EuroVis 2017
Machine Learning Methods in Visualization for Big Data 2017
12 June 2017, Barcelona, Spain

Supervised dimensionality reduction

Jaakko Peltonen[3]

3 University of Tampere, Aalto University

Outline

- Supervised methods (regression and classification) and supervised dimensionality reduction
- Methods for text data
- Methods for dynamic data
- HCI for dimensionality reduction
- Data lab: bring your own data to be analyzed with help from the presenters

Supervised dimensionality reduction

- Supervised dimensionality reduction has two main purposes: support further automated prediction/classification/regression tools, or support human analysis
- For **automated tools**, dimensionality reduction (supervised or not) leaves out information. Why use it?
- Dimensionality reduction could focus on relevant variables, leave out noise/outliers/distortions, and further methods may be faster/more robust with smaller dimensionality.
- For **human analysis**, the motivation is to discover novelties in data, using existing outputs/categories/annotations/expert knowledge as supervision.
- If we already have annotations, what remains to be discovered? Relationships of features to annotations, relationships between annotations, features that are not captured by existing annotation.

Supervised dimensionality reduction

- Two main approaches to supervised dimensionality reduction:
  1. Directly propose a **dimensionality reduction criterion** that makes use of annotation. Example: linear discriminant analysis.
  2. Use a dimensionality reduction method that relies on some essential statistics of data, and learn these statistics in a supervised way using the annotation.
     Typical example: **distance-based** dimensionality reduction can be done using a **supervised metric** learned from annotation.

- A danger in both approaches: if only few data with annotation are available, conclusions (annotation-feature relationships, etc.) can be based on artifacts of individual samples.

Methods that directly optimize supervised criteria

Linear discriminant analysis 1

- PCA is an unsupervised method (class information is not usually used).
- Linear Discriminant Analysis (LDA) is a supervised method for dimensionality reduction in classication problems.
- As PCA, LDA can be accomplished with standard matrix algebra (eigenvalue decompositions etc.). This makes it relatively simple and useful.
- PCA is a good general purpose dimensionality reduction method, LDA is a good alternative if we want to optimize the **separability of classes** in a specific classication task, and are happy with a dimensionality of less than the number of classes ($k < K$).

Linear discriminant analysis 2

- Originally introduced for two-class problems, idea: transform the data so that the classes (c_1, c_2) are separated as much as possible
- Within-class scatter matrix $\hat{\Sigma}_w = \sum_i \sum_{\mathbf{x} \in c_i} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)'$

  where $\bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in c_i} \mathbf{x}$ and m_i is the number of samples in c_i
- Between-class scatter matrix $\hat{\Sigma}_b = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)'$

- Optimize projection matrix Φ to maximize ratio of between-class to within-class scatter: $\mathcal{J}(\Phi) = \frac{|\Phi^T \hat{\Sigma}_b \Phi|}{|\Phi^T \hat{\Sigma}_w \Phi|}$
- Optimized matrix Φ given by eigenvectors of $\hat{\Sigma}_w^{-1} \hat{\Sigma}_b$

Linear discriminant analysis 3

- Multi-class case is similar:
- Within-class scatter matrix $\hat{\Sigma}_w = \sum_{i=1}^{n} \sum_{\mathbf{x} \in c_i} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)'$

  where $\bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in c_i} \mathbf{x}$ and m_i is the number of samples in c_i
- Between-class scatter matrix

$$\hat{\Sigma}_b = \sum_{i=1}^{n} m_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})'$$

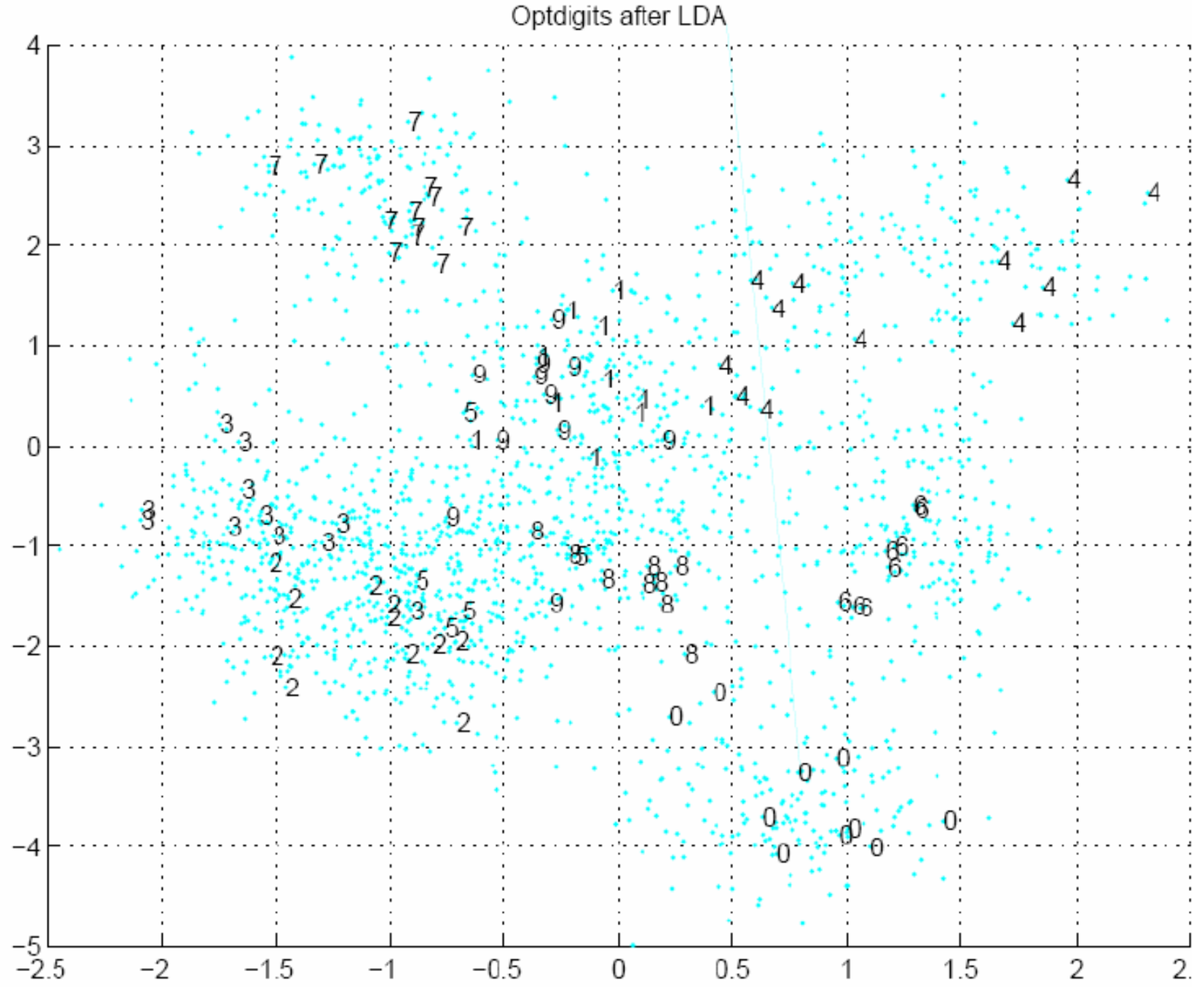
- Optimize projection matrix Φ to maximize ratio of between-class to within-class scatter: $\mathcal{J}(\Phi) = \frac{|\Phi^T \hat{\Sigma}_b \Phi|}{|\Phi^T \hat{\Sigma}_w \Phi|}$
- Optimized matrix Φ given by solving the generalized eigenvalue problem $\hat{\Sigma}_b \Phi = \lambda \hat{\Sigma}_w \Phi$

Linear discriminant analysis 4

- The rank of the within-class scatter matrix is upper-bounded by m-n, and the rank of the between-class scatter matrix is upper bounded by n-1. ---> LDA cannot give more projection directions than n-1 (number of classes - 1).
- Classification in the low-dimensional space can be done e.g. by finding the nearest class centroid of a new point
- LDA projection maximizes mean-squared distance between classes in the projected space, not the same as minimizing classification error. Pairs of classes that are far apart dominate the LDA criterion, and can leave overlap between the remaining classes.

Linear discriminant analysis 5

- OptDigits example:



Supervised Neighbor Retrieval Visualizer by **feature constraint**

Neighbor Retrieval Visualizer: nonlinear dimensionality reduction method, optimizes information retrieval performance of original data neighbors from the display. Minimizes misses and false neighbors, the user can set the desired tradeoff. Stochastic neighbor embedding is one end of the tradeoff. (J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. J. Machine Learning Research, 2010.)

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

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

Severity of missed neighbors, minimizing it maximizes a generalization of **recall**

Severity of false neighbors, minimizing it maximizes a generalization of **precision**

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

Minimize with respect to output coordinates y_i

Supervised Neighbor Retrieval Visualizer by **feature constraint**

Neighbor Retrieval Visualizer: nonlinear dimensionality reduction method, optimizes information retrieval performance of original data neighbors from the display. Minimizes misses and false neighbors, the user can set the desired tradeoff. Stochastic neighbor embedding is one end of the tradeoff. (J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information retrieval perspective to nonlinear dimensionality reduction for data visualization. J. Machine Learning Research, 2010.)

