Dimensional Reduction for Data Mapping

A practical guide using R

by Jonathan Edwards and Paul Oman

Introduction

Techniques that produce two dimensional maps of multi-variate data are of increasing importance, given the recent advances in processing capability, data collection and storage capacity. These techniques are mostly used in an exploratory capacity, to highlight relationships within the data-set, often as a precursor to further analysis. An example usage on a data-set of shape descriptors is given in Edwards et al. (2003). There are many statistical approaches to this problem, and this article serves as an initial overview and practical guide to their application in R. A small comparative study is also presented, to highlight the ease with which the analysis can be performed, and to demonstrate the capabilities of several of the more established techniques. The interested reader is directed to Ripley (1996) for thorough treatments and derivations.

Principal component analysis

Principal Component Analysis (PCS, Pearson, 1901) is the most widely used general dimension reduction technique, and can easily be applied to mapping multi-variate data. PCA is a centring (to mean), scaling (to variance) and rotation (to principal axes) produced by an orthogonal linear transform (equation 1) of the data ($\{\mathbf{x}\}_{i=1}^{n} \in \mathbb{R}^D$), with the aim of generating a series of uncorrelated variables ($\mathbf{y}$).

$$\mathbf{y} = \mathbf{U}^{\mathbf{T}}(\mathbf{x} - \bar{\mathbf{x}}) \tag{1}$$

A standard solution to finding $\mathbf{U}^{\mathbf{T}}$, and hence $\mathbf{y}$, is the spectral decomposition ($\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathbf{T}}$) of the covariance matrix [1] ($\mathbf{\Sigma} = \mathbb{E}((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T)$), where$\mathbf{\Lambda}$ is the diagonal eigenvalue matrix, with eigenvalues ordered $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$. For dimension reduction to dimension L (for mapping, normally 2), PCA has the desirable property that in a linear sense, the largest variance is explained by the first L components. Due to its ubiquity and also because it forms a basis for other approaches, many packages implement PCA. In practice they mainly rely on the **base** routines of `svd` and `eigen` to perform eigen-decomposition [2]. The author's preferred implementations are the `prcomp` and `princomp` (with formula interface) functions supplied by the **mva** package (The interested reader should also examine the **ade4** package). These functions make the application of PCA to dataframes a trivial task. The following lines of R code perform PCA on the data-set `g54` (which is used as data in all the proceeding examples, see section "data generation" at the end of this article for details):

```
> out.pca   <- prcomp(g54[,1:4])
```

If we want to use correlations instead of covariances, scaling is required, so the function call becomes `prcomp(g54[,1:4], scale=TRUE)`.

The mapping process is then simply the plotting of the first two principal components. This can be performed using a `biplot`, or via the standard `plot` call

```
> plot(out.pca$x[,1:2],
+   main ="prcomp")
```

see Figure 1.

Multi-dimensional scaling

Multi-Dimensional Scaling (MDS) is a method of embedding the distance information of a multi-variate data-set, in a reduced dimension L, by seeking a set of vectors ($\{\hat{\mathbf{x}}\}_{i=1}^{n} \in \mathbb{R}^L$) that reproduce these distances. One of the advantages of this direct approach is that an arbitrary distance metric can be used in the original multi-variate domain. Both metric and non-metric approaches to MDS have been developed. Metric MDS aims to embed the distance directly in to the mapping domain. The classical analytical approach (Gower, 1966) (often referred to as Principal Co-ordinate Analysis (PCoA)) utilises the first two components of the spectral decomposition of the distance matrix ($\mathbf{U}\mathbf{\Sigma}^{\frac{1}{2}}$ generates the co-ordinates) to explain the maximal distance. If Euclidean distance is used then this approach is equivalent to covariance based PCA (see (Webb, 2002) page 346). Sammon mapping (Sammon jnr, 1969) is a more commonly used approach. A distortion measure (in MDS parlance referred to as *Stress*) is used to compare the data-set distances with an equal number of randomly distributed, two-dimensional points in the mapping space. The Stress measure S is represented by,

$$S_{sam} = \frac{1}{\sum_{i<j} d_{ij}^2} \sum_{i<j} \frac{(d_{ij} - \hat{d}_{ij})^2}{d_{ij}} \tag{2}$$

[1] Correlation can also be used if one requires scaled variables in $\mathbf{x}$

[2] These techniques are $O(D^3)$ (Carreira-Perpinan, 1997, page 10). It appears that MATLAB versions of these functions allow the user to select the number of eigenvectors/values calculated (Nabney, 2001, page 230), the author is interested in how these approaches might be adopted in R.

