HUDM5124: Introduction to Multidimensional Scaling, Clustering, and Related Methods

Session 4:

- nonmetric MDS
    - -Kruskal's algorithm
    - -Some Issues in MDS

Nonmetric MDS: the problem

GIVEN: a matrix Δ of proximities among n objects, that are assumed to be of at least ORDINAL measurement level

CONSTRUCT: an N x R matrix X (= the configuration of N points in a geometric space of R dimensions), such that the distances in the geometric space, D, (= the "model distances") are monotonically related to the proximities (increasing for dissim; decreasing for sim)

TWO MAIN GOALS:

1) Find the configuration matrix X representing the positions of the N stimuli on R dimensions

2) Find the shape of the function relating the model distances to the proximities ("optimal scaling" problem)

Issues to be assumed or investigated: find the best Minkowski distance metric; determine the "true" number of dimensions; etc.

Approximately linear functions relating proximities to distances (but derived from nonmetric MDS)

(source: Kruskal & Wish, 1977)

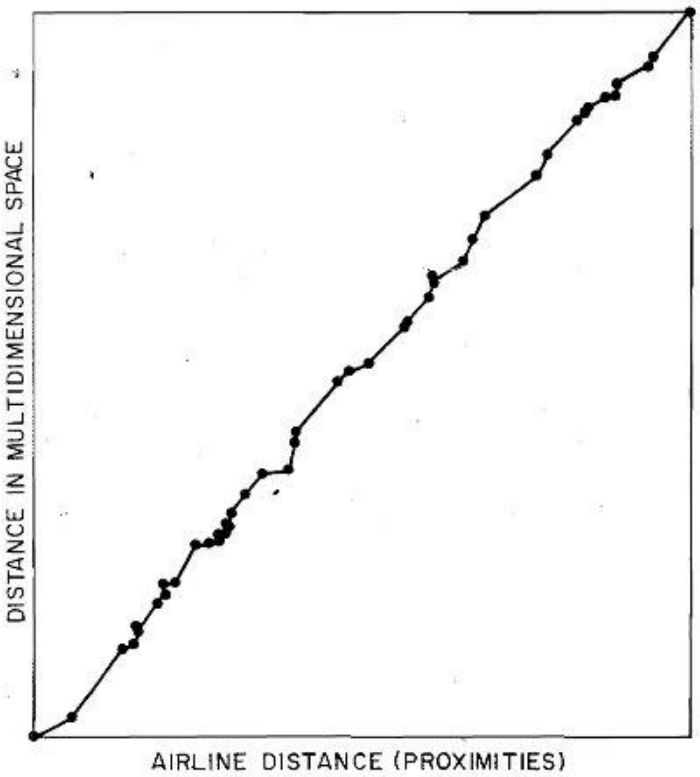


Figure 5A: Scatter Diagram Associated with Configuration in Figure 1(c)

