

Distance Preservation - Part I

Markus Ojala

October 2, 2007

- 1 Introduction
- 2 Multidimensional scaling
 - Scalar product
 - Equivalence with PCA
 - Euclidean distance
 - Summary
- 3 Sammon's nonlinear mapping
 - Method
 - Generalizing
 - Example
- 4 Curvilinear component analysis
 - Method
 - Generalizing
 - Example
- 5 Summary

Spatial distances

Only the coordinates of the points affects the distances.

- L_p norm: $\|\mathbf{a}\|_p = \sqrt[p]{\sum_{k=1}^D |a_k|^p}$
- Minkowski distance: $d(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_p$
 - Maximum ($p = \infty$): $\|\mathbf{a} - \mathbf{b}\|_\infty = \max_{1 \leq k \leq D} |a_k - b_k|$
 - City-block ($p = 1$): $\|\mathbf{a} - \mathbf{b}\|_1 = \sum_{k=1}^D |a_k - b_k|$
 - Euclidean ($p = 2$): $\|\mathbf{a} - \mathbf{b}\|_2 = \sqrt{\sum_{k=1}^D (a_k - b_k)^2}$
- Mahalanobis norm: $\|\mathbf{a}\|_{\text{Mahalanobis}} = \mathbf{a}^T \mathbf{M}^{-1} \mathbf{a}$
 - Usually $\mathbf{M} = \mathbf{C}_{\mathbf{a}\mathbf{a}} = E\{\mathbf{a}\mathbf{a}^T\}$
 - $\|\mathbf{a}\|_{\text{Mahalanobis}} = \|\mathbf{a}\|_2 \iff \mathbf{M} = \mathbf{I}$

Classical metric multidimensional scaling (MDS)

- Preserves pairwise scalar products instead of distances.
- Assumes a simple generative model as with PCA:
 - $\mathbf{y} = \mathbf{W}\mathbf{x}$
 - “observed” variables \mathbf{y} , uncorrelated latent variables \mathbf{x}
 - variables are assumed to be centered
 - orthogonal D -by- P matrix \mathbf{W} : $\mathbf{W}^T\mathbf{W} = \mathbf{I}_P$
- N points in matrix form: $\mathbf{Y} = [\mathbf{y}(1), \dots, \mathbf{y}(N)]$
- Scalar products are known: $s_{\mathbf{y}}(i, j) = \langle \mathbf{y}(i), \mathbf{y}(j) \rangle = \mathbf{y}(i)^T \mathbf{y}(j)$
- Then $\mathbf{S} = [s_{\mathbf{y}}(i, j)]_{1 \leq i, j \leq N} = \mathbf{Y}^T \mathbf{Y} = \mathbf{X}^T \mathbf{W}^T \mathbf{W} \mathbf{X} = \mathbf{X}^T \mathbf{X}$
- Usually \mathbf{Y} and \mathbf{X} are unknown

MDS: Finding latent variables

- The eigenvalue decomposition of the Gram matrix \mathbf{S} :
$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = (\mathbf{\Lambda}^{1/2}\mathbf{U}^T)^T(\mathbf{\Lambda}^{1/2}\mathbf{U}^T)$$
- Eigenvalues sorted in descending order
- P -dimensional latent variables: $\hat{\mathbf{X}} = \mathbf{I}_{P \times N}\mathbf{\Lambda}^{1/2}\mathbf{U}^T$

Equivalence of PCA and MDS

PCA and MDS give the same projection.

