MTTTS17
Dimensionality Reduction and Visualization

Spring 2020

Jaakko Peltonen

Lecture 11:
Neighbor Embedding Methods continued

This Lecture

Neighbor embedding by generative modeling

Some supervised neighbor embedding approaches

Fast approximations for embedding large data

Quality assessment techniques

Part 1: Neighbor embedding as generative modeling

Neighbor embedding by generative modeling

We've seen that neighbor embedding can be done by minimizing information theoretic **divergences** (Stochastic Neighbor Embedding)

We've seen that neighbor embedding can be optimized for an **information retrieval task** (Neighbor Retrieval Visualizer)

In machine learning, a prominent approach is **generative modeling**: maximization of the likelihood of observations, given a probabilistic model of the observations

Can neighbor embedding be done by generative modeling? Yes!

Neighbor embedding by generative modeling

Most probabilistic generative models generate the observed data: original feature values of points. Here we will not generate the original features, instead we **generate the original neighbor relationships**.

Stochastic Neighbor Embedding can be seen as a **generative model of the original neighbor relationships in the input space**.

$$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} = \sum_i KL(P_i || Q_i)$$

$$= \text{Const.} + \sum_i \sum_{j \neq i} p_{ij} \log q_{ij}$$

This is a sum over **observed center-neighbor pairs,** weighted by their proportional counts p_{ij}, We sum the log-likelihoods of those observations!

SNE maximizes likelihood of observing the original neighbor pairs. The generative model gives neighbor probabilities, based on locations of points on the display.

NeRV by generative modeling

Stochastic Neighbor Embedding can be seen as a generative model, but it only focuses on recall (misses) because its cost function is **dominated by misses.**

Idea: **change the retrieval model so that misses become less dominant, so that the model can also focus on false positives**.

New retrieval distribution: mixture of the **user model** and an **explaining away model**.

$$q_{ij} \propto r_{ij} + \gamma p_{ij}$$

new retrieval distribution

plain user model

explaining away model = true neighborhood distribution

amount of explaining away

Cost function is log-likelihood (generative modeling):

$$L = \sum_i \sum_{j \neq i} p_{ij} \log q_{ij}$$

fMRI measurements of 6 adults who received four types of stimuli:
- tactile (red)
- auditory tone (yellow)
- auditory voice (green)
- visual (blue).

Visualization by the new method, strong explaining away used during training. Different stimuli types become separated in the (unsupervised) visualization.

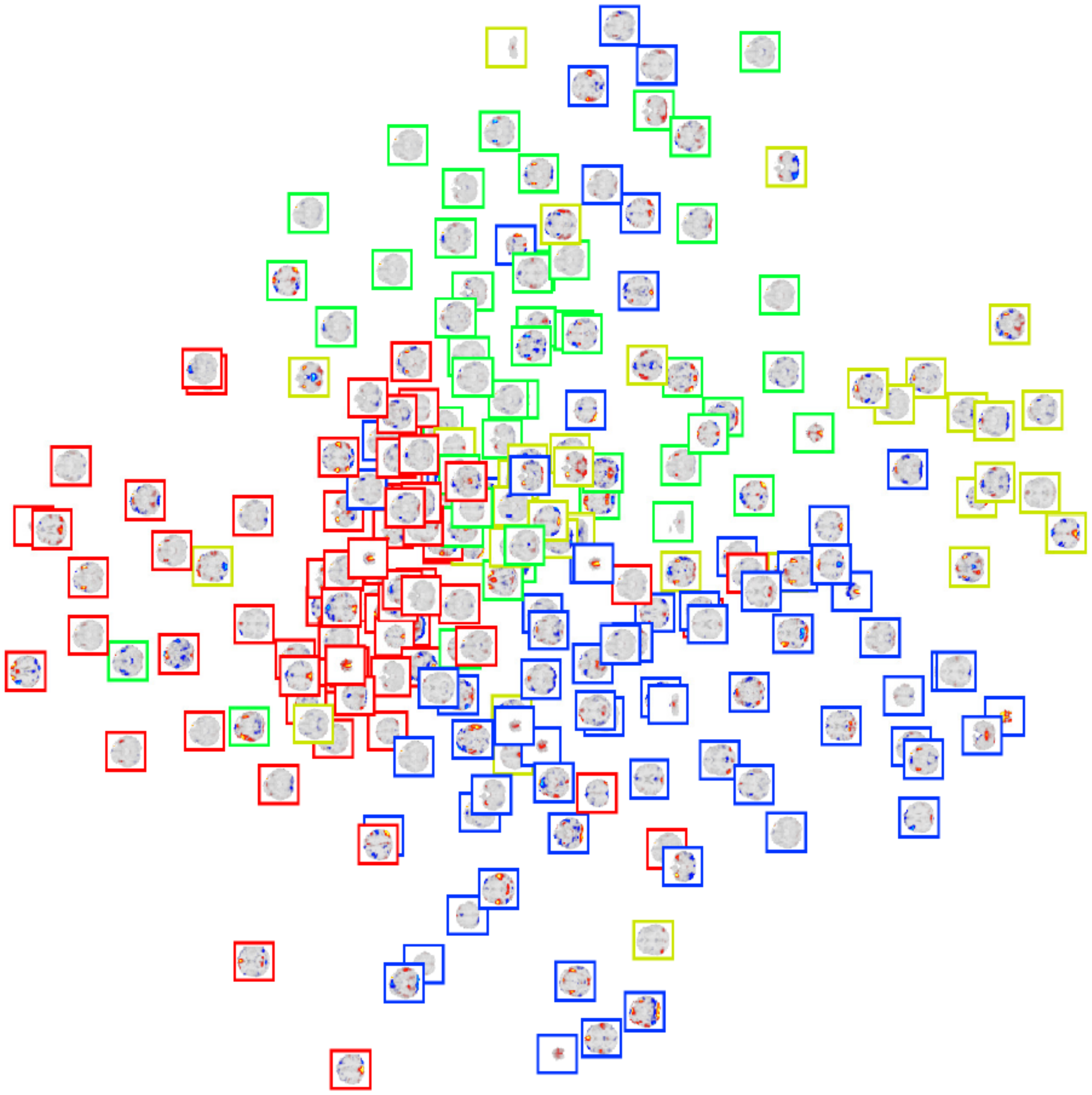


Image from ref. [1]

Part 2: Supervised neighbor embedding

Neighborhoods in Supervised Linear Projections

Consider finding a supervised parametric mapping, here a linear projection, whose aim is to separate classes of data as well as possible.

Linear discriminant analysis does this by making very rough assumptions about the class distributions: each class is Gaussian, and they are assumed to have the same covariance matrix. Optimizes abstract function: ratio of between-class variance to within-class variance

Can we build a method where
1) we still keep the projection linear, but we don't make any parametric assumptions about the class distributions?
2) task-based approach: we optimize the projection **for a task**

Yes, and it turns out this is related to neighborhoods.

Neighborhoods in Supervised Linear Projections

- Idea: use a linear projection for the mapping
- Use a **nonparametric class predictor (conditional class density estimator)** operating on the display to estimate classification performance.
- Optimize likelihood of observed class labels on the display.

Nonparametric estimation makes the class predictions **local**: they depend on local arrangements of the classes in each small neighborhood.

Discriminative Components of Data

- Mapping: y = f(x) = w^{T}x

- **Conditional class density estimator:**

$$\hat{p}(c|\mathbf{f}(\mathbf{x})) = \frac{G(\mathbf{f}(\mathbf{x}), c)}{\sum_{c'} G(\mathbf{f}(\mathbf{x}, c'))}$$

$$G(\mathbf{f}(\mathbf{x}), c) = \sum_{m=1}^{M} \psi_{mc} g(\mathbf{f}(\mathbf{x}), m)$$

$$g(\mathbf{f}(\mathbf{x}), m) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{(-\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{r}_m)\|^2 / 2\sigma^2)}$$

- Optimize log-likelihood of observed class labels on the display.

$$L = \sum_{(\mathbf{x}, c)} \log \hat{p}(c|\mathbf{f}(\mathbf{x}))$$

Nonparametric estimation makes class predictions **local**: they depend on arrangement of classes in each small neighborhood.

