Fixed Point Methods for Embedding Large Graphs

Michael W. Trosset

Department of Statistics

Indiana University

This research was supported by a grant from the Office of Naval Research.

Dimension Reduction

Multivariate data are usually represented as points in an ambient feature space, e.g., $y_1, \ldots, y_n \in \Re^q$.

By dimension reduction, we mean the representation of $y_1, \ldots, y_n$ as $x_1, \ldots, x_n \in \Re^d$ for $d < q$.

Some dimension reduction techniques construct $x_1, \ldots, x_n$ by linear operations. For example, extracting the first d principal components of $y_1, \ldots, y_n$ is equivalent to projecting $y_1, \ldots, y_n$ into the best-fitting affine linear subspace of $\leq d$ dimensions.

We consider several techniques for nonlinear dimension reduction, each motivated by the conceit that $y_1, \ldots, y_n$ lie on a low-dimensional manifold in $\Re^q$.

Definition: A subset M $\subset \Re^q$ is called a smooth manifold of dimension p iff each $y \in$ M has a neighborhood that is diffeomorphic to an open subset of $\Re^p$.

Techniques for *manifold learning* attempt to localize structure in the ambient space, then exploit properties of manifolds to construct the low-dimensional representation.

Manifold Learning

- Tenenbaum, de Silva & Langford (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323.

  Isomap "seeks to preserve the intrinsic geometry of the data, as captured in the geodesic manifold distances between all pairs of data points."

- Roweis & Saul (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326.

  The same weights that locally reconstruct y_i from its neighbors in the feature space should also reconstruct x_i from its neighbors in the representation space.

- Belkin & Niyogi (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396.

  "Drawing on the correspondence between the graph Laplacian, the Laplace-Beltrami operator on the manifold, and the connections to the heat equation. . . "

Example: Isomap

Suppose that $y_1, \ldots, y_n$ lie on a manifold in a high-dimensional Euclidean space.

Fix $\epsilon > 0$. The ϵ-Isomap embedding of $y_1, \ldots, y_n \in \Re^q$ as $x_1, \ldots, x_n \in \Re^d$ is constructed as follows:

1. **Construct Graph** Let $w_{ij} = \|y_i - y_j\|$ if $\|y_i - y_j\| < \epsilon$ and $w_{ij} = 0$ otherwise. Define a weighted graph G with n vertices as follows: vertices i and j are connected iff $w_{ij} > 0$, in which case edge $i \sim j$ is weighted by w_{ij}.

2. **Compute Proximities** Let δ_{ij} denote the shortest path distance in G between vertices i and j, in which case $\Delta = [\delta_{ij}]$ and $\Delta_2 = [\delta_{ij}^2]$ are dissimilarity matrices.

3. **Embed Proximities** Embed Δ_2 in $\Re^d$ by classical multidimensional scaling (CMDS). CMDS solves the nonlinear least squares problem problem of approximating fallible inner products with Euclidean inner products. The global solution can be computed from the d largest eigenvalues and corresponding eigenvectors of a dense symmetric $n \times n$ matrix.

Embedding

Given n objects and pairwise proximities, either dissimilarities $\Delta = [\delta_{ij}]$ or similarities $\Gamma = [\gamma_{ij}]$, we want to represent the objects as $x_1, \ldots, x_n \in \Re^d$.

Inner product methods embed as follows:

1. Estimate inner products, b_{ij}, from proximities.

2. Approximate the b_{ij} with Euclidean inner products, $\langle x_i, x_j \rangle$, e.g., by minimizing the *strain criterion*,
$$\sum_{i \sim j} \left(\langle x_i, x_j \rangle - b_{ij} \right)^2 .$$

Distance methods embed by approximating dissimilarities with Euclidean distances, $\|x_i - x_j\|$, e.g., by minimizing the *raw stress criterion*,

$$\sum_{i \sim j} w_{ij} \left(\|x_i - x_j\| - \delta_{ij} \right)^2 .$$

Minimizing the Raw Stress Criterion

The weighted raw stress criterion is

$$\sigma(X) = \sum_{i \sim j} w_{ij} \left[d_{ij}(X) - \delta_{ij} \right]^2 ,$$

where $d_{ij} = \|x_i - x_j\|$ and $w_{ij} \geq 0$.

