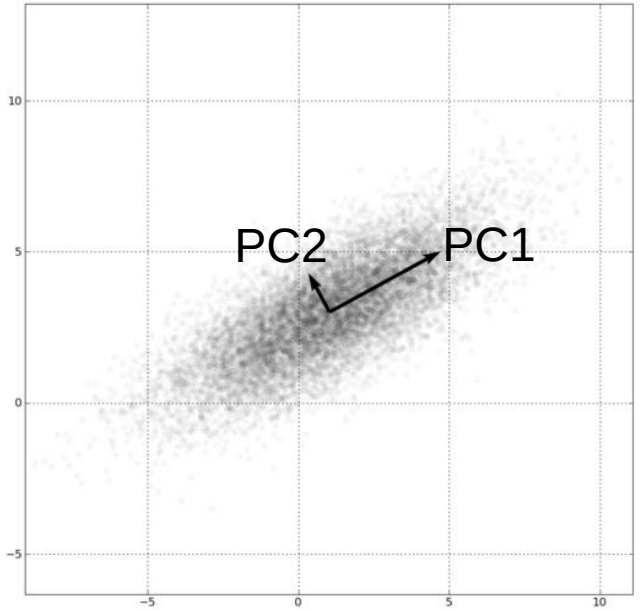
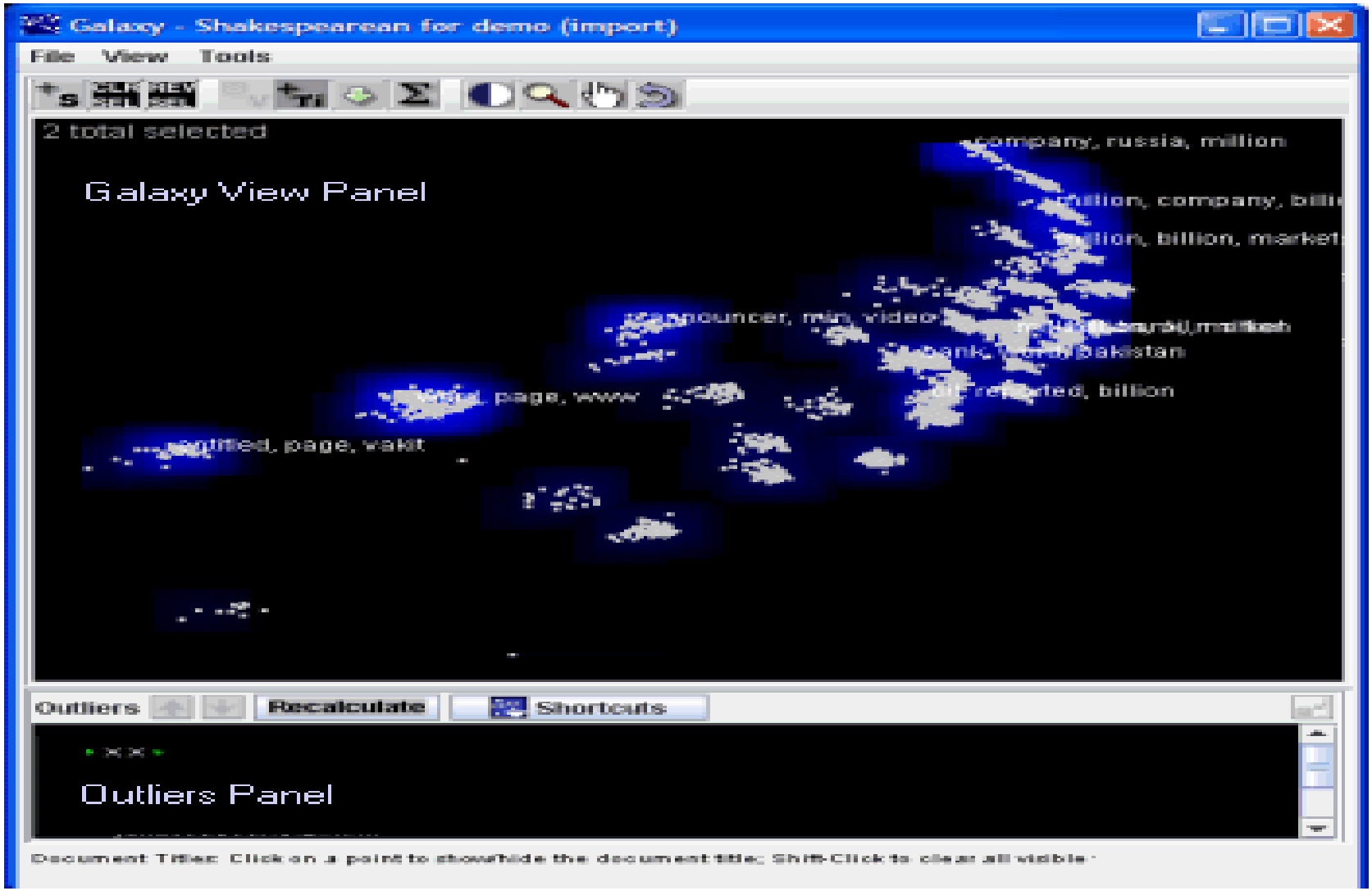


Image source: http://en.wikipedia.org/wiki/Principal_component_analysis

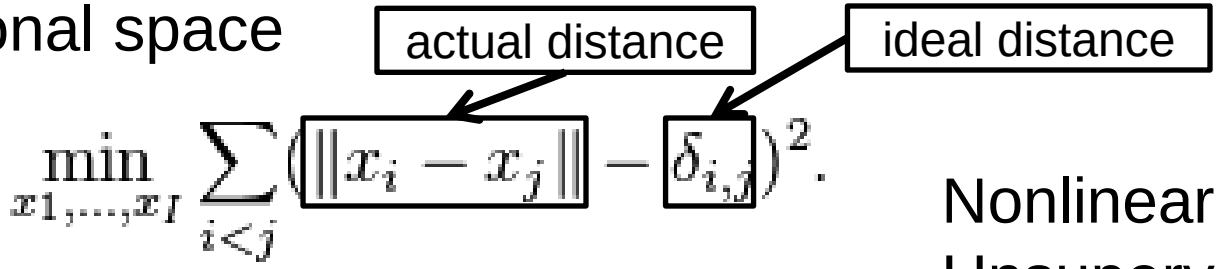
Principal Component Analysis
Document Visualization



Multidimensional Scaling (MDS)

Main idea

- Tries to preserve given pairwise distances in low-dimensional space



Nonlinear
Unsupervised
Global
Similarity input

- Metric MDS
  - Preserves given distance values
- Nonmetric MDS
  - When you only know/care about ordering of distances
  - Preserves only **the orderings** of distance values

- Algorithm: gradient-decent type

c.f. classical MDS is the same as PCA

Multidimensional Scaling
Sammon's mapping

Sammon's mapping

- Local version of MDS
- Down-weights errors in large distances

$$E = \frac{1}{\sum_{i<j} d_{ij}^*} \sum_{i<j} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*}.$$

- Algorithm: gradient-decent type

Nonlinear
Unsupervised
Local
Similarity input

Multidimensional Scaling
Force-directed graph layout

Force-directed graph layout

- Rooted from graph visualization, but essentially a variant of metric MDS
- Spring-like attractive + repulsive forces between nodes
- Algorithm: gradient-decent type

- Widely-used in visualization
  - Aesthetically pleasing results
  - Simple and intuitive
  - **Interactive**

Multidimensional Scaling
Force-directed graph layout

Demos

- Prefuse
  - http://prefuse.org/gallery/graphview/
- D3
  - http://bl.ocks.org/4062045

Multidimensional Scaling

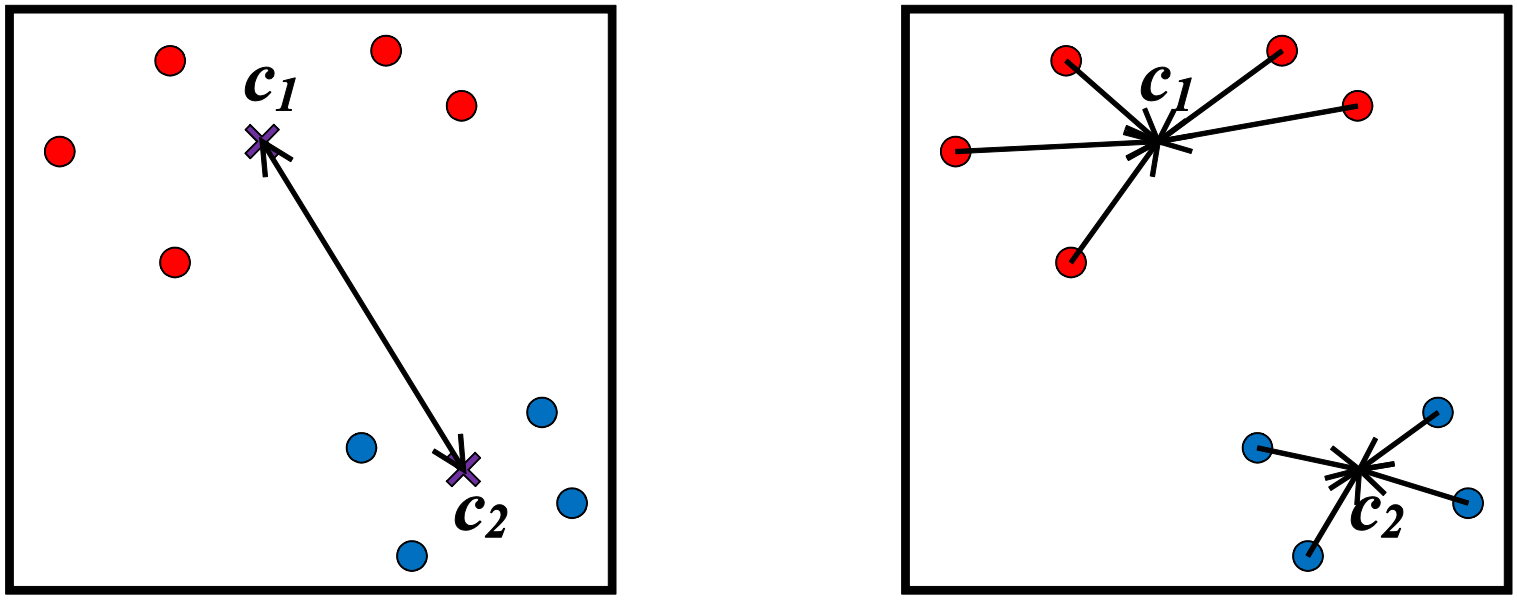
- Pros: widely-used (works well in general)
- Cons: slow (*n*-body problem)
  - Nonmetric MDS is even much slower than metric MDS
  - Fast algorithm are available.
    - Barnes-Hut algorithm
    - GPU-based implementations

Linear Discriminant Analysis

What if clustering information is available?

