**DATA.STAT.770 Dimensionality Reduction and Visualization, Spring 2021, Exercise set 7**

# Part E: Nonlinear Dimensionality Reduction

## Problem E1: Intrinsic Dimension Estimation by the Box-counting Dimension

Suppose we have a (large) set of data samples in a five-dimensional space: the first three coordinates of the data samples are uniformly distributed over a three-dimensional box of width $w$, height $h$, and length $l$. The remaining two data coordinates are zero for all data samples. Use the Box-counting dimension (see lecture 6) to show that the intrinsic dimension of the data set is three.

You do not need to run any computational experiments in this exercise, the data definition is simple enough that you can show the result as a mathematical proof. If you cannot show it mathematically, you can alternatively create a data set corresponding to the definition, and estimate the result from the data using any suitable programming language.

## Problem E2: Intrinsic Dimension Estimation by Local PCA

The file `swillroll.dat` included with this exercise pack contains 1000 samples from a swiss roll, which is a two-dimensional manifold embedded in a three-dimensional space (like a sheet of paper folded into a roll).

Let us use the Local Principal Component Analysis approach (see lecture 6) to estimate the intrinsic dimension of the swiss roll data. You can use any suitable software (e.g. matlab, R, Python) to perform this exercise.

In detail: the data coordinates are within the unit cube (i.e. each coordinate is between 0 and 1). Divide each coordinate axis into $k = 5$ intervals, so that the unit cube becomes divided into $k \times k \times k$ small cubes. For each small cube, if it contains at least 5 samples, perform PCA on those samples (estimate the covariance matrix, and perform an eigenvalue decomposition for it) and count how many top eigenvalues are needed to explain at least 90% of the variance: that is, count how many $l$ top eigenvalues are needed so that $(\sum_{i=1}^{l} \lambda_i)/(\sum_{i=1}^{3} \lambda_i) \geq 0.9$. Report the mean of the estimated dimensions over the small cubes.

## Problem E3: Various Suggestions for Nonlinear Dimensionality Reduction Cost Functions

The properties of a nonlinear dimensionality reduction method largely depend on whether its cost function is meaningful, and on whether its optimization algorithm can find a good enough optimum for its cost function.

Suppose we have a data set of $n$ original high-dimensional data points $\mathbf{x}_i$ with some dimensionality $d_{high}$. The suggested cost functions below each aim

to minimize a cost with respect to a corresponding set of $n$ lower-dimensional coordinates $\mathbf{y}_i$, $i = 1, \ldots, n$, with some dimensionality $d_{low}$. However, all but one of the cost functions has a problem. If there is a problem, what is the problem? Are the definitions valid definitions of a cost function? If they are, what will happen if you try to use these cost functions as a goal of nonlinear dimensionality reduction?

1. Cost function 1:
$$Cost = \sum_i \sum_{j \neq i} ||\mathbf{y}_i - \mathbf{y}_j||^2 - \sum_i \sum_{j \neq i} ||\mathbf{x}_i - \mathbf{x}_j||^2$$

2. Cost function 2:
$$Cost = \sum_i \sum_{j \neq i} (||\mathbf{x}_i - \mathbf{x}_j||^2 - ||\mathbf{y}_i - \mathbf{y}_j||^2)^2$$

3. Cost function 3:
$$Cost = \sum_i ||\mathbf{x}_i - \mathbf{y}_i||^2$$

4. Cost function 4: let $\mathbf{x}_{Reference}$ be a chosen "reference point" such as the center of the data set, let $\mathbf{y}_{Reference}$ be its corresponding low-dimensional coordinates, and set
$$Cost = \sum_i (||\mathbf{x}_i - \mathbf{x}_{Reference}||^2 - ||\mathbf{y}_i - \mathbf{y}_{Reference}||^2)^2$$

## Problem E4: Multidimensional Scaling

Suppose you have $n$ high-dimensional data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and want to use absolute multidimensional scaling (absolute MDS; see lecture 6) to find low-dimensional coordinates $\mathbf{y}_1, \ldots, \mathbf{y}_n$ for them. The absolute multidimensional scaling cost function is

$$Cost_{Absolute-MDS} = \sum_{i<j} (f(p_{ij}) - d_{ij}(X))^2$$

where $f(p_{ij}) = p_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ and $d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j|| = \sqrt{(\mathbf{y}_i - \mathbf{y}_j)^\top (\mathbf{y}_i - \mathbf{y}_j)}$. Thus the cost function can be rewritten as

$$\sum_{i,j} \left( ||\mathbf{x}_i - \mathbf{x}_j|| - \sqrt{(\mathbf{y}_i - \mathbf{y}_j)^\top (\mathbf{y}_i - \mathbf{y}_j)} \right)^2 .$$

While absolute MDS if often done with a linear projection, and is then equivalent to PCA, it is also possible to optimize each output coordinate separately to achieve a nonlinear mapping. In this exercise we will do that.

