

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/271642170>

Principal Manifolds for Data Visualisation and Dimension Reduction, LNCSE 58

Book · January 2008

CITATIONS

91

READS

3,377

4 authors, including:



Balázs Kégl

French National Centre for Scientific Research

174 PUBLICATIONS 6,261 CITATIONS

[SEE PROFILE](#)



Donald C. Wunsch

Missouri University of Science and Technology

394 PUBLICATIONS 10,482 CITATIONS

[SEE PROFILE](#)



Andrei Zinov'yev

Institut Curie

269 PUBLICATIONS 3,690 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Apical ISP [View project](#)



Apical CV engine (Spirit) [View project](#)

Preface

In 1901, Karl Pearson [1] explained to the scientific community that the problem of data approximation is (i) important, (ii) nice, and (iii) differs from the regression problem. He demonstrated how to approximate data sets by straight lines and planes. That is, he invented Principal Component analysis (PCA). Why and when do we need to solve the data approximation problem instead of regression? Let us look on the Pearson explanation:

“(1) In many physical, statistical, and biological investigations it is desirable to represent a system of points in plane, three, or higher dimensioned space by the “best-fitting” straight line or plane. Analytically this consists in taking

$$y = a_0 + a_1x, \text{ or } z = a_0 + a_1x + b_1y,$$
$$\text{or } z = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n,$$

where $y, x, z, x_1, x_2, \dots, x_n$ are variables, and determining the “best” values for constants $a_0, a_1, b_1, a_0, a_1, a_2, \dots, a_n$ in relation to the observed corresponding values of the variables. In nearly all the cases dealt with in the text-books of least squares, the variables on the right of our equations are treated as the independent, those on the left as the dependent variables. The result of this treatment is that we get one straight line or plane if we treat some one variable as independent, and a quite different one if we treat another variable as the independent variable. There is no paradox about this; it is, in fact, an easily understood and most important feature of the theory of a system of correlated variables. The most probable value of y for a given value of x , say, is not given by the same relation as the most probable value of x for the given value of y . Or, to take a concrete example, the most probable stature of a man with a given length of leg l being s , the most probable length of leg for a man with stature s will not be l . The “best-fitting” lines and planes ... depend upon

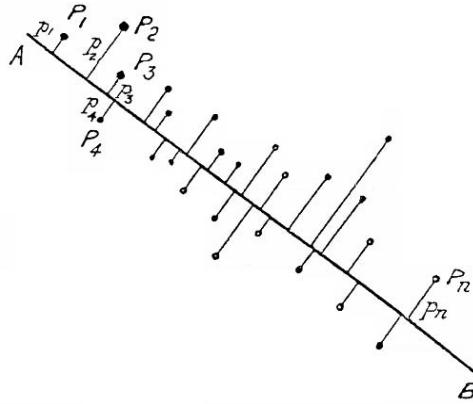


Fig. 1. Data approximation by a straight line. The famous illustration from Pearson's paper [1]

a determination of the means, standard-deviations, and correlation-coefficients of the system. In such cases the values of the independent variables are supposed to be accurately known, and the probable value of the dependent variable is ascertained.

(2) In many cases of physics and biology, however, the “independent” variable is subject to just as much deviation or error as the “dependent” variable. We do not, for example, know x accurately and then proceed to find y , but both x and y are found by experiment or observation. We observe x and y and seek for a unique functional relation between them. Men of given stature may have a variety of leg-length; but a point at a given time will have one position only, although our observation of *both* time and position may be in error, and vary from experiment to experiment. In the case we are about to deal with, we suppose the observed variables – all subject to error – to be plotted in plane, three-dimensioned or higher space, and we endeavour to take a line (or plane) which will be the “best fit” to such system of points.

Of course the term “best fit” is really arbitrary; but a good fit will clearly be obtained if we make the sum of the squares of the perpendiculars from the system of points upon the line or plane a minimum.

For example:—Let P_1, P_2, \dots, P_n be the system of points with coordinates $x_1, y_1; x_2, y_2; \dots, x_n, y_n$, and perpendicular distances p_1, p_2, \dots, p_n from a line \mathbf{AB} . Then we shall make¹

$$U = S(p^2) = \text{a minimum.}''$$

¹ $S(p^2)$ stands for $\sum_i p_i^2$

This explanation sounds very modern: in “many cases of physics and biology” there is significant noise in the “independent variables”, and it appears better to approximate data points than the regression functions that transform one set of data coordinates (the “independent variables”) into other. “Of course the term “best fit” is really arbitrary”, but the least squares approach remains now the method of choice, if there exist no strong arguments for another choice of metrics. This method was applied to many problems, has several times transformed and rediscovered, and now it is known under several names: mostly as PCA or as *proper orthogonal decomposition*. But the main idea remains the same: we approximate the data set by a point (this is the mean point), than by a line (first principal component), than by a plain, etc.

What was invented in the data approximation during the century? First of all, the approximation by linear manifolds (lines, plains, ...) was supplemented by reach choice of the approximate objects. The important discovery is approximation of data set by a smaller finite set of “centroids”. In the least squares approach to the best fit this gives the famous K -means algorithm [2]. Usually, this method is discussed as a clustering algorithm, but its application field is much wider. It is useful for adaptive coding and data binning, and is a model reduction method, as well as the PCA: the PCA allows to substitute a high-dimensional vector by its projection on a best fitted low-dimensional linear manifold, the K -means approach gives an approximation of a big data set by K best fitted centroids.

Between the “most rigid” linear manifolds and “most soft” unstructured finite sets there is the whole universe of approximants. If we change the PCA linear manifolds to algebraic curves and manifolds, then a branch of the *algebraic statistics* appears. This field is still relatively new (less than ten years old) [3]. Algebraic curves and manifolds are much more flexible than the linear ones, but remain rigid in the following sense: it is impossible to change the algebraic manifold locally, only near a point. Differential manifolds give more freedom, but require specific efforts for regularization.

A step from absolute flexibility of finite sets gives the Self-Organizing Map (SOM) approach [4]. SOM can be formalized either as a manifold learner which represents the manifold as a discretized grid, or a K -means-like clustering algorithm which adds a topology to the cluster centroids. Although SOM has been slowly being replaced by theoretically better founded and better behaving algorithms, its simplicity and computational efficiency makes it one of the most popular data analysis techniques even today. An important improvement of SOM came with the introduction of the Generative Topographic Mapping (GTM) [6], establishing a probabilistic framework and a well-defined objective function. The generative probabilistic model has also become an analytical tool to formalize the faithfulness-conciseness trade-off.

Another big shift of the century is appearance of the whole framework of machine learning which significantly extends the initial Pearson’s “geometrical” approach. It is a common practice in general discussions on machine learning to use the dichotomy of supervised and unsupervised learning to

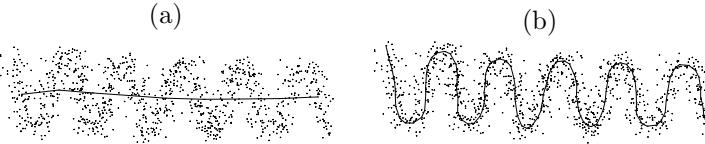


Fig. 2. An ill-defined unsupervised learning problem. Which curve describes the data better, (a) a short curve that is “far” from the data, or a (b) long curve that follows the data closely?

categorize learning methods. Supervised learning algorithms assume that a training set of *(input, output)* observations is given (e.g., digitized images of characters and their class labels). The goal is then to learn a function that predicts the output for previously unseen input patterns. This is a very far generalization of the Pearson’s linear regression onto various types of inputs, outputs, and functions.

In unsupervised learning, we only have a set of (input) observations without a desired target, and the goal is to find an efficient representation of the data (for example by reducing the number of attributes or grouping the data into a small number of clusters), or to characterize the data-generating process.

From a conceptual point of view, unsupervised learning is substantially more difficult than supervised learning. Whereas in supervised learning the cost of mis-predicting the output provides a well-defined criteria to optimize, in unsupervised learning we often face a trade-off of representing the data as faithfully as possible while being as concise as possible (Fig. 2). In a certain sense, an unsupervised learner can be considered as a supervised learner where the target is the input itself. In other words, the task is to find a function as close to the identity function as possible. Of course, without restricting the set of admissible predictors this is a trivial problem. These restrictions originate from the other objective of unsupervised learning of finding a mapping which is simple in a certain sense. The trade-off between these two competing objectives depends on the particular problem.

Manifold learning is a sub-domain of unsupervised learning where the goal is to project the input data into a new space which is simpler in a certain sense than the input space, or in which the data distribution is more regular than originally.

Two distinct groups of methods exist for this purpose that differ in their way of representing the manifold. Thus, Non-linear PCA (NLPCA) extends PCA by replacing the linear encoder and decoder by non-linear functions (for example, feed-forward neural networks [7]), and optimize them in an auto-encoding setup. The embedded manifold appears only implicitly as the decoded image of the input space, and the geometric notion of projection does not apply.

Principal curves and manifolds [8], on the other hand, extend the geometric interpretation of PCA by explicitly constructing an embedded manifold, and by encoding using standard geometric projection onto the manifold. How to define the “simplicity” of the manifold is problem-dependent, however, it is commonly measured by the intrinsic dimensionality and/or the smoothness of the manifold.

Clustering, another important sub-domain of unsupervised learning, can also be formalized in this framework: the clustering “manifold” is a finite partitioning of the input space, in the simplest case represented as a finite set of singular centroid points. Obviously, in this case simplicity cannot be measured by either smoothness or dimensionality, nevertheless, manifold learning and clustering methodologies are intimately connected both in their theoretical underpinning and on a technical-algorithmic level.

Most of the modern manifold learners find their theoretical and algorithmic roots in one of three basic and well-known data analysis techniques: PCA, K -means, and Multidimensional Scaling (MDS) [5] also known as Torgerson or Torgerson-Gower scaling. Thus, the basic loop of K -means that alternates between a projection and an optimization step became the algorithmic skeleton of many non-linear manifold learning algorithms. The SOM algorithm is arguably the torch holder of this batch of nonlinear manifold learners.

The objective of original MDS is somewhat different: find a linear projection that preserves pairwise distances as well as possible. The method does not explicitly construct an embedded manifold, but it has the important role of being the algorithmic forefather of “one-shot” (non-iterative) manifold learners. The most recent representatives of this approach are Local Linear Embedding (LLE) [9] and ISOMAP [10]. Both methods find their origins in MDS in the sense that their goal is to preserve pairwise relationships between data points. LLE conserves local linear patterns whereas ISOMAP applies MDS using the geodesic (manifold) distance approximated by the shortest path on the neighborhood graph (the graph constructed by connecting nearby points). Since the birth of these two methods, several neighborhood-graph-based techniques have emerged, stimulating the development of a common theory around Laplacian eigenmaps and spectral clustering and embedding.

Despite the significant progress made in the last decade, the manifold learning problem is far from being solved. The main drawback of iterative methods is that they are sensitive to initialization, and they can be stuck easily in suboptimal local minima, especially if the manifold is “loopy” or has a complicated topology. Neighborhood-graph-based “one-shot” techniques behave much better in this aspect, their disadvantage is computational inefficiency (the complexity of the construction of the neighborhood graph by itself is quadratic in the number of data points) and increased sensitivity to noise around the manifold. One of today’s challenges in manifold learning is to find techniques that combine the advantages of these often incompatible approaches. Another exciting area is non-local manifold learning [11], which abandons two of the implicit premises of manifold learning: that manifolds are

smooth (locally linear) and that we have enough observations in every neighborhood to locally estimate the manifold. A third, very recent but promising, new domain is building deep networks (multiply nested functions) using an unsupervised paradigm (building all the layers except for the last using, for example, an autoassociative objective function [12]). These new areas share the ambitious goal of embedding manifold learning into artificial intelligence in a broad sense.

This book is a collection of reviews and original papers presented partially at the workshop “Principal manifolds for data cartography and dimension reduction” (Leicester, August 24-26, 2006). The problems of Large Data Sets analysis and visualisation, model reduction and the struggle with complexity of data sets are important for many areas of human activity. There exist many scientific and engineering communities that attack these problems from their own sides, and now special efforts are needed to organize communication between these groups, to support exchange of ideas and technology transfer among them. Heuristic algorithms and seminal ideas come from all application fields and from mathematics also, and mathematics has a special responsibility to find a solid basis for heuristics, to transform ideas into exact knowledge, and to transfer the resulting ideal technology to all the participants of the struggle with complexity. The workshop was focused on modern theory and methodology of geometric data analysis and model reduction. Mathematicians, engineers, software developers and advanced users from different areas of applications attended this workshop.

The first chapter of the book presents a general review of existing NLPCA algorithms (U. Kruger, J. Zhang, and L. Xie). Next, M. Scholz, M. Fraunholz, and J. Selbig focus attention on autoassociative neural network approach for NLPCA with applications to metabolite data analysis and gene expression analysis. H. Yin provides an overview on the SOM in the context of manifold learning. Its variant, the visualisation induced SOM (ViSOM) proposed for preserving local metric on the map, is introduced and reviewed for data visualisation. The relationships among the SOM, ViSOM, multidimensional scaling, and principal curves are analysed and discussed. A. Gorban and A. Zinovyev developed a general geometric framework for constructing “principal objects” of various dimensions and topologies with the simple quadratic form of the smoothness penalty. The approach was proposed in the middle of 1990s. It is based on mechanical analogy between principal manifolds and elastic membranes and plates.

M. Peña, W. Barbakh, and C. Fyfe present a family of topology preserving mappings similar to SOM and GTM. These techniques can be considered as a non-linear projection from input or data space to the output or latent space. B. Mirkin develops the iterative extraction approach to clustering and describes additive models for clustering entity-to-feature and similarity. This approach emerged within the PCA framework by extending the bilinear Singular Value Decomposition model to that of clustering.

In their contribution, J. Einbeck, L. Evers, and C. Bailer-Jones give a short review of localized versions of PCA, focusing on local principal curves and local partitioning algorithms. These methods can work with branched and disconnected principal components. S. Girard and S. Iovleff introduce auto-associative models, a new tool for building NLPPCA methods, and compare it to other modern methods. A. Gorban, N. Sumner, and A. Zinovyev propose new type of low-dimensional “principal object”: *principal cubic complex*, the product of one-dimensional branching principal components. This complex is a generalization of linear and non-linear principal manifolds and includes them as a particular case. To construct such an object, they combine method of *topological grammars* with minimization of elastic energy defined for its embedment into multidimensional data space.

B. Nadler, S. Lafon, R. Coifman, and I. G. Kevrekidis provide a diffusion based probabilistic analysis of embedding and clustering algorithms that use the normalized graph Laplacian. They define a random walk on the graph of points and a diffusion distance between any two points. The characteristic relaxation times and processes of the random walk on the graph govern the properties of spectral clustering and spectral embedding algorithms. Specifically, for spectral clustering to succeed, a necessary condition is that the mean exit times from each cluster need to be significantly larger than the largest (slowest) of all relaxation times inside all of the individual clusters. Diffusion metrics is studied also by S. Damelin in context of optimal discretization problem. He shows that a general notion of extremal energy defines a diffusion metric on X which is equivalent to a discrepancy on X . The diffusion metric is used to learn X via normalized graph Laplacian dimension reduction and the discrepancy is used to discretize X .

Modern biological applications inspire development of new approaches to data approximation. In many chapters biological applications play central role. For the comparison of various algorithms, several test datasets were selected and presented to the attention of workshop participants. These datasets contain results of a high-throughput experimental technology application in molecular biology (microarray data). Principal component analysis and principal manifolds are highly demanded methods for analysis of this kind of data, where the “dimension curse” is an important issue. Because of it some variant of dimension reduction is absolutely required, for example, for regularization of classification problems that simply can not be solved otherwise. An interesting and underexplored question is: can non-linear principal manifolds serve better for this purpose as compared to the linear PCA or feature preselection?

M. Journée, A. E. Teschendorff, P.-A. Absil, S. Tavaré, and R. Sepulchre present an overview of the most popular algorithms to perform ICA. These algorithms are then applied on a microarray breast-cancer data set. D. Elizondo, B. N. Passow, R. Birkenhead, and A. Huemer present a comparison study of the performance of the linear principal component analysis and the non linear local tangent space alignment principal manifold methods to the problem of dimensionality reduction of microarray data.

The volume ends with a tutorial “PCA and K -Means decipher genome”. This exercise on principal component analysis and K -Means clustering can be used for courses of statistical methods and bioinformatics. By means of PCA students “discover” that the information in genome is encoded by non-overlapping triplets. Next, they learn to find gene positions. In Appendix the program listings for MatLab are presented.

The methods of data approximation, data visualization and model reduction developed during last century, form the important part of the modern intellectual technology of data analysis and modeling. In this book we present some slices of this interdisciplinary technology and aim at eliminating some of the traditional language barriers that, unnecessarily sometimes, impede scientific cooperation and interaction of researchers across disciplines.

References

1. Pearson, K.: On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, Ser. VI 2, 559–572 (1901)
2. MacQueen, J. B.: Some methods for classification and analysis of multivariate observations. In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. University of California Press, Berkeley, 281–297 (1967)
3. Pachter, L., Sturmfels, B. (eds): *Algebraic Statistics for Computational Biology*, Cambridge University Press, Cambridge, United Kingdom (2005)
4. Kohonen, T.: *The Self-Organizing Map*. Springer, Berlin Heidelberg New York (1997)
5. Torgerson, W. S.: *Theory and Methods of Scaling*. Wiley, New York (1958)
6. Bishop, C. M., Svensén, M., and Williams, C. K. I.: The generative topographic mapping. *Neural Computation*, **10**, 215–235 (1998)
7. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, **37**, 233–243 (1991)
8. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association*, **84**, 502–516 (1989)
9. Roweis, S. and Saul L. K.: Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**, 2323–2326 (2000)
10. Tenenbaum, J. B., de Silva, V., and Langford J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**, 2319–2323 (2000)
11. Bengio, Y., Monperrus, M., and Larochelle, H.: Nonlocal estimation of manifold structure. *Neural Computation*, **18** 2509–2528 (2006)
12. Hinton, G. E. and Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. *Science*, **313**, 504–507 (2006)

Leicester, UK
 Orsay, France
 Rolla, MO, USA
 Paris, France

May, 2007

*Alexander N. Gorban
 Balázs Kégl
 Donald C. Wunsch
 Andrei Y. Zinov'yev*

Contents

1 Developments and Applications of Nonlinear Principal Component Analysis – a Review

<i>Uwe Kruger, Junping Zhang, Lei Xie</i>	1
1.1 Introduction	1
1.2 PCA Preliminaries	3
1.3 Nonlinearity Test for PCA Models	6
1.3.1 Assumptions	7
1.3.2 Disjunct Regions	7
1.3.3 Confidence Limits for Correlation Matrix	8
1.3.4 Accuracy Bounds	10
1.3.5 Summary of the Nonlinearity Test	11
1.3.6 Example Studies	12
1.4 Nonlinear PCA Extensions	16
1.4.1 Principal Curves and Manifolds	16
1.4.2 Neural Network Approaches	24
1.4.3 Kernel PCA	29
1.5 Analysis of Existing Work	31
1.5.1 Computational Issues	31
1.5.2 Generalization of Linear PCA?	33
1.5.3 Roadmap for Future Developments (Basics and Beyond) ..	37
1.6 Concluding Summary	38
References	39

2 Nonlinear Principal Component Analysis: Neural Network Models and Applications

<i>Matthias Scholz, Martin Fraunholz, Joachim Selbig</i>	45
2.1 Introduction	45
2.2 Standard Nonlinear PCA	48
2.3 Hierarchical nonlinear PCA	49
2.3.1 The Hierarchical Error Function	51
2.4 Circular PCA	52

2.5	Inverse Model of Nonlinear PCA	53
2.5.1	The Inverse Network Model	55
2.5.2	NLPCA Models Applied to Circular Data	56
2.5.3	Inverse NLPCA for Missing Data	57
2.5.4	Missing Data Estimation	58
2.6	Applications	59
2.6.1	Application of Hierarchical NLPCA.....	60
2.6.2	Metabolite Data Analysis	61
2.6.3	Gene Expression Analysis	63
2.7	Summary.....	65
	References	66

3 Learning Nonlinear Principal Manifolds by Self-Organising Maps

<i>Hujun Yin</i>	69	
3.1	Introduction	69
3.2	Biological Background	70
3.2.1	Lateral Inhibition and Hebbian Learning	70
3.2.2	From Von Marsburg and Willshaw's Model to Kohonen's SOM	73
3.2.3	The SOM Algorithm.....	76
3.3	Theories	77
3.3.1	Convergence and Cost Functions	77
3.3.2	Topological ordering measures	80
3.4	SOMs, Multidimensional Scaling and Principal Manifolds	81
3.4.1	Multidimensional Scaling.....	81
3.4.2	Principal manifolds	83
3.4.3	Visualisation induced SOM (ViSOM)	85
3.5	Examples.....	87
3.5.1	Data visualisation	88
3.5.2	Document organisation and content management	88
	References	92

4 Elastic Maps and Nets for Approximating Principal Manifolds and Their Application to Microarray Data Visualization

<i>Alexander N. Gorban, Andrei Y. Zinovyev</i>	97	
4.1	Introduction and Overview	97
4.1.1	Fréchet Mean and Principal Objects: K-Means, PCA, what else?	97
4.1.2	Principal Manifolds.....	99
4.1.3	Elastic Functional and Elastic Nets	101
4.2	Optimization of Elastic Nets for Data Approximation	104
4.2.1	Basic Optimization Algorithm	104

4.2.2	Missing Data Values	106
4.2.3	Adaptive Strategies	107
4.3	Elastic Maps	110
4.3.1	Piecewise Linear Manifolds and Data Projectors	110
4.3.2	Iterative Data Approximation	110
4.4	Principal Manifold as Elastic Membrane	111
4.5	Method Implementation	113
4.6	Examples	113
4.6.1	Test Examples	113
4.6.2	Modeling Molecular Surfaces	114
4.6.3	Visualization of Microarray Data	115
4.7	Discussion	125
	References	128

5 Topology-Preserving Mappings for Data Visualisation

<i>Marian Peña, Wesam Barbakh, Colin Fyfe</i>	132	
5.1	Introduction	132
5.2	Clustering Techniques	133
5.2.1	K -Means	133
5.2.2	K-Harmonic Means	134
5.2.3	Neural Gas	136
5.2.4	Weighted K -Means	137
5.2.5	The Inverse Weighted K -Means	138
5.3	Topology Preserving Mappings	139
5.3.1	Generative Topographic Map	139
5.3.2	Topographic Product of Experts ToPoE	141
5.3.3	The Harmonic Topographic Map	142
5.3.4	Topographic Neural Gas	144
5.3.5	Inverse-Weighted K -Means Topology-Preserving Map	144
5.4	Experiments	144
5.4.1	Projections in Latent Space	145
5.4.2	Responsibilities	145
5.4.3	U-matrix, Hit Histograms and Distance Matrix	145
5.4.4	The Quality of The Map	148
5.5	Conclusions	150
	References	151

6 The Iterative Extraction Approach to Clustering

<i>Boris Mirkin</i>	153	
6.1	Introduction	153
6.2	Clustering Entity-to-feature Data	154
6.2.1	Principal Component Analysis	154
6.2.2	Additive Clustering Model and ITEX	156
6.2.3	Overlapping and Fuzzy Clustering Case	158
6.2.4	K -Means and iK-Means Clustering	159

6.3	ITEX Structuring and Clustering for Similarity Data	163
6.3.1	Similarity Clustering: a Review	163
6.3.2	The additive structuring model and ITEX	165
6.3.3	Additive clustering model	167
6.3.4	Approximate partitioning	168
6.3.5	One cluster clustering	170
6.3.6	Some applications	172
	References	176

7 Representing Complex Data Using Localized Principal Components with Application to Astronomical Data

<i>Jochen Einbeck, Ludger Evers, Coryn Bailer-Jones</i>	180	
7.1	Introduction	180
7.2	Localized Principal Component Analysis	183
7.2.1	Cluster-wise PCA	183
7.2.2	Principal Curves	187
7.2.3	Further Approaches	190
7.3	Combining Principal Curves and Regression	191
7.3.1	Principal Component Regression and its Shortcomings	191
7.3.2	The Generalization to Principal Curves	192
7.3.3	Using Directions Other than the Local Principal Components	194
7.3.4	A Simple Example	195
7.4	Application to the Gaia Survey Mission	197
7.4.1	The Astrophysical Data	197
7.4.2	Principal Manifold Based Approach	198
7.5	Conclusion	200
	References	201

8 Auto-Associative Models, Nonlinear Principal Component Analysis, Manifolds and Projection Pursuit

<i>Stéphane Girard, Serge Iovleff</i>	205	
8.1	Introduction	205
8.2	Auto-Associative Models	206
8.2.1	Approximation by Manifolds	206
8.2.2	A Projection Pursuit Algorithm	208
8.2.3	Theoretical Results	209
8.3	Examples	210
8.3.1	Linear Auto-Associative Models and PCA	210
8.3.2	Additive Auto-Associative Models and Neural Networks	211
8.4	Implementation Aspects	212
8.4.1	Estimation of the Regression Functions	212
8.4.2	Computation of Principal Directions	214
8.5	Illustration on Real and Simulated Data	216
	References	220