Minimizing $\sigma()$ requires numerical optimization. One popular technique constructs an initial configuration by minimizing strain (classical MDS), then minimizes $\sigma()$ by the *Guttman majorization algorithm* (GMA), a fixed point method that converges to a connected set of stationary X.

GMA can be interpreted as a weighted gradient method. It does not use second-order information about $\sigma()$.

Digression

Researchers once thought that $\sigma()$ had many nonglobal minimizers. For example, Groenen (1993) used SMACOF-1 (a popular implementation of GMA) to study one 17×17 dissimilarity matrix and apparently found 1098 local minimizers. This understanding led some researchers to apply expensive global optimization methods, e.g., tunneling, simulated annealing, genetic algorithms, etc.

Kearsley, Tapia & Trosset (1998) proposed a globalized Newton's method for minimizing $\sigma()$ and showed that SMACOF-1 tends to stop prematurely. In numerical experiments. . .

- SMACOF-1 exhibited better global behavior and Newton exhibited better local behavior.

- In contrast to SMACOF-1, Newton consistently found the same minimum stress value from different initial configurations, including putative minimizers returned by SMACOF-1.

For more information. . .

Groenen (1993). *The Majorization Approach to Multidimensional Scaling*, DSWO Press.

Trosset and Mathar (1997). On the existence of nonglobal minimizers of the stress criterion for metric multidimensional scaling. *Proceedings of the Statistical Computing Section*, American Statistical Association, pp. 158–162.

Kearsley, Tapia, and Trosset (1998). The solution of the metric stress and sstress problems in multidimensional scaling using Newton's method. *Computational Statistics*, 13:369–396.

Malone, Tarazaga, and Trosset (2002). Better initial configurations for metric multidimensional scaling. *Computational Statistics and Data Analysis*, 41:143–156.

Extreme Multidimensional Scaling

Trosset and Groenen (2005). Multidimensional scaling algorithms for large data sets.

Suppose that n is large. Instead of CMDS followed by GMA. . .

1. Construct an initial configuration by the *method of standards*:
   - Embed a fixed number of *anchor points*; then, individually position the remaining points in relation to the anchor points. This construction is $O(n)$.
   - Instead of minimizing a traditional error criterion, use a fast heuristic that solves $Ax = b_i$, where A is $d \times d$, for $b_1, \ldots, b_{n-d-1}$.

2. Decrease $\sigma()$ by several iterations of a new *diagonal majorization algorithm* (DMA). Use $O(n)$ dissimilarities and stop after a fixed number of iterations.

Linear Embedding

Let δ_{ij} denote the dissimilarity of objects i and j.

For $d = 1$, suppose that x_1 and x_2 have been embedded. Individually embed each remaining x_i by solving the 1×1 linear system induced by

$$\begin{aligned} \|x - x_1\|^2 &= \delta_{i1}^2 \\ \|x - x_2\|^2 &= \delta_{i2}^2, \end{aligned}$$

viz.,

$$Ax = [2(x_1 - x_2)]\, x = \left(x_1^2 - x_2^2\right) - \left(\delta_{i1}^2 - \delta_{i2}^2\right) = b_i.$$

Thus, we position each x_i so that it minimizes

$$\begin{aligned} \|Ax - b_i\|^2 &= \left[(x - x_1)^2 - \delta_{i1}^2\right]^2 + \left[(x - x_2)^2 - \delta_{i2}^2\right]^2 \\ &\quad -2\left[(x - x_1)^2 - \delta_{i1}^2\right]\left[(x - x_2)^2 - \delta_{i2}^2\right]. \end{aligned}$$

For $d > 1$,

- Kearsley, Tapia, Trosset (1998); Dong & Wu (2002)

  Suppose that $x_1, \ldots, x_{d+1}$ have been embedded. Individually embed each remaining x_i by solving the $d \times d$ linear system $Ax = b_i$ induced by

$$\begin{aligned} \|x - x_1\|^2 &= \delta_{i1}^2 \\ &\vdots \\ \|x - x_{d+1}\|^2 &= \delta_{i,d+1}^2. \end{aligned}$$

- FastMap (Faloutsos & Lin, 1995)

  Choose an initial axis, embed in the initial axis, update the δ_{ij}^2 by "projecting" into an orthogonal hyperplane H, choose a second axis in H, embed the updated δ_{ij}^2 in the second axis, etc.