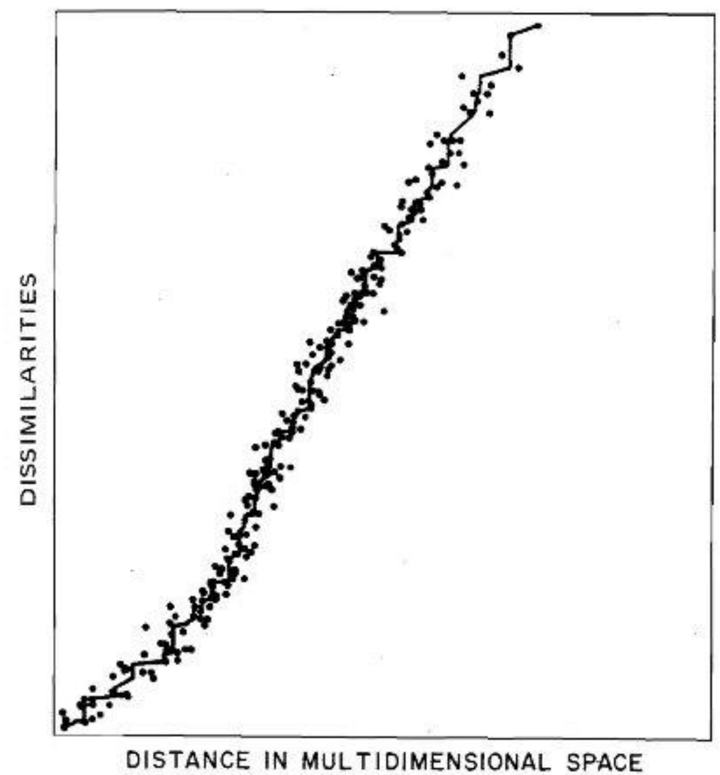
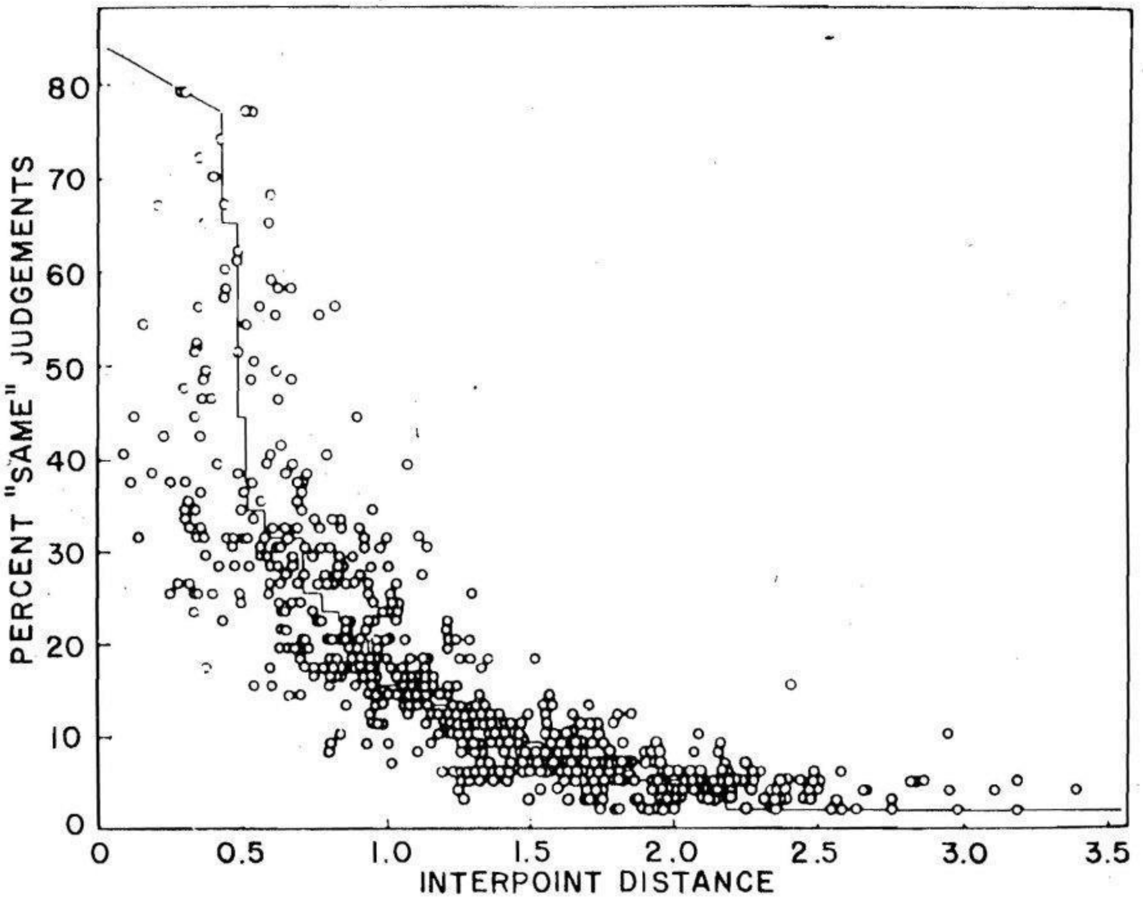
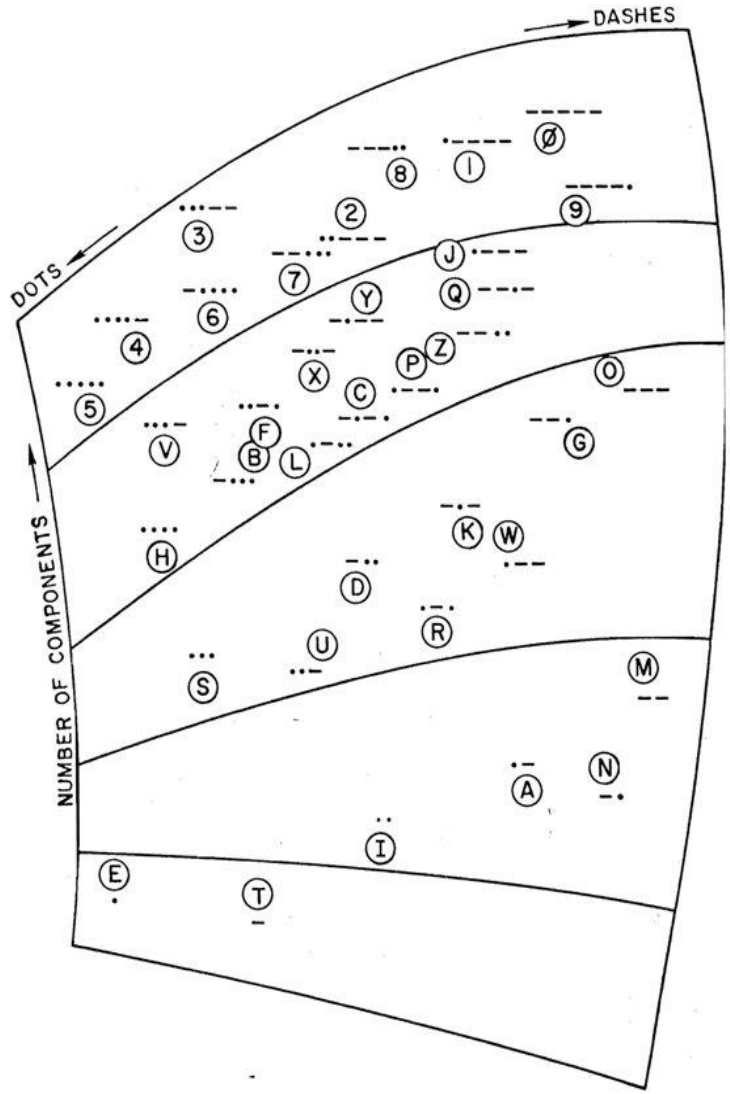


