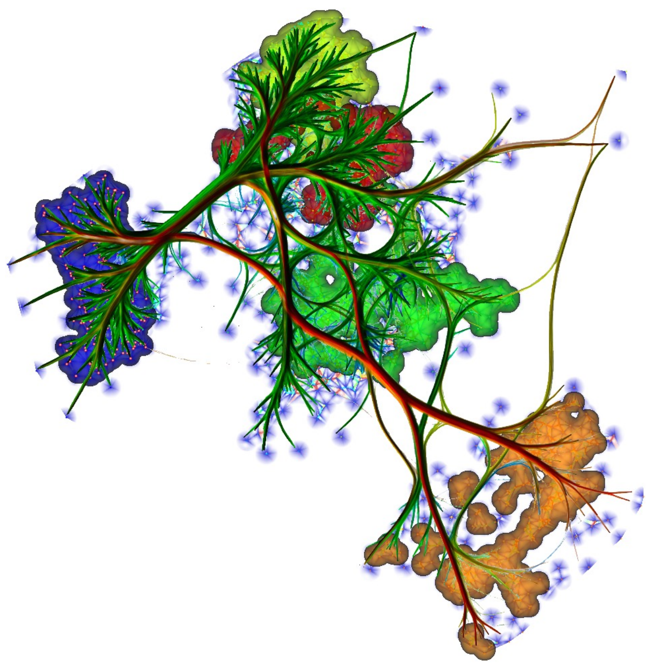
Multidimensional Data Visualization

High-dimensional Data



prof. dr. Alexandru (Alex) Telea

Department of Information and Computing Science
Utrecht University, the Netherlands

Utrecht University

Summary: Low-dimensional data visualization

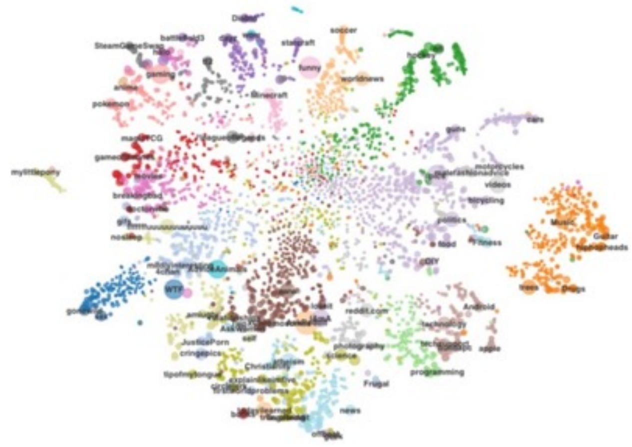
For what
- datasets with many samples *N* but few (2..10) dimensions *n*

Main design idea
- allocate one visual variable for one..a few dimensions

Techniques
- scatterplots, scatterplot matrices
- table lenses
- table-tree duality
- icicle plots, treemaps
- parallel coordinates

Open challenge: What to do with many dimensions?

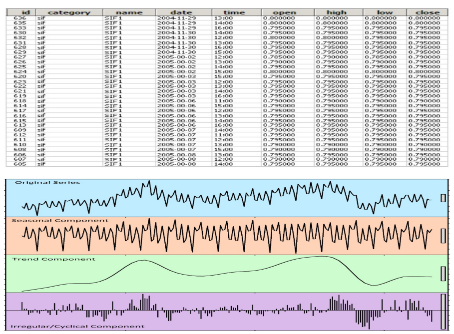
1. Multidimensional Projections

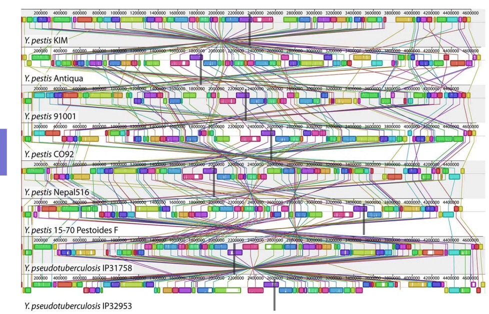


What is *really high-dimensional* data?

Hundreds of dimensions with often no clear meaning

High-dimensional data → **ML / DL**





Solution: Multidimensional projections

What is a multidimensional projection?

Short answer: A tool to look into really high-dimensional data

Longer answer

Consider a multivariate dataset, like a **table**

$$T = \{r_i\}_{i=1..m}\ ,\quad r = \{c_j\}_{j=1..n}\ ,\quad c_j \in D_j\ ,\ D_j \subseteq \mathit{Continuous} \cup \mathit{Discrete} \cup \mathit{Ordinal} \cup \mathit{Categorical}$$

- rows: observations (measurements)
- columns: dimensions (attributes)

How to visualize T?

id	category	name	date	time	open	high	low	close
636	sif	SIF1	2004-11-29	13:00	0.800000	0.800000	0.800000	0.800000
635	sif	SIF1	2004-11-29	14:00	0.800000	0.800000	0.800000	0.800000
633	sif	SIF1	2004-11-29	16:00	0.795000	0.795000	0.795000	0.795000
630	sif	SIF1	2004-11-30	14:00	0.795000	0.795000	0.795000	0.795000
632	sif	SIF1	2004-11-30	12:00	0.800000	0.800000	0.795000	0.795000
631	sif	SIF1	2004-11-30	13:00	0.795000	0.795000	0.795000	0.795000
628	sif	SIF1	2004-11-30	16:00	0.795000	0.795000	0.795000	0.795000
629	sif	SIF1	2004-11-30	15:00	0.795000	0.795000	0.795000	0.795000
627	sif	SIF1	2005-00-02	12:00	0.785000	0.790000	0.785000	0.790000
626	sif	SIF1	2005-00-02	13:00	0.790000	0.795000	0.790000	0.795000
625	sif	SIF1	2005-00-02	14:00	0.795000	0.795000	0.795000	0.795000
624	sif	SIF1	2005-00-02	15:00	0.800000	0.800000	0.800000	0.800000
620	sif	SIF1	2005-00-03	15:00	0.795000	0.795000	0.795000	0.795000
623	sif	SIF1	2005-00-03	12:00	0.795000	0.795000	0.795000	0.795000
622	sif	SIF1	2005-00-03	13:00	0.795000	0.795000	0.795000	0.795000
621	sif	SIF1	2005-00-03	14:00	0.795000	0.795000	0.795000	0.795000
619	sif	SIF1	2005-00-03	16:00	0.795000	0.795000	0.795000	0.795000
618	sif	SIF1	2005-00-06	11:00	0.790000	0.790000	0.790000	0.790000
614	sif	SIF1	2005-00-06	15:00	0.795000	0.795000	0.795000	0.795000
617	sif	SIF1	2005-00-06	12:00	0.795000	0.795000	0.795000	0.795000
616	sif	SIF1	2005-00-06	13:00	0.795000	0.795000	0.795000	0.795000
615	sif	SIF1	2005-00-06	14:00	0.795000	0.795000	0.795000	0.795000
613	sif	SIF1	2005-00-06	16:00	0.795000	0.795000	0.795000	0.795000
609	sif	SIF1	2005-00-07	14:00	0.790000	0.795000	0.790000	0.795000
612	sif	SIF1	2005-00-07	11:00	0.795000	0.795000	0.795000	0.795000
611	sif	SIF1	2005-00-07	12:00	0.795000	0.795000	0.795000	0.795000
610	sif	SIF1	2005-00-07	13:00	0.790000	0.790000	0.790000	0.790000
608	sif	SIF1	2005-00-07	15:00	0.790000	0.790000	0.790000	0.790000
606	sif	SIF1	2005-00-08	13:00	0.795000	0.795000	0.795000	0.795000
607	sif	SIF1	2005-00-08	12:00	0.790000	0.790000	0.790000	0.790000
605	sif	SIF1	2005-00-08	14:00	0.795000	0.795000	0.795000	0.795000

m transactions

n attributes (fields) of a transaction

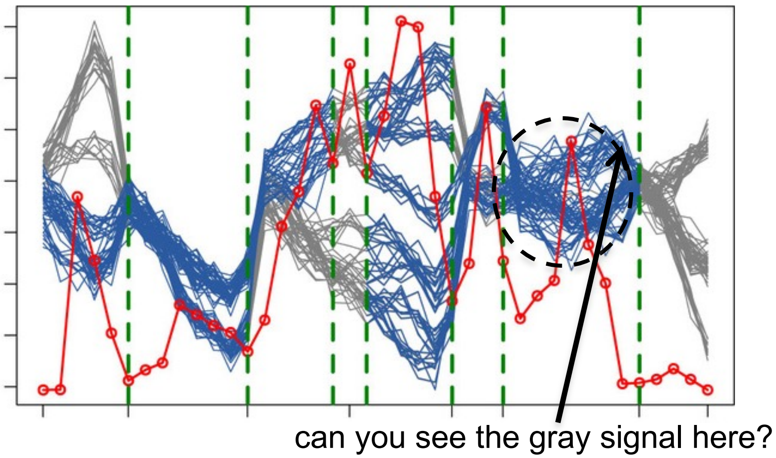
Example: stock exchange data

Drawing a **large** table
(> 10 columns/rows)
becomes **useless**…

Visualizing high-dimensional data

Methods discussed so far do not scale well ☹

Multivariate charts



Parallel coordinates

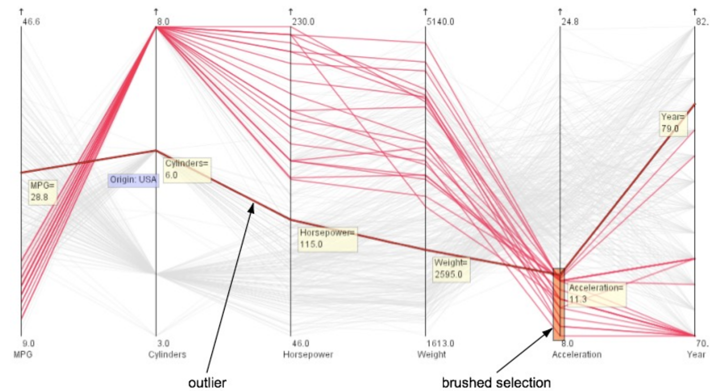
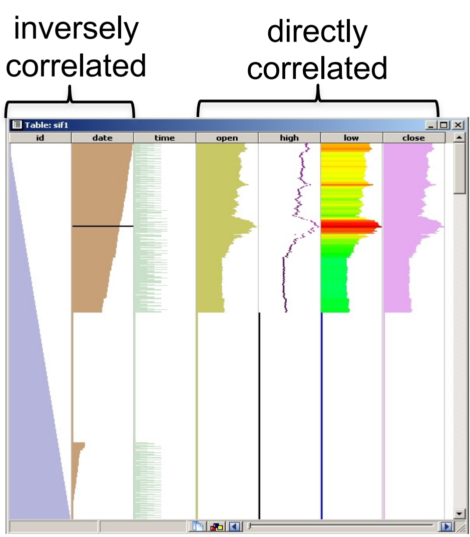
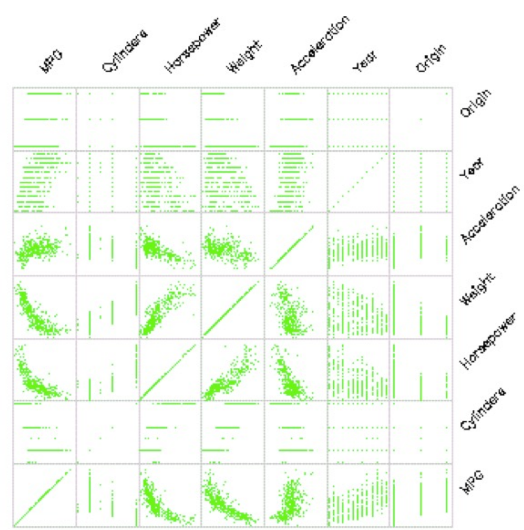


Table lenses



Scatterplot matrices



Projections

Table

n attributes

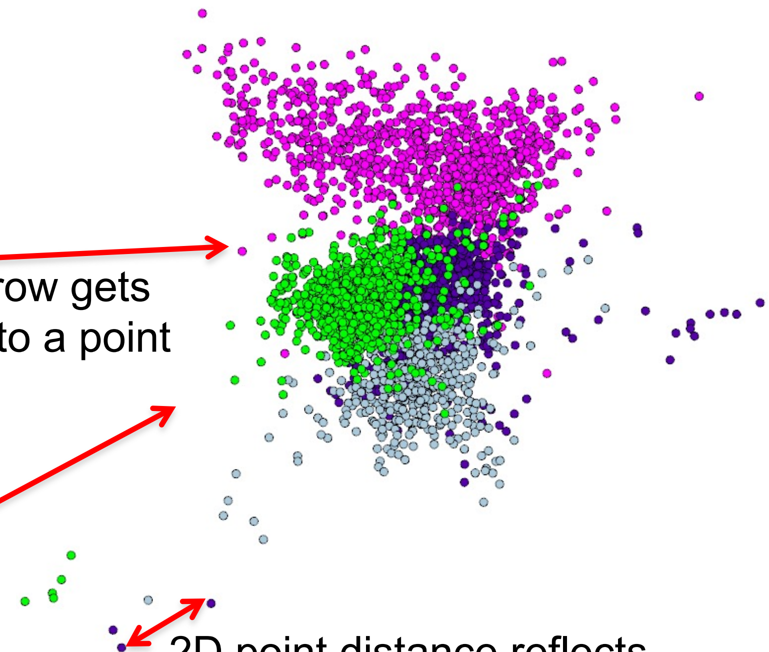
m observations

id	category	name	date	time	open	high	low	close
636	sif	SIF1	2004-11-29	13:00	0.800000	0.800000	0.800000	0.800000
635	sif	SIF1	2004-11-29	14:00	0.800000	0.800000	0.800000	0.800000
633	sif	SIF1	2004-11-29	16:00	0.795000	0.795000	0.795000	0.795000
630	sif	SIF1	2004-11-30	14:00	0.795000	0.795000	0.795000	0.795000
632	sif	SIF1	2004-11-30	12:00	0.800000	0.800000	0.795000	0.795000
631	sif	SIF1	2004-11-30	13:00	0.795000	0.795000	0.795000	0.795000
628	sif	SIF1	2004-11-30	16:00	0.795000	0.795000	0.795000	0.795000
629	sif	SIF1	2004-11-30	15:00	0.795000	0.795000	0.795000	0.795000
627	sif	SIF1	2005-00-02	12:00	0.785000	0.790000	0.785000	0.790000
626	sif	SIF1	2005-00-02	13:00	0.790000	0.795000	0.790000	0.795000
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
624	sif	SIF1	2005-00-02	15:00	0.800000	0.800000	0.800000	0.800000
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
623	sif	SIF1	2005-00-03	12:00	0.795000	0.795000	0.795000	0.795000
622	sif	SIF1	2005-00-03	13:00	0.795000	0.795000	0.795000	0.795000
621	sif	SIF1	2005-00-03	14:00	0.795000	0.795000	0.795000	0.795000
619	sif	SIF1	2005-00-03	16:00	0.795000	0.795000	0.795000	0.795000
618	sif	SIF1	2005-00-06	11:00	0.790000	0.790000	0.790000	0.790000
614	sif	SIF1	2005-00-06	15:00	0.795000	0.795000	0.795000	0.795000
617	sif	SIF1	2005-00-06	12:00	0.795000	0.795000	0.795000	0.795000
616	sif	SIF1	2005-00-06	13:00	0.795000	0.795000	0.795000	0.795000
615	sif	SIF1	2005-00-06	14:00	0.795000	0.795000	0.795000	0.795000
613	sif	SIF1	2005-00-06	16:00	0.795000	0.795000	0.795000	0.795000
609	sif	SIF1	2005-00-07	14:00	0.790000	0.795000	0.790000	0.795000
612	sif	SIF1	2005-00-07	11:00	0.795000	0.795000	0.795000	0.795000
611	sif	SIF1	2005-00-07	12:00	0.795000	0.795000	0.795000	0.795000
610	sif	SIF1	2005-00-07	13:00	0.790000	0.790000	0.790000	0.790000
608	sif	SIF1	2005-00-07	15:00	0.790000	0.790000	0.790000	0.790000
606	sif	SIF1	2005-00-08	13:00	0.795000	0.795000	0.795000	0.795000
607	sif	SIF1	2005-00-08	12:00	0.790000	0.790000	0.790000	0.790000
605	sif	SIF1	2005-00-08	14:00	0.795000	0.795000	0.795000	0.795000

color map values of
a selected column

2D projection

a table row gets
mapped to a point

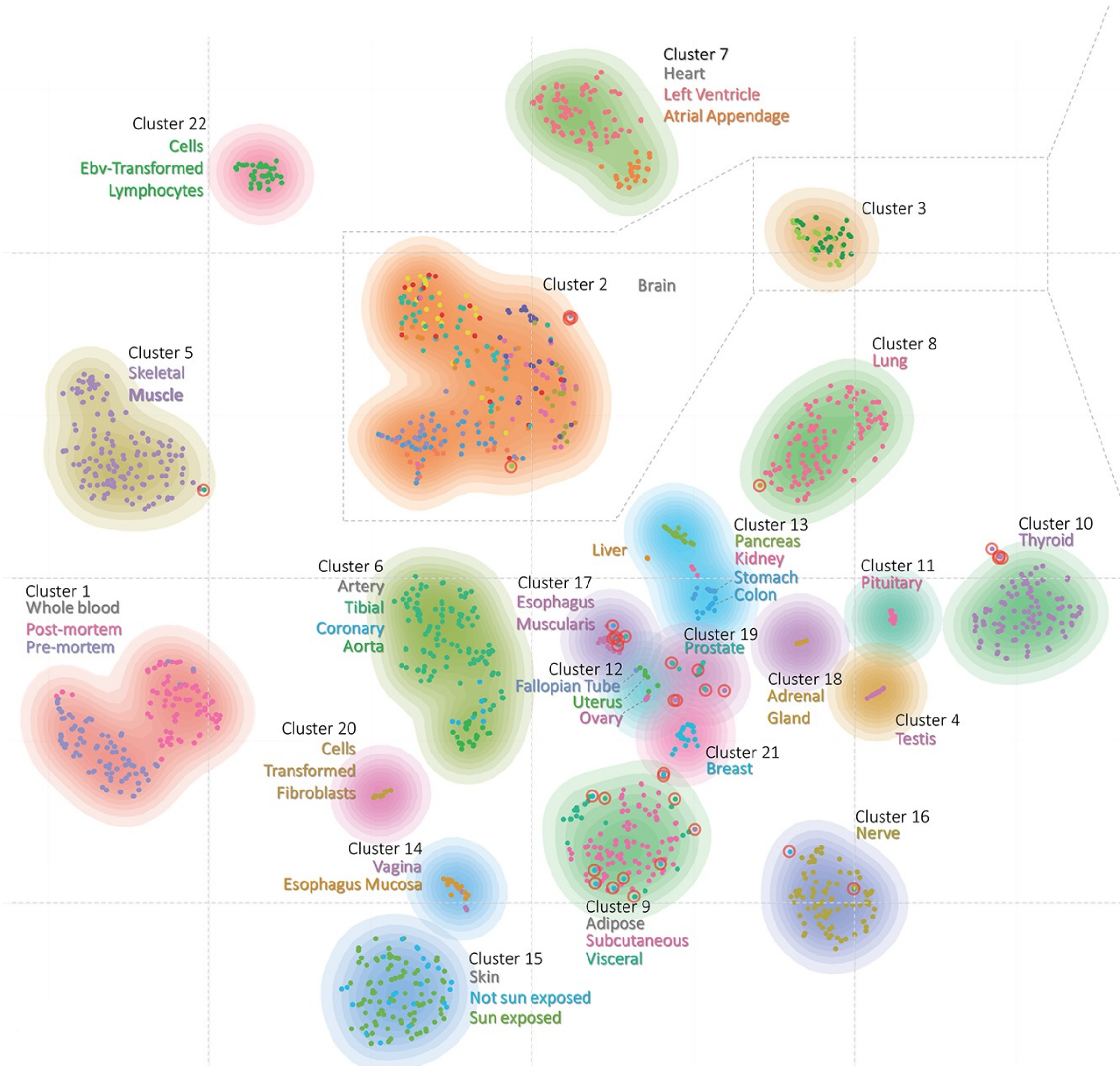


2D point distance reflects
*n*D row distance

Why is this useful?