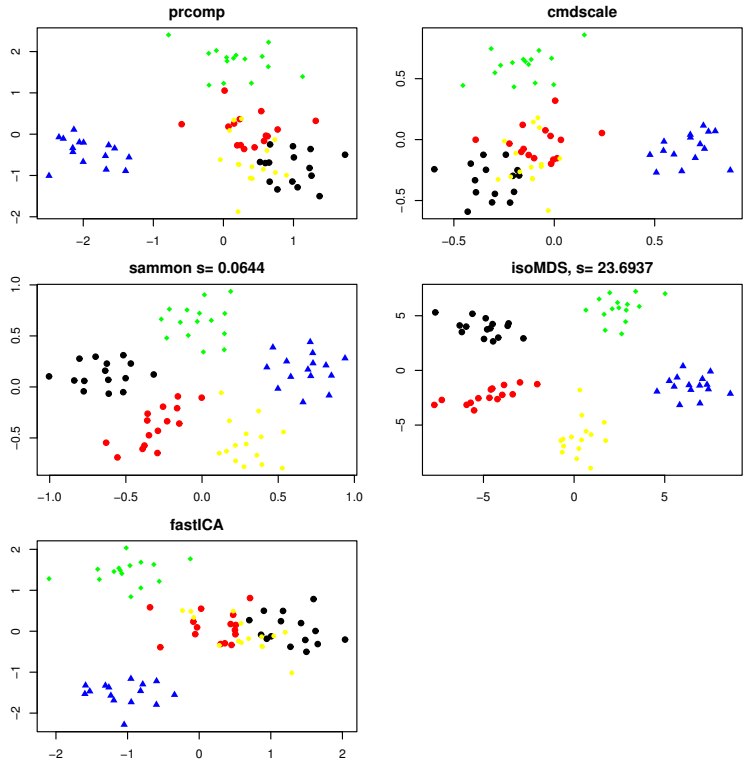

Figure 1: Maps of the Gaussian5 data-set (5 multivariate Gaussian clusters in 4-dimensional space)

d_{ij} is the distance vectors i and j,$\hat{d}$ is the estimated distance in dimension L. S_{sam} is minimised using an appropriate iterative error minimisation scheme, Sammon proposed a pseudo-Newtonian method with a step size, α normally set to 0.3. Ripley ((Ripley, 1996) page 309) discusses a crude but extremely effective line search for step size, which appears to significantly enhance the optimisation process.

Non-Metric MDS produces a map that maintains the rank of distances within the original data-set. The most widely known approach is due to Kruskal (Kruskal, 1964) which uses a similar iterative Stress function minimisation to the Sammon map, the stress function

$$S_{kruskal} = \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{i<j} \hat{d}_{ij}^2}} \qquad (3)$$

which is again minimised, using a suitable gradient descent technique.

All the MDS techniques discussed above (including Ripley's adaptive step-size optimisation scheme) are supported in R as a part of the **mva** and **MASS** packages (classical MDS (`cmdscale`), Sammon mapping (`sammon`) and non metric MDS (`isoMDS`)). Additionally, a further implementation of the Sammon map is available in the **multiv** package, however, this appears to be less numerically stable. All the MDS functions utilise a distance matrix (normally generated using the function `dist`) rather than a matrix or data.frame, hence there is considerable flexibility to employ arbitrary distance measures. Although the function's defaults are often adequate as a first pass, the two iterative techniques tend to require a more forceful optimisation regime, with a lower "desired" stress, and more iterations, from a random initial configuration, to achieve the best mapping. The R code below performs all three mapping techniques, using the more rigorous optimisation strategy

```
> dist.data <- dist(g54[,1:4])
> out.pcoa  <- cmdscale(dist.data)
> randstart <- matrix(runif(nrow(g54[,1:4])*2),
+   nrow(g54[,1:4]),2)
> out.sam <- sammon(dist.data,y=randstart,
+   tol=1e-6,niter=500)
> out.iso <- isoMDS(dist.data,y=randstart,
+   tol=1e-6,maxit=500)
```

Again, plotting can be done with the standard R plotting commands.

```
> plot(out.pcoa, main="cmdscale")
> plot(out.sam$points), ,ylab="",xlab="",
+   main=paste("sammon s=",as.character(
+   round(out.sam$stress,4))))
> plot(out.iso$points,xlab="",ylab="",
+   main=paste("IsoMDS, s=",
+   as.character(round(out.iso$stress,4))))
```