LDA tries to separate clusters by

- Putting different cluster as far as possible
- Putting each cluster as compact as possible

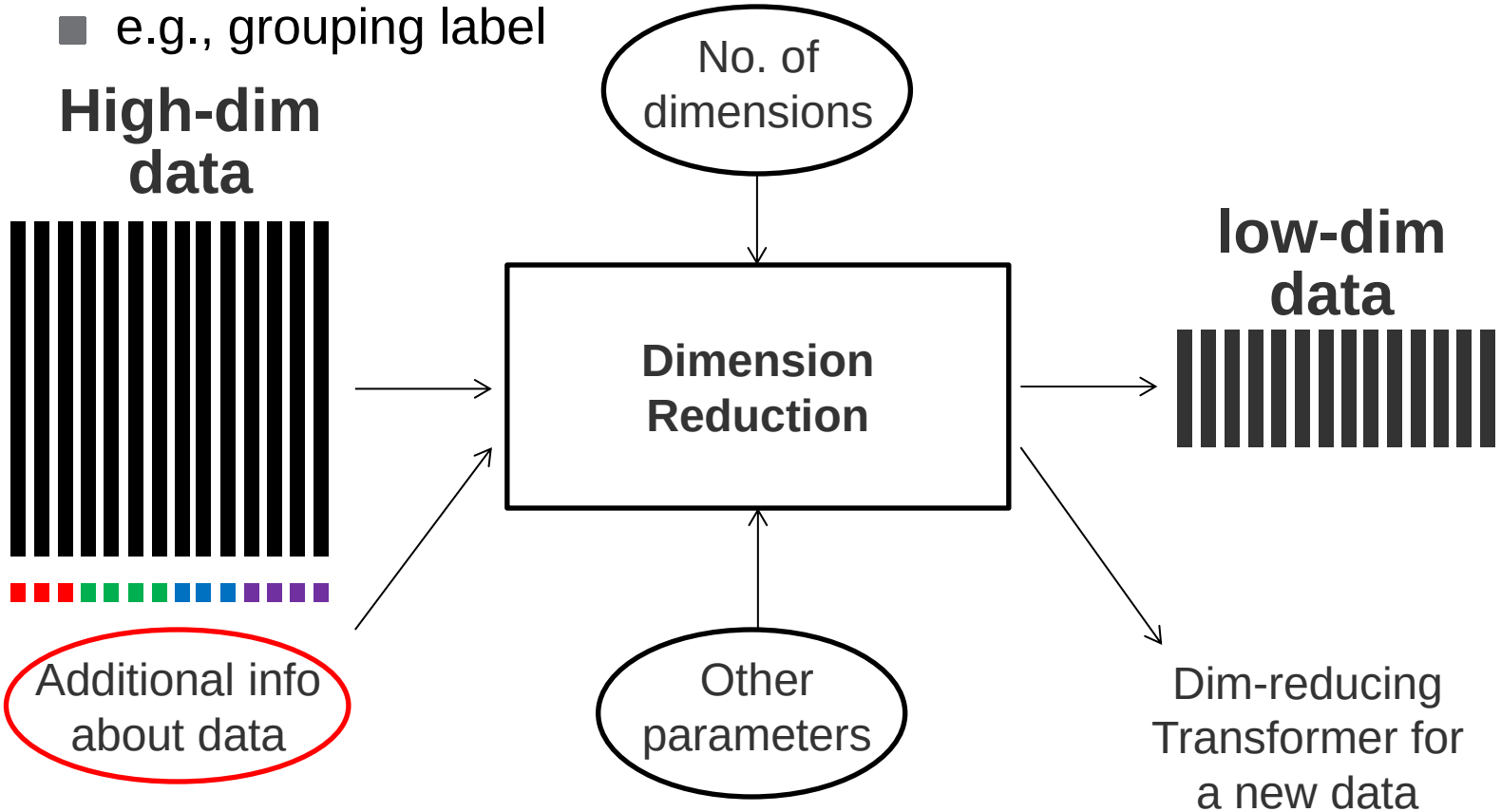


(a) (b)

Aspects of Dimension Reduction
Unsupervised vs. Supervised

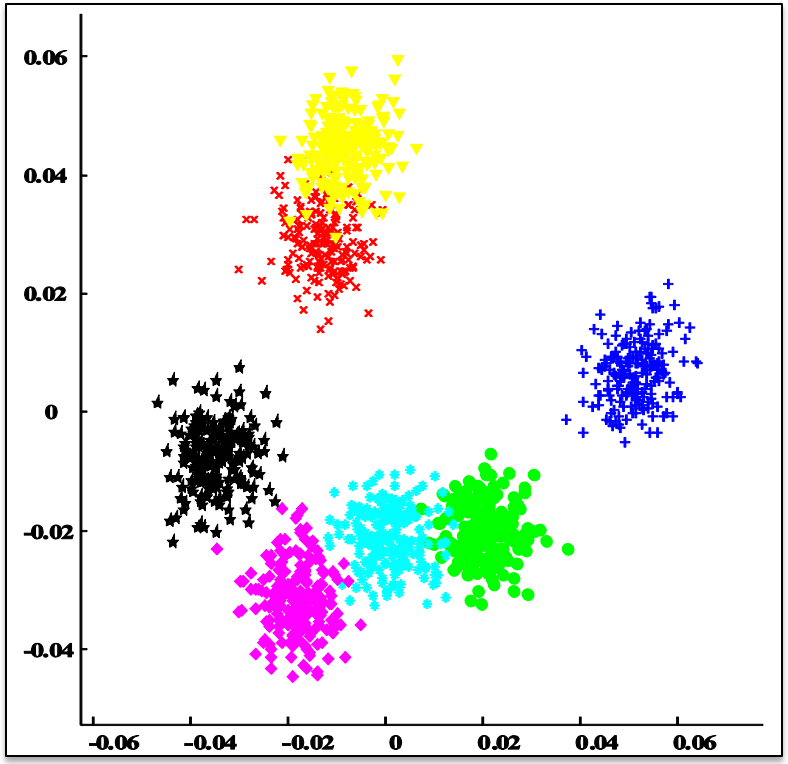
Supervised

- Uses the input data + additional info
  - e.g., grouping label



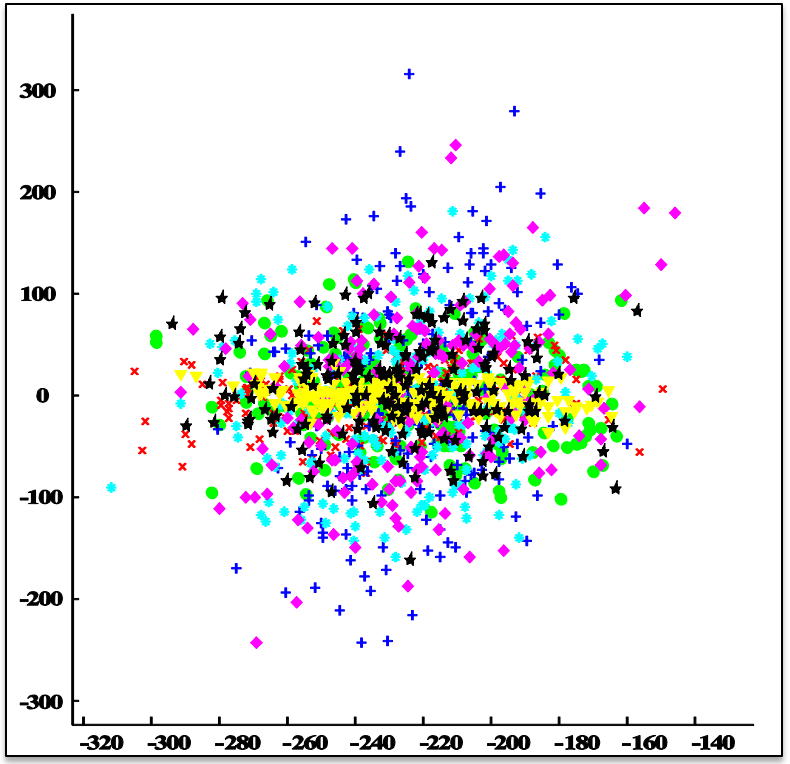
Linear Discriminant Analysis
vs. Principal Component Analysis

2D visualization of 7 Gaussian mixture of 1000 dimensions



Linear discriminant analysis

(Supervised)



Principal component analysis

(Unsupervised)

Linear Discriminant Analysis

Maximally separates clusters by

- Putting different cluster far apart
- Shrinking each cluster compactly

- Algorithm: generalized eigendecomposition
- Pros: better show cluster structure
- Cons: may distort original relationships of data

Linear
Supervised
Global
Feature vectors

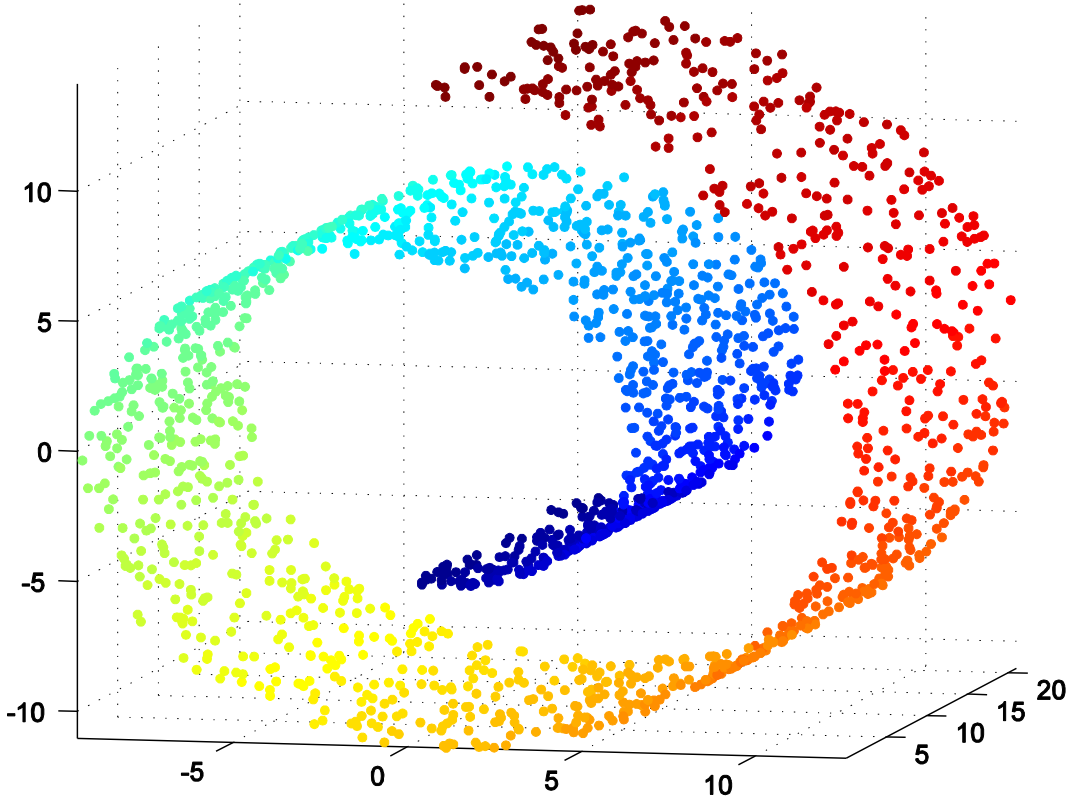
Methods

- Traditional
  - Principal component analysis (PCA)
  - Multidimensional scaling (MDS)
  - Linear discriminant analysis (LDA)

- Advanced (nonlinear, kernelized, manifold learning)
  - Isometric feature mapping (Isomap)
  - t-distributed stochastic neighborhood embedding (t-SNE)

* Matlab codes are available at
http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