- no matter how large *n* is, we obtain a 2D scatterplot-like image (so it's visually scalable)
- point-to-point distance (in 2D) shows similarity of observations (in *n*D)
- coloring points by one attribute can show additional information on the observations

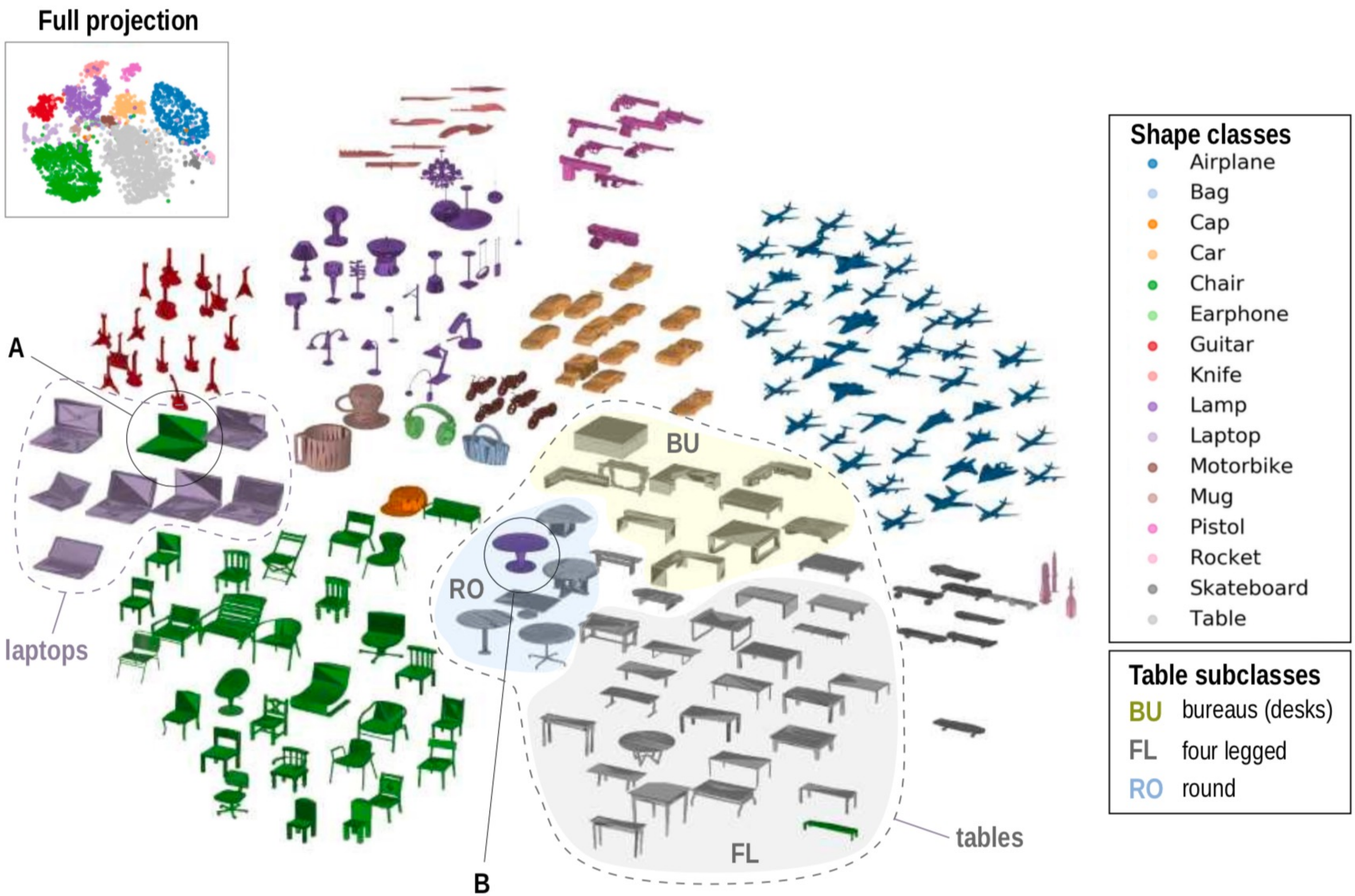
Projection example: Finding similar tissues



1 point = 1 tissue sample
data = RNA profiles
close points = similar data
similar data = similar tissue

E. Taskesen, M. Reinder (2016) 2D Representation of Transcriptomes by t-SNE Exposes Relatedness between Human Tissues. PLoS ONE 11(2).

Projection example: Browsing a 3D database



DR creates clusters of similar shapes!

X. Chen *et al.* (2021) Scalable Visual Exploration of 3D Shape Databases via Feature Synthesis and Selection, CCIS Springer

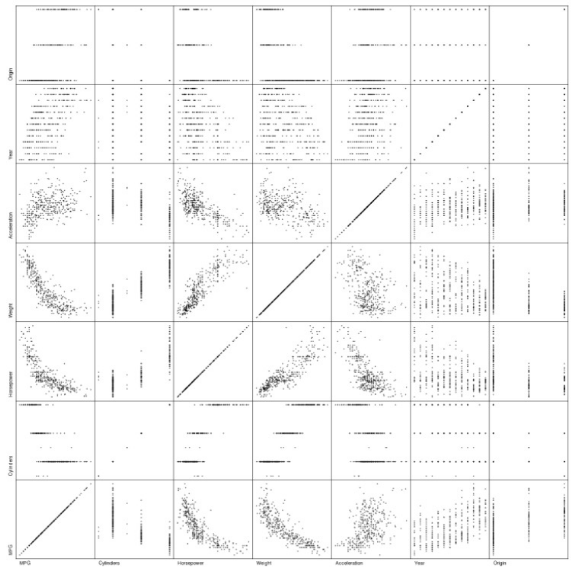
Projections *vs* other techniques

Projection

Pro's
- show similar observations
- no clutter
- scalable

Con's
- don't show why points are similar

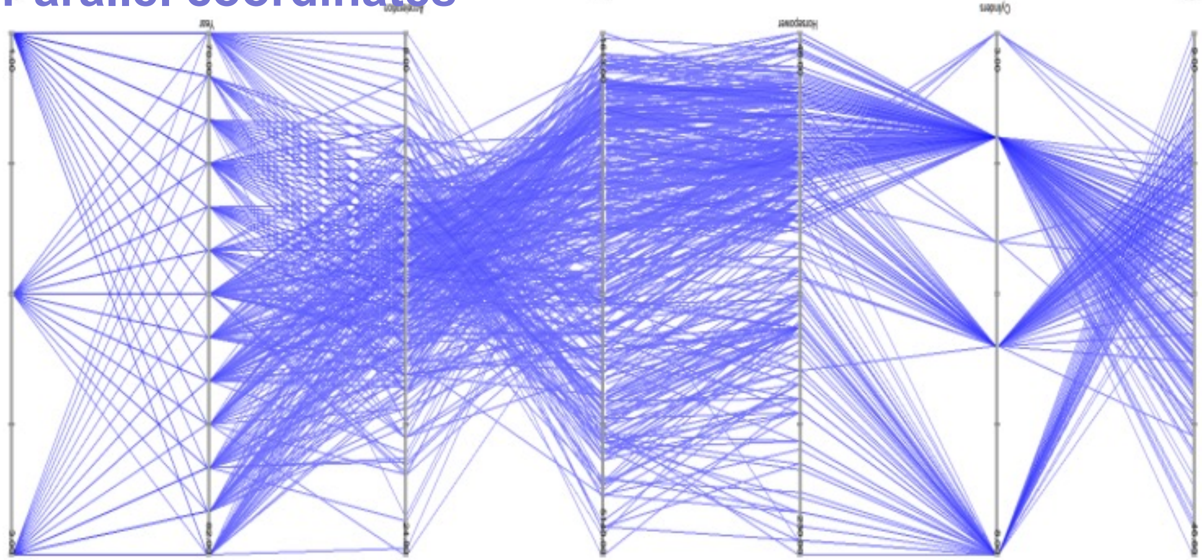
Scatterplot matrix



Pro's
- show variable correlations

Con's
- doesn't show similar points
- not scalable to many columns

Parallel coordinates



Pro's
- show variable correlations
- more scalable than SPLOMs

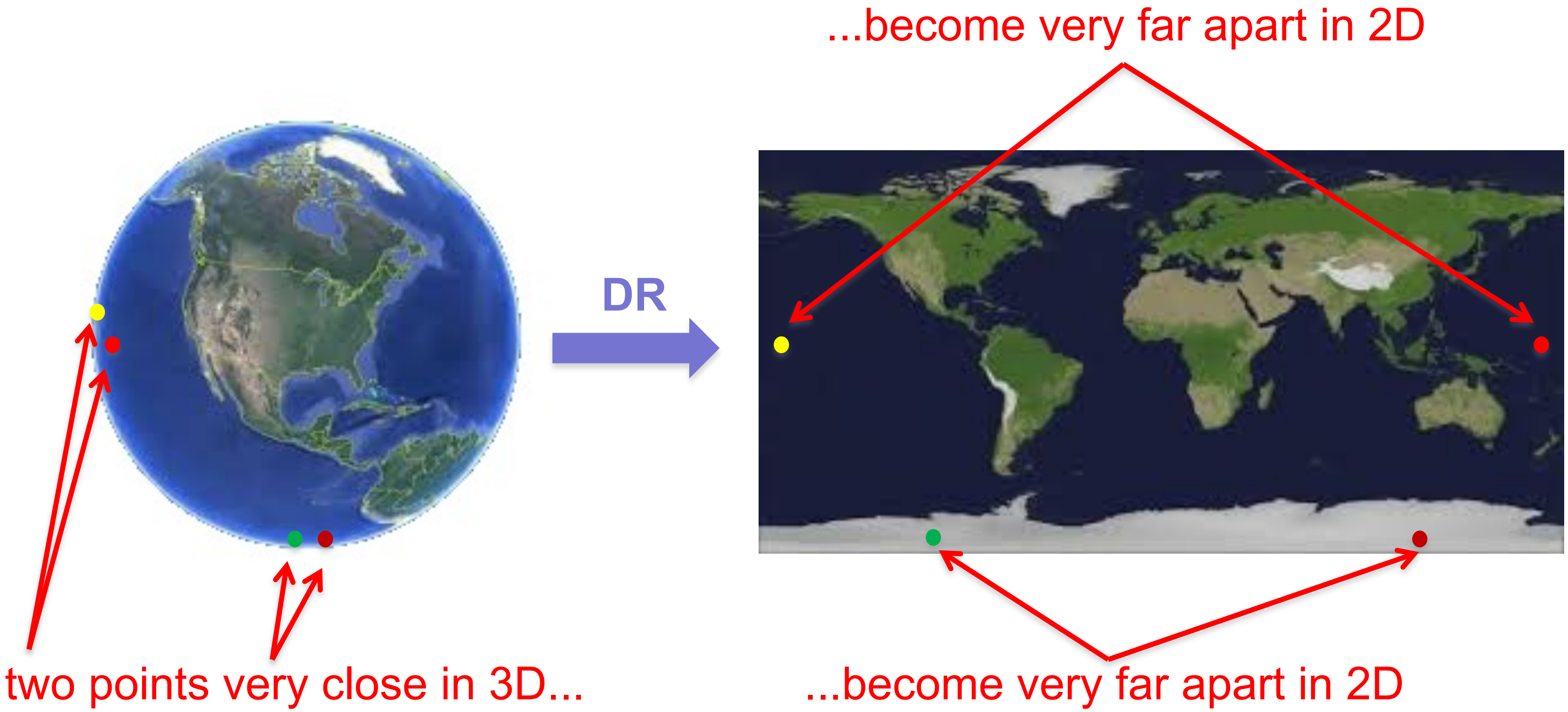
Con's
- generates clutter
- requires one to order axes

Dimensionality Reduction Methods

Overall considerations

In general, perfect distance-preserving of the data in DR is **not possible**!

Take the ‘simple’ problem of reducing $d=3$ to $m=2$ dimensions in cartography



Dimensionality Reduction Methods

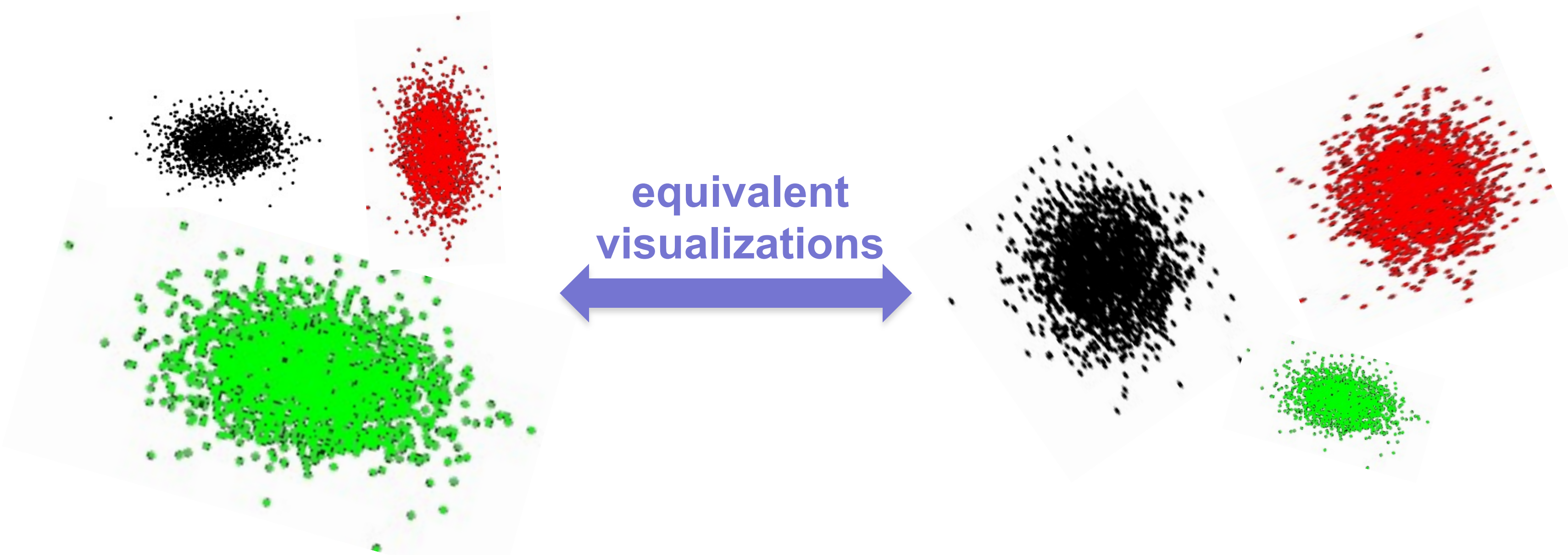
How to deal with distance preservation

1. **Approximately** preserve distance

$$d_{dD}(x, y) \approx d_{mD}(x, y)$$

…since our distances are anyway computed from heuristic features...

2. Preserve **k-nearest neighbors,** not distances



…since we care about **who**'s most similar, and not **how similar** precisely

DR Methods: Principal Component Analysis

Simplest solution for DR

- compute covariance matrix A of feature vectors $f_1 \ldots f_N$
- do PCA on A to find
  - its eigenvectors $e_1 \ldots e_d$
  - its eigenvalues $\lambda_1 \ldots \lambda_d$ (sorted so that $\lambda_1 > \lambda_2 > \ldots > \lambda_d$)
- select **m largest eigenvectors** $e_1 \ldots e_m$
- project $f_1 \ldots f_N$ onto the subspace spanned by $e_1 \ldots e_m$
- intuition: we preserve this way the **most variance** in $f_1 \ldots f_N$ that we can describe with only m dimensions

Example: MNIST dataset[1]

- 1 point = 1 image (28x28 pixels) of handwritten digit (0..9)
- 28x28 = 764 features (luminances of all pixels)
- points colored by class (0..9)

This projection is not very good!
Classes (colors) are mixed, so we cannot use 2D features to reason well about image similarities

[1] MNIST dataset: http://yann.lecun.com/exdb/mnist/

DR Methods: Multidimensional Scaling (MDS)

Addresses some of the PCA problems

- MDS aims to preserve the **pairwise distances** of points
- do this by minimizing the so-called stress

$$\sum_{i=1}^{N}\sum_{i=1}^{N}\left(d_{ij}^{(d)} - d_{ij}^{(m)}\right)^2$$

where $d_{ij}^{(d)}$ = distance between samples i, j in d-D (data space)
$d_{ij}^{(m)}$ = distance between samples i, j in m-D (projection space)

- minimization: done by linear algebra or force-directed methods[1]

Intuition

- if the stress is zero, then all pairwise distances in m-D are identical to the corresponding pairwise distances in d-D
- unlike PCA, we don't care here about the actual **coordinates** in d-D or m-D, but only about the **distances** between points

[1]https://www.math.uwaterloo.ca/~aghodsib/courses/f06stat890/readings/tutorial_stat890.pdf

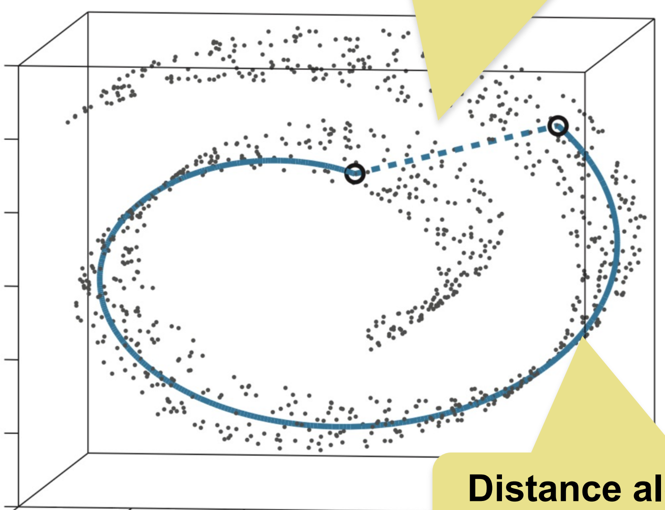
DR Methods: Isomap

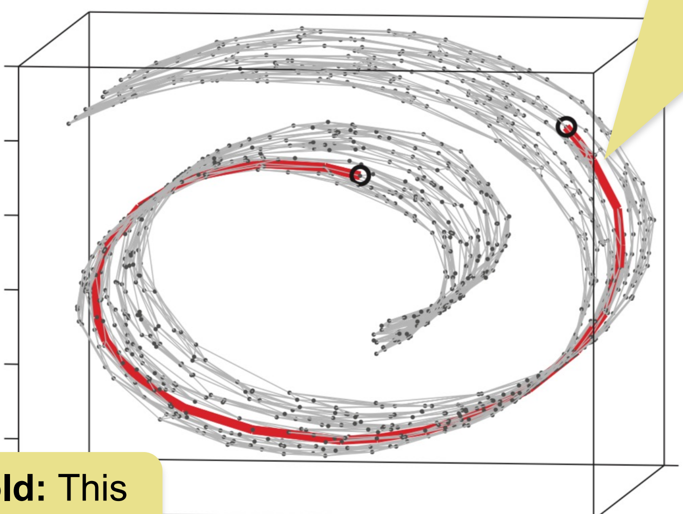
Addresses linearity problem of PCA, MDS

- PCA, MDS are linear: create a single (linear) transformation to map the **entire** d-D space to the lower-dimensional m-D space
- this is exact **only** when data in d-D is spread over a **hyperplane**!

Isomap

- assume the data is spread in d-D over a **manifold** (curved surface)
- this is more flexible than a hyperplane
- compute d-D distances **along this manifold**, and not in $\mathbf{R}^d$

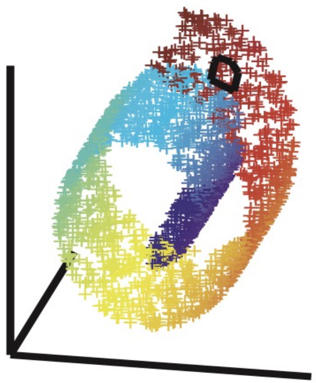




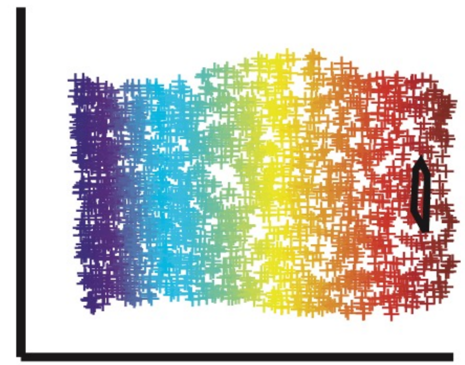
DR Methods: Isomap

Isomap: How to compute the high-dimensional manifold?

- manifold obtained by nearest-neighbor graph G of points in d-D
- distances: shortest-path distances in G (computed e.g. by Dijkstra's algorithm)
- use MDS with these distances



curved sheet dataset



Isomap: unfolds the manifold

Works very well!
Data is on a manifold

MNIST dataset

PCA: Poor separation

Isomap: A bit better separation

Works poorly
Data has a high intrinsic dimensionality

DR Methods: t-SNE

Drops Isomap's manifold assumption

- data in d-D can have higher intrinsic dimensionality than two
- d (data dimensionality) can be very large
- we now consider preserving **neighborhoods**, and not **distances**

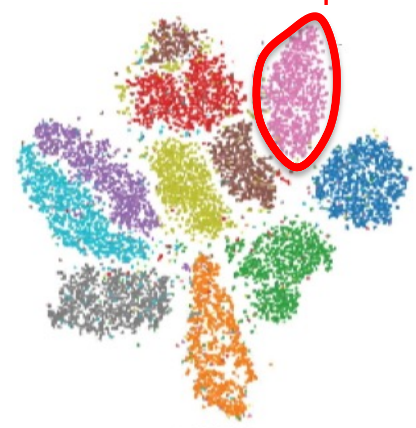
t-Stochastic Neighbor Embedding (t-SNE)

Let's show the **results** first for the MNIST dataset ☺

clusters (digit images) are almost perfectly separated

PCA: Poor separation...

Isomap: Bit better separation...



t-SNE: Excellent separation!

Good video introducing t-SNE: https://www.youtube.com/watch?v=RJVL80Gg3lA&list=UUtXKDgv1AVoG88PLl8nGXmw

DR Methods: t-SNE

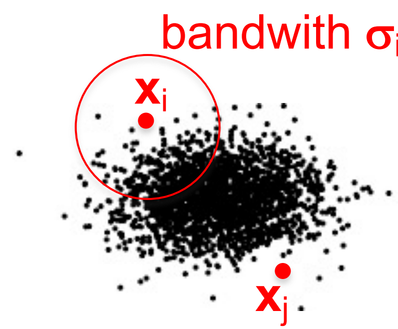
How t-SNE works

Notations

$\mathbf{x}_1 \ldots \mathbf{x}_N$	points (feature vectors) in high-dimensional feature space ($\mathbf{R}^d$)
$\mathbf{y}_1 \ldots \mathbf{y}_N$	points in latent (low-dimensional) feature space ($\mathbf{R}^m$)

1. Compute similarities in high-dim. space

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



normalizes density (considers similar #neighbors for all points)

similarity of $\mathbf{x}_j$ with $\mathbf{x}_i$ is defined as probability that $\mathbf{x}_i$ has $\mathbf{x}_j$ as close neighbor

DR Methods: t-SNE

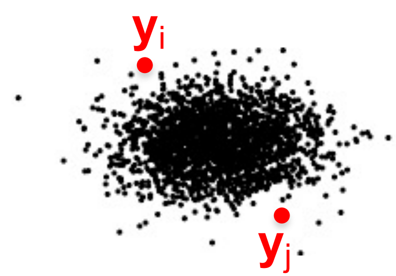
2. Symmetrize to obtain a true similarity metric

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

3. Model similarities in low-dim. space

$$q_{ij} = \frac{\boxed{\left(1 + \left\|\mathbf{y}_i - \mathbf{y}_j\right\|^2\right)^{-1}}}{\sum_k \sum_{l \neq k} \left(1 + \left\|\mathbf{y}_k - \mathbf{y}_l\right\|^2\right)^{-1}}$$

Student t-distribution (not Gaussian!)