Figure 5B: Scatter Diagram for Some Color Data from Indow and Kanazawa (not discussed in text)

A nonlinear function relating proximities to distances (nonmetric MDS of Rothkopf's Morse code data)



Derived 2D configuration for Rothkopf's Morse code data



Kruskal's (1964) algorithm for nonmetric MDS

BASIC OUTLINE OF ITERATIVE ALGORITHM:

0. specify dimensionality & Minkowski metric; select initial configuration (X) matrix (→ use random configuration, or metric MDS solution, as starting configuration)

ITERATE:

1. Compute interobject distances using X matrix
2. Find optimal scaling transformation of Δ (= "least-squares monotonic transformation")
3. Test for convergence: if yes, terminate; if no, continue
4. Adjust entries in configuration (X) matrix in direction of steepest descent (the negative gradient) of loss function (STRESS)

[return to Step 1]

Kruskal's least-squares monotonic transformation (= "monotonic regression")

Problem: find a monotonic transformation of the proximities, **f(δ)**, that is closest (in a least-squares sense) to the model distances

→ this is really trying to determine how closely the model distances correspond to the <u>rank-order information</u> in the data

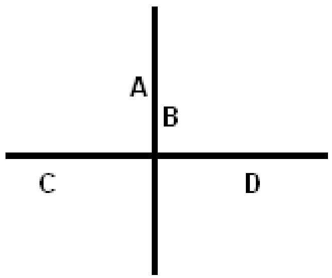
Method: given model distances D and dissimilarities Δ, expressed as vectors of length n(n-1)/2:

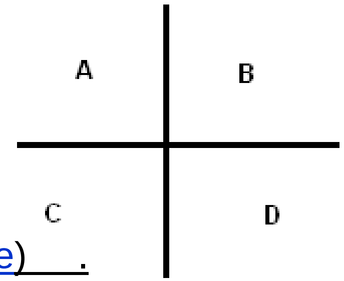
1. Order the distances according to the rank order of the proximities - these values are the initial estimate of f(δ).
2. Put the entries of f(δ) into nondecreasing order ("primary approach" = ties in data may be broken; "secondary approach" = tied entries must remain tied)

Measure of fit:

$$\text{Stress}(1) = \left[\frac{\sum_{i<j} (f(\delta_{ij}) - d_{ij})^2}{\sum_{i<j} d_{ij}^{\ 2}} \right]^{1/2}$$

Example: Kruskal's Least Squares Monotonic Transform

"true" config: 

initial config: 

= X

(Note tie) .

Assume rank(Δ) = (δ_{AB}=1, δ_{AC}=3.5 , δ_{BC}=3.5, δ_{AD}=5, δ_{BD}=2, δ_{CD}=6)

From config. above, D = (d_{AB}=1, d_{AC}=1.2, d_{BC}=1.8, d_{AD}=1.7, d_{BD}=1.1, d_{CD}=1.3)

LSMT: First, rewrite model distances in order of the proximities:

→D = (d_{AB}=1, d_{BD}=1.1, [d_{AC}=1.2, d_{BC}=1.8], d_{AD}=1.7, d_{CD}=1.3)

Transform proximities: (Note tie in dissim's)

D = (1 1.1 1.2 1.8 1.7 1.3) (primary approach to ties: "free" ordering)

f(δ) = (1 1.1 1.2 1.75 1.75 1.3)

f(δ) = (1 1.1 1.2 1.6 1.6 1.6) .