9 Beyond The Concept of Manifolds: Principal Trees, Metro Maps, and Elastic Cubic Complexes	
<i>Alexander N. Gorban, Neil R. Sumner, Andrei Y. Zinov'yev</i>	223
9.1 Introduction and Overview	223
9.1.1 Elastic Principal Graphs	225
9.2 Optimization of Elastic Graphs for Data Approximation	226
9.2.1 Elastic Functional Optimization	226
9.2.2 Optimal Application of Graph Grammars	227
9.2.3 Factorization and Transformation of Factors	228
9.3 Principal Trees (Branching Principal Curves)	229
9.3.1 Simple Graph Grammar ("Add a Node", "Bisect an Edge")	229
9.3.2 Visualization of Data Using "Metro Map" Two-Dimensional Tree Layout	229
9.3.3 Example of Principal Cubic Complex: Product of Principal Trees	231
9.4 Analysis of the Universal 7-Cluster Structure of Bacterial Genomes	233
9.4.1 Brief Introduction	234
9.4.2 Visualization of the 7-Cluster Structure	236
9.5 Visualization of Microarray Data	238
9.5.1 Dataset Used	238
9.5.2 Principal Tree of Human Tissues	238
9.6 Discussion	238
References	240
10 Diffusion Maps - a Probabilistic Interpretation for Spectral Embedding and Clustering Algorithms	
<i>Boaz Nadler, Stephane Lafon, Ronald Coifman, Ioannis G. Kevrekidis</i>	242
10.1 Introduction	242
10.2 Diffusion Distances and Diffusion Maps	244
10.2.1 Asymptotics of the Diffusion Map	249
10.3 Spectral Embedding of Low Dimensional Manifolds	250
10.4 Spectral Clustering of a Mixture of Gaussians	254
10.5 Summary and Discussion	262
References	262
11 On Bounds for Diffusion, Discrepancy and Fill Distance Metrics	
<i>Steven B. Damelin</i>	265
11.1 Introduction	265
11.2 Energy, Discrepancy, Distance and Integration on Measurable Sets in Euclidean Space	266
11.3 Set Learning via Normalized Laplacian Dimension Reduction and Diffusion Distance	270

11.4 Main Result: Bounds for Discrepancy, Diffusion and Fill Distance Metrics	271
References	272
12 Geometric Optimization Methods for the Analysis of Gene Expression Data	
<i>Michel Journée, Andrew E. Teschendorff, Pierre-Antoine Absil, Simon Tavaré, Rodolphe Sepulchre</i>	274
12.1 Introduction	274
12.2 ICA as a Geometric Optimization Problem	276
12.3 Contrast Functions	277
12.3.1 Mutual Information [8, 10]	278
12.3.2 \mathcal{F} -Correlation [14]	279
12.3.3 Non-Gaussianity [17]	280
12.3.4 Joint Diagonalization of Cumulant Matrices [19]	281
12.4 Matrix Manifolds for ICA	283
12.5 Optimization Algorithms	284
12.5.1 Line-Search Algorithms	284
12.5.2 FastICA	286
12.5.3 Jacobi Rotations	287
12.6 Analysis of Gene Expression Data by ICA	288
12.6.1 Some Issues About the Application of ICA	288
12.6.2 Evaluation of the Biological Relevance of the Expression Modes	290
12.6.3 Results Obtained on the Breast Cancer Microarray Data Set	292
12.7 Conclusion	294
References	294
13 Dimensionality Reduction and Microarray data	
<i>David A. Elizondo, Benjamin N. Passow, Ralph Birkenhead, Andreas Huemer</i>	297
13.1 Introduction	297
13.2 Background	299
13.2.1 Microarray data	299
13.2.2 Methods for Dimension Reduction	300
13.2.3 Linear Separability	301
13.3 Comparison Procedure	304
13.3.1 Data sets	304
13.3.2 Dimensionality Reduction	306
13.3.3 Perceptron Models	307
13.4 Results	307
13.5 Conclusions	310
References	311

14 PCA and K-Means Decipher Genome	
<i>Alexander N. Gorban, Andrei Y. Zinovyev</i>	313
14.1 Introduction	313
14.2 Required Materials	314
14.3 Genomic Sequence	315
14.3.1 Background	315
14.3.2 Sequences for the Analysis	315
14.4 Converting Text to a Numerical Table	316
14.5 Data Visualization	317
14.5.1 Visualization	317
14.5.2 Understanding Plots	317
14.6 Clustering and Visualizing Results	319
14.7 Task List and Further Information	320
14.8 Conclusion	322
References	323
Index	329

List of Contributors

Pierre-Antoine Absil

Department of Mathematical
Engineering,
Université catholique de Louvain,
Batiment Euler - Parking 13
Av. Georges Lemaître 4
1348 Louvain-la-Neuve
Belgium

Coryn Bailer-Jones

Max-Planck-Institut für Astronomie,
Königstuhl 17,
69117 Heidelberg,
Germany,
calj@mpia-hd.mpg.de

Wesam Barbakh

Applied Computational
Intelligence Research Unit,
The University of Paisley,
Paisley PA1 2BE
Scotland,
United Kingdom
wesam.barbakh@paisley.ac.uk

Ralph Birkenhead

Centre for Computational
Intelligence,
School of Computing,
Faculty of Computing Sciences
and Engineering,

De Montfort University,

The Gateway,
Leicester, LE1 9BH
United Kingdom
rab@dmu.ac.uk

Ronald Coifman

Department of Mathematics,
Yale University,
New Haven, CT, 06520-8283,
USA
coifman@math.yale.edu

Steven B. Damelin

Department of Mathematical
Sciences,
Georgia Southern University,
PO Box 8093,
Statesboro, GA 30460,
USA
damelin@georgiasouthern.edu

Jochen Einbeck

Department of Mathematical
Sciences,
Durham University,
Science Laboratories,
South Road,
Durham DH1 3LE,
United Kingdom
jochen.einbeck@durham.ac.uk

David A. Elizondo

Centre for Computational Intelligence,
 School of Computing,
 Faculty of Computing Sciences
 and Engineering,
 De Montfort University,
 The Gateway,
 Leicester, LE1 9BH
 United Kingdom
 elizondo@dmu.ac.uk

Ludger Evers

Department of Mathematics,
 University of Bristol,
 University Walk,
 Bristol BS8 1TW,
 United Kingdom
 l.evers@bris.ac.uk

Martin Fraunholz

Institute for Microbiology,
 Ernst-Moritz-Arndt-University
 Greifswald,
 F.-L.-Jahn-Str. 15,
 17487 Greifswald,
 Germany
 Martin.Fraunholz
 @uni-greifswald.de

Colin Fyfe

Applied Computational
 Intelligence Research Unit,
 The University of Paisley,
 Paisley PA1 2BE
 Scotland,
 United Kingdom
 colin.fyfe@paisley.ac.uk

Stéphane Girard

INRIA Rhône-Alpes, projet Mistis,
 Inovallée, 655,
 av. de l'Europe,
 Montbonnot, 38334 Saint-Ismier
 cedex,
 France
 Stephane.Girard@inrialpes.fr

Alexander N. Gorban

Department of Mathematics,
 University of Leicester,
 University Road,
 Leicester, LE1 7RH,
 United Kingdom
 ag153@le.ac.uk
 and Institute of Computational
 Modeling
 Russian Academy of Sciences,
 Siberian Branch
 Krasnoyarsk, Russia

Andreas Huemer

Centre for Computational
 Intelligence,
 School of Computing,
 Faculty of Computing Sciences
 and Engineering,
 De Montfort University,
 The Gateway,
 Leicester, LE1 9BH
 United Kingdom
 ahuemer@dmu.ac.uk

Serge Iovleff

Laboratoire Paul Painlevé,
 59655 Villeneuve d'Ascq Cedex,
 France
 serge.iovleff@univ-lille1.fr

Michel Journée

Department of Electrical Engineering
 and Computer Science,
 University of Liège,
 B-4000 Liège Sart-Tilman,
 Belgium

Ioannis G. Kevrekidis

Department of Chemical Engineering
 and Program in Applied
 and Computational Mathematics,
 Princeton University,
 Princeton, NJ 08544,
 USA
 yannis@princeton.edu

Uwe Kruger

School of Electronics,
Electrical Engineering and
Computer Science,
Queen's University Belfast,
Belfast, BT9 5AH,
United Kingdom
uwe.kruger@ee.qub.ac.uk

Stephane Lafon

Department of Mathematics,
Yale University,
New Haven, CT, 06520-8283,
USA
stephane.lafon@yale.edu
and Google, Inc.

Boris Mirkin

School of Computer Science
and Information Systems,
Birkbeck College
University of London,
Malet Street
London WC1E 7HX,
United Kingdom
mirkin@dcs.bbk.ac.uk

Boaz Nadler

Department of Computer Science
and Applied Mathematics,
Weizmann Institute of Science,
Rehovot, 76100,
Israel
boaz.nadler@weizmann.ac.il

Benjamin N. Passow

Centre for Computational
Intelligence,
School of Computing,
Faculty of Computing Sciences
and Engineering,
De Montfort University,
The Gateway,
Leicester, LE1 9BH
United Kingdom
passow@dmu.ac.uk

Marian Peña

Applied Computational
Intelligence Research Unit,
The University of Paisley,
Paisley PA1 2BE
Scotland,
United Kingdom
marian.pena@paisley.ac.uk

Matthias Scholz

Institute for Microbiology,
Ernst-Moritz-Arndt-University
Greifswald,
F.-L.-Jahn-Str. 15,
17487 Greifswald,
Germany,
Matthias.Scholz@uni-greifswald.de

Joachim Selbig

Institute for Biochemistry
and Biology,
University of Potsdam,
c/o Max Planck Institute
for Molecular Plant Physiology
Am Mühlenberg 1,
14424 Potsdam,
Germany
Selbig@mpimp-golm.mpg.de

Rodolphe Sepulchre

Department of Electrical Engineering
and Computer Science,
University of Liège,
B-4000 Liège Sart-Tilman,
Belgium
r.sepulchre@ulg.ac.be

Neil R. Sumner

Department of Mathematics,
University of Leicester,
University Road,
Leicester, LE1 7RH,
United Kingdom
nrs7@le.ac.uk

Simon Tavaré

Breast Cancer
Functional Genomics Program,
Cancer Research UK
Cambridge Research Institute,
Department of Oncology,
University of Cambridge,
Robinson Way,
Cambridge CB2 0RE,
United Kingdom

Andrew E. Teschendorff

Breast Cancer
Functional Genomics Program,
Cancer Research UK
Cambridge Research Institute,
Department of Oncology,
University of Cambridge,
Robinson Way,
Cambridge CB2 0RE,
United Kingdom

Lei Xie

National Key Laboratory
of Industrial Control Technology,
Zhejiang University,
Hangzhou 310027,
P.R. China
leix@iipc.zju.edu.cn

Hujun Yin

School of Electrical
and Electronic Engineering
The University of Manchester,
Manchester, M60 1QD,
United Kingdom
hujun.yin@manchester.ac.uk

Junping Zhang

Department of Computer Science
and Engineering,
Fudan University,
Shanghai 200433,
P.R. China
leix@iipc.zju.edu.cn

Andrei Y. Zinovyev

Institut Curie,
26, rue d'Ulm,
Paris, 75248,
France
andrei.zinovyev@curie.fr
and Institute of Computational
Modeling
Russian Academy of Sciences,
Siberian Branch
Krasnoyarsk, Russia

Developments and Applications of Nonlinear Principal Component Analysis – a Review

Uwe Kruger¹, Junping Zhang², and Lei Xie³

¹ School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT9 5AH, UK,
uwe.kruger@ee.qub.ac.uk

² Department of Computer Science and Engineering, Fudan University, Shanghai 200433, P.R. China,
jpzhang@fudan.edu.cn

³ National Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, P.R. China,
leix@iipc.zju.edu.cn

Summary. Although linear principal component analysis (PCA) originates from the work of Sylvester [67] and Pearson [51], the development of nonlinear counterparts has only received attention from the 1980s. Work on nonlinear PCA, or NLPCA, can be divided into the utilization of autoassociative neural networks, principal curves and manifolds, kernel approaches or the combination of these approaches. This article reviews existing algorithmic work, shows how a given data set can be examined to determine whether a conceptually more demanding NLPCA model is required and lists developments of NLPCA algorithms. Finally, the paper outlines problem areas and challenges that require future work to mature the NLPCA research field.

1.1 Introduction

PCA is a data analysis technique that relies on a simple transformation of recorded observation, stored in a vector $\mathbf{z} \in \mathbb{R}^N$, to produce statistically independent score variables, stored in $\mathbf{t} \in \mathbb{R}^n$, $n \leq N$:

$$\mathbf{t} = \mathbf{P}^T \mathbf{z} . \quad (1.1)$$

Here, \mathbf{P} is a transformation matrix, constructed from *orthonormal* column vectors. Since the first applications of PCA [21], this technique has found its way into a wide range of different application areas, for example signal processing [75], factor analysis [29, 44], system identification [77], chemometrics [20, 66] and more recently, general data mining [11, 70, 58] including image processing [17, 72] and pattern recognition [47, 10], as well as process

monitoring and quality control [1, 82] including multiway [48], multiblock [52] and multiscale [3] extensions. This success is mainly related to the ability of PCA to describe significant information/variation within the recorded data typically by the first few score variables, which simplifies data analysis tasks accordingly.

Sylvester [67] formulated the idea behind PCA, in his work the removal of redundancy in bilinear quantics, that are polynomial expressions where the sum of the exponents are of an order greater than 2, and Pearson [51] laid the conceptual basis for PCA by defining lines and planes in a multivariable space that present the closest fit to a given set of points. Hotelling [28] then refined this formulation to that used today. Numerically, PCA is closely related to an eigenvector-eigenvalue decomposition of a data covariance, or correlation matrix and numerical algorithms to obtain this decomposition include the iterative NIPALS algorithm [78], which was defined similarly by Fisher and MacKenzie earlier on [80], and the singular value decomposition. Good overviews concerning PCA are given in Mardia *et al.* [45], Jolliffe [32], Wold *et al.* [80] and Jackson [30].

The aim of this article is to review and examine nonlinear extensions of PCA that have been proposed over the past two decades. This is an important research field, as the application of linear PCA to nonlinear data may be inadequate [49]. The first attempts to present nonlinear PCA extensions include a generalization, utilizing a nonmetric scaling, that produces a nonlinear optimization problem [42] and constructing a curves through a given cloud of points, referred to as principal curves [25]. Inspired by the fact that the reconstruction of the original variables, $\hat{\mathbf{z}}$ is given by:

$$\hat{\mathbf{z}} = \mathbf{P}\mathbf{t} = \underbrace{\mathbf{P}}_{\text{mapping}} \underbrace{(\mathbf{P}^T \mathbf{z})}_{\text{demapping}}, \quad (1.2)$$

that includes the determination of the score variables (mapping stage) and the determination of $\hat{\mathbf{z}}$ (demapping stage), Kramer [37] proposed an *autoassociative neural network* (ANN) structure that defines the mapping and demapping stages by neural network layers. Tan and Mavrovouniotis [68] pointed out, however, that the 5 layers network topology of autoassociative neural networks may be difficult to train, i.e. network weights are difficult to determine if the number of layers increases [27].

To reduce the network complexity, Tan and Mavrovouniotis proposed an *input training* (IT) network topology, which omits the mapping layer. Thus, only a 3 layer network remains, where the reduced set of nonlinear principal components are obtained as part of the training procedure for establishing the IT network. Dong and McAvoy [16] introduced an alternative approach that divides the 5 layer autoassociative network topology into two 3 layer topologies, which, in turn, represent the nonlinear mapping and demapping functions. The output of the first network, that is the mapping layer, are

the score variables which are determined using the principal curve approach. The second layer then represents the demapping function for which the score variables are the inputs and the original variables are the outputs. Jia *et al.* [31] presented a critical review of the techniques in references [68, 16] and argued that the incorporation of a principal curve algorithm into a neural network structure [16] may only cover a limited class of nonlinear functions. Hence, the IT network topology [68] may provide a more effective nonlinear compression than the technique by Dong and McAvoy [16]. In addition, Jia *et al.* [31] further refined the IT concept by introducing a linear compression using PCA first, which is followed by the application of the IT algorithm using the scaled linear principal components.

More recently, *Kernel PCA* (KPCA) has been proposed by Schölkopf [57, 56]. KPCA first maps the original variable set \mathbf{z} onto a high-dimensional feature space using the mapping function $\Phi(\mathbf{z})$. Then, KPCA performs a conventional linear principal component analysis on $\Phi(\mathbf{z})$. The KPCA approach takes advantage of the fact that the mapping function $\mathbf{z} \mapsto \Phi(\mathbf{z})$ does not need to be known *a priori*. Furthermore, this mapping function can be approximated using Kernel functions in a similar fashion to a radial basis function neural network. In fact, the identification of a KPCA model utilizes scalar products of the observations, which are then nonlinearly transformed using Kernel functions. This presents a considerable advantage over neural network approaches since no nonlinear optimization procedure needs to be considered. Resulting from this conceptual simplicity and computational efficiency, KPCA has recently found its way into a wide range of applications, most notably in the areas of face recognition [36], image de-noising [40] and industrial process fault detection [12, 81].

This article is divided into the following sections. A brief review of PCA including its most important properties is given next, prior to the introduction of a nonlinearity test. Section 4 then details nonlinear extensions of PCA. Section 5 then critically evaluates existing work on NLP PCA in terms of computational demand in computing a model as well as generalization issues and provides a roadmap for future research work.

1.2 PCA Preliminaries

PCA analyses a data matrix $\mathbf{Z} \in \mathbb{R}^{K \times N}$ that possesses the following structure:

$$\mathbf{Z} = \begin{bmatrix} z_{11} & z_{21} & z_{13} & \cdots & z_{1j} & \cdots & z_{1N} \\ z_{21} & z_{22} & z_{23} & \cdots & z_{2j} & \cdots & z_{2N} \\ z_{31} & z_{32} & z_{33} & \cdots & z_{3j} & \cdots & z_{3N} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ z_{i1} & z_{i2} & z_{i3} & \cdots & z_{ij} & \cdots & z_{iN} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ z_{K-1,1} & z_{K-1,2} & z_{K-1,3} & \cdots & z_{K-1,j} & \cdots & z_{K-1,N} \\ z_{K1} & z_{K2} & z_{K3} & \cdots & z_{Kj} & \cdots & z_{KN} \end{bmatrix}, \quad (1.3)$$

where N and K are the number of recorded variables and the number of available observations, respectively. Defining the rows and columns of \mathbf{Z} by vectors $\mathbf{z}_i \in \mathbb{R}^N$ and $\zeta_j \in \mathbb{R}^K$, respectively, \mathbf{Z} can be rewritten as shown below:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \mathbf{z}_3^T \\ \vdots \\ \mathbf{z}_i^T \\ \vdots \\ \mathbf{z}_{K-1}^T \\ \mathbf{z}_K^T \end{bmatrix} = [\zeta_1 \zeta_2 \zeta_3 \cdots \zeta_j \cdots \zeta_N]. \quad (1.4)$$

The first and second order statistics of the original set variables $\mathbf{z}^T = (z_1 z_2 z_3 \cdots z_j \cdots z_N)$ are:

$$E\{\mathbf{z}\} = \mathbf{0} \quad E\{\mathbf{z}\mathbf{z}^T\} = \mathbf{S}_{ZZ} \quad (1.5)$$

with the correlation matrix of \mathbf{z} being defined as \mathbf{R}_{ZZ} .

The PCA analysis entails the determination of a set of score variables t_k , $k \in \{1 2 3 \cdots n\}$, $n \leq N$, by applying a linear transformation of \mathbf{z} :

$$t_k = \sum_{j=1}^N p_{kj} z_j \quad (1.6)$$

under the following constraint for the parameter vector

$$\mathbf{p}_k^T = (p_{k1} p_{k2} p_{k3} \cdots p_{kj} \cdots p_{kN}) : \sqrt{\sum_{j=1}^N p_{kj}^2} = \|\mathbf{p}_k\|_2 = 1. \quad (1.7)$$

Storing the score variables in a vector $\mathbf{t}^T = (t_1 t_2 t_3 \cdots t_j \cdots t_n)$, $\mathbf{t} \in \mathbb{R}^n$ has the following first and second order statistics:

$$E \{ \mathbf{t} \} = \mathbf{0} \quad E \{ \mathbf{t} \mathbf{t}^T \} = \mathbf{\Lambda}, \quad (1.8)$$

where $\mathbf{\Lambda}$ is a diagonal matrix. An important property of PCA is that the variance of the score variables represent the following maximum:

$$\lambda_k = \arg \max_{\mathbf{p}_k} \{ E \{ t_k^2 \} \} = \arg \max_{\mathbf{p}_k} \{ E \{ \mathbf{p}_k^T \mathbf{z} \mathbf{z}^T \mathbf{p}_k \} \}, \quad (1.9)$$

that is constraint by:

$$E \left\{ \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_{k-1} \end{pmatrix} t_k \right\} = \mathbf{0} \quad \| \mathbf{p}_k \|_2^2 - 1 = 0. \quad (1.10)$$

Anderson [2] indicated that the formulation of the above constrained optimization can alternatively be written as:

$$\lambda_k = \arg \max_{\mathbf{p}} \{ E \{ \mathbf{p}^T \mathbf{z} \mathbf{z}^T \mathbf{p} \} - \lambda_k (\mathbf{p}^T \mathbf{p} - 1) \} \quad (1.11)$$

under the assumption that λ_k is predetermined. Reformulating (1.11) to determine \mathbf{p}_k gives rise to:

$$\mathbf{p}_k = \arg \frac{\partial}{\partial \mathbf{p}} \{ E \{ \mathbf{p}^T \mathbf{z} \mathbf{z}^T \mathbf{p} \} - \lambda_k (\mathbf{p}^T \mathbf{p} - 1) \} = \mathbf{0} \quad (1.12)$$

and produces

$$\mathbf{p}_k = \arg \{ E \{ \mathbf{z} \mathbf{z}^T \} \mathbf{p} - 2\lambda_k \mathbf{p} \} = \mathbf{0}. \quad (1.13)$$

Incorporating (1.5) allows constructing an analytical solution of this constrained optimization problem:

$$[\mathbf{S}_{ZZ} - \lambda_k \mathbf{I}] \mathbf{p}_k = \mathbf{0}, \quad (1.14)$$

which implies that the k th largest eigenvalue of \mathbf{S}_{ZZ} is the variance of the k th score variable and the parameter vector \mathbf{p}_k , associated with λ_k , stores the k th set of coefficients to obtain the k th linear transformation of the original variable set \mathbf{z} to produce t_k . Furthermore, given that \mathbf{S}_{ZZ} is a positive definite or semidefinite matrix it follows that the eigenvalues are positive and real and the eigenvectors are mutually orthonormal. The solution of Equation (1.14) also implies that the score variables are statistically independent, as defined in (1.10), which follows from:

$$\widehat{\mathbf{S}}_{ZZ} = \frac{1}{K-1} \widehat{\mathbf{Z}}^T \widehat{\mathbf{Z}} = \widehat{\mathbf{P}} \widehat{\mathbf{\Lambda}} \widehat{\mathbf{P}}^T \implies \frac{1}{K-1} \widehat{\mathbf{P}}^T \mathbf{Z}^T \mathbf{Z} \widehat{\mathbf{P}} = \frac{1}{K-1} \widehat{\mathbf{T}}^T \widehat{\mathbf{T}} = \widehat{\mathbf{\Lambda}}. \quad (1.15)$$

Here, the index $\widehat{\cdot}$ represents estimates of the covariance matrix, its eigenvectors and eigenvalues and the score matrix using the reference data stored in \mathbf{Z} . A solution of Equations (1.9) and (1.10) can be obtained using a singular value decomposition of the data covariance matrix $\widehat{\mathbf{S}}_{ZZ}$ or the iterative Power method [22].

1.3 Nonlinearity Test for PCA Models

This section discusses how to determine whether the underlying structure within the recorded data is linear or nonlinear. Kruger *et al.* [38] introduced this nonlinearity test using the principle outlined in Figure 1.1. The left plot in

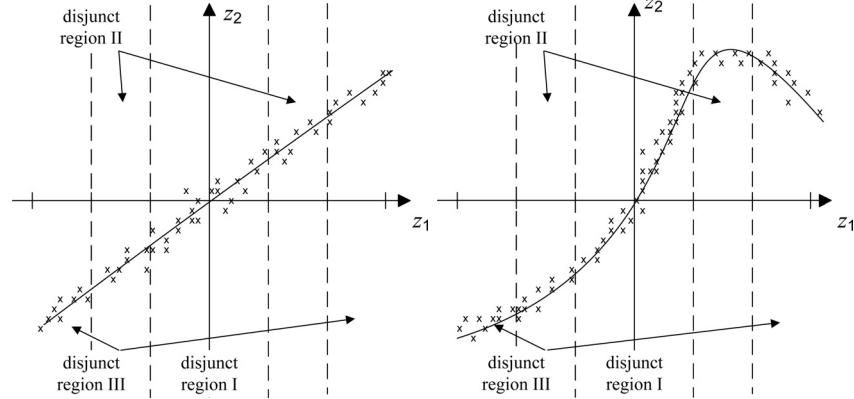


Fig. 1.1. Principle of nonlinearity test

this figure shows that the first principal component describes the underlying linear relationship between the two variables, z_1 and z_2 , while the right plot describes some basic nonlinear function, indicated by the curve.