Note q_{ij} has **very different formula** from p_{ij}

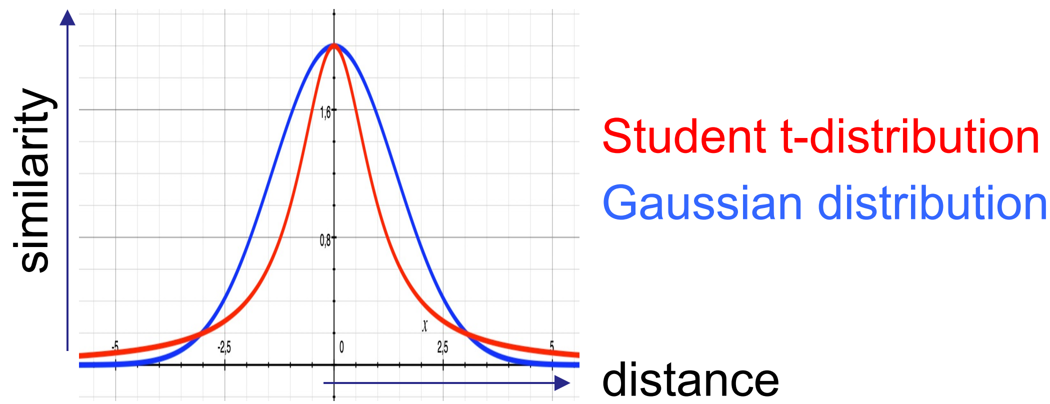
- we use a Student t-distribution, not a Gaussian one
- we don't have a bandwith σ

Why we model low-dim and high-dim similarities *differently*?

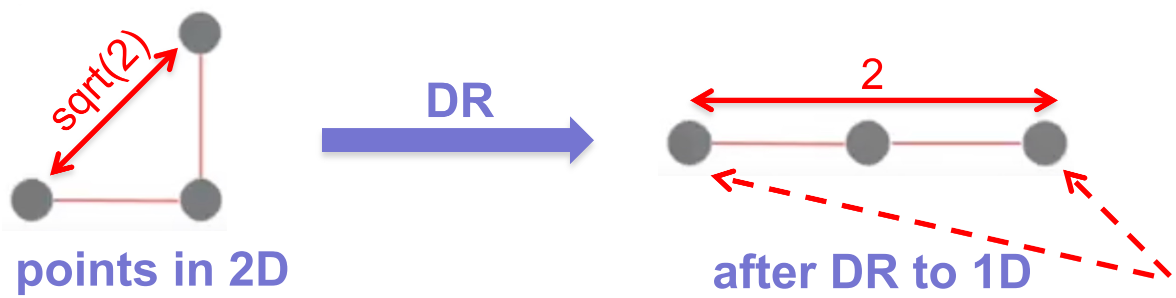
DR Methods: t-SNE

Why we model low-dim and high-dim similarities differently

We use a Student t-distribution in 2D, and a Gaussian one in high-dim



This allows modeling points that are far apart in high-dim **accurately** in 2D



if we want to preserve local structure, we *must* make the red distance correspond to a *higher* similarity

Using t-distribution makes points **far-apart** in low-dim look **closer**, so they better match their high-dim distances

DR Methods: t-SNE

4. Compute low-dim positions

Find $\mathbf{y}_j$ so that $q_{ij} \simeq q_{ij}$

For this, we first need to somehow **compare** p_{ij} and q_{ij}

Use **Kullback-Leibler divergence** to compare the **distributions** p_{ij} , q_{ij}

$$KL(P \| Q) = \sum_i \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

says that we want $p_{ij}=q_{ij}$

says that we care more about preserving *close* neighbors (high p_{ij} values)

Minimize KL by **gradient descent**

- initialize $\mathbf{y}_j$ randomly in 2D space
- while KL > ε

  compute $\nabla KL = 4 \sum_j \left(p_{ij} - q_{ij}\right)\left(\mathbf{y}_i - \mathbf{y}_j\right)\left(1 + \| \mathbf{y}_i - \mathbf{y}_j \|^2\right)^{-1}$

  move $\mathbf{y}_i = \mathbf{y}_i - \nabla KL$

DR Methods: t-SNE

Advantages

Can **keep local data structure** (clusters) better than most..all other DR methods

Works very well even with **very high-dimensional** spaces

Only needs **similarities** of points, not actual feature vectors

Does not care how data is **distributed** (on a plane, manifold, ...)

Disadvantages

Non-deterministic (starts each time with a random initialization)

Slow (seconds for hundreds of points, many minutes for 100K or more)

Tricky to parameterize (how to set σ? Not clear – trial and error...)

Let's see some simple t-SNE demos[1,2] and how to do parameter setting!

t-SNE source code: https://lvdmaaten.github.io/tsne/

[1] https://distill.pub/2016/misread-tsne/ [2] https://projector.tensorflow.org

Other DR Methods

UMAP

Approximation of t-SNE

Keeps all advantages, but about 10x faster and deterministic

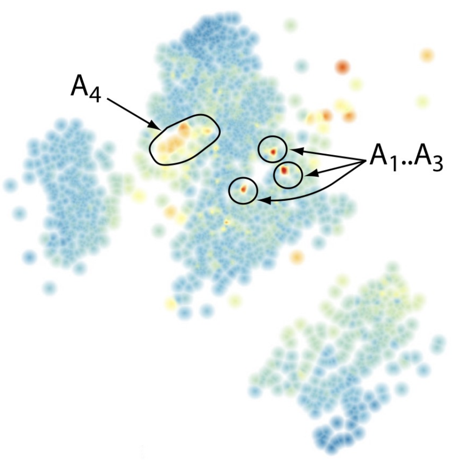
Even better same-item cluster separation

MNIST with t-SNE

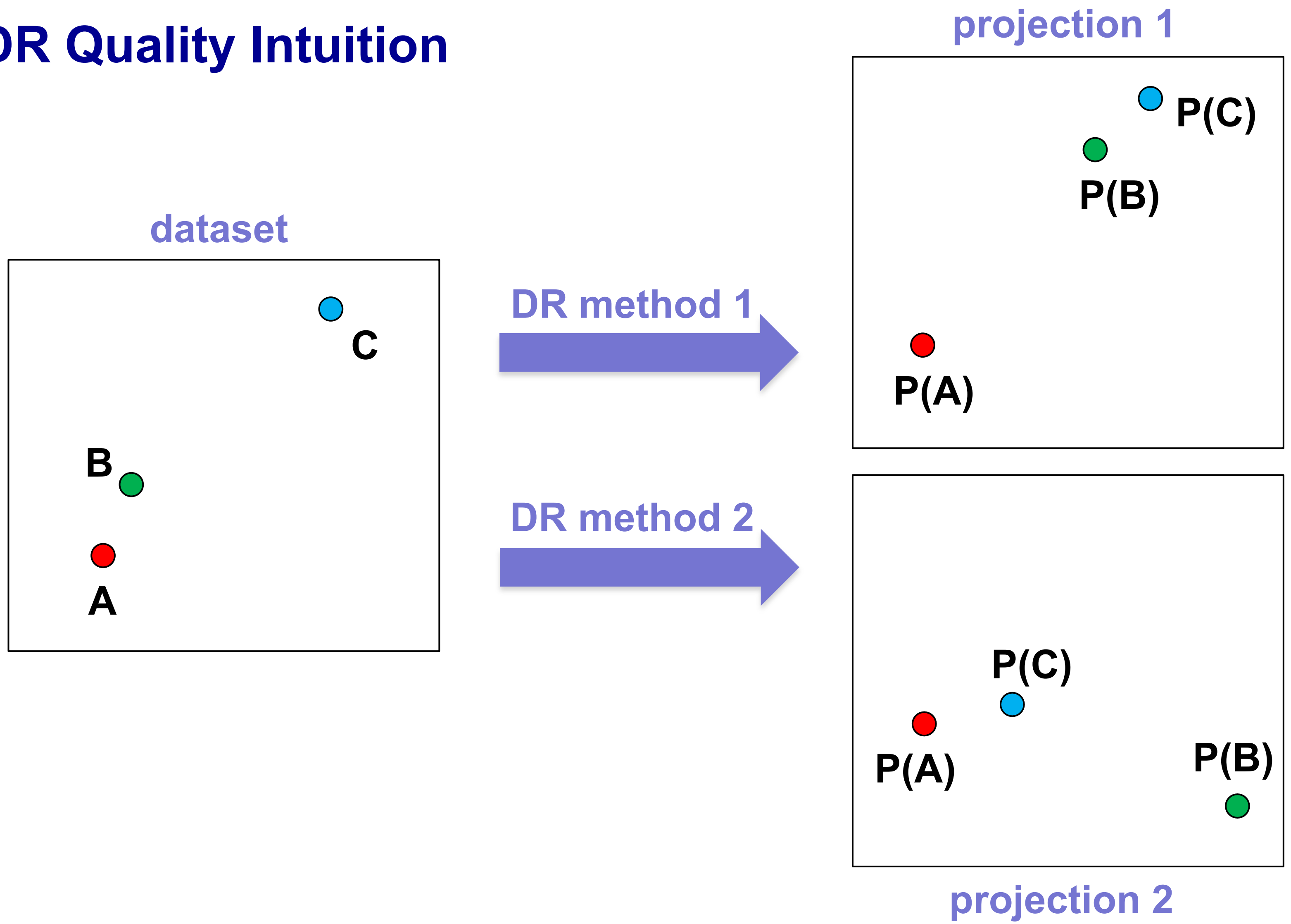
MNIST with UMAP

UMAP code and documentation: https://umap-learn.readthedocs.io/en/latest/

2. Measuring DR quality

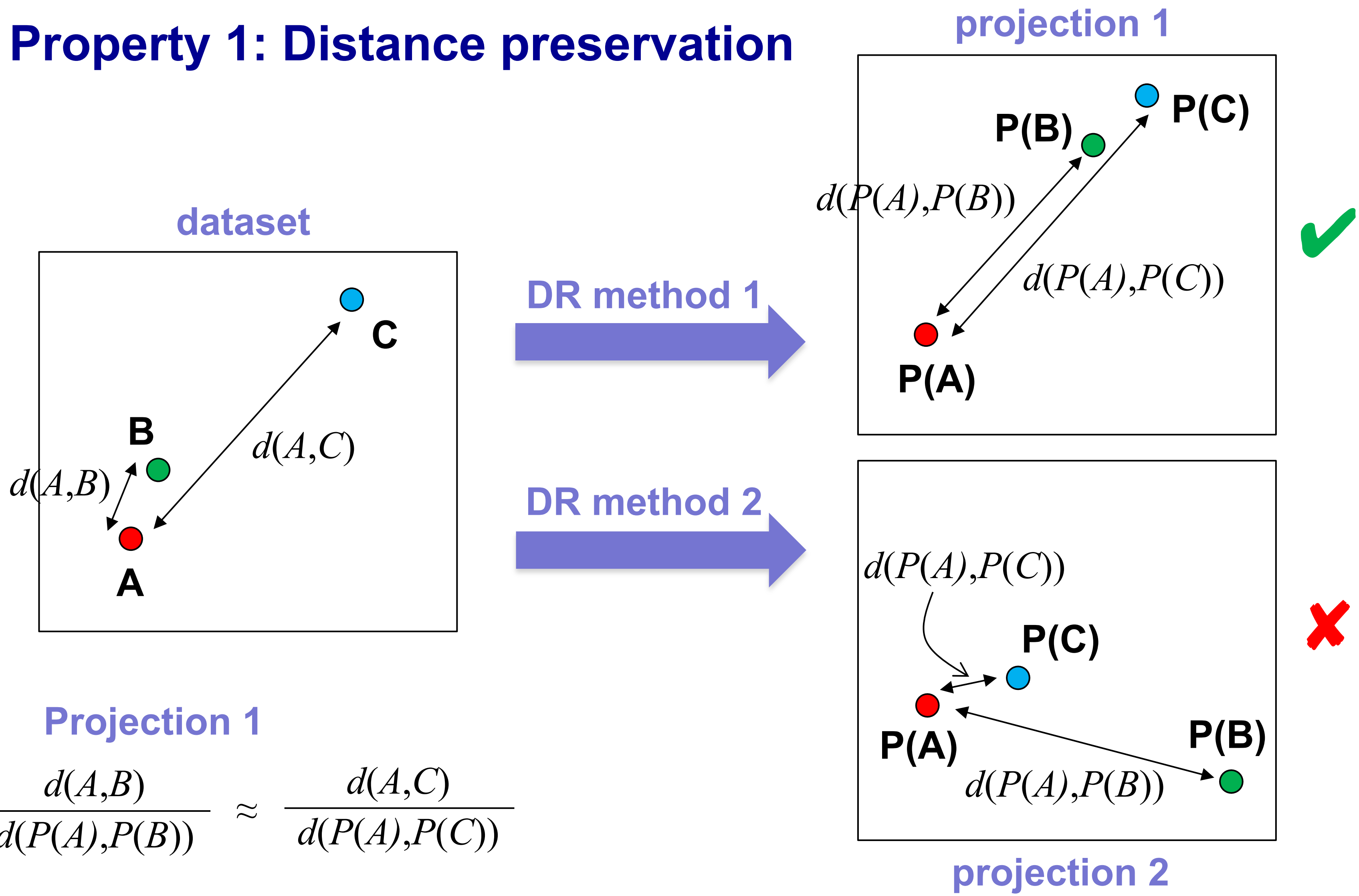


DR Quality Intuition



Which projection (1 or 2) keeps better the structure of D? Why?

Property 1: Distance preservation



Projection 1

$$\frac{d(A,B)}{d(P(A),P(B))} \approx \frac{d(A,C)}{d(P(A),P(C))}$$

Projection 2

$$\frac{d(A,B)}{d(P(A),P(B))} \neq \frac{d(A,C)}{d(P(A),P(C))}$$

Measuring distance preservation

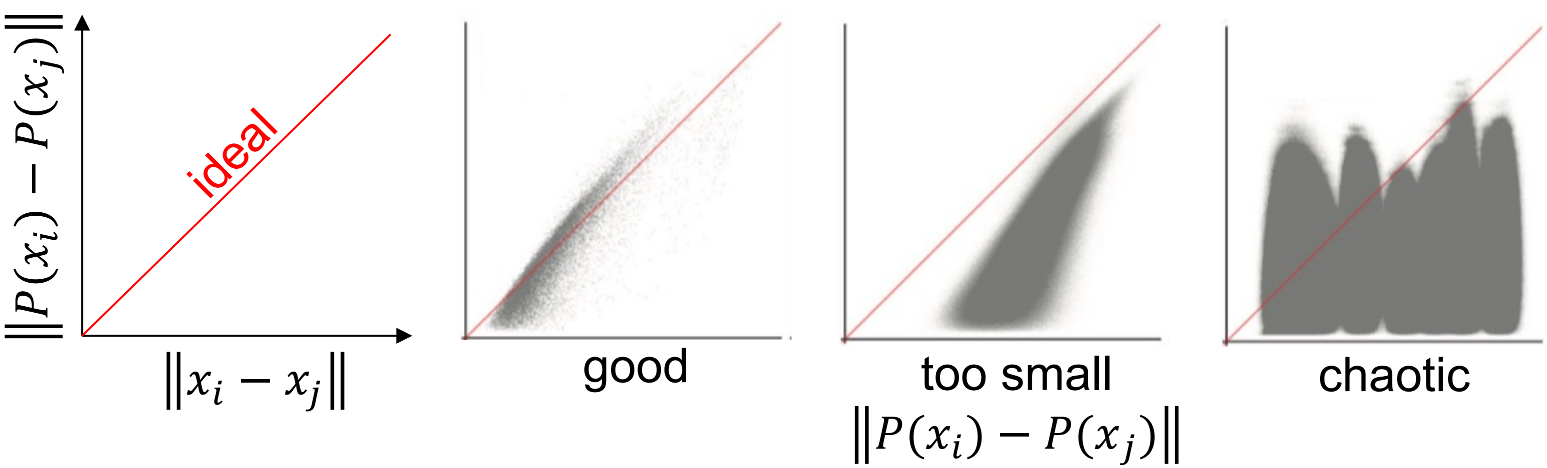
1. Normalized stress

distance in nD

distance in 2D

$$\sigma = \frac{\sum_{ij}\left(\|x_i - x_j\| - \|P(x_i) - P(x_j)\|\right)^2}{\sum_{ij}\|x_i - x_j\|^2}$$

2. Shepard diagram

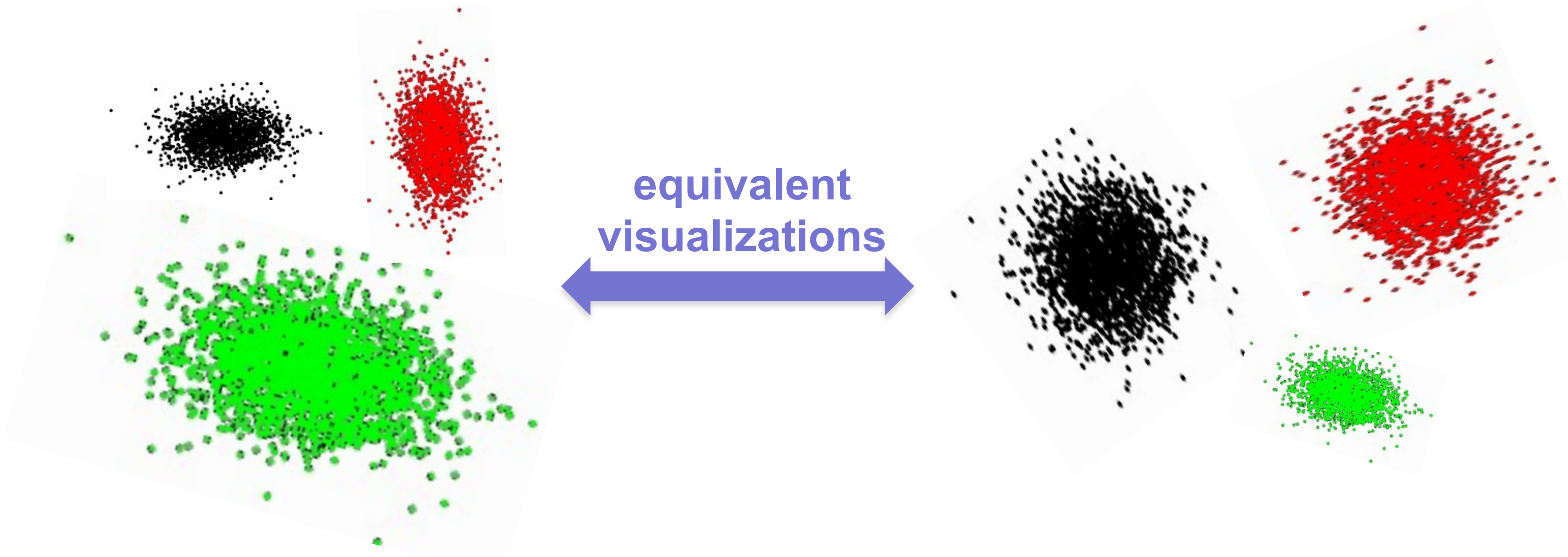


Quantify diagram goodness by its **Spearman rank correlation** ρ

Ideally ρ should be close to 1

Distance preservation limitations

Take these two projections



They tell the same story

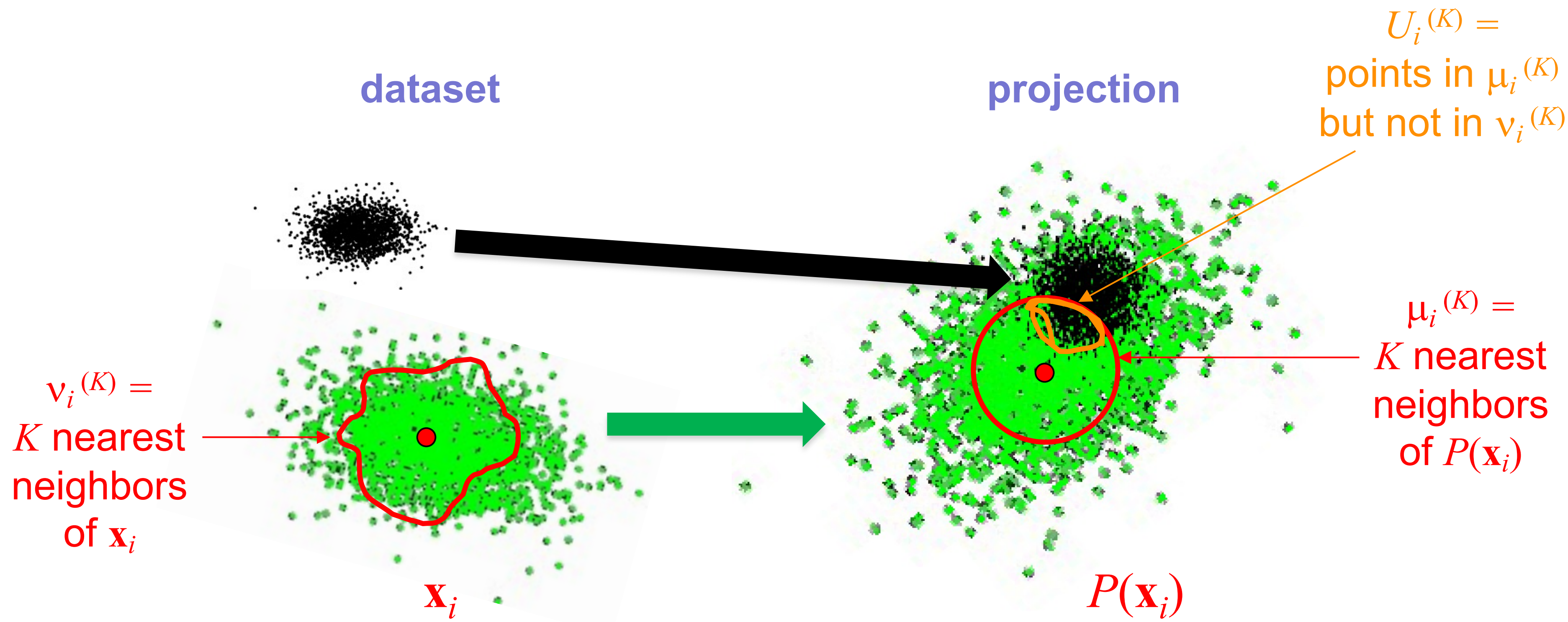
- we see three well-separated clusters of points

But inter-point distances are different

- so they will have very different σ and ρ metrics

We need to measure something else!