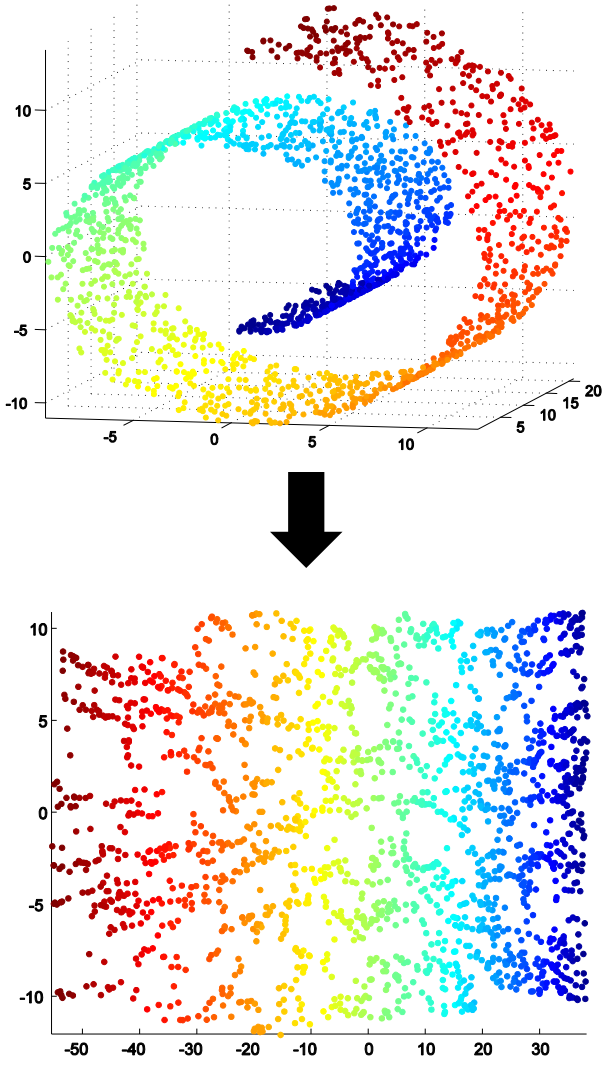
Manifold Learning
Swiss Roll Data



Swiss roll data

- Originally in 3D
- What is the **intrinsic** dimensionality? (allowing flattening)

Manifold Learning
Swiss Roll Data



Swiss roll data

- Originally in 3D
- What is the **intrinsic** dimensionality? (allowing flattening)

  → 2D

What if your data has **low** intrinsic dimensionality but resides in **high**-dimensional space?

Manifold Learning
Goal and Approach

Manifold

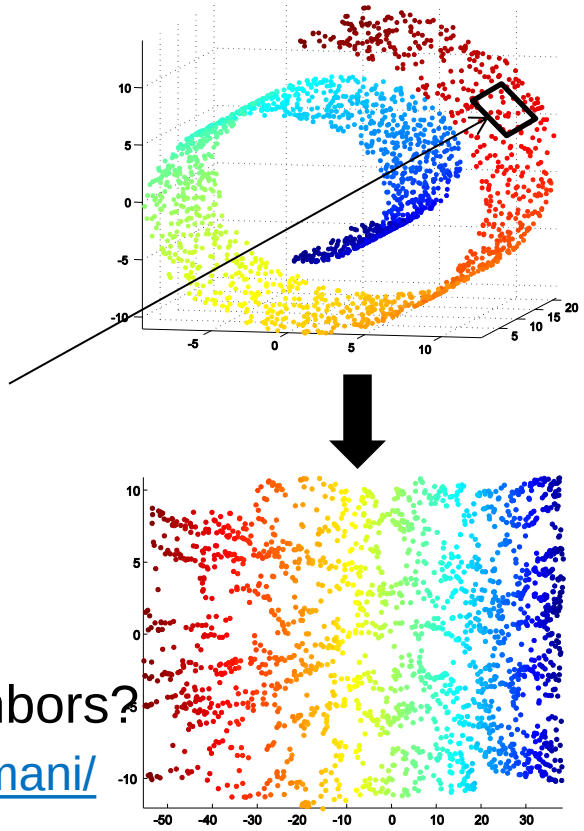
- "Curvi-linear" low-dimensional structure of your data based on intrinsic dimensionality

Manifold learning

- Map intrinsic dimensions to axes of dimension-reduced output space

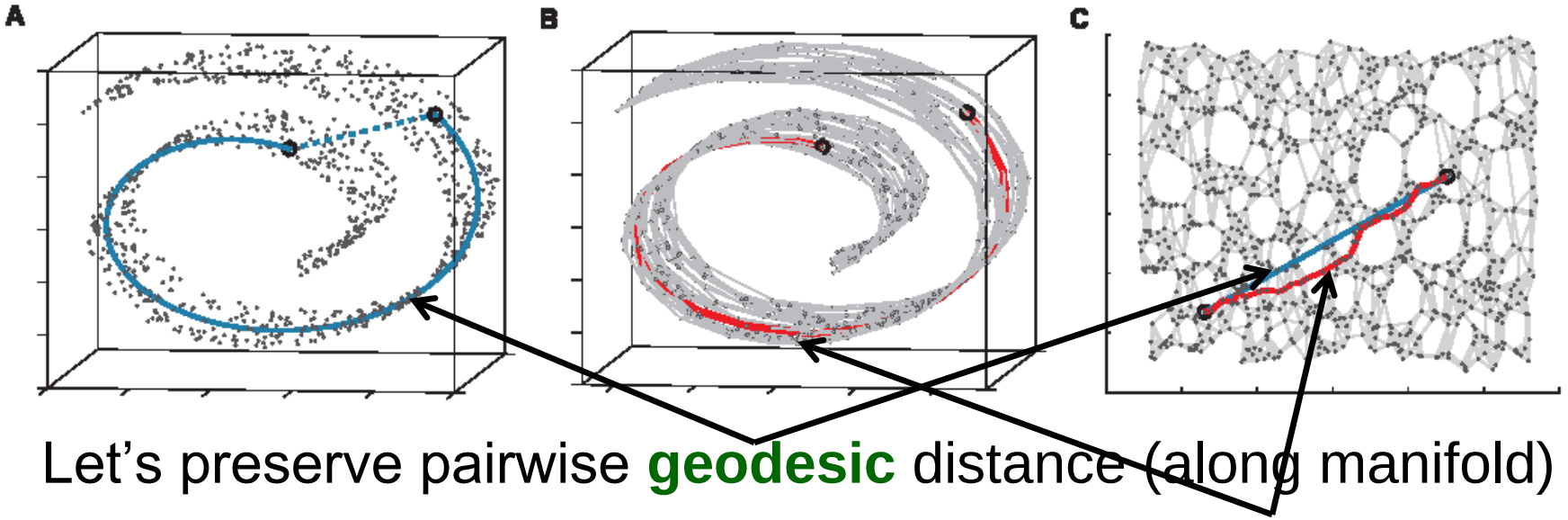
How?

- Each patch of manifold is roughly linear
- Utilize local neighborhood information
  - e.g. for a particular point,
    - Who are my neighbors?
    - What are my relationships to these neighbors?



Demo available at http://www.math.ucla.edu/~wittman/mani/

Isomap
(Isometric Feature Mapping)

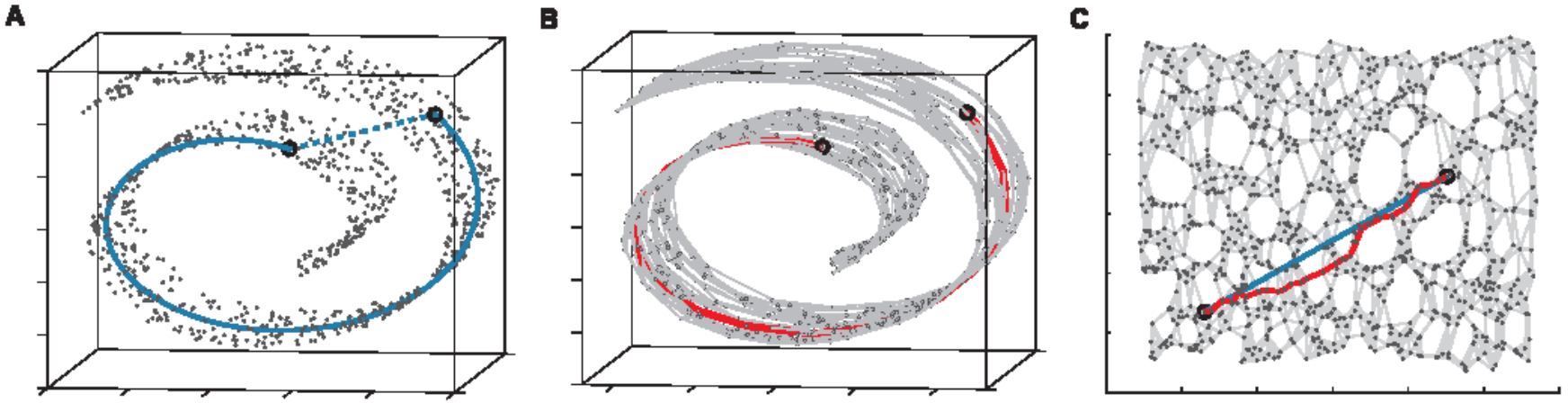


Let's preserve pairwise **geodesic** distance (along manifold)

- Compute geodesic distance as the **shortest path length** from *k*-nearest neighbor (*k*-NN) graph
- *Eigen-decomposition on pairwise geodesic distance matrix to obtain embedding that best preserves given distances

* Eigen-decomposition is the main algorithm of PCA

Isomap
(Isometric Feature Mapping)

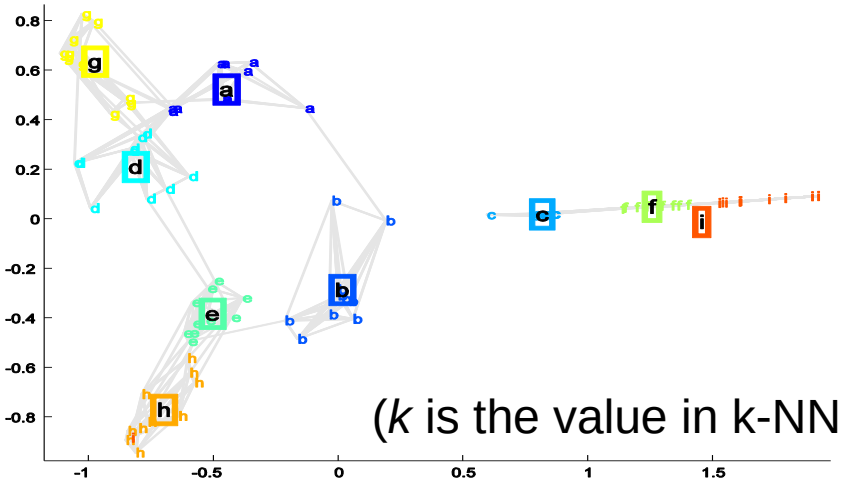


- Algorithm: all-pair shortest path computation + eigen-decomposition
- Pros: performs well in general
- Cons: slow (shortest path), sensitive to parameters

Nonlinear
Unsupervised
Global: all pairwise distances are considered
Feature vectors