It is also worthwhile to produce a distortion plot (see Figure 2), this plots the original distances against the mapped distances, and can be used to assess the accuracy of distance reproduction in L.

```
> plot(as.matrix(dist.data),
+    as.matrix(dist(out.sam$points)),
+    main="Distortion plot: Sammon map",
+    ylab="2d distance",xlab="nd distance")
```

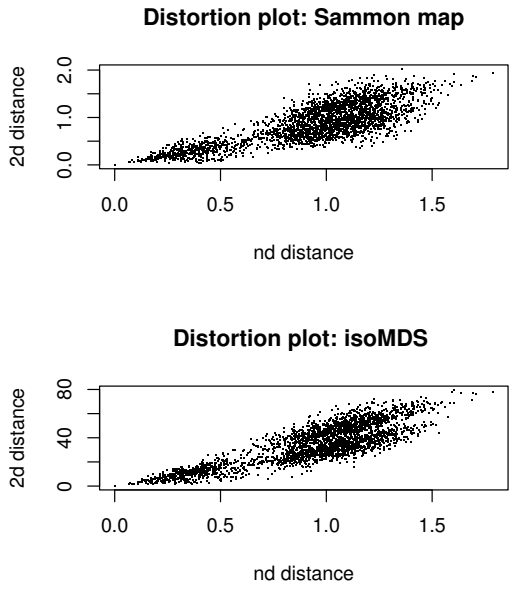

Figure 2: Distortion plot for the Gaussian5 data-set. Note pch=20 was used.

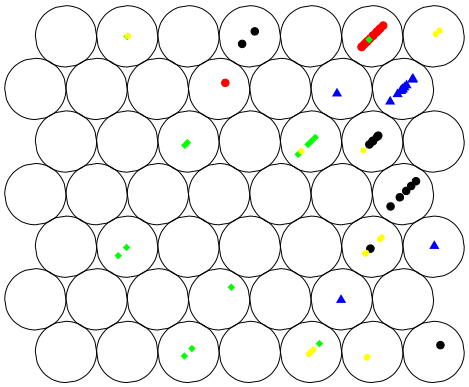

Figure 3: SOM of the Gaussian5 data-set.

The self organising map

The Self Organising Map (SOM) (Kohonen, 1982) is a variation on the K-means algorithm, which generates a map (typically a hexagonal lattice) of topologically arranged cluster centroids. The algorithm proceeds in the same manner as K-means with a *winner-takes-all* approach to updating the centroids (m_i where i is some positional reference). The update rule is

$$m_i(t+1) = m_i(t) + h_{ci}[x(t) - m_i(t)] \qquad (4)$$

Neighbourhoods h_{ci} are normally calculated as $h_{ci} = \alpha(t)$ if $i \epsilon N$ else 0, where N is a set of centroids in proximity to the "winning" centroid. α is the learning rate, which is decreased to schedule the optimisation of the map. There are two packages that implement SOM's, **som** and **class**. These packages both appear to be interfaces to the standard SOMPAK C library (Kohonen et al., 1996).

Once the map has settled, several visualisation approaches can be applied (see (Vesanto, 1999) for overview). The most straightforward method in R (suggested in (Ripley, 1996)) is to allocate, via *nearest neighbour* search, each example in the data-set to a cluster centre. The following R code, using **class** (adapted from the SOM help file in R) produces a SOM and maps it using the nearest neighbour approach (see Figure 3):

```
> gr <- somgrid(topo = "hexagonal",
+    xdim=5,ydim=5)

> out.som <- SOM(g54[,1:4], gr, alpha =
+    list(seq(0.05, 0, len = 1e4),
+    seq(0.02, 0, len = 1e5)),
+    radii = list(seq(8, 1, len = 1e4),
+    seq(4, 1, len = 1e5)))

> plot(out.som$grid, type = "n")

> symbols(out.som$grid$pts[, 1],
+    out.som$grid$pts[, 2],
+    circles = rep(0.4, 25),
+    inches = FALSE, add = TRUE)

> bins <- as.numeric(knn1(out.som$codes,
+    g54[,1:4], 0:24))

> points(out.som$grid$pts[bins, ] +
+    rnorm(nrow(g54), 0, 0.1),
+    col=classcol[rep(1:5,e=15)],pch=20)
```

Independent component analysis and projection pursuit

Independent Component Analysis (ICA) (Jutten and Hérault, 1991) and Projection Pursuit (Huber, 1985) are variations on classical factor analysis incorporating a search for maximally "Interesting" projections. ICA takes the general form

$$\mathbf{x} = \mathbf{G}\mathbf{s} \qquad (5)$$

where $\mathbf{G}$ is the *mixing matrix*, and $\mathbf{s} \in \mathbb{R}^{L<D}$ are the latent variables.