Discriminative Components of Data

Comparison on a handwritten digit data set

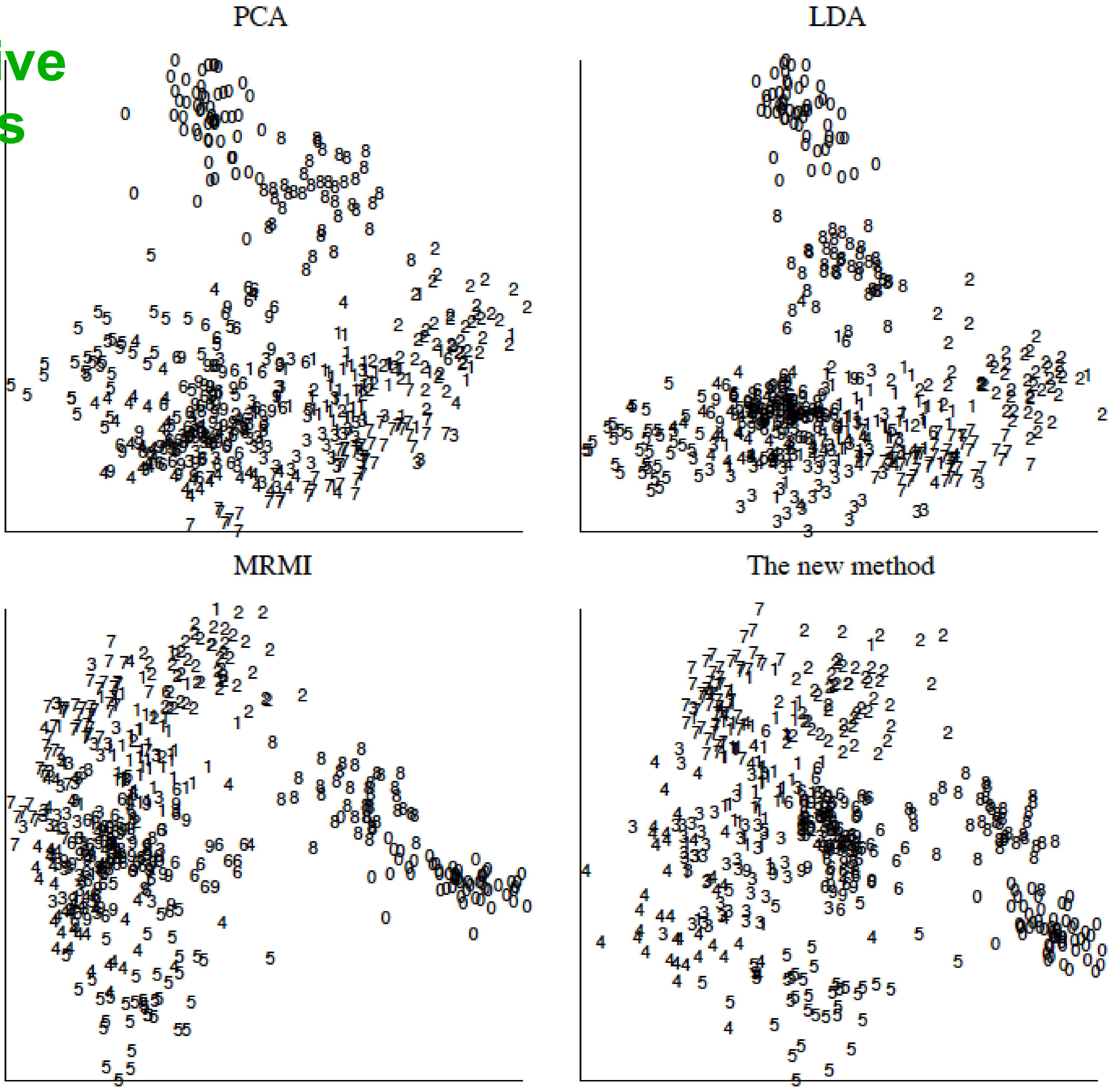


Image from ref. [2]

Supervised Distances

In distance-based methods, a basic form of supervision would change input distances to **d(i,j)=1[class(i)=class(j)].**
Potential problem: if supervised distances are based on classes only, originally far-away same-class points become collapsed, and originally close-by different-class points become separated. The collapsed distances tell nothing about the arrangement of the points in the feature space.---->Not good for exploring?

Idea: create a **topology-preserving** supervised metric, so that:
- class information is used locally (for infinitesimally small distances).
- local distances are extended to global ones as geodesics over the neighborhood graph
- does not collapse originally far-away points, or rip apart originally nearby points

The Learning Metric

The Learning Metric is a supervised “topology-preserving” distance metric learned from class probabilities
- a Riemannian metric

- assume we have a class estimator $p(c|\mathbf{x})$

- local distance:

$$d_L^2(\mathbf{x}, \mathbf{x} + d\mathbf{x}) \equiv D_{KL}(p(c|\mathbf{x})||p(c|\mathbf{x} + d\mathbf{x})) = d\mathbf{x}^T \mathbf{J}(\mathbf{x}) d\mathbf{x}$$

$$\mathbf{J}(\mathbf{x}) = E_{p(c|\mathbf{x})} \left\{ \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x}) \right) \left(\frac{\partial}{\partial \mathbf{x}} \log p(c|\mathbf{x}) \right)^T \right\}$$

- global distance:

$$d(p, q) = \inf_{\{\gamma|\gamma(0)=p,\gamma(1)=q\}} \int_0^1 d_L(\gamma(t), \gamma(t+dt))dt$$