Isomap: Effect of Parameter

Facial Data Example



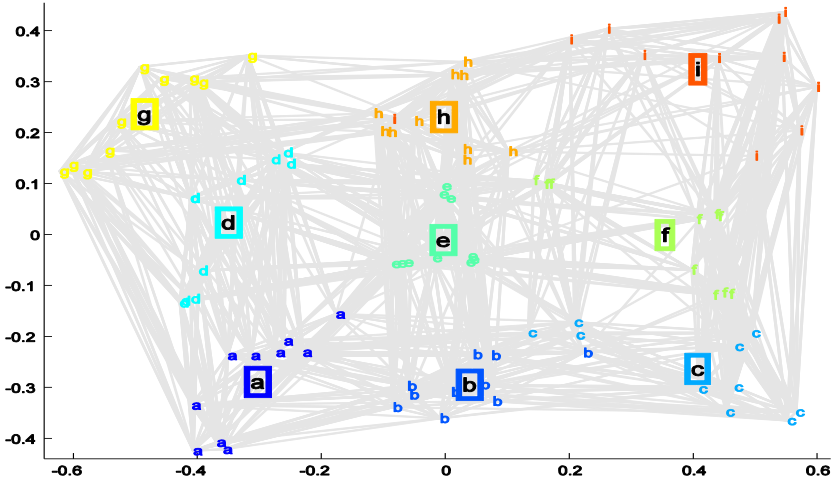
(*k* is the value in k-NN graph)

Angle

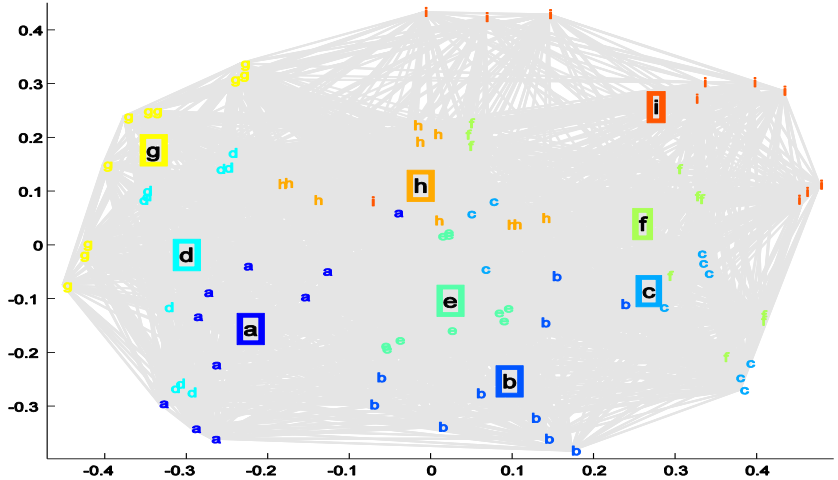
Person ↓

***k*=8**

Cluster structure



***k*=22**



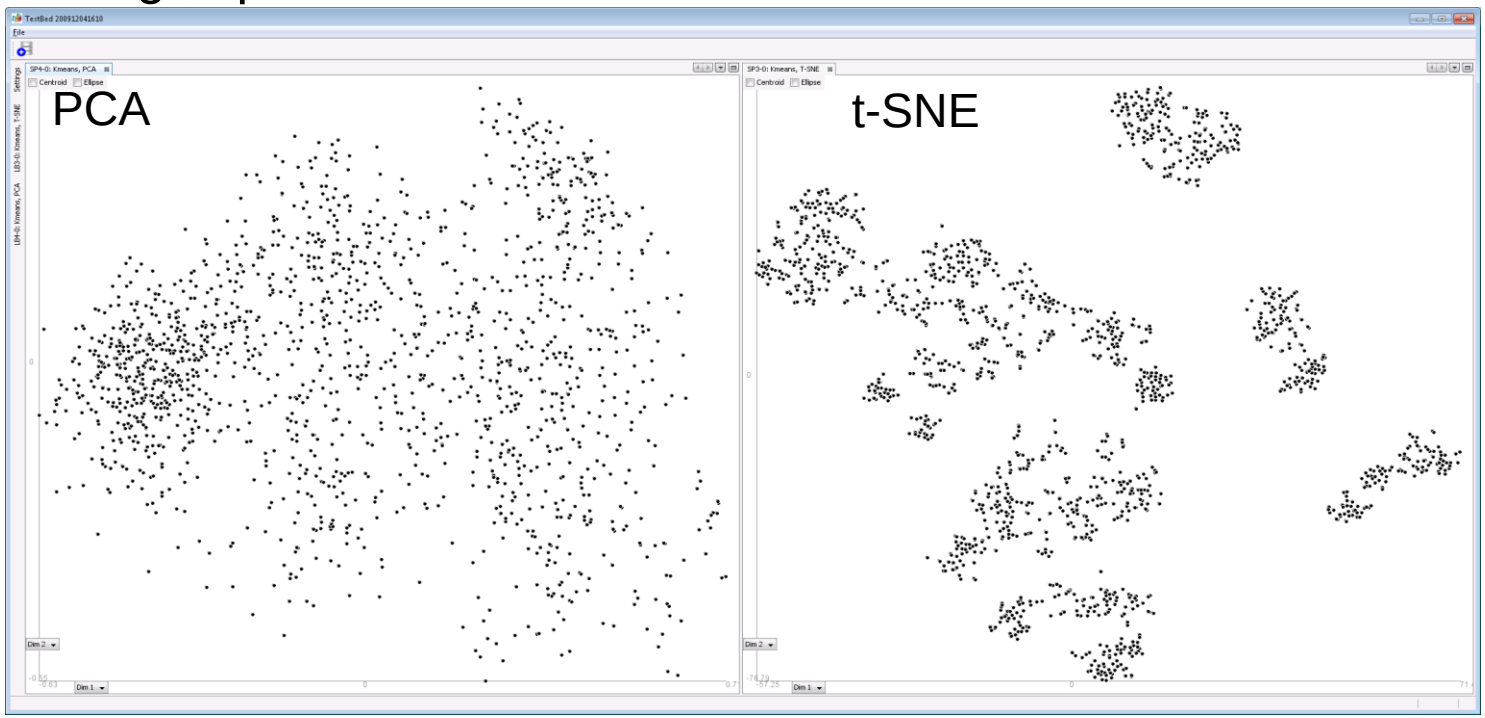
***k*=49**

Which one do you think is the best?

t-SNE
(t-distributed Stochastic Neighborhood Embedding)

Made specifically for visualization (very low dimension)

- Can reveal clusters without any supervision
  - e.g., spoken letter data



http://homepage.tudelft.nl/19j49/t-SNE.html

t-SNE
(t-distributed Stochastic Neighborhood Embedding)

How it works

- Converts distance into probability
  - Farther distance gets lower probability
- Then, minimize differences in probability distribution between high- and low-dimensional spaces
  - KL divergence naturally focuses on neighborhood relationships
- Difference from SNE
  - t-SNE uses heavy-tailed *t*-distribution instead of Gaussian.
  - Suitable for dimension reduction to a very low dimension

t-SNE
(t-distributed Stochastic Neighborhood Embedding)

- Algorithm: gradient-decent type

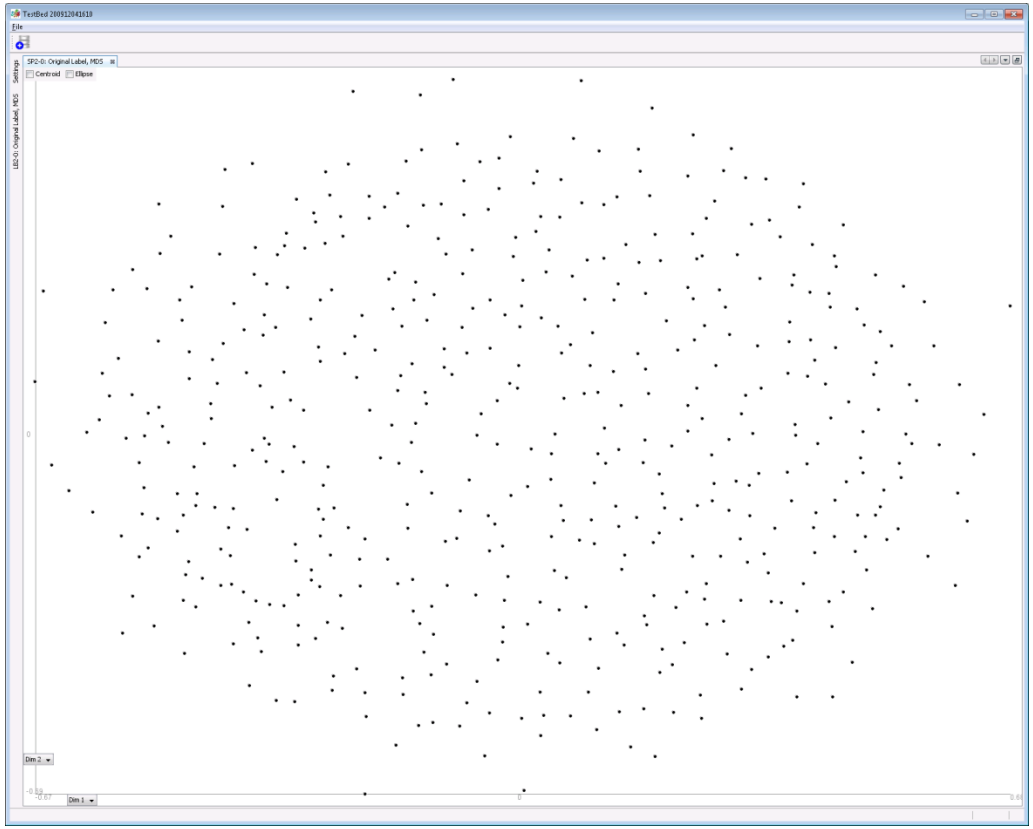
- Pros: works surprisingly well in 2D/3D visualization
- Cons: slow (*n*-body problem)

Nonlinear
Unsupervised
Local
Similarity input

Dimension Reduction in Interactive Visualization

What can you do from visualization via dimension reduction?

- e.g., Multidimensional scaling applied to document data



Dimension Reduction in Interactive Visualization

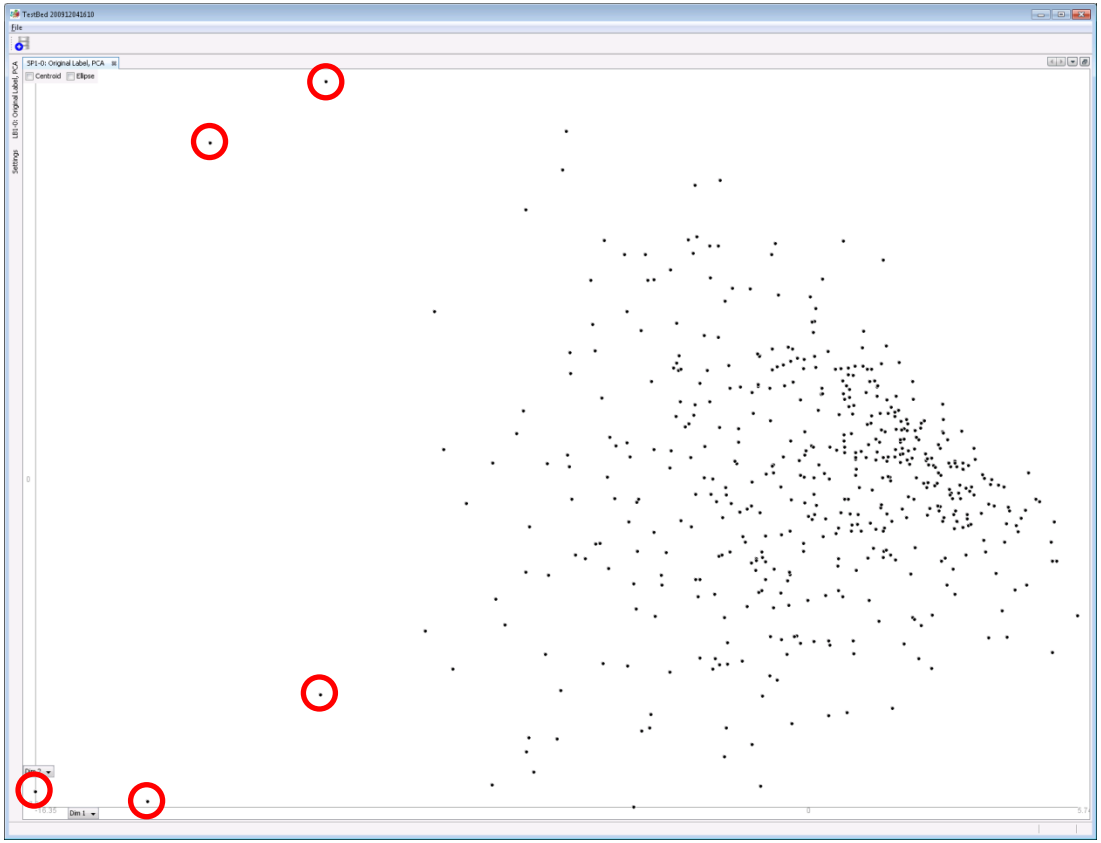
As many data items involve, it's harder to analyze

- For n data items, users are given $O(n^2)$ relations spatially encoded in visualization
  - Too many to understand in general

What to first look at?