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

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

Severity of missed neighbors, minimizing it maximizes a generalization of **recall**

Severity of false neighbors, minimizing it maximizes a generalization of **precision**

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

Minimize with respect to output coordinates y_i

The optimization can be done with a mapping constraint, such as linear mapping **$\mathbf{y} = \mathbf{w}^T\mathbf{x}$.** (J. Peltonen. Visualization by Linear Projections as Information Retrieval. In proc. WSOM 2009.) **Allows supervision: neighborhoods from supervised annotation, constraint from unsupervised inputs.**

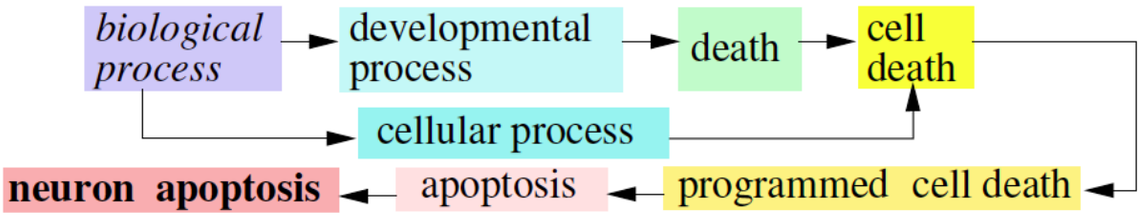
Supervised Neighbor Retrieval Visualizer by **feature constraint**

(Peltonen, Aidos, Gehlenborg, Brazma, and Kaski. An information retrieval perspective on visualization of gene expression data with ontological annotation. In proc. ICASSP 2010)

Bioinformatics case study: you have measurements and annotations. Then...

Expression distance: any suitable distance between measured activity, e.g. simply euclidean distance between gene expression profiles as vectors, or any more advanced distance (e.g. time series distance messures if the profiles are over time).

Ontology distance:
Given ontology annotations of two genes, compute Jaccard distance between their **true paths** (paths from annotations to ontology root)



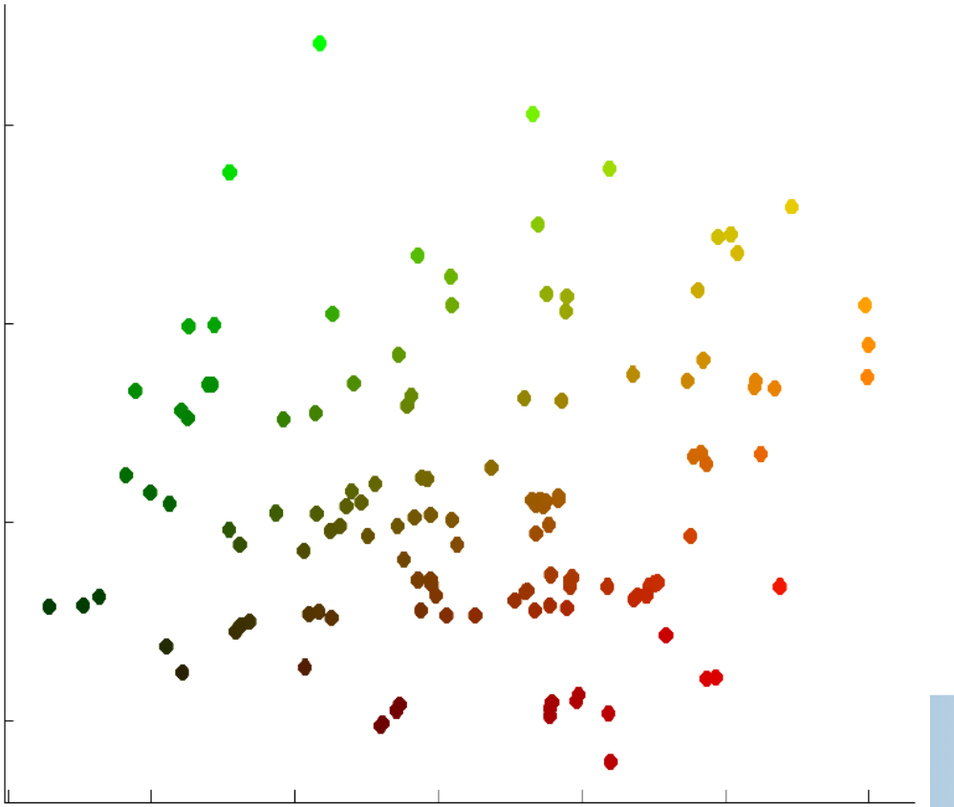
one of the 19 GO true paths for human gene AIFM1

$$J(S_i, S_j) = (|S_i \cup S_j| - |S_i \cap S_j|)/|S_i \cup S_j|$$

Supervised Neighbor Retrieval Visualizer by **feature constraint**

Example: Yeast genes significantly expressed in a study of 300 comparisons of mutant yeast strains to wild-type (normal) strain

To visualize regularities in annotation, give the Jaccard distances as input to NeRV ---> visualizes which genes are neighbors in terms of annotation.

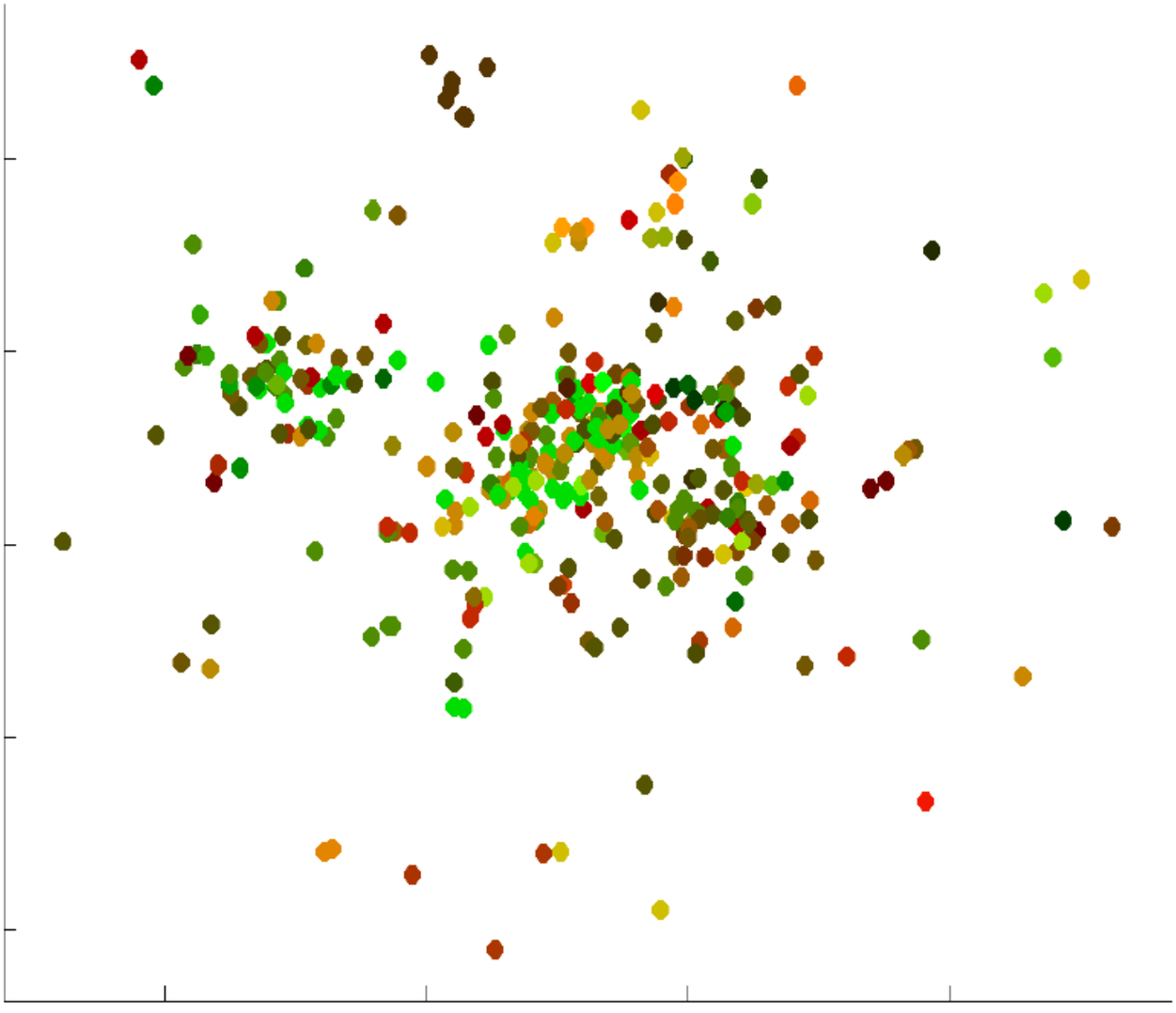


Supervised Neighbor Retrieval Visualizer by **feature constraint**

To visualize regularities in gene expression, give the distances of gene expression profiles as input to NeRV ---> visualizes which genes are neighbors in terms of gene expression.

300 comparisons of strains ---> 300 dim. gene expression profile for each gene.

Visualize similarities of expression. **Color** by ontology similarity.