D = (1 1.1 1.5 1.5 1.7 1.3) (secondary approach to ties: tie = constraint)

f(δ) = (1 1.1 1.5 1.5 1.7 1.3)

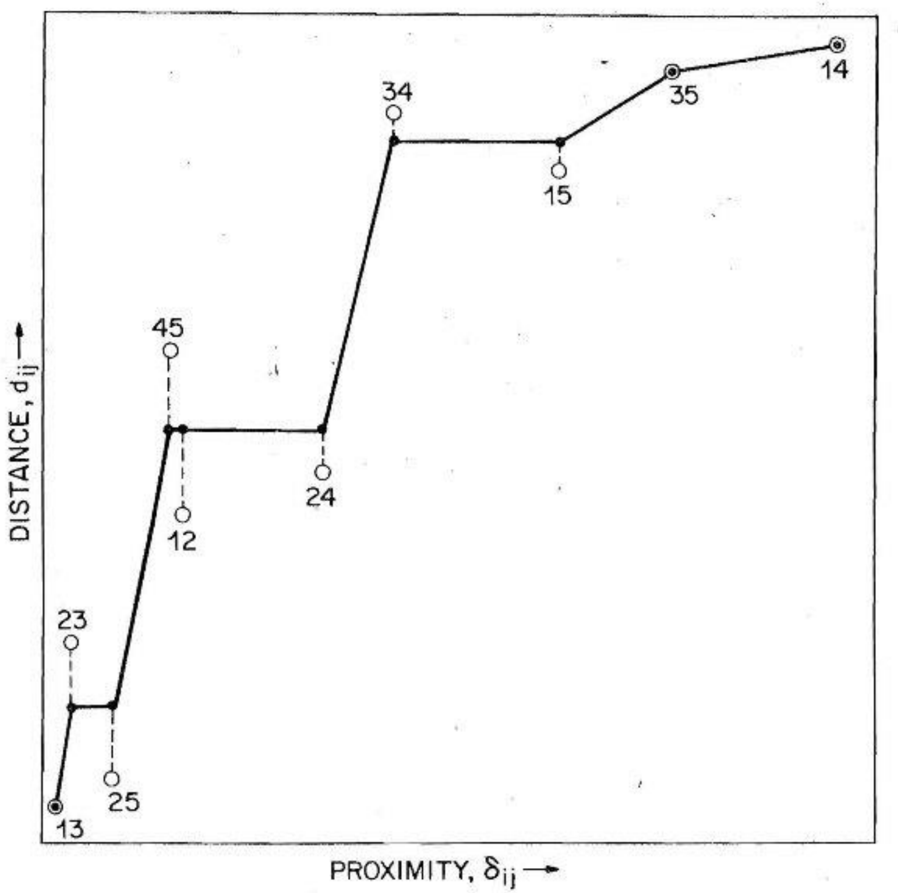
f(δ) = (1 1.1 1.5 1.5 1.5 1.5)

$$\mathrm{SSE} = \sum_{i<j} (\mathrm{f}(\delta_{ij}) - d_{ij})^2 \quad \rightarrow \quad \mathrm{Stress}(1) = \left[\frac{\sum_{i<j} (\mathrm{f}(\delta_{ij}) - d_{ij})^2}{\sum_{i<j} d_{ij}^{\,2}} \right]^{1/2}$$

Graph of a least-squares monotonic transformation

(from Kruskal, 1964)

(numeric labels for points in the graph below identify specific object pairs)



Kruskal's (1964) algorithm - detail

OUTLINE OF ITERATIVE ALGORITHM:

0. Select dimensionality; select initial configuration (X_0) matrix
   (use random configuration, or metric MDS solution)

ITERATE:

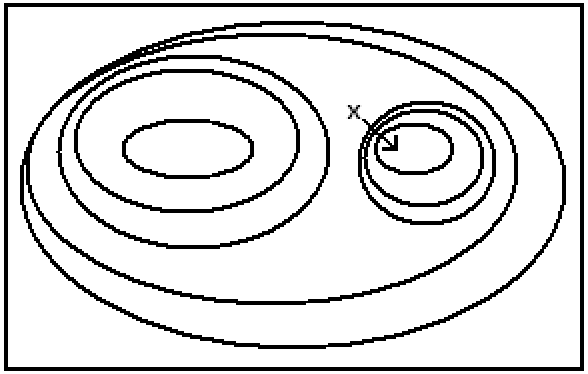
1. Normalize the configuration matrix X_i
2. Compute interpoint distances D using X_i matrix
3. Find optimal scaling transformation of Δ (= least-squares monotonic transformation)
4. Calculate the (negative) gradient of Stress w.r.t. X (= direction of steepest descent of loss function, Stress)
5. Test for convergence: if yes, terminate; if no, continue
6. Calculate new value for step size, α_i
7. Adjust entries in configuration (X_i) matrix in direction of gradient (by some step size)

[return to Step 1]

Note: missing data requires no special solution – just run algorithm on non-missing data entries.