The Learning Metric

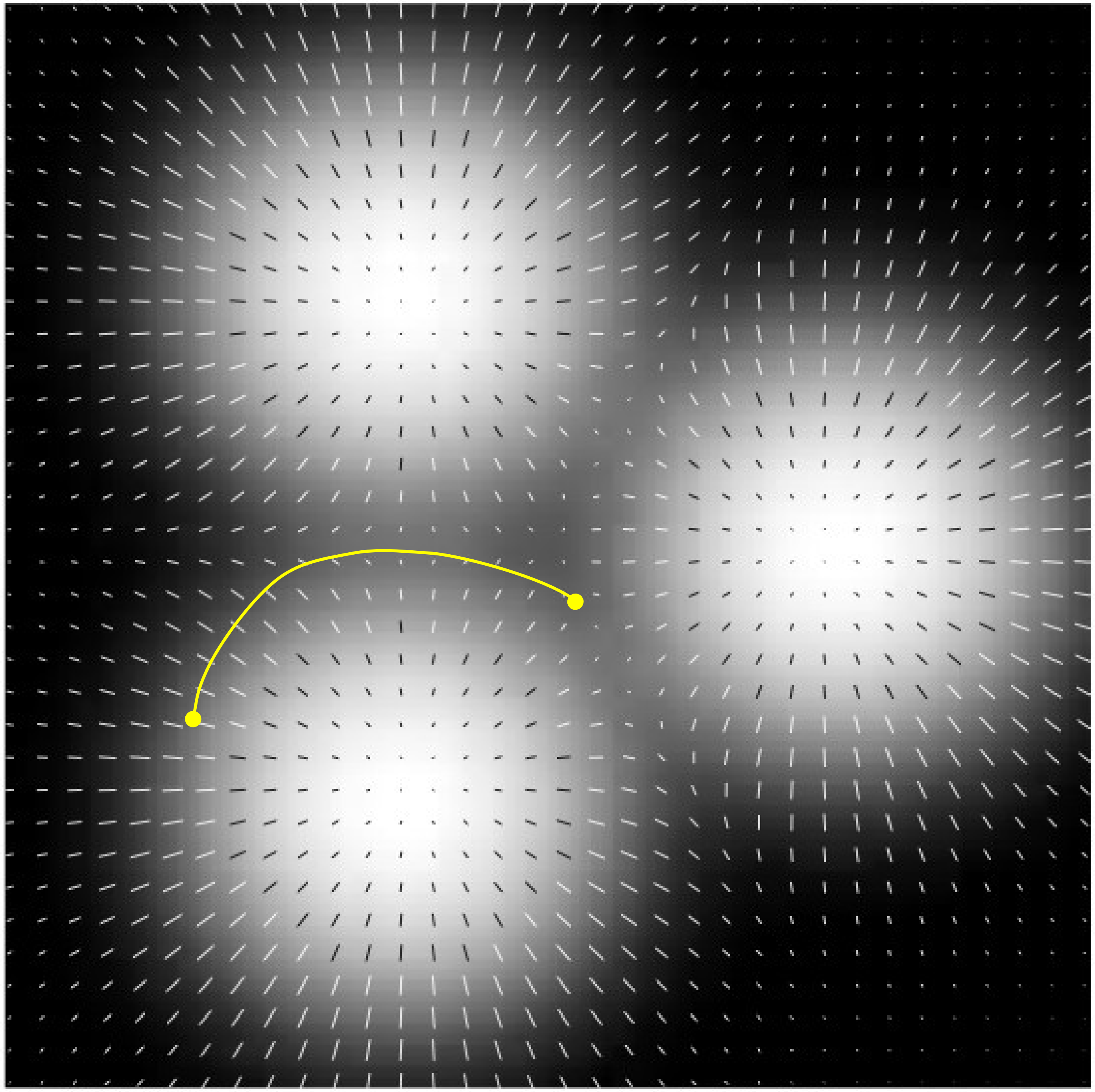


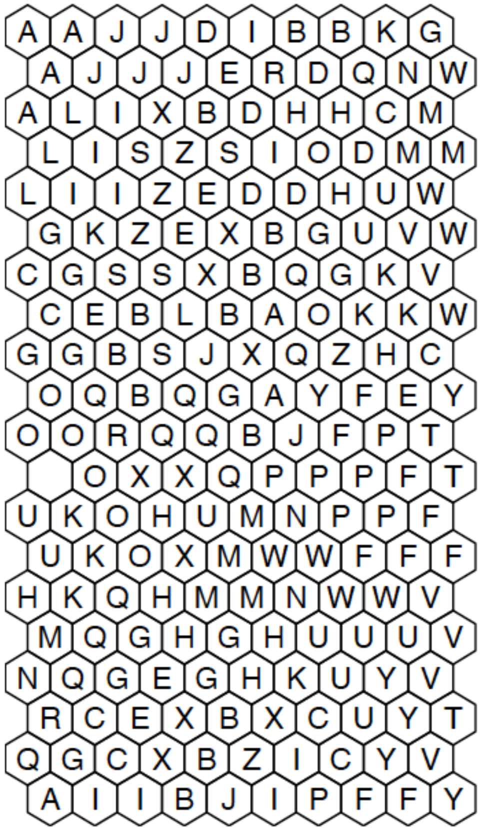
Image from
J. Peltonen's
D.Sc. thesis

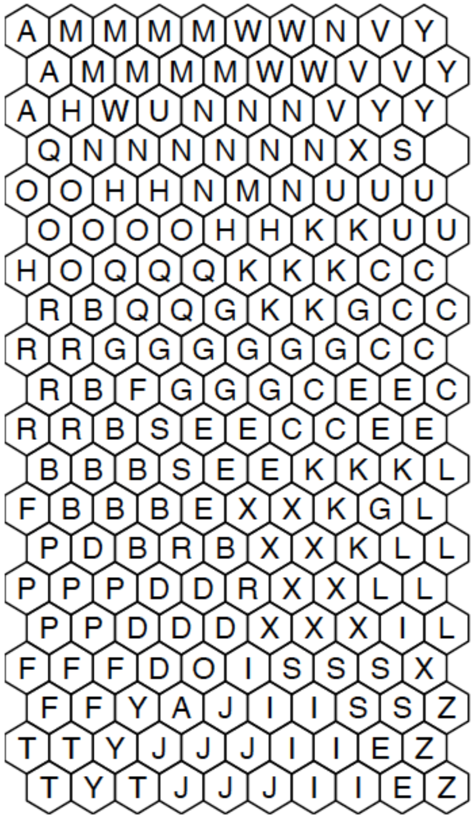
Supervision for the Self-Organizing Map

Once you have a topology-preserving metric, you can apply it when distances are needed.

In SOM, distance is used to 1) find the winner node for an input, and 2) compute a gradient, to adjust nodes towards the input.
- Use the learning metric for the distance.
- The *natural gradient* is same as the Euclidean metric gradient

Traditional SOM





SOM in learning metric

Image from ref. [3]

Supervision for NeRV

Simply compute the input neighborhoods based on distances in the learning metric.

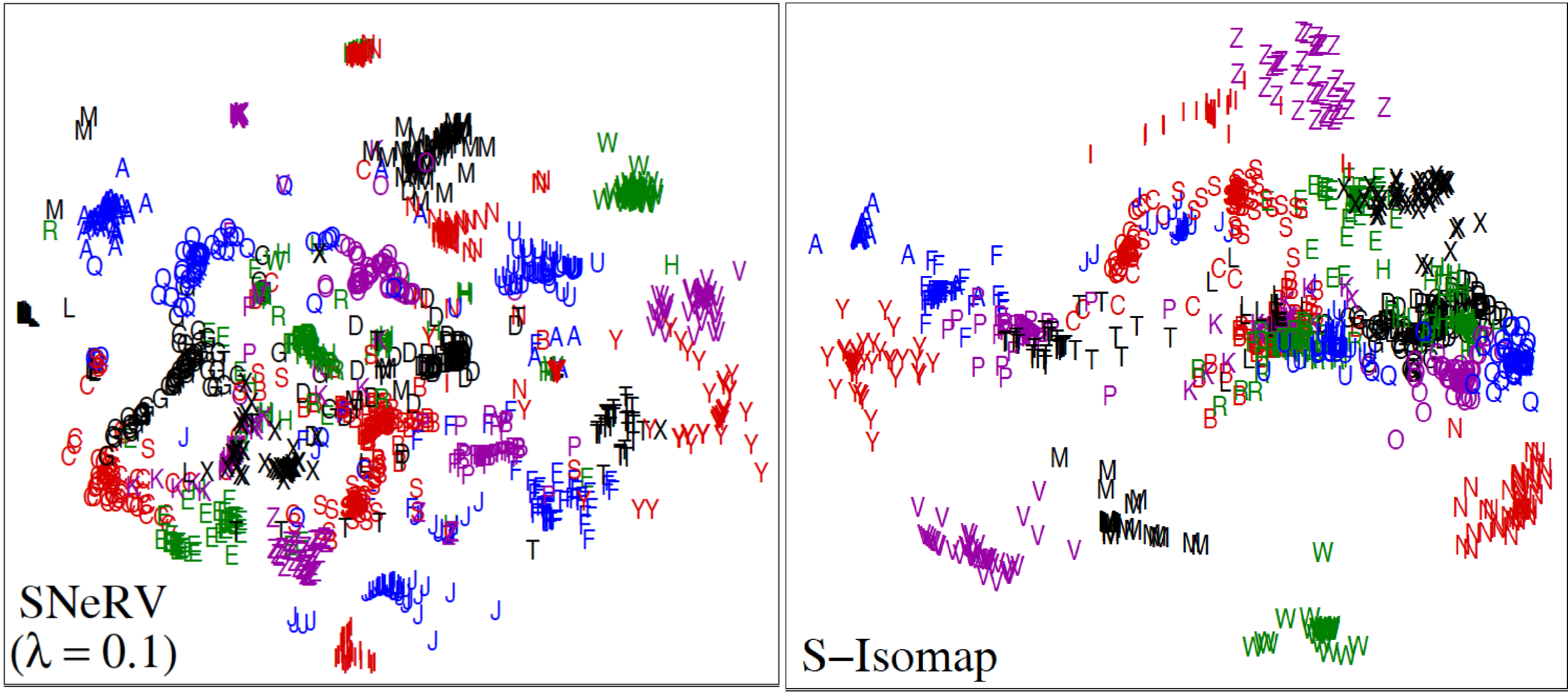


Image from ref. [4]

NeRV with a linear projection

Input neighborhood

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

Output neighborhood

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

Tradeoff measure = NeRV cost function

$$E_{\mathrm{NeRV}} = \lambda \mathbb{E}_i[D(p_i,q_i)] + (1-\lambda)\mathbb{E}_i[D(q_i,p_i)]$$

Restrict $\mathbf{y}_{\mathrm{i}} = \mathbf{W}^{\mathrm{T}}\mathbf{x}_{\mathrm{i}}$

Minimize cost with respect to projection W

NeRV with a linear projection

Input neighborhood

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

Output neighborhood

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

Tradeoff measure = NeRV cost function

$$E_{\text{NeRV}} = \lambda \mathbb{E}_i[D(p_i,q_i)] + (1-\lambda)\mathbb{E}_i[D(q_i,p_i)]$$

Restrict $\mathbf{y}_\mathrm{i} = \mathbf{W}^\mathrm{T}\mathbf{x}_\mathrm{i}$

Minimize cost with respect to projection W

Features do not need to be from the same source as input neighborhoods.

Input neighborhoods = supervision

Part 3: Scalable neighbor embedding

Scalability Issues

Neighbor embedding approaches typically need to evaluate all pairwise distances in the display during iterative optimization. This gives $O(N^2)$ computation time per iteration for N data points. (Some manifold learning approaches need to invert kernel matrices which can take $O(N^3)$ computation time!)

In large data sets (millions of points) this takes prohibitively much time – computation would not finish in a reasonable time. In interactive applications very fast changes to plots may be needed.

----> Need faster computation, with less than quadratic dependence on data set size.

Naive approach: subsample the data set. Problem 1: can lose interesting details of the data distribution and relationships. Problem 2: in many methods, not easy to embed left-out points.

Solution approach 1 (for NeRV)

The cost function is based on several sums over neighbors:

$$E_{\mathrm{NeRV}} = \lambda \sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} + (1-\lambda) \sum_i \sum_{j \neq i} q_{j|i} \log \frac{q_{j|i}}{p_{j|i}}$$

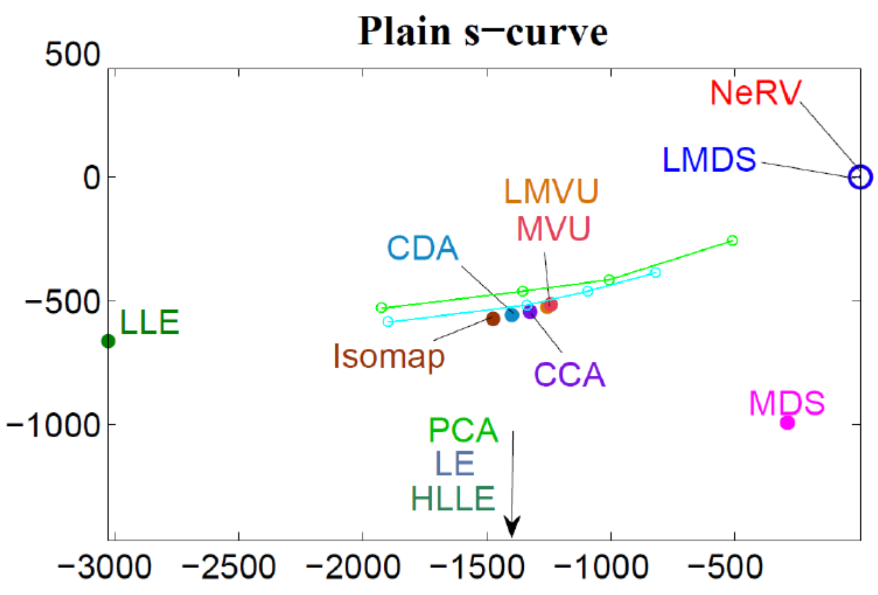
Suppose we have estimated the data distribution in the input and output. Then we can replace each sum over j with an **expected value** of the sum.