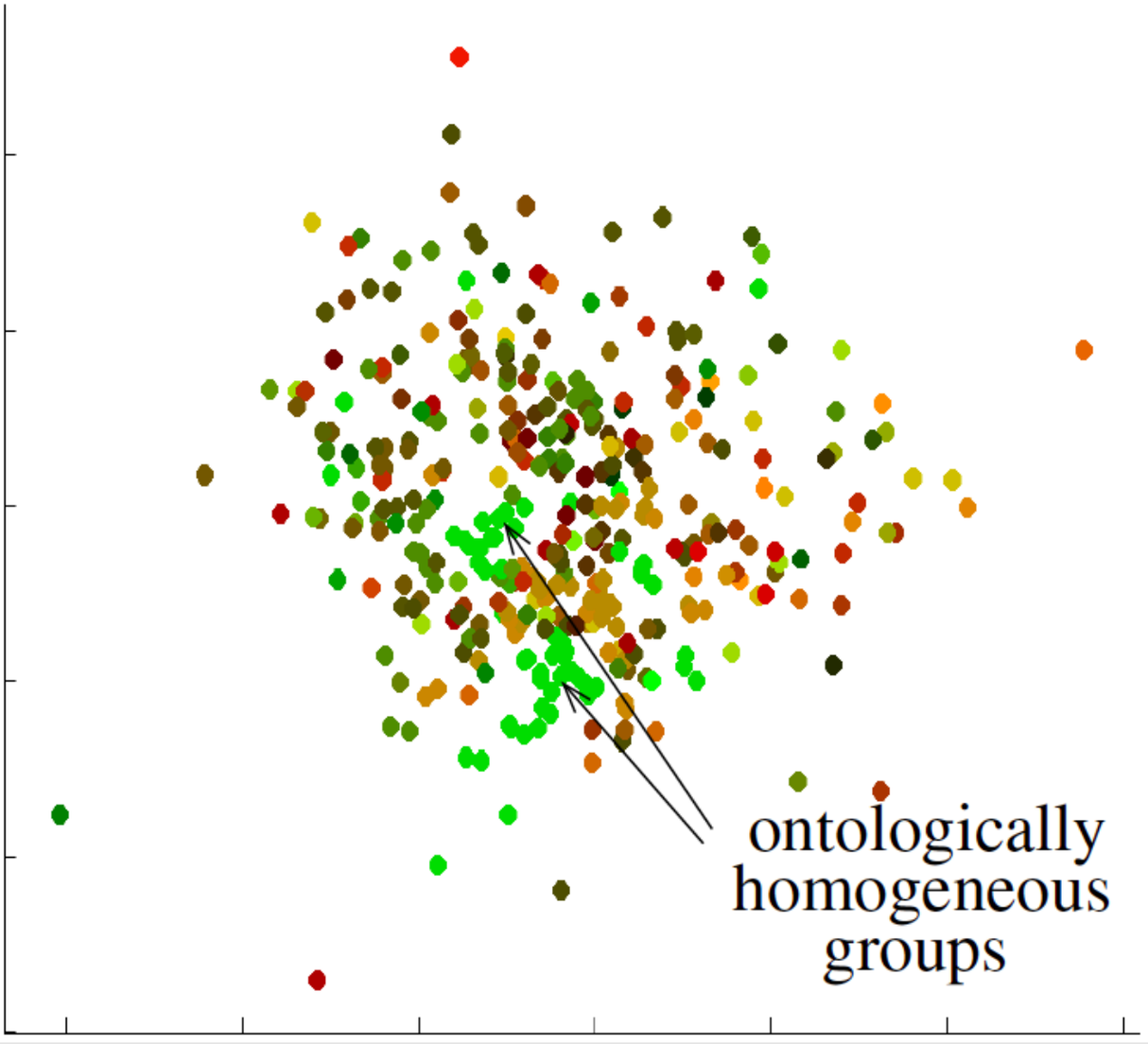


Supervised Neighbor Retrieval Visualizer by **feature constraint**

To visualize correspondences of gene expression similarity and ontology similarity, give the distances of gene expression profiles as inputs to linear NeRV, and give ontology distances as targets --->

Finds a subspace of expression profiles,

so that neighbors in the subspace best match neighbors in the ontology.



Methods that perform dimensionality reduction in supervised metrics

Metric learning

- Metric learning means learning a better metric (better distance function) between the original high-dimensional data than the original metric that one starts with
- “Better” can mean many things, for example better separation between classes of data, better correspondence to some known properties, etc.
- Metric learning is not dimensionality reduction by itself, but can be used as part of dimensionality reduction.

Metric learning – easy to apply

- Many statistical methods rely on distances as much or more than they do on feature values:
  - nearest neighbor regression/classification uses distances to find the nearest neighbors
  - many clustering approaches such as k-means use distances as part of the algorithm to optimize the clustering
  - in information retrieval, “best” results are often the ones most similar to the query according to some distance
- Dimensionality reduction methods such as multidimensional scaling, Sammon mapping, Self-organizing maps, Stochastic Neigbor Embedding, Neighbor Retrieval Visualizer, and others are distance-based
- In many cases distances from a new distance function can be just plugged in to dimensionality reduction methods. (In some cases more is needed.)

Types of metrics

- Metrics can be simple or complicated functions of data features.

- The Euclidean metric is a simple squared sum of coordinate differences.

$$d^2(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||^2 = (\mathbf{x} - \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}') = \sum_{k=1}^{d} (x_k - x'_k)^2$$

- Norm-independent distance is related to cosine similarity:

$$d^2(\mathbf{x}, \mathbf{x}') = \left\| \frac{\mathbf{x}}{||\mathbf{x}||} - \frac{\mathbf{x}'}{||\mathbf{x}'||} \right\|^2 = 2 - 2\frac{\mathbf{x}^\top \mathbf{x}'}{||\mathbf{x}|| \cdot ||\mathbf{x}'||} = 2 - 2\cos(\mathbf{x}, \mathbf{x}')$$

Types of metrics

- A Mahalanobis metric is described by a positive semidefinite metric matrix **A**:

$$d^2_{\mathbf{A}}(\mathbf{x},\mathbf{x}') = (\mathbf{x}-\mathbf{x}')^\top \mathbf{A}(\mathbf{x}-\mathbf{x}') = \sum_{k=1}^{d}\sum_{l=1}^{d}(x_k - x'_k)A_{kl}(x_l - x'_l)$$

- If **A** is diagonal the metric is just feature weighting:

$$d^2_{\mathbf{A}}(\mathbf{x},\mathbf{x}') = \sum_{k=1}^{d}(x_k - x'_k)^2 A_{kk} = \sum_{k=1}^{d}(\sqrt{A_{kk}}(x_k - x'_k))^2$$

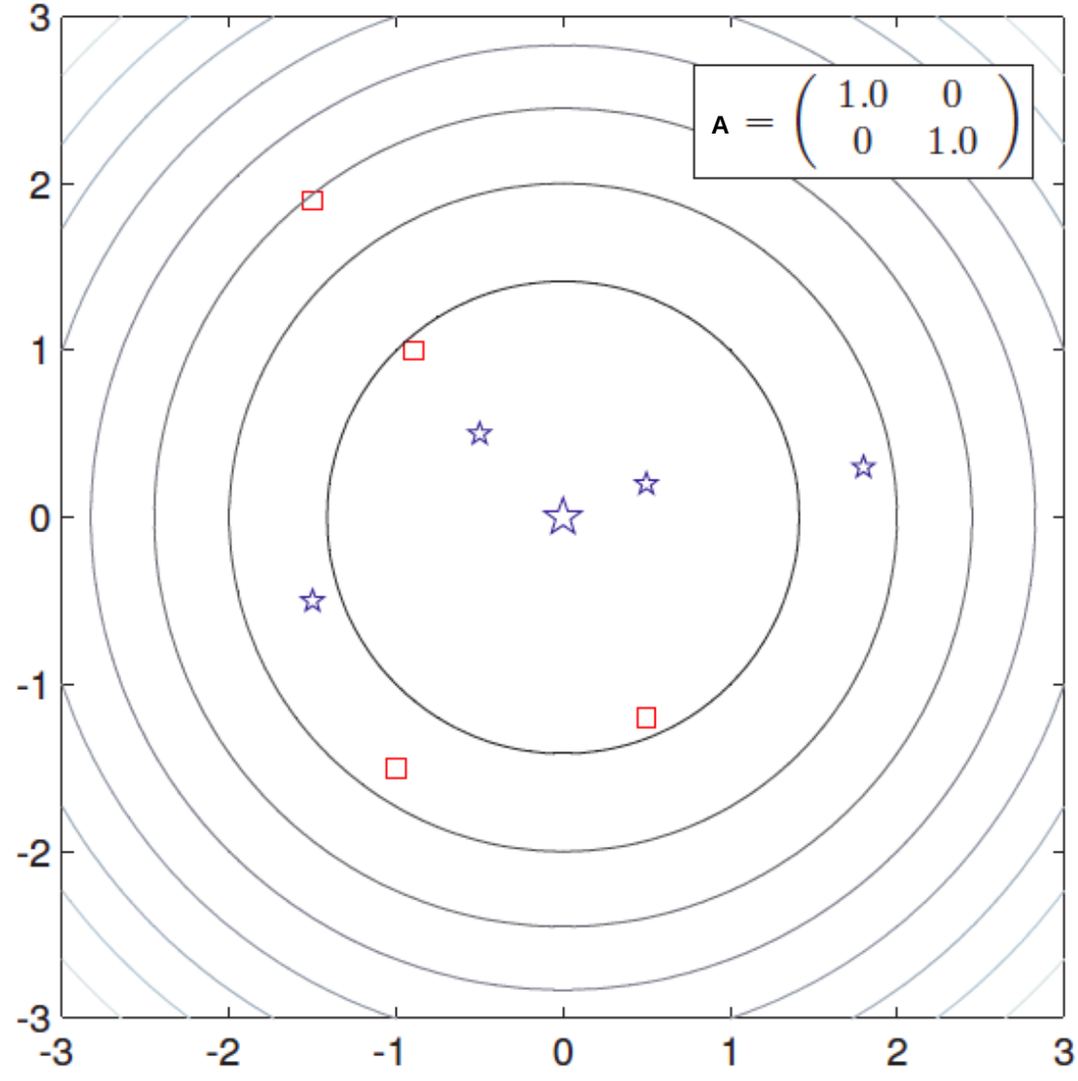
- The traditional Mahalanobis metric uses $\mathbf{A}=\mathbf{C}^{-1}$ where **C** is the covariance matrix of the data. This metric appears inside the exponential term of a multidimensional Gaussian density function:

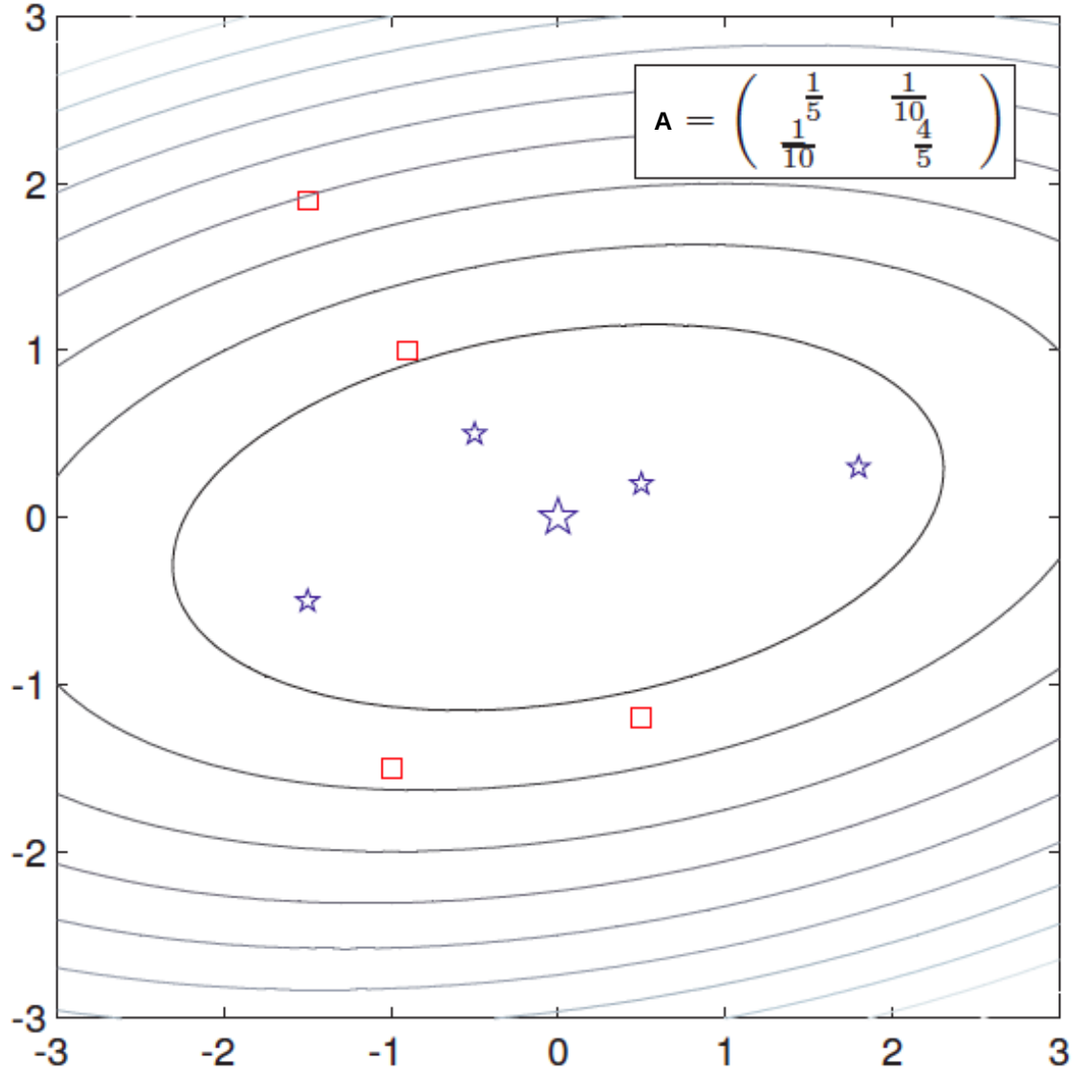
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sqrt{|\mathbf{C}|}}\exp\left(-(\mathbf{x}-\boldsymbol{\mu})^\top \mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2\right)$$

- We call the metric with any **A** a Mahalanobis metric.

Types of metrics

- Adjusting a Mahalanobis metric can make it easier to, for example, distinguish between classes of data: assume $d(\mathbf{x},\mathbf{w}) = (\mathbf{x}-\mathbf{w})^T \mathbf{A} (\mathbf{x}-\mathbf{w})$
- A non-diagonal Mahalanobis metric can take into account not just feature importances, but importance of feature combinations





Types of metrics

- Nonlinear metrics can be described in several ways:

  Globally through an explicit transformation: For example any nonlinear transformation **y** = **f**(**x**), followed by an Euclidean or Mahalanobis metric between the transformed features.

  - Thus learning any transformation **f** for the data (followed by a traditional metric) can be seen as learning a metric for the data:

$$d^2_{\mathbf{f}}(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2)||^2 = (\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2))^\top (\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2))$$

  - The output of the transformation can be higher-dimensional or lower-dimensional than the original features
  - In particular, learning any dimensionality reduction (feature selection/feature extraction) can be seen as learning a metric where the left-out features have no effect on distance.

Types of metrics

- Nonlinear metrics can be described in several ways:

  Globally through an implicit transformation: Sometimes the transformation does not need to be known, as long as the metric between the transformed features is known.

  - **Kernel methods** like kernel PCA use **kernel functions** to compute inner products in a transformed space.
  - Valid kernel functions (so-called “Mercer kernels”) always correspond to inner products in some transformed space, even if the transformation is unknown/hard to compute.
  - Distances can be computed using kernels only: assume **f** is the unknown nonlinear function and $k_{\mathbf{f}}(\mathbf{x}_1, \mathbf{x}_2)$ is the known kernel function. Then the distance can be computed as

$$
\begin{aligned}
d^2_{\mathbf{f}}(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2)||^2 &= (\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2))^\top (\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2)) \\
&= \mathbf{f}(\mathbf{x}_1)^\top \mathbf{f}(\mathbf{x}_1) - 2\mathbf{f}(\mathbf{x}_1)^\top \mathbf{f}(\mathbf{x}_2) + \mathbf{f}(\mathbf{x}_2)^\top \mathbf{f}(\mathbf{x}_2) \\
&= k_{\mathbf{f}}(\mathbf{x}_1, \mathbf{x}_1) - 2k_{\mathbf{f}}(\mathbf{x}_1, \mathbf{x}_2) + k_{\mathbf{f}}(\mathbf{x}_2, \mathbf{x}_2)
\end{aligned}
$$

Types of metrics

- Nonlinear metrics can be described in several ways:

  Alternatively, a metric can be described locally:

  - In each very small (infinitesimally small) neighborhood N(**x**) of the feature space around point **x**, distances inside the neighborhood are described by a Mahalanobis metric with a metric matrix **A**(**x**).
  - Distances between two far-apart points $\mathbf{x}_1$, $\mathbf{x}_2$ are given by integrals of local distances: for any path from $\mathbf{x}_1$ to $\mathbf{x}_2$, the distance along the path is the integral over local distances along the path.

    The shortest path (minimal integral) defines the distance d($\mathbf{x}_1$, $\mathbf{x}_2$).

    Shortest path may be difficult to compute analytically, but can be approximated.

Supervision for metrics

- **Class labels:** some subset of training data points has a known class label, out of a set of N_C different classes. For example data points might be pictures of people, for some pictures the identity of the person is known.
- **Must-link / cannot-link constraints**: for some pairs of training data points, it is known that they should be similar or dissimilar.

$$\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ should be similar}\}$$

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ should be dissimilar}\}$$

For example video footage might contain several pictures of the same person, and the pictures can be considered “similar” even if the person's identity is unknown.
Or: in social data sets data points might be people, and some people are known to be “similar” (friends/colleagues, etc.)

- Some methods instead use **constraint triplets** as annotation:

$$\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) : \mathbf{x}_i \text{ should be more similar to } \mathbf{x}_j \text{ than } \mathbf{x}_k\}$$

Supervision for metrics

- If we have a dimensionality reduction/visualization method that works based on distances (or based on a distance function), we can often simply give the optimized distances as input.
- For example the “Sammon's Mapping in the Learning Metric” method (Sammon-L; Jaakko Peltonen, Arto Klami, and Samuel Kaski, “Improved Learning of Riemannian Metrics for Exploratory Data Analysis”) infers the supervised Learning Metric, computes pairwise distances in it, and gives them as input to a traditional Sammon's mapping algorithm.
- One could similarly give supervised distances like the Learning Metric as input to Multidimensional scaling, or Curvilinear Component Analysis.

Metric learning = dim. reduction?

- In general: if the Mahalanobis metric matrix A has an eigendecomposition **A** = **VDV**^{T}, where **D** is diagonal, then

$$d^2_{\mathbf{A}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^\top \mathbf{A}(\mathbf{x} - \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^\top \mathbf{V}\mathbf{D}\mathbf{V}^\top(\mathbf{x} - \mathbf{x}')$$
$$= \left(\mathbf{D}^{1/2}\mathbf{V}^\top(\mathbf{x} - \mathbf{x}')\right)^\top \left(\mathbf{D}^{1/2}\mathbf{V}^\top(\mathbf{x} - \mathbf{x}')\right) = ||\mathbf{D}^{1/2}\mathbf{V}^\top(\mathbf{x} - \mathbf{x}')||^2$$

  where $D^{1/2}$ is D with square roots taken from diagonal entries.

- Thus learning Mahalanobis metric corresponds to learning a linear data transformation!
- If some eigenvalues (diagonals of D) are zero, then the metric effectively performs dimensionality reduction
- Some methods learn a metric with penalties that encourage features to be left out.

Informative discriminant analysis

Method first proposed in Samuel Kaski and Jaakko Peltonen, ”Informative discriminant analysis”, in proceedings of ICML 2003. Soon after proposed in Jacob Goldberger, Sam Roweis, Geoffrey Hinton, Ruslan Salakhutdinov, “Neighbourhood components analysis”, proceedings of NIPS 2004.

Idea: assume training data have labels. Learn a Mahalanobis metric matrix **A**. Maximize log-likelihood of predicting labels of a point from its nearby neighbors in the metric. (This method is thus a maximum-likelihood method to estimate the metric.)