By dividing the operating region into several disjunct regions, where the first region is centered around the origin of the coordinate system, a PCA model can be obtained from the data of each of these disjunct regions. With respect to Figure 1.1, this would produce a total of 3 PCA models for each disjunct region in both cases, the linear (left plot) and the nonlinear case (right plot). To determine whether a linear or nonlinear variable interrelationship can be extracted from the data, the principle idea is to take advantage of the residual variance in each of the regions. More precisely, *accuracy bounds* that are based on the residual variance are obtained for *one of the PCA models*, for example that of disjunct region I, and the residual variance of the *remaining PCA models* (for disjunct regions II and III) are benchmarked against *these bounds*. The test is completed if *each of the PCA models* has been used to determine accuracy bounds which are then benchmarked against the residual variance of the respective *remaining PCA models*.

The reason of using the residual variance instead of the variance of the retained score variables is as follows. The residual variance is independent of the region if the underlying interrelationship between the original variables is linear, which the left plot in Figure 1.1 indicates. In contrast, observations that have a larger distance from the origin of the coordinate system will, by default, produce a larger projection distance from the origin, that is a

larger score value. In this respect, observations that are associated with an adjunct region that are further outside will logically produce a larger variance irrespective of whether the variable interrelationships are linear or nonlinear.

The detailed presentation of the nonlinearity test in the remainder of this section is structured as follows. Next, the assumptions imposed on the nonlinearity test are shown, prior to a detailed discussion into the construction of disjunct regions. Subsection 3.3 then shows how to obtain statistical confidence limits for the nondiagonal elements of the correlation matrix. This is followed by the definition of the accuracy bounds. Finally, a summary of the nonlinearity test is presented and some example studies are presented to demonstrate the working of this test.

1.3.1 Assumptions

The assumptions imposed on the nonlinearity test are summarized below [38].

1. The variables are mean-centered and scaled to unit variance with respect to disjunct regions for which the accuracy bounds are to be determined.
2. Each disjunct region has the same number of observations.
3. A PCA model is determined for one region where the the accuracy bounds describe the variation for the sum of the discarded eigenvalues in that region.
4. PCA models are determined for the remaining disjunct regions.
5. The PCA models for each region include the same number of retained principal components.

1.3.2 Disjunct Regions

Here, we investigate how to construct the disjunct regions and how many disjunct regions should be considered. In essence, dividing the operating range into the disjunct regions can be carried out through prior knowledge of the process or by directly analyzing the recorded data. Utilizing *a priori* knowledge into the construction of the disjunct regions, for example, entails the incorporation of knowledge about distinct operating regions of the process. A direct analysis, on the other hand, by applying scatter plots of the first few retained principal components could reveal patterns that are indicative of distinct operating conditions. Wold *et al.* [80], page 46, presented an example of this based on a set of 20 “natural” amino acids.

If the above analysis does not yield any distinctive features, however, the original operating region could be divided into two disjunct regions initially. The nonlinearity test can then be applied to these two initial disjunct regions. Then, the number of regions can be increased incrementally, followed by a subsequent application of the test. It should be noted, however, that increasing the number of disjunct regions is accompanied by a reduction in the number of observations in each region. As outlined the next subsection, a sufficient

number of observations are required in order to prevent large Type I and II errors for testing the hypothesis of *using a linear model* against the alternative hypothesis of *rejecting that a linear model can be used*.

Next, we discuss which of the disjunct regions should be used to establish the accuracy bounds. Intuitively, one could consider the most centered region for this purpose or alternatively, a region that is at the margin of the original operating region. More practically, the region at which the process is known to operate most often could be selected. This, however, would require *a priori* knowledge of the process. However, a simpler approach relies on the incorporation of the cross-validation principle [65, 64] to automate this selection. In relation to PCA, cross-validation has been proposed as a technique to determine the number of retained principal components by Wold [79] and Krzanowski [39].

Applied to the nonlinearity test, the cross-validation principle could be applied in the following manner. First, select one disjunct region and compute the accuracy bounds of that region. Then, benchmark the residual variance of the remaining PCA models against this set of bounds. The test is completed if accuracy bounds have been computed for each of the disjunct regions and the residual variances of the PCA models of the respective remaining disjunct regions have been benchmarked against these accuracy bounds. For example, if 3 disjunct regions are established, the PCA model of the first region is used to calculate accuracy bounds and the residual variances of the 3 PCA models (one for each region) is benchmarked against this set of bounds. Then, the PCA model for the second region is used to determine accuracy bounds and again, the residual variances of the 3 PCA models are benchmarked against the second set of bounds. Finally, accuracy bounds for the PCA model of the 3rd region are constructed and each residual variance is compared to this 3rd set of bounds. It is important to note that the PCA models will vary depending upon which region is currently used to compute accuracy bounds. This is a result of the normalization procedure, since the mean and variance of each variable may change from region to region.

1.3.3 Confidence Limits for Correlation Matrix

The data correlation matrix, which is symmetric and positive semidefinite, for a given set of N variables has the following structure:

$$\mathbf{R}_{ZZ} = \begin{bmatrix} 1 & r_{12} & \cdots & r_{1N} \\ r_{21} & 1 & \cdots & r_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N1} & r_{N2} & \cdots & 1 \end{bmatrix}. \quad (1.16)$$

Given that the total number of disjunct regions is m the number of observations used to construct any correlation matrix is $\tilde{K} = K/m$, rounded to the nearest integer. Furthermore, the correlation matrix for constructing the

PCA model for the h th disjunct region, which is utilized to determine of the accuracy bound, is further defined by $\mathbf{R}_{ZZ}^{(h)}$. Whilst the diagonal elements of this matrix are equal to one, the nondiagonal elements represent correlation coefficients for which confidence limits can be determined as follows:

$$r_{ij}^{(h)} = \frac{\exp\left(2\zeta_{ij}^{(h)}\right) - 1}{\exp\left(2\zeta_{ij}^{(h)}\right) + 1} \text{ if } i \neq j, \quad (1.17)$$

where $\zeta_{ij}^{(h)} = \zeta_{ij}^{(h)*} \pm \varepsilon$, $\zeta_{ij}^{(h)*} = \ln\left(\frac{1 + r_{ij}^{(h)*}}{1 - r_{ij}^{(h)*}}\right)/2$, $r_{ij}^{(h)*}$ is the sample correlation coefficient between the i th and j th process variable, $\varepsilon = c_\alpha/\sqrt{\tilde{K} - 3}$ and c_α is the critical value of a normal distribution with zero mean, unit variance and a significance level α . This produces two confidence limits for each of the nondiagonal elements of $\mathbf{R}_{ZZ}^{(h)}$, which implies that the estimate nondiagonal elements with a significance level of α , is between

$$\mathbf{R}_{ZZ}^{(h)} = \begin{bmatrix} 1 & | & r_{12L}^{(h)} \leq r_{12}^{(h)} \leq r_{12U}^{(h)} & | & \cdots & | & r_{1N_L}^{(h)} \leq r_{1N}^{(h)} \leq r_{1N_U}^{(h)} \\ \hline r_{21L}^{(h)} \leq r_{21}^{(h)} \leq r_{21U}^{(h)} & | & 1 & | & \cdots & | & r_{2N_L}^{(h)} \leq r_{2N}^{(h)} \leq r_{2N_U}^{(h)} \\ \vdots & | & \vdots & | & \ddots & | & \vdots \\ \hline r_{N1L}^{(h)} \leq r_{N1}^{(h)} \leq r_{N1U}^{(h)} & | & r_{N2L}^{(h)} \leq r_{N2}^{(h)} \leq r_{N2U}^{(h)} & | & \cdots & | & 1 \end{bmatrix}, \quad (1.18)$$

where the indices U and L refer to the upper and lower confidence limit, that is $r_{ijL}^{(h)} = \frac{\exp(2(\zeta_{ij}^{(h)} - \varepsilon)) - 1}{\exp(2(\zeta_{ij}^{(h)} - \varepsilon)) + 1}$ and $r_{iju}^{(h)} = \frac{\exp(2(\zeta_{ij}^{(h)} + \varepsilon)) - 1}{\exp(2(\zeta_{ij}^{(h)} + \varepsilon)) + 1}$. A simplified version of Equation (1.18) is shown below

$$\mathbf{R}_{ZZL}^{(h)} \leq \mathbf{R}_{ZZ}^{(h)} \leq \mathbf{R}_{ZZU}^{(h)} \quad (1.19)$$

which is valid elementwise. Here, $\mathbf{R}_{ZZL}^{(h)}$ and $\mathbf{R}_{ZZU}^{(h)}$ are matrices storing the lower confidence limits and the upper confidence limits of the nondiagonal elements, respectively.

It should be noted that the confidence limits for each correlation coefficient is dependent upon the number of observations contained in each disjunct region, \tilde{K} . More precisely, if \tilde{K} reduces the confidence region widens according to (1.17). This, in turn, undermines the sensitivity of this test. It is therefore important to record a sufficiently large reference set from the analyzed process in order to (i) guarantee that the number of observations in each disjunct region does not produce excessively wide confidence regions for each correlation coefficient, (ii) produce enough disjunct regions for the test and (iii) extract information encapsulated in the recorded observations.

1.3.4 Accuracy Bounds

Finally, (1.19) can now be taken advantage of in constructing the accuracy bounds for the h th disjunct region. The variance of the residuals can be calculated based on the Frobenius norm of the residual matrix \mathbf{E}_h . Beginning with the PCA decomposition of the data matrix \mathbf{Z}_h , storing the observations of the h th disjunct region, into the product of the associated score and loading matrices, $\mathbf{T}_h \mathbf{P}_h^T$ and the residual matrix $\mathbf{E}_h = \mathbf{T}_h^* \mathbf{P}_h^{*T}$:

$$\mathbf{Z}_h = \mathbf{T}_h \mathbf{P}_h^T + \mathbf{E}_h = \mathbf{T}_h \mathbf{P}_h^T + \mathbf{T}_h^* \mathbf{P}_h^{*T}, \quad (1.20)$$

the sum of the residual variances for each original variable, ρ_{i_h} , $\rho_h = \sum_{i=1}^N \rho_{i_h}$ can be determined as follows:

$$\rho_h = \frac{1}{\tilde{K}-1} \sum_{i=1}^{\tilde{K}} \sum_{j=1}^N e_{ij_h}^2 = \frac{1}{\tilde{K}-1} \|\mathbf{E}_h\|_2^2. \quad (1.21)$$

which can be simplified to:

$$\rho_h = \frac{1}{\tilde{K}-1} \|\mathbf{T}_h^* \mathbf{P}_h^{*T}\|_2^2 = \frac{1}{\tilde{K}-1} \|\mathbf{U}_h^* \mathbf{\Lambda}_h^{*1/2} \sqrt{\tilde{K}-1} \mathbf{P}_h^{*T}\|_2^2 \quad (1.22)$$

and is equal to:

$$\rho_h = \frac{\tilde{K}-1}{\tilde{K}-1} \|\mathbf{\Lambda}_h^{*1/2}\|_2^2 = \sum_{i=n+1}^N \lambda_i. \quad (1.23)$$

Equations (1.20) and (1.22) utilize a singular value decomposition of \mathbf{Z}_h and reconstructs the discarded components, that is

$$\mathbf{E}_h = \mathbf{U}_h^* \left[\mathbf{\Lambda}_h^* \sqrt{\tilde{K}-1} \right] \mathbf{P}_h^{*T} = \mathbf{T}_h^* \mathbf{P}_h^{*T}.$$

Since $\mathbf{R}_{ZZ}^{(h)} = [\mathbf{P}_h \mathbf{P}_h^*] \begin{bmatrix} \mathbf{\Lambda}_h & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_h^* \end{bmatrix} \begin{bmatrix} \mathbf{P}_h^T \\ \mathbf{P}_h^{*T} \end{bmatrix}$, the discarded eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_N$ depend on the elements in the correlation matrix \mathbf{R}_{ZZ} . According to (1.18) and (1.19), however, these values are calculated within a confidence limits obtained for a significance level α . This, in turn, gives rise to the following optimization problem:

$$\begin{aligned} \rho_{h_{\max}} &= \arg \max_{\Delta \mathbf{R}_{ZZ_{\max}}} \rho_h (\mathbf{R}_{ZZ} + \Delta \mathbf{R}_{ZZ_{\max}}), \\ \rho_{h_{\min}} &= \arg \min_{\Delta \mathbf{R}_{ZZ_{\min}}} \rho_h (\mathbf{R}_{ZZ} + \Delta \mathbf{R}_{ZZ_{\min}}), \end{aligned} \quad (1.24)$$

which is subject to the following constraints:

$$\begin{aligned} \mathbf{R}_{ZZ_L} &\leq \mathbf{R}_{ZZ} + \Delta \mathbf{R}_{ZZ_{\max}} \leq \mathbf{R}_{ZZ_U}, \\ \mathbf{R}_{ZZ_L} &\leq \mathbf{R}_{ZZ} + \Delta \mathbf{R}_{ZZ_{\min}} \leq \mathbf{R}_{ZZ_U}, \end{aligned} \quad (1.25)$$

where $\Delta\mathbf{R}_{ZZ_{\max}}$ and $\Delta\mathbf{R}_{ZZ_{\min}}$ are perturbations of the nondiagonal elements in \mathbf{R}_{ZZ} that result in the determination of a maximum value, $\rho_{h_{\max}}$, and a minimum value, $\rho_{h_{\min}}$, of ρ_h , respectively.

The maximum and minimum value, $\rho_{h_{\max}}$ and $\rho_{h_{\min}}$, are defined as the accuracy bounds for the h th disjunct region. The interpretation of the accuracy bounds is as follows.

Definition 1. *Any set of observations taken from the same disjunct operating region cannot produce a larger or a smaller residual variance, determined with a significance of α , if the interrelationship between the original variables is linear.*

The solution of Equations (1.24) and (1.25) can be computed using a genetic algorithm [63] or the more recently proposed particle swarm optimization [50].

1.3.5 Summary of the Nonlinearity Test

After determining the accuracy bounds for the h th disjunct region, detailed in the previous subsection, a PCA model is obtained for each of the remaining $m-1$ regions. The sum of the $N-n$ discarded eigenvalues is then benchmarked against these limits to examine whether they fall inside or at least one residual variance value is outside. The test is completed if accuracy bounds have been computed for each of the disjunct regions including a benchmarking of the respective remaining $m-1$ residual variance. If for each of these combinations the residual variance is within the accuracy bound the process *is said to be linear*. In contrast, if at least one of the residual variances is outside one of the accuracy bounds, *it must be concluded that the variable interrelationships are nonlinear*. In the latter case, the uncertainty in the PCA model accuracy is smaller than the variation of the residual variances, implying that a nonlinear PCA model must be employed.

The application of the nonlinearity test involves the following steps.

1. Obtain a sufficiently large set of process data;
2. Determine whether this set can be divided into disjunct regions based on *a priori* knowledge; if yes, goto step 5 else goto step 3;
3. Carry out a PCA analysis of the recorded data, construct scatter diagrams for the first few principal components to determine whether distinctive operating regions can be identified; if so goto step 5 else goto step 4;
4. Divide the data into two disjunct regions, carry out steps 6 to 11 by setting $h = 1$, and investigate whether nonlinearity within the data can be proven; if not, increase the number of disjunct regions incrementally either until the sum of discarded eigenvalues violate the accuracy bounds or the number of observations in each region is insufficient to continue the analysis;
5. Set $h = 1$;
6. Calculate the confidence limits for the nondiagonal elements of the correlation matrix for the h th disjunct region (Equations (1.17) and (1.18));

7. Solve Equations (1.24) and (1.25) to compute accuracy bounds $\sigma_{h_{\max}}$ and $\sigma_{h_{\min}}$;
8. Obtain correlation/covariance matrices for each disjunct region (scaled with respect to the variance of the observations within the h th disjunct region);
9. Carry out a singular value decomposition to determine the sum of eigenvalues for each matrix;
10. Benchmark the sums of eigenvalues against the h th set of accuracy bounds to test the hypothesis that *the interrelationships between the recorded process variables are linear* against the alternative hypothesis that *the variable interrelationships are nonlinear*;
11. if $h = N$ terminate the nonlinearity test else goto step 6 by setting $h = h + 1$.

Examples of how to employ the nonlinearity test is given in the next subsection.

1.3.6 Example Studies

These examples have two variables, z_1 and z_2 . They describe (a) a linear interrelationship and (b) a nonlinear interrelationship between z_1 and z_2 . The examples involve the simulation of 1000 observations of a single score variable t that stem from a uniform distribution such that the division of this set into 4 disjunct regions produces 250 observations per region. The mean value of t is equal to zero and the observations of t spread between +4 and -4.

In the linear example, z_1 and z_2 are defined by superimposing two independently and identically distributed sequences, e_1 and e_2 , that follow a normal distribution of zero mean and a variance of 0.005 onto t :

$$z_1 = t + e_1, e_1 = \mathcal{N}\{0, 0.005\} \quad z_2 = t + e_2, e_2 = \mathcal{N}\{0, 0.005\}. \quad (1.26)$$

For the nonlinear example, z_1 and z_2 , are defined as follows:

$$z_1 = t + e_1 \quad z_2 = t^3 + e_2 \quad (1.27)$$

with e_1 and e_2 described above. Figure 1.2 shows the resultant scatter plots for the linear example (right plot) and the nonlinear example (left plot) including the division into 4 disjunct regions each.

Application of nonlinearity test to linear example

Table 1.1 summarizes the resultant values for the correlation coefficients, their confidence limits and the calculated upper and lower accuracy bound for each of the 4 disjunct regions. Table 1.2 shows the second eigenvalues of each of the correlation/covariance matrices for $h = 1, 2, 3$ and 4. Note that the values in italics correspond to the disjunct region for which the accuracy bounds have

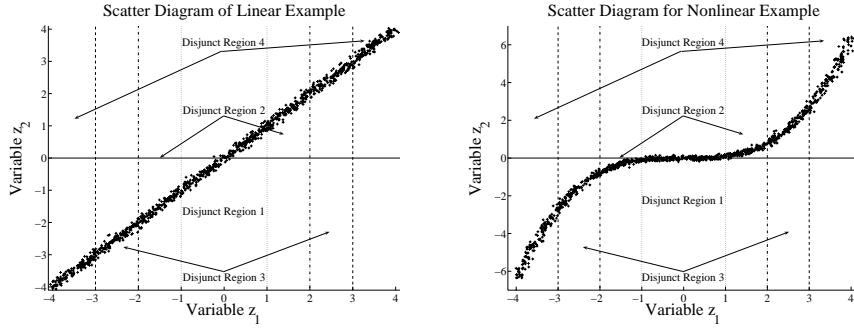


Fig. 1.2. Scatter diagrams for linear (left plot) and nonlinear simulation example (right plot) including boundaries for disjunct regions

Table 1.1. Correlation coefficients, their confidence limits and accuracy bounds for linear example.

h	$r_{21}^{(h)}$	$r_{12L}^{(h)}$	$r_{12U}^{(h)}$	$\sigma_{h_{\min}}$	$\sigma_{h_{\max}}$
1	0.9852	0.9826	0.9874	0.0126	0.0174
2	0.9978	0.9975	0.9982	0.0018	0.0025
3	0.9992	0.9991	0.9993	0.0007	0.0009
4	0.9996	0.9995	0.9997	0.0003	0.0005

Table 1.2. Residual variances (second eigenvalues) for each combination of disjunct regions.

$h/\text{DisjunctRegion}$	1	2	3	4
1	0.0148	0.0143	0.0147	0.0142
2	0.0022	0.0022	0.0022	0.0021
3	0.0008	0.0008	0.0008	0.0007
4	0.0004	0.0004	0.0004	0.0004

been calculated. Figure 1.3 benchmarks these residual variances against the accuracy bounds for each of the disjunct regions. This comparison yields that no violations of the accuracy bounds arise, which, as expected, leads to the acceptance of the hypothesis that the underlying relationship between z_1 and z_2 is linear. Next, we investigate whether the nonlinearity test can reveal that the second example describes a nonlinear relationship between both variables.

Application of nonlinearity test to nonlinear example

Using the data generated by (1.27), Table 1.3 summarizes the resultant values for the correlation coefficients, their confidence limits and the calculated upper and lower accuracy bound for each of the 4 disjunct regions. Again, there is

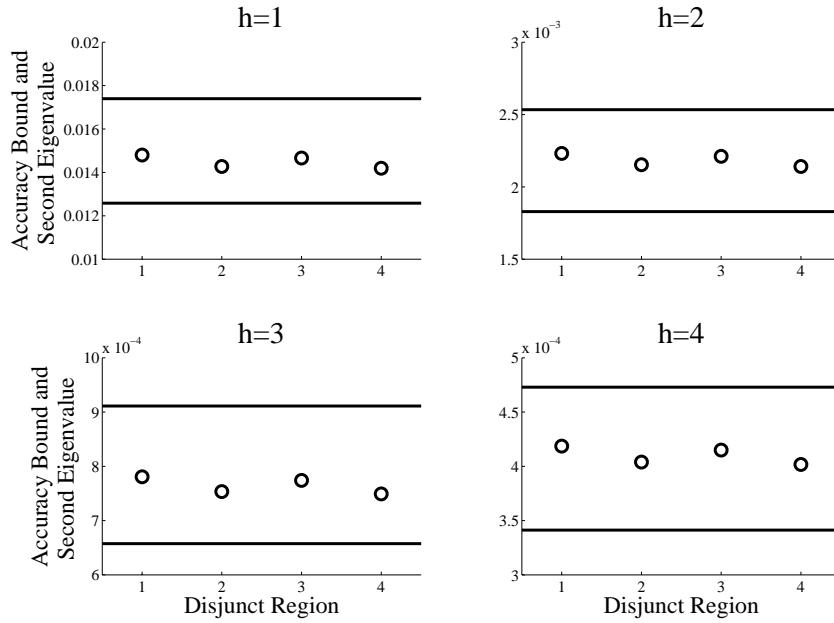


Fig. 1.3. Benchmarking of the residual variances against accuracy bounds of each disjunct region

Table 1.3. Correlation coefficients, their confidence limits and accuracy bounds for nonlinear example.

h	$r_{21}^{(h)}$	$r_{12_L}^{(h)}$	$r_{12_U}^{(h)}$	$\sigma_{h,\min}$	$\sigma_{h,\max}$
1	0.4341	0.3656	0.4979	0.5021	0.6344
2	0.9354	0.9244	0.9449	0.0551	0.0756
3	0.9752	0.9709	0.9789	0.0211	0.0291
4	0.9882	0.9861	0.9900	0.0100	0.0139

one principal component that describes the underlying relationship between z_1 and z_2 . In contrast, the other component represents the superimposed noise sequences e_1 and e_2 . However, this time, the underlying relationship is nonlinear.

Using the correlation/covariance matrices for each combination, Table 1.4 shows their second eigenvalue for $h = 1, 2, 3$ and 4 . As before, the diagonal elements are marked in italics and represent the residual variance of the reconstructed data inside the disjunct region for which accuracy bounds have been computed and, by default, must fall inside these bounds. In contrast to the linear example, the residual variance of the reconstructed data for each

Table 1.4. Residual variances (second eigenvalues) for each combination of disjunct regions.

$h/\text{DisjunctRegion}$	1	2	3	4
1	0.5659	0.7164	0.8866	0.8515
2	0.0258	0.0646	0.1145	0.1188
3	0.0017	0.0067	0.0248	0.0358
4	0.0002	0.0011	0.0056	0.0118

of the other disjunct regions, fall outside the accuracy bounds. Note that the violating elements in Table 1.4 are marked in bold. This result is expected given the construction of the nonlinear data set. Figure 1.4 gives a graphical illustration for the results in Table 1.4.

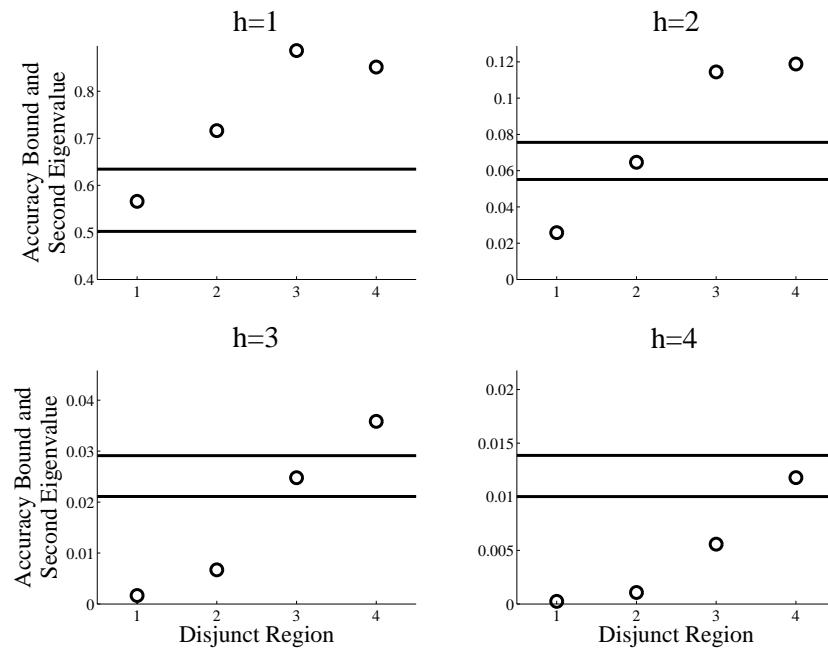


Fig. 1.4. regions

1.4 Nonlinear PCA Extensions

This section reviews nonlinear PCA extensions that have been proposed over the past two decades. Hastie and Stuetzle [25] proposed *bending* the loading vectors to produce curves that approximate the nonlinear relationship between a set of two variables. Such curves, defined as *principal curves*, are discussed in the next subsection, including their multidimensional extensions to produce principal surfaces or *principal manifolds*.