Gradient method for minimizing Stress

BASIC IDEA: iteratively adjust the entries in X (the coordinates), to move in the direction of the negative gradient (direction of steepest descent of the loss function, S=stress(1)) or stress(2)



$$-\mathrm{G} = -\frac{\partial S}{\partial X} = [-\partial S/\partial x_{11}, -\partial S/\partial x_{11}, \ldots, -\partial S/\partial x_{NR}]$$

For Minkowski p-metric (Kruskal, 1964b):

$$g_{kl} = \sum_{i,j} (\delta^{ki} - \delta^{kj}) \left[\frac{d_{ij} - \hat{d}_{ij}}{S^*} - \frac{d_{ij}}{T^*}\right] \frac{\left|x_{il} - x_{jl}\right|^{p-1}}{d_{il}^{p-1}} signum(x_{il} - x_{jl})$$

ISSUE: How far to move in this direction? (“step-size” issue)

SOLUTION: Use dynamic step-size adjustment:

if multiple moves in same direction → increase step size;

if successive steps in “opposite” directions → decrease step size

Kruskal's (1964) algorithm (cont): Adjusting the step size, α

Initial value: α=.2 (for random config), smaller for "rational" start

On step i,

$\alpha_i = \alpha_{(i-1)}$ (angle)(relax)(good-luck)

where

"angle" = $4.0^{\cos(\theta)**3}$

θ = angle between present gradient (step i) and previous gradient (i-1)

"relax" = $\dfrac{1.3}{1+(\text{5-step-ratio})^5}$

"5-step-ratio" = MIN[1,(stress$_i$ / stress$_{(i-5)}$)]

"good-luck" = MIN[1,(stress$_i$ / stress$_{(i-1)}$)]

Note: $\cos(\theta)$ may be calculated as $r(G_i,G_{i-1})$

Two versions of stress

$$\text{Stress}(1) = \left[\frac{\sum_{i<j} (\mathrm{f}(\delta_{ij}) - d_{ij})^2}{\sum_{i<j} d_{ij}{}^2} \right]^{1/2} \qquad \text{Stress}(2) = \left[\frac{\sum_{i<j} (\mathrm{f}(\delta_{ij}) - d_{ij})^2}{\sum_{i<j} (d_{ij} - \bar{d})^2} \right]^{1/2}$$

Use of Stress(1) can result in degenerate configurations (i.e., points collapsing into a few clumps, a simplex, a ring).

Thus Stress(2) is generally recommended.

Degeneracy may also be affected by normalization of X (step 2).

Availability of software for nonmetric MDS:

The Kruskal (1964) algorithm:

program	Author	source	distributed as:
MDSCALE	Kruskal	NETLIB	FORTRAN source
KYST-2A	Young	NETLIB	FORTRAN source
SYSTAT	Wilkinson	SPSS	commercial package
isoMDS		R	public domain package

Other algorithms/software for nonmetric MDS:

program	Author	source	distributed as:
ALSCAL	Young	SPSS, SAS	commercial package
PROXSCAL	Leiden	SPSS	commercial package
SMACOF	DeLeeuw	PROXSCAL,R	commercial,public

Some practical issues:

Choosing the dimensionality. The problem of determining the dimensionality R of the solution space must be addressed. Often the dimensionality is selected *a priori* based on theory or *post-hoc* based on interpretability or on perception of an "elbow" in stress function.

How much data? Because in nonmetric MDS we are using *only the ordinal information* in the proximities, we must have a higher ratio of # of data points (obs) to # of estimated parameters, the (n-1)R coordinates of X (rule of thumb: at least 7 stimuli per dimension).