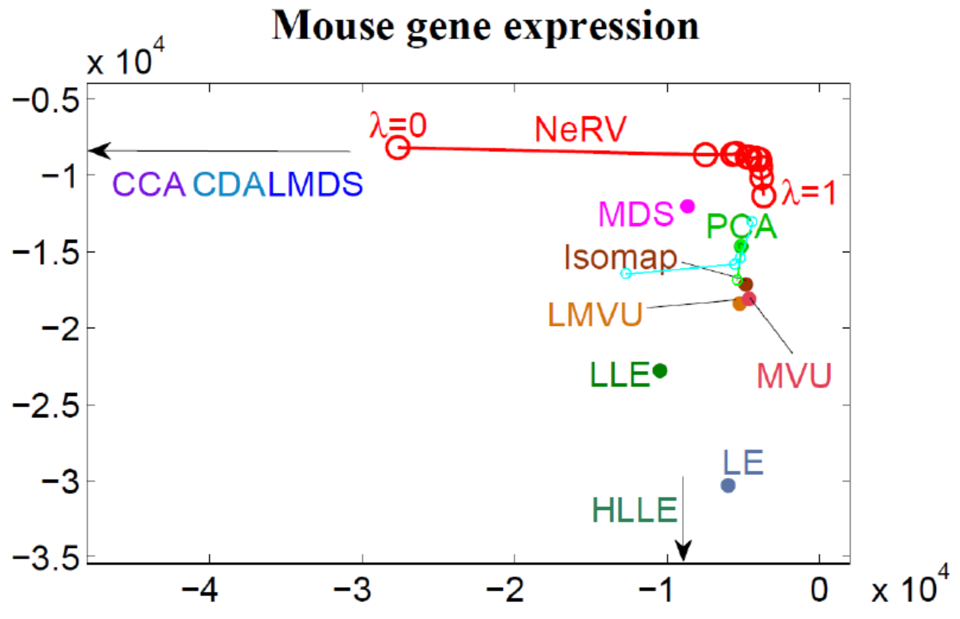
Computing the expected value depends on the complexity of the data distribution, not on the amount of observed samples ----> computational speedup!

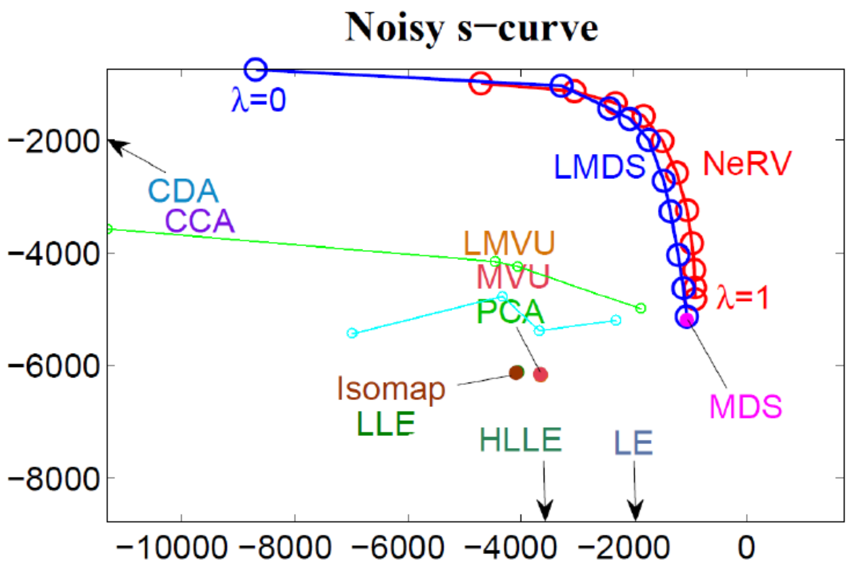
Simple approach: estimate a Gaussian mixture model in the original space. Can be done in O(N) time. Estimate the corresponding locations/widths of mixture components in the output: can be done in O(N) time each iteration. (All times also depend on complexity of the mixture.)

Solution approach 1 (for NeRV)

Results not as good as original NeRV but faster







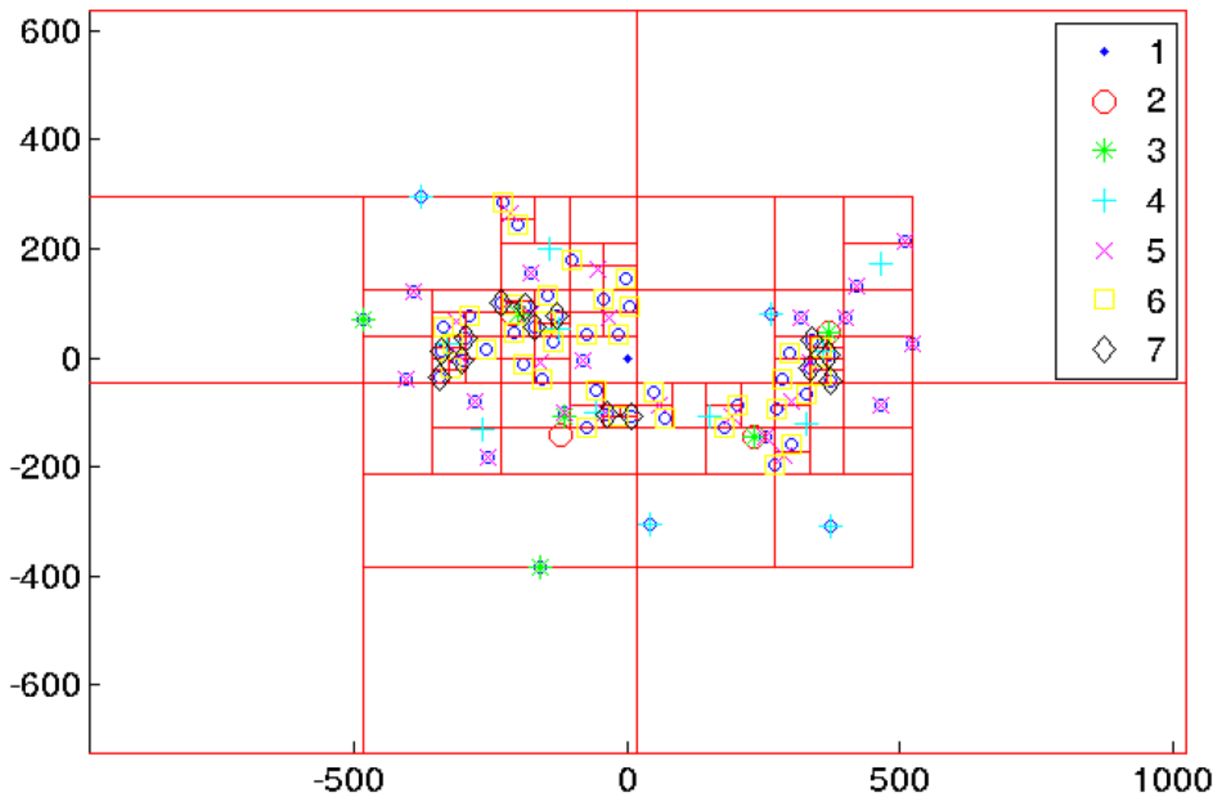
Method	DataSize	Time	−1·Recall	−1·Precision
NeRV	1000	375.6	-3.911	-85.212
NeRV	2000	1498	-4.605	-120.295
NeRV	4000	5986	-6.008	-162.443
NeRV	8000	23358	-6.510	-205.822
Our	1000	35.2	-5.198	-84.832
Our	2000	31.9	-6.365	-119.180
Our	4000	135.7	-6.085	-162.230
Our	8000	170.1	-233.125	-170.4125

Image from ref. [6]

Approach 2 (Barnes-Hut for SNE/NeRV)

Use a hierarchical clustering to approximate locations on the output display. Hierarchical clustering is very fast to compute in 2D by a **QuadTree**.

Hierarchical clustering around point i: nearby clusters from deep levels of the hierarchy (precise, lots of clusters), far-off clusters from upper levels (rough approximation, fast computation).



Approximate location of a neighbor by its cluster center. ----->To compute costs/gradients, it is enough to sum over cluster centers weighted by their point totals, instead of summing over individual neighbors. ----> Speedup!