- SVD: $\mathbf{Y} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T$
- Then $\hat{\mathbf{C}}_{yy} \propto \mathbf{Y}\mathbf{Y}^T = \mathbf{V}\mathbf{\Lambda}_{\text{PCA}}\mathbf{V}^T$ and $\mathbf{S} = \mathbf{Y}^T\mathbf{Y} = \mathbf{U}\mathbf{\Lambda}_{\text{MDS}}\mathbf{U}^T$,
 where $\mathbf{\Lambda}_{\text{PCA}} = \mathbf{\Sigma}\mathbf{\Sigma}^T$ and $\mathbf{\Lambda}_{\text{MDS}} = \mathbf{\Sigma}^T\mathbf{\Sigma}$
- We get:

$$\begin{aligned}\hat{\mathbf{X}}_{\text{MDS}} &= \mathbf{I}_{P \times N} \mathbf{\Lambda}_{\text{MDS}}^{1/2} \mathbf{U}^T = \mathbf{I}_{P \times N} \mathbf{\Sigma} \mathbf{U}^T \\ &= \mathbf{I}_{P \times N} \mathbf{V}^T \mathbf{V} \mathbf{\Sigma} \mathbf{U}^T = \mathbf{I}_{P \times N} \mathbf{V}^T \mathbf{Y} = \hat{\mathbf{X}}_{\text{PCA}}\end{aligned}$$

- Thus MDS minimizes the criterion:
 $E_{\text{MDS}} = \sum_{i,j=1}^N (s_y(i,j) - s_{\hat{x}}(i,j))^2$

Three ways to calculate PCA

The P -by- N matrix \mathbf{Y} is known. PCA:

① $P \ll N$:

- $\hat{\mathbf{C}}_{yy} \propto \mathbf{Y}\mathbf{Y}^T = \mathbf{V}\mathbf{\Lambda}_{\text{PCA}}\mathbf{V}^T$
- $\hat{\mathbf{X}}_{\text{PCA}} = \mathbf{I}_{P \times N}\mathbf{V}^T\mathbf{Y}$

② $P \gg N$

- $\mathbf{S} = \mathbf{Y}^T\mathbf{Y} = \mathbf{U}\mathbf{\Lambda}_{\text{MDS}}\mathbf{U}^T$
- $\hat{\mathbf{X}}_{\text{PCA}} = \hat{\mathbf{X}}_{\text{MDS}} = \mathbf{I}_{P \times N}\mathbf{\Lambda}_{\text{MDS}}^{1/2}\mathbf{U}^T$

③ $P \approx N$

- $\mathbf{Y} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T$
- $\hat{\mathbf{X}}_{\text{PCA}} = \mathbf{I}_{P \times N}\mathbf{V}^T\mathbf{Y}$

MDS with Euclidean distances

- Instead of scalar products, pairwise distances are known
- $\mathbf{D} = [d_{\mathbf{y}}^2(i, j)]_{1 \leq i, j \leq N}$
- Solution: transform distances to scalar products
- $d_{\mathbf{y}}^2(i, j) = \langle \mathbf{y}(i) - \mathbf{y}(j), \mathbf{y}(i) - \mathbf{y}(j) \rangle = s_{\mathbf{y}}(i, i) - 2s_{\mathbf{y}}(i, j) + s_{\mathbf{y}}(j, j)$
- $s_{\mathbf{y}}(i, j) = -\frac{1}{2} (d_{\mathbf{y}}^2(i, j) - s_{\mathbf{y}}(i, i) - s_{\mathbf{y}}(j, j))$

Double centering of \mathbf{D}

- Calculate the mean:

$$\begin{aligned}\mu_j(d_{\mathbf{y}}^2(i, j)) &= \mu_j(\langle \mathbf{y}(i) - \mathbf{y}(j), \mathbf{y}(i) - \mathbf{y}(j) \rangle) \\ &= \langle \mathbf{y}(i), \mathbf{y}(i) \rangle - 2\langle \mathbf{y}(i), \mu_j(\mathbf{y}(j)) \rangle + \mu_j(\langle \mathbf{y}(j), \mathbf{y}(j) \rangle) \\ &= s_{\mathbf{y}}(i, i) + \mu_j(s_{\mathbf{y}}(j, j))\end{aligned}$$