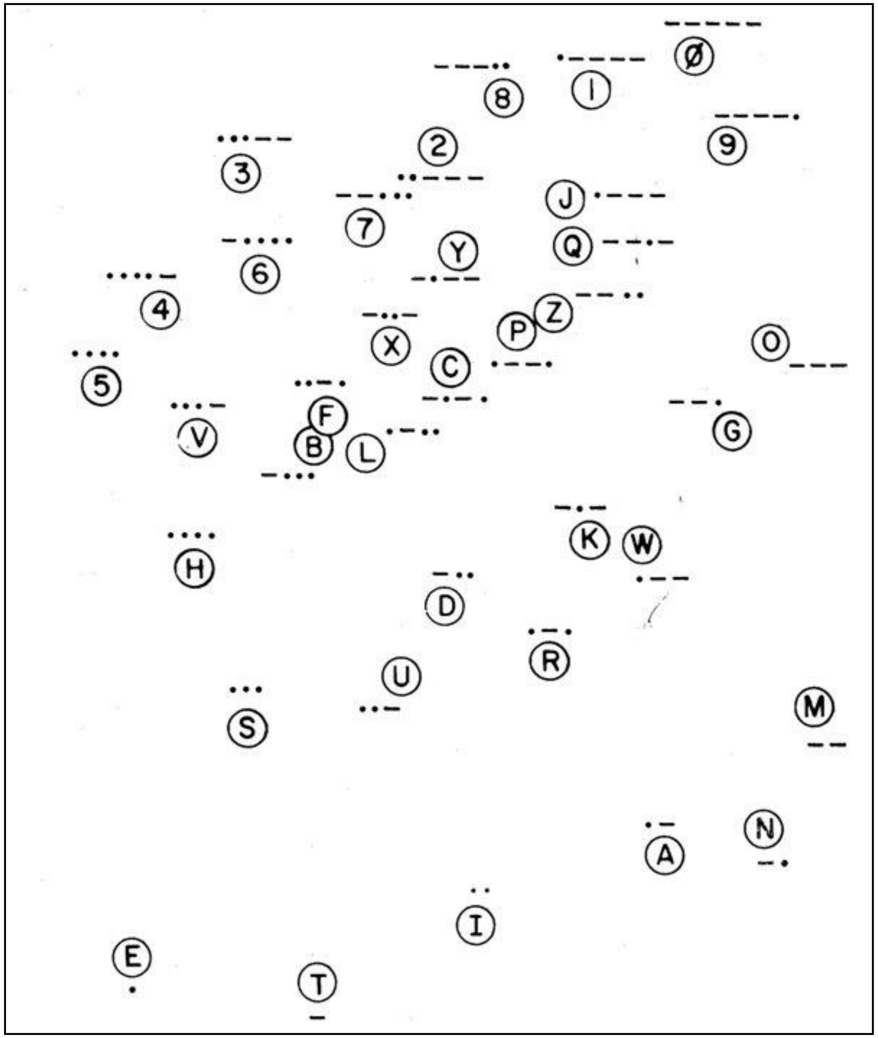
Degenerate solutions. Rings, clumps of multiple stimuli, etc. are often signs of a degenerate solution. **Fixes:** increase # of data points, decrease dimensionality, use secondary approach to ties, try a metric solution.

Interpreting solutions by eye. Remember that the orientation of the solution w.r.t. the axes is arbitrary. High-dimensional graphical rotation software may be useful if R>2.

Interpreting solutions "objectively". Is there a relatively objective way to interpret dimensions? → regression of single "attributes" into the space: $A = b0 + b1X_1 + b2X_2 + ..$ Then plot regression coeff's.