Another paradigm, which has been proposed by Kramer [37], is related to the construction of an artificial neural network to represent a nonlinear version of (1.2). Such networks that map the variable set \mathbf{z} to itself by defining a reduced dimensional bottleneck layer, describing nonlinear principal components, are defined as *autoassociative neural networks* and are revisited in Subsection 4.2.

A more recently proposed NLPCA technique relates to the definition of nonlinear mapping functions to define a feature space, where the variable space \mathbf{z} is assumed to be a nonlinear transformation of this feature space. By carefully selecting these transformation using Kernel functions, such as radial basis functions, polynomial or sigmoid kernels, conceptually and computationally efficient NLPCA algorithms can be constructed. This approach, referred to as *Kernel PCA*, is reviewed in Subsection 4.3.

1.4.1 Principal Curves and Manifolds

A brief introduction into principal curves (PCs) is given next, followed by a geometric interpretation illustrating the progression from the PCA weight vector, associated to the largest eigenvalue of \mathbf{S}_{ZZ} to a principal curve. The characteristics of principal curves are then outlined prior to algorithmic developments and refinements.

Introduction to principal curves

Principal Curves (PCs), presented by Hastie and Stuetzle [24, 25], are smooth one-dimensional curves passing through the *middle of a cloud representing a data set*. Utilizing probability distribution, a principal curve satisfies the self-consistent property, which implies that any point on the curve is the average of all data points projected onto it. As a nonlinear generalization of principal component analysis, PCs can be also regarded as a one-dimensional manifold embedded in high dimensional data space. In addition to the statistical property inherited from linear principal components, PCs also reflect the geometrical structure of data due. More precisely, the natural parameter *arc-length* is regarded as a projection index for each sample in a similar fashion to the score variable that represents the distance of the projected data point from the origin. In this respect, a one-dimensional nonlinear topological relationship between two variables can be estimated by a principal curve [85].

From a weight vector to a principal curve

Inherited from the basic paradigm of PCA, PCs assume that the intrinsic *middle structure* of data is a curve rather than a straight line. In relation to the total least squares concept [71], the cost function of PCA is to minimize the sum of projection distances from data points to a line. This produces the same solution as that presented in Section 2, Eq. (1.14). Geometrically, eigenvectors and their corresponding eigenvalues of \mathbf{S}_{ZZ} reflect the principal directions and the variance along the principal directions of data, respectively. Applying the above analysis to the first principal component, the following properties can be established [5]:

1. Maximize the variance of the projection location of data in the principal directions.
2. Minimize the squared distance of the data points from their projections onto the 1st principal component.
3. Each point of the first principal component is the conditional mean of all data points projected into it.

Assuming the underlying interrelationships between the recorded variables are governed by:

$$\mathbf{z} = \mathbf{At} + \mathbf{e}, \quad (1.28)$$

where $\mathbf{z} \in \mathbb{R}^N$, $\mathbf{t} \in \mathbb{R}^n$ is the latent variable (or projection index for the PCs), $\mathbf{A} \in \mathbb{R}^{N \times n}$ is a matrix describing the linear interrelationships between data \mathbf{z} and latent variables \mathbf{t} , and \mathbf{e} represent statistically independent noise, i.e. $E\{\mathbf{e}\} = \mathbf{0}$, $E\{\mathbf{ee}^T\} = \delta\mathbf{I}$, $E\{\mathbf{et}^T\} = \mathbf{0}$ with δ being the noise variance. PCA, in this context, uses the above principles of the first principal component to extract the n latent variables \mathbf{t} from a recorded data set \mathbf{Z} .

Following from this linear analysis, a general nonlinear form of (1.28) is as follows:

$$\mathbf{z} = \mathbf{f}(\mathbf{t}) + \mathbf{e}, \quad (1.29)$$

where $\mathbf{f}(\mathbf{t})$ is a nonlinear function and represents the interrelationships between the latent variables \mathbf{t} and the original data \mathbf{z} . Reducing $\mathbf{f}(\cdot)$ to be a linear function, Equation (1.29) clearly becomes (1.28), that is a special case of Equation (1.29).

To uncover the intrinsic latent variables, the following cost function, defined as

$$R = \sum_{i=1}^K \|\mathbf{z}_i - \mathbf{f}(\mathbf{t}_i)\|_2^2, \quad (1.30)$$

where K is the number available observations, can be used.

With respect to (1.30), linear PCA calculates a vector \mathbf{p}_1 for obtaining the largest projection index t_i of Equation (1.28), that is the diagonal elements of $E\{t^2\}$ represent a maximum. Given that \mathbf{p}_1 is of unit length, the location of

the projection of \mathbf{z}_i onto the first principal direction is given by $\mathbf{p}_1 t_i$. Incorporating a total of n principal directions and utilizing (1.28), Equation (1.30) can be rewritten as follows:

$$R = \sum_{i=1}^K \|\mathbf{z}_i - \mathbf{P}\mathbf{t}_i\|_2^2 = \text{trace} \left\{ \mathbf{Z}\mathbf{Z}^T - \mathbf{Z}^T \mathbf{A} [\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A} \mathbf{Z}^T \right\}, \quad (1.31)$$

where $\text{trace}\{\cdot\}$ is the sum of the diagonal elements of matrix. Minimizing Equation (1.31) is equivalent to the determination of the largest eigenvalue of $\mathbf{Z}\mathbf{Z}^T$. Similarly, the distance function for PCs is defined as:

$$D^2(\mathbf{f}(\mathbf{t})) = E \left\{ (\mathbf{z} - \mathbf{f}(\mathbf{t}))^T (\mathbf{z} - \mathbf{f}(\mathbf{t})) \right\}, \quad (1.32)$$

where the variational approach is a main technique to minimize $D^2(\mathbf{f})$. It can be proven that the solution of Equation (1.32) is equivalent to that of Equation (1.31) if $\mathbf{f}(\cdot)$ reduces to a linear function. In the nonlinear case, it can be shown that the critical value for constructing the PC is the distance function.

Mathematically, let \mathbf{f} be a PC and \mathbf{g} be any curve of a family of curves \mathbf{G} and define $\mathbf{f}_\epsilon = \mathbf{f}(\mathbf{t}) + \epsilon \mathbf{g}(\mathbf{t})$ where ϵ is a distortion factor, $0 \leq \epsilon \leq 1$. The distance from the data \mathbf{z} to \mathbf{f} is defined as follows:

$$D^2(h, \mathbf{f}_\epsilon) = E_h \|\mathbf{z} - \mathbf{f}_\epsilon(\mathbf{t}_{\mathbf{f}_\epsilon}(\mathbf{z}))\|_2^2, \quad (1.33)$$

where $\mathbf{t}_{\mathbf{f}_\epsilon}$ is the projection index of the data point closest to the projection location on the curve \mathbf{f}_ϵ and $E_h(\cdot)$ is mathematical expectation of the given data distribution density h .

It can be proven that \mathbf{f} is a critical value of the distance function in (1.33) under the assumption that Equation (1.34) is satisfied.

$$\frac{dD^2(h, \mathbf{f}_\epsilon)}{d\epsilon} \Big|_{\epsilon=0} \quad \forall \mathbf{g} \in \mathbf{G}. \quad (1.34)$$

Therefore, the objective function (distance function) of PCs is a nonlinear generalization of PCA. The major difference is that the shapes of PCs are uncertain, whereas those of PCA are lines. Hence, it is necessary to address the differences as it is discussed below.

Characteristics of principal curves

To adhere to the properties of principal curves, some basic definitions of principal curves are given first.

Definition 2. A one-dimensional curve embedded in \mathbb{R}^N is a continuous function $\mathbf{f} : \Xi \rightarrow \mathbb{R}^N$, where $\Xi = [a, b] \subset \mathbb{R}$.

The curve $\mathbf{f}(\cdot)$ is a function that is parameterized by a single parameter $t \in \Xi$, that is $\mathbf{f}^T(t) = (\mathbf{f}_1(t) \cdots \mathbf{f}_n(t))$, where $\mathbf{f}_1(t) \cdots \mathbf{f}_n(t)$ are referred to as coordinate functions.

Definition 3. For any $\mathbf{z} \in \mathbb{R}^N$, the corresponding projection index $t_f(\mathbf{z})$ on the curve $\mathbf{f}(t)$ is defined as

$$t_f(\mathbf{z}) = \sup \left\{ t : \|\mathbf{z} - \mathbf{f}(t)\|_2^2 = \inf_{\tau} \|\mathbf{z} - \mathbf{f}(\tau)\|_2^2 \right\}, \quad (1.35)$$

where $\mathbf{f}(t)$ is a curve in \mathbb{R}^N parameterized by $t \in \mathbb{R}$.

The above definition can be interpreted as follows. The projection index $t_f(\mathbf{z})$ of \mathbf{z} is the value of t for which $\mathbf{f}(t)$ is closest to \mathbf{z} . If there are multiple projection points that have an equal orthogonal distance to \mathbf{z} , the largest value of t is selected to remove ambiguity.

Hastie has proven that although ambiguous points may exist in the computation of PCs, the set of ambiguous points has a Lebesgue measure zero if the length of the curve is restricted. Hastie has further proven that for almost every \mathbf{z} , the projection function $t_{f_\epsilon}(\mathbf{z})$ is continuous under the compact support of probability density \mathbf{h} . The difference between ‘continuous’ and ‘ambiguous’ is: if $t_f(\mathbf{z})$ is continuous in \mathbf{z} , \mathbf{z} is not an ambiguous point. The basic idea of projection index is illustrated in Figure 1.5.

Definition 4. Based on the Definition 3, the distance between \mathbf{z} and the curve $\mathbf{f}(t)$ is computed to be the squared distance between \mathbf{z} and its projection point $\mathbf{f}(t_f(\mathbf{z}))$, that is:

$$\Delta(\mathbf{z}, \mathbf{f}) = \|\mathbf{z} - \mathbf{f}(t_f(\mathbf{z}))\|_2^2. \quad (1.36)$$

The projection distances from a data point to curve is an orthogonal distance rather than the vertical distances typically used by conventional regression methods.

Definition 5. Given a curve $\mathbf{f}(t), t \in \mathbb{R}$, the arc-length, l , between t_0 and t_1 is given by:

$$l = \int_{t_0}^{t_1} \|\mathbf{f}'(t)\|_2^2 dt = \sum_{i=1}^{K-1} \|\mathbf{f}(t_{i+1}) - \mathbf{f}(t_i)\|_2^2, \quad (1.37)$$

where $\mathbf{f}'(t)$ is tangent to the curve \mathbf{f} at projection index t and is often described as the velocity vector.

If $\|\mathbf{f}'(t)\|_2^2 \equiv 1$ then $l = t_1 - t_0$ and such a curve is defined as a unit speed parameterized curve.

Definition 6. The smooth curve $\mathbf{f}(t)$ is a principal curve if it:

1. does not intersect itself
2. has finite length inside any bounded subset of \mathbb{R}^n

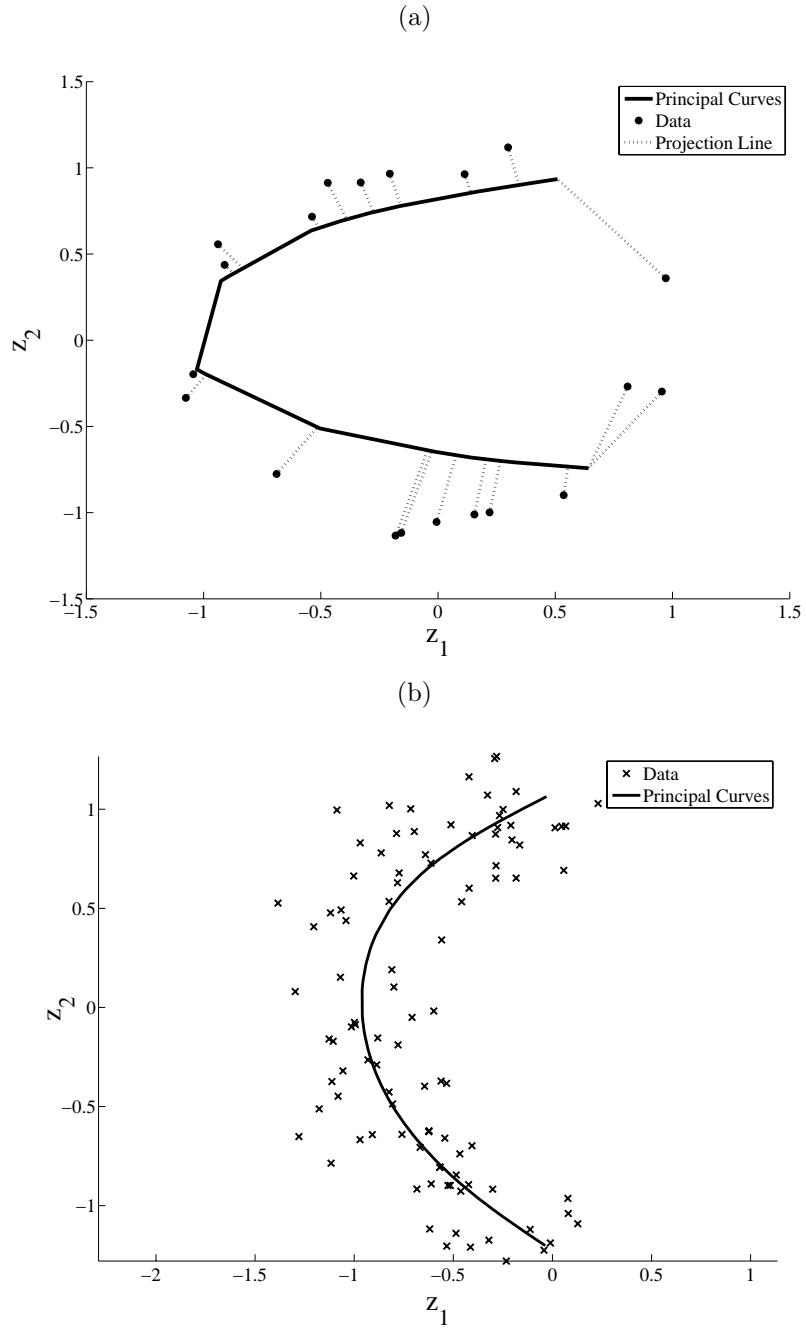


Fig. 1.5. Basic projection illustration from data point to curve (a) and a principal curve for a given data set (b)

3. is self-consistent, that is

$$\mathbf{f}(t) = E\{\mathbf{z} | t_f(\mathbf{z}) = t\} \quad \forall t \in \Xi. \quad (1.38)$$

The self-consistent property implies that each point on the curve is the conditional mean of the data points projected onto it. Thus, the principal curve is a smooth nonparametric self-consistent curve, which passes through the *middle of the distribution* and provides a one-dimensional nonlinear summary of the data.

Based on above definitions, Hastie and Stuetzle proposed a basic iterative algorithm for PCs, abbreviated as HSPCs for a given data distribution:

- Step1: Let the original curve $\mathbf{f}^{(0)}(t)$ be the first principal component where the subscript of \mathbf{f} denotes the actual iteration number, commencing with j being equal to set zero.
- Step2 (projection step): $\forall \mathbf{z} \in \mathbb{R}^N$, compute:

$$t_{f^{(j)}}(\mathbf{z}) = \sup \left\{ t : \left\| \mathbf{z} - \mathbf{f}^{(j)}(t) \right\|_2^2 = \inf_{\tau} \left\| \mathbf{z} - \mathbf{f}^{(j)}(\tau) \right\|_2^2 \right\}. \quad (1.39)$$

- Step3 (expectation): According to self-consistency, recompute the curve $\mathbf{f}^{(j+1)}(t) = E\{\mathbf{Z} | t_{f^{(j)}}(\mathbf{Z}) = t\}$.
- Step4 (judgement): If $1 - \frac{\Delta(\mathbf{f}^{(j+1)})}{\Delta(\mathbf{f}^{(j)})} < \epsilon$, then stop, else set $j = j + 1$ and goto Step 2.

For the above iteration, ϵ is a predefined convergence criterion, which can be set to 0.01 for example.

If the data distribution is unknown, cubic smoothing splines can be used as an alternative strategy for the estimation of HSPCs. This entails finding $\mathbf{f}(t)$ and $t_i \in [0, 1], i = \{1, \dots, K\}$ so that

$$D^2(\mathbf{f}) = \sum_{i=1}^K \| \mathbf{z}_i - \mathbf{f}(t_i) \|_2^2 + \mu \int_0^1 \| \mathbf{f}''(t) \|_2^2 dt \quad (1.40)$$

is minimized under the condition that the arc-length t is constrained to lie between $[0, 1]$, μ is fixed smoothing factor, and $\mathbf{f}''(t)$ denotes the second-order derivative. More details of splines may be available in [60] for example.

Algorithmic developments

Since the concept was proposed by Hastie and Stuetzle in 1989, a considerable number of refinements and further developments have been reported. The first thrust of such developments address the issue of bias. The HSPCs algorithm has two biases, a *model bias* and an *estimation bias*.

Assuming that the data are subjected to some distribution function with gaussian noise, a *model bias* implies that that the radius of curvature in the

curves is larger than the actual one. Conversely, spline functions applied by the algorithm results in an estimated radius that becomes smaller than the actual one.

With regards to the *model bias*, Tibshirani [69] assumed that data are generated in two stages (i) the points on the curve $\mathbf{f}(t)$ are generated from some distribution function μ_t and (ii) \mathbf{z} are formed based on conditional distribution $\mu_{z|t}$ (here the mean of $\mu_{z|t}$ is $\mathbf{f}(t)$). Assume that the distribution functions μ_t and $\mu_{z|t}$ are consistent with μ_z , that is $\mu_z = \int \mu_{z|t}(\mathbf{z}|t) \mu_t(t) dt$. Therefore, \mathbf{z} are random vectors of dimension N and subject to some density μ_z . While the algorithm by Tibshirani [69] overcomes the *model bias*, the reported experimental results in this paper demonstrate that the practical improvement is marginal. Moreover, the self-consistent property is no longer valid.

In 1992, Banfield and Raftery [4] addressed the *estimation bias* problem by replacing the squared distance error with residual and generalized the PCs into closed-shape curves. However, the refinement also introduces numerical instability and may form a smooth but otherwise incorrect principal curve.

In the mid 1990s, Duchamp and Stuezle [18, 19] studied the holistical differential geometrical property of HSPCs, and analyzed the first and second variation of principal curves and the relationship between self-consistent and curvature of curves. This work discussed the existence of principal curves in the sphere, ellipse and annulus based on the geometrical characters of HSPCs. The work by Duchamp and Stuezle further proved that under the condition that curvature is not equal to zero, the expected square distance from data to principal curve in the plane is just a saddle point but not a local minimum unless low-frequency variation is considered to be described by a constraining term. As a result, cross-validation techniques can not be viewed as an effective measure to be used for the model selection of principal curves.

At the end of the 1990s, Kégl proposed a new principal curve algorithm that incorporates a length constraint by combining vector quantization with principal curves. For this algorithm, further referred to as the KPC algorithm, Kégl proved that if and only if the data distribution has a finite second-order moment, a KPC exists and is unique. This has been studied in detail based on the principle of structural risk minimization, estimation error and approximation error. It is proven in references [34, 35] that the KPC algorithm has a faster convergence rate than the other algorithms described above. This supports to use of the KPC algorithm for large databases.

While the KPCs algorithm presents a significant improvement, several problems still remain. For example, the first principal component is often assumed to be an initial segment for the KPCs algorithm. For complex data which are subject to an uneven and/or sparse distribution, however, a good estimate of the initial curve plays a crucial role in order to guarantee that the algorithm converges to the actual principal curve. Secondly, the computational complexity gradually rises with an increase in the number of segments. However, if some vertices to be optimized go outside the domain of data, the algorithm has no ability to detect and remove this so that the subsequent

optimization and projection steps may fail. Thirdly, many parameters need to be predetermined and adjusted based on heuristic experience, which may hamper the practical usefulness of the KPC algorithm.

Addressing these drawbacks, Zhang and Chen [83] proposed a *constraint K-Segment principal curve* or CKPC algorithm. For this algorithm, the initial and final points of the curve are predefined by introducing data from the unobservable region or prior knowledge so that a better initial curves can be extracted. Secondly, a new constrained term for the removal of some abnormal vertices is presented to prevent subsequent optimization and projection steps to fail. Experiments involving intelligent transportation systems demonstrated that the CKPC algorithm provides a stronger generalization property than the KPC algorithm [83].

Morales [46] stated that from a differential manifold viewpoint a principal curves is a special case of manifold fitting. Morales work further generalized principal curves into principal embedding and introduced harmonic energy to be a regularizing term for determining a local minimum of the principal embedding. However, this work does not provide a practical algorithm for constructing a principal embedding. However, Smola [61] pointed out that most of the unsupervised learning approaches, such as principal curves, can rely on vector quantization, and proposed regularized principal manifold or RPM approach. Smola further proved the equivalence of the approach with the KPC algorithm, and derived a consistent convergence bound based on statistical learning theory.

Delicado [14] reconsidered the relation between principal curves and linear principal component analysis and introduced the concept of principal curves of oriented points or PCOP. This analysis was motivated by the fact that the first principal component goes through the conditional mean in a hyperplane and is orthogonal to the hyperplane which minimizes conditional total variance. When repeated searching from different samples, multiple points which satisfy the property of conditional mean value can be found. These points are called PCOP and the principal curve is the one across the PCOP. Similarly, the total-variance property can be recursively generalized to higher-order continuous principal curves. The drawback of this approach, however, is its considerable computational complexity.

Alternative approaches to principal curves include the work by Chang and Ghosh [7, 8], who define probabilistic principal curves, PPCs, and probabilistic principal surfaces, PPSs based on a generative topography mapping [6]. Different from the other approaches discussed above, PPCs and PPSs assume that high-dimensional topographical relationships among data are explicitly generated from some low-dimensional latent variables, and the algorithms are devoted to calculating the probability density of given data sets based on their latent variable structure. Advantages of these algorithms are that they not only keep the self-consistency, but also generalize principal curves into 2 dimensional to 3 dimensional principal manifolds through the introduction of parameterized models.

Verbeek [73] proposed a soft K-segments principal curves (SKPCs) algorithm, which first constructs K segments where each segment is obtained by local principal component analysis. These segments are then connected by minimizing the total degree of smoothness. A potential drawback of the SKPC algorithm is that it cannot be generalized into a high-dimension surface and the cost function does not consider a curvature penalty. Sandilya and Kulkarni [55] presented principal curves with bounded turn (PCBT). In a similar fashion to the KPC algorithm, PCBTs exists if and only if the data has a finite second-order moment.

Practically, principal curve algorithms have gained attention in a variety of applications, such as the alignment of magnets of the Stanford linear collier, identifying profiles of ice floes in the satellite images, handwriting ossification, speech recognition etc. [85, 53, 54, 26]. More recently, the principal curves algorithm by Hastie and Stuetzle [25] has also been applied to the modeling of freeway traffic streams [9] and the learning of high-precision GPS data from low-precision counterpart [84].

1.4.2 Neural Network Approaches

Using the structure shown in Figure 1.6, Kramer [37] proposed an alternative NLPCA implementation to principal curves and manifolds. This structure

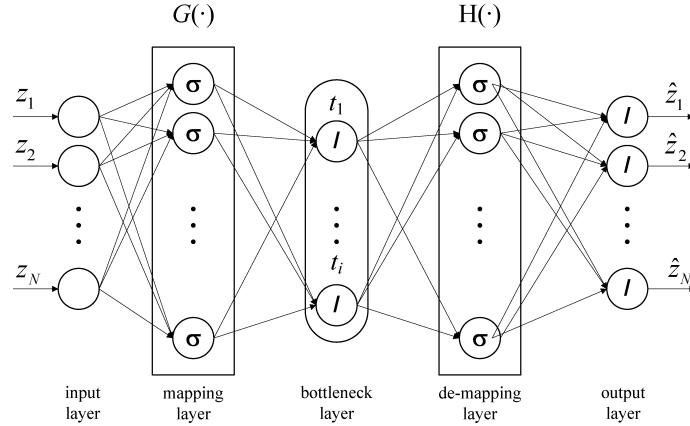


Fig. 1.6. Topology of autoassociative neural networks.

represents an autoassociative neural network (ANN), which, in essence, is an identify mapping that consists of a total of 5 layers. Identify mapping relates to this network topology is optimized to reconstruct the N network input variables as accurately as possible using a reduced set of bottleneck nodes $n < N$. From the left to right, the first layer of the ANN is the *input layer* that passes weighted values of the original variable set \mathbf{z} onto the second layer,

that is the *mapping layer*:

$$\xi_i = \sum_{j=1}^N w_{ij}^{(1)} z_j + b_i^1, \quad (1.41)$$

where $w_{ij}^{(1)}$ are the weights for the first layer and b_i^1 is a bias term. The sum in (1.41), ξ_i , is the input to the i th node in the mapping layer that consists of a total of M_m nodes. A scaled sum of the outputs of the nonlinearly transformed values $\sigma(\xi_i)$, then produce the nonlinear scores in the bottleneck layer. More precisely, the p th nonlinear score t_p , $1 \leq p \leq n$ is given by:

$$t_p = \sum_{i=1}^{M_m} w_{pi}^{(2)} \sigma(\xi_i) + b_p^{(2)} = \sum_{i=1}^{M_m} w_{pi}^{(2)} \sigma \left(\sum_{j=1}^N w_{ij}^{(1)} z_j + b_i^1 \right) + b_p^{(2)}. \quad (1.42)$$

To improve the modeling capability of the ANN structure for mildly nonlinear systems, it is useful to include linear contributions of the original variables $z_1 z_2 \cdots z_N$:

$$t_p = \sum_{i=1}^{M_m} w_{pi}^{(2)} \sigma \left(\sum_{j=1}^N w_{ij}^{(1)} z_j + b_i^1 \right) + \sum_{j=1}^N w_{pj}^{(1l)} z_j + b_p^{(2)}, \quad (1.43)$$

where the index l refers to the linear contribution of the original variables. Such a network, where a direct linear contribution of the original variables is included, is often referred to as a *generalized neural network*. The middle layer of the ANN topology is further referred to as the *bottleneck layer*.