Suppose the density of each class can be written as a mixture of multivariate Gaussian distributions with class-dependent weights:

$$p(c, \mathbf{x}) \propto \sum_{m=1}^{M} \psi_{mc} N(\mathbf{x};\ \boldsymbol{\mu_m}, \mathbf{A}^{-1})$$

Informative discriminant analysis

The conditional probability of classes at location x is

$$p(c|\mathbf{x}) = \frac{\sum_{m=1}^{M} \psi_{mc} N(\mathbf{x};\ \boldsymbol{\mu_m}, \mathbf{A}^{-1})}{\sum_{m=1}^{M} (\sum_{c} \psi_{mc}) N(\mathbf{x};\ \boldsymbol{\mu_m}, \mathbf{A}^{-1})}$$

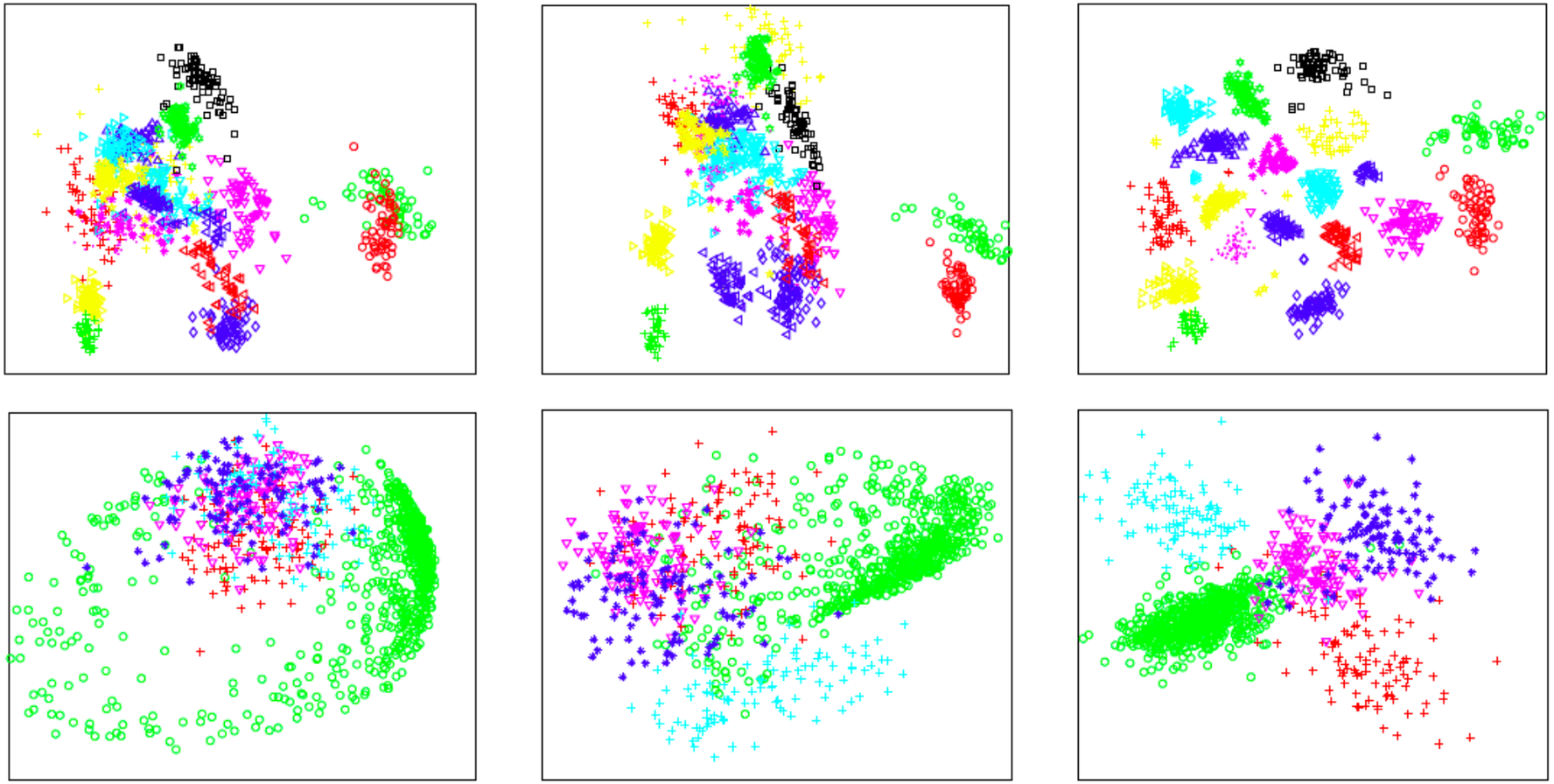
And the log-likelihood of observed class-labels c_i of points $\mathbf{x}_i$ is

$$L = \sum_{i=1}^{N} \log p(c_i|\mathbf{x}_i)$$

which can be maximized with respect to **A**

Informative discriminant analysis

IDA/NCA can also be used for dimensionality reduction by restricting the rank of the metric matrix **A**, or by directly optimizing **A** as a product of a linear projection matrix **W**, $\mathbf{A}=\mathbf{W}\mathbf{W}^T$. The probabilities and cost function are computed the same as before. The matrix W can be used to project data to lower dimensionality.



PCA LDA NCA

Pictures from J. Goldberger, S. Roweis, G. Hinton, R. Salakhutdinov, proc. NIPS 2004.

Interactive vis. by metric learning

- A metric tells which points are similar (they have small distance in the metric) and which points are dissimilar (they have large distance). The desired metric is unknown - we want to learn it.
- If we have **examples of similar point pairs** and **examples of dissimilar point pairs,** can we learn a metric from them?
- Yes! Use probabilistic modeling: given a metric, define a **probability that two points in the metric will be labeled similar** vs. dissimilar. Then optimize the metric to maximize the likelihood of the observed pairs!
- For example, use a Mahalanobis metric (matrix $\mathbf{A} = \mathbf{W}\mathbf{W}^T$) and a logistic probability:

$$p_{similar}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{1+\exp\left((\boldsymbol{x}_i - \boldsymbol{x}_j)^T A (\boldsymbol{x}_i - \boldsymbol{x}_j) - threshold\right)}$$

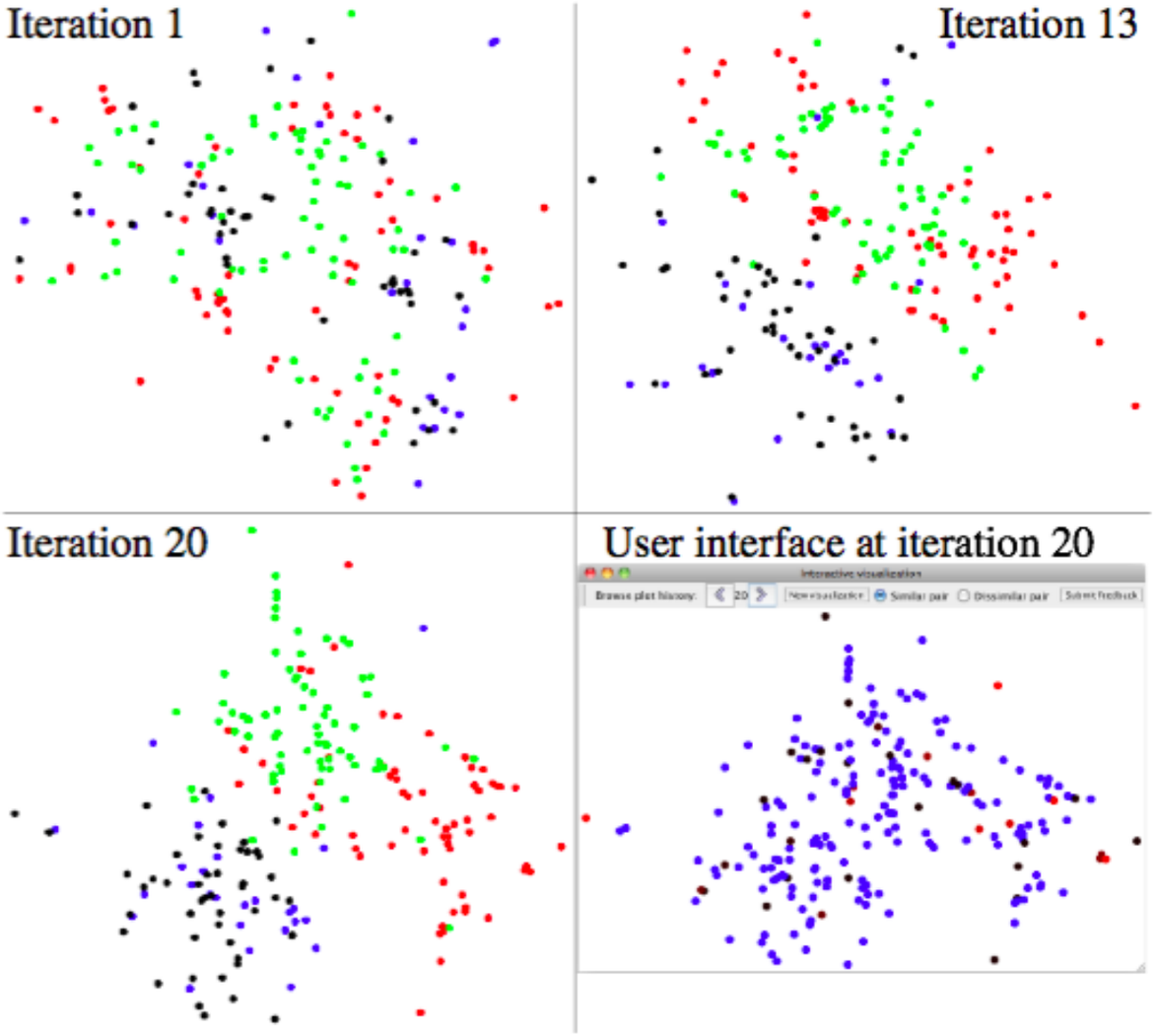
"points closer than *threshold* are probably called similar"

- Then maximize the log-likelihood of observed similarities with respect to elements of **W**, e.g. by gradient descent:

$$max_W \left[\sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in S_{similar}} \log p_{similar}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{(\boldsymbol{x}_i, \boldsymbol{x}_j) \in S_{dissimilar}} \log\left(1 - p_{similar}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \right]$$

Interactive vis. by metric learning

- This idea was used for **interactive visualization (**Peltonen et al.)
- **Experts** inspected a scatterplot of scientific documents and pointed out pairs of documents that were similar or dissimilar.
- A **metric** was learned for document features (=unigram content) based on the pointed-out pairs
- A **visualization** was created in the new metric by the Neighbor Retrieval Visualizer
- The metric and visualization **converged** to focus on important features of data.
- Resulting data organization corresponded to a hidden **ground-truth** classification of documents



Pictures from Peltonen et al., Information Retrieval Perspective to Interactive Data Visualization, in Eurovis 2013.