Image from ref. [7]

Approach 2 (Barnes-Hut for SNE/NeRV)

For each centerpoint i, approximate location of a neighbor by its cluster center. ----->To compute costs/gradients, it is enough to sum over cluster centers weighted by their point totals, instead of summing over individual neighbors. ----> Speedup: O(n log n)

$$\sum_j f\left(\|y_i - y_j\|^2\right) = \sum_t \sum_{j \in G_t^i} f\left(\|y_i - y_j\|^2\right)$$
$$\approx \sum_t |G_t^i| f\left(\|y_i - \hat{y}_t\|^2\right),$$

Barnes-Hut approximation often used in N-body problems in physics

$$g_{ij} = f'\left(\|y_i - y_j\|^2\right)$$
$$\sum_j g_{ij}\left(y_i - y_j\right) = \sum_t \sum_{j \in G_t^i} g_{ij}\left(y_i - y_j\right)$$
$$\approx \sum_t |G_t^i| f'\left(\|y_i - \hat{y}_t^i\|^2\right)\left(y_i - \hat{y}_t^i\right)$$

Approach 2 (Barnes-Hut for SNE/NeRV)

t-SNE

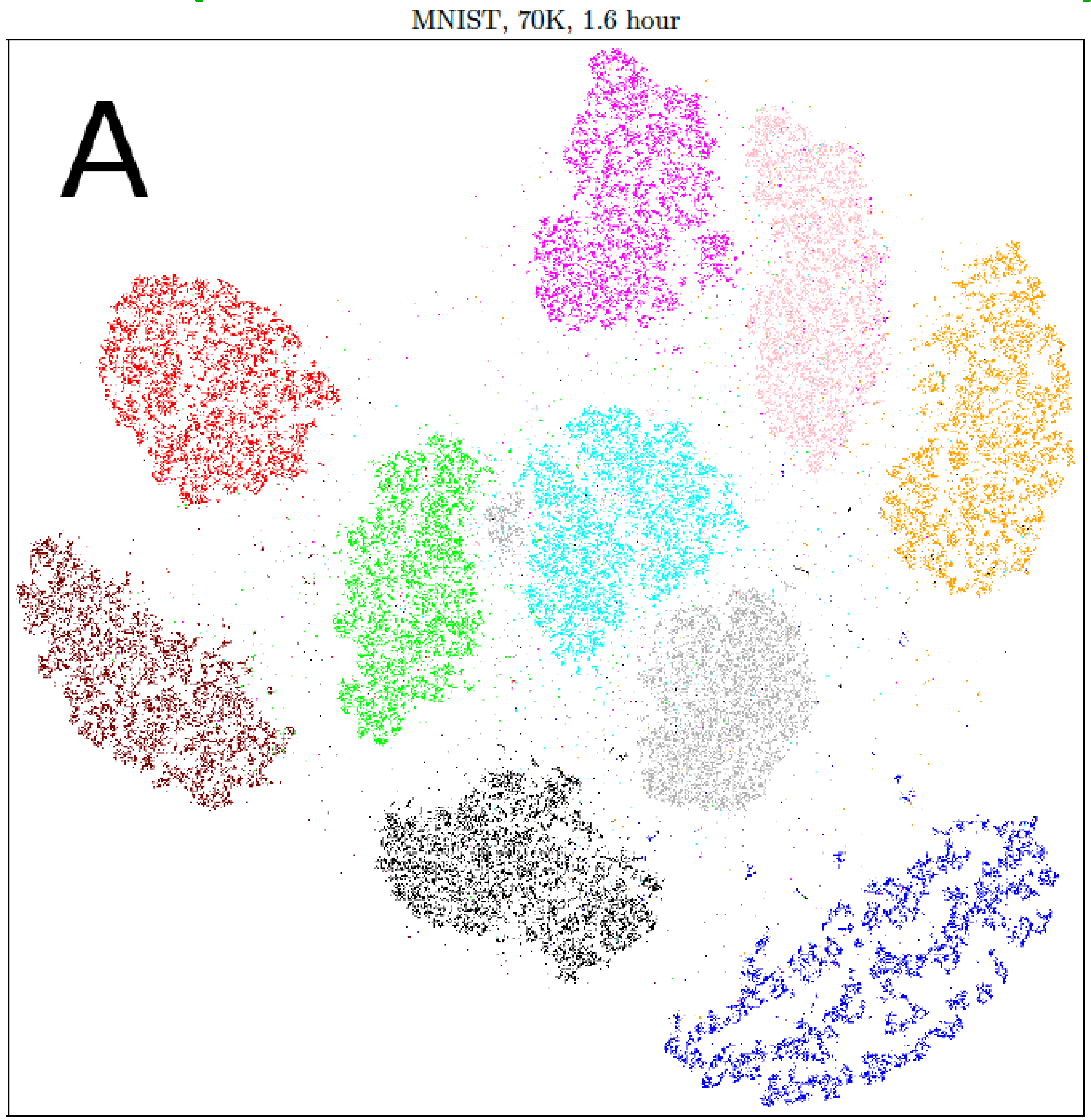
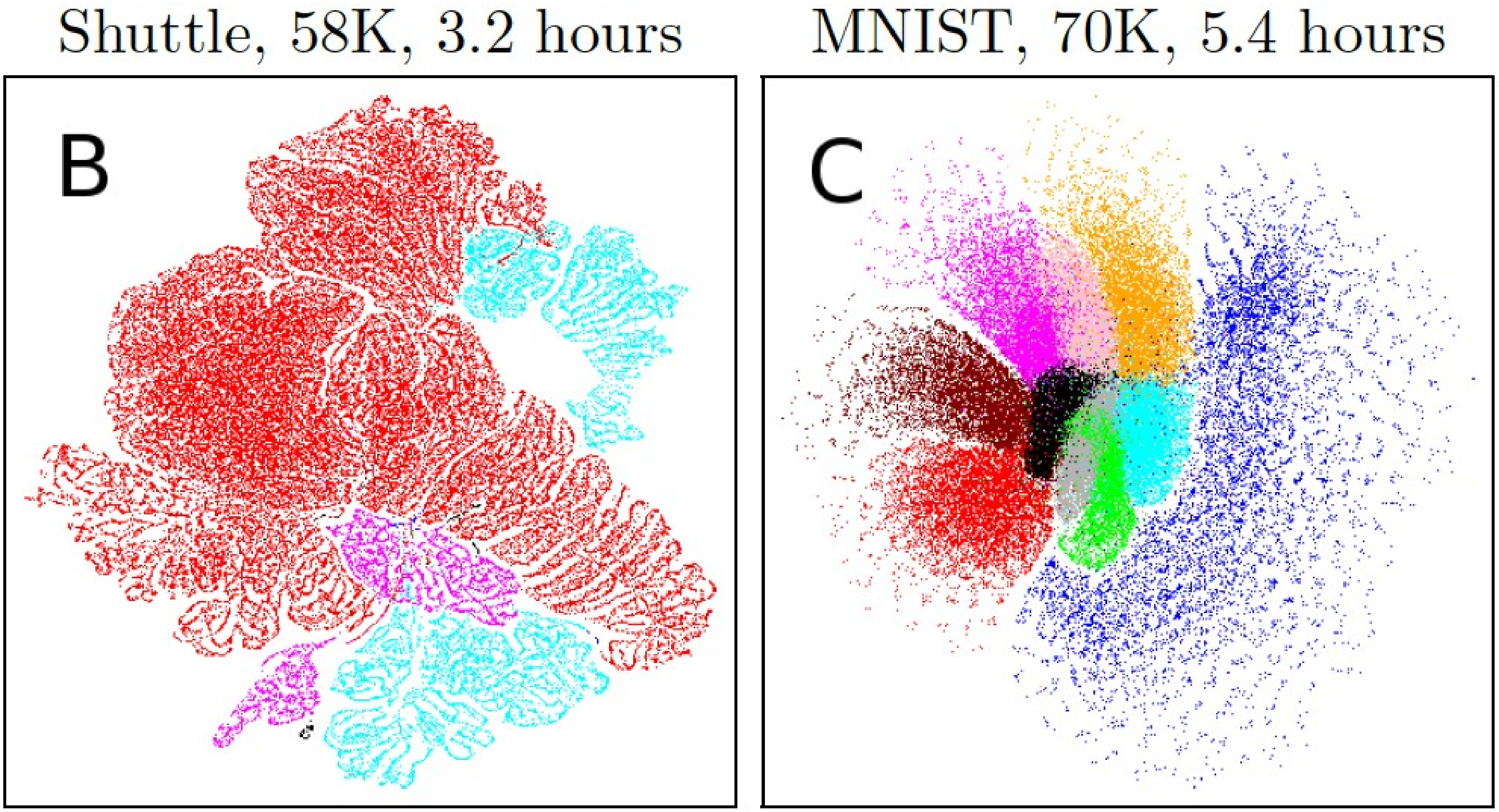


Image from ref. [7]

Approach 2 (Barnes-Hut for SNE/NeRV)