The task then is to find $\mathbf{G}$ which satisfies the above equation, and maximises an "interestingness" function on $\mathbf{s}$, $I(\mathbf{s})$. Predictably, given the subjectivity of the term "interesting", there are many forms of $I(\mathbf{s})$, exponents of ICA and Projection Pursuit both agree that an important aspect is deviation from Gaussianity. Originally algorithms focused on maximising kurtosis based measurements, as these can be efficiently calculated. However, this approach is not robust to outliers, and hence a variety of other

approaches have been suggested (see (Hyvärinen, 1999b) for discussion on why non-Gaussianity is interesting, weaknesses of kurtosis and survey of alternative "interestingness" functions). A standard robust approach, which has been discussed in the literature of both Projection Pursuit and ICA, is to estimate (via approximation) the minimum mutual information via maximisation of the *negentropy*.

$$J(s) = (\mathbb{E}\, f(s) - \mathbb{E}\, f(v))^2 \qquad (6)$$

where v is a standard Normal *r.v*, and f is $f(u) = \log\cosh a_1 u$ ($a_1 \geq 1$ is a scaling constant). This method is used in the **fastICA** implementation of ICA (Hyvärinen, 1999a), and can be applied and plotted by the following R function calls:

```
> out.ica <- fastICA(g54[,1:4], 2,
+   alg.typ = "deflation",
+   fun = "logcosh", alpha = 1,
+   row.norm = FALSE, maxit = 200,
+   tol =0.0001)
```

again, the first two components (`out.ica$[,1:2]`)are plotted. XGobi and Ggobi (Swayne et al., 1998) are additional methods of performing Projection Pursuit, which although not strictly a part of R can be called using the functions in the R interface package **Rggobi**. This has less flexibility than a traditional R package, however there are some attractive features that are worth considering for exploratory multi-variate analysis, particularly the *grand tour* capability, where the data-set is rotated along each of it's multivariate axes.

A small comparative study

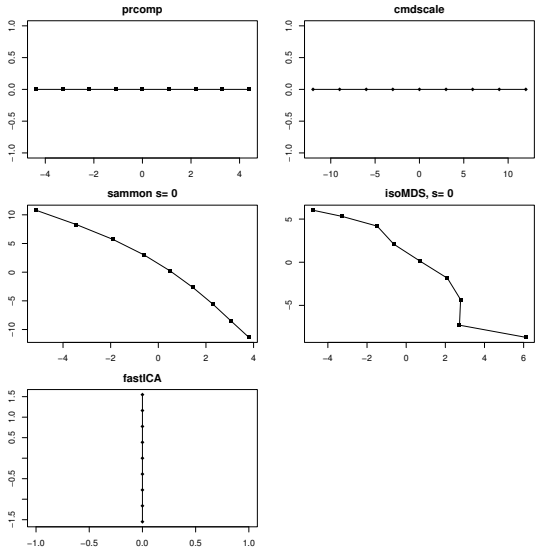

Figure 4: Maps for the Line data-set.

Data-set	*n,s*	c	Description
Line	9,9	x	line in 9d
Helix	3,30	x	3d spiral
Gaussian54	4,75	✓	5 Gaussians in a 4d simplex
Iris	4,150	✓	classic classifier problem

Table 1: The data sets, n, s are dimension and size.

To illustrate the capabilities of the above techniques (SOMs were not included as there are problems visualising geometric structures) a series of example maps have been generated using the data-sets from Sammon's paper (Sammon jnr, 1969)(see Table 1) [3]. These data-sets are used as they are easily understood and contrast the mapping performance. Interpretation of these results is fairly objective, for the geometric functions one requires a sensible reproduction, for data-sets with class information (labeled 'c' in the table) a projection that maintains cluster information is desirable. For the iterative MDS techniques I will admit repeating the mapping process a few times (maximum 4!).