Local supervised "Learning metric"

- Learns a local metric from class labels of data.
- Suppose we know conditional probabilities of classes at different points of the feature space.
- Idea: Locally, distances should increase the most in directions where the class probabilities (class distribution) changes the most. If we have good local distances, we can derive a full metric from them.
- Difference between two class probability distributions can be measured by Kullback-Leibler divergence

$$D_{KL}(p||q) = \sum_c p(c) \log \frac{p(c)}{q(c)}$$

- It turns out Kullback-Leibler divergence between conditional class distributions at nearby points (**x**, **x**+d**x**) can be expressed as a squared Mahalanobis distance!

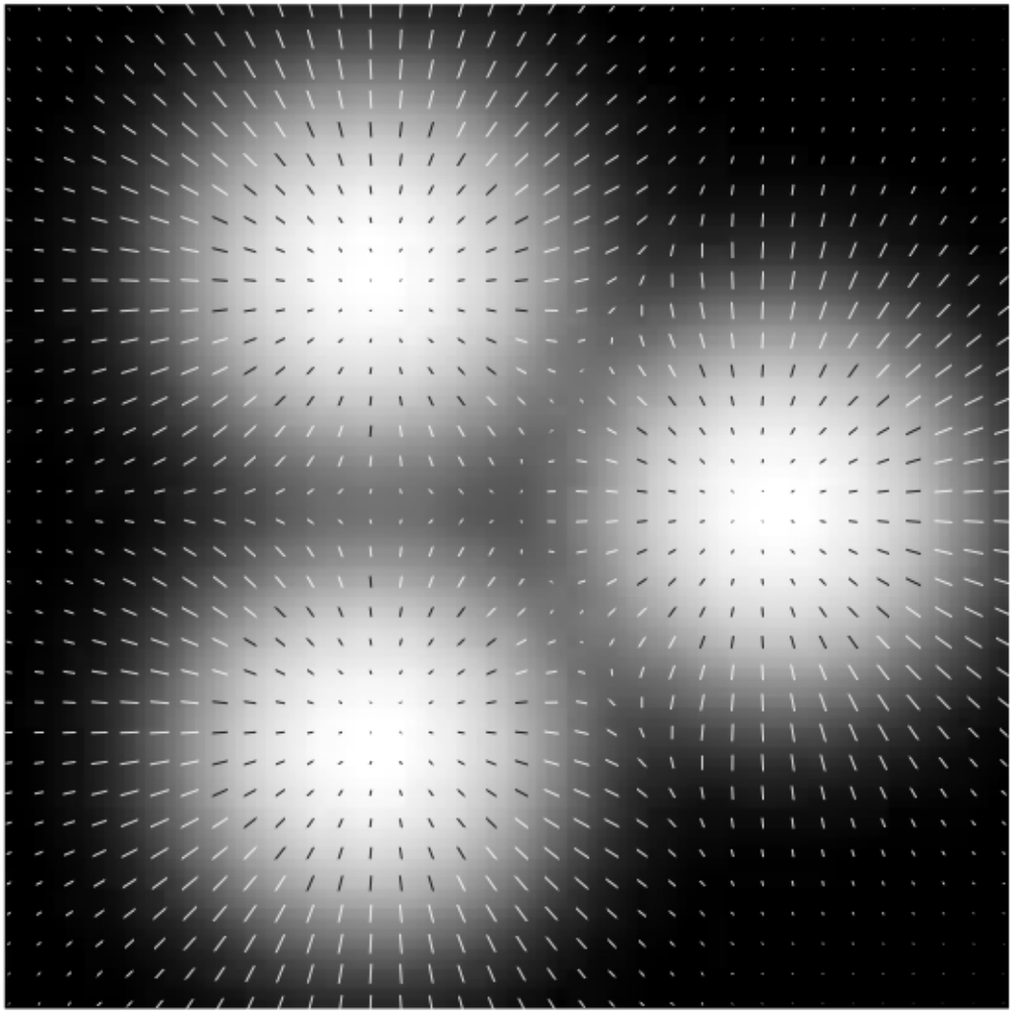
$$d_L^2(\mathbf{x}, \mathbf{x}+d\mathbf{x}) \equiv D_{KL}(p(c|\mathbf{x})||p(c|\mathbf{x}+d\mathbf{x})) = d\mathbf{x}^T \mathbf{J}(\mathbf{x}) d\mathbf{x}$$

Local supervised ”Learning metric”

- As mentioned before, local Mahalanobis distances can be extended to global distances between two points as minimal integrals of the local distances (minimal integral over possible paths between the points)
- Simple approach: just compute the local Mahalanobis from **x** to **x**+d**x**, regardless how large the difference d**x** is.
- More advanced approach: compute an approximate integral over the line connecting **x** to **x**+d**x** (e.g.divide line into 10 segments, compute local Mahalanobis over each segment).
- Even more advanced approach (similar to Isomap): compute initial distance matrix as above, then compute minimal path using other data points as possible waypoints. Can be done by Dijkstra's algorithm / Floyd's algorithm.

Local supervised ”Learning metric”

- Example of local Mahalanobis metrics



2 classes,
grayscale background:
probability of class 1.

lines: direction where
local Mahalanobis
distance increases

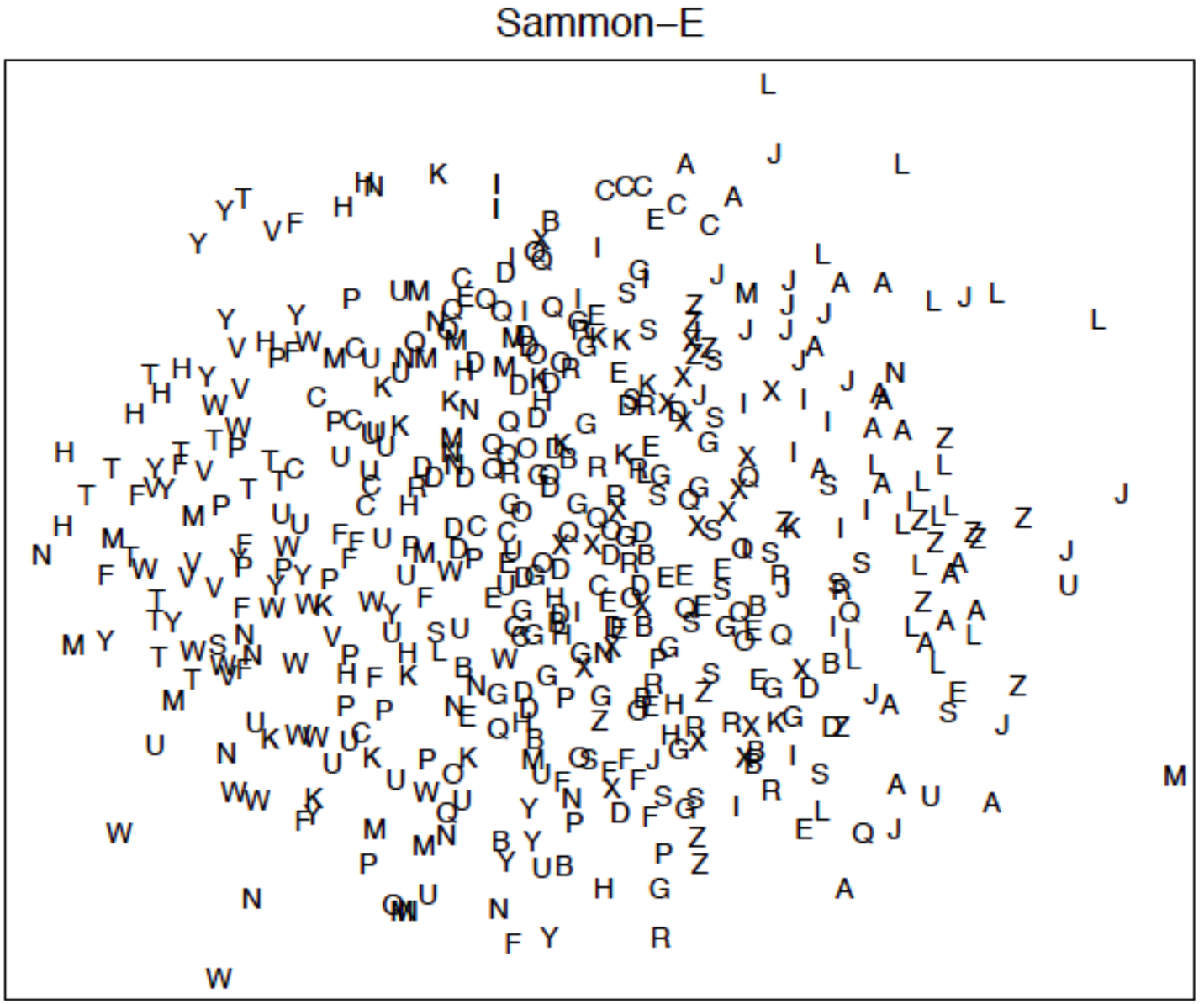
distance increases only
in directions where
class probability changes!

- The learning metric can be computed between any points (either training data or any other points; class labels not required). Thus it can be applied to any method that works based on distances.

Sammon mapping, sup. metric

Traditional Sammon's mapping on a data set of images of different letters A-Z, using various geometrical descriptors about the letter shapes as the features, with the Euclidean metric ("Sammon-E").