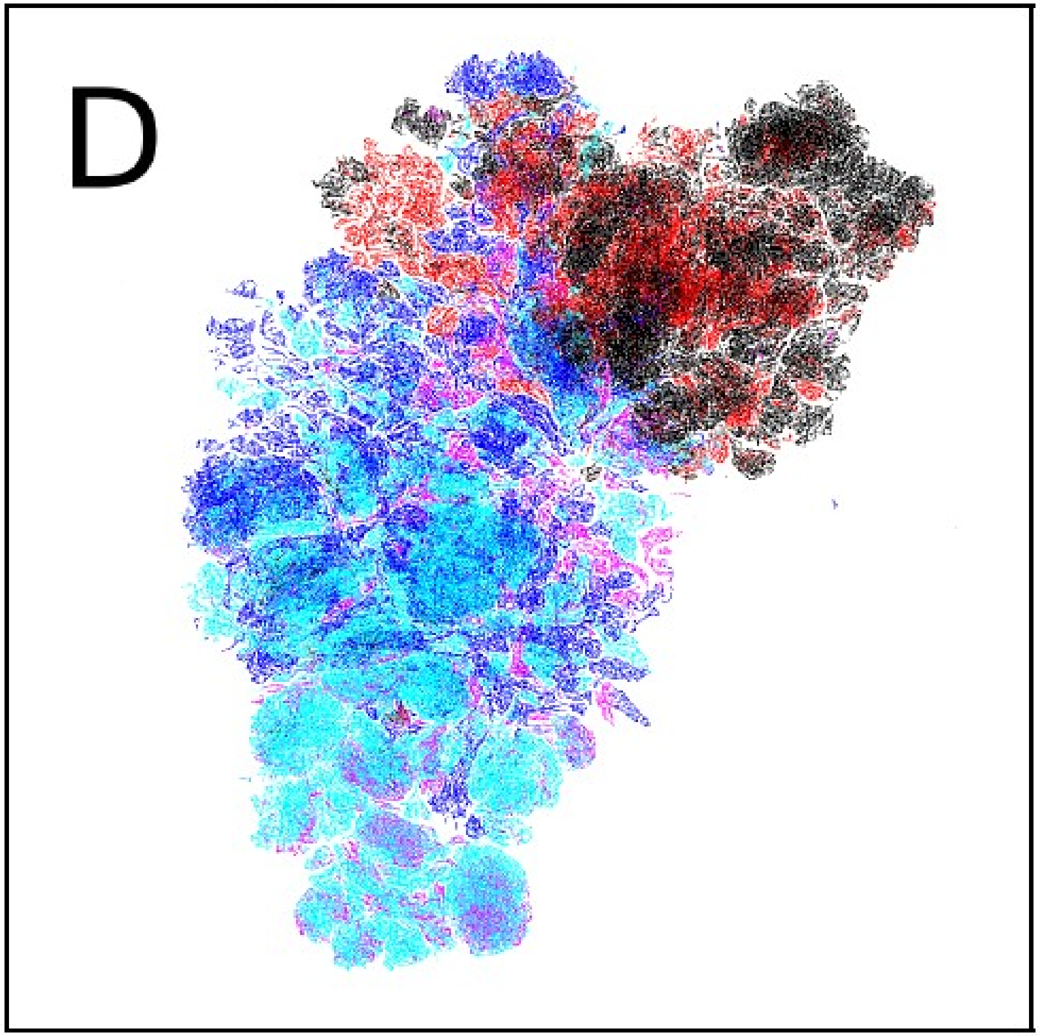
S-SNE

NeRV

Image from ref. [7]

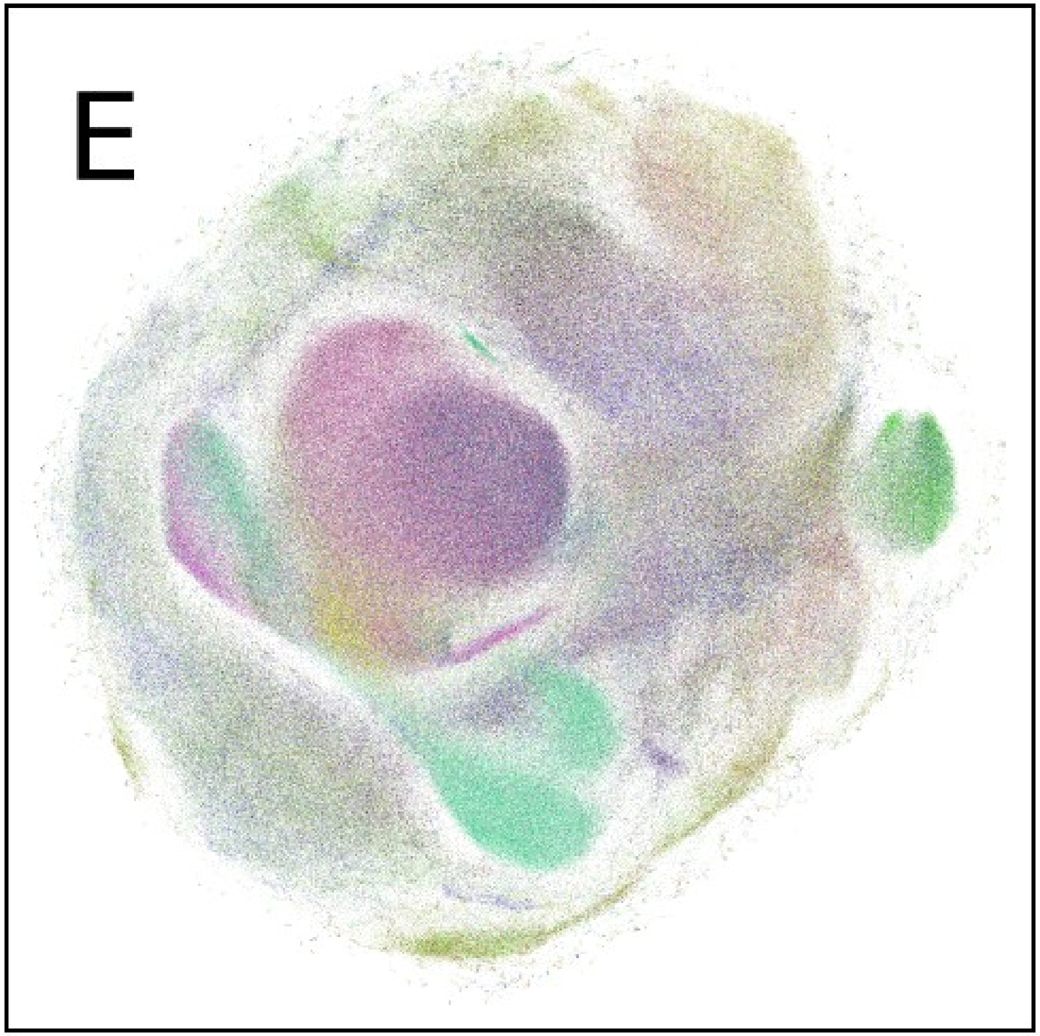
Approach 2 (Barnes-Hut for SNE/NeRV)

Covertype, 581K, 46 hours



S-SNE

TIMIT, 1.3M, 33 hours



t-SNE

Image from ref. [7]

Part 4: Quality assessment of visualization

Quality Assessment

As we discussed on the previous lecture, the purpose of visualization could be to “generate insight” about data. Most methods use abstract criteria to optimize their visualizations; NeRV optimizes the visualization for a concrete retrieval task.

- I argue that if the actual task the visualization will be used for is known, then performance in that task is the ultimate measure of quality.

- However, often the precise task is too high-level to be specified exactly, or the visualization may be used for several tasks. It is then worthwhile to measure quality of visualization by several indirect measures of quality that have been proposed.

Quality Assessment

Subjective qualitative appeal: whether the plots “look good”. May be hard to judge if good-looking plots are truthful. Not quantitative: may be hard to compare severity of visual artefacts.

Abstract internal measures: internal abstract cost funtions of different methods. Typically each method will be good on its own function. From abstract functions it's hard to say why one function is more relevant than another. Useful for **convergence analysis**, but not for **comparing methods**.

Task-based internal measures: cost functions of e.g. NeRV (information retrieval). Distance preservation can be considered a task-based measure for a task of measuring distances. Typically again each method is good on its own function. Now at least the tasks are understandable, so one could know which tasks the analyst will perform.

Quality Assessment

Reconstruction error:

$$E_{\mathrm{rec}} = \mathrm{E}\{(\xi - \mathscr{M}^{-1}(\mathscr{M}(\xi)))^2\}$$

Assumes that original coordinates are the important thing to reconstruct. Requires an inverse mapping, many methods only provide output locations of training points.
Reconstruction error is essentially the internal cost function of e.g. autoencoder neural networks.

Classification error on the display:
Use some classifier (e.g. k-nearest neighbor classifier) to measure how well classes are separated on the display.
Requires labeled data. Can be a good measure if the class labels were not used in training.

Note: Some errors on this slide were corrected after the lecture recording.