Guttman Majorization Algorithm

The stationary equation $\nabla\sigma(X) = 0$ can be written as $VX = B(X)X$, where

$$V = \sum_{i<j} w_{ij} \left(e_i - e_j\right) \left(e_i - e_j\right)^t$$

and $B(X)$ are $n \times n$ matrices. This suggests an iterative algorithm: choose X_{k+1} to solve

$$VX = B\left(X_k\right) X_k, \quad \text{i.e., solve } d \text{ linear systems, } \quad Vx = b_k.$$

GMA has traditionally been written as

$$X_{k+1} = V^{\dagger} B\left(X_k\right) X_k = \Gamma\left(X_k\right),$$

where Γ is the Guttman transform.

GMA: Equal Weights

If $w_{ij} = c$ for $i \neq j$, then

$$\begin{aligned} V &= c\sum_{i<j}(e_i - e_j)(e_i - e_j)^t = cn\left(I - \frac{ee^t}{n}\right), \\ V^\dagger &= \frac{1}{cn}\left(I - \frac{ee^t}{n}\right), \quad \text{and} \\ B(X)X &= c\sum_{i<j}\frac{\delta_{ij}}{d_{ij}(X)}(e_i - e_j)(e_i - e_j)^t X. \end{aligned}$$

$B(X)X$ is already centered, so $V^\dagger B(X)X$ is

$$\frac{1}{n}\sum_{i<j}\frac{\delta_{ij}}{d_{ij}(X)}\begin{pmatrix} y_{ij1}^t \\ \vdots \\ y_{ijn}^t \end{pmatrix}, \quad \text{where} \quad y_{ijs} = \begin{Bmatrix} x_i - x_j & s = i \\ x_j - x_i & s = j \\ 0 & s \neq i, j \end{Bmatrix}.$$

GMA: General Weights

Because col$(V) = e^{\perp}$, $\tilde{V} = V + ee^t > 0$.

Instead of computing $V^\dagger$ and $V^\dagger B(X_k)X_k$, we can obtain X_{k+1} by solving

$$\tilde{V}X = B\left(X_k\right)X_k = B_k.$$

To compute K iterations, one must solve $\tilde{V}x = b$ with Kd choices of b. Because $\tilde{V} > 0$, we can do so as follows:

1. Compute the Cholesky decomposition $\tilde{V} = LL^t$.
   This requires approximately $n^3/6$ multiplications.

2. To solve each $LL^t x = b$,

   (a) Backsolve the triangular system $Ly = b$ to obtain $\tilde{y}$; then
   (b) Backsolve the triangular system $L^t x = \tilde{y}$ to obtain $\tilde{x}$.

   This requires approximately dn^2 multiplications per iteration.

From GMA to DMA

The diagonal majorization algorithm is based on the observation that

$$V = \begin{pmatrix} \sum w_{1s} & & & \\ & \ddots & -w_{ij} & \\ & & \ddots & \\ & -w_{ij} & & \ddots \\ & & & \sum w_{ns} \end{pmatrix}$$

is dominated by its diagonal, which suggests approximating V with $\text{diag}(V)$.

For future use, note that the symmetric matrix

$$2\,\text{diag}(V) - V = \begin{pmatrix} \sum w_{1s} & & & \\ & \ddots & w_{ij} & \\ & & \ddots & \\ & w_{ij} & & \ddots \\ & & & \sum w_{ns} \end{pmatrix}$$

is diagonally dominant, hence positive semidefinite. It follows that

$$\text{trace}\left[(X-Y)^t\left[2\,\text{diag}(V)-V\right](X-Y)\right] \geq 0,$$

hence

$$\text{trace}\left(X^tVX\right) \leq$$
$$\text{trace}\left(X^t\left[2\,\text{diag}(V)\right]X\right) - 2\,\text{trace}\left(X^t\left[2\,\text{diag}(V)-V\right]Y\right) + c(Y).$$

Gradient Interpretation

Because $e^t X = 0$, an iteration of GMA is

$$\begin{aligned} X_{k+1} &= X_k - X_k + X_{k+1} = X_k - V^\dagger V X_k + V^\dagger B\left(X_k\right) X_k \\ &= X_k - \frac{1}{2} V^\dagger \, 2\left[V X_k - B\left(X_k\right) X_k\right] \\ &= X_k - \frac{1}{2} V^\dagger \, \nabla\sigma\left(X_k\right). \end{aligned}$$

DMA replaces V with $2\,\mathrm{diag}(V)$: an iteration of DMA is

$$X_{k+1} = X_k - \frac{1}{2}\left[2\,\mathrm{diag}(V)\right]^{-1} \, \nabla\sigma\left(X_k\right).$$

To understand why DMA works, we revisit de Leeuw's (1988) convergence analysis of GMA.