A linear combination of these nonlinear score variables then produces the inputs for the nodes in the 4th layer, that is the *demapping layer*:

$$\tau_j = \sum_{p=1}^n w_{jp}^{(3)} t_p + b_p^{(3)}. \quad (1.44)$$

Here, $w_{jp}^{(3)}$ and $b_p^{(3)}$ are the weights and the bias term associated with the bottleneck layer, respectively, that represents the input for the j th node of the demapping layer. The nonlinear transformation of τ_j finally provides the reconstruction of the original variables \mathbf{z} , $\hat{\mathbf{z}} = (\hat{z}_1 \hat{z}_2 \cdots \hat{z}_N)^T$ by the *output layer*:

$$\hat{z}_q = \sum_{j=1}^{M_d} w_{qj}^{(4)} \sigma \left(\sum_{p=1}^n w_{jp}^{(3)} t_p + b_p^{(3)} \right) + \sum_{j=1}^n w_{qj}^{(3l)} t_j + b_q^{(4)}, \quad (1.45)$$

which may also include a linear contribution of the nonlinear score variables, indicated by the inclusion of the term $\sum_{j=1}^n w_{qj}^{(3l)} t_j$. Usually, the training of the

network weights and bias terms is done using a gradient descent approach like the computationally efficient Levenberg-Marquardt algorithm. It should be noted that the functional relationships between the original variable set $\mathbf{z} \in \mathbb{R}^N$ and the nonlinear score variables $\mathbf{t} \in \mathbb{R}^n$ is further referred to as the mapping function $\mathbf{G}(\cdot)$. Furthermore, the functional relationship between the nonlinear score variables and the reconstructed original variables $\hat{\mathbf{z}} \in \mathbb{R}^N$ is defined as the demapping function $\mathbf{H}(\cdot)$.

To symbolize a nonlinear version of the iterative Power method for computing linear PCA, Kramer [37] proposed the following network topology, shown in Figure 1.7. In a close resemblance to the total least squares prob-

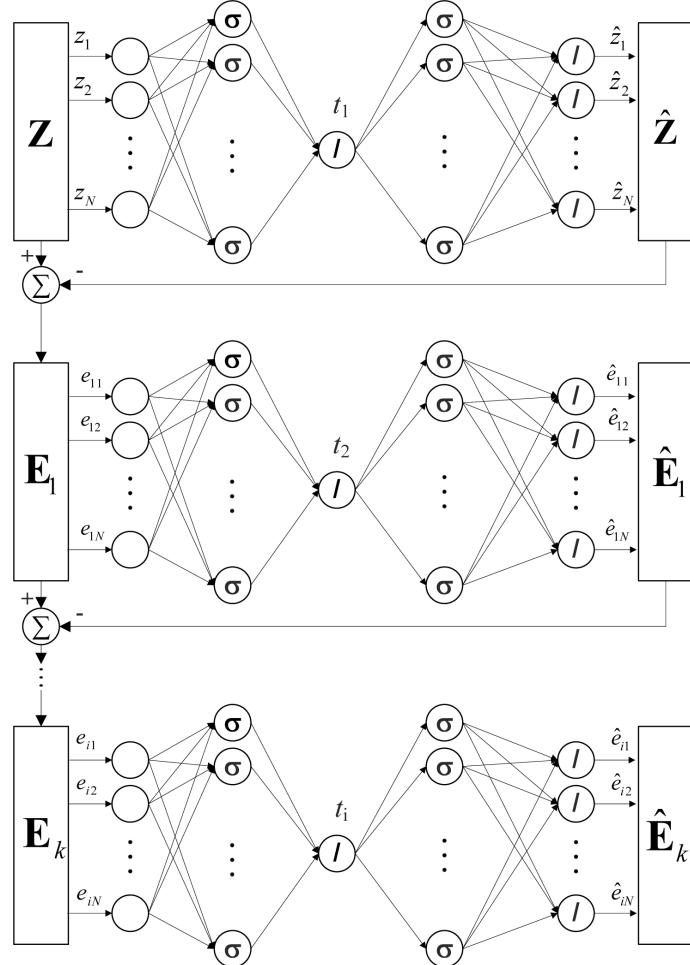


Fig. 1.7. Topology of a sequential autoassociative neural network.

lem [71] and the properties for first few principal components by Barnett [5], summarized in Subsection 4.1, this sequential ANN approach minimizes the squared distance of the data points from their reconstructions using the first nonlinear principal component t_1 . After deflating the data matrix $\mathbf{E}_1 = \mathbf{Z} - \hat{\mathbf{Z}}$, the second nonlinear principal component t_2 is again obtained such that the squared distance of the residuals, stored in \mathbf{E}_1 and the reconstruction of \mathbf{E}_1 , $\hat{\mathbf{E}}_2$ is minimized. This iteration continues until the residual variance of \mathbf{E}_k is sufficiently small.

Tan and Mavrovouniotis [68] pointed out that the ANN topology is complex and difficult to train. As a rule of thumb, an increase in the number of hidden layers produces a deterioration in the performance of backpropagation based algorithms, such as the Levenberg-Marquardt one [27]. The article by Tan and Mavrovouniotis also presents a simplified application study describing a circle which demonstrates the difficulties of training such a large network topology. To overcome this deficiency, they proposed an *input training* network that has the topology shown in Figure 1.8 for a total of 5 original variables, $z_1 z_2 z_3 z_4 z_5$ and two nonlinear principal components $t_1 t_2$. The

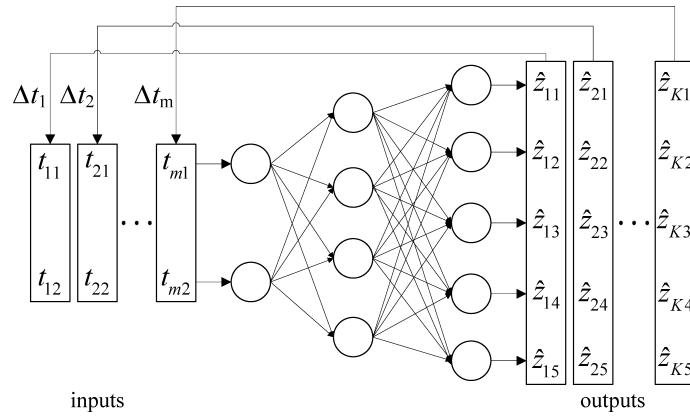


Fig. 1.8. Topology of the input training network.

input training or IT network determines the network weights as well as the score variables in an iterative fashion. For a minimum of the squared distance between the original and reconstructed observations, the following cost function can be used:

$$J = \sum_{i=1}^K \sum_{j=1}^N (z_{ij} - \hat{z}_{ij})^2 = \sum_{i=1}^K \sum_{j=1}^N (z_{ij} - \theta_j(\mathbf{t}_i))^2 , \quad (1.46)$$

where $\theta_j(\cdot)$ is the network function for reconstructing the j th original variable z_j and $\mathbf{t}_i^T = (t_1 \ t_2 \ \dots \ t_n)$. Under the assumption that the network weights are constant and predetermined, the *steepest descent* in the direction of optimal

network inputs is given by the gradient of J with respect to the nonlinear scores:

$$\Delta t_{ik} = -\frac{\partial J}{\partial t_{ik}} = \sum_{j=1}^N 2(z_{ij} - \theta_j(\mathbf{t}_i)) \frac{\partial \theta_j(\mathbf{t}_i)}{\partial t_{ik}}. \quad (1.47)$$

Reconstructing z_{ij} , that is determining \hat{z}_{ij} using the IT network, is based on the input and middle layers:

$$\hat{z}_{ij} = \theta_j(\mathbf{t}_i) = \sum_{p=1}^{M_d} w_{pj}^{(2)} \sigma \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right), \quad (1.48)$$

where the indices (1) and (2) refer to the input and middle layer weights and b_p is a bias term. For simplicity, linear terms are not included, which, however, does not restrict generality. Combining Equations (1.47) and (1.48) gives rise to determine the steepest descent in the direction of the network training inputs between the input and hidden layer:

$$\Delta t_{ik} = \sum_{p=1}^N w_{pj}^{(1)} \delta_{ip}, \quad (1.49)$$

where:

$$\delta_{ip} = \sum_{j=1}^N 2(z_{ij} - \theta_j(\mathbf{t}_i)) \frac{\partial \theta_j(\mathbf{t}_i)}{\partial t_{ik}} \quad (1.50)$$

which is given by:

$$\frac{\partial \theta_j(\mathbf{t}_i)}{\partial t_{ik}} = \sum_{p=1}^{M_d} w_{pj}^{(2)} \frac{\partial}{\partial t_{ik}} \sigma \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right), \quad (1.51)$$

and, hence,

$$\frac{\partial \theta_j(\mathbf{t}_i)}{\partial t_{ik}} = w_{kj}^{(2)} \sigma' \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_q \right), \quad (1.52)$$

so that δ_{ip} can finally be determined as

$$\delta_{ip} = \sigma' \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_q \right) \sum_{j=1}^N 2w_{kj}^{(2)} (z_{ij} - \theta_j(\mathbf{t}_i)). \quad (1.53)$$

Following the above derivative, the steepest descent direction for the training network weights between the input and hidden layer can be obtained as follows:

$$\Delta w_{pk}^{(1)} = -\frac{\partial J}{\partial w_{pk}^{(1)}} = \sum_{i=1}^K \sum_{j=1}^N \frac{\partial}{\partial w_{pk}^{(1)}} \left(z_{ij} - \sum_{p=1}^{M_d} w_{pj}^{(2)} \sigma \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right) \right)^2, \quad (1.54)$$

which is given by

$$\Delta w_{pk}^{(1)} = \sum_{i=1}^K \sum_{j=1}^N 2 \left(z_{ij} - \sum_{p=1}^{M_d} w_{pj}^{(2)} \sigma \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right) \right) t_{ik} \sigma' \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right), \quad (1.55)$$

which in a simplified form is given by

$$\Delta w_{pk}^{(1)} = \sum_{i=1}^K \eta_{ik}^{(1)} \sum_{j=1}^N \left(z_{ij} - \sum_{p=1}^{M_d} w_{pj}^{(2)} \sigma \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right) \right), \quad (1.56)$$

where $\eta_{ik}^{(1)} = t_{ik} \sigma' \left(\sum_{q=1}^n w_{pq}^{(1)} t_{iq} + b_p \right)$. In a similar fashion to (1.55), a steepest descent can also be derived for the network weights $w_{pk}^{(2)}$. Using the steepest descents, derived above, the IT network can be trained using backpropagation techniques.

Dong and McAvoy [16] proposed another approach to simplify the structure of the original complex 5 layer structure by Kramer. This work relies on a separation of the 5 layer network into the 3 layer mapping function $\mathbf{G}(\cdot)$ and another 3 layer network representing the demapping function $\mathbf{H}(\cdot)$. According to Figure 1.6, the input of the mapping function are the original variables $\mathbf{z} \in \mathbb{R}^N$ and the output are the nonlinear score variables $\mathbf{t} \in \mathbb{R}^n$, while the inputs and outputs of the demapping function are the score variables and reconstructed variables $\hat{\mathbf{z}} \in \mathbb{R}^N$, respectively. Instead of determining the inputs to the demapping function $\mathbf{H}(\cdot)$ by optimizing Equation (1.46) with respect to the nonlinear score variables for each observation, the approach by Dong and McAvoy utilizes principal curves to determine nonlinear score variables.

A potential problem of this approach has been discussed by Jia *et al.* [31]. The principal curve approach by Dong and McAvoy assumes that the approximation of a nonlinear function can be achieved by a linear combination of a number of univariate nonlinear functions. This, however, is a restriction of generality and implies that only a limited class of nonlinear functions can be approximated using this technique. In contrast, the algorithm by Tan and Mavrovouniotis does not suffer from this inherent limitation.

1.4.3 Kernel PCA

This subsection details the principles of kernel PCA, which has been proposed by Schölkopf *et al.* [57]. An introduction into the principles of kernel PCA, including the definition of the covariance matrix in the feature space, is given next, followed by an outline of how to compute a kernel PCA model using the kernel matrix. It is finally shown how to extract the score variables from the kernel PCA model.

Introduction to kernel PCA

This technique first maps the original input vectors \mathbf{z} onto a high-dimensional feature space $\mathbf{z} \mapsto \Phi(\mathbf{z})$ and then perform the principal component analysis on $\Phi(\mathbf{z})$. Given a set of observations $\mathbf{z}_i \in \mathbb{R}^N$, $i = \{1, 2, \dots, K\}$, the mapping of \mathbf{z}_i onto a feature space, that is $\Phi(\mathbf{z})$ whose dimension is considerably larger than N , produces the following sample covariance matrix:

$$\mathbf{S}_{\Phi\Phi} = \frac{1}{K-1} \sum_{i=1}^K (\Phi(\mathbf{z}_i) - \mathbf{m}_\Phi)(\Phi(\mathbf{z}_i) - \mathbf{m}_\Phi)^T = \frac{1}{K-1} \bar{\Phi}(\mathbf{Z})^T \bar{\Phi}(\mathbf{Z}) . \quad (1.57)$$

Here, $\mathbf{m}_\Phi = \frac{1}{K} \Phi(\mathbf{Z})^T \mathbf{1}_K$, where $\mathbf{1}_K \in \mathbb{R}^K$ is a column vector storing unity elements, is the sample mean in the feature space, and $\Phi(\mathbf{Z}) = [\Phi(\mathbf{z}_1) \Phi(\mathbf{z}_2) \dots \Phi(\mathbf{z}_K)]^T$ and $\bar{\Phi}(\mathbf{Z}) = \Phi(\mathbf{Z}) - \frac{1}{K} \mathbf{E}_K \Phi(\mathbf{Z})$, with \mathbf{E}_K being a matrix of ones, are the original and mean centered feature matrices, respectively.

KPCA now solves the following eigenvector-eigenvalue problem,

$$\mathbf{S}_{\Phi\Phi} \mathbf{p}_i = \frac{1}{K-1} \bar{\Phi}(\mathbf{Z})^T \bar{\Phi}(\mathbf{Z}) \mathbf{p}_i = \lambda_i \mathbf{p}_i \quad i = 1, 2, \dots, N , \quad (1.58)$$

where λ_i and \mathbf{p}_i are the eigenvalue and its associated eigenvector of $\mathbf{S}_{\Phi\Phi}$, respectively. Given that the explicit mapping formulation of $\Phi(\mathbf{z})$ is usually unknown, it is difficult to extract the eigenvector-eigenvalue decomposition of $\mathbf{S}_{\Phi\Phi}$ directly. However, KPCA overcomes this deficiency as shown below.

Determining a kernel PCA model

Starting from the eigenvector-eigenvalue decomposition of $\mathbf{G} = \bar{\Phi}(\mathbf{Z}) \bar{\Phi}(\mathbf{Z})^T$, which is further defined as the Gram matrix:

$$\bar{\Phi}(\mathbf{Z}) \bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i = \zeta_i \mathbf{v}_i , \quad (1.59)$$

where ζ_i and \mathbf{v}_i are the eigenvalue and its eigenvector, respectively, carrying out a pre-multiplication of (1.59) by $\bar{\Phi}(\mathbf{Z})^T$ produces:

$$\bar{\Phi}(\mathbf{Z})^T \bar{\Phi}(\mathbf{Z}) \bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i = \zeta_i \bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i \quad i = 1, 2, \dots, N . \quad (1.60)$$

By comparing (1.60) and (1.58), it now follows that $\zeta_i/(K-1)$ and $\bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i / \|\bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i\|_2$ are also corresponding eigenvalues and eigenvectors of $\mathbf{S}_{\Phi\Phi}$, that is:

$$\begin{aligned} \lambda_i &= \zeta_i/(K-1), \\ \mathbf{p}_i &= \bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i / \sqrt{\mathbf{v}_i^T \bar{\Phi}(\mathbf{Z}) \bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i} = \bar{\Phi}(\mathbf{Z})^T \mathbf{v}_i / \sqrt{\zeta_i} . \end{aligned} \quad (1.61)$$

By defining a kernel function $\psi(\mathbf{z}_i, \mathbf{z}_j) = \Phi(\mathbf{z}_i)^T \Phi(\mathbf{z}_j)$, the Gram matrix \mathbf{G} can be constructed from a *kernel matrix* $\Psi(\mathbf{Z}) \in \mathbb{R}^{K \times K}$ whose elements ψ_{ij} are $\psi(\mathbf{z}_i, \mathbf{z}_j)$,

$$\mathbf{G} = \Psi(\mathbf{Z}) - \frac{1}{K}\Psi(\mathbf{Z})\mathbf{E}_K - \frac{1}{K}\mathbf{E}_K\Psi(\mathbf{Z}) + \frac{1}{K^2}\mathbf{E}_K\Psi(\mathbf{Z})\mathbf{E}_K. \quad (1.62)$$

It is important to note that the calculation of \mathbf{G} (i) only requires the kernel formulation of $\psi(\mathbf{z}_i, \mathbf{z}_j)$ and (ii) but no *a priori* knowledge of the exact mapping $\Phi(\mathbf{z})$. The most commonly used kernel functions include polynomial, RBF and Sigmoid kernels [59].

Calculation of the score variables

Assuming that a PCA model has been constructed from the covariance matrix $\mathbf{S}_{\Phi\Phi}$, that is $\mathbf{S}_{\Phi\Phi} = \mathbf{P}\Lambda\mathbf{P}^T$, incorporating Equation (1.61) gives rise to:

$$\mathbf{P} = [\mathbf{p}_1 \mathbf{p}_2 \cdots \mathbf{p}_n] = \bar{\Phi}(\mathbf{Z})^T \left[\mathbf{v}_1/\sqrt{\zeta_1} \mathbf{v}_2/\sqrt{\zeta_2} \cdots \mathbf{v}_n/\sqrt{\zeta_n} \right]. \quad (1.63)$$

Redefining $\bar{\Phi}(\mathbf{Z})$, as shown in (1.57), and rewriting (1.63) produces:

$$\mathbf{P} = \left[\Phi(\mathbf{Z})^T - \frac{1}{K}\Phi(\mathbf{Z})^T\mathbf{E}_N \right] \mathbf{V} = \Phi(\mathbf{Z})^T \mathbf{A}, \quad (1.64)$$

where $\mathbf{V} = [\mathbf{v}_1/\sqrt{\zeta_1} \mathbf{v}_2/\sqrt{\zeta_2} \cdots \mathbf{v}_n/\sqrt{\zeta_n}]$, $\mathbf{A} = [\mathbf{I}_K - \frac{1}{K}\mathbf{E}_K]\mathbf{V}$ with \mathbf{I}_K being the identity matrix of dimension K . Utilizing Equation (1.64), the score variables, stored in the vector \mathbf{t} , can now be obtained as follows:

$$\mathbf{t} = \mathbf{P}^T [\Phi(\mathbf{z}) - \mathbf{m}_\Phi] = \mathbf{A}^T \Phi(\mathbf{Z}) \left[\Phi(\mathbf{z}) - \frac{1}{K}\Phi(\mathbf{Z}), \mathbf{1}_K \right] \quad (1.65)$$

which, using the definition of the kernel function $\psi(\cdot)$, can finally be written as shown below:

$$\mathbf{t} = \mathbf{A}^T \left(\psi(\mathbf{Z}, \mathbf{z}) - \frac{1}{K}\Psi(\mathbf{Z})\mathbf{1}_K \right). \quad (1.66)$$

Here, $\psi(\mathbf{Z}, \mathbf{z})$ is the *kernel vector* for the new observation \mathbf{z} based on the set of reference observations \mathbf{Z} , that is $\psi(\mathbf{Z}, \mathbf{z}) = (\psi(\mathbf{z}_1, \mathbf{z}) \psi(\mathbf{z}_2, \mathbf{z}) \cdots \psi(\mathbf{z}_K, \mathbf{z}))^T$.

1.5 Analysis of Existing Work

This section provides a comparison between each of the proposed nonlinear approaches in terms of their computational demand and their ability to represent a generalization of linear PCA. The section finally investigates potential research areas that have not been addressed and require further attention.

1.5.1 Computational Issues

This subsection investigates computationally related issues for principal curves and manifolds first, followed by the analysis of neural network techniques and finally the Kernel PCA approach.

Principal curve and manifold approaches

Resulting from the fact that the nearest projection coordinate of each sample in the curve is searched along the whole line segments, the computational complexity of the HSPCs algorithm is of order $O(n^2)$ [25] which is dominated by the projection step. The HSPCs algorithm, as well as other algorithms proposed by [69, 4, 18, 19], may therefore be computationally expensive for large data sets.

For addressing the computational issue, several strategies are proposed in subsequently refinements. In reference [8], the PPS algorithm supposes that the data are generated from a collection of latent nodes in low-dimensional space, and the computation to determine the projections is achieved by comparing the distances among data and the high-dimensional counterparts in the latent nodes. This results in a considerable reduction in the computational complexity if the number of the latent nodes is less than that number of observations. However, the PPS algorithm requires additional $O(N^2n)$ operations (Where n is the dimension of latent space) to compute an orthonormalization. Hence, this algorithm is difficult to generalize in high-dimensional spaces.

In [73], local principal component analysis in each neighborhood is employed for searching a local segment. Therefore, the computational complexity is closely relate to the number of local PCA models. However, it is difficulty for general data to combine the segments into a principal curve because a large number of computational steps are involved in this combination.

For the work by Kégl [34, 35], the KPCs algorithm is proposed by combining the vector quantization with principal curves. Under the assumption that data have finite second moment, the computational complexity of the KPCs algorithm is $O(n^{5/3})$ which is slightly less than that of the HSPCs algorithm. When allowing to add more than one vertex at a time, the complexity can be significantly decreased. Furthermore, a speed-up strategy discussed by Kégl [33] is employed for the assignments of projection indices for the data during the iterative projection procedure of the ACKPCs algorithms. If $\delta v^{(j)}$ is the maximum shift of a vertex v_j in the j th optimization step defined by:

$$\delta v^{(j)} = \max_{i=1, \dots, k+1} \|v_i^{(j)} - v_i^{(j+1)}\|,$$

then after the $(j + j_1)$ optimization step, s_{i_1} is still the nearest line segment to x if

$$d(x, s_{i_1}^{(j)}) \leq d(x, s_{i_2}^{(j)}) - 2 \sum_{l=j}^{j+j_1} \delta v^{(l)}. \quad (1.67)$$

Further reference to this issue may be found in [33], pp. 66-68. Also, the stability of the algorithm is enhanced while the complexity is the equivalent to that of the KPCs algorithm.

Neural network approaches

The discussion in Subsection 4.2 highlighted that neural network approaches to determine a NLPICA model are difficult to train, particularly the 5 layer network by Kramer [37]. More precisely, the network complexity increases considerably if the number of original variables \mathbf{z} , N , rises. On the other hand, an increasing number of observations also contribute to a drastic increase in the computational cost. Since most of the training algorithms are iterative in nature and employ techniques based on the backpropagation principle, for example the Levenberg-Marquardt algorithm for which the Jacobian matrix is updated using backpropagation, the performance of the identified network depends on the initial selection for the network weights. More precisely, it may be difficult to determine a minimum for the associated cost function, that is the sum of the minimum distances between the original observations and the reconstructed ones.

The use of the IT network [68] and the approach by Dong and McAvoy [16], however, provide considerably simpler network topologies that are accordingly easier to train. Jia *et al.* [31] argued that the IT network can generically represent smooth nonlinear functions and raised concern about the technique by Dong and McAvoy in terms of its flexibility in providing generic nonlinear functions. This concern related to the concept of incorporating a linear combination of nonlinear function to estimate the nonlinear interrelationships between the recorded observations. It should be noted, however, that the IT network structure relies on the condition that a functional injective relationship exists between the score variables and the original variables, that is a unique mapping between the scores and the observations exist. Otherwise, the optimization step to determine the scores from the observations using the identified IT network may converge to different sets of score values depending on the initial guess, which is undesirable. In contrast, the technique by Dong and McAvoy does not suffer from this problem.