Idea: Measure neighborhood preservation



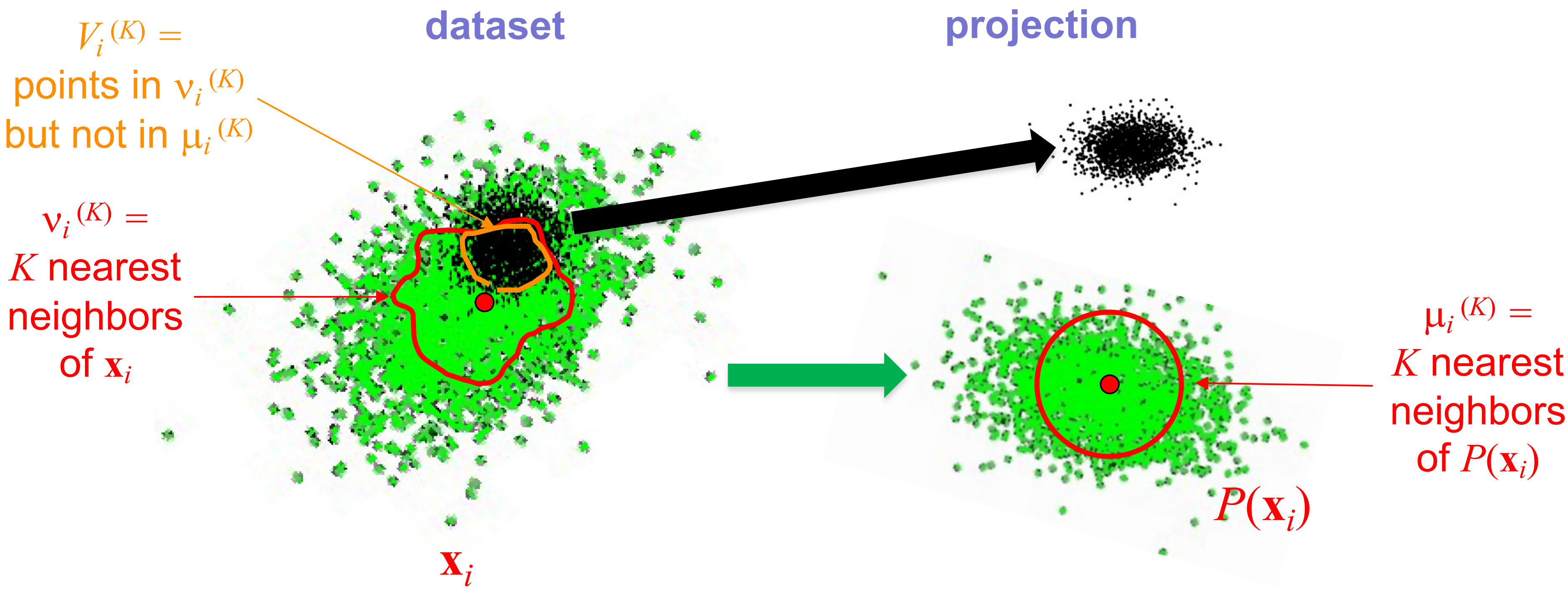
$U_i^{(K)}$ are **false neighbors** of $P(\mathbf{x}_i)$
Ideally, $U_i^{(K)}$ is **empty** (all neighbors of $P(\mathbf{x}_i)$ come from neighbors of $\mathbf{x}_i$)
We measure this by **trustworthiness**

$$T = 1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(K)}} (r(i,j) - K)$$

$r(i,j)$ = rank of j in sorted set $\mu_i^{(K)}$

https://mespadoto.github.io/proj-quant-eval/

Measure neighborhood preservation (cont.)



$V_i^{(K)}$ are **missing neighbors** of $P(\mathbf{x}_i)$
Ideally, $V_i^{(K)}$ is **empty** (all neighbors of $\mathbf{x}_i$ go into neighbors of $P(\mathbf{x}_i)$)
We measure this by **continuity**

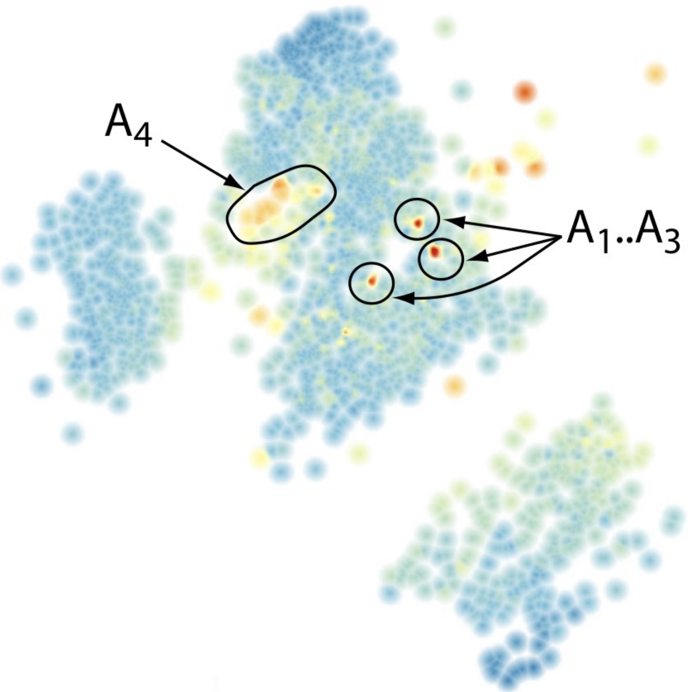
$$C = 1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(K)}} (\hat{r}(i,j) - K)$$

$\hat{r}(i,j)$ = rank of j
in sorted set $\mu_i^{(K)}$

https://mespadoto.github.io/proj-quant-eval/

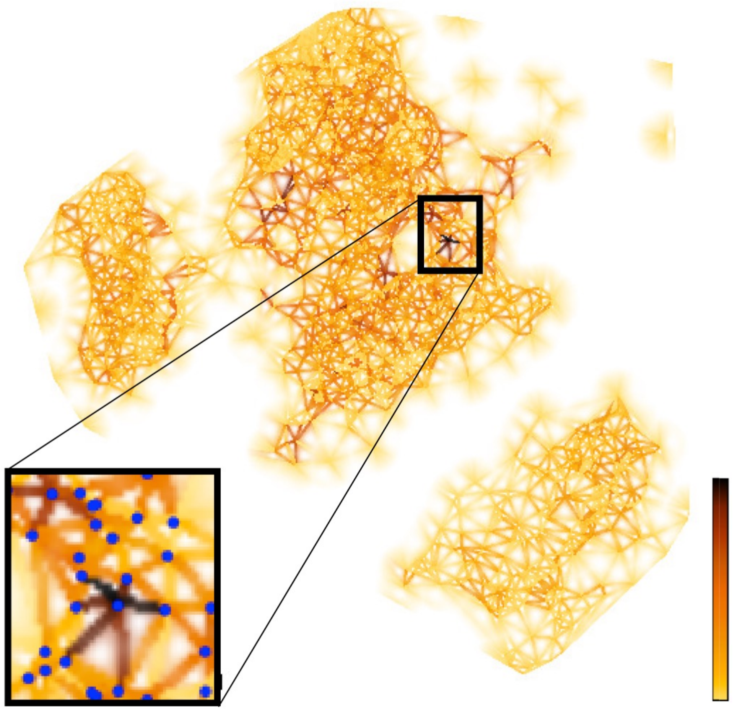
Visualizing projection errors

Large stress: points whose spacing (in 2D) does not reflect their spacing in *n*D



$$e_{ij} = \frac{d^m(\mathbf{q}_i, \mathbf{q}_j)}{\max_{i,j} d^m(\mathbf{q}_i, \mathbf{q}_j)} - \frac{d^n(\mathbf{p}_i, \mathbf{p}_j)}{\max_{i,j} d^n(\mathbf{p}_i, \mathbf{p}_j)}$$

False neighbors: points that are far (in *n*D) but placed close (in 2D)

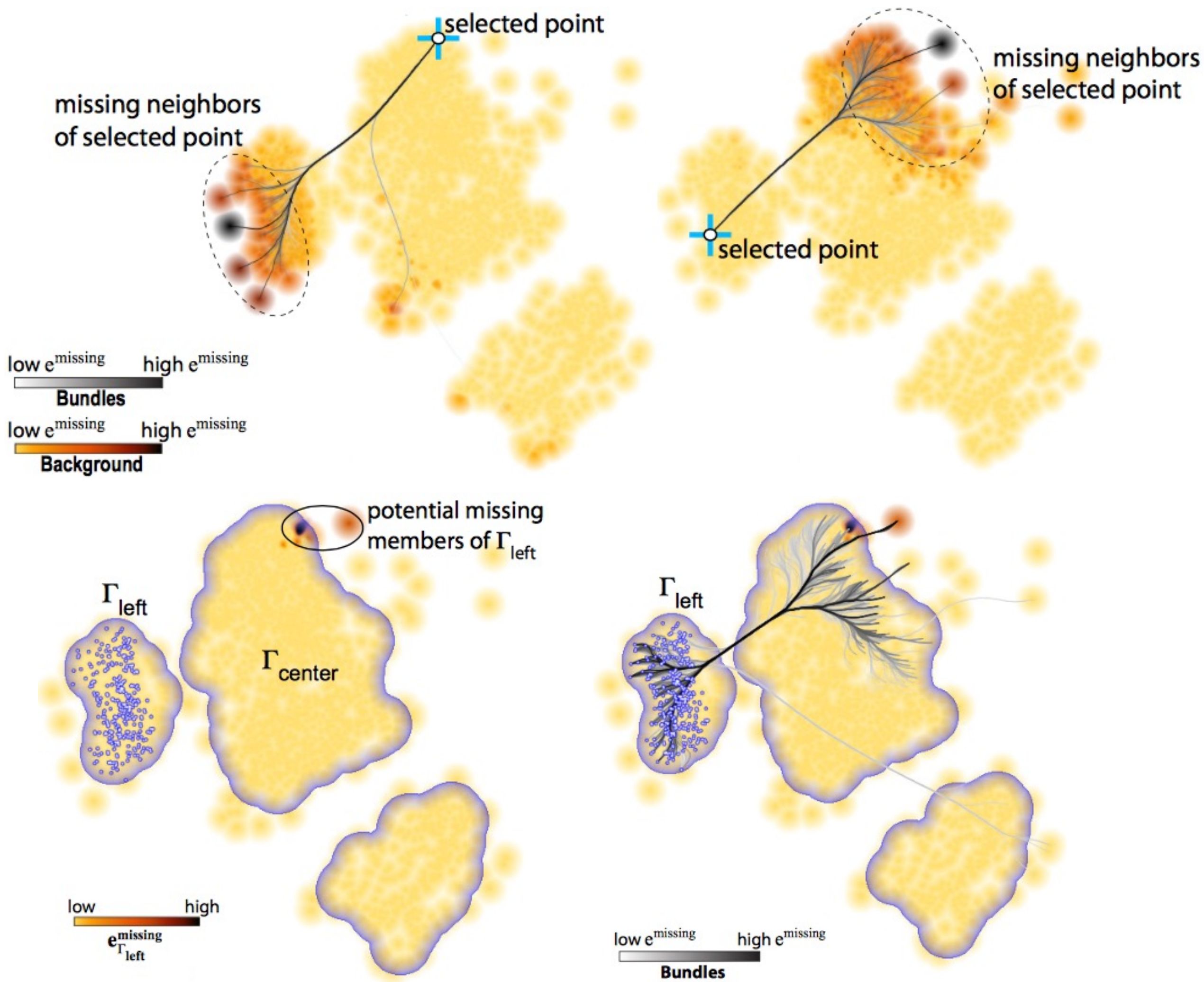


$$e_k^{false} = |\min(e_{ij}, 0)|$$

R. Martins *et al* (2014) Visual analysis of dimensionality reduction quality for parameterized projections. Computers & Graphics 41, 26-42

Visualizing projection errors

Missing neighbors: points that are close (in *n*D) but placed far apart (in 2D)

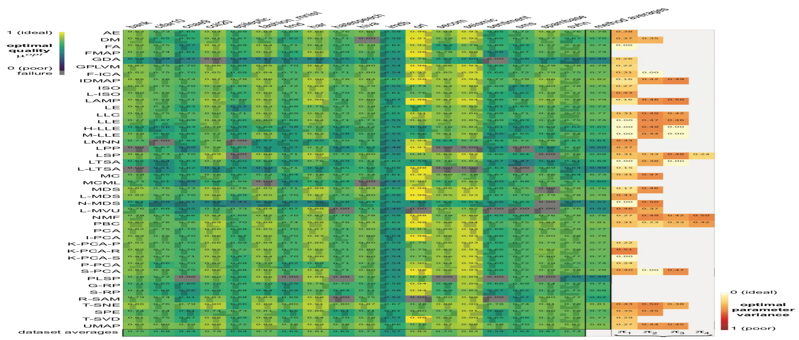


…for a single point

…for a point group

R. Martins *et al* (2014) Visual analysis of dimensionality reduction quality for parameterized projections. Computers & Graphics 41, 26-42

3. DR quality in practice



Which DR technique to *use* ?

surveys

techniques

Projection Acronym	Projection Full Name	Fodor *et al.* [18]	Hoffman *et al.* [1]	Yin *et al.* [19]	Maaten *et al.* [13]	Bunte *et al.* [15]	Engel *et al.* [27]	Sorzano *et al.* [12]	Cunningham *et al.* [23]	Gisbrecht *et al.* [21]	Liu *et al.* [2]	Xie *et al.* [24]	Nonato *et al.* [10]	Ours
AE	Autoencoder				•									•
CCA	CCA (Canonical Correlations Analysis)								•					
CHL	Chalmers												•	
CLM	ClassiMap												•	
CuCA	CCA (Curvilinear Component Analysis)												•	
DM	Diffusion Maps				•									•
DML	Distance Metric Learning								•					
EM	Elastic Maps							•						
FA	Factor Analysis	•						•	•					•
FD	Force-Directed												•	
FMAP	FastMap												•	•
FS	Feature Selection											•		
GDA	Generalized Discriminant Analysis													•
GPLVM	Gaussian Process Latent Variable Model													•
GTM	Generative Topographic Mapping							•					•	
ICA	Independent Component Analysis	•						•	•					
F-ICA	FastICA													•
NL-ICA	Nonlinear ICA	•												
IDMAP	IDMAP													•
ISO	Isomap		•	•	•	•	•			•			•	•
L-ISO	Landmark Isomap													•
KECA	Kernel Entropy Component Analysis							•						
KLP	Kelp												•	
LAMP	LAMP												•	•
LDA	Linear Discriminant Analysis								•		•	•	•	
LE	Laplacian Eigenmaps				•	•				•	•		•	•
LLC	Locally Linear Coordination				•									•
LLE	Locally Linear Embedding		•	•	•	•	•			•	•		•	•
H-LLE	Hessian LLE				•									•
M-LLE	Modified LLE													•
LMNN	Large-Margin Nearest Neighbor Metric													•
LoCH	Local Convex Hull												•	
LPP	Locality Preserving Projection								•					•
LR	Linear Regression								•					
LSP	Least Square Projection												•	•
LTSA	Local Tangent Space Alignment				•								•	•
L-LTSA	Linear Local Tangent Space Alignment													•
MAF	Maximum Autocorrelation Factors								•					
MC	Manifold Charting				•					•				•
MCA	Multiple Correspondence Analysis												•	
MCML	Maximally Collapsing Metric Learning													•
MDS	Metric Multidimensional Scaling	•	•	•	•	•	•	•	•		•		•	•
L-MDS	Landmark MDS													•
MG-MDS	Multi-Grid MDS						•							
N-MDS	Nonmetric MDS (Kruskal)		•				•						•	•
ML	Manifold Learning						•							
MVU	Maximum Variance Unfolding				•	•				•			•	
FMVU	Fast MVU													
L-MVU	Landmark MVU													•
NeRV	Neighborhood Retrieval Visualizer					•								
t-NeRV	t-NeRV					•								
NMF	Nonnegative Matrix Factorization							•	•					•
NLM	Nonlinear Mapping												•	
NN	Neural Networks	•												
PBC	Projection By Clustering													•
PC	Principal Curves	•						•						
PCA	Principal Component Analysis	•	•		•		•	•	•	•	•	•	•	•
I-PCA	Incremental PCA							•						•
K-PCA-P	Kernel PCA (Polynomial)													•
K-PCA-R	Kernel PCA (RBF)		•		•		•	•		•				•
K-PCA-S	Kernel PCA (Sigmoid)													•
L-PCA	Localized PCA							•						
NL-PCA	Nonlinear PCA	•		•				•						
P-PCA	Probabilistic PCA								•					•
R-PCA	Robust PCA							•						
S-PCA	Sparse PCA							•						•
PLMP	Part-Linear Multidimensional Projection												•	
PLP	Piecewise Laplacian-based Projection						•						•	
PLSP	Piecewise Least Square Projection													•
PM	Principal Manifolds			•										
PP	Projection Pursuit	•												
RBF-MP	RBF Multidimensional Projection												•	
RP	Random Projections	•										•		
G-RP	Gaussian Random Projection													•
S-RP	Sparse Random Projection													•
SAM	Sammon Mapping				•									
R-SAM	Rapid Sammon (Pekalska)												•	•
SDR	Sufficient Dimensionality Reduction								•					
SFA	Slow Feature Analysis								•					
SMA	Smacof												•	
SNE	Stochastic Neighborhood Embedding					•							•	
T-SNE	t-Dist. Stochastic Neighborhood Embedding					•				•	•			•
SOM	Self-Organizing Maps	•		•				•					•	
ViSOM	ViSOM (Visualization-induced SOM)			•										
SPE	Stochastic Proximity Embedding													•
G-SVD	Generalized SVD							•						
T-SVD	Truncated SVD													•
TF	Tensor Factorization							•						
UMAP	Uniform Manifold Approximation and Proj.													•
VQ	Vector Quantization	•						•						
Total		12	6	7	14	9	9	19	14	8	6	4	28	44

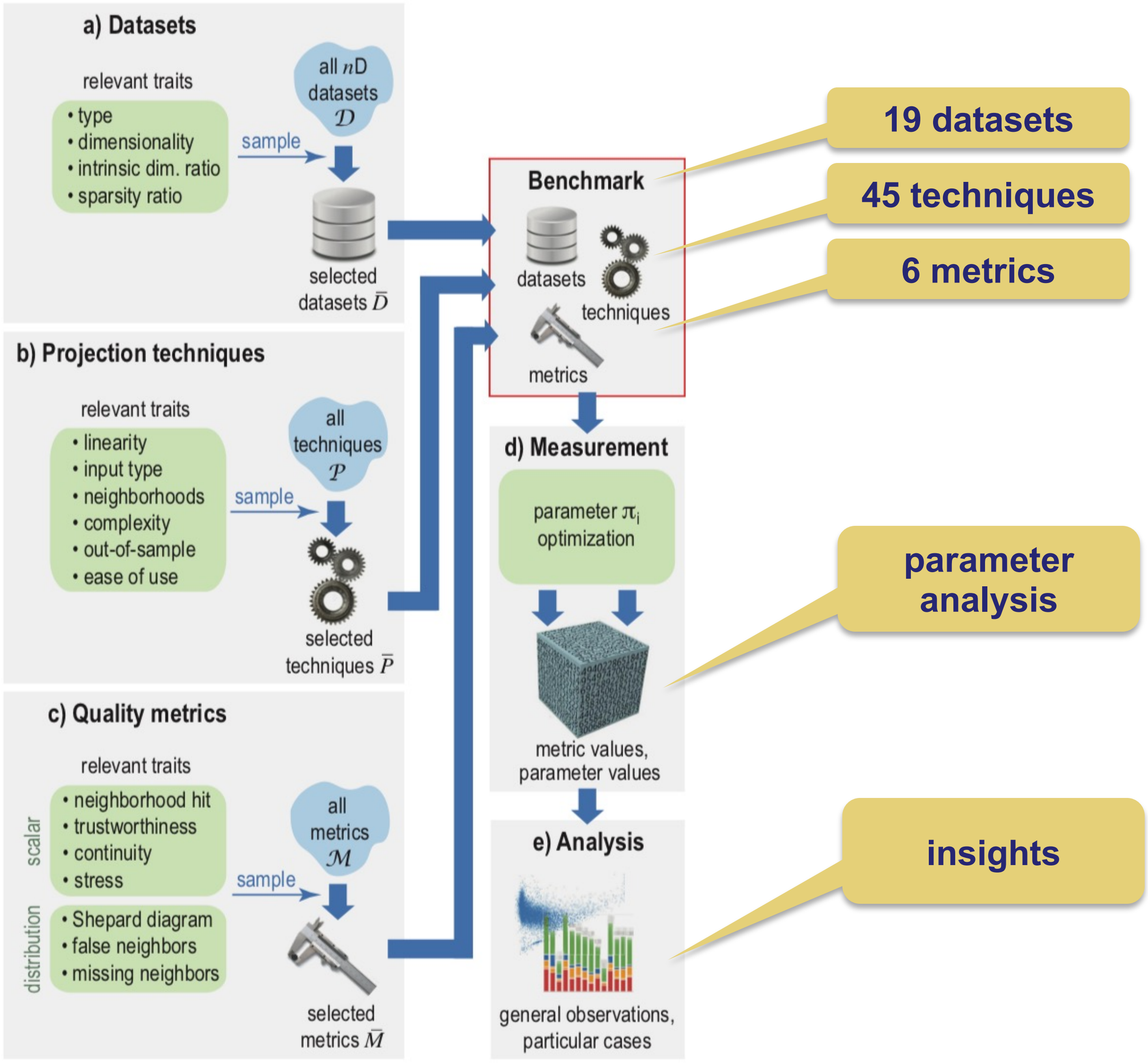
Big and unclear 'choice space'

- 50+ techniques
- 12 main surveys
- mainly theoretical discussion
- many parameters
- very limited practical comparison

Practitioner questions

- which projection is **best** for **my** context (requirements, data, …)?
- how to set its **parameters**?
- how to measure its **quality**?

Let’s measure projection errors big-scale!