Majorization

De Leeuw (1988) showed that

$$\sigma(X) \leq \omega_Y(X) = c(Y) + \text{trace}\left(X^t V X\right) - 2\,\text{trace}\left(X^t B(Y) Y\right).$$

If X is centered, then we can write

$$\begin{aligned}\omega_Y(X) &= c(Y) + \text{trace}\left(X^t V X\right) - 2\,\text{trace}\left(X^t V V^\dagger B(Y) Y\right) \\ &= c(Y) + \left\|X - V^\dagger B(Y) Y\right\|_V^2.\end{aligned}$$

The minimizer of $\omega_Y(X)$ is

$$X_* = V^\dagger B(Y) Y.$$

GMA sets $Y = X_k$ and solves $VX = B(X_k)X_k$ to obtain X_{k+1}.

To derive DMA, recall that

$$\begin{aligned}&\text{trace}\left(X^t V X\right) \leq \\ &\quad \text{trace}\left(X^t\left[2\,\text{diag}(V)\right]X\right) - 2\,\text{trace}\left(X^t\left[2\,\text{diag}(V) - V\right]Y\right) + c(Y).\end{aligned}$$

It follows that

$$\begin{aligned}\sigma(X) &\leq \omega_Y(X) = c(Y) + \text{trace}\left(X^t V X\right) - 2\,\text{trace}\left(X^t B(Y) Y\right) \\ &\leq \bar{\omega}_Y(X) = c(Y) + \text{trace}\left(X^t\left[2\,\text{diag}(V)\right]X\right) - \\ &\quad 2\,\text{trace}\left(X^t\left[2\,\text{diag}(V) - V + B(Y)\right]Y\right) \\ &= c(Y) + \text{trace}\left(X^t\left[2\,\text{diag}(V)\right]X\right) - \\ &\quad 2\,\text{trace}\left(X^t\left[2\,\text{diag}(V)\right]\left[2\,\text{diag}(V)\right]^{-1}\left[2\,\text{diag}(V) - V + B(Y)\right]Y\right) \\ &= c(Y) + \left\|X - \left[2\,\text{diag}(V)\right]^{-1}\left[2\,\text{diag}(V) - V + B(Y)\right]Y\right\|^2_{2\,\text{diag}(V)}.\end{aligned}$$

Diagonal Majorization Algorithm

The minimizer of $\bar{\omega}_Y(X)$ is

$$\begin{aligned} X_* &= [2\,\mathsf{diag}(V)]^{-1}\,[2\,\mathsf{diag}(V) - V + B(Y)]\,Y \\ &= Y + \frac{1}{2}\,\mathsf{diag}(V)^{-1}\,[B(Y) - V]\,Y. \end{aligned}$$