Kernel PCA

In comparison to neural network approaches, the computational demand for a KPCA insignificantly increase for larger values of N , size of the original variables set \mathbf{z} , which follows from (1.59). In contrast, the size of the Gram matrix increases quadratically with a rise in the number of analyzed observations, K . However, the application of the numerically stable singular value decomposition to obtain the eigenvalues and eigenvectors of the Gram matrix does not present the same computational problems as those reported for the neural network approaches above.

1.5.2 Generalization of Linear PCA?

The generalization properties of NLPICA techniques is first investigated for neural network techniques, followed for principal curve techniques and finally

kernel PCA. Prior to this analysis, however, we revisit the cost function for determining the k th pair of the score and loading vectors for linear PCA. This analysis is motivated by the fact that neural network approaches as well as principal curves and manifolds minimize the residual variances. Reformulating Equations (1.9) and (1.10) to minimize the residual variance for linear PCA gives rise to:

$$\mathbf{e}_k = \mathbf{z} - t_k \mathbf{p}_k , \quad (1.68)$$

which is equal to:

$$J_k = E \{ \mathbf{e}_k^T \mathbf{e}_k \} = E \left\{ (\mathbf{z} - t_k \mathbf{p}_k)^T (\mathbf{z} - t_k \mathbf{p}_k) \right\} , \quad (1.69)$$

and subject to the following constraints

$$t_k^2 - \mathbf{p}_k^T \mathbf{z} \mathbf{z}^T \mathbf{p}_k = 0 \quad \mathbf{p}_k^T \mathbf{p}_k - 1 = 0 . \quad (1.70)$$

The above constraints follow from the fact that an orthogonal projection of an observation, \mathbf{z} , onto a line, defined by \mathbf{p}_k is given by $t_k = \mathbf{p}_k^T \mathbf{z}$ if \mathbf{p}_k is of unit length. In a similar fashion to the formulation proposed by Anderson [2] for determining the PCA loading vectors in (1.11), (1.69) and (1.70) can be combined to produce:

$$J_k = \arg \min_{\mathbf{p}_k} \left\{ E \left\{ (\mathbf{z} - t_k \mathbf{p}_k)^T (\mathbf{z} - t_k \mathbf{p}_k) - \lambda_k^{(1)} (t_k^2 - \mathbf{p}_k^T \mathbf{z} \mathbf{z}^T \mathbf{p}_k) \right\} - \lambda_k^{(2)} (\mathbf{p}_k^T \mathbf{p}_k - 1) \right\} . \quad (1.71)$$

Carrying out the differentiation of J_k with respect to \mathbf{p}_k yields:

$$E \left\{ 2t_k^2 \mathbf{p}_k - 2t_k \mathbf{z} + 2\lambda_k^{(1)} \mathbf{z} \mathbf{z}^T \mathbf{p}_k \right\} - 2\lambda_k^{(2)} \mathbf{p}_k = \mathbf{0} . \quad (1.72)$$

A pre-multiplication of (1.72) by \mathbf{p}_k^T now reveals

$$E \left\{ \underbrace{t_k^2 - \mathbf{p}_k^T \mathbf{z} \mathbf{z}^T \mathbf{p}_k}_{=0} + \lambda_k^{(1)} \underbrace{\mathbf{p}_k^T \mathbf{z} \mathbf{z}^T \mathbf{p}_k}_{=t_k^2} - \lambda_k^{(2)} \right\} = 0 . \quad (1.73)$$

It follows from Equation (1.73) that

$$E \{ t_k^2 \} = \frac{\lambda_k^{(2)}}{\lambda_k^{(1)}} . \quad (1.74)$$

Substituting (1.74) into Equation (1.72) gives rise to

$$\frac{\lambda_k^{(2)}}{\lambda_k^{(1)}} \mathbf{p}_k + E \left\{ \lambda_k^{(1)} \mathbf{z} \mathbf{z}^T \mathbf{p}_k - \mathbf{z} \mathbf{z}^T \mathbf{p}_k \right\} - \lambda_k^{(2)} \mathbf{p}_k = \mathbf{0} . \quad (1.75)$$

Utilizing (1.5), the above equation can be simplified to

$$\left(\lambda_k^{(2)} - 1\right) \mathbf{S}_{ZZ} \mathbf{p}_k + \left(\frac{\lambda_k^{(2)}}{\lambda_k^{(1)}} - \lambda_k^{(2)}\right) \mathbf{p}_k = \mathbf{0}, \quad (1.76)$$

and, hence,

$$\left[\mathbf{S}_{ZZ} + \frac{\lambda_k^{(2)}}{\lambda_k^{(1)}} \frac{1 - \lambda_k^{(1)}}{\lambda_k^{(2)} - 1} \mathbf{I} \right] \mathbf{p}_k = [\mathbf{S}_{ZZ} - \lambda_k \mathbf{I}] \mathbf{p}_k = \mathbf{0} \quad (1.77)$$

with $\lambda_k = \frac{\lambda_k^{(2)}}{\lambda_k^{(1)}} \frac{1 - \lambda_k^{(1)}}{\lambda_k^{(2)} - 1}$. Since Equation (1.77) is identical to Equation (1.14), maximizing the variance of the score variables produces the same solution as minimizing the residual variance by orthogonally projecting the observations onto the k th weight vector. It is interesting to note that a closer analysis of Equation (1.74) yields that $E\{t_k^2\} = \frac{\lambda_k^{(2)}}{\lambda_k^{(1)}} = \lambda_k$, according to Equation (1.9), and hence, $\lambda_k^{(1)} = \frac{2}{1 + \lambda_k}$ and $\lambda_k^{(2)} = 2 \frac{\lambda_k}{1 + \lambda_k}$, which implies that $\lambda_k^{(2)} \neq 1$ and $\frac{\lambda_k^{(2)}}{\lambda_k^{(1)} - \lambda_k^{(2)}} = \lambda_k > 0$.

More precisely, minimizing residual variance of the projected observations and maximizing the score variance are equivalent formulations. This implies that determining a NLPCA model using a minimizing of the residual variance would produce an equivalent linear model if the nonlinear functions are simplified to be linear. This is clearly the case for principal curves and manifolds as well as the neural network approaches. In contrast, the kernel PCA approach computes a linear PCA analysis using nonlinearly transformed variables and directly addresses the variance maximization and residual minimization as per the discussion above.

Neural network approaches

It should also be noted, however, that residual variance minimization alone is a necessary but not a sufficient condition. This follows from the analysis of the ANN topology proposed by Kramer [37] in Figure 1.6. The nonlinear scores, which can be extracted from the bottleneck layer, do not adhere to the fundamental principle that the first component is associated with the largest variance, the second component with the second largest variance etc. However, utilizing the sequential training of the ANN, detailed in Figure 1.7, provides an improvement, such that the first nonlinear score variables minimizes the residual variance $\mathbf{e}_1 = \mathbf{z} - \hat{\mathbf{z}}$ and so on. However, given that the network weights and bias terms are not subject to a length restriction as it is the case for linear PCA, this approach does also not guarantee that the first score variables possesses a maximum variance.

The same holds true for the IT network algorithm by Tan and Mavrovouniotis [68], the computed score variables do not adhere to the principle that the

first one has a maximum variance. Although score variables may not be extracted that maximize a variance criterion, the computed scores can certainly be useful for feature extraction [15, 62]. Another problem of the technique by Tan and Mavrovouniotis is its application as a condition monitoring tool. Assuming the data describe a fault condition the score variables are obtained by an optimization routine to best reconstruct the fault data. It therefore follows that certain fault conditions may not be noticed. This can be illustrated using the following linear example

$$\mathbf{z}_f = \mathbf{z} + \mathbf{f} \implies \mathbf{P}(\mathbf{z}_0 + \mathbf{f}) , \quad (1.78)$$

where \mathbf{f} represents a step type fault superimposed on the original variable set \mathbf{z} to produce the recorded fault variables \mathbf{z}_f . Separating the above equation produces by incorporating the statistical first order moment:

$$E\{\mathbf{z}_0 + \mathbf{f}_0\} + \mathbf{P}_0^{-T} \mathbf{P}_1^T E\{\mathbf{z}_1 + \mathbf{f}_1\} = \mathbf{P}_0^{-T} \mathbf{t} , \quad (1.79)$$

where the subscript $-T$ is the transpose of an inverse matrix, respectively, $\mathbf{P}^T = [\mathbf{P}_0^T \mathbf{P}_1^T]$, $\mathbf{z}^T = (\mathbf{z}_0 \mathbf{z}_1)$, $\mathbf{f}^T = (\mathbf{f}_0 \mathbf{f}_1)$, $\mathbf{P}_0 \in \mathbb{R}^{n \times n}$, $\mathbf{P}_1 \in \mathbb{R}^{N-n \times n}$, \mathbf{z}_0 and $\mathbf{f}_0 \in \mathbb{R}^N$, and \mathbf{z}_1 and $\mathbf{f}_1 \in \mathbb{R}^{N-n}$. Since the expectation of the original variables are zero, Equation (1.79) becomes:

$$\mathbf{f}_0 + \mathbf{P}_0^{-T} \mathbf{P}_1^T \mathbf{f}_1 = \mathbf{0} \quad (1.80)$$

which implies that if the fault vector \mathbf{f} is such that $\mathbf{P}_0^{-T} \mathbf{P}_1^T \mathbf{f}_1 = -\mathbf{f}_0$ the fault condition cannot be detected using the computed score variables. However, under the assumption that the fault condition is a step type fault but the variance of \mathbf{z} remains unchanged, the first order moment of the residuals would clearly be affected since

$$E\{\mathbf{e}\} = E\{\mathbf{z} + \mathbf{f} - \mathbf{Pt}\} = \mathbf{f} . \quad (1.81)$$

However, this might not hold true for an NLPCA model, where the PCA model plane, constructed from the retained loading vectors, becomes a surface. In this circumstances, it is possible to construct incipient fault conditions that remain unnoticed given that the optimization routine determines scores from the faulty observations and the IT network that minimize the mismatch between the recorded and predicted observations.

Principal curves and manifolds

By virtue of its construction, a principle curve for a two variable example can geometrically provide an NLPCA model that reduces to linear PCA if the variable interrelationship is linear. A principal manifold, on the other hand, suffers from the same problem as neural network approaches, i.e. it is difficult to extract score variables that have the same intrinsic maximization properties as those determined by linear PCA.

Kernel PCA

Since a SVD is one mathematical tool to determine a PCA decomposition from a given data covariance or correlation matrix, the application of an SVD to the Gram matrix, to produce the score variables, will inherit the properties of linear PCA. In this respect, if the nonlinear transformation of the original variables is replaced by a linear identity mapping, kernel PCA reduces to linear PCA and therefore constitutes a true nonlinear extension to PCA.

1.5.3 Roadmap for Future Developments (Basics and Beyond)

Here, we discuss a number of issues that have only sporadically been addressed in the research literature and need, in our opinion, further attention by the research community.

Dynamic NLPCA extensions

Issues that have only been sporadically addressed are dynamic extensions of NLPCA techniques, with a notable exception being [13]. The additional computational complexity in the light of dynamic extensions mainly contributed to the lack research work being proposed thus far. However, the work in reference [13] advocates that the use of kernel PCA is a preferred technique. This confirms our analysis in the previous subsection, which raised concern about the computational demanding for training neural network based NLPCA approaches and the fact that the scores, determined by a principal manifold, do not adhere to the principle of maximum variance.

Adaptive NLPCA modeling

Adaptive modeling of nonlinear variable interrelationships is another aspects that requires a considerable research effort to develop mature, robust and efficient algorithms. This is of particular concern for process monitoring applications of systems that are time-varying and nonlinear in nature.

Parameterization of kernel functions

Although the recently proposed kernel PCA appears to be computationally efficient and maintains the properties of linear PCA, a fundamental issue that has not received considerable attention is the parameterization of the kernel functions $\psi(\cdot)$. A general framework as to which kernel function is to be preferred for certain data pattern has not been introduced. These issues will certainly impact the performance of the kernel PCA model and need to be addressed by future research.

Extension of kernel techniques to other uniblock techniques

The extension of kernel methods to be produce nonlinear extensions other approaches that rely on the analysis of a single variable set, e.g. fisher's discriminant analysis and independent component analysis has also not received much attention in the research literature and would be an area of considerable interest for pattern, face and speech recognition as well as general feature extraction problems.

Nonlinear subspace identification

Subspace identification has been extensively studied over the past decade. This technique enables the identification of a linear state space model using input/output observations of the process. Nonlinear extensions of subspace identification have been proposed in references [76, 74, 43, 23, 41] mainly employ Hammerstein or Wiener models to represent a nonlinear steady state transformation of the process outputs. As this is restrictive, kernel PCA may be considered to determine nonlinear filters to efficiently determine this nonlinear transformation.

Variable contributions

Finally, it may be of importance to determine how an individual variable contributes to the determination of a nonlinear score variable. This issue features in process monitoring applications, where the variable contribution provides vital information for determining the root cause of abnormal process behaviour. Another area of interest is feature extraction, where a certain set of variables is responsible for a specific property observed.

1.6 Concluding Summary

This article has reviewed and critically analyzed work on nonlinear principal component analysis. The revision showed that a nonlinearity test can be applied to determine whether a conceptually and computationally more demanding NLPCA model is required. The article then showed that 3 principal directions for developing NLPCA algorithms have emerged. The first of these relate to principal curves that were initially proposed for variable sets including two variables. Extensions of principal curves are principal manifolds, which inherit the same underlying theory. Neural network implementation represent the second research direction for implementing NLPCA. These determine nonlinear score variables either by the reduced bottleneck layer of an autoassociative neural network or by a reduced input layer whose inputs are determined by an input-training network or a predetermined principal curve.

Finally, the third research direction is the recently proposed kernel PCA approach.

The analysis into (i) computational issues and (ii) their generalization of linear PCA yielded the following. Principal curves and manifolds are conceptually simple but computationally demanding for larger data and variable sets. Furthermore, whilst principal curves do produce a maximum covariance of the score variables in a similar fashion to linear PCA if only two variables are analyzed, the score obtained by a principal manifold for high-dimensional problems do not adhere to this maximization principle. NLPCA implementations based on autoassociative neural networks are cumbersome as a result of excessive computation for training the unknown network weights. Although the computationally less demanding IT networks and the incorporation of principal curves considerably reduce network complexity, neural network approaches produce nonlinear score variables that are not obtained with respect to a maximum variance criterion either. Consequently, principal manifolds and neural network approaches have been utilized in pattern, face and speech recognition, as well as feature extraction for example, they violate one fundamental principal of linear PCA, namely that of maximizing the variance of the score variables. Kernel PCA, on the other hand, apply a SVD, or a PCA analysis, on a larger set of nonlinearly transformed variables. Hence, the score variables are obtained such that the first one possesses a maximum variance, the second one the second largest variance and so on.

The paper finally outlines research areas concerning NLPCA developments and applications that require further research efforts. These include dynamic NLPCA extensions, adaptive NLPCA modelling, the parameterization of kernel functions to construct a kernel PCA model, extensions of the kernel approach to (i) other uniblock techniques such as FDI and ICA and (ii) nonlinear subspace identification and finally the evaluation of variable contributions to individual nonlinear score variables. These points have either only received sporadic attention or have not been investigated to the best of the authors knowledge.

References

1. Abdel-Qadar, I., Pashaie-Rad, S., Abudayeh, O., and Yehia, S.: PCA-based algorithm for unsupervised bridge crack detection. *Advances in Engineering Software*, **37** (12), 771–778 (2006)
2. Anderson, T.W.: *An Introduction into Multivariate Statistical Analysis*. John Wiley & Sons, New York, (1958)
3. Bakshi, B.R., Multiscale pca with application to multivariate statistical process monitoring. *AICHE Journal*, **44** (7), 1596–1610 (1998)
4. Banfield, J.D. and Raftery A.E.: Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *Journal of the American Statistical Association*, **87** (417), 7–16 (1992)

5. Barnett, V.: *Interpreting Multivariate Data*. John Wiley & Sons, New York (1981)
6. Sevensen M., Bishop, C.M., and Williams C.K.I.: GTM: The generative topographic mapping. *Neural Computation*, **10**, 215–234 (1998)
7. Chang, K. and Ghosh, J.: Principal curve classifier - a nonlinear approach to pattern classification. In: IEEE International Joint Conference on Neural Networks, 4-9 May 1998, 695–700, Anchorage, Alaska (1998)
8. Chang, K. and Ghosh, J.: A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23** (1), 22–41 (2001)
9. Chen, D., Zhang, J., Tang, S., and Wang J.: Freeway traffic stream modelling based on principal curves and its analysis. *IEEE Transactions on Intelligent Transportation Systems*, **5** (4), 246–258 (2004)
10. Chen, P. and Suter, D.: An analysis of linear subspace approaches for computer vision and pattern recognition. *International Journal of Computer Vision*, **68** (1), 83–106 (2006)
11. Chennubhotla, C. and Jepson, A.: Sparse pca extracting multi-scale structure from data. In: Proceedings of the IEEE International Conference on Computer Vision, vol. 1, 641–647 (2001)
12. Cho, H.W.: Nonlinear feature extraction and classification of multivariate process data in kernel feature space. *Expert Systems with Applications*, **32** (2), 534–542 (2007)
13. Choi, S.W. and Lee, I.-B.: Nonlinear dynamic process monitoring based on dynamic kernel pca. *Chemical Engineering Science*, **59** (24), 5897–5908 (2004)
14. Delicado, P.: Another look at principal curves and surfaces. *Journal of Multivariate Analysis*, **77** (1), 84–116 (2001)
15. Denoeux, T. and Masson, M.-H.: Principal component analysis of fuzzy data using autoassociative neural networks. *IEEE Transactions on Fuzzy Systems*, **12** (3), 336–349 (2004)
16. Dong, D. and McAvoy, T.J.: Nonlinear principal component analysis-based on principal curves and neural networks. *Computers & Chemical Engineering*, **20** (1), 65–78 (1996)
17. Du, Q. and Chang, C.: Linear mixture analysis-based compression for hyperspectral image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, **42** (4), 875–891 (2004)
18. Duchamp, T. and Stuetzle, W.: Extremal properties of principal curves in the plane. *Annals of Statistics*, **24** (4), 1511–1520 (1996)
19. Duchamp, T. and Stuetzle, W.: Geometric Properties of Principal Curves in the Plane. In: Robust statistics, data analysis, and computer intensive methods: in honor of Peter Huber's 60th birthday, (Lecture Notes in Statistics), vol. 109, 135–152. Springer, New York (1996)
20. Esbensen, K. and Geladi, P.: Strategy of multivariate image analysis (MIA). *Chemometrics & Intelligent Laboratory Systems*, **7** (1-2), 67–86 (1989)
21. Fisher, R. and MacKenzie, W.: Studies in crop variation, ii, the manurial response of different potato varieties. *Journal of Agricultural Science*, **13**, 411–444 (1923)
22. Golub, G.H. and van Loan, C.F.: *Matrix Computation*. John Hopkins, Baltimore, (1996)

23. Gomez, J.C. and Baeyens, E.: Subspace-based identification algorithms for hammerstein and wiener models. *European Journal of Control*, **11** (2), 127–136 (2005)
24. Hastie, T.: Principal curves and surfaces. Technical report no. 11, Department of Statistics, Stanford University (1984)
25. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association* **84** (406), 502–516 (1989)
26. Hermann, T., Meinicke, P., and Ritter, H.: Principal curve sonification. In: Proceedings of International Conference on Auditory Display, 2-5 April 2000, Atlanta, Georgia, 81–86 (2000)
27. Hertz, J., Krogh, A., and Palmer, R.G.: Introduction to the Theory of Neural Computing. Addison-Wesley, Redwood City, CA (1991)
28. Hotelling, H.: Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, **24** 417–441 (1933)
29. Jackson, J.E.: Principal components and factor analysis: Part III: What is factor analysis. *Journal of Quality Technology*, **13** (2), 125–130 (1981)
30. Jackson, J.E.: A Users Guide to Principal Components. Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York (1991)
31. Jia, F., Martin, E.B., and Morris, A.J.: Non-linear principal component analysis for process fault detection. *Computers & Chemical Engineering*, **22** (Supplement), S851–S854 (1998)
32. Jolliffe, I.T.: Principal Component Analysis. Springer, New York, (1986)
33. Kégl, B.: Principal Curves: Learning, Design and Applications. PhD thesis, Department of Computer Science, Concordia University, Montréal, Québec, Canada, 2000
34. Kégl, B., Krzyzak, A., Linder, T., and Zeger, K.: A polygonal line algorithm for constructing principal curves. In: Neural Information Processing (NIPS '98), Denver, CO, 1-3 December 1998, 501–507. MIT Press (1998)
35. Kégl, B., Krzyzak, A., Linder, T., and Zeger, K.: Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22** (3), 281–297 (2000)
36. Kim, K.I., Jung, K., and Kim, H. J.: Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, **9** (2), 40–42 (2002)
37. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, **37** (3), 233–243 (1991)
38. Kruger, U., Antory, D., Hahn, J., Irwin, G.W., and McCullough, G.: Introduction of a nonlinearity measure for principal component models. *Computers & Chemical Engineering*, **29** (11-12), 2355–2362 (2005)
39. Krzanowski, W. J.: Cross-validatory choice in principal component analysis: Some sampling results. *Journal of Statistical Computation and Simulation*, **18**, 299–314 (1983)
40. Kwok, J.T.Y. and Tsang, I.W.H.: The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, **15** (6), 1517–1525 (2004)
41. Lacy, S.L. and Bernstein, D. S.: Subspace identification for non-linear systems with measured input non-linearities. *International Journal of Control*, **78** (12), 906–921 (2005)
42. Leeuw J. d.: Nonlinear principal component analysis. In: Caussinus, H., Etinger, P., and Tomassone, R. (eds) *Proceedings in Computational Statistics (COMPSTAT 1982)* October 30 – September 3, Toulouse, France 1982. Physica-Verlag, Wien (1982)

43. Lovera, M., Gustafsson, T., and Verhaegen, M.: Recursive subspace identification of linear and nonlinear wiener type models. *Automatica*, **36** (11), 1639–1650 (2000)
44. Malinowski, E.R.: Factor Analysis in Chemistry. John Wiley & Sons, New York (2002)
45. Mardia, K.V., Kent, J.T., and Bibby, J.M.: Multivariate Analysis. Probability and Mathematical Statistics. Academic Press, London, (1979)
46. Morales, M.: Geometric Data Fitting. PhD thesis, University of Washington (1998)
47. Nara, Y., Jianming Y., and Suematsu, Y.: Face recognition using improved principal component analysis. In: Proceedings of 2003 International Symposium on Micromechatronics and Human Science (IEEE Cat. No.03TH8717), 77–82 (2003)
48. Nomikos, P. and MacGregor, J.F.: Monitoring of batch processes using multiway principal component analysis. *AICHE Journal*, **40** (8), 1361–1375 (1994)
49. Paluš, M. and Dvořák, I.: Singular-value decomposition in attractor reconstruction: Pitfalls and precautions. *Physica D: Nonlinear Phenomena*, **5** (1-2), 221–234 (1992)
50. Parsopoulos, K.E. and Vrahatis, M. N.: Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, **1** (2-3), 235–306 (2002)
51. Pearson, C.: On lines and planes of closest fit to systems of points in space. *Phil. Mag.*, Series B., **2** (11), 559–572 (1901)
52. Qin, S.J., Valle, S., and Piovoso, M. J.: On unifying multi-block analysis with application to decentralized process monitoring. *Journal of Chemometrics*, **10**, 715–742 (2001)
53. Reinhard, K., and Niranjan, M.: Subspace models for speech transitions using principal curves. *Proceedings of Institute of Acoustics*, **20** (6), 53–60 (1998)
54. Reinhard, K., and Niranjan, M.: Parametric subspace modeling of speech transitions. *Speech Communication*, **27** (1), 19–42 (1999)
55. Sandilya, S. and Kulkarni, S.R.: Principal curves with bounded turn. In: Proceedings of the IEEE International Symposium on Information Theory, Sorento, 25-30 June 2000, Sorento, Italy (2000)
56. Schölkopf, B. and Smola, A. J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press, Cambridge, MA (2002)
57. Schölkopf, B. and Smola, A. J., and Müller, K.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, **10** (5), 1299–1319 (1998)
58. Shanmugam, R. and Johnson, C.: At a crossroad of data envelopment and principal component analyses. *Omega*, **35** (4), 351–364 (2007)
59. Shawe-Taylor, J. and Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, West Nyack, NY (2004)
60. Silverman, B. W.: Some aspects of spline smoothing. *Journal of the Royal Statistical Society, Series B*, **47** (1), 1–52 (1985)
61. Smola, A.J., Williamson, R.C., Mika, S., and Schölkopf, B.: Regularized principal manifolds. In: Fischer, P., and Simon, H.U. (eds.) Computational Learning Theory (EuroCOLT '99), Lecture Notes in Artificial Intelligence, vol. 1572, 214–229, Springer, Heidelberg (1999)
62. Socas-Navarro, H.: Feature extraction techniques for the analysis of spectral polarization profiles. *Astrophysical Journal*, **620**, 517–522 (2005)