- $\mu_i(d_{\mathbf{y}}^2(i, j)) = \mu_i(s_{\mathbf{y}}(i, i)) + s_{\mathbf{y}}(j, j)$
- $\mu_{i,j}(d_{\mathbf{y}}^2(i, j)) = \mu_i(s_{\mathbf{y}}(i, i)) + \mu_j(s_{\mathbf{y}}(j, j))$
- $s_{\mathbf{y}}(i, j) = -\frac{1}{2}(d_{\mathbf{y}}^2(i, j) - \mu_j(d_{\mathbf{y}}^2(i, j)) - \mu_i(d_{\mathbf{y}}^2(i, j)) + \mu_{i,j}(d_{\mathbf{y}}^2(i, j)))$
- $\mathbf{S} = -\frac{1}{2} \left(\mathbf{D} - \frac{1}{N} \mathbf{D} \mathbf{1}_N \mathbf{1}_N^T - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \mathbf{D} + \frac{1}{N^2} \mathbf{1}_N \mathbf{1}_N^T \mathbf{D} \mathbf{1}_N \mathbf{1}_N^T \right)$

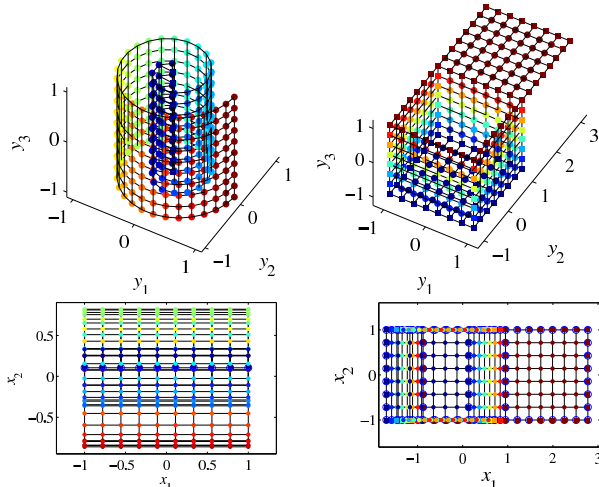
MDS: Algorithm

- 1 If data is \mathbf{Y} , center it, compute $\mathbf{S} = \mathbf{Y}^T \mathbf{Y}$ and go to step 3
- 2 If pairwise distances \mathbf{D} , transform them to scalar products \mathbf{S} by double centering
- 3 EVD: $\mathbf{S} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$
- 4 $\hat{\mathbf{X}} = \mathbf{I}_{P \times N} \mathbf{\Lambda}^{1/2} \mathbf{U}^T$

Embedding of test set

- Test set as coordinates:
 - Test point \mathbf{y}
 - $\hat{\mathbf{x}} = \mathbf{I}_{P \times D} \mathbf{V}^T \mathbf{y}$
- Test set as scalar products:
 - Test point $\mathbf{s} = \mathbf{Y}^T \mathbf{y}$
 - $\hat{\mathbf{x}} = \mathbf{I}_{P \times N} \mathbf{\Lambda}^{-1/2} \mathbf{U}^T \mathbf{s}$
- Test set as distances
 - Test point $\mathbf{d} = [\langle \mathbf{y}(i) - \mathbf{y}, \mathbf{y}(i) - \mathbf{y} \rangle]_{1 \leq i \leq N}$
 - $\mathbf{s} \approx -\frac{1}{2} \left(\mathbf{d} - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^T \mathbf{d} - \frac{1}{N} \mathbf{D} \mathbf{1}_N + \frac{1}{N^2} \mathbf{1}_N \mathbf{1}_N^T \mathbf{D} \mathbf{1}_N \right)$

Example: Embeddings with MDS



MDS variants

- Classical metric MDS preserves only the pairwise scalar products
- Variants try to preserve the pairwise distances directly by minimizing the *stress function* of metric MDS:

$$E_{\text{mMDS}} = \frac{1}{2} \sum_{i,j=1}^N w_{ij} (d_{\mathbf{y}}(i,j) - d_{\mathbf{x}}(i,j))^2$$

- Variants do not depend on any generative model

Sammon's nonlinear mapping (NLM)