Thus, people tend to look for a small number of objects that perceptually/visually stand out, such as …

- Outliers (if any)



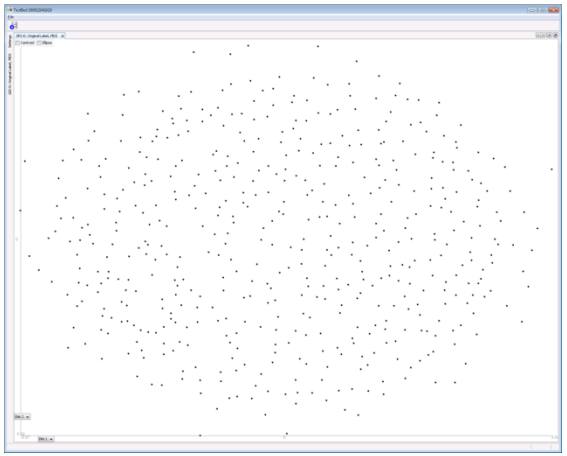
What to first look at?

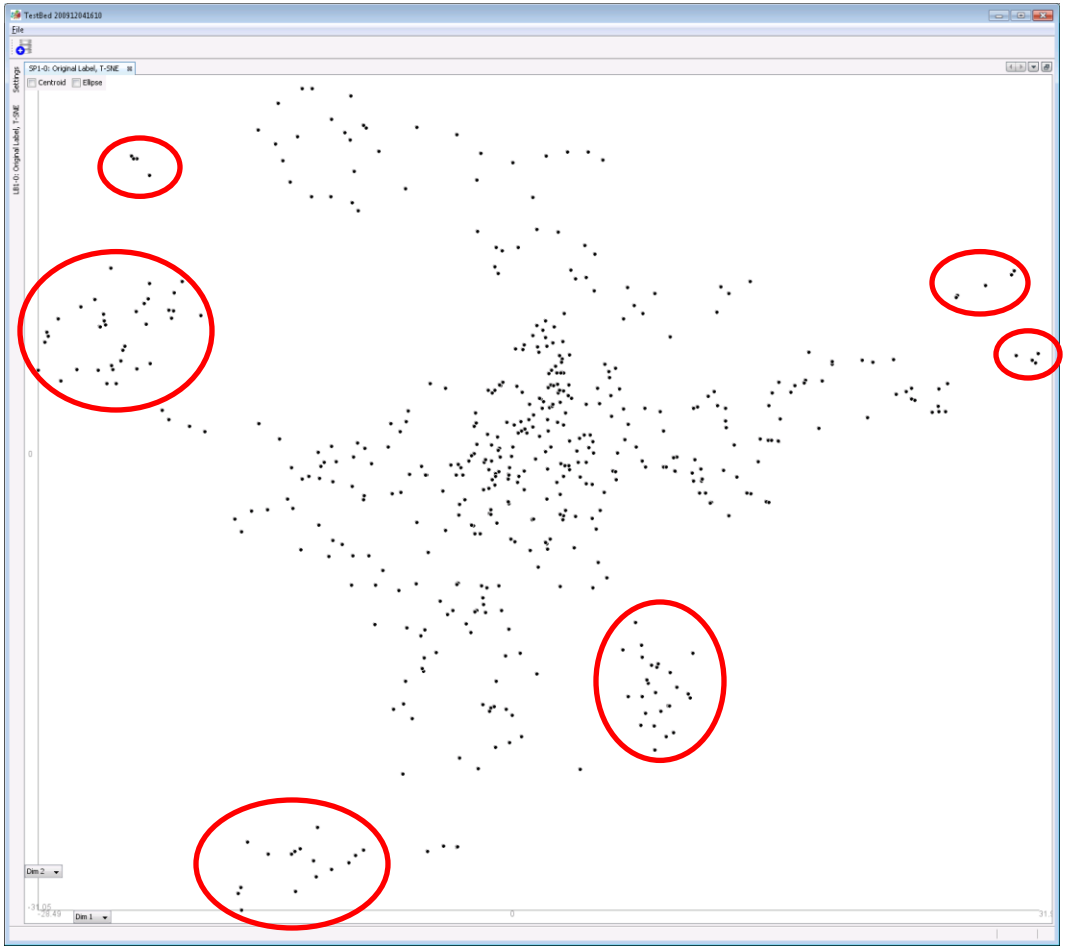
Thus, people tend to look for a small number of objects that perceptually/visually stand out, e.g.,

- Outliers (if any)

More commonly,

- Subgroups/clusters
  - However, it is hard to expect for DR to always reveal clusters, e.g.,





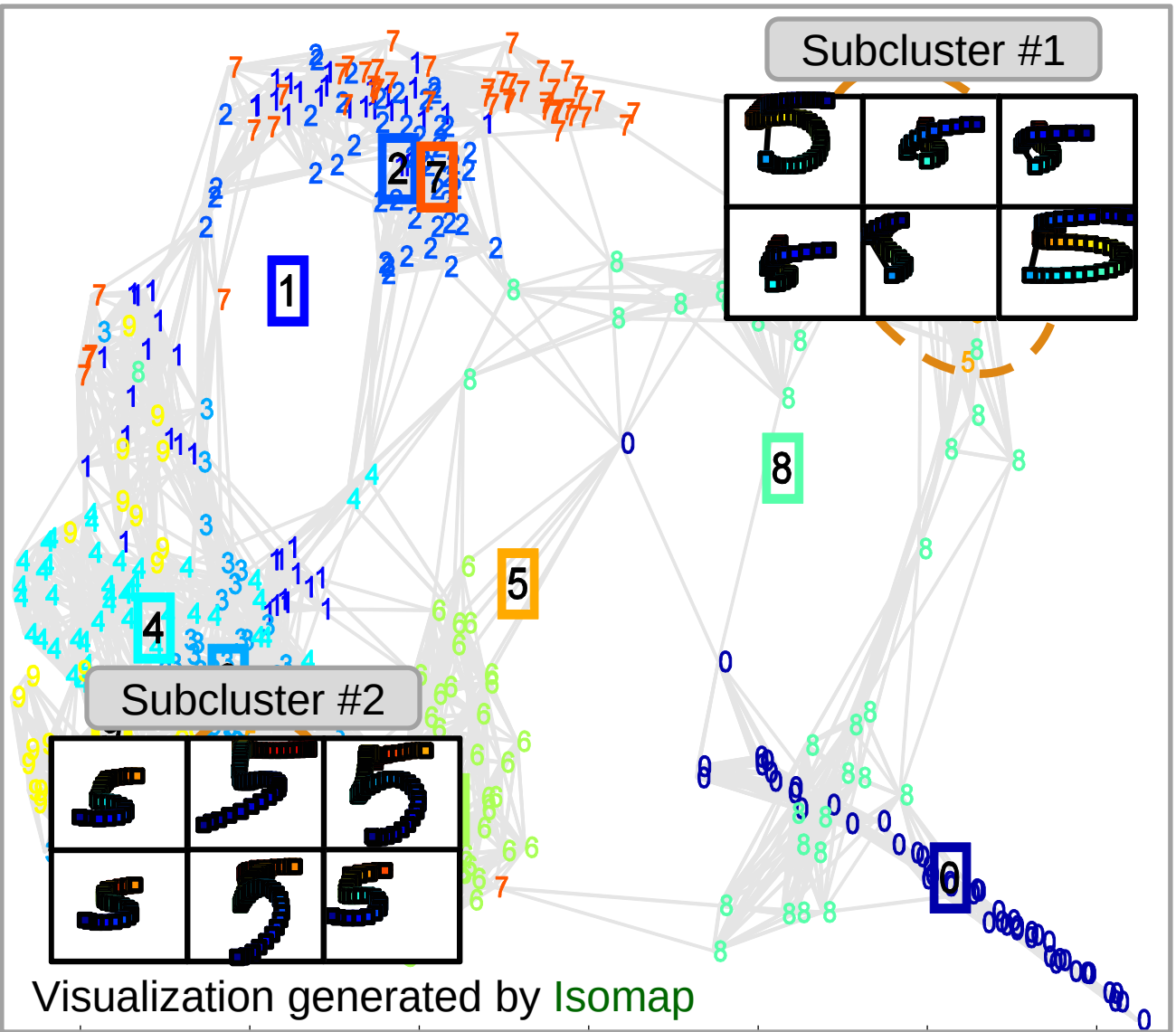
What to first look at?

What if DR cannot reveal subgroups/clusters clearly?

Or even worse, what if our data do not originally have any?