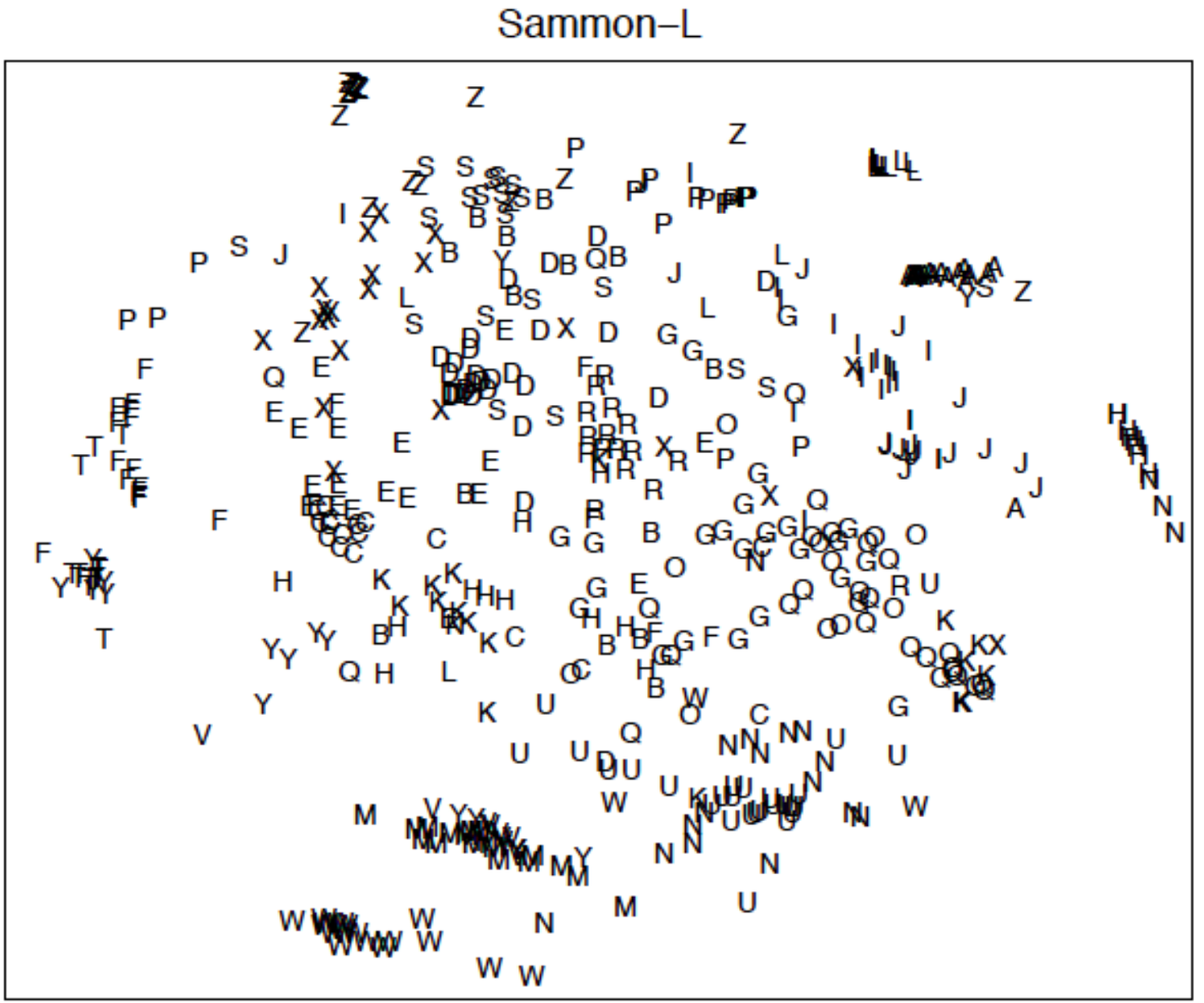
From:
Jaakko Peltonen, Arto Klami, and Samuel Kaski. Improved Learning of Riemannian Metrics for Exploratory Data Analysis. Neural Networks, vol. 17, pages 1087-1100, 2004.



Sammon mapping, sup. metric

Traditional Sammon's mapping on a data set of images of different letters A-Z, using various geometrical descriptors about the letter shapes as the features, with the local supervised Learning Metric ("Sammon-L").

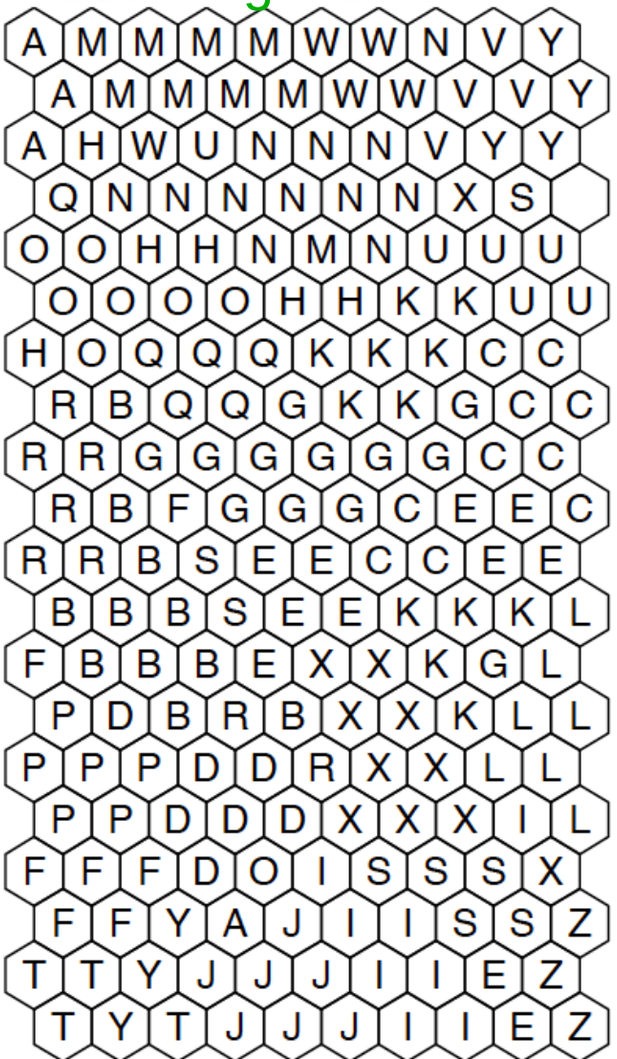
From:
Jaakko Peltonen, Arto Klami, and Samuel Kaski. Improved Learning of Riemannian Metrics for Exploratory Data Analysis. Neural Networks, vol. 17, pages 1087-1100, 2004.



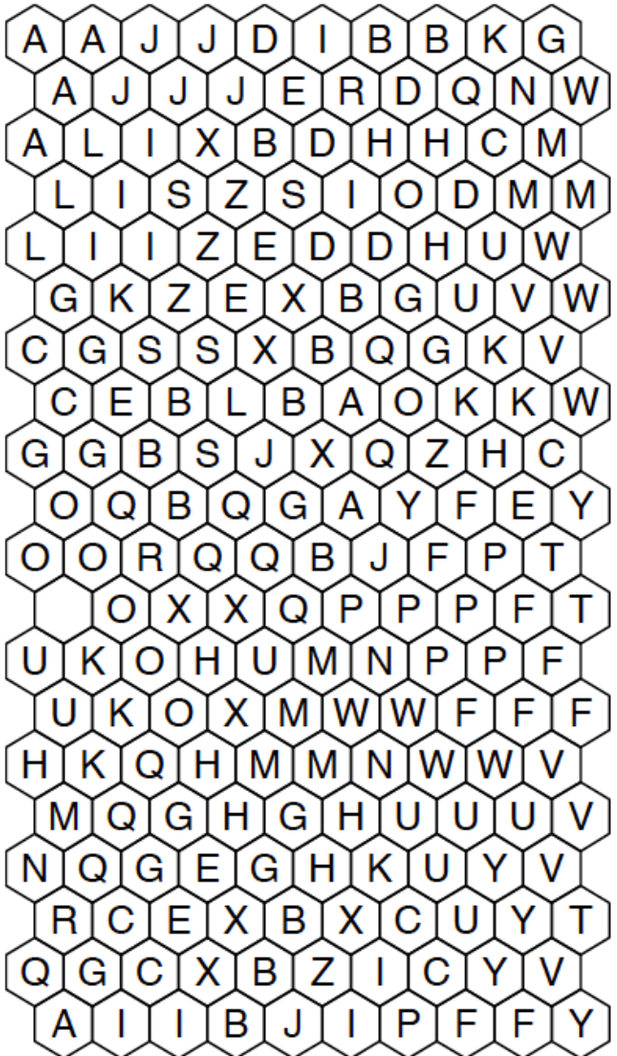
SOM in a supervised metric

Idea: at each iteration of Self-Organizing Map training, find the nearest prototype for a data point using distances in the learning metric, instead of the simple Euclidean metric. The rest of the Self-Organizing Map training (the way propotypes are adapted towards data) is the same as before.

Learning metric



Euclidean metric



Self-Organizing Map trained for letter images (features = geometric descriptions of the letters), with classes A-Z shown on the map.

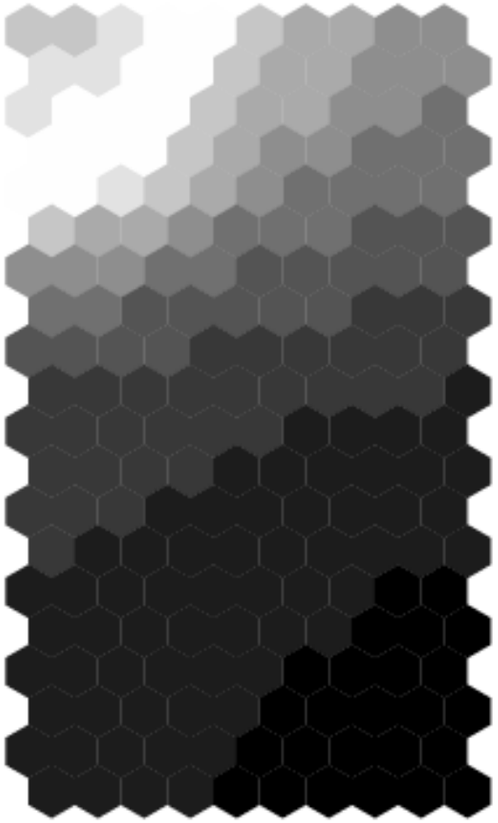
The Learning Metric leads to better class organization

From:
Samuel Kaski, Janne Sinkkonen, and Jaakko Peltonen. Bankruptcy analysis with self-organizing maps in learning metrics. IEEE Transactions on Neural Networks, 12:936-947, 2001.