63. Srinivas, M. and Patnaik, L.M.: Genetic algorithms: A survey. *Computer*, **27** (6), 17–26 (1994)
64. Stoica, P., Eykhoff, P., Janssen, P., and Söderström, T.: Model structure selection by cross-validation. *International Journal of Control*, **43** (6), 1841–1878 (1986)
65. Stone, M.: Cross-validatory choice and assessment of statistical prediction (with discussion). *Journal of the Royal Statistical Society (Series B)*, **36**, 111–133 (1974)
66. Stoyanova, R. and Brown, T. R.: Nmr spectral quantitation by principal component analysis. *NMR in Biomedicine*, **14** (4), 271–277 (2001)
67. Sylvester, J.J.: On the reduction of a bilinear quantic of the n th order to the form of a sum of n products by a double orthogonal substitution. *Messenger of Mathematics*, **19**, 42–46 (1889)
68. Tan, S. and Mavrovouniotis, M.L.: Reducing data dimensionality through optimizing neural network inputs. *AICHE Journal*, **41** (6), 1471–1480 (1995)
69. Tibshirani, R.: Principal curves revisited. *Statistics and Computation*, **2** (4), 183–190 (1992)
70. Trafalis, T.B., Richman, M.B., White, A., and Santosa, B.: Data mining techniques for improved wsrf-88d rainfall estimation. *Computers & Industrial Engineering*, **43** (4), 775–786 (2002)
71. Huffel, S. van, and Vandewalle, J.: *The Total Least Squares Problem: Computational Aspects and Analysis*. SIAM, Philadelphia (1991)
72. Vaswani, N. and Chellappa, R.: Principal components null space analysis for image and video classification. *IEEE Transactions on Image Processing*, **15** (7), 1816–1830 (2006)
73. Vlassis, N., Verbeek, J.J., and Kr'ose, B.: K-segments algorithm for finding principal curves. Technical Report IAS-UVA-00-11, Institute of Computer Science, University of Amsterdam (2000)
74. Verhaegen, M. and Westwick, D.: Identifying mimo hammerstein systems in the context of subspace model identification methods. *International Journal of Control*, **63** (2), 331–350 (1996)
75. Wax, M. and Kailath, T.: Detection of signals by information theoretic criteria. *IEEE Transactions on Acoustics, Speech & Signal Processing*, **33** (2), 387–392 (1985)
76. Westwick, D. and Verhaegen, M.: Identifying mimo wiener systems using subspace model identification methods. *Signal Processing*, **52** (2), 235–258 (1996)
77. Wise, B.M. and Ricker, N.L.: Identification of finite impulse response models by principal components regression: Frequency response properties. *Process Control & Quality*, **4**, 77–86 (1992)
78. Wold, H.: Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P.R. (ed.) *Multivariate Analysis*, 391–420. Academic Press, N.Y. (1966)
79. Wold, S.: Cross validatory estimation of the number of principal components in factor and principal component models. *Technometrics*, **20** (4), 397–406 (1978)
80. Wold, S., Esbensen, K., and Geladi, P.: Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, **2**, 37–52 (1987)
81. Yoo, C.K. and Lee, I.: Nonlinear multivariate filtering and bioprocess monitoring for supervising nonlinear biological processes. *Process Biochemistry*, **41** (8), 1854–1863 (2006)

82. Zeng, Z. and Zou, X.: Application of principal component analysis to champ radio occultation data for quality control and a diagnostic study. *Monthly Weather Review*, **134** (11), 3263–3282 (2006)
83. Zhang, J. and Chen, D.: Constraint k-segment principal curves. In: Huang, De-Sh., Li, K. and Irwin, G.W. (eds.) *Intelligent Computing, Lecture Notes in Computer Sciences*, vol. 4113, 345–350. Springer, Berlin Heidelberg New York (2006)
84. Zhang, J., Chen, D., and Kruger, U.: Constrained k-segments principal curves and its applications in intelligent transportation systems. Technical report, Department of Computer Science and Engineering, Fudan University, Shanghai, P. R. China (2007)
85. Zhang, J. and Wang, J.: An overview of principal curves (in chinese). *Chinese Journal of Computers*, **26** (2), 137–148 (2003)

Nonlinear Principal Component Analysis: Neural Network Models and Applications

Matthias Scholz¹, Martin Fraunholz¹, and Joachim Selbig²

¹ Institute for Microbiology, Ernst-Moritz-Arndt-University Greifswald,
F.-L.-Jahn-Str. 15, 17487 Greifswald, Germany,

Matthias.Scholz@uni-greifswald.de
Martin.Fraunholz@uni-greifswald.de

² Institute for Biochemistry and Biology, University of Potsdam,
c/o Max Planck Institute for Molecular Plant Physiology
Am Mühlenberg 1, 14424 Potsdam, Germany,
Selbig@mpimp-golm.mpg.de

Summary. *Nonlinear principal component analysis* (NLPCA) as a nonlinear generalisation of standard *principal component analysis* (PCA) means to generalise the principal components from straight lines to curves. This chapter aims to provide an extensive description of the autoassociative neural network approach for NLPCA. Several network architectures will be discussed including the hierarchical, the circular, and the inverse model with special emphasis to missing data. Results are shown from applications in the field of molecular biology. This includes metabolite data analysis of a cold stress experiment in the model plant *Arabidopsis thaliana* and gene expression analysis of the reproductive cycle of the malaria parasite *Plasmodium falciparum* within infected red blood cells.

2.1 Introduction

Many natural phenomena behave in a nonlinear way meaning that the observed data describe a curve or curved subspace in the original data space. Identifying such nonlinear manifolds becomes more and more important in the field of molecular biology. In general, molecular data are of very high dimensionality because of thousands of molecules that are simultaneously measured at a time. Since the data are usually located within a low-dimensional subspace, they can be well described by a single or low number of components. Experimental time course data are usually located within a curved subspace which requires a nonlinear dimensionality reduction as illustrated in Figure 2.1.

Visualising the data is one aspect of molecular data analysis, another important aspect is to model the mapping from original space to component space

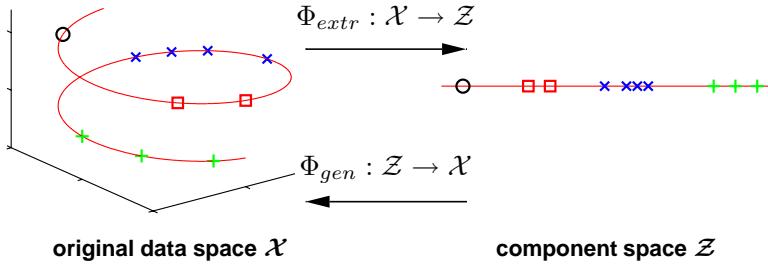


Fig. 2.1. Nonlinear dimensionality reduction. Illustrated are three-dimensional samples that are located on a one-dimensional subspace, and hence can be described without loss of information by a single variable (the component). The transformation is given by the two functions Φ_{extr} and Φ_{gen} . The extraction function Φ_{extr} maps each three-dimensional sample vector (left) onto a one-dimensional component value (right). The inverse mapping is given by the generation function Φ_{gen} which transforms any scalar component value back into the original data space. Such helical trajectory over time is not uncommon in molecular data. The horizontal axes may represent molecule concentrations driven by a circadian rhythm, whereas the vertical axis might represent a molecule with an increase in concentration.

in order to interpret the impact of observed variables on the subspace (component space). Both the component values (scores) and the mapping function is provided by the neural network approach for nonlinear PCA.

Three important extensions of nonlinear PCA are discussed in this chapter: the hierarchical NLPCA, the circular PCA, and the inverse NLPCA. All of them can be used in combination. *Hierarchical NLPCA* means to enforce the nonlinear components to have the same hierarchical order as the linear components of standard PCA. This hierarchical condition yields a higher meaning of individual components. *Circular PCA* enables nonlinear PCA to extract circular components which describe a closed curve instead of the standard curve with an open interval. This is very useful for analysing data from cyclic or oscillatory phenomena. *Inverse NLPCA* defines nonlinear PCA as an inverse problem, where only the assumed data generation process is modelled, which has the advantage that more complex curves can be identified and NLPCA becomes applicable to incomplete data sets.

Bibliographic notes

Nonlinear PCA based on autoassociative neural networks was investigated in several studies [1, 2, 3, 4]. Kirby and Miranda [5] constrained network units to work in a circular manner resulting in a *circular PCA* whose components are closed curves. In the fields of atmospheric and oceanic sciences, circular PCA is applied to oscillatory geophysical phenomena, for example, the ocean-atmosphere El Niño-Southern oscillation [6] or the tidal cycle at the German

North Sea coast [7]. There are also applications in the field of robotics in order to analyse and control periodic movements [8]. In molecular biology, circular PCA is used for gene expression analysis of the reproductive cycle of the malaria parasite *Plasmodium falciparum* in red blood cells [9]. Scholz and Vigário [10] proposed a *hierarchical nonlinear PCA* which achieves a hierarchical order of nonlinear components similar to standard linear PCA. This hierarchical NLPCA was applied to spectral data of stars and to electromyographic (EMG) recordings of muscle activities. Neural network models for *inverse NLPCA* were first studied in [11, 12]. A more general Bayesian framework based on such inverse network architecture was proposed by Valpola and Honkela [13, 14] for a nonlinear factor analysis (NFA) and a nonlinear independent factor analysis (NIFA). In [15], such inverse NLPCA model was adapted to handle missing data in order to use it for molecular data analysis. It was applied to metabolite data of a cold stress experiment with the model plant *Arabidopsis thaliana*. Hinton and Salakhutdinov [16] have demonstrated the use of the autoassociative network architecture for visualisation and dimensionality reduction by using a special initialisation technique.

Even though the term nonlinear PCA (NLPCA) is commonly referred to as the autoassociative approach, there are many other methods which visualise data and extract components in a nonlinear manner. *Locally linear embedding* (LLE) [17, 18] and *Isomap* [19] were developed to visualise high dimensional data by projecting (embedding) them into a two or low-dimensional space, but the mapping function is not explicitly given. *Principal curves* [20] and *self organising maps* (SOM) [21] are useful for detecting nonlinear curves and two-dimensional nonlinear planes. Practically both methods are limited in the number of extracted components, usually two, due to high computational costs. *Kernel PCA* [22] is useful for visualisation and noise reduction [23].

Several efforts are made to extend independent component analysis (ICA) into a *nonlinear ICA*. However, the nonlinear extension of ICA is not only very challenging, but also intractable or non-unique in the absence of any *a priori* knowledge of the nonlinear mixing process. Therefore, special nonlinear ICA models simplify the problem to particular applications in which some information about the mixing system and the factors (source signals) is available, e.g., by using sequence information [24]. A discussion of nonlinear approaches to ICA can be found in [25, 26]. This chapter focuses on the less difficult task of nonlinear PCA. A perfect nonlinear PCA should, in principle, be able to remove all nonlinearities in the data such that a standard linear ICA can be applied subsequently to achieve, in total, a nonlinear ICA. This chapter is mainly based on [9, 10, 15, 27].

Data generation and component extraction

To extract components, linear as well as nonlinear, we assume that the data are determined by a number of factors and hence can be considered as being generated from them. Since the number of varied factors is often smaller than

the number of observed variables, the data are located within a subspace of the given data space. The aim is to represent these factors by components which together describe this subspace. Nonlinear PCA is not limited to linear components, the subspace can be curved, as illustrated in Figure 2.1.

Suppose we have a data space \mathcal{X} given by the observed variables and a component space \mathcal{Z} which is a subspace of \mathcal{X} . Nonlinear PCA aims to provide both the subspace \mathcal{Z} and the mapping between \mathcal{X} and \mathcal{Z} . The mapping is given by nonlinear functions Φ_{extr} and Φ_{gen} . The *extraction* function $\Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z}$ transforms the sample coordinates $x = (x_1, x_2, \dots, x_d)^T$ of the d -dimensional data space \mathcal{X} into the corresponding coordinates $z = (z_1, z_2, \dots, z_k)^T$ of the component space \mathcal{Z} of usually lower dimensionality k . The *generation* function $\Phi_{gen} : \mathcal{Z} \rightarrow \hat{\mathcal{X}}$ is the inverse mapping which reconstructs the original sample vector x from their lower-dimensional component representation z . Thus, Φ_{gen} approximates the assumed data generation process.

2.2 Standard Nonlinear PCA

Nonlinear PCA (NLPCA) is based on a multi-layer perceptron (MLP) with an autoassociative topology, also known as an autoencoder, replicator network, bottleneck or sandglass type network. An introduction to multi-layer perceptrons can be found in [28].

The autoassociative network performs an identity mapping. The output \hat{x} is enforced to equal the input x with high accuracy. It is achieved by minimising the squared reconstruction error $E = \frac{1}{2} \| \hat{x} - x \|^2$.

This is a nontrivial task, as there is a ‘bottleneck’ in the middle: a layer of fewer units than at the input or output layer. Thus, the data have to be projected or compressed into a lower dimensional representation Z .

The network can be considered to consist of two parts: the first part represents the extraction function $\Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z}$, whereas the second part represents the inverse function, the generation or reconstruction function $\Phi_{gen} : \mathcal{Z} \rightarrow \hat{\mathcal{X}}$. A hidden layer in each part enables the network to perform nonlinear mapping functions. Without these hidden layers, the network would only be able to perform linear PCA even with nonlinear units in the component layer, as shown by Bourlard and Kamp [29]. To regularise the network, a *weight decay* term is added $E_{total} = E + \nu \sum_i w_i^2$ in order to penalise large network weights w . In most experiments, $\nu = 0.001$ was a reasonable choice.

In the following, we describe the applied network topology by the notation $l_1-l_2-l_3\dots-l_S$ where l_s is the number of units in layer s . For example, 3-4-1-4-3 specifies a network of five layers having three units in the input and output layer, four units in both hidden layers, and one unit in the component layer, as illustrated in Figure 2.2.

$$\Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z} \quad \Phi_{gen} : \mathcal{Z} \rightarrow \hat{\mathcal{X}}$$

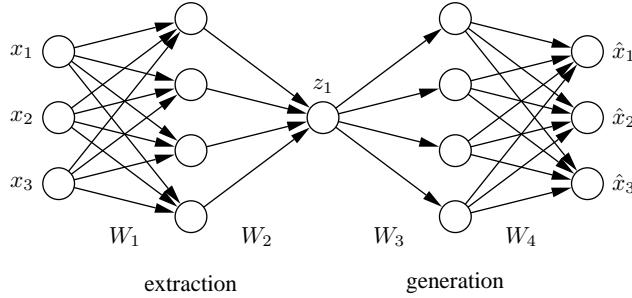


Fig. 2.2. Standard autoassociative neural network. The network output $\hat{\mathbf{x}}$ is required to be equal to the input \mathbf{x} . Illustrated is a 3-4-1-4-3 network architecture. Biases have been omitted for clarity. Three-dimensional samples \mathbf{x} are compressed to one component value z in the middle by the extraction part. The inverse generation part reconstructs $\hat{\mathbf{x}}$ from z . The sample $\hat{\mathbf{x}}$ is usually a noise-reduced representation of \mathbf{x} . The second and fourth layer, with four nonlinear units each, enable the network to perform nonlinear mappings. The network can be extended to extract more than one component by using additional units in the component layer in the middle.

2.3 Hierarchical nonlinear PCA

In order to decompose data in a PCA related way, linearly or nonlinearly, it is important to distinguish applications of pure *dimensionality reduction* from applications where the identification and discrimination of unique and meaningful components is of primary interest, usually referred to as *feature extraction*. In applications of pure dimensionality reduction with clear emphasis on noise reduction and data compression, only a subspace with high descriptive capacity is required. How the individual components form this subspace is not particularly constrained and hence does not need to be unique. The only requirement is that the subspace explains maximal information in the mean squared error sense. Since the individual components which span this subspace, are treated equally by the algorithm without any particular order or differential weighting, this is referred to as symmetric type of learning. This includes the nonlinear PCA performed by the standard autoassociative neural network which is therefore referred to as s-NLPCA.

By contrast, *hierarchical nonlinear PCA (h-NLPCA)*, as proposed by Scholz and Vigário [10], provides not only the optimal nonlinear subspace spanned by components, it also constrains the nonlinear components to have the same hierarchical order as the linear components in standard PCA.

Hierarchy, in this context, is explained by two important properties: scalability and stability. Scalability means that the first n components explain the maximal variance that can be covered by a n -dimensional subspace. Stability means that the i -th component of an n component solution is identical to the

$$\Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z} \quad \Phi_{gen} : \mathcal{Z} \rightarrow \hat{\mathcal{X}}$$

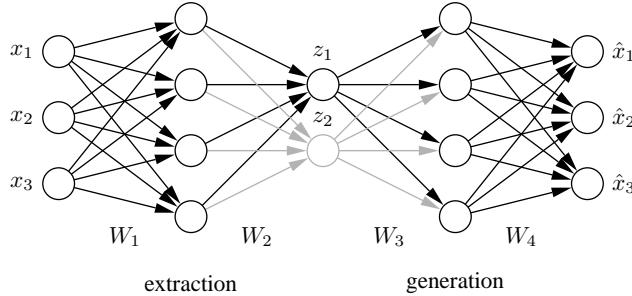


Fig. 2.3. Hierarchical NLPICA. The standard autoassociative network is hierarchically extended to perform the hierarchical NLPICA (h-NLPICA). In addition to the whole 3-4-2-4-3 network (grey+black), there is a 3-4-1-4-3 subnetwork (black) explicitly considered. The component layer in the middle has either one or two units which represent the first and second components, respectively. Both the error E_1 of the subnetwork with one component and the error of the total network with two components are estimated in each iteration. The network weights are then adapted at once with regard to the total hierachic error $E = E_1 + E_{1,2}$.

i -th component of an m component solution.

A hierarchical order essentially yields uncorrelated components. Nonlinearly, this even means that h-NLPICA is able to remove complex nonlinear correlations between components. This can yield useful and meaningful components as will be shown by applications in Section 2.6. Additionally, by scaling the nonlinear uncorrelated components to unit variance, we obtain a complex nonlinear whitening (sphering) transformation [10]. This is a useful pre-processing step for applications such as regression, classification, or blind separation of sources. Since a nonlinear whitening removes the nonlinearities in the data, subsequently applied methods can be linear. This is particularly important for ICA which can be extended to a nonlinear approach by using this nonlinear whitening.

How can we achieve a hierarchical order? The naive approach to simply sort the symmetrically treated components by variance does not yield the required hierarchical order, neither linearly nor nonlinearly. In principle, hierarchy can be achieved by two strongly related ways: either by a constraint to the variance in the component space or by a constraint to the squared reconstruction error in the original space. Similar to linear PCA, the i -th component could be forced to account for the i -th highest variance. But nonlinearly, such constraint can be ineffective or non-unique without additional constraints to the nonlinear transformation. In contrast, the reconstruction error can be much better controlled, since it is an absolute amount, invariant to any scaling in the transformation. A hierarchical constraint to the error is therefore much more effective. In the simple linear case, we can achieve hierarchically ordered

components by a sequential (deflationary) approach in which the components are successively extracted one after the other on the remaining variance given by the squared error of the previous ones. However, this does not work in the nonlinear case, neither successively nor simultaneously by training several networks in parallel. The remaining variance cannot be interpreted by the squared error regardless of the nonlinear transformation [30]. The solution is to use only one network with a hierarchy of subnetworks as illustrated in Figure 2.3. This enables us to formulate the hierarchy directly in the error function [10]. For simplicity, we first restrict our discussion to the case of a two-dimensional component space, but all conclusions can then be generalised to any other dimensionality.

2.3.1 The Hierarchical Error Function

E_1 and $E_{1,2}$ are the squared reconstruction errors when using only the first or both the first and the second component, respectively. In order to perform the h-NLPCA, we have to impose not only a small $E_{1,2}$ (as in s-NLPCA), but also a small E_1 . This can be done by minimising the hierarchical error:

$$E_H = E_1 + E_{1,2} . \quad (2.1)$$

To find the optimal network weights for a minimal error in the h-NLPCA as well as in the standard symmetric approach, the *conjugate gradient descent* algorithm [31] is used. At each iteration, the single error terms E_1 and $E_{1,2}$ have to be calculated separately. This is performed in the standard s-NLPCA way by a network either with one or with two units in the component layer. Here, one network is the subnetwork of the other, as illustrated in Figure 2.3. The gradient ∇E_H is the sum of the individual gradients $\nabla E_H = \nabla E_1 + \nabla E_{1,2}$. If a weight w_i does not exist in the subnetwork, $\frac{\partial E_1}{\partial w_i}$ is set to zero. To achieve more robust results, the network weights are set such that the sigmoidal nonlinearities work in the linear range, corresponding to initialise the network with the simple linear PCA solution.

The hierarchical error function (2.1) can be easily extended to k components ($k \leq d$):

$$E_H = E_1 + E_{1,2} + E_{1,2,3} + \cdots + E_{1,2,3,\dots,k} . \quad (2.2)$$

The hierarchical condition as given by E_H can then be interpreted as follows: we search for a k -dimensional subspace of minimal mean square error (MSE) under the constraint that the $(k-1)$ -dimensional subspace is also of minimal MSE. This is successively extended such that all $1, \dots, k$ dimensional subspaces are of minimal MSE. Hence, each subspace represents the data with regard to its dimensionalities best. Hierarchical nonlinear PCA can therefore be seen as a true and natural nonlinear extension of standard linear PCA.

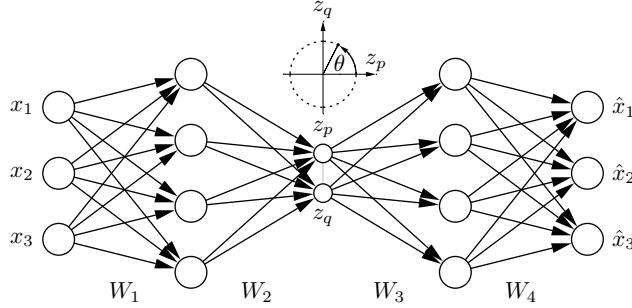


Fig. 2.4. Circular PCA network. To obtain circular components, the auto-associative neural network contains a circular unit pair (p, q) in the component layer. The output values z_p and z_q are constrained to lie on a unit circle and hence can be represented by a single angular variable θ .

2.4 Circular PCA

Kirby and Miranda [5] introduced a circular unit at the component layer in order to describe a potential circular data structure by a closed curve. As illustrated in Figure 2.4, a circular unit is a pair of network units p and q whose output values z_p and z_q are constrained to lie on a unit circle

$$z_p^2 + z_q^2 = 1 . \quad (2.3)$$

Thus, the values of both units can be described by a single angular variable θ .

$$z_p = \cos(\theta) \quad \text{and} \quad z_q = \sin(\theta) . \quad (2.4)$$

The *forward propagation* through the network is as follows: First, equivalent to standard units, both units are weighted sums of their inputs z_m given by the values of all units m in the previous layer.

$$a_p = \sum_m w_{pm} z_m \quad \text{and} \quad a_q = \sum_m w_{qm} z_m . \quad (2.5)$$

The weights w_{pm} and w_{qm} are of matrix W_2 . Biases are not explicitly considered, however, they can be included by introducing an extra input with activation set to one.

The sums a_p and a_q are then corrected by the radial value

$$r = \sqrt{a_p^2 + a_q^2} \quad (2.6)$$

to obtain circularly constraint unit outputs z_p and z_q

$$z_p = \frac{a_p}{r} \quad \text{and} \quad z_q = \frac{a_q}{r} . \quad (2.7)$$

For *backward propagation*, we need the derivatives of the error function

$$E = \frac{1}{2} \sum_n^N \sum_i^d [\hat{\mathbf{x}}_i^n - \mathbf{x}_i^n]^2 \quad (2.8)$$

with respect to all network weights w . The dimensionality d of the data is given by the number of observed variables, N is the number of samples.

To simplify matters, we first consider the error e of a single sample \mathbf{x} , $e = \frac{1}{2} \sum_i^d [\hat{\mathbf{x}}_i - \mathbf{x}_i]^2$ with $\mathbf{x} = (x_1, \dots, x_d)^T$. The resulting derivatives can then be extended with respect to the total error E given by the sum over all n samples, $E = \sum_n e^n$.

While the derivatives of weights of matrices W_1 , W_3 , and W_4 are obtained by standard back-propagation, the derivatives of the weights w_{pm} and w_{qm} of matrix W_2 , which connect units m of the second layer with the units p and q of the component layer, are obtained as follows: We first need the partial derivatives of e with respect to z_p and z_q :

$$\tilde{\sigma}_p = \frac{\partial e}{\partial z_p} = \sum_j w_{jp} \sigma_j \quad \text{and} \quad \tilde{\sigma}_q = \frac{\partial e}{\partial z_q} = \sum_j w_{jq} \sigma_j , \quad (2.9)$$

where σ_j are the partial derivatives $\frac{\partial e}{\partial a_j}$ of units j in the fourth layer.

The required partial derivatives of e in respect to a_p and a_q of the circular unit pair are

$$\sigma_p = \frac{\partial e}{\partial a_p} = (\tilde{\sigma}_p z_q - \tilde{\sigma}_q z_p) \frac{z_q}{r^3} \quad \text{and} \quad \sigma_q = \frac{\partial e}{\partial a_q} = (\tilde{\sigma}_q z_p - \tilde{\sigma}_p z_q) \frac{z_p}{r^3} . \quad (2.10)$$

The final back-propagation formulas for all n samples are

$$\frac{\partial E}{\partial w_{pm}} = \sum_n \sigma_p^n z_m^n \quad \text{and} \quad \frac{\partial E}{\partial w_{qm}} = \sum_n \sigma_q^n z_m^n . \quad (2.11)$$

2.5 Inverse Model of Nonlinear PCA

In this section we define nonlinear PCA as an inverse problem. While the classical forward problem consists of predicting the output from a given input, the inverse problem involves estimating the input which matches best a given output. Since the model or data generating process is not known, this is referred to as a *blind inverse problem*.

