Statistical Geometry Processing
Winter Term 2011/2012

Assignment Sheet #2: Classical Multi-Dimensional Scaling

Author: Michael Wand
Contact: mwand@mpi-inf.mpg.de

Handed out: Nov. 15th 2011
Due: Nov. 30th 2011

Instructions

This assignment sheet contains theoretical (2.1) and practical (2.2) assignment. The solutions to the theoretical assignments should be written down and handed in on paper (during the tutorial course). As always, the practical assignments can be solved in group-work. In the tutorial course, we will discuss both theoretical and practical assignments. The write-ups will be graded separately, per person. For the write-ups, please do not just state the solution but also explain (briefly) *why/how* it solves the problem.

Important: Every student must hand in a separate write-up for the theory part!

Assignment 2.1 (Theory): Applications of Square Roots of Matrices (score: 60%)

(a) Consider the following problem.

- We have a consider of points $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$.
- We know only the *Gram matrix* $\mathbf{G}$ of the points. This is the matrix of inner products (a.k.a. scalar products) between all points:

$$\mathbf{G}_X := \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \ddots & \iddots & \\ \vdots & & \langle \mathbf{x}_i, \mathbf{x}_j \rangle & \vdots \\ & \iddots & \ddots & \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & & \cdots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{pmatrix}$$

- Our goal is to retrieve the original points $\mathbf{x}_i$, knowing *only* the Gram matrix $\mathbf{G}_X$.

How can we retrieve the points from the Gram matrix? Instructions: Write $\mathbf{G}$ as a product $\mathbf{X}^T\,\mathbf{X}$ of a suitable matrix $\mathbf{X}$. X can then be computed as the "square root" of $\mathbf{G}_X$, as discussed in the lecture.

(b) Now we change the problem slightly. Instead of $\mathbf{G}_X$, we are given the matrix $\mathbf{D}_X$ of pairwise distances between *all pairs* of points $\mathbf{x}_i$:

$$\mathbf{D}_X := \begin{pmatrix} \mathrm{dist}(\mathbf{x}_1, \mathbf{x}_1) & \mathrm{dist}(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \mathrm{dist}(\mathbf{x}_1, \mathbf{x}_n) \\ \mathrm{dist}(\mathbf{x}_2, \mathbf{x}_1) & \ddots & \iddots & \\ \vdots & & \mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j) & \vdots \\ & \iddots & \ddots & \\ \mathrm{dist}(\mathbf{x}_n, \mathbf{x}_1) & & \cdots & \mathrm{dist}(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

How can we retrieve the original points X from this matrix? Instructions: Distances can be computed from inner products.

Hint: As the title of this assignment sheet suggests, this is a classical problem. If you get stuck, Google is your friend.

(c) When retrieving points X from $\mathbf{D}_X$ information is lost. What kind or information *cannot* be reconstructed from pairwise distances only? In other words, under which class of the transformations are the reconstructions invariant?

(d) In machine learning, this technique is frequently used to layout points according to a customly defined distance function (and/or custom inner product). This technique is known as *kernel principal component analysis (kernel PCA*; we will see later in the lecture why this name is justified). However, not every distance function/inner product can be used. Can you name a restriction on $\mathbf{G}_X$ that is necessary for your algorithm to work?

Remark: For a general distance function, an exact embedding (matching the prescribed distances) might only be possible in $\mathbb{R}^n$ (Why is this always possible?) However, least-squares optimal approximations in $\mathbb{R}^k$ can be obtained by using only the first k coordinates of the reconstruction (after sorting eigenvalues by decreasing magnitude). This technique is used for *dimensionality reduction* (projecting point sets to lower-dimensional spaces while preserving pairwise distances as prescribed).

Assignment 2.2 (Practice): Now let's try this… (score: 40%)

(a) Implement a GeoX experiment that allows you to input a 2D point set (for example by drawing with the mouse and recording clicked points / mouse "move" events). Create some example shapes this way.

(b) Compute all pairwise distances and implement the "classic MDS" algorithm developed above to reconstruct the original shape.

(c) Kernel PCA: Try a custom distance function or inner product and experiment with the result. For example, the Gaussian kernel

$$\langle \mathbf{x}, \mathbf{y} \rangle_{Gauss} := \exp\left(-\frac{(\mathbf{x}-\mathbf{y})^2}{\sigma^2} \right)$$

is frequently used in machine learning as replacement for the standard scalar product. One can prove that this "kernel" (i.e., symmetric two-parameter function) meets all the requirements for the MDS algorithm to succeed, as discussed in Assignment 2.1(d). The parameter σ is typically chosen to be in the range of a small factor times the point spacing of the input point set (say 2×…3×).

Remarks

Support: In case you run into difficulties solving this assignment (conceptually or implementation-wise), do not hesitate to contact us. The easiest way to get feedback is to contact us via email to make an appointment for a personal meeting.