- Sammon's stress function:

$$E_{\text{NLM}} = \frac{1}{c} \sum_{i,j=1}^N \frac{(d_y(i,j) - d_x(i,j))^2}{d_y(i,j)}$$
$$c = \sum_{\substack{i=1 \\ i < j}}^N d_y(i,j)$$

- Minimizes it iteratively with quasi-Newton optimization:

$$x_k(i) \leftarrow x_k(i) - \alpha \frac{\frac{\partial E_{\text{NLM}}}{\partial x_k(i)}}{\left| \frac{\partial^2 E_{\text{NLM}}}{\partial x_k(i)^2} \right|}$$

NLM: Derivation

- Direct calculation gives

$$\begin{aligned}\frac{\partial E_{\text{NLM}}}{\partial x_k(i)} &= \frac{\partial E_{\text{NLM}}}{\partial d_x(i,j)} \frac{\partial d_x(i,j)}{\partial x_k(i)} \\ &= \frac{-2}{c} \sum_{\substack{j=1 \\ j \neq i}}^N \frac{d_y(i,j) - d_x(i,j)}{d_y(i,j)d_x(i,j)} (x_k(i) - x_k(j)) \\ \frac{\partial^2 E_{\text{NLM}}}{\partial x_k^2(i)} &= \frac{-2}{c} \sum_{\substack{j=1 \\ j \neq i}}^N \left(\frac{d_y(i,j) - d_x(i,j)}{d_y(i,j)d_x(i,j)} - \frac{(x_k(i) - x_k(j))^2}{d_x^3(i,j)} \right)\end{aligned}$$

NLM: Algorithm

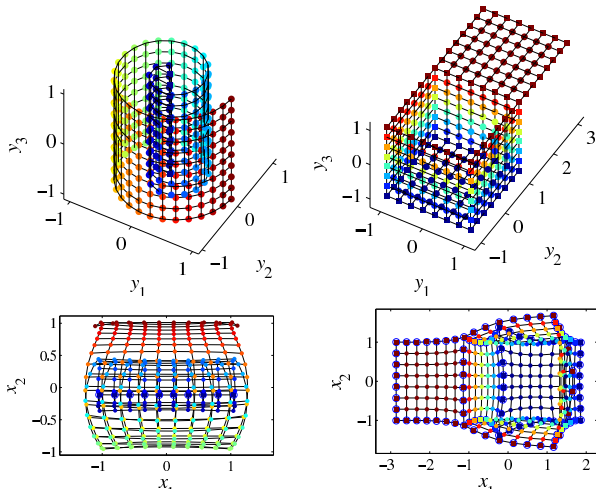
- 1 Compute pairwise distances $d_y(i, j)$
- 2 Initialize points $\mathbf{x}(i)$ randomly or by PCA
- 3 Calculate the quasi-Newton update for each point
- 4 Update the coordinates of all points $\mathbf{x}(i)$
- 5 Return to step 3 until convergence

Embedding of test set

No easy way to generalize the embedding for new points:

- Updating only the new point with quasi-Newton
- Interpolation procedure of Curvilinear Component Analysis
- Neural variants of NLM, like the SAMANN

Example: Embeddings with NLM



Curvilinear component analysis (CCA)

- Minimizes stress function:

$$E_{CCA} = \frac{1}{2} \sum_{i,j=1}^N (d_y(i,j) - d_x(i,j))^2 F_\lambda(d_x(i,j))$$

- Typically F_λ is monotonically decreasing:
 - $F_\lambda(d_x) = \exp\left(-\frac{d_x}{\lambda}\right)$
 - $F_\lambda(d_x) = H(\lambda - d_x)$, where $H(u) = 0$ if $u \leq 0$ and 1 otherwise

CCA: Derivation

- Minimization by gradient descent:

$$\mathbf{x}(i) \leftarrow \mathbf{x}(i) - \alpha \nabla_{\mathbf{x}(i)} E_{CCA}$$