DMA sets $Y = X_k$ and computes

$$X_{k+1} = X_k + \frac{1}{2}\,\mathsf{diag}(V)^{-1}\,[B\,(X_k) - V]\,X_k.$$

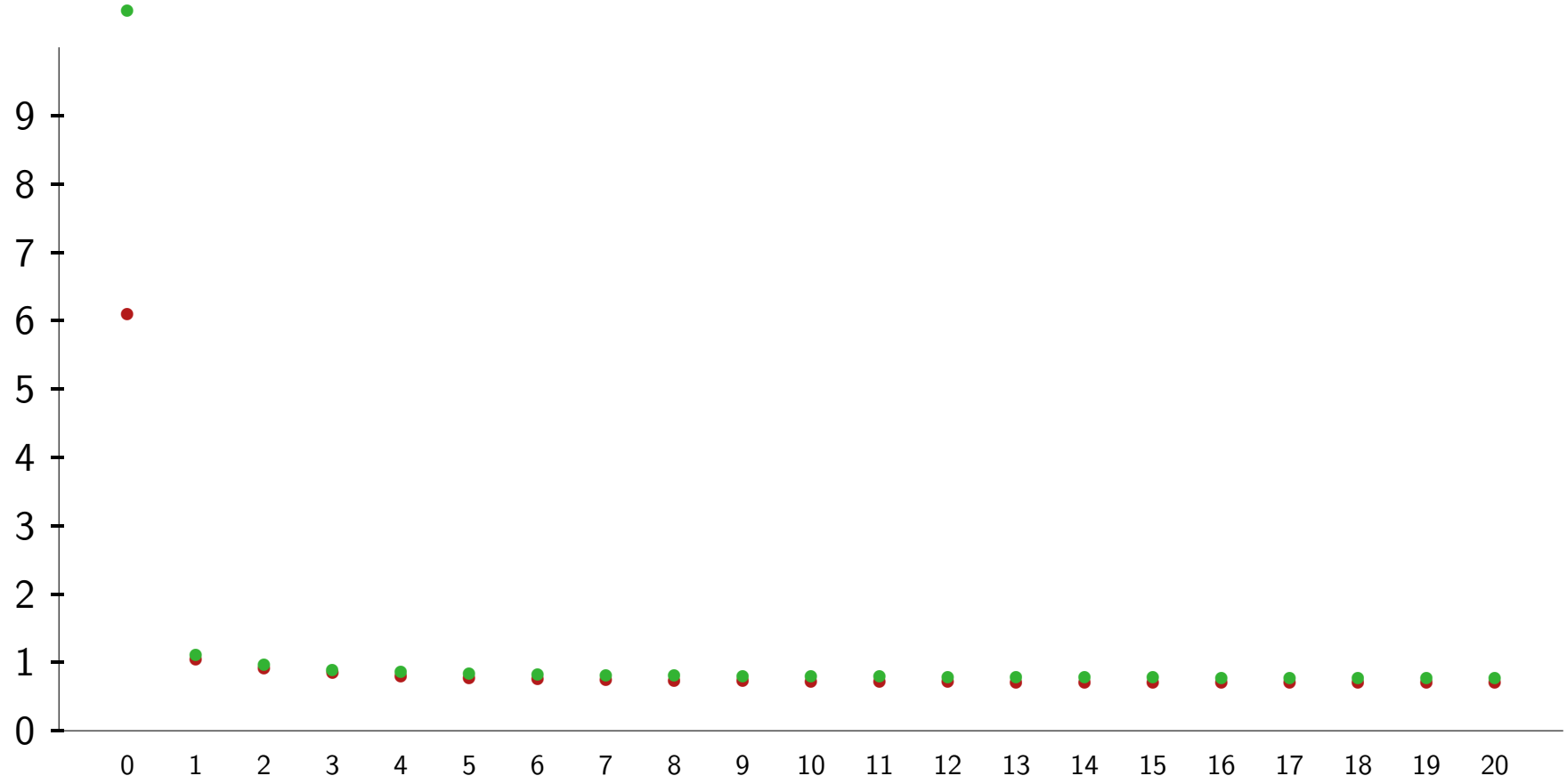
After computing $[B(X_k) - V]X_k$, DMA requires $2dn$ additional multiplications per iteration.

In contrast, after computing $B(X_k)X_k$, GMA with general w_{ij} requires dn^2 additional multiplications per iteration, plus an initial $n^3/6$ multiplications to compute a Cholesky factor.

Experiment 1

$n = 2818$ documents, $d = 5$, equal weights.

Raw Stress Criterion, 20 iterations of PCA-GMA v LIN-GMA,



How much does this cost?

Computational expense (in seconds):

	PCA-GMA	LIN-GMA
Data input	45.48	45.08
Preprocessing	0.16	0.16
Initial configuration	9.11	$<$0.01
Recover dissimilarities	0.32	
Initial stress evaluation	0.42	0.46
20 iterations w/ stress evaluation	24.28	24.16