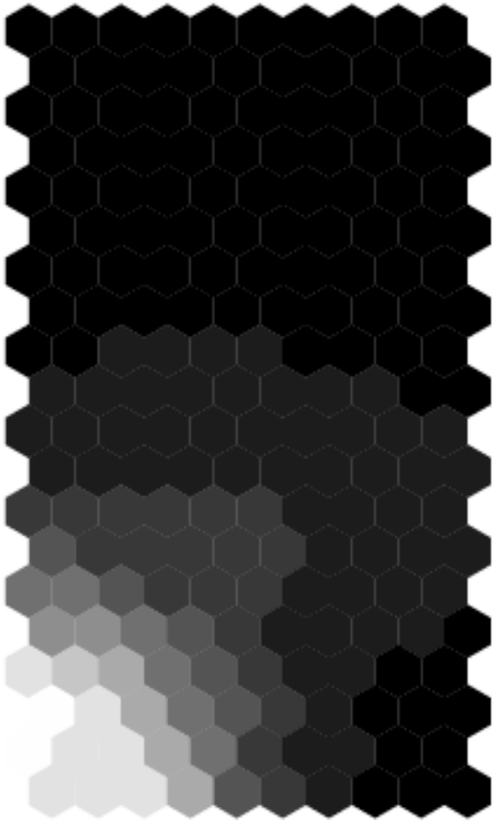
SOM in a supervised metric

Idea: at each iteration of Self-Organizing Map training, find the nearest prototype for a data point using distances in the learning metric, instead of the simple Euclidean metric. The rest of the Self-Organizing Map training (the way propotypes are adapted towards data) is the same as before.

Probability of bankruptcy

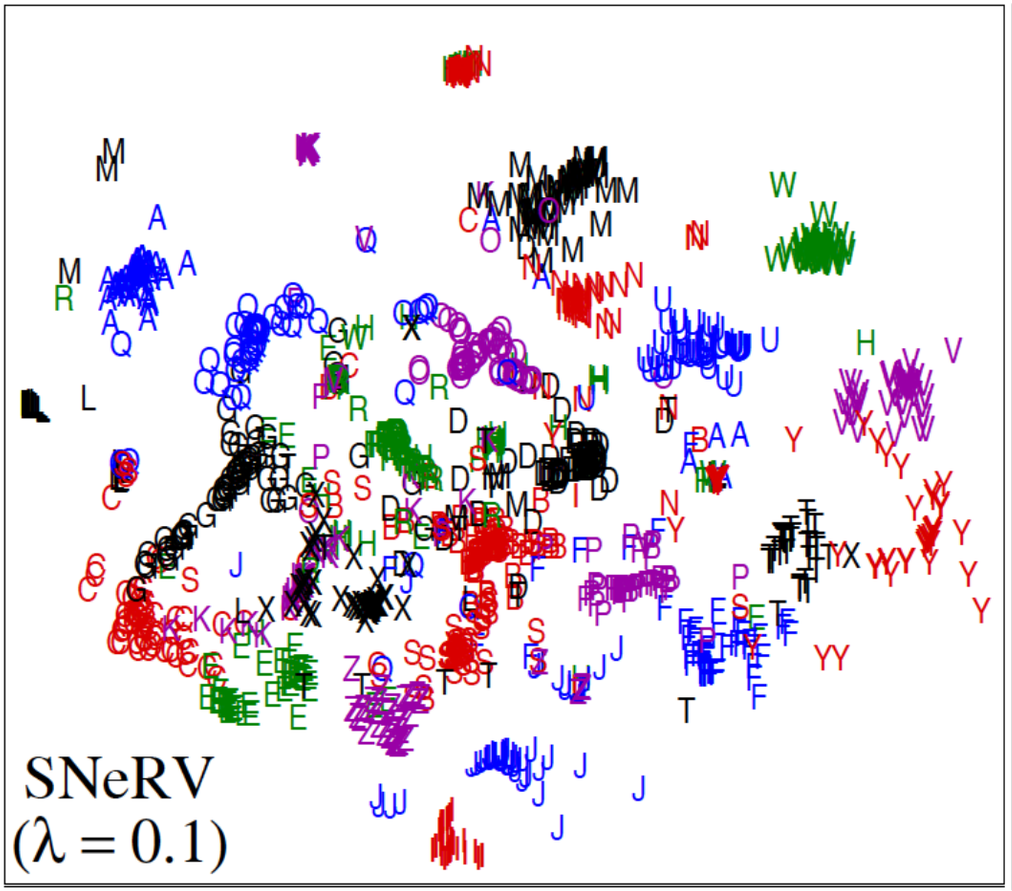


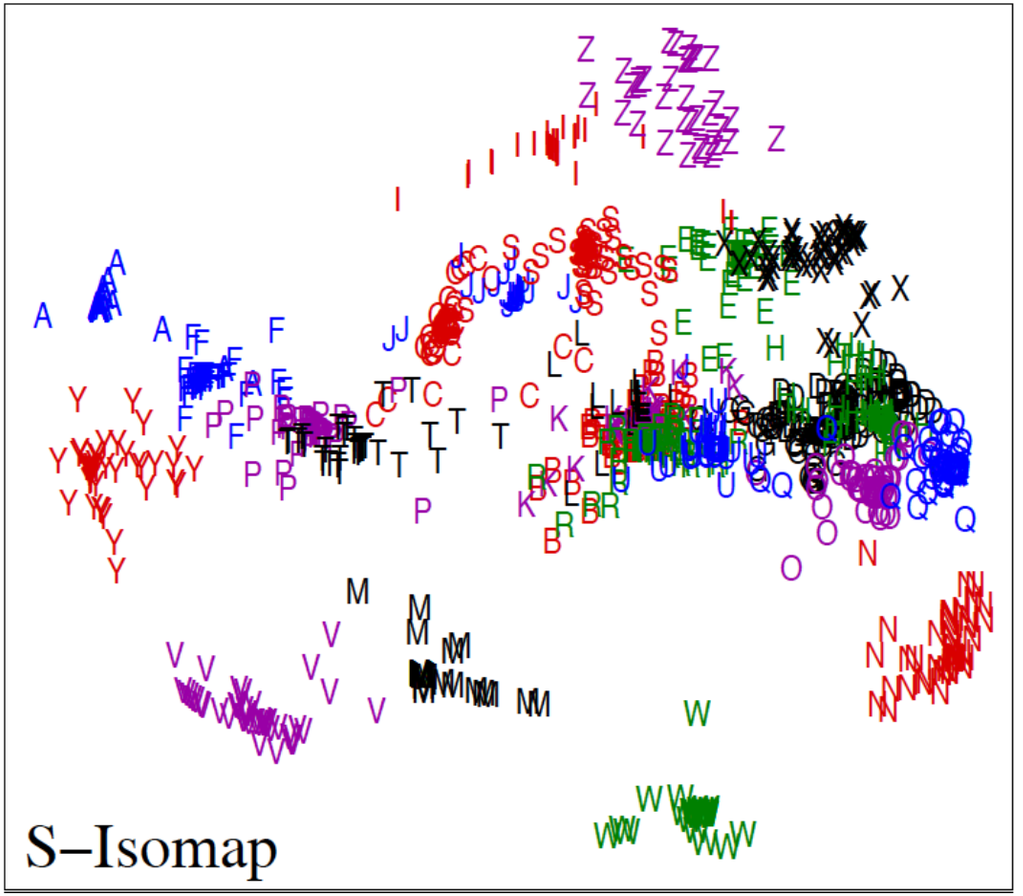
Example of a feature over the SOM



Self-Organizing Map trained for company data (features = financial indicators) in the learning metric. Classes = whether the company went bankrupt.

Neighbor Retrieval Visualizer
in a locally supervised metric

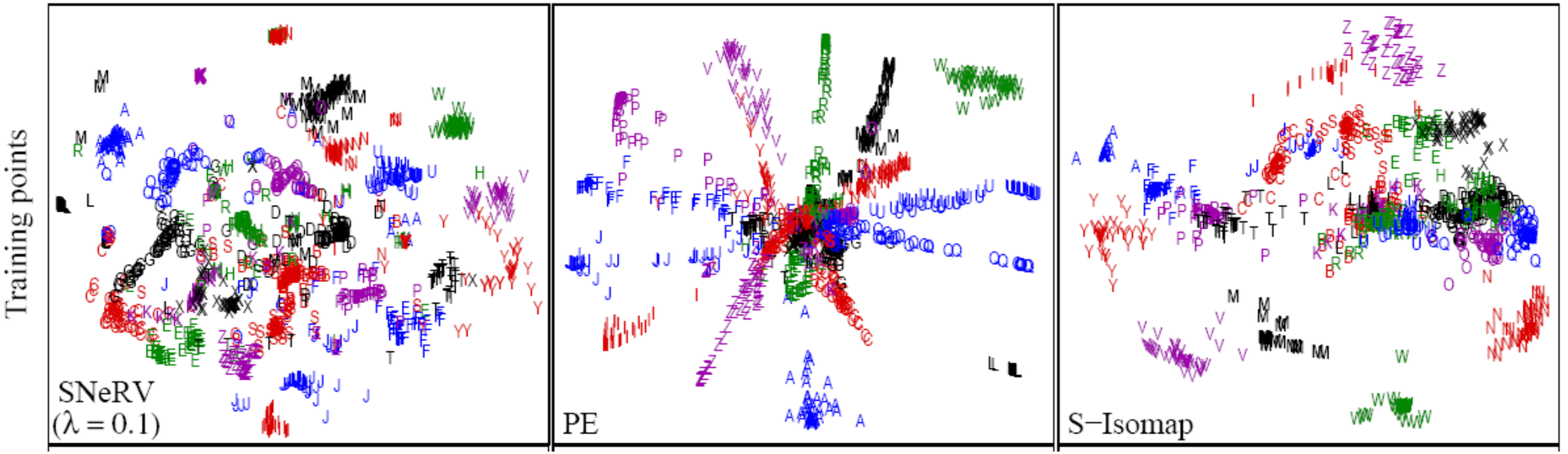




Reference: Peltonen et al., ICASSP 2009

Neighbor Retrieval Visualizer

in a locally supervised metric



Neighbor Retrieval Visualizer
in a supervised metric