- Direct calculation gives:

$$\nabla_{\mathbf{x}(i)} E_{CCA} = \sum_{j=1}^N (d_y - d_x) (2F_\lambda(d_x) - (d_y - d_x)F'_\lambda(d_x)) \frac{\mathbf{x}(j) - \mathbf{x}(i)}{d_x},$$

where $d_y = d_y(i, j)$ and $d_x = d_x(i, j)$

Condition for λ

- The condition

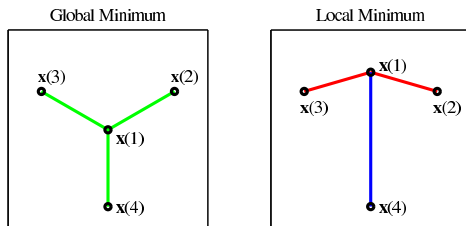
$$2F_\lambda(d_x) > (d_y - d_x)F'_\lambda(d_x)$$

guarantees that distances change reasonably

- $F_\lambda(d_x) = \exp(-\frac{d_x}{\lambda}) : \lambda > \frac{1}{2}(d_x - d_y)$
- $F_\lambda(d_x) = H(\lambda - d_x)$: The condition is always fulfilled
- The parameters α and λ can be decreased during the convergence

CCA: Problem with traditional gradient descent

- Gradient descent can get stuck into local minimum:



- Better solution: Stochastic gradient descent

CCA: Stochastic gradient descent

- Decompose E_{CCA} :

$$E_{CCA} = \sum_{i=1}^N E_{CCA}^i$$

$$E_{CCA}^i = \frac{1}{2} \sum_{j=1}^N (d_y(i,j) - d_x(i,j))^2 F_\lambda(d_x(i,j))$$

- Separate optimization:

$$\begin{aligned} \mathbf{x}(j) &\leftarrow \mathbf{x}(j) - \alpha \nabla_{\mathbf{x}(j)} E_{CCA}^i \\ &\leftarrow \mathbf{x}(j) - \alpha \beta(i,j) \frac{\mathbf{x}(i) - \mathbf{x}(j)}{d_x}, \end{aligned}$$

where $\beta(i,j) = (d_y - d_x) (2F_\lambda(d_x) - (d_y - d_x)F'_\lambda(d_x))$

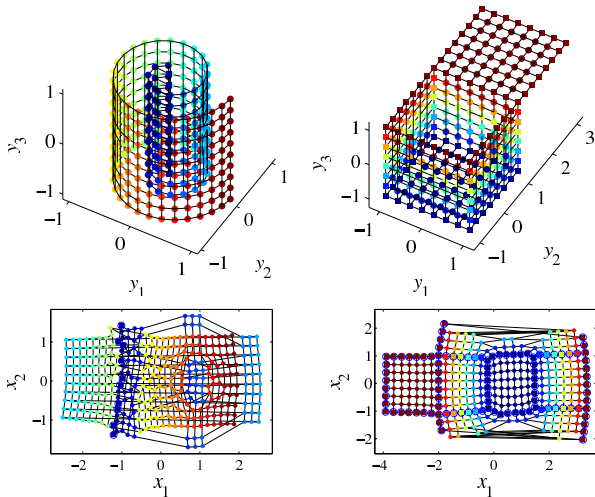
CCA: Algorithm

- 1 Perform vector quantization for size reduction
- 2 Compute pairwise distances $d_y(i, j)$
- 3 Initialize points $\mathbf{x}(i)$ randomly or by PCA
- 4 Give learning rate α and neighborhood width λ
- 5 Select a point $\mathbf{x}(i)$ and update all others
- 6 Return to step 5 until all points $\mathbf{x}(i)$ selected in this epoch
- 7 If not converged, return to step 4

Embedding of test set

- Original points are fixed
- For each test point, the update rule is applied to move it to the right position

Example: Embeddings with CCA



Summary

- Three dimensionality reduction methods based on distance preservation: multidimensional scaling, Shannon's nonlinear mapping, curvilinear component analysis
- MDS is a generalization of PCA to pairwise scalar products and distances
- NLM and CCA preserve distances directly by minimizing a corresponding stress function