Quality Assessment

Many of the measures can be analyzed based on a co-ranking matrix:

rank of j as a neighbor of i in the input space

rank of j as a neighbor of i in the output space

$$\mathbf{Q} = [q_{kl}]_{1 \leqslant k,l \leqslant N-1} \quad \text{with} \quad q_{kl} = |\{(i,j) : \rho_{ij} = k \text{ and } r_{ij} = l\}|$$

each element is a count of how many neighbors have rank k in the input space and l in the output space. Computing this takes O(N^2 log(N)) time.

Rank error: $\rho_{ij} - r_{ij}$ positive-->intrusion, negative->extrusion (cf. false neighbor) (cf. missed neighbor)

K-intrusion/extrusion: when original/output rank is less than K

Note: Some errors on this slide were corrected after the lecture recording.

Quality Assessment

Preservation of data structure: measure preservation of higher-level structural concepts in the data.
- topographic product, topographic function in SOMs

Neighborhood based measures:

- Trustworthiness and continuity (T&C): essentially precursors to the information retrieval measures in NeRV.

$$M_T(K) = 1 - \frac{2}{G_K}\sum_{i=1}^{N}\sum_{j\in n_i^K \setminus v_i^K}(\rho_{ij} - K) = 1 - \frac{2}{G_K}\sum_{(k,l)\in \mathbb{LL}_K}(k-K)q_{kl},$$

rank of j as a neighbor of i in the input space

input-space ranks k>K and output-space ranks l<=K

$n_i^K \setminus v_i^K$ points j that are neighbors of i in the output space but not in the input space

$$M_C(K) = 1 - \frac{2}{G_K}\sum_{i=1}^{N}\sum_{j\in v_i^K \setminus n_i^K}(r_{ij} - K) = 1 - \frac{2}{G_K}\sum_{(k,l)\in \mathbb{UR}_K}(l-K)q_{kl},$$

rank of j as a neighbor of i in the output space

input-space ranks k<=K and output-space ranks l>K

$v_i^K \setminus n_i^K$ points j that are neighbors of i in the input space but not in the output space

$$G_K = \begin{cases} NK(2N-3K-1) & \text{if } K < N/2, \\ N(N-K)(N-K-1) & \text{if } K \geqslant N/2 \end{cases}$$

Note: Some errors on this slide were corrected after the lecture recording.

Quality Assessment

- mean relative rank errors (MRREs)

$$W_n(K) = \frac{1}{H_K} \sum_{i=1}^{N} \sum_{j \in n_i^K} \frac{|\rho_{ij} - r_{ij}|}{r_{ij}} = \frac{1}{H_K} \sum_{(k,l) \in \mathbb{UL}_K \cup \mathbb{LL}_K} \frac{|k-l|}{l} q_{kl},$$

output-space neighbors

output-space rank

output-space rank l <= K, any input-space rank

$$W_v(K) = \frac{1}{H_K} \sum_{i=1}^{N} \sum_{j \in v_i^K} \frac{|\rho_{ij} - r_{ij}|}{\rho_{ij}} = \frac{1}{H_K} \sum_{(k,l) \in \mathbb{UL}_K \cup \mathbb{UR}_K} \frac{|k-l|}{k} q_{kl},$$

input-space neighbors

output-space rank

input-space rank l <= K, any output-space rank

$$H_K = N \sum_{k=1}^{K} \frac{|N - 2k + 1|}{k}$$

- local continuity meta-criterion (LCMC)

$$U_{LC}(K) = \frac{1}{NK} \sum_{i=1}^{N} \left(|n_i^K \cap v_i^K| - \frac{K^2}{N-1} \right) = \frac{K}{1-N} + \frac{1}{NK} \sum_{(k,l) \in \mathbb{UL}_K} q_{kl},$$

Quality Assessment

Different criteria use different parts (shaded) of the co-ranking matrix:

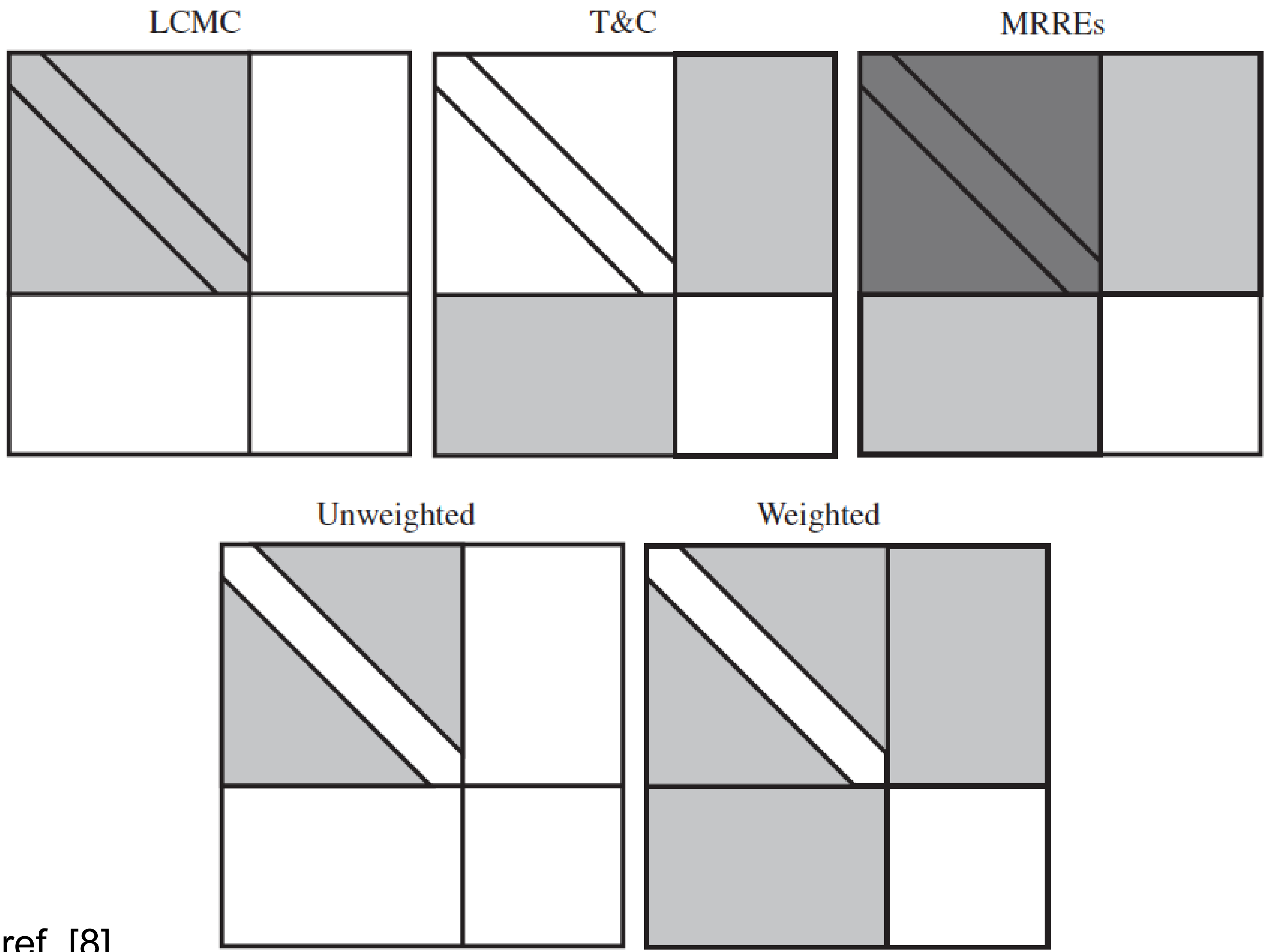


Image from ref. [8]

Quality Assessment

Some observations from artificial data -type experiments:
- NLM tends to produce intrusive plots, CCA works in an extrusive way.