- Often, pre-defined grouping information is imposed and color-coded.
- Grouping information is obtained by
  - Pre-given labels along with data
  - Computed labels by clustering

Insight from Visualization
Handwritten Digit Recognition

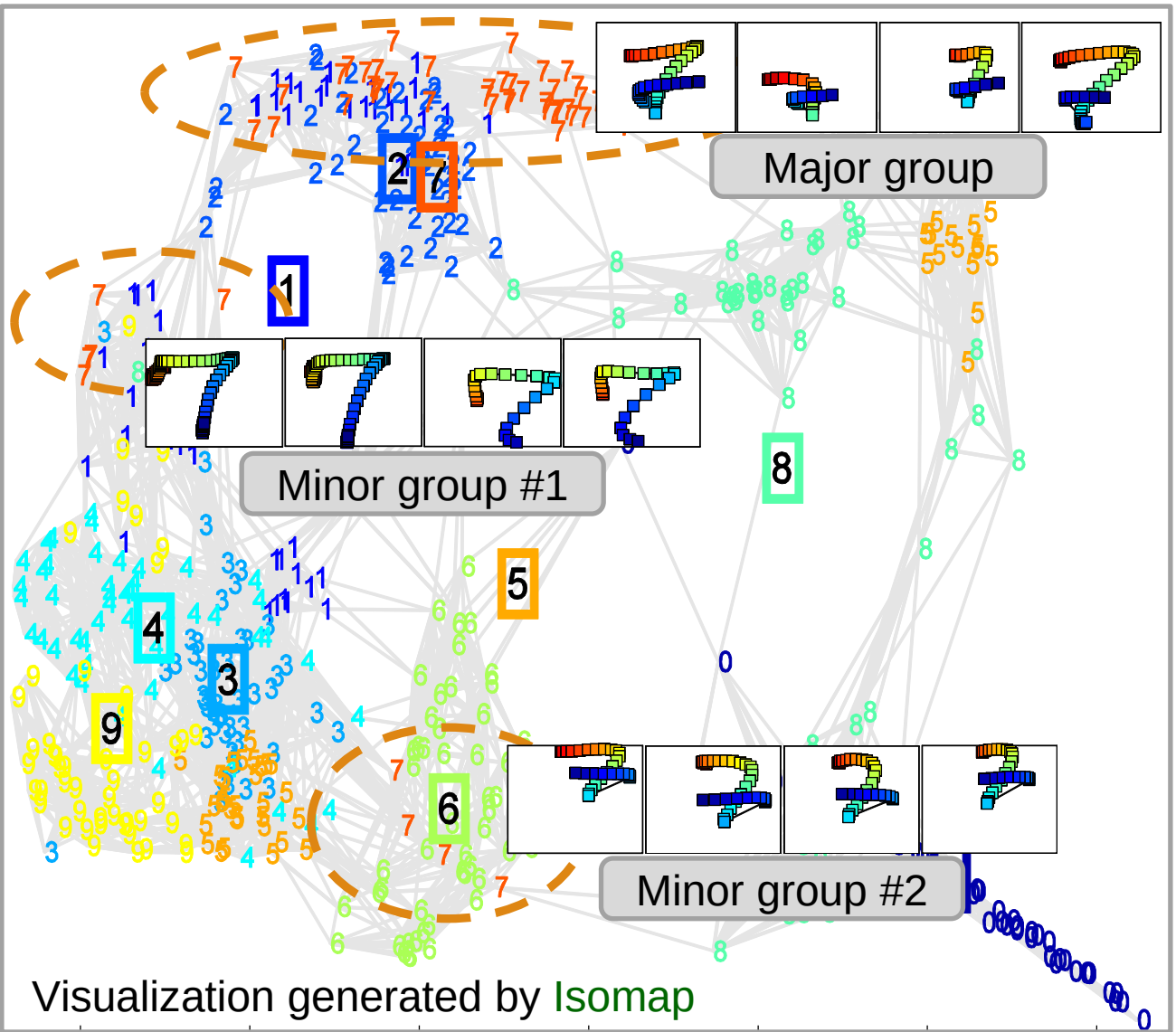


Subclusters in digit '**5**'

Handling them as separate clusters

Better prediction (89%→ 93%)

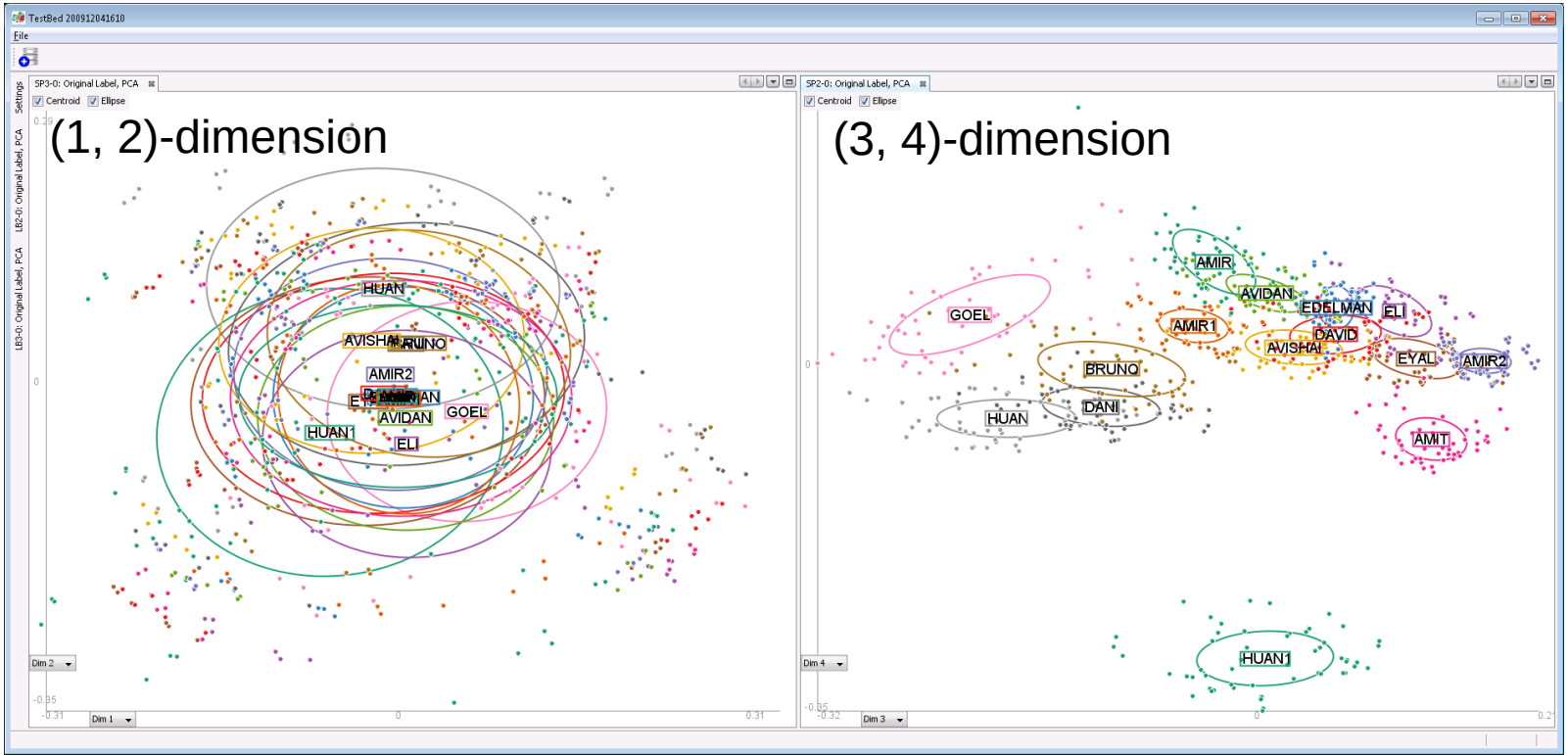
Insight from Visualization
Handwritten Digit Recognition



Practitioner's Guide
Caveats

Trustworthiness of dimension reduction results

- Inevitable distortion/information loss in 2D/3D
- The best result of a method may not align with what we want, e.g., PCA visualization of facial image data



Practitioner's Guide
Caveats

Determining the best method and its parameters for one's own need

- Unlike typical data mining problems where only one shot is allowed, you can freely try out different methods with different parameters
- Basic understanding of methods will greatly help applying them properly
  - What is a particular method trying to achieve? And how suitable is it to your needs?
  - What are the effects of increasing/decreasing parameters?

Practitioner's Guide
General Recommendation

Want something simple and fast to visualize data?

- PCA, force-directed layout

Want to first try some manifold learning methods?

- Isomap
  - if it doesn't show any good, probably neither will anything else.

Have cluster label to use? (pre-given or computed)

- LDA (supervised)
  - Supervised approach is sometimes the only viable option when your data do not have clearly separable clusters

No labels, but still want some clusters to be revealed? Or simply, want some state-of-the-art method for visualization?

- t-SNE

Practitioner's Guide
Results Still Not Good?

Pre-process data properly as needed

- Data centering
  - Subtract the global mean from each vector
- Normalization
  - Make each vector have unit Euclidean norm
  - Otherwise, a few outlier can affect dimension reduction significantly
- Application-specific pre-processing
  - Document: TF-IDF weighting, remove too rare and/or short terms
  - Image: histogram normalization

Practitioner's Guide
Too Slow?

- Apply PCA to reduce to an intermediate dimensions before the main dimension reduction step
  - t-SNE does it by default
  - The results may be even better due to noise removed by PCA
- See if there is any approximated but faster version
  - Landmarked versions (only using a subset of data items)
    - e.g., landmarked Isomap
  - Linearized versions (the same criterion, but only allow linear mapping)
    - e.g., Laplacian Eigenmaps → Locality preserving projection

Practitioner's Guide
Still need more?

Tweak dimension reduction for your own purpose

- Play with its algorithm, convergence criteria, etc.
  - See if you can impose label information

Original t-SNE

t-SNE with simple modification

Practitioner's Guide
Still need more?

Tweak dimension reduction for your own purpose

- Play with its algorithm, convergence criteria, etc.
  - See if you can impose label information
  - Restrict the number of iterations to save computational time.

The main purpose of DR is to serve us in exploring data and solving complicated real-world problems

Useful Resource

Review article

- http://www.iai.uni-bonn.de/~jz/dimensionality_reduction_a_comparative_review.pdf