Derive the gradient of the cost function with respect to the output coordinates $\mathbf{y}_m$ of the $m$th data point. The MDS cost function must be minimized,

which can be done by moving each point in the opposite direction of the gradient (that is, in the direction of the negative gradient). Can you interpret the (negative) gradient as a sum of forces pulling the $m$th point towards or away from the other points $j$? When does the gradient pull $m$ towards another point $j$, when does it pull away from the point? When is the gradient zero with respect to $m$?

Hint: terms involving only the $\mathbf{x}$ coordinates are constants. Note that $m$ can be involved as either index term $i$ or $j$ in the double sum.

## Problem E5: Multidimensional Scaling versus Sammon's Mapping, Shepard diagram

Familiarize yourself with the Matlab toolbox for dimensionality reduction, available at `https://lvdmaaten.github.io/drtoolbox/`. It is also possible to use the toolbox with Octave instead of Matlab, instructions for doing so are at the bottom of this exercise; if you encounter any problems contact the lecturer.

Unfortunately currently we are not aware of a fully corresponding R toolbox, but many methods exist in R, see notes at the end of the problem description.

To start using the toolbox, you will need to recompile some of its functions. To do so, start Matlab or Octave and type the following lines on the Matlab/Octave command line:
`cd myinstalldir/drtoolbox/techniques` (where myinstalldir is the directory where you uncompressed the toolbox archive)

```
mex computegr.c
mex kernel_function.c
mex mexCCACollectData.c
mex mexCCACollectData2.c
addpath myinstalldir/drtolbox/
addpath myinstalldir/drtolbox/techniques/
```

You may need to type the last two lines every time you restart Matlab/Octave.

The toolbox implements several dimensionality reduction methods including a nonlinear version of absolute Multidimensional Scaling (denoted NMDS in the toolbox) and Sammon's Mapping. Here we will compare NMDS and Sammon's mapping to see how the difference between their cost functions affects the results of dimensionality reduction.

Use NMDS and Sammon's Mapping to reduce artificial data sets included with the exercise package to two dimensions:

- The three-dimensional swiss roll data set `swissroll.dat`
- The three-dimensional "two interlocked rings" data set `tworings.dat`
- The three-dimensional half-sphere surface data set `halfsphere.dat`
- The four-dimensional data set `foursquares.dat` where four clusters of data each vary as a square along dimensions 1 and 2, but the clusters are separated from one another along dimensions 3 and 4.

Plot the results for each method and each data set.

For both the NMDS and Sammon's Mapping result, draw the Shepard diagram (see lecture 6). That is, plot the set of distances between pairs of data points in the output space as a function of the corresponding distances (proximities) in the input space. How do the Shepard diagrams differ between the two methods? Does the difference correspond to the way the cost functions of the methods have been defined? (Hint: these are easy data sets, the difference is not very big. The difference is clearest in the four squares data set. To make the individual values easier to see, try plotting only a subset of the distances, same subset for each method. Try also plotting the Shepard diagrams of the two methods onto the same plot.)

**Instructions for using the toolbox with Octave.** Octave is a free program whose syntax is essentially identical to that of Matlab. Octave is available at `https://www.gnu.org/software/octave/`. If you are running Octave on a MacOS X Mavericks computer or laptop, you may need to install the XCode developer tools in order to compile some functions needed in the toolbox. To do so, install XCode from the Mac App Store, and then run the command:
`xcode-select --install`

**Nonlinear dimensionality reduction methods in R.** There does not seem to be a good integrated R package with several nonlinear dimensionality reduction methods. A few separate packages exist for individual methods:

- Package 'lle' implements the locally linear embedding algorithm.
- Package 'kohonen' implements the self-organizing map algorithm.
- Classical multidimensional scaling can be done with the cmdscale() function and nonmetric multidimensional scaling with the isoMDS function in the 'MASS' package.
- Sammon mapping can be done with the sammon function in the 'MASS' package.
- The package 'dimRed' implements LLE, Hessian LLE (variant of LLE), Isomap, Kernel PCA, Laplacian eigenmap, Fruchterman-Reingold and Kamada-Kawai graph embeddings, FastICA (an ICA method), MDS and nMDS, tSNE, and others.