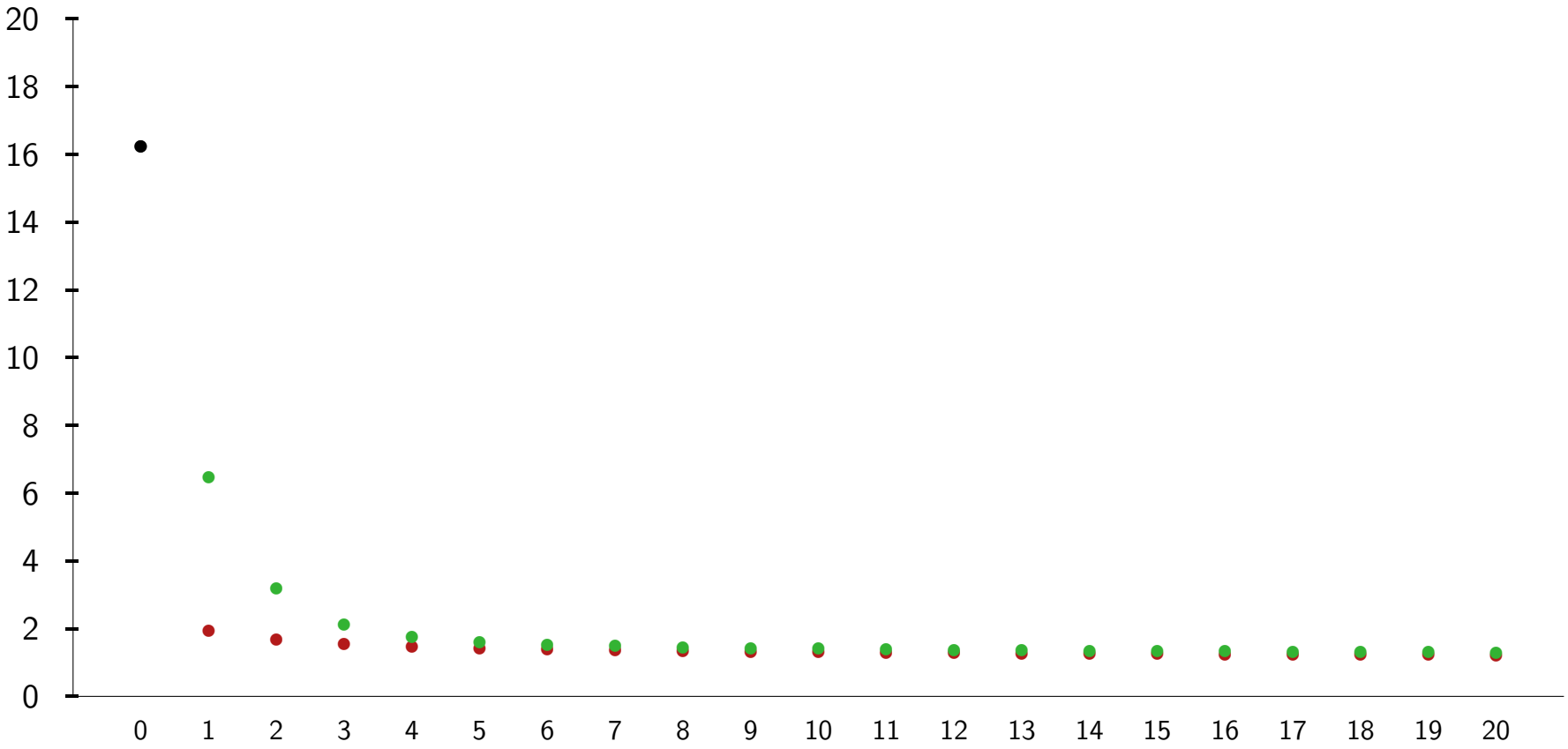
Notice that. . .

- PCA is surprisingly affordable, but orders of magnitude more expensive than LIN. LIN-GMA with one iteration is substantially less expensive and produces a substantially better configuration than PCA.

- Stress evaluation is expensive. Unlike general methods for numerical optimization, GMA does not require evaluation of the objective function—we only computed values in order to monitor progress. Eliminating this extravagance substantially decreases the expense of GMA.

Now let $w_{ij} = 0.4/(0.4 + \delta_{ij})$ when $\delta_{ij} < 0.6$ and $w_{ij} = 0$ otherwise, resulting in 58% of the pairs having zero weight.