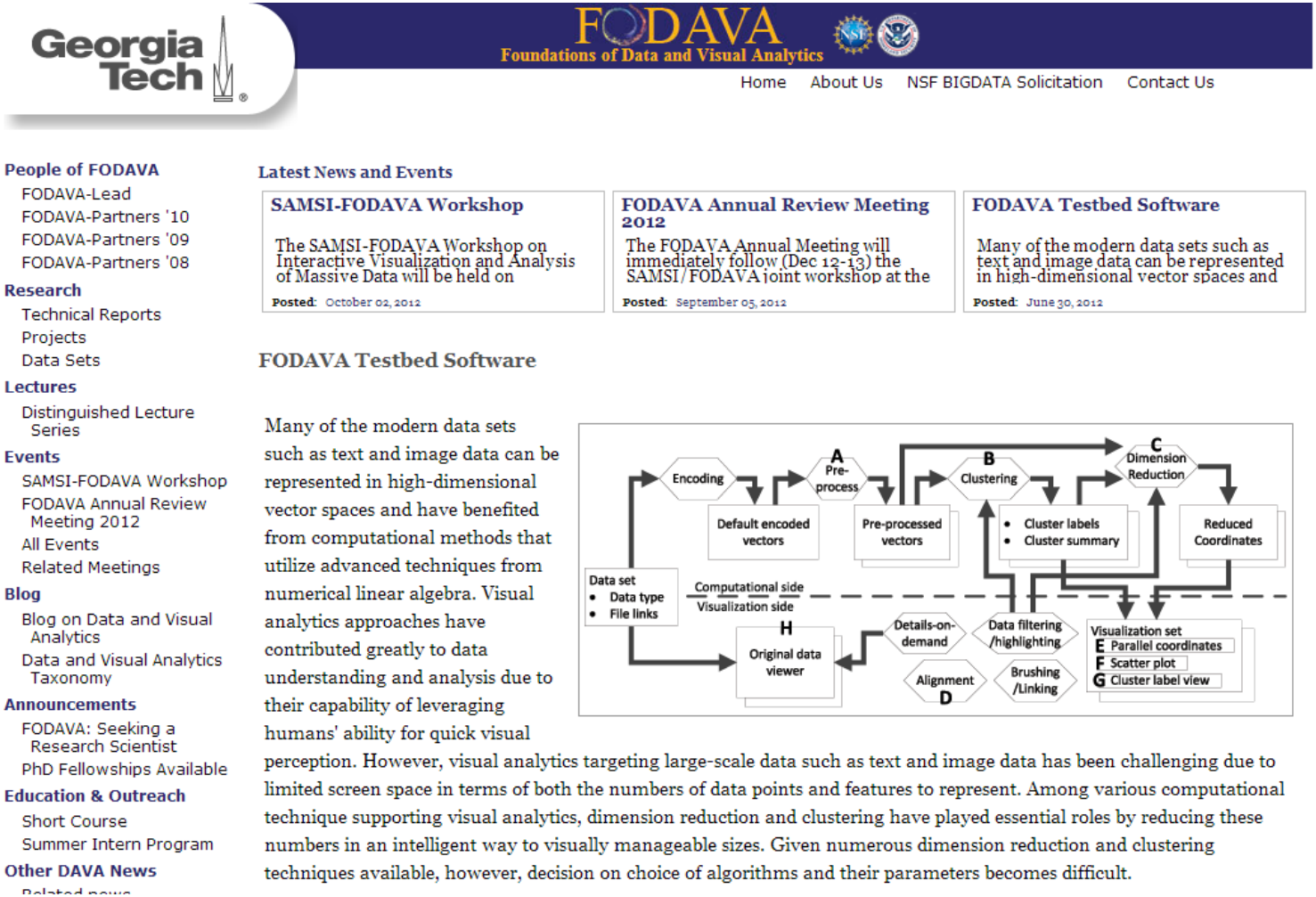
Matlab toolbox for dimension reduction

- http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html

Matlab manifold learning demo

- http://www.math.ucla.edu/~wittman/mani/

Useful Resource
FODAVA Testbed Software



Available at http://fodava.gatech.edu/fodava-testbed-software

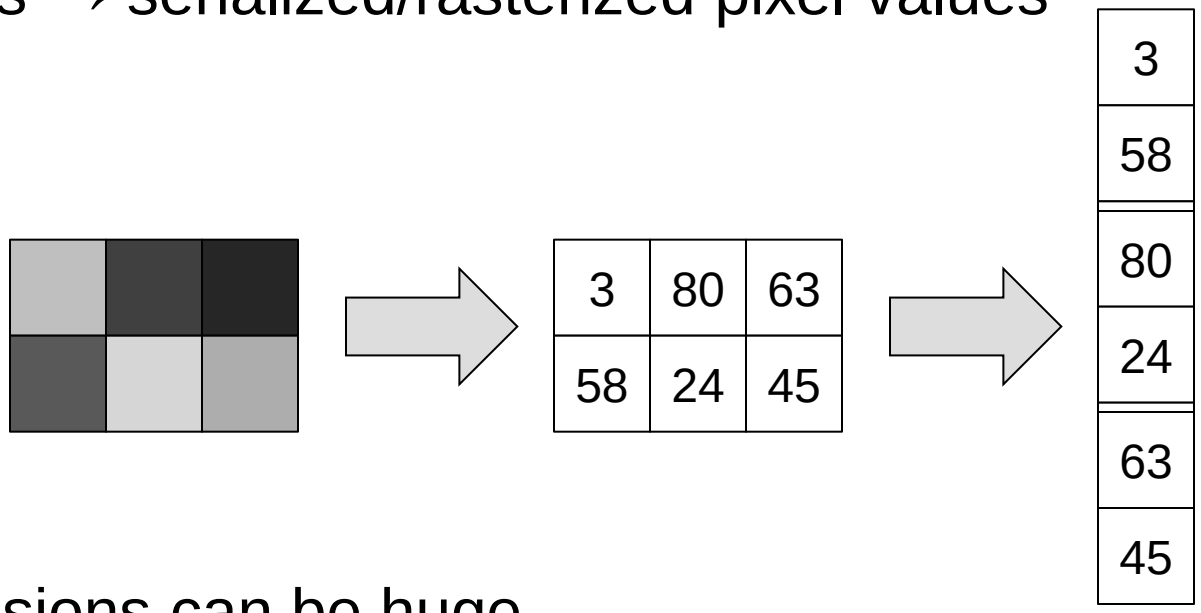
For a recent version, contact me at jaegul.choo@cc.gatech.edu

Thank You

Jaegul Choo
jaegul.choo@cc.gatech.edu
http://www.cc.gatech.edu/~joyfull/

Face Recognition
Vector Representation of Images

- Images → serialized/rasterized pixel values



- Dimensions can be huge.
  - 640x480 size: 307,200 dimensions

Face Recognition

Vector Representation of Images

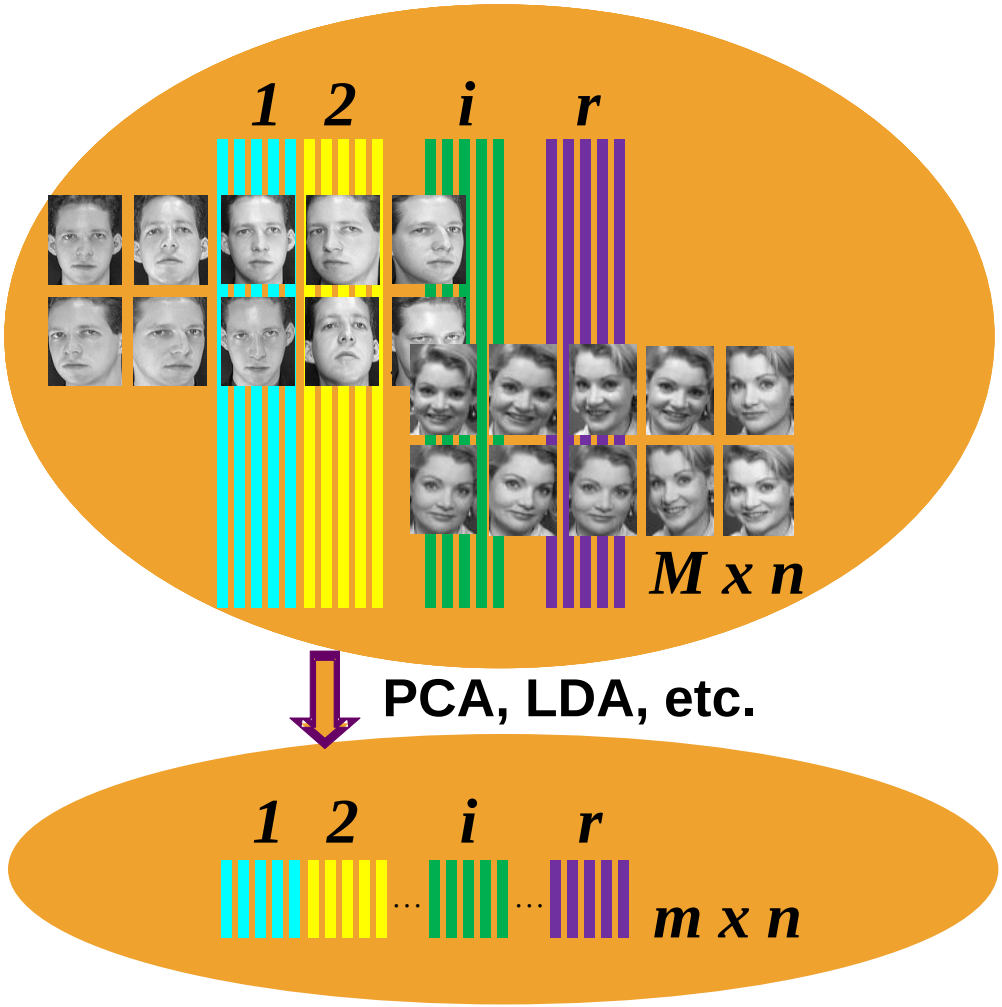
↓

Dimension Reduction

↓

Classification on dim-reduced data

→ Better accuracy than on original-dim data



Document Retrieval

Latent semantic indexing

- Term-document matrix via bag-of-words model
  - D1 = “I like data”
  - D2 = “I hate hate data”
- Dimensions can be hundreds of thousands
  - i.e., #distinct words

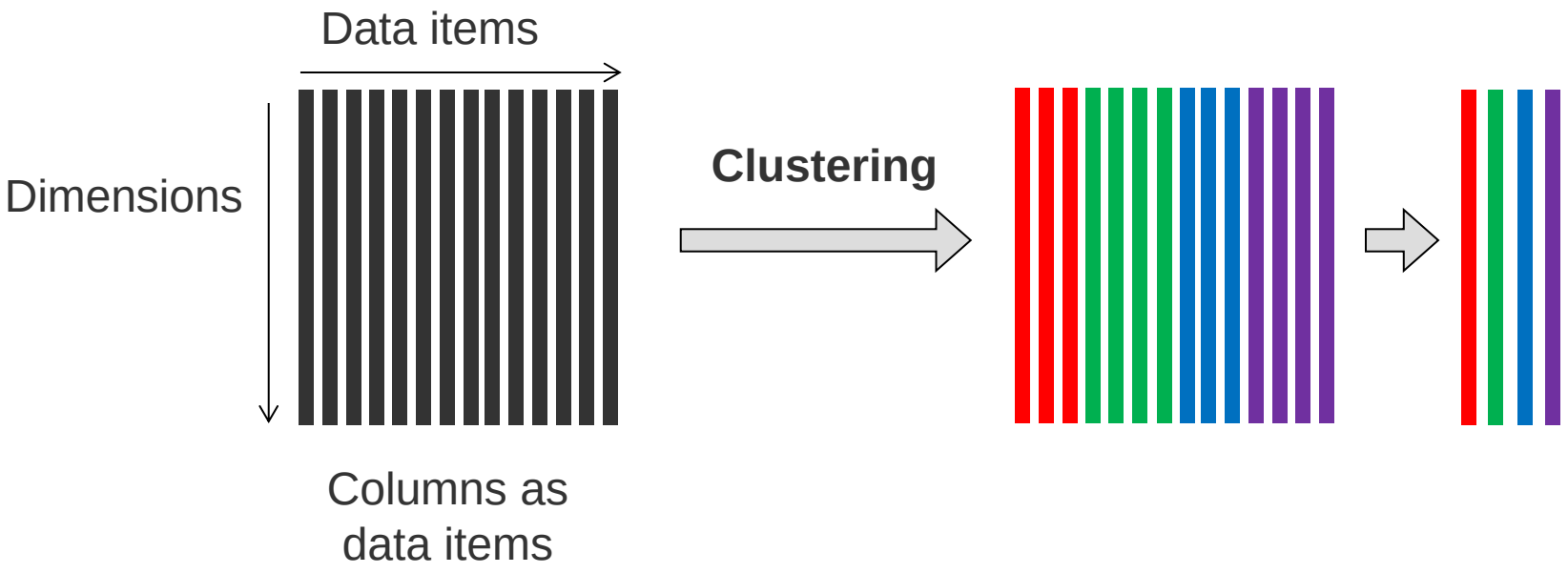
	D1	D2	…
I	1	1	…
like	1	0	…
hate	0	2	…
data	1	1	…
…	…	…	

→ Dimension Reduction →

	D1	D2	…
Dim1	1.75	-0.27	…
Dim2	-0.21	0.58	…
Dim3	1.32	0.25	

→ Search-Retrieval on dim-reduced data leads to better semantics

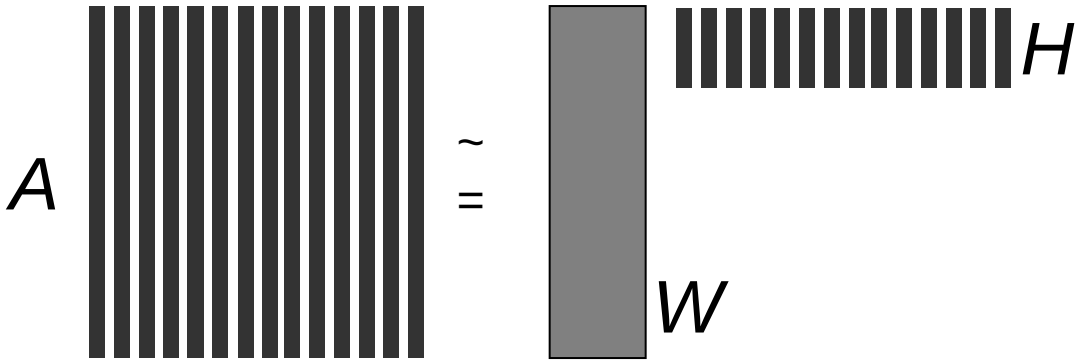
Other Techniques for High-Dim Data
Clustering