M. Espadoto *et al* (2019) Towards a Quantitative Survey of Dimension Reduction Techniques (IEEE TVCG)

Datasets and Metrics

Datasets

Dataset	Type (τ_D)	Size (N)	Size class	Dimensionality (n)	Dimensionality class	Intrinsic dim. (ρ_n)	Intrinsic dim. class	Sparsity (γ_n)	Sparsity class
bank	*tables*	2059	*medium*	63	*low*	0.0317	*low*	0.6963	*medium*
cifar10	*images*	3250	*large*	1024	*high*	0.0706	*low*	0.0024	*dense*
cnae9	*text*	1080	*medium*	856	*high*	0.3201	*medium*	0.9922	*sparse*
coil20	*images*	1440	*medium*	400	*medium*	0.0105	*low*	0.3858	*medium*
epileptic	*tables*	5750	*large*	178	*medium*	0.2191	*medium*	0.0067	*dense*
fashion_mnist	*images*	3000	*medium*	784	*high*	0.2385	*medium*	0.5021	*medium*
fmd	*images*	997	*small*	1536	*high*	0.3073	*medium*	0.0095	*dense*
har	*tables*	735	*small*	561	*high*	0.1194	*medium*	0.0001	*dense*
hatespeech	*text*	3222	*large*	100	*medium*	0.6130	*high*	0.9993	*sparse*
hiva	*tables*	3076	*large*	1617	*high*	0.2498	*medium*	0.9091	*sparse*
imdb	*text*	3250	*large*	700	*high*	0.5790	*high*	0.9945	*sparse*
orl	*images*	400	*small*	396	*medium*	0.0006	*low*	0.9000	*sparse*
secom	*tables*	1567	*medium*	590	*high*	0.0102	*low*	0.2617	*medium*
seismic	*tables*	646	*small*	24	*low*	0.0417	*low*	0.5883	*medium*
sentiment	*text*	2748	*medium*	200	*medium*	0.8080	*high*	0.9936	*sparse*
sms	*text*	836	*small*	500	*medium*	0.7240	*high*	0.9947	*sparse*
spambase	*text*	4601	*large*	57	*low*	0.0351	*low*	0.7741	*medium*
svhn	*images*	733	*small*	1024	*high*	0.8734	*high*	0.0001	*dense*

Metrics

Metric	**Definition**	**Type**	**Range**
Trustworthiness (M_t)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in U_i^{(K)}} (r(i,j) - K)$	scalar	$[0, \mathbf{1}]$
Continuity (M_c)	$1 - \frac{2}{NK(2n-3K-1)} \sum_{i=1}^{N} \sum_{j \in V_i^{(K)}} (\hat{r}(i,j) - K)$	scalar	$[0, \mathbf{1}]$
Normalized stress (M_σ)	$\frac{\sum_{ij}(\Delta^n(\mathbf{x}_i,\mathbf{x}_j) - \Delta^q(P(\mathbf{x}_i),P(\mathbf{x}_j)))^2}{\sum_{ij}\Delta^n(\mathbf{x}_i,\mathbf{x}_j)^2}$	scalar	$[\mathbf{0}, 1]$
Neighborhood hit (M_{NH})	$\sum_{i=1}^{N} \frac{\lvert j \in N_i^{(K)} : l_j = l_i \rvert}{KN}$	scalar	$[0, \mathbf{1}]$
Shepard diagram (S)	Scatterplot $(\Vert\mathbf{x}_i - \mathbf{x}_j\Vert, \Vert P(\mathbf{x}_i) - P(\mathbf{x}_j)\Vert), 1 \le i \le N, i \neq j$	point-pair	-
Shepard goodness (M_S)	Spearman rank correlation of Shepard diagram	scalar	$[0, \mathbf{1}]$
Average local error ($M_a(i)$)	$\frac{1}{N-1} \sum_{j \neq i} \left\lvert \frac{\Delta^n(\mathbf{x}_i,\mathbf{x}_j)}{\max_{i,j}\Delta^n(\mathbf{x}_i,\mathbf{x}_j)} - \frac{\Delta^q(P(\mathbf{x}_i),P(\mathbf{x}_j))}{\max_{i,j}\Delta^q(P(\mathbf{x}_i),P(\mathbf{x}_j))} \right\rvert$	local (per-point)	$[\mathbf{0}, 1]$

aggregate into a single quality metric μ

Insights (1)

How good are projections, for which data?

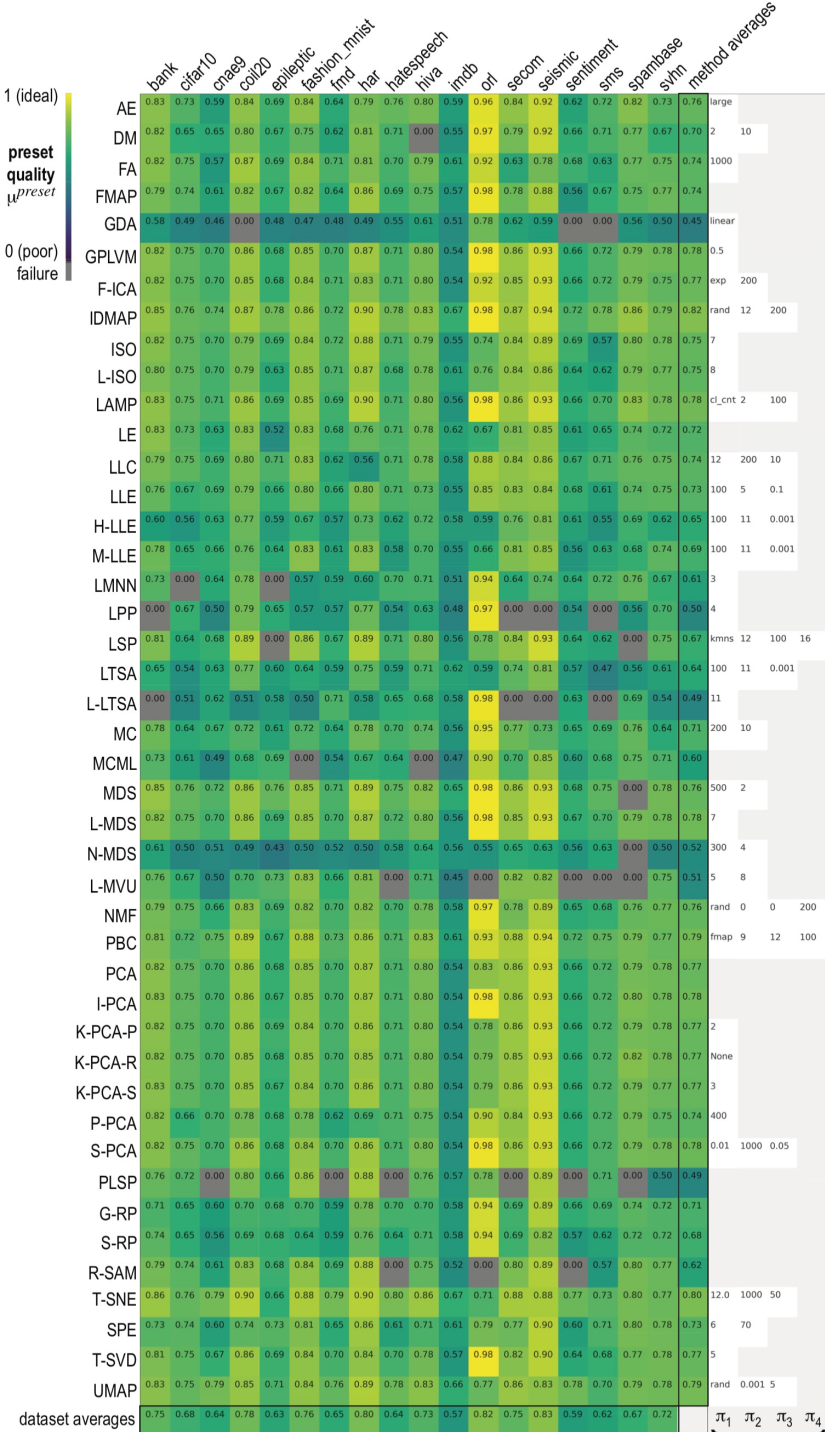
for each projection P_i
  for each dataset D_j
    compute *optimal* quality μ_{ij} (param. grid search)

How easy is to get optimal quality?

for each projection P_i
    compute *variance* of params π_i yielding optimal quality over all datasets D_j

What we see

- no projection **best** for all dataset types
- some are quite **poor** in general (N-MDS, GDA)
- dataset **type** strongly influences quality (*imdb*: hard; *orl*: easy)
- hard to **tune** parameters to get optimal quality (large variance of π_i)



Insights (2)

How good are parameter-preset projections?

for each projection P_i

π_i^{pre} = param values yielding most times optimal quality over all datasets D_j

for each projection P_i

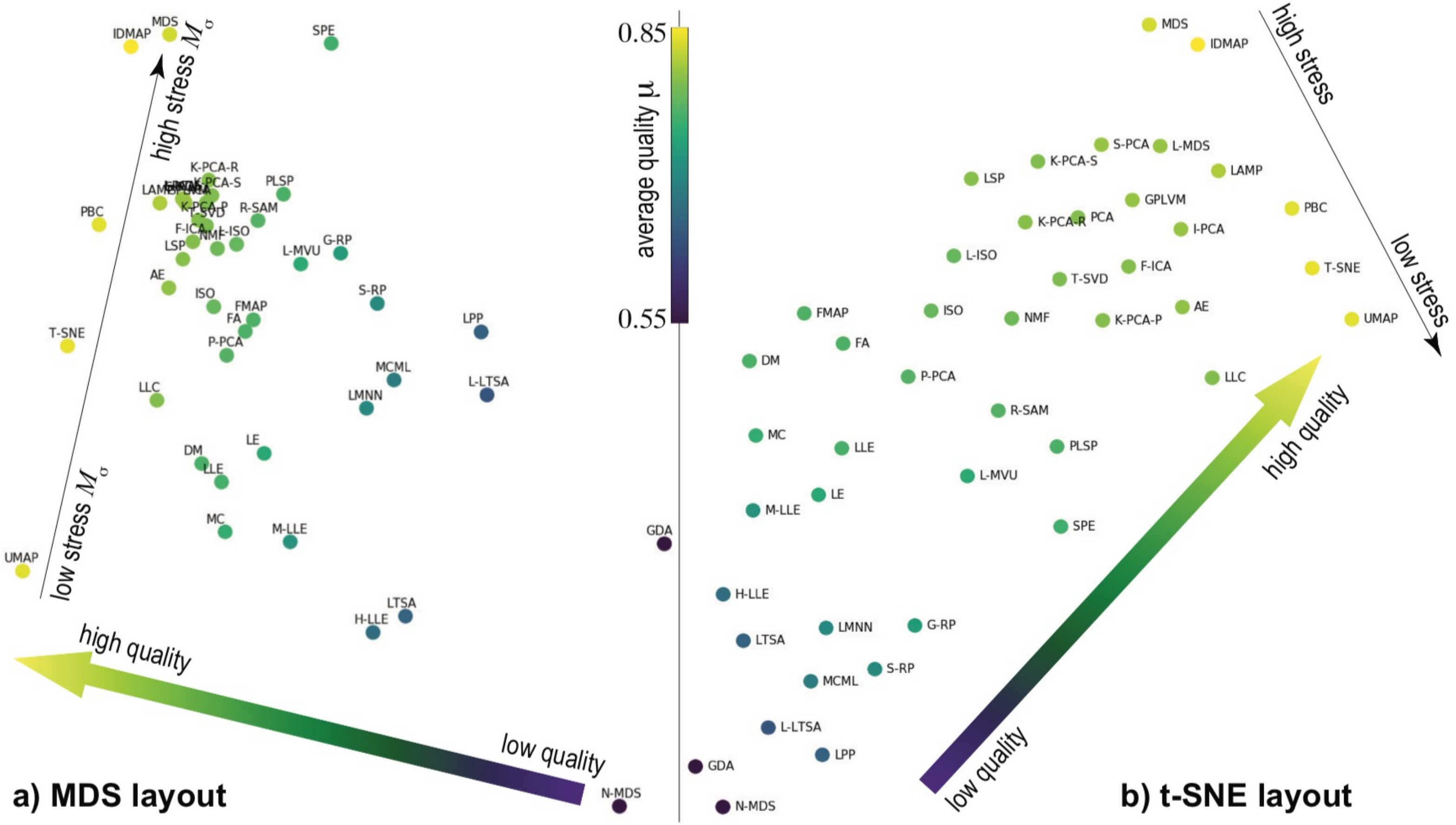
for each dataset D_j

compute quality μ_{ij} using π_i^{pre}

What we see

- very similar image to earlier one (optimal techniques stay good when using **presets**)
- again, quality strongly depends on dataset type
- t-SNE, UMAP, IDMAP, PBC score **best** on average

Insights (3): Which projections perform similarly?



‘Projection of projections’ map

- one point = one technique
- 5 attributes (trustworthiness, continuity, norm. stress, neighborhood hit, Shepard goodness; averaged over all tested datasets)
- we see a clear **quality trend**
- helps choosing projections that behave similarly to a user-chosen one

Benchmark

Towards A Quantitative Survey of Dimension Reduction Techniques

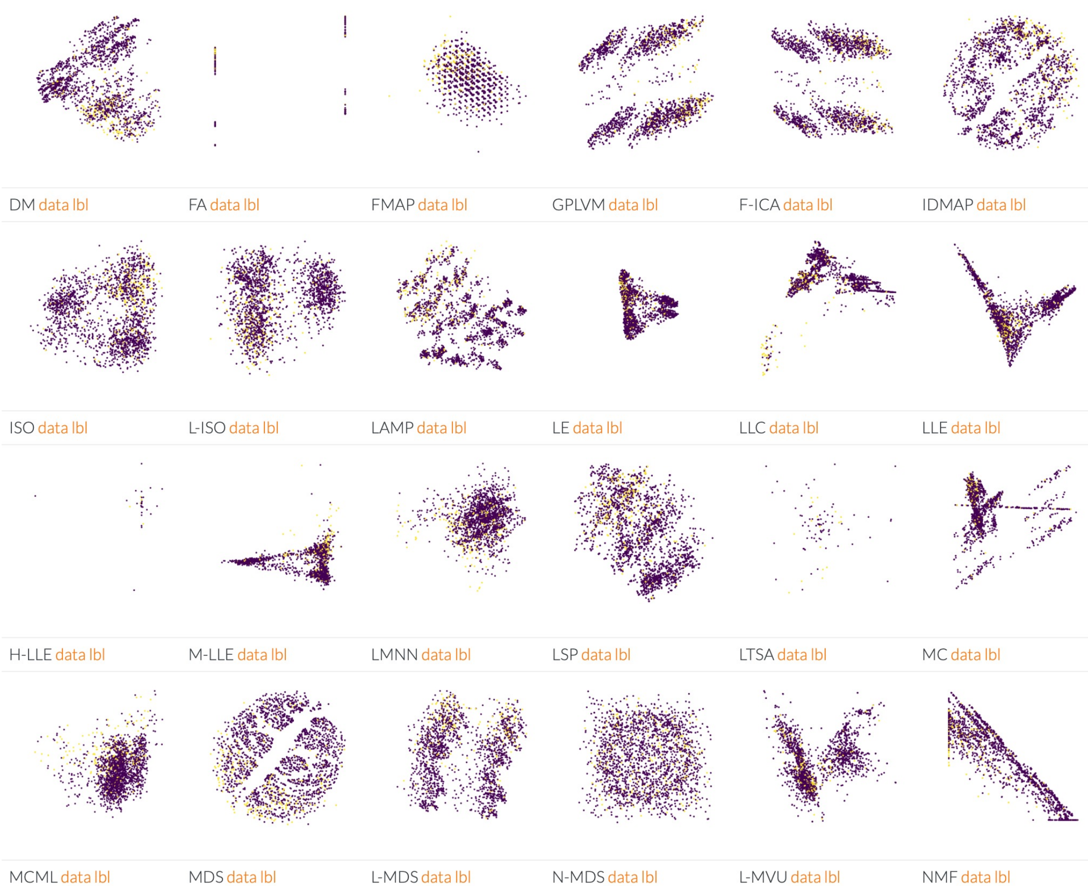
MATEUS ESPADOTO, RAFAEL M. MARTINS, ANDREAS KERREN, NINA S. T. HIRATA AND ALEXANDRU C. TELEA

DATASETS EXPERIMENT MEASUREMENTS PROJECTIONS

Projections for all datasets (best parameter set for each projection)

All projections, in csv format

bank dataset



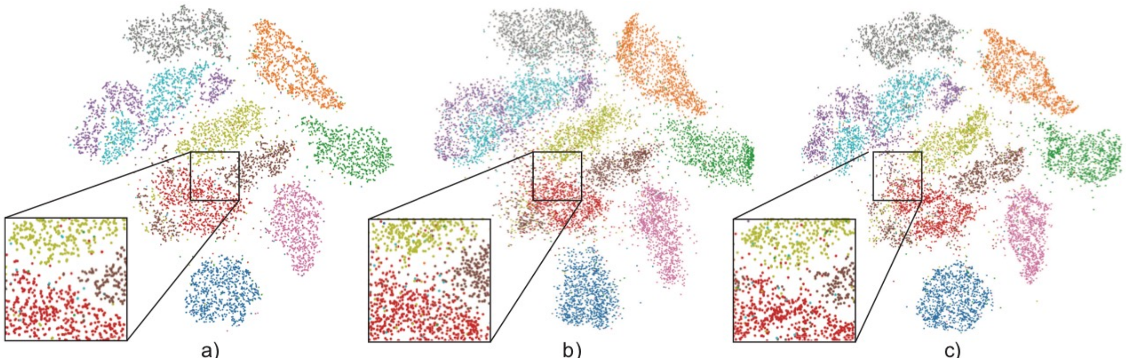
https://mespadoto.github.io/proj-quant-eval

All open source

- projection implementations
- datasets
- metric engines
- visualization engines
- optimization engines
- test harness
- all Python code

Please share, use, and extend!

4. Learning Projections



Insights from our survey

No ideal projection technique ☹

- UMAP: easy to use, quite fast, but quality not ideal
- t-SNE: quality is (very) high, but very slow, hard to tweak parameters, non-deterministic

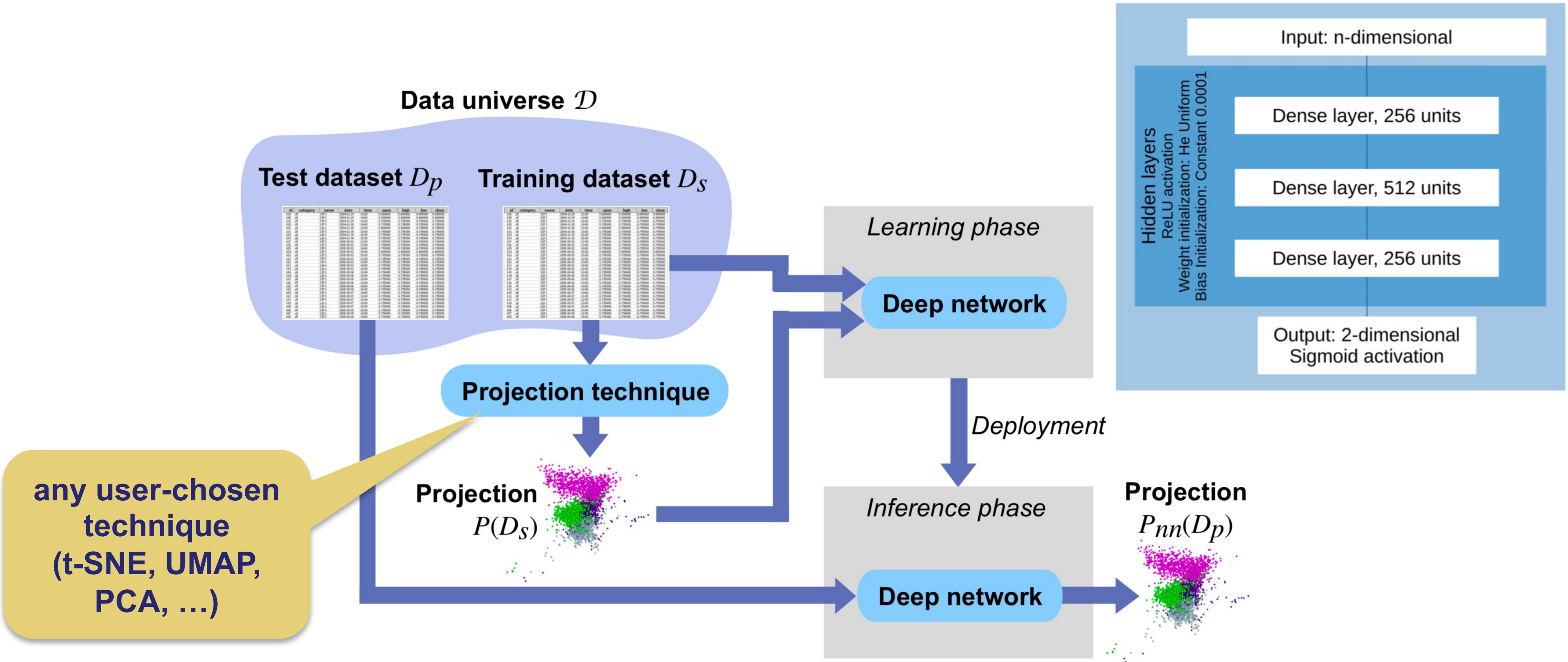
- quality depends a lot on type of data

What we want to have

- high-**quality** projection
- having `**style**’ of any projection deemed good by user
- working very **fast** (millions of samples, hundreds of dimensions: seconds)
- **easy** to use (no complex parameters, ideally none)
- **stable** (same input data: same output projection)
- **out-of-sample** (add some more data: project along existing data)

How to achieve this?