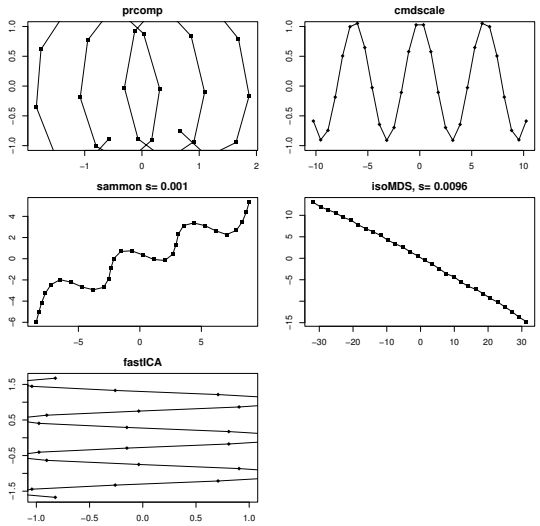

Figure 5: Maps for the Helix data-set.

Results

Line Figure 4, all techniques generate a line, apart from Non-Metric MDS which has greater freedom in the positioning of points, and hence produces a somewhat "wavy" line.

Helix Figure 5, this is an excellent data-set for a comparative study as the data is easily understood, and each algorithm gives a different interpretation. At a fundamental level the non-metric MDS (`isoMDS`) approach gives the correct interpretation since the helix is expressed in it's most "compressed" form. This approach has done too well! and isn't using all of the available dimensions. Of the other mapping techniques, most capture the sinusoidal vari-

[3] The non-linear helix data-set described in Sammon's paper has been omitted from this study as the results are similar to the those for the Helix data-set

ation (especially PCA), and hence subjectively present a better "story".

Gaussian5 Figure 1, again this data is excellent for highlighting the power of MDS techniques. *If* a reasonable minima can be found - MDS techniques offer a significant advantage in maintaining clustering information. One of the main characteristics of a cluster is that members tend to be relatively close together, hence MDS, fundamentally a *distance* based approach, has a greater capacity to maintain this relationship.

Iris Figure 6, there appears to be only minor difference in the maps produced. Of note perhaps is the tighter clustering in the MDS based approaches.

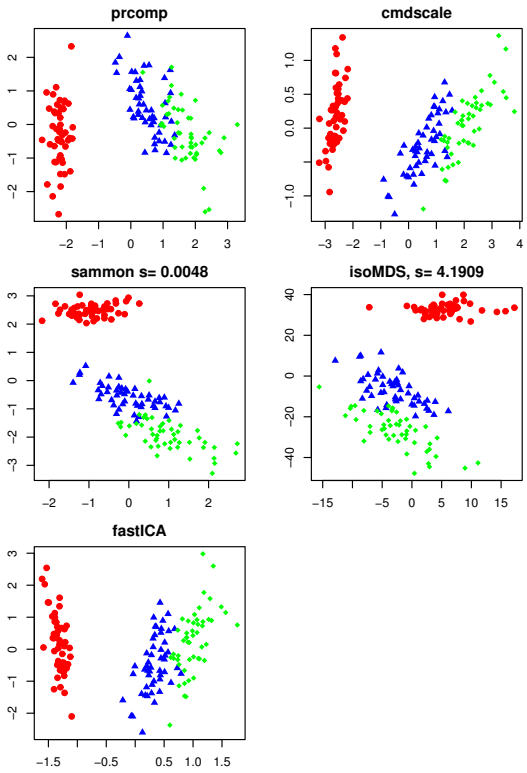

Figure 6: Maps of the Iris data-set.

Summary

As the results of the above study show, R is an excellent tool to perform dimension reduction and mapping. The **mva** package in particular provides an excellent set of classical approaches which often give the most revealing results, particularly when data contains class information. **fastICA** and the `SOM` function from **class** provide a welcome addition to the repertoire, even if they are perhaps aimed more at alternative problems. Clearly, there is scope to extend the number of projection pursuit indices, in line with those implemented in XGobi.

Dimensional reduction is an extremely active area of research, mainly within the Neural Network community. Few comparative practical studies of the application of recent techniques to mapping exist (particularly between newer techniques and "classical" MDS approaches), hence there is no strong indication of what might be usefully ported to R. A free MATLAB implementation of many of the newer techniques (Probabilistic Principal Component Analysis (Tipping and Bishop, 1997) (PPCA) mixtures of PPCA (Tipping and Bishop, 1999), Generative Topographic Mapping (GTM) (Bishop et al., 1998) and NeuroScale (Tipping and Lowe, 1998)), with excellent supporting documentation is provided by the NETLAB package (Nabney, 2001). This highlights some of the advances that have yet to appear in R, most notably techniques that are derived from a Bayesian prospective. Further notable omissions are Local Linear Embedding (Saul and Roweis, 2003) and Isomap (Tenenbaum et al., 2000) which generate a series of local representations, which are then linked together into a global mapping.