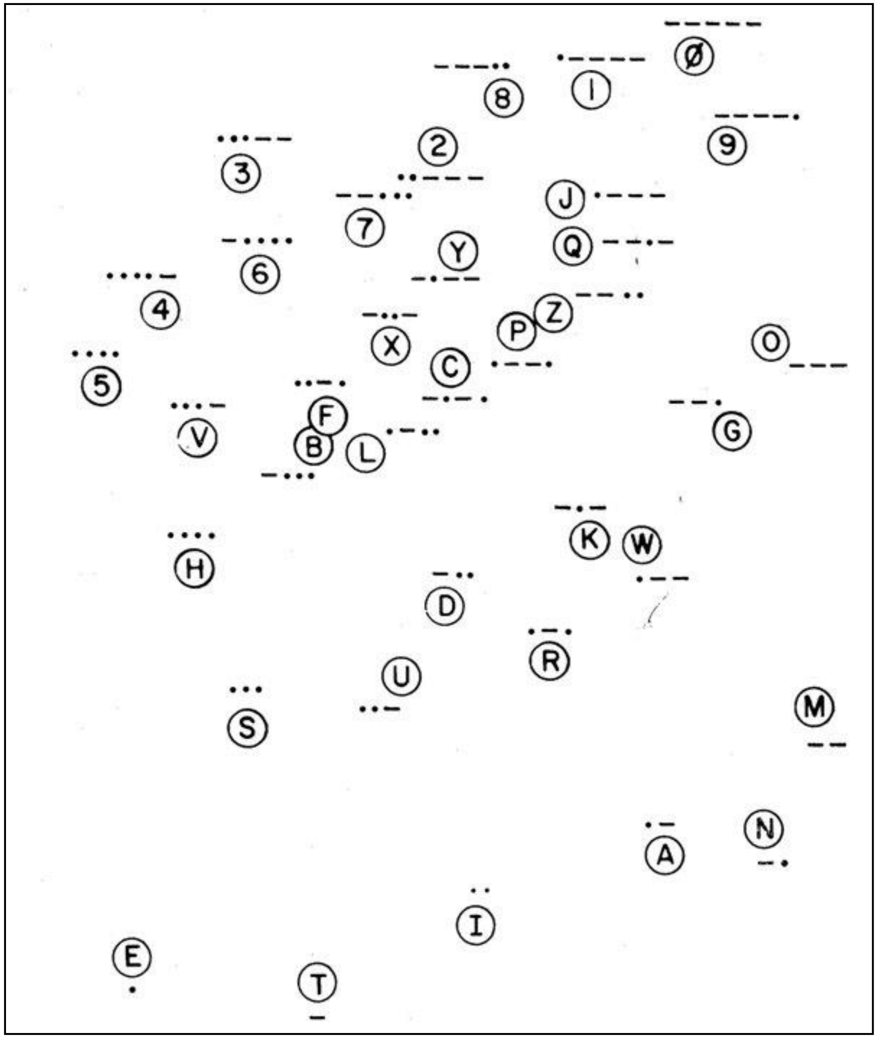
Recovery of metric information: even though this technique is "nonmetric", Young (1980) showed that good recovery of metric information is achieved.

2D config, Morse code data - interpreting via attribute regression



?

2D config, Morse code data - interpreting via attribute regression



METHOD:

1) Define attribute vectors on stimuli:

A1 = # components of signal
A2 = proportion of dashes

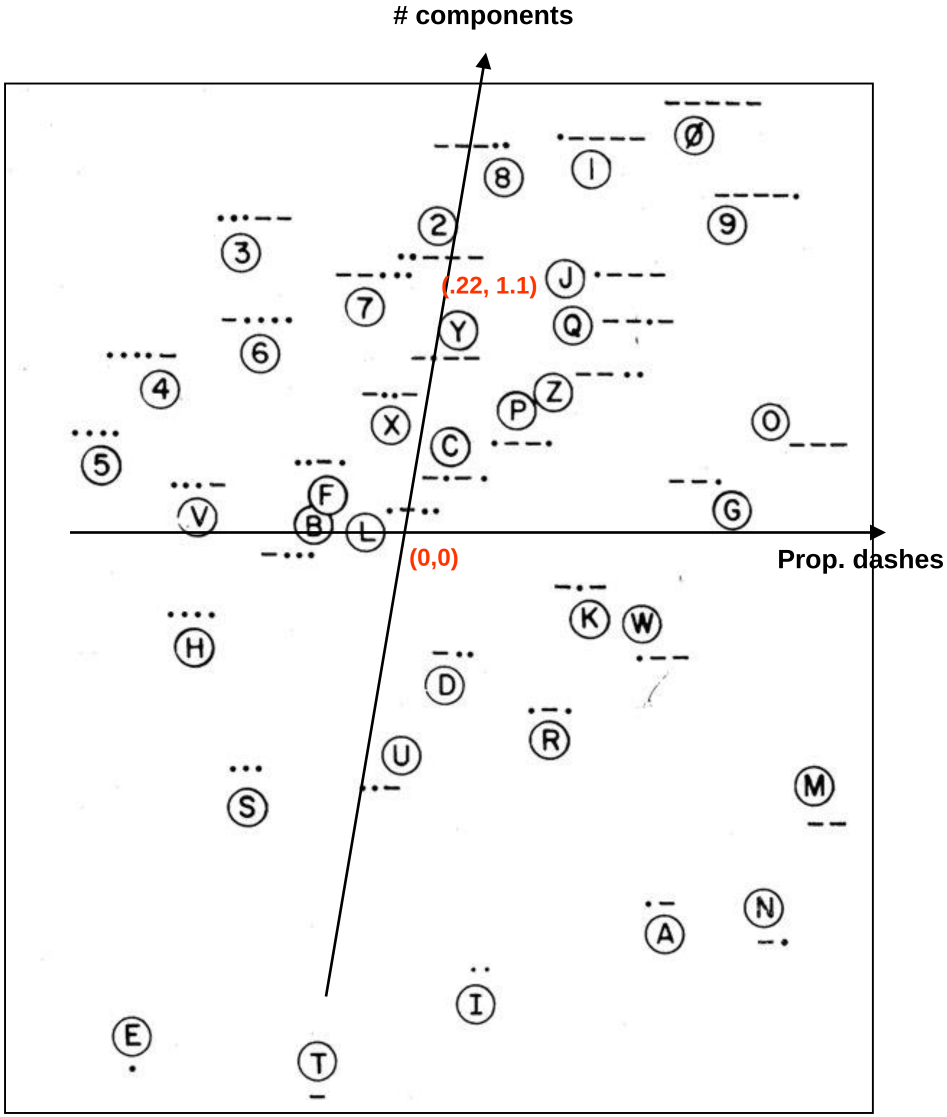
2) Regress each vector into config. space (X1 = horiz, X2 = vert)