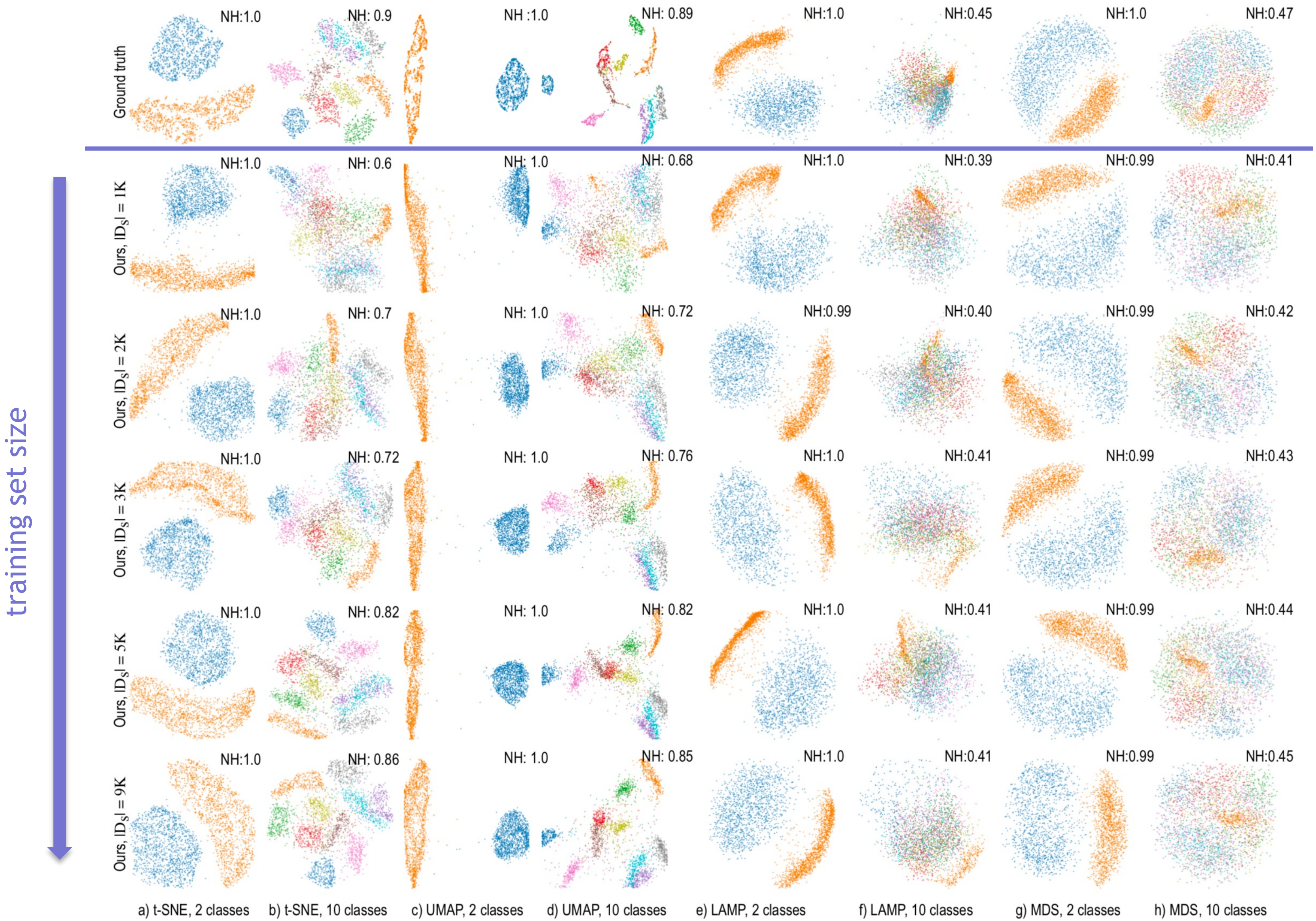
Idea: Learn the projection!



- take *any* dataset D_S and *any* projection technique of choice P
- project D_S with P, tweak P's parameters, obtain good scatterplot $P(D_S)$
- pass D_S and $P(D_S)$ to network, learn the mapping
- use trained network P_{nn} to project any other similar dataset D_P

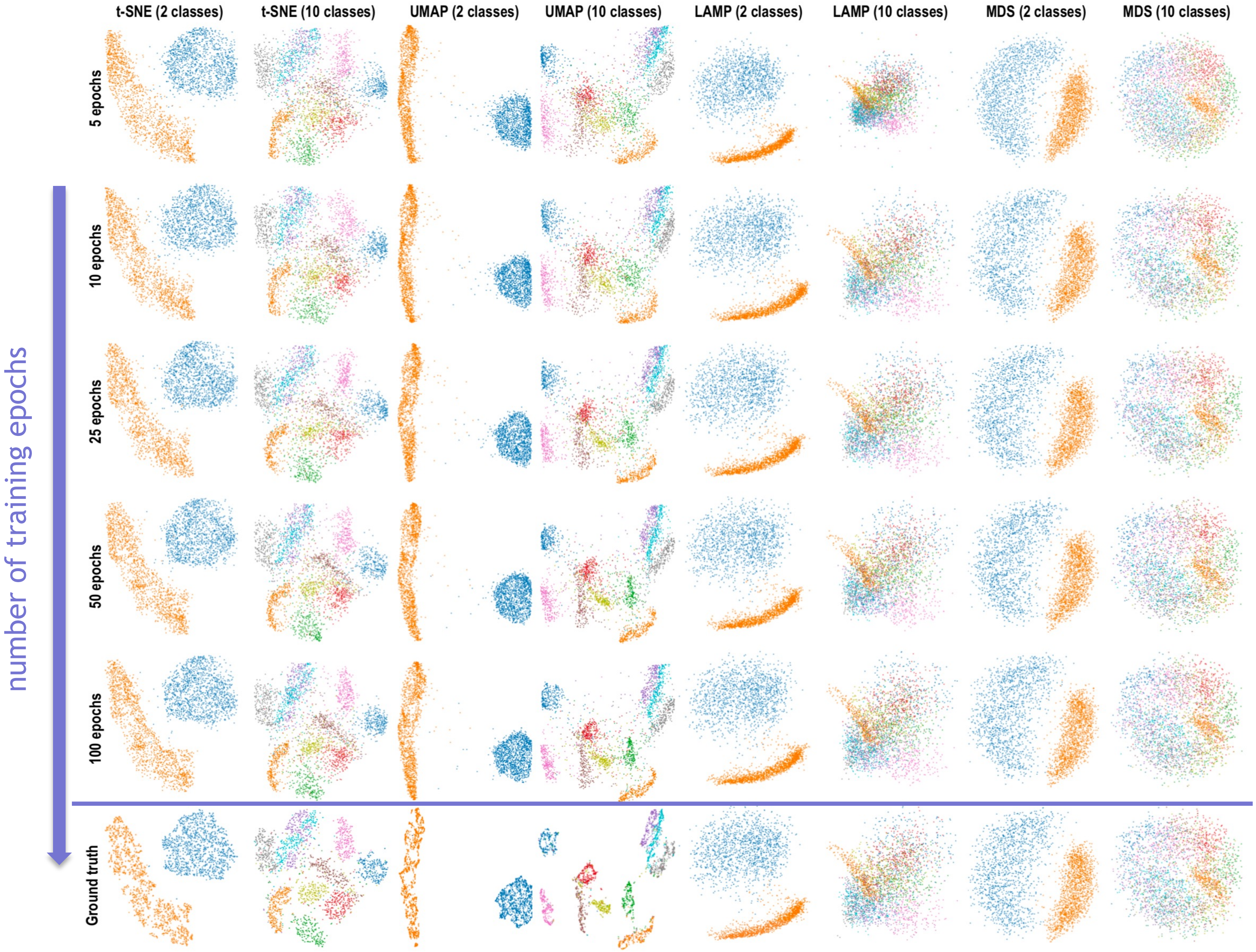
M. Espadoto *et al* (2020) Deep Learning Multidimensional Projections. *Information Visualization* 9(3), 247-269

Training-set size influence



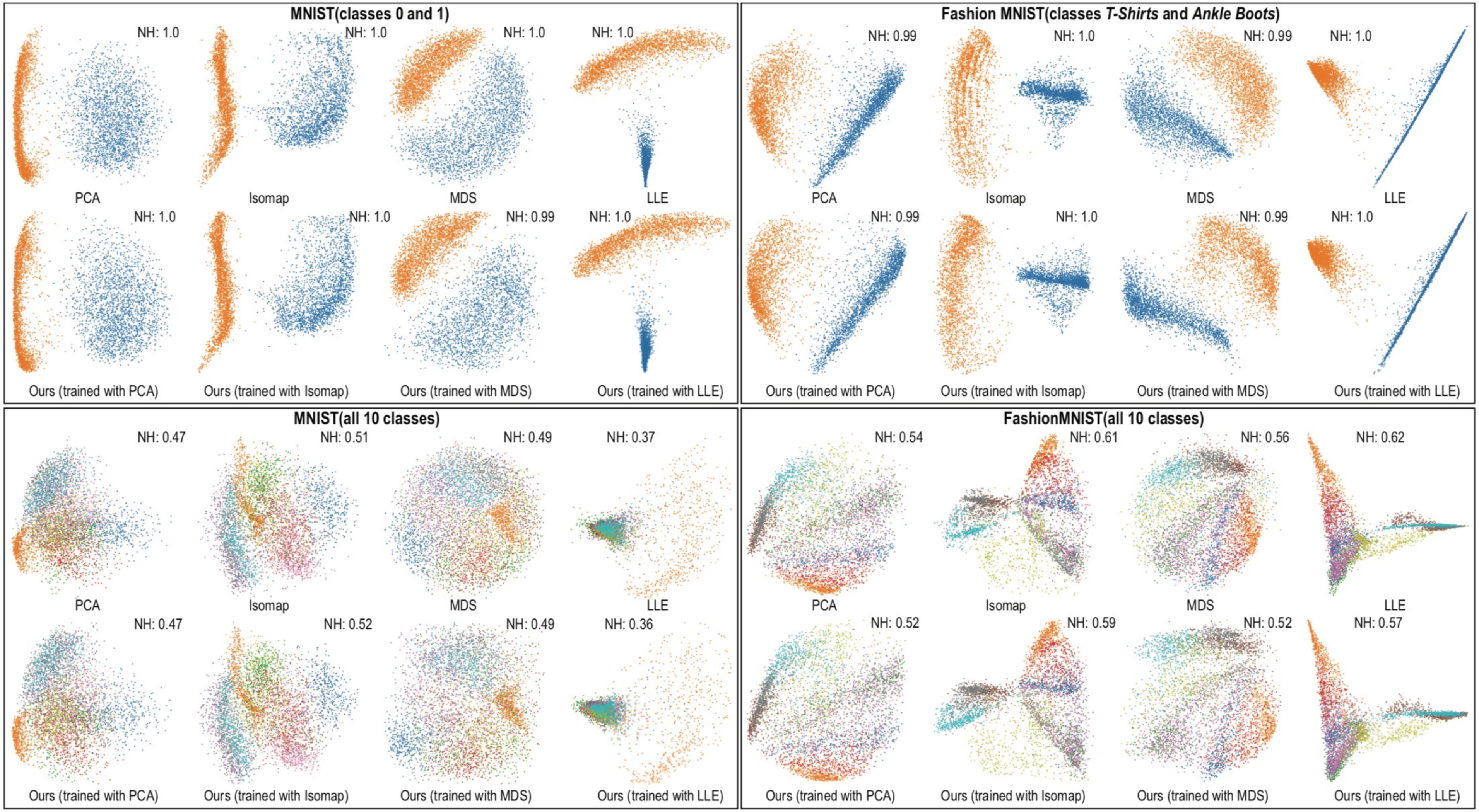
Quite good results with a **few thousand** training samples

Training effort influence



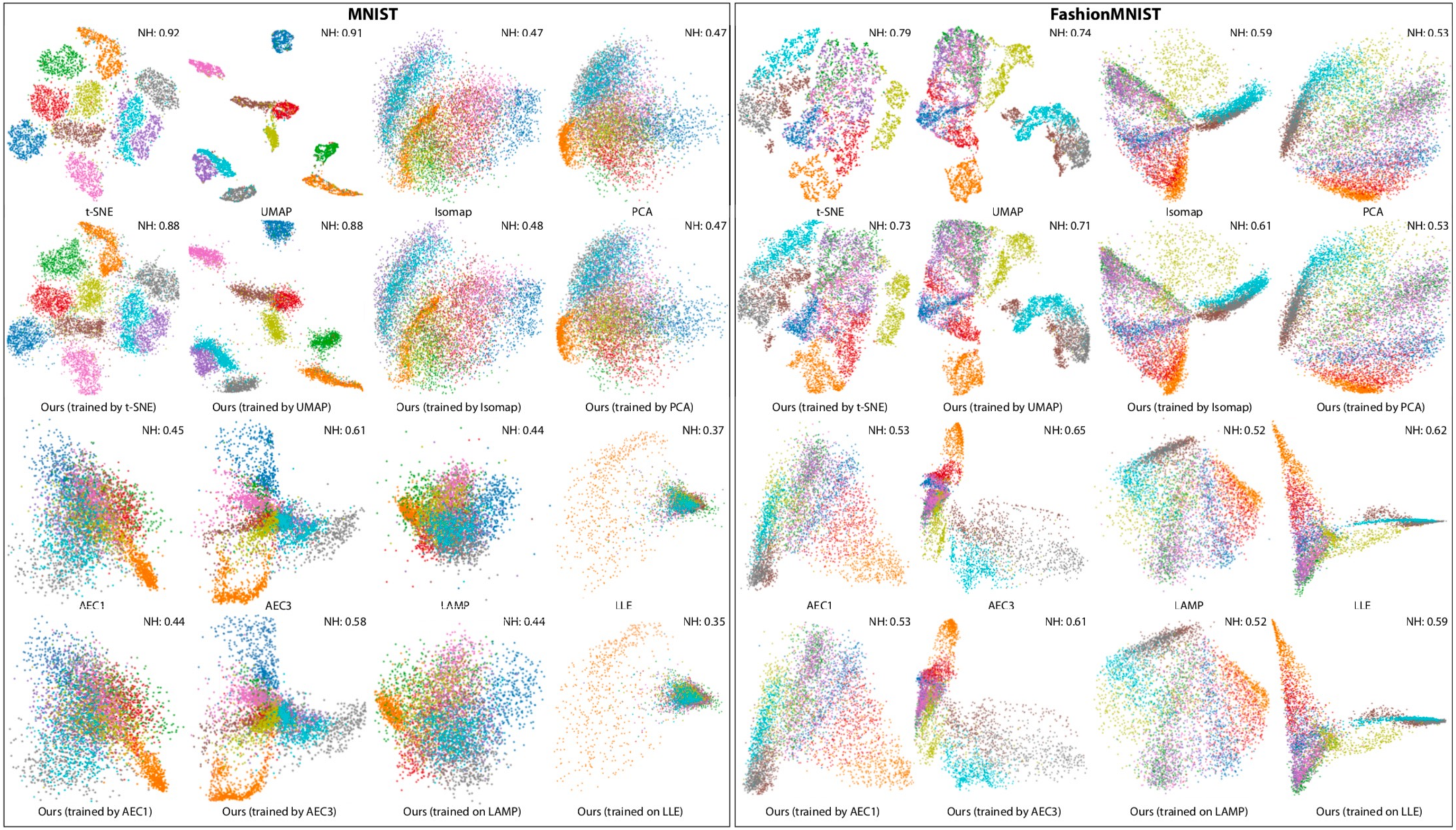
Quite good results with about **50 training epochs**

Learning different projection styles

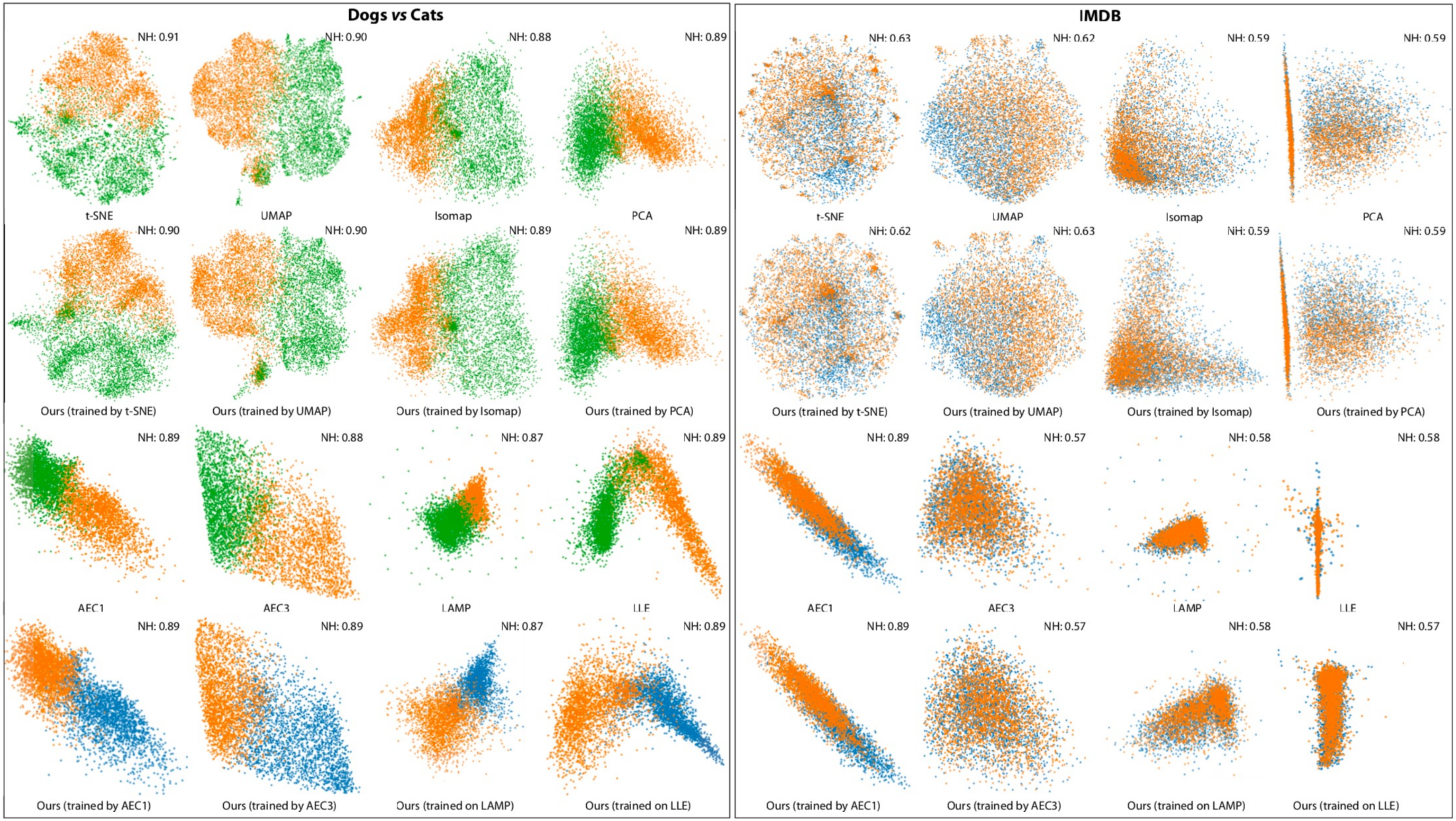


- we can imitate basically **any style**
- but, of course, the output quality will depend on the training material's quality (good `professor' = good quality, and conversely ☺)

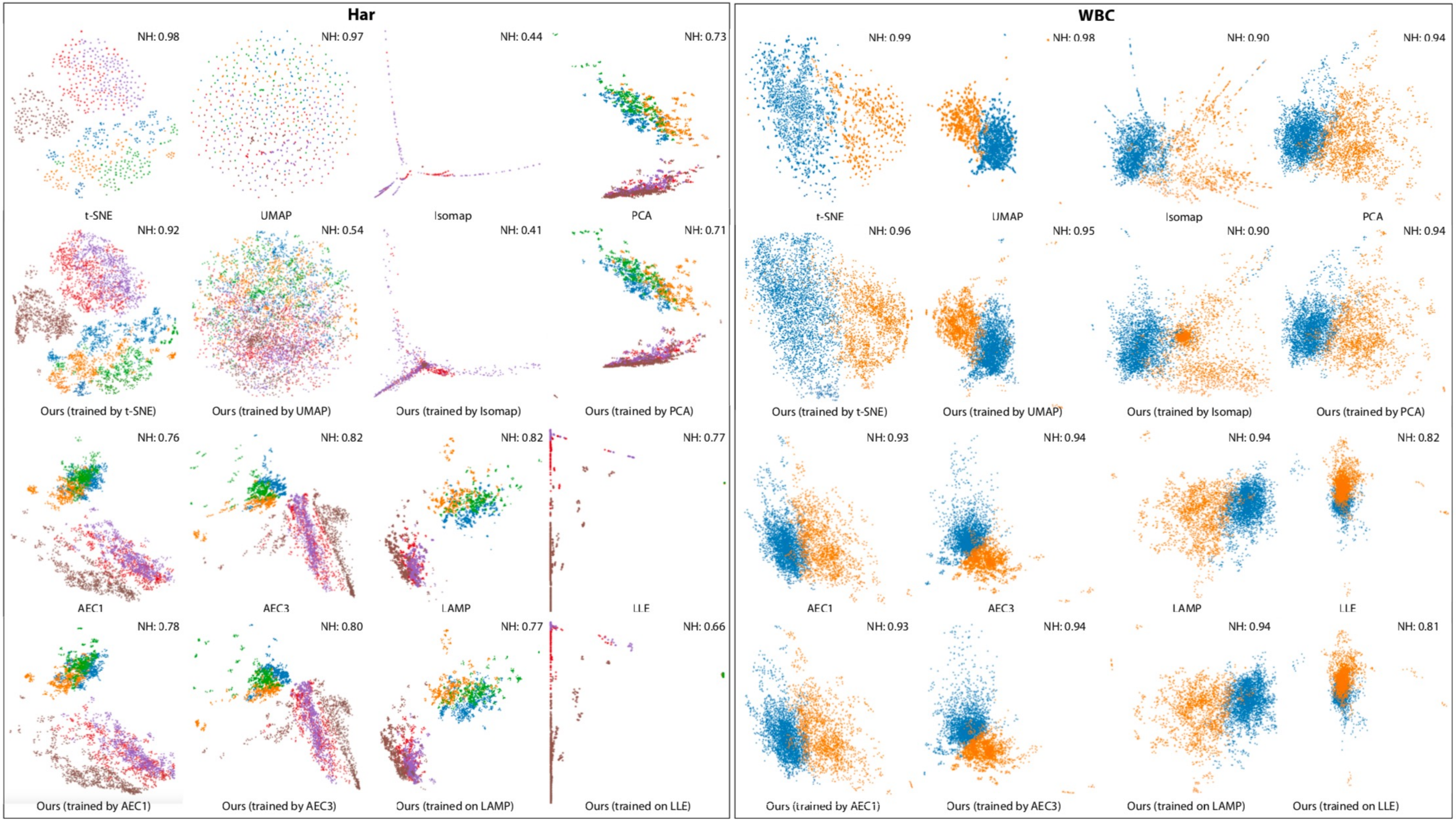
Learning different projection styles (cont'd)