- e.g., *K*-means, hierarchical clustering, Gaussian mixture model, nonnegative matrix factorization, semi-supervised methods, …

Nonnegative Matrix Factorization

Dimension reduction via matrix factorization



$\rightarrow min \| A - WH \|_F$

W>=0, H>=0

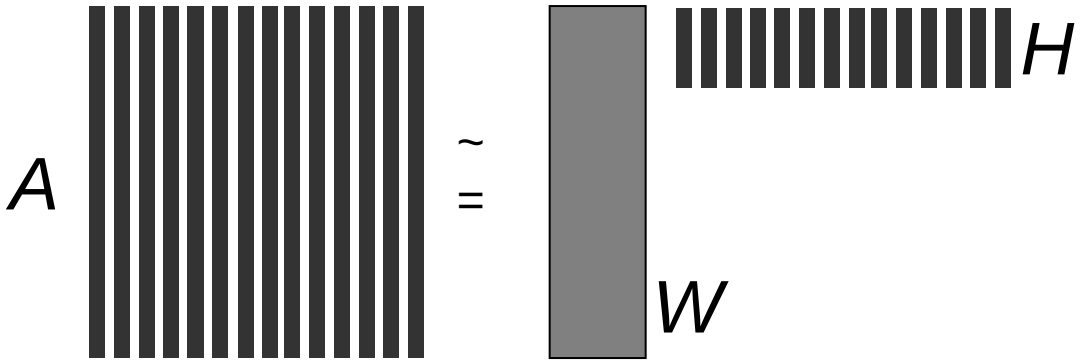
Why nonnegativity constraints?

- Better approximation vs. better interpretation
- Often physically/semantically meaningful

- Algorithm: alternating nonnegativity-constrained least squares

Nonnegative Matrix Factorization
as clustering

Dimension reduction via matrix factorization



➔ $min \; || A - WH ||_F$

$W>=0, H>=0$

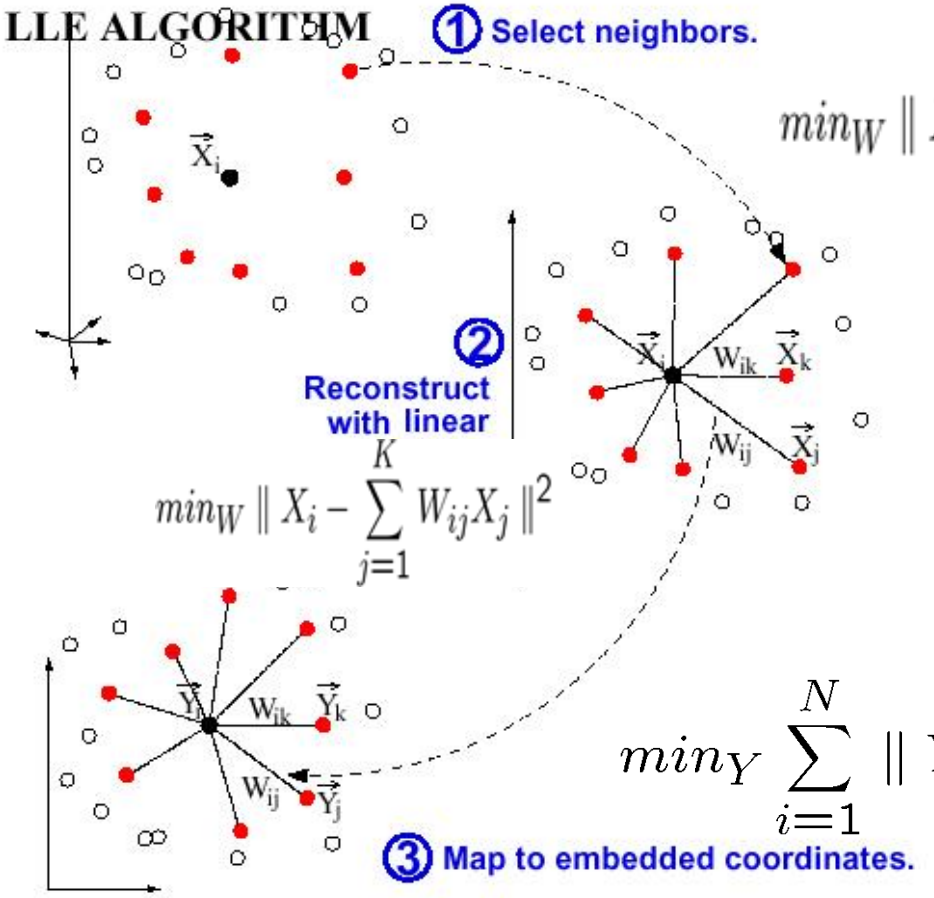
Often NMF performs better and faster than *k*-means

- *W*: centroids, *H*: soft-clustering membership

LLE (Locally Linear Embedding) Classical Method

Let's preserve linear reconstruction weight from neighbors

1. **Compute the neighbors of each data point, $\vec{X}_i$.**
2. **Compute the weights W_{ij} that best reconstruct each data point $\vec{X}_i$ from its neighbors, minimizing the cost in Equation (1) by constrained linear fits.**
3. **Compute the vectors $\vec{Y}_i$ best reconstructed by the weights W_{ij}, minimizing the quadratic form in Equation (2) by its bottom nonzero eigenvectors.**



LLE (Locally Linear Embedding) Classical Method

- Algorithm: least squares + eigen-decomposition
- Pros: fast
- Cons: often outperformed by other methods

Nonlinear
Unsupervised
Local
Feature vectors

Laplacian Eigenmaps
Another Classical Method

Criterion

$$\min \sum_{ij} (y_i - y_j)^2 W_{ij} \text{ , where}$$

$$W_{ij} = \exp(-||x_i - x_j|| / t) \text{ for neighbors, } W_{ij} = 0 \text{ otherwise.}$$

x_i : high-dimensional vector, y_i : low-dimensional data vector

t : user-specified parameter

Main idea

- Basically, fit all the distances to zero
- But, closer distances -> higher weight W_{ij} -> fit to zero more strongly

c.f.) MDS criterion: fits them to given ideal distances

$$\min \sum_{ij} (||y_i - y_j|| - d_{ij})^2$$

Laplacian Eigenmaps
Another Classical Method

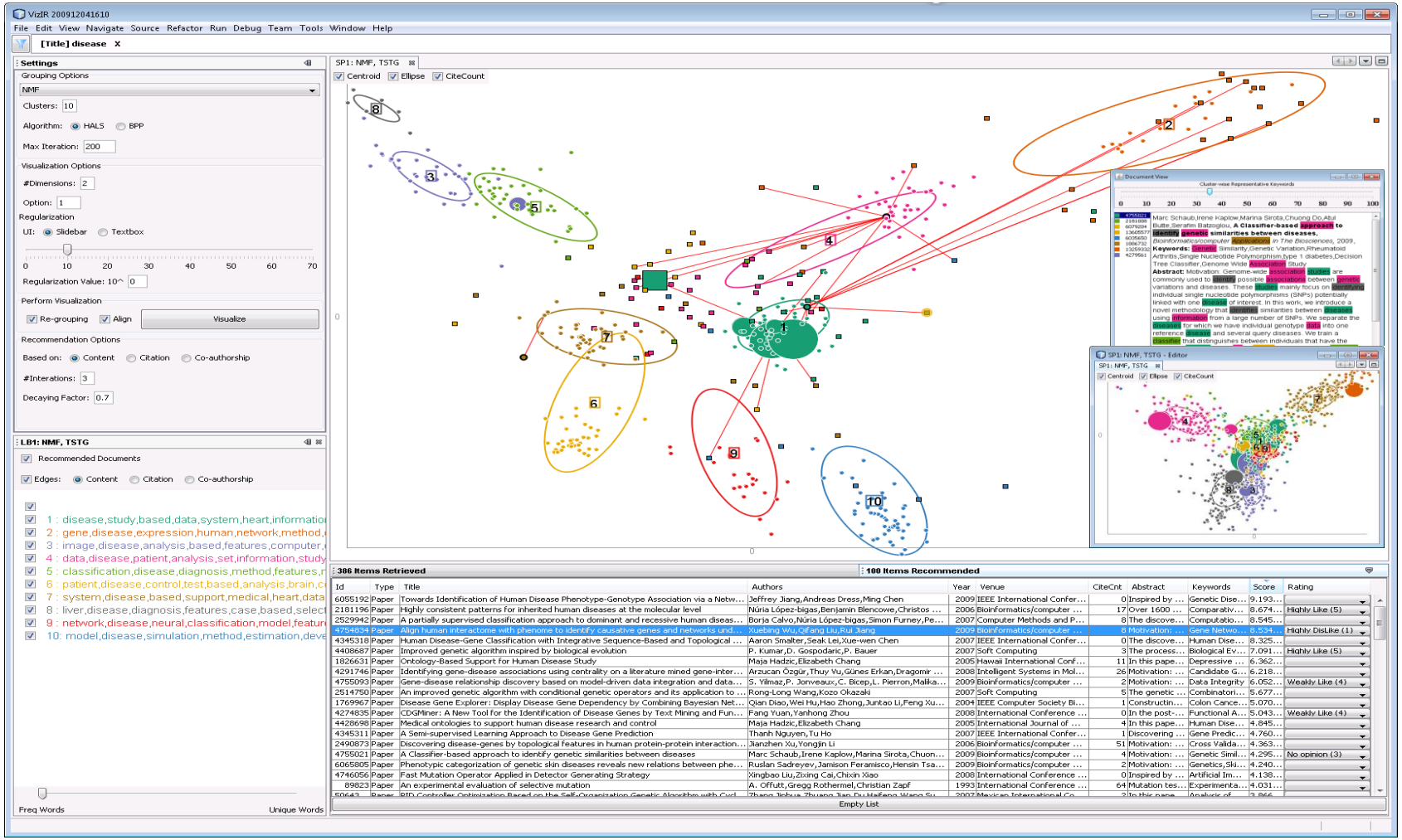
- Nicely backed by graph theory and algorithm
- Algorithm: generalized eigen-decomposition

- Pros: really fast
- Cons: Has two parameters difficult to determine

Nonlinear
Unsupervised
Local
Similarity input

Dimension Reduction in Action
Visual Analytics for Document

VisIRR: Visual Information Retrieval and Recommendation



Or…

Start your own research to make DR more useful in interactive visualization!

Gold Mine for Researchers

Compared to recent advancement in dimension reduction, its application in visualization is highly under-explored

That is, dimension reduction for visualization still needs to

- Be faster
- Be more interpretable
- Give more semantically meaningful results
- Be more interactive and responsive to users

Research Example
Visualize It-Wise*

Motivation

- Force directed layout revisited
  - http://prefuse.org/gallery/graphview/

- Why not making other methods like this?

*Joint work with Changhyun Lee and Haesun Park