$$A1 = 3.5 + 0.22\ X1 + 1.10\ X2$$

$$A2 = 0.5 + 0.92\ X1 - 0.01\ X2$$

3) For each attribute, plot the regression coefficients as a vector in the configuration space, through (0,0)

2D config, Morse code data - interpreting via attribute regression



METHOD:

1) Define attribute vectors on stimuli:

A_1 = # components in signal
A_2 = proportion of dashes

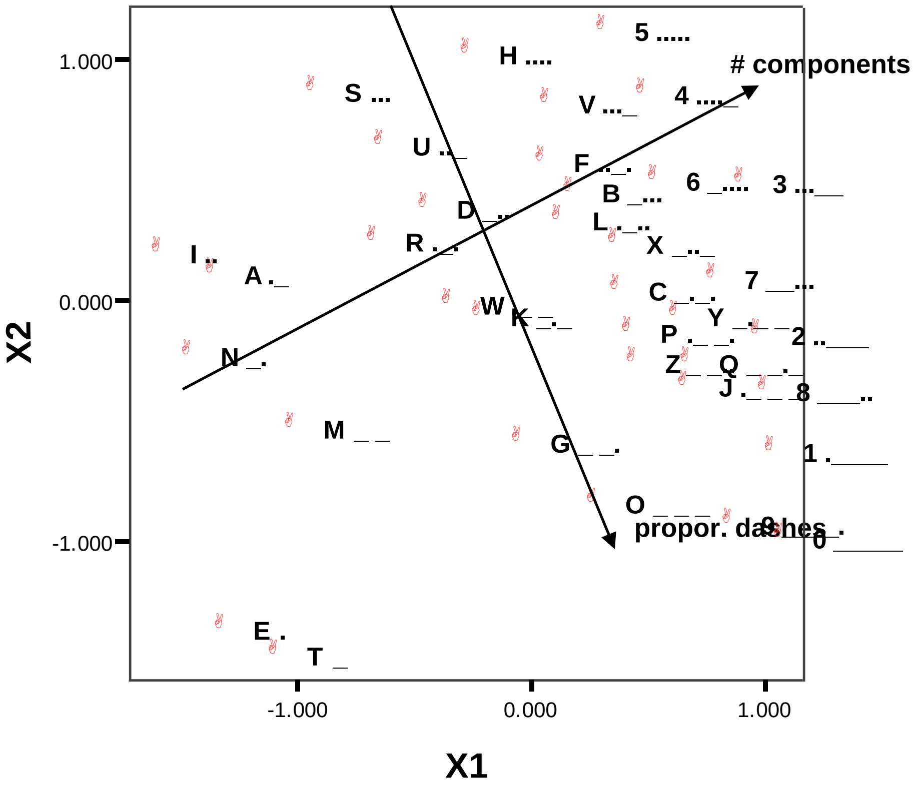
2) Regress each vector into config. space (X1 = horiz dim, X2 = vertical)

$$A_1 = 3.5 + 0.22\ X1 + 1.10\ X2$$

$$A_2 = 0.5 + 0.92\ X1 - 0.01\ X2$$

3) For each attribute, plot the regression coefficients as a vector in the configuration space, through the origin (0,0)

Example: plot of 2D Morse code solution (SPSS)
interpretation via attribute regression



Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.
1	(Constant)	3.667	.051		72.542	.000
	X1	**1.356**	.065	.910	20.784	.000
	X2	**.603**	.080	.330	7.544	.000

a. Dependent Variable: Ncomp

Coefficients[a]

Model		Unstandardized Coefficients B	Unstandardized Coefficients Std. Error	Standardized Coefficients Beta	t	Sig.
1	(Constant)	.479	.034		14.302	.000
	X1	**.111**	.043	.287	2.573	.015
	X2	**-.338**	.053	-.712	-6.379	.000

a. Dependent Variable: PRdash