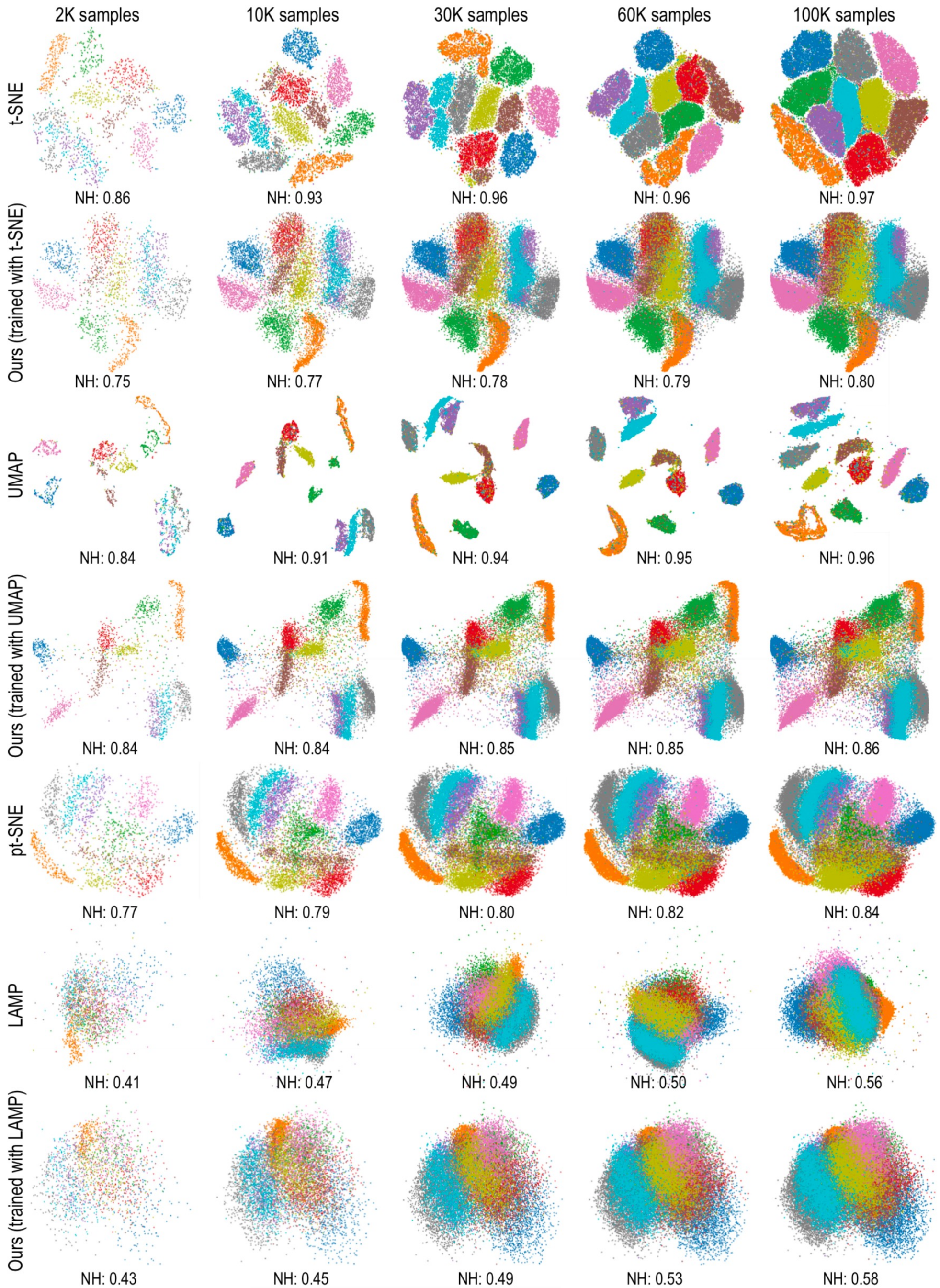
Learning different projection styles (cont'd)



Learning different projection styles (cont'd)



Out of sample capability



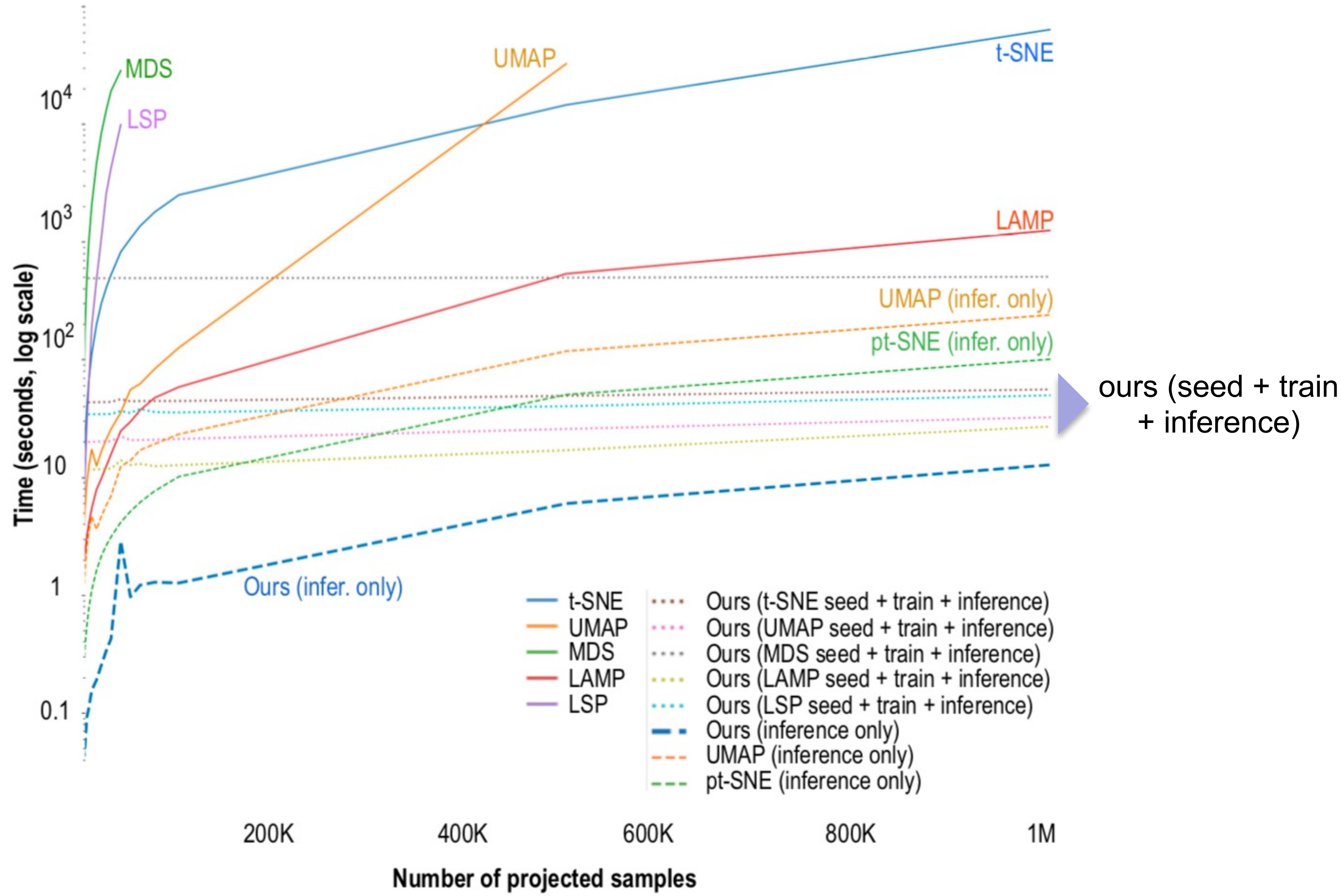
Testing

- train on a dataset D_0
- add samples to D_0 to create D_1, D_2, ...D_n
- project $P(D_0)$,...,$P(D_n)$
- compare with ground-truth $P^g(D_0)$,... $P^g(D_n)$

Results

- our method is always **stable** (out-of-sample capability by construction)
- most other methods are **not**
- we are close to the quality of parametric t-SNE (pt-SNE)

Computational scalability



Training + inference costs

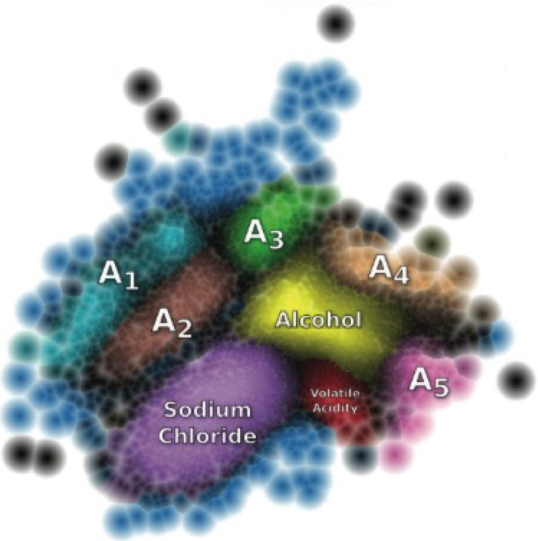
- **3K faster** than t-SNE, 2K faster than LAMP
- UMAP, LSP, MDS failed handling 1M points

Inference-only costs

- **3.5K faster** than t-SNE, 2K faster than LAMP
- 10x faster than pt-SNE

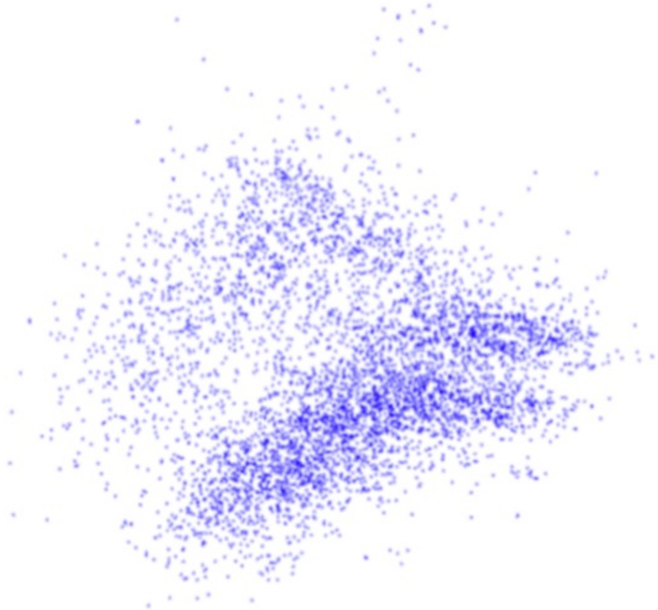
Code freely available: https://github.com/mespadoto/dlmp

5. Explaining Projections



How to Explain Projections?

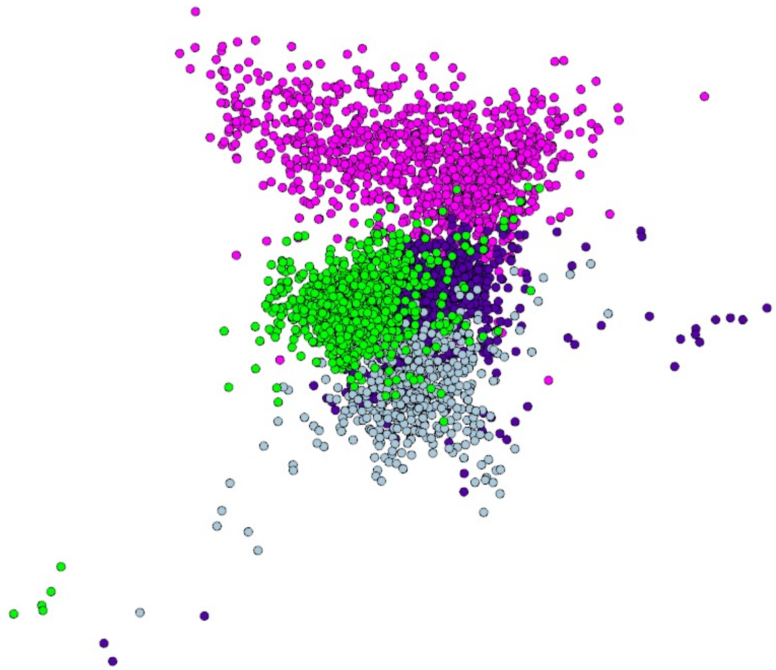
Raw projection visualization



- all we can see here is that some clusters and/or outliers exist
- this visualization is useless in most cases

How to Explain Projections?

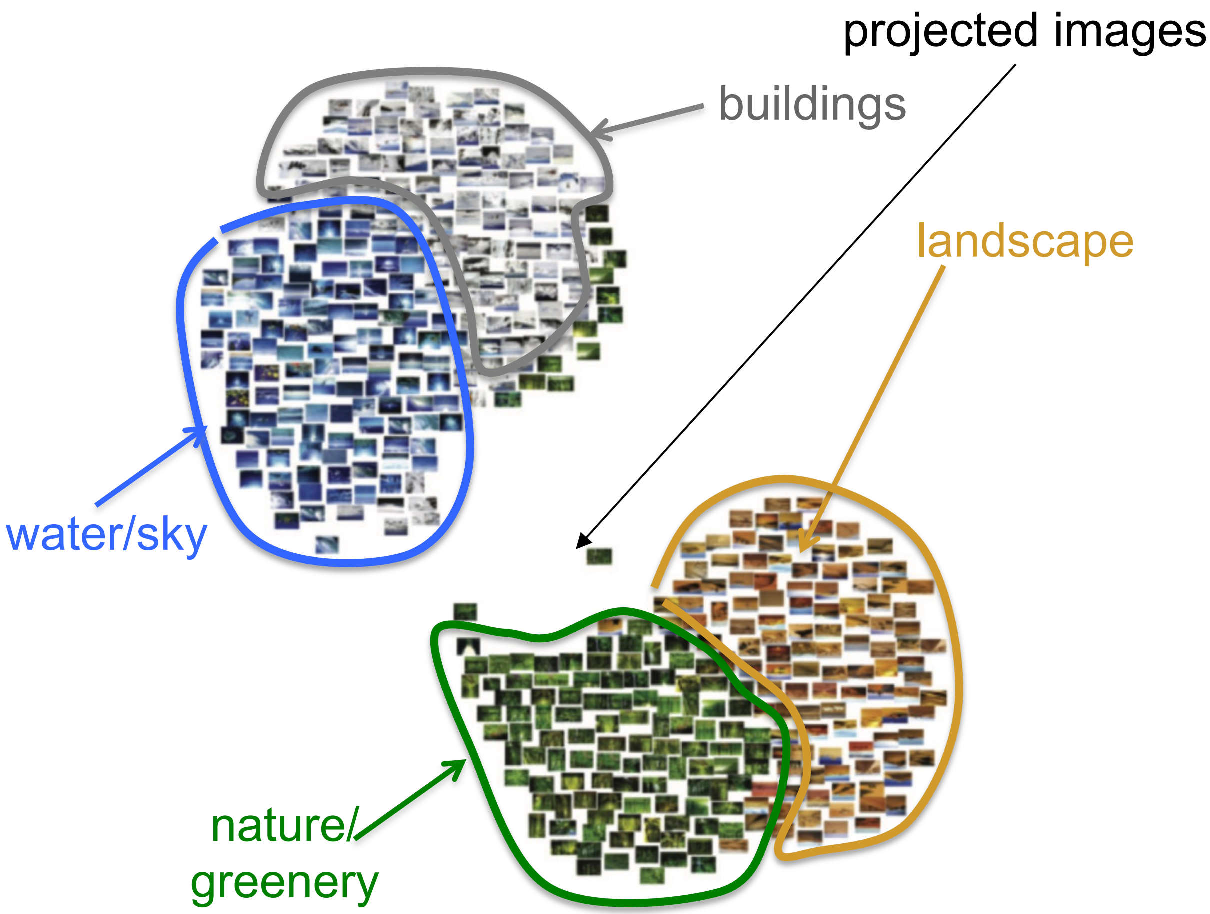
Explain by one dimension



- color code points on the value of **one dimension**
- if dimension was used in projection: explains what makes clusters similar
- if dimension not used in projection: shows its correlation with the projected dimensions
- user must **hand-pick** the dimension to color code
- only works if we have not-too-many, and meaningful, dimensions

How to Explain Projections?

Explain by depicting observations

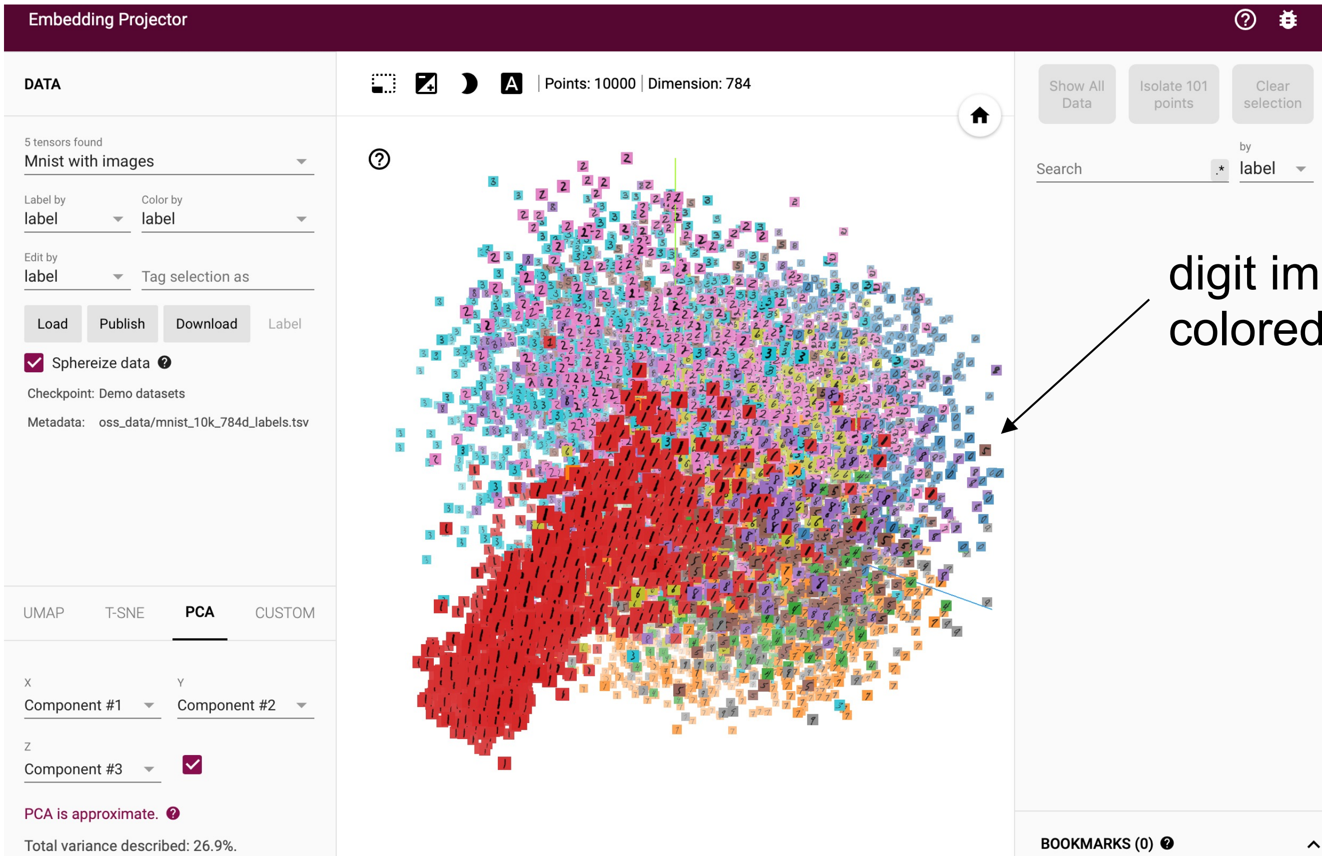


- only works if input data is **directly depictable** (e.g. images)
- scales poorly with number of observations

P. Joia et al., “Local Affine Multidimensional Projection,” IEEE TVCG, vol. 17, no. 12, 2011, pp. 2563–2571

How to Explain Projections?