Weighted Raw Stress Criterion, 20 iterations of LIN-GMA v LIN-DMA



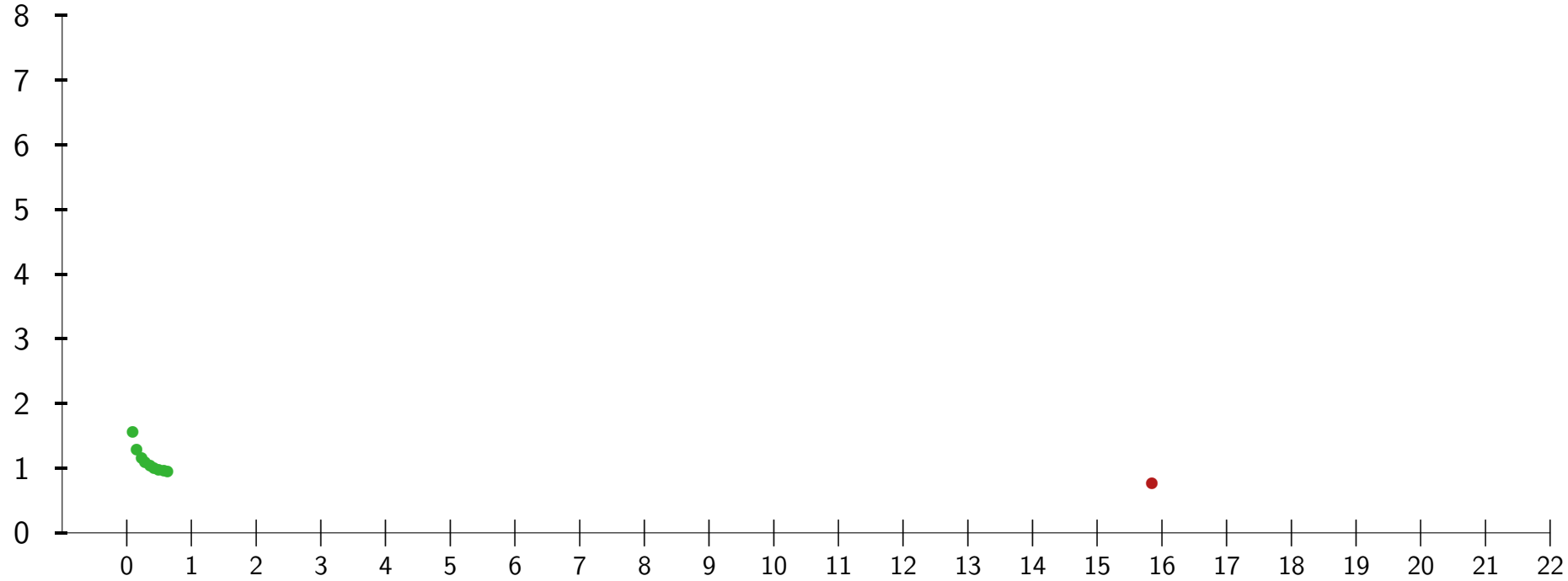
How much does this cost?

Computational expense (in seconds):

	LIN-GMA	LIN-DMA
Data input	45.23	45.54
Preprocessing	0.30	0.33
Initial configuration	0.01	0.01
Initial stress evaluation	0.26	0.28
Cholesky factorization	47.96	
20 iterations w/ stress evaluation	23.01	14.07

Finally, we set $w_{ij} = 1$ for $6k$ "cycles" of δ_{ij}, e.g., $i \leftrightarrow i \pm 1$, $i \leftrightarrow i \pm 2$, etc. and $w_{ij} = 0$ otherwise. (Thus, for $k = 7$, DMA uses $\approx 3\%$ of the 3969153 δ_{ij}.)

We compare the cpu-stress tradeoff for 20 iterations of LIN-GMA with all $w_{ij} = 1$ versus LIN-DMA with $k = 1 : 9$.



For d and k fixed, a fixed number of iterations of LIN-DMA requires $O(n)$ operations. We expect the tradeoff to increasingly favor LIN-DMA as n increases.

Experiment 2

$n = 17,000$ objects, $n(n-1)/2 = 144,491,500$ pairwise dissimilarities.

$y_1, \ldots, y_n \in \Re^5$; $\delta_{ij} = \|y_i - y_j\| \cdot \exp(Z/100)$, where $Z \sim$ Normal$(0,1)$.

With this model, a typical distance is 2 and a typical error is 1%, i.e., ± 0.02.

Embed $\Delta = [\delta_{ij}]$ in $\Re^5$ using LIN-DMA with 54 cycles.

Initial raw stress criterion:	3,058,174
After 200 DMA iterations:	57,921

A typical error in the initial configuration is ± 0.145; a typical error after 200 iterations is ± 0.020.

2 stress evaluations:	86.5 seconds
Total embedding time:	66.0 seconds