Data generation

The data-sets used in this study were generated using the following functions:

```
helix <- function( size = 30 )
{
# input :  size, sample size
# output:  3d helix
    t <- 0:(size-1)
    z <- sqrt(2)/2*t
    y <- sin(z)
    x <- cos(z)
    cbind(x,y,z)
}

gauss54 <- function ( s=15 , sig=0.02 )
{
# input :  s, sample size (per Gaussian)
#          sigma, cluster _covariance_
# output:  5 4d Gaussian clusters

 simp4d <-  matrix( c(0,0,0,0,1,0,
 0, 0, 1/2, sqrt(3)/2, 0, 0, 1/2,
 sqrt(3)/6 ,sqrt(2/3),0 ,1/2 ,sqrt(3)/6 ,
 sqrt(2)/(4*sqrt(3)),sqrt(5/8))
 ,5,4,byrow=T)
 #simplex4d can easily be checked
 #using 'dist'
 rbind(
 mvrnorm(simp4d[1,],S=diag(4)*sig,n=s),
 mvrnorm(simp4d[2,],S=diag(4)*sig,n=s),
 mvrnorm(simp4d[3,],S=diag(4)*sig,n=s),
 mvrnorm(simp4d[4,],S=diag(4)*sig,n=s),
 mvrnorm(simp4d[5,],S=diag(4)*sig,n=s))
}
```

```
l9  <- matrix(rep(1:9,9),9,9)
h30 <- helix()
#g54, last column is class label
g54 <- cbind(gauss54(),rep(1:5,e=15))
#iris data is a standard data-set
data(iris)
#check for duplicates else
#distance algorithms 'blow up'
iris.data <- unique(iris[,1:4])
iris.class <- iris[row.names(iris.data),5]
#some colours for the maps
classcol <- c("red","blue","green",
"black","yellow")
```

Bibliography

C. M. Bishop, M. S., and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998. URL citeseer.nj.nec.com/bishop98gtm.html. 6

M. Carreira-Perpinan. A review of dimension reduction techniques. Technical Report CS–96–09, Dept. of Computer Science, University of Sheffield, January 1997. 2

J. Edwards, K. Riley, and J. Eakins. A visual comparison of shape descriptors using multi-dimensional scaling. In *CAIP 2003, the 10th interantional conference on computer analysis of images and patterns*, pages 393–402, 2003. 2

J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, (53):325–328, 1966. 2

P. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985. 4

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999a. URL citeseer.nj.nec.com/hyv99fast.html. 5

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999b. 5

C. Jutten and J. Hérault. Blind separation of sources. *Signal Processing*, 24:1–10, 1991. 4

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982. 4

T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. Som pak: The self-organizing map program package, 1996. URL citeseer.nj.nec.com/kohonen96som.html. 4

J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 1-27(29):115–129, 1964. 3

I. Nabney. *NETLAB: Algorithms for Pattern Recognition*. Springer, 2001. 2, 6

K. Pearson. Principal components analysis. *London Edinburgh and Dublin Philosophical Magazine and Journal*, 6(2):559, 1901. 2

B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996. 2, 3, 4

J. Sammon jnr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18: 401–409, 1969. 2, 5

L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4: 119–155, 2003. 6

D. F. Swayne, D. Cook, and A. Buja. XGobi: Interactive dynamic data visualization in the X Window System. *Journal of Computational and Graphical Statistics*, 7(1):113–130, 1998. URL citeseer.nj.nec.com/article/swayne98xgobi.html. 5

J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22):2319–2323, 2000. 6

M. Tipping and C. Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, 1997. 6

M. Tipping and D. Lowe. Shadow targets: a novel algorithm for topographic projections by radial basis functions. *Neurocomputing*, 19(1):211–222, 1998. 6

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999. URL citeseer.nj.nec.com/tipping98mixtures.html. 6

J. Vesanto. SOM-based data visualization methods. *Intelligent-Data-Analysis*, 3:111–26, 1999. URL citeseer.nj.nec.com/392167.html. 4

A. Webb. *Statistical Pattern Recognition*. Wiley, 2002. ISBN 0470845139. 2

Jonathan Edwards & Paul Oman
Department of Informatics, University of Northumbria
Newcastle upon tyne, UK
{jonathan.edwards,paul.oman}@unn.ac.uk