Explain by depicting observations

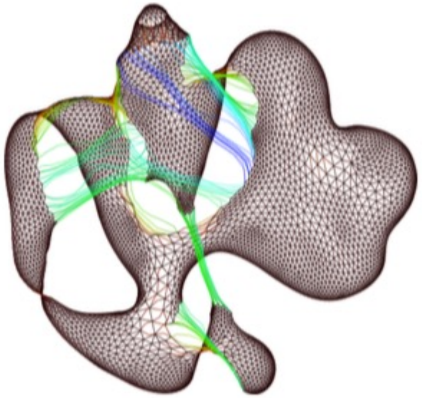


digit images colored by class

- only works if input data is **directly depictable** (e.g. images)
- scales poorly with number of observations

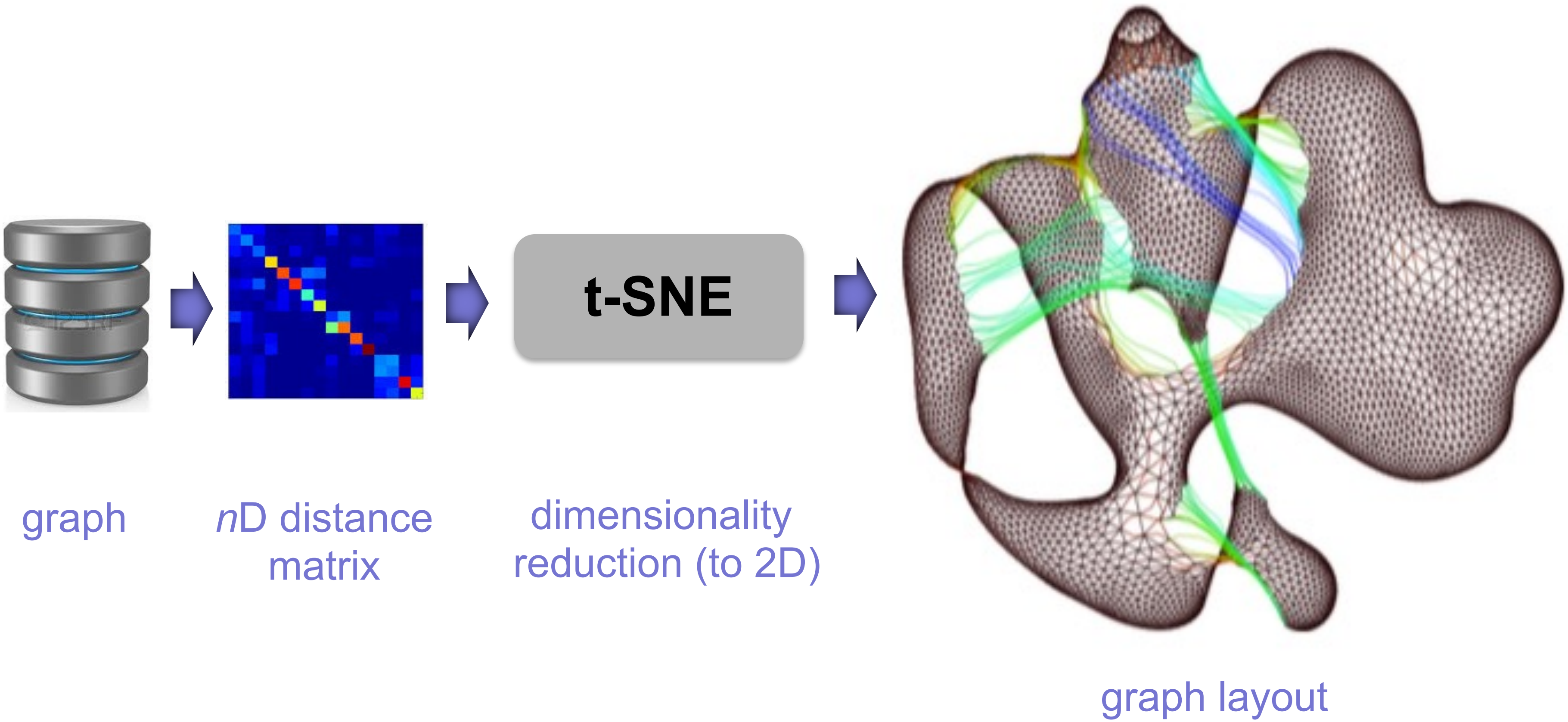
Projector Tensorflow (https://projector.tensorflow.org)

6. Connections



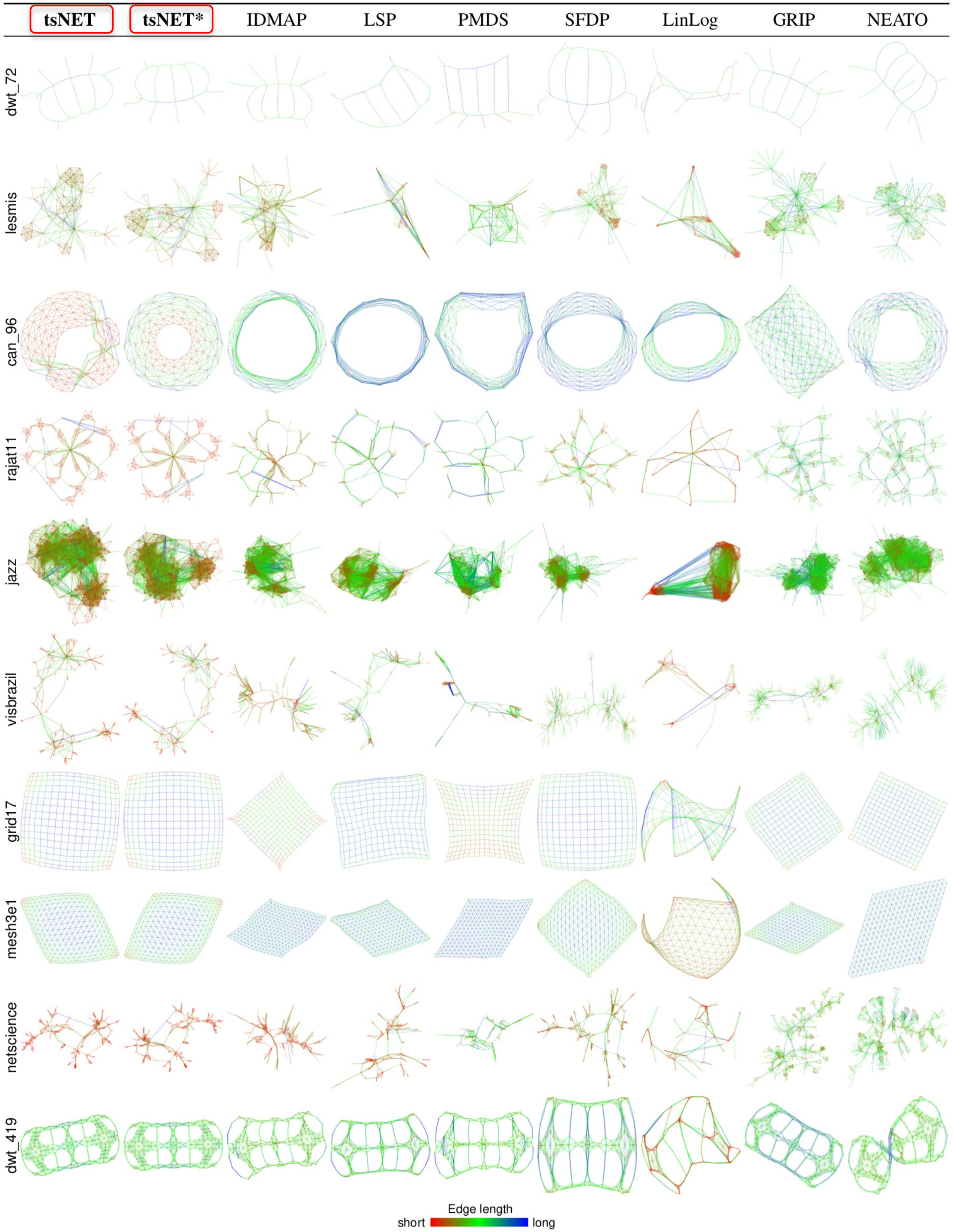
tsNET: Drawing Large Graphs with t-SNE

Consider a graph as a multidimensional (Euclidean) dataset!



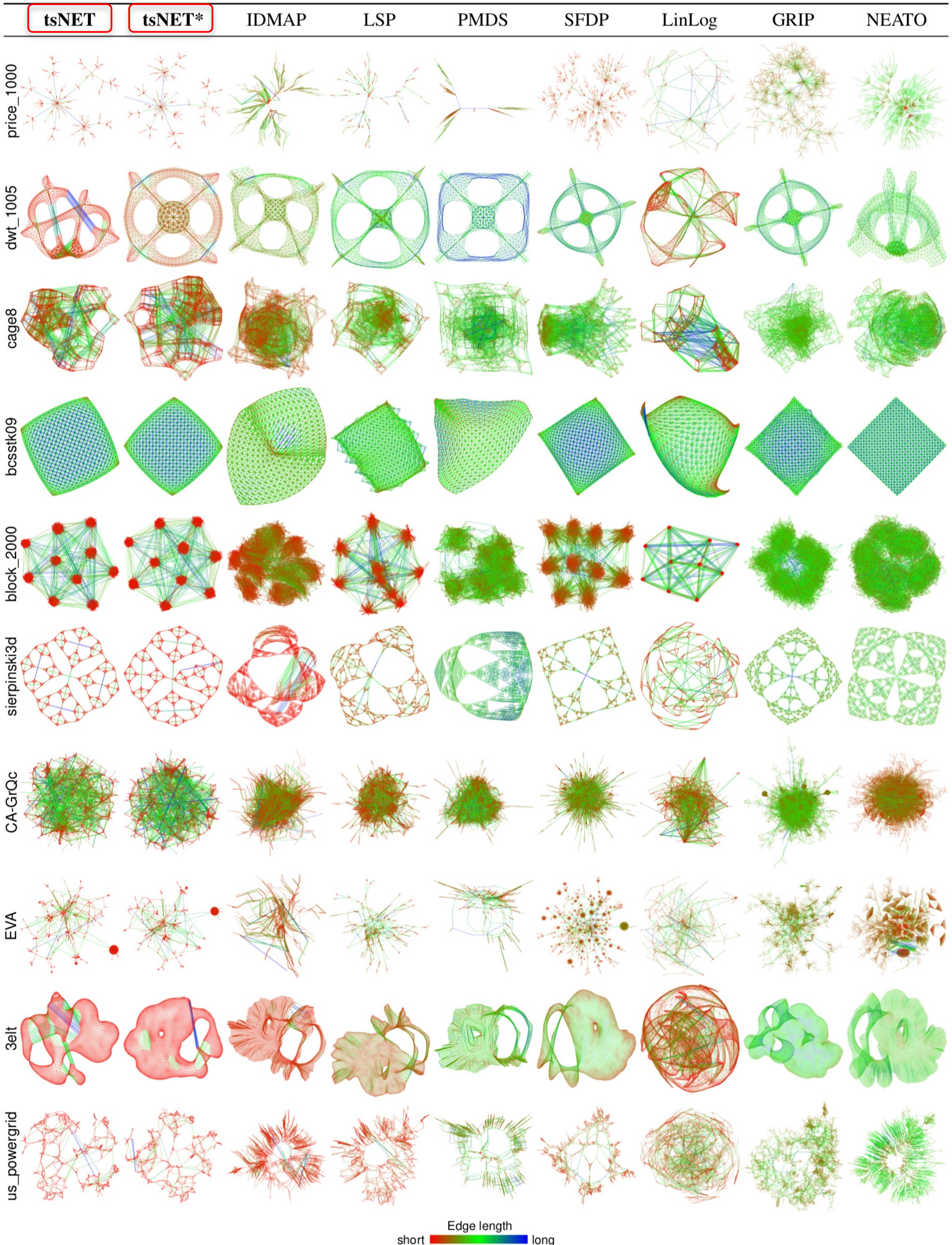
J. Kruiger, A. Telea *et al.* (2017) Graph layouts by t-SNE; Comp Graph Forum

Examples



tsNET
tsNET*
IDMAP
LSP
PMDS
SFDP
LinLog
GRIP
NEATO
dwt_72
lesmis
can_96
rajat11
jazz
visbrazil
grid17
mesh3e1
netscience
dwt_419
Edge length
short
long

More examples



Drawing Large Graphs with tsNET: Quality

	tsNET	tsNET*	IDMAP	LSP	PMDS	SFDP	LinLog	GRIP	NEATO
dwt_72	0.048	0.048	0.039	0.118	0.072	0.061	0.201	0.038	0.043
lesmis	0.109	0.111	0.111	0.226	0.162	0.112	0.219	0.099	0.084
can_96	0.112	0.084	0.085	0.088	0.092	0.075	0.091	0.104	0.072
rajat11	0.096	0.097	0.097	0.098	0.107	0.096	0.194	0.074	0.064
jazz	0.127	0.128	0.126	0.155	0.158	0.137	0.387	0.114	0.110
visbrazil	0.098	0.098	0.083	0.113	0.157	0.081	0.474	0.089	0.068
grid17	0.021	0.021	0.016	0.026	0.025	0.023	0.210	0.018	0.014
mesh3e1	0.014	0.014	0.004	0.006	0.003	0.036	0.076	0.009	0.005
netscience	0.101	0.100	0.075	0.096	0.103	0.105	0.182	0.070	0.063
dwt_419	0.024	0.024	0.023	0.022	0.026	0.052	0.112	0.022	0.054
price_1000	0.165	0.160	0.117	0.159	0.242	0.133	0.190	0.126	0.093
dwt_1005	0.152	0.035	0.030	0.030	0.029	0.029	0.219	0.026	0.096
cage8	0.185	0.203	0.151	0.142	0.140	0.147	0.207	0.150	0.122
bcsstk09	0.022	0.022	0.037	0.027	0.066	0.024	0.096	0.021	0.015
block_2000	0.193	0.189	0.164	0.205	0.181	0.162	0.302	0.155	0.144
sierpinski3d	0.077	0.093	0.152	0.092	0.091	0.079	0.310	0.068	0.063
CA-GrQc	0.182	0.189	0.150	0.175	0.182	0.148	0.220	0.172	0.129
EVA	0.171	0.161	0.148	0.141	0.233	0.124	0.325	0.149	0.098
3elt	0.110	0.090	0.052	0.045	0.057	0.060	0.317	0.049	0.046
us_powergrid	0.150	0.101	0.074	0.080	0.090	0.094	0.271	0.091	0.058
average	0.103	0.094	0.083	0.097	0.106	0.085	0.219	0.078	0.069

Normalized stress (rel.)

low (good) high (bad)

mean

	tsNET	tsNET*	IDMAP	LSP	PMDS	SFDP	LinLog	GRIP	NEATO
dwt_72	0.855	0.855	0.770	0.676	0.732	0.714	0.692	0.914	0.828
lesmis	0.715	0.712	0.748	0.642	0.674	0.729	0.649	0.666	0.695
can_96	0.658	0.671	0.535	0.530	0.515	0.547	0.540	0.533	0.565
rajat11	0.716	0.717	0.675	0.638	0.624	0.661	0.637	0.634	0.655
jazz	0.805	0.804	0.842	0.791	0.827	0.840	0.777	0.824	0.817
visbrazil	0.589	0.584	0.476	0.449	0.414	0.471	0.542	0.452	0.425
grid17	0.785	0.785	0.812	0.750	0.727	0.751	0.369	0.804	1.000
mesh3e1	0.904	0.904	0.993	0.957	0.994	0.809	0.587	0.896	0.999
netscience	0.711	0.707	0.539	0.583	0.473	0.622	0.614	0.559	0.510
dwt_419	0.739	0.741	0.723	0.741	0.695	0.654	0.542	0.751	0.658
price_1000	0.639	0.639	0.483	0.469	0.422	0.528	0.594	0.216	0.284
dwt_1005	0.609	0.619	0.512	0.503	0.485	0.523	0.390	0.516	0.455
cage8	0.435	0.437	0.207	0.278	0.221	0.235	0.349	0.193	0.200
bcsstk09	0.867	0.867	0.767	0.795	0.602	0.835	0.565	0.856	0.973
block_2000	0.374	0.372	0.205	0.279	0.166	0.287	0.339	0.155	0.160
sierpinski3d	0.579	0.580	0.387	0.492	0.326	0.534	0.438	0.549	0.561
CA-GrQc	0.480	0.483	0.170	0.207	0.179	0.183	0.349	0.081	0.119
EVA	0.801	0.802	0.707	0.706	0.717	0.696	0.780	0.406	0.459
3elt	0.663	0.715	0.415	0.485	0.384	0.595	0.248	0.576	0.506
us_powergrid	0.454	0.457	0.234	0.353	0.253	0.429	0.409	0.233	0.215
average	0.842	0.852	0.469	0.426	0.296	0.541	0.333	0.386	0.399

Neighborhood preservation (rel.)

least preserving (bad) most preserving (good)

mean

Normalized stress

best with NEATO

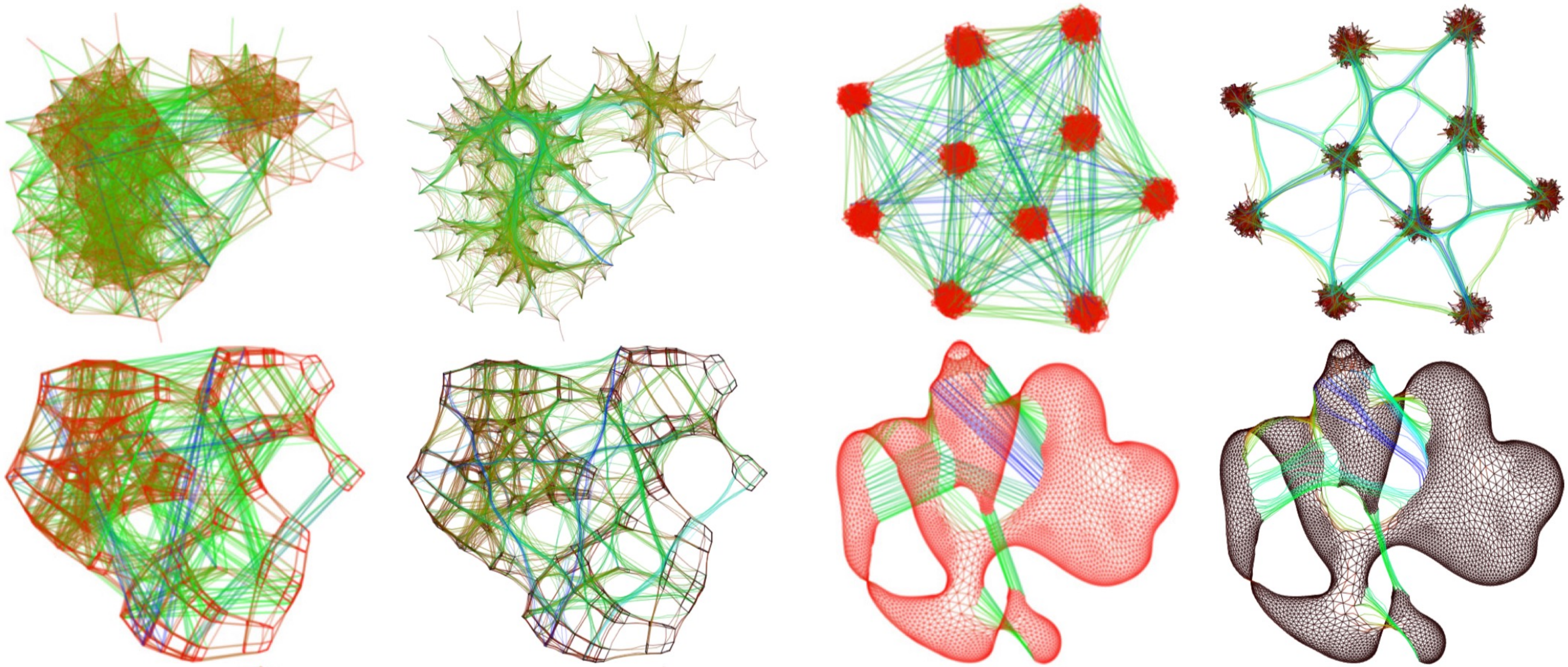
tsNET performs average

Normalized stress

tsNET is by far the best

all other are similarly poor

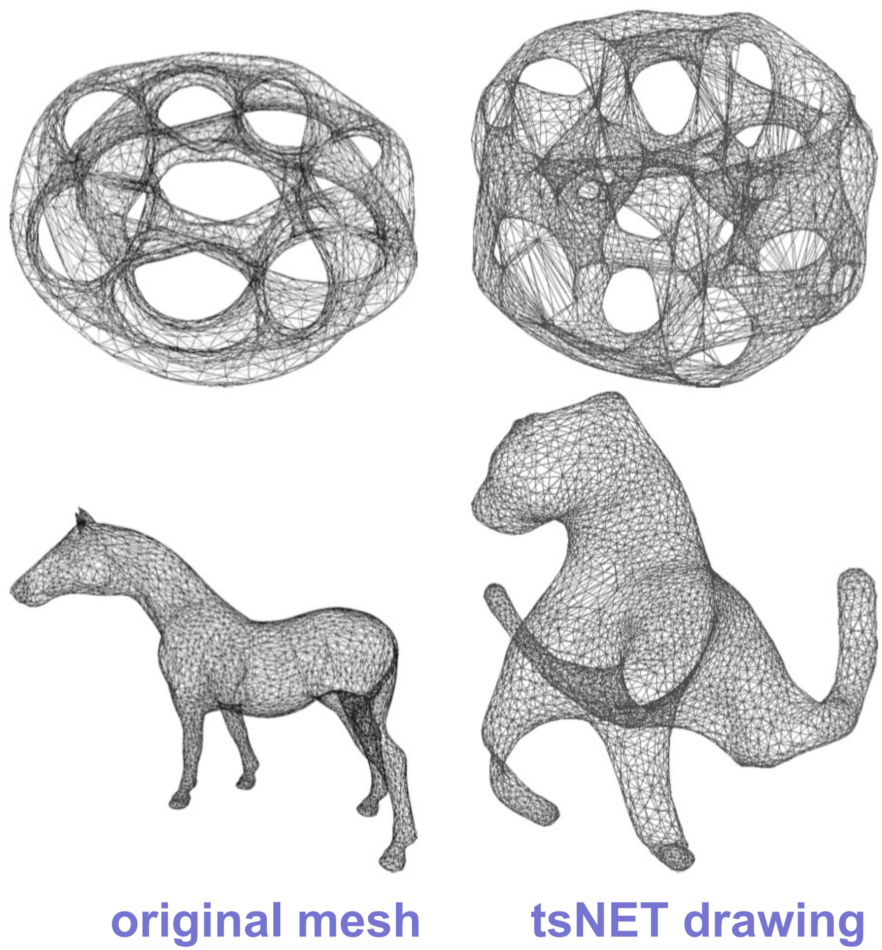
Drawing Large Graphs: Minimizing the Impact of Long Edges



standard drawing | bundling long | standard drawing | bundling long

- long edges are cluttering a graph drawing ☹
- it is however hard to avoid them
- minimize their **impact** by **bundling** edges over given length

To end: A crazy experiment



- take a 3D mesh
- throw away vertex coordinates, keep edges only
- draw the edge-graph with tsNET
- see how some shape information was recovered ☺

Summary: High-dimensional data visualization

For what

- datasets with many samples N and many (10..1000) dimensions n

Dimensionality reduction

- synthesize few (2..3) dimensions out of the n ones to encode sample similarity

Techniques

- PCA
- MDS
- Isomap
- t-SNE, UMAP
- NNP

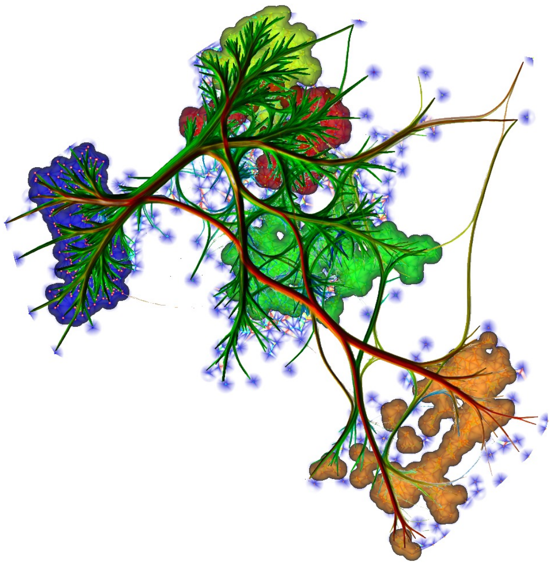
Challenges

- no perfect projection exists
- we must always measure projection errors

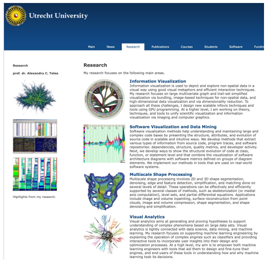
Thank you for your interest!

Alex Telea

a.c.telea@uu.nl



webspace.science.uu.nl/~telea001



- examples, applications
- code
- datasets
- papers
- people and projects

vig.science.uu.nl