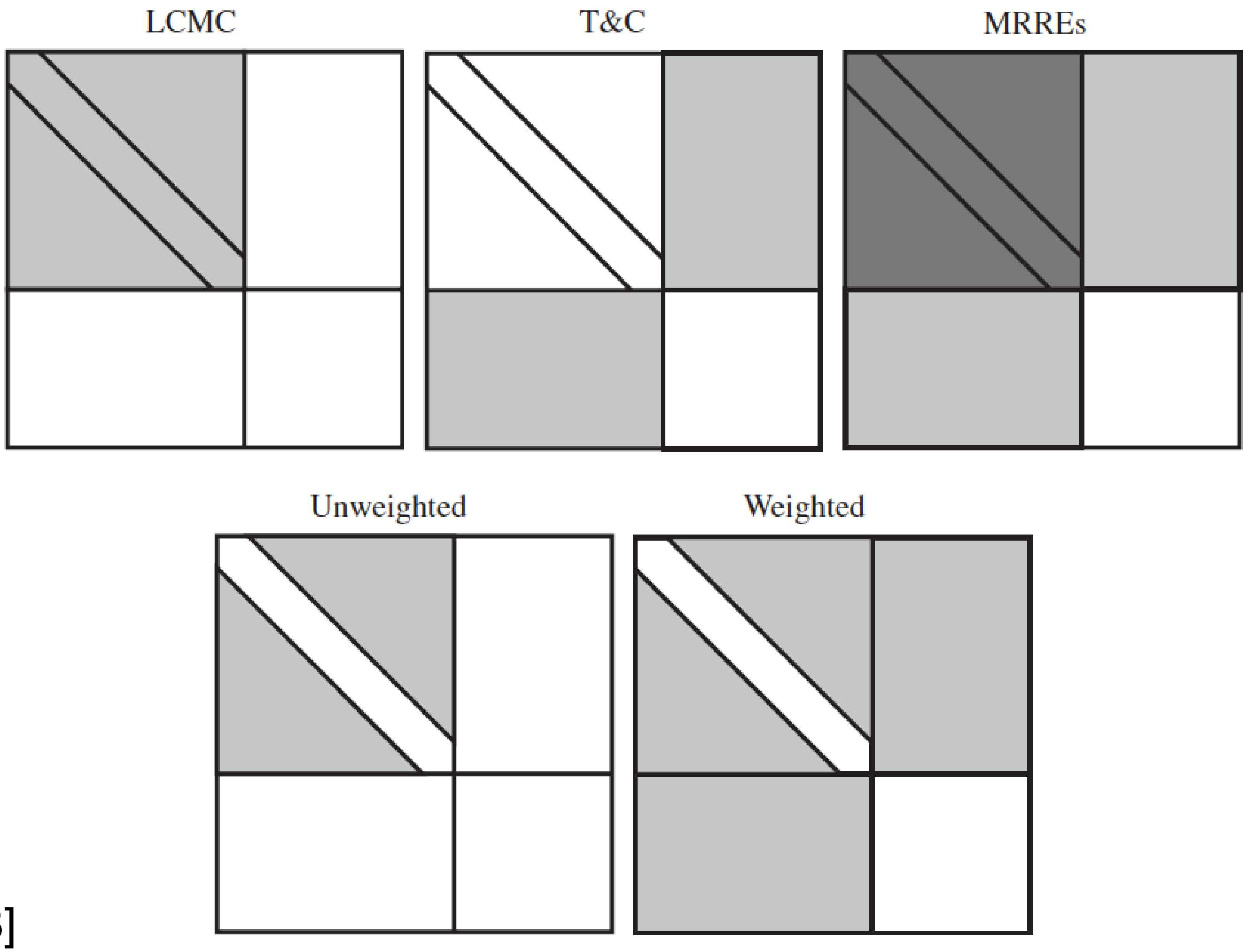


Image from ref. [8]

Quality Assessment, retrieval-based measures

LCMC focuses on true positives, T&C focus on false positives and false negatives, MRREs encompass positives and negatives

MRREs can be combined as

$$Q_{\mathrm{wNX}}^{v,w} = 1 - \frac{W_{\mathrm{N}}^{v,w}(K) + W_{\mathrm{X}}^{v,w}(K)}{2},$$

"overall quality"

$$B_{\mathrm{wNX}}^{v,w} = W_{\mathrm{N}}^{v,w}(K) - W_{\mathrm{X}}^{v,w}(K),$$

"intrusive or extrusive"

And T&C can be combined similarly

Quality Assessment, retrieval-based measures

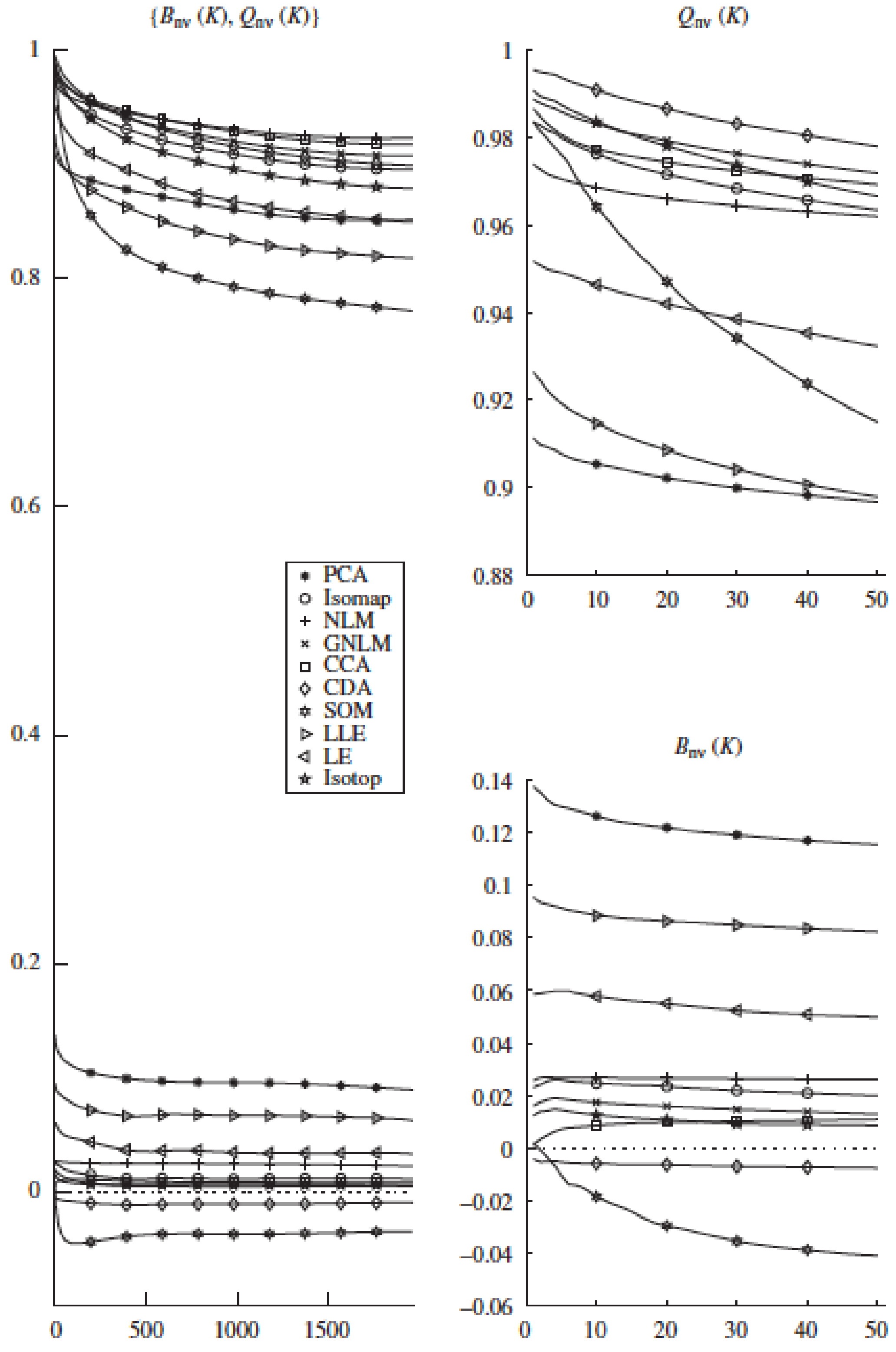


Image from ref. [8]

Quality Assessment, retrieval-based measures

Mean smoothed precision, mean smoothed recall: the two measures proposed for NeRV.

F-measure: 2*(precision*recall)/(precision+recall)
Combined measure, is low if either precision or recall is low

Precision-recall curve: rank all neighbors in order of retrieval, predicted likeliest neighbors first. Use different cutoffs to select the retrieved set, calculate precision and recall at each cutoff.
----> Curve of precision vs recall: high recall-->low precision

Rank-based mean smoothed precision, rank-based mean smoothed recall: same as the measures in NeRV, except instead of distances we use ranks of distances. Less sensitive to the precise distances.

References (pictures were from these papers)

1. Jaakko Peltonen and Samuel Kaski. **Generative Modeling for Maximizing Precision and Recall in Information Visualization.** In Proceedings of AISTATS 2011.

2. Jaakko Peltonen and Samuel Kaski. **Discriminative Components of Data**. IEEE Transactions on Neural Networks, 2005.

3. Jaakko Peltonen, Arto Klami, and Samuel Kaski. **Improved Learning of Riemannian Metrics for Exploratory Data Analysis**. Neural Networks, 2004.

4. Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. **Information retrieval perspective to nonlinear dimensionality reduction for data visualization**. Journal of Machine Learning Research, 2010.

5. Jaakko Peltonen. **Visualization by Linear Projections as Information Retrieval**. In Proceedings of WSOM 2009.

6. Jaakko Peltonen and Konstantinos Georgatzis. **Efficient Optimization for Data Visualization as an Information Retrieval Task.** In Proceedings of MLSP 2012.

7. Zhirong Yang, Jaakko Peltonen, and Samuel Kaski. **Scalable Optimization of Neighbor Embedding for Visualization.** In Proceedings of ICML 2013

8. John A. Lee and Michel Verleysen. **Quality assessment of dimensionality reduction: Rank-based criteria**. Neurocomputing, 2009