The simple linear PCA can be considered equally well either as a forward or inverse problem depending on whether the desired components are predicted as outputs or estimated as inputs by the respective algorithm. The autoassociative network models both the forward and the inverse model simultaneously. The forward model is given by the first part, the extraction

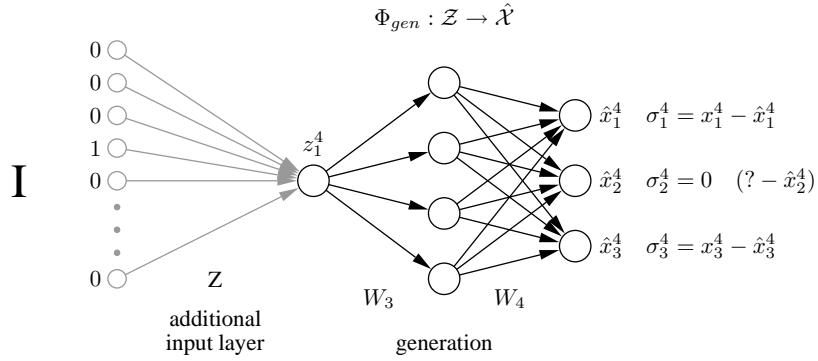


Fig. 2.5. The inverse NLPCA network. Only the second part of the autoassociative network (Figure 2.2) is needed, as illustrated by a 1-4-3 network (black). This generation part represents the inverse mapping Φ_{gen} which generates or reconstructs higher-dimensional samples \mathbf{x} from their lower dimensional component representations \mathbf{z} . These component values \mathbf{z} are now unknown inputs that can be estimated by propagating the partial errors σ back to the input layer \mathbf{z} . This is equivalent to the illustrated prefixed input layer (grey), where the weights are representing the component values \mathbf{z} . The input is then a (sample x sample) identity matrix \mathbf{I} . For the 4th sample ($n=4$), as illustrated, all inputs are zero except the 4th, which is one. On the right, the second element x_2^4 of the 4th sample \mathbf{x}^4 is missing. Therefore, the partial error σ_2^4 is set to zero, identical to ignoring or non-back-propagating. The parameter of the model can thus be estimated even when there is missing data.

function $\Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z}$. The inverse model is given by the second part, the generation function $\Phi_{gen} : \mathcal{Z} \rightarrow \hat{\mathcal{X}}$. Even though a forward model is appropriate for linear PCA, it is less suitable for nonlinear PCA, as it sometimes can be functionally very complex or even intractable due to a one-to-many mapping problem. Two identical samples \mathbf{x} may correspond to distinct component values \mathbf{z} , for example, the point of self-intersection in Figure 2.6B.

By contrast, modelling the inverse mapping $\Phi_{gen} : \mathcal{Z} \rightarrow \hat{\mathcal{X}}$ alone, provides a number of advantages: we directly model the assumed data generation process which is often much easier than modelling the extraction mapping. We also can extend the inverse NLPCA model to be applicable to incomplete data sets, since the data are only used to determine the error of the model output. And, it is more efficient than the entire autoassociative network, since we only have to estimate half of the network weights.

Since the desired components now are unknown inputs, the *blind inverse problem* is to estimate both the inputs and the parameters of the model by only given outputs. In the inverse NLPCA approach, we use one single error function for simultaneously optimising both the model weights \mathbf{w} and the components as inputs \mathbf{z} .

2.5.1 The Inverse Network Model

Inverse NLPCA is given by the mapping function Φ_{gen} , which is represented by a multi-layer perceptron (MLP) as illustrated in Figure 2.5. The output $\hat{\mathbf{x}}$ depends on the input \mathbf{z} and the network weights $w \in W_3, W_4$.

$$\hat{\mathbf{x}} = \Phi_{gen}(\mathbf{w}, \mathbf{z}) = W_4 g(W_3 \mathbf{z}). \quad (2.12)$$

The nonlinear activation function g (e.g., $tanh$) is applied element-wise. Biases are not explicitly considered. They can be included by introducing extra units with activation set to one.

The aim is to find a function Φ_{gen} which generates data $\hat{\mathbf{x}}$ that approximate the observed data \mathbf{x} by a minimal squared error $\|\hat{\mathbf{x}} - \mathbf{x}\|^2$. Hence, we search for a minimal error depending on \mathbf{w} and \mathbf{z} : $\min_{w,z} \|\Phi_{gen}(\mathbf{w}, \mathbf{z}) - \mathbf{x}\|^2$. Both the lower dimensional component representation \mathbf{z} and the model parameters \mathbf{w} are unknown and can be estimated by minimising the reconstruction error:

$$E(\mathbf{w}, \mathbf{z}) = \frac{1}{2} \sum_n^N \sum_i^d \left[\sum_j^h w_{ij} g\left(\sum_i^m w_{jk} z_k^n\right) - x_i^n \right]^2, \quad (2.13)$$

where N is the number of samples and d the dimensionality.

The error can be minimised by using a gradient optimisation algorithm, e.g., *conjugate gradient descent* [31]. The gradients are obtained by propagating the partial errors σ_i^n back to the input layer, meaning one layer more than usual. The gradients of the weights $w_{ij} \in W_4$ and $w_{jk} \in W_3$ are given by the partial derivatives:

$$\frac{\partial E}{\partial w_{ij}} = \sum_n \sigma_i^n g'(a_j^n) \quad ; \quad \sigma_i^n = \hat{x}_i^n - x_i^n, \quad (2.14)$$

$$\frac{\partial E}{\partial w_{jk}} = \sum_n \sigma_j^n z_k^n \quad ; \quad \sigma_j^n = g'(a_j^n) \sum_i w_{ij} \sigma_i^n. \quad (2.15)$$

The partial derivatives of linear input units ($z_k = a_k$) are:

$$\frac{\partial E}{\partial z_k^n} = \sigma_k^n = \sum_j w_{jk} \sigma_j^n. \quad (2.16)$$

For circular input units given by equations (2.6) and (2.7), the partial derivatives of a_p and a_q are:

$$\frac{\partial E}{\partial a_p^n} = (\tilde{\sigma}_p^n z_q^n - \tilde{\sigma}_q^n z_p^n) \frac{z_q^n}{r_n^3} \quad \text{and} \quad \frac{\partial E}{\partial a_q^n} = (\tilde{\sigma}_q^n z_p^n - \tilde{\sigma}_p^n z_q^n) \frac{z_p^n}{r_n^3} \quad (2.17)$$

with $\tilde{\sigma}_p^n$ and $\tilde{\sigma}_q^n$ given by

$$\tilde{\sigma}_p^n = \sum_j w_{jp} \sigma_j^n \quad \text{and} \quad \tilde{\sigma}_q^n = \sum_j w_{jq} \sigma_j^n . \quad (2.18)$$

Biases can be added by using additional weights w_{i0} and w_{j0} and associated constants $z_0 = 1$ and $g(a_0) = 1$.

The weights \mathbf{w} and the inputs \mathbf{z} can be optimised simultaneously by considering (\mathbf{w}, \mathbf{z}) as one vector to optimise by given gradients. This would be equivalent to an approach where an additional input layer is representing the components \mathbf{z} as weights, and new inputs are given by a (sample x sample) identity matrix, as illustrated in Figure 2.5. However, this layer is not needed for implementation. The purpose of the additional input layer is only to explain that the inverse NLPCA model can be converted to a conventionally trained multi-layer perceptron, with known inputs and simultaneously optimised weights, including the weights \mathbf{z} , representing the desired components. Hence, an alternating approach as used in [11] is not necessary. Besides providing a more efficient optimisation, it also avoids the risk of oscillations during training in an alternating approach.

A disadvantage of such an inverse approach is that we have no mapping function $\mathcal{X} \rightarrow \mathcal{Z}$ to map new data \mathbf{x} to the component space. However, we can achieve the mapping by searching for an optimal input \mathbf{z} to a given new sample \mathbf{x} . For that, the network weights \mathbf{w} are set constant while the input \mathbf{z} is estimated by minimising the squared error $\| \hat{\mathbf{x}}(\mathbf{z}) - \mathbf{x} \|^2$. This is only a low dimensional optimisation by given gradients efficiently performed by a gradient optimisation algorithm.

The inverse NLPCA is able to extract components of higher nonlinear complexity than the standard NLPCA, even self-intersecting components can be modelled, as shown in Figure 2.6B. Inverse NLPCA can be used to extract more than one component by increasing the number of units in the input layer.

2.5.2 NLPCA Models Applied to Circular Data

In Figure 2.6, a circular data structure is used to illustrate the behaviour of NLPCA models: the standard autoassociative network (NLPCA), the inverse model with standard units (NLPCA.inv), and the circular PCA (NLPCA.cir). The data are located on a unit circle, disturbed by Gaussian noise with standard deviation 0.1. The standard autoassociative network is not able to describe the circular structure all-around due to the problem to map at least one point on the circle to two distinct component values. This problem does not occur in inverse NLPCA since it is only a mapping from component values to the data. The circular structure is approximated by a component that intersects with itself but it has still an open interval. Thus, the closed curve solution as provided by circular PCA gives a more useful description of the circular structure of the data. Circular PCA can also be used as inverse model to be more efficient and to handle missing data.

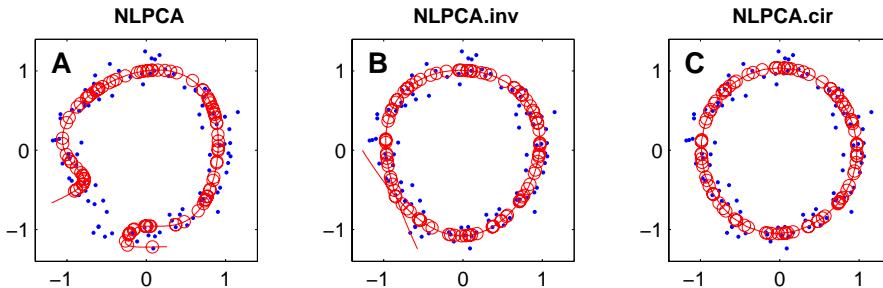


Fig. 2.6. Nonlinear PCA (NLPCA). Shown are results of different variants of NLPCA applied to a two-dimensional artificial data set of a noisy circle. The original data \mathbf{x} (‘.’) are projected onto a nonlinear component (line). The projection or noised-reduced reconstruction $\hat{\mathbf{x}}$ is marked by a circle ‘o’. **(A)** The standard NLPCA cannot describe a circular structure completely. There is always a gap. **(B)** The inverse NLPCA can provide self-intersecting components and hence approximates the circular data structure already quite well. **(C)** The circular PCA is most suitable for a circular data structure, since it is able to approximate the data structure continuously by a closed curve.

2.5.3 Inverse NLPCA for Missing Data

There are many methods for estimating missing values [32]. Some good approaches are based on maximum likelihood in conjunction with an expectation–maximisation (EM) algorithm [33]. To analyse incomplete data, it is common to estimate the missing values first in a separate step. But this can lead to problems caused by distinct assumptions in the missing data estimation step and the subsequent analysis. For example, a *linear* missing data estimation can run counter to a subsequent *nonlinear* analysis. Therefore, our strategy is to adapt the analysis technique to be applicable to incomplete data sets, instead of estimating missing values separately. Even though the aim is to extract nonlinear components directly from incomplete data, once the nonlinear mapping is modelled, the missing values can be estimated as well.

As shown in Figure 2.5, the inverse NLPCA model can be extended to be applicable to incomplete data sets [15]: If the i th element x_i^n of the n th sample vector \mathbf{x}^n is missing, the corresponding partial error σ_i^n is omitted by setting to zero before back-propagating, hence it does not contribute to the gradients. The nonlinear components are extracted by using all available observations. By using these components, the original data can be reconstructed including the missing values. The network output \hat{x}_i^n gives the estimation of the missing value x_i^n .

The same approach can be used to weight each value differently. This might be of interest when for each value an additional probability value p is available. Each partial error σ_i^n can then be weighted $\tilde{\sigma}_i^n = p * \sigma_i^n$ before back-propagating.

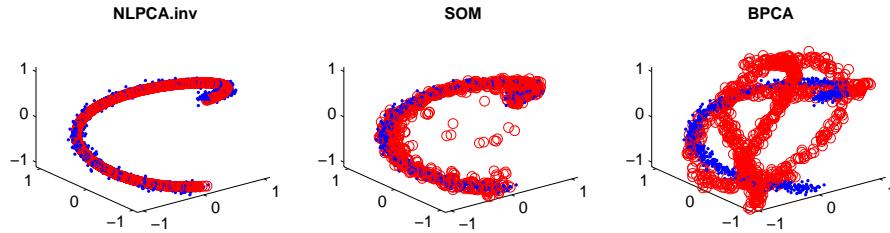


Fig. 2.7. Missing data estimation. Used is an artificial data set which describes a helical loop, plotted as dots ('.'). From each sample, one of the three values is rejected and have to be reconstructed by different missing data estimation algorithms. The reconstructed samples are plotted as circles ('o'). The inverse NLPICA identifies the nonlinear component best and hence gives a very good estimation of the missing values. SOM also gives a reasonably good estimation, but the linear approach BPCA fails on this nonlinear test set, see also Table 2.1.

2.5.4 Missing Data Estimation

Even though an artificial data set does not reflect the whole complexity of real biological data, it is useful to illustrate the problem of missing data estimation in order to give a better understanding of how missing data are handled by different methods.

The inverse NLPICA approach is applied to an artificial data set and the results are compared to results of other missing value estimation techniques. This includes the nonlinear estimation by *self organising maps* (SOM) [21] implemented in the SOM TOOLBOX 2.0³ [34]. Furthermore, we applied a linear PCA-based approach for missing value estimation, an adapted *Bayesian principal component analysis* (BPCA)⁴ [35] based on [36].

The data \mathbf{x} lie on a one-dimensional manifold (a helical loop) embedded in three dimensions, plus Gaussian noise η of standard deviation $\sigma = 0.05$, see Figure 2.7. 1,000 samples \mathbf{x} were generated from a uniformly distributed factor t over the range [-1,1], t represents the angle:

$$\begin{aligned}x_1 &= \sin(\pi t) + \eta, \\x_2 &= \cos(\pi t) + \eta, \\x_3 &= t + \eta.\end{aligned}$$

From each three-dimensional sample, one value is randomly removed and regarded as missing. This generates a high missing value rate of 33.3 percent. However, if the nonlinear component (the helix) is known, the estimation of a missing value is exactly given by the two other coordinates, except at the first and last position of the helix loop, where in the case of missing vertical coordinate x_3 , the sample can be assigned either to the first or to the last

³ <http://www.cis.hut.fi/projects/somtoolbox/>

⁴ <http://hawaii.aist-nara.ac.jp/~shige-o/tools/>

MSE of missing value estimation		
	noise	noise-free
NLPCA.inv	0.0021	0.0013
SOM	0.0405	0.0384
BPCA	0.4191	0.4186
mean	0.4429	0.4422

Table 2.1. Mean square error (MSE) of different missing data estimation techniques applied to the helical data (Figure 2.7). The inverse NLPCA model provides a very good estimation of the missing values. Although the model was trained with noisy data, the noise-free data were better represented than the noisy data, confirming the noise-reducing ability of the model. SOM also gives a good estimation on this nonlinear data, but the linear technique BPCA is only as good as the naive substitution by the mean over the residuals of each variable.

position. There are two valid solutions. Thus, missing value estimation is not always unique in the nonlinear case.

In Figure 2.7 and Table 2.1 it is shown that even if the data sets are incomplete for all samples, the inverse NLPCA model is able to detect the nonlinear component and provides a very accurate missing value estimation. The nonlinear technique SOM also achieves a reasonably good estimation, but the linear approach BPCA is unsuitable for this nonlinear application.

2.6 Applications

The purpose of nonlinear PCA is to identify and to extract nonlinear components from a given data set. The extracted components span a component space which is supposed to cover the most important information of the data. We would like to demonstrate this in examples of NLPCA applications. First, we discuss results of hierarchical NLPCA in order to illustrate the potential curvature of a components subspace. Then we describe two applications of NLPCA to experimental time courses from molecular biology. This includes both a non-periodic and a periodic time course. The periodic one demonstrates the use of circular PCA. In order to handle missing values, a frequent problem in molecular data, NLPCA is applied in the inverse mode. In both experiments nonlinear PCA is able to identify the time factor already with the first nonlinear component thereby confirming that time is the most important factor in the data. Since NLPCA models explicitly the nonlinear mapping between component space and original data space, it provides a model of the biological process which is used here to interpret the impact of individual molecules.

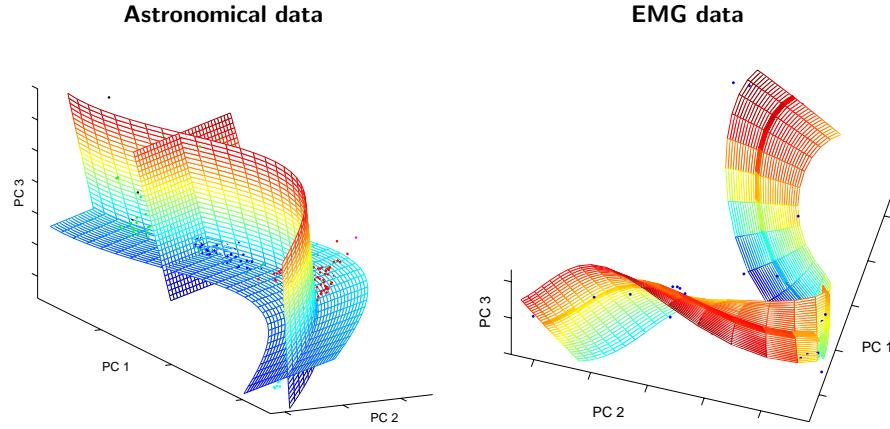


Fig. 2.8. Hierarchical nonlinear PCA is applied to a star spectral data set and to electromyographic (EMG) recordings. Both data sets show a clear nonlinear behaviour. The first three nonlinear components are visualised in the space of the first three PCA components. The grids represent the new coordinate system of the component space. Each grid is spanned by two of the three components while the third is set to zero.

2.6.1 Application of Hierarchical NLPCA

First, we illustrate the performance of hierarchical NLPCA on two separate data sets [10]. The first consists of 19-dimensional spectral information of 487 stars [37]. The second data set is based on electromyographic (EMG) recordings for different muscle activities (labelled as 0, 10, 30, 50 and 70% of maximal personal strength). The one-dimensional EMG signal is then embedded into a d -dimensional space and analysed as a recurrence plot [38]. The final data set then consists of 10 recurrence qualification analysis (RQA) variables for 35 samples, given by the 5 force levels of 7 subjects [39].

The nonlinear components are extracted by minimising the hierarchical error function $E_H = E_1 + E_{1,2} + E_{1,2,3}$. The autoassociative mappings are based on a 19-30-10-30-19 network for the star spectral data and a 10-7-3-7-10 network for the EMG data.

Figure 2.8 shows that both data sets have clear nonlinear characteristics. While in the star data set the nonlinearities seem moderate, this is clearly not the case for the EMG data. Furthermore, in the EMG data, most of the variance is explained by the first two components. The principal curvature given by the first nonlinear component is found to be strongly related to the force level [10]. Since the second component is not related to the force, the force information is supposed to be completely explained by the first component. The second component might be related to another physiological factor.

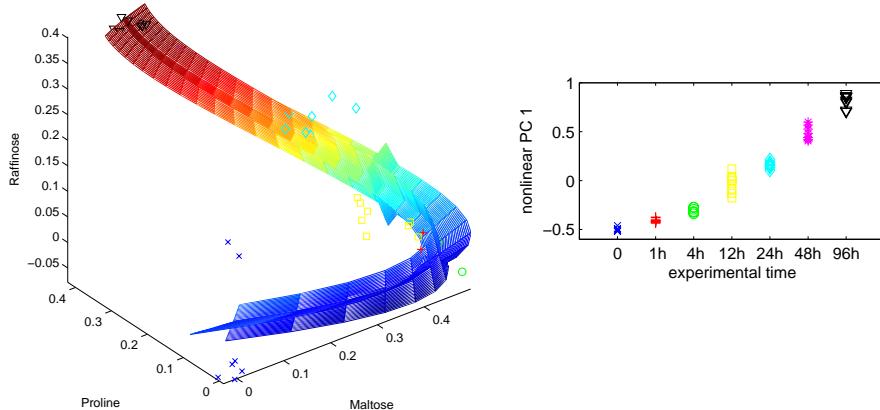


Fig. 2.9. Cold stress metabolite data. Left: The first three extracted nonlinear components are plotted into the data space given by the top three metabolites of highest variance. The grid represents the new coordinate system of the component space. The principal curvature given by the first nonlinear component represents the trajectory over time in the cold stress experiment as shown on the right by plotting the first component against the experimental time.

2.6.2 Metabolite Data Analysis

Cold stress can cause rapid changes of metabolite levels within cells. Analysed is the metabolite response to 4°C cold stress of the model plant *Arabidopsis thaliana* [15, 40]. Metabolite concentrations are measured by *gas chromatography / mass spectrometry (GC/MS)* at 7 time points in the range up to 96 hours. With 7-8 replicas at each time, we have a total number of 52 samples. Each sample provides the concentration levels of 388 metabolites. Precisely, we consider the relative concentrations given by the \log_2 -ratios of absolute concentrations to a non-stress reference. In order to handle missing data, we applied NLPCA in the inverse mode. A neural network of a 3-20-388 architecture was used to extract three nonlinear components in a hierarchical order, shown in Figure 2.9. The extracted first nonlinear component is directly related to the experimental time factor. It shows a strong curvature in the original data space, shown in Figure 2.10. The second and third component are not related to time and the variance of both is much smaller and of similar amount. This suggests that the second and third component represent only the noise of the data. It also confirms our expectations that time is the major factor in the data. NLPCA approximates the time trajectory of the data by use of the first nonlinear component which can therefore be regarded as time component.

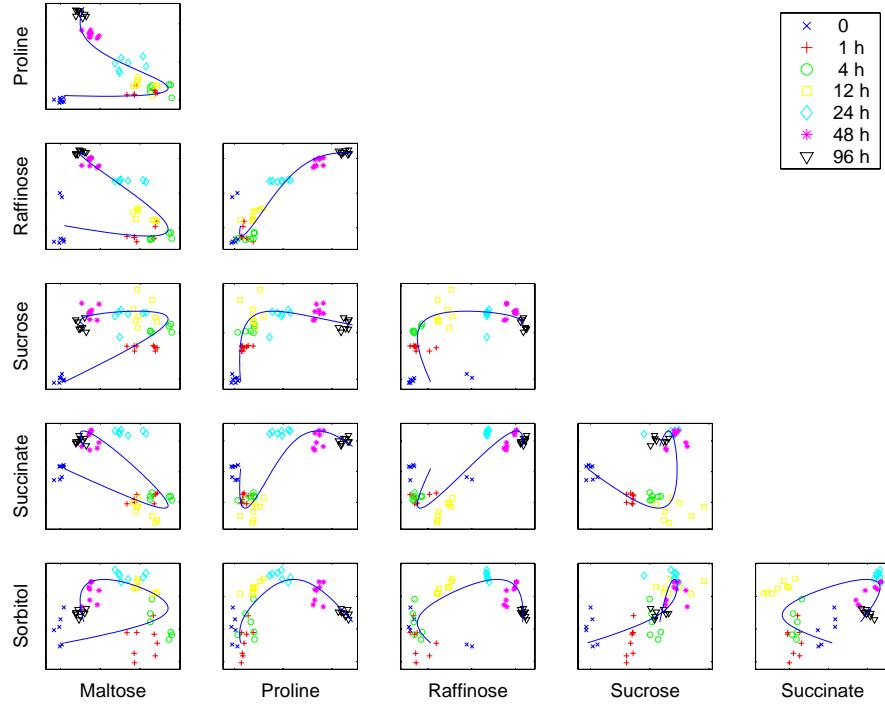


Fig. 2.10. Time trajectory. Scatter plots of pair-wise metabolite combinations of the top six metabolites of highest relative variance. The extracted time component (nonlinear PC 1), marked by a line, shows a strong nonlinear behaviour.

In this analysis, NLPCA provides a model of the cold stress adaptation of *Arabidopsis thaliana*. The inverse NLPCA model gives us a mapping function $\mathcal{R}^1 \rightarrow \mathcal{R}^{388}$ from a time point t to the response \mathbf{x} of all considered 388 metabolites $\mathbf{x} = (x_1, \dots, x_{388})^T$. Thus, we can analyse the approximated response curves of each metabolite, shown in Figure 2.11. The cold stress is reflected in almost all metabolites, however, the response behaviour is quite different. Some metabolites have a very early positive or negative response, e.g., maltose and raffinose, whereas other metabolites only show a moderate increase.

In standard PCA, we can present the variables that are most important to a specific component by a rank order given by the absolute values of the corresponding eigenvector, sometimes termed *loadings* or *weights*. As the components are curves in nonlinear PCA, no global ranking is possible. The rank order is different for different positions on the curved component, meaning that the rank order depends on time. The rank order for a specific time t is given by the values of the tangent vector $\mathbf{v} = \frac{d\mathbf{x}}{dt}$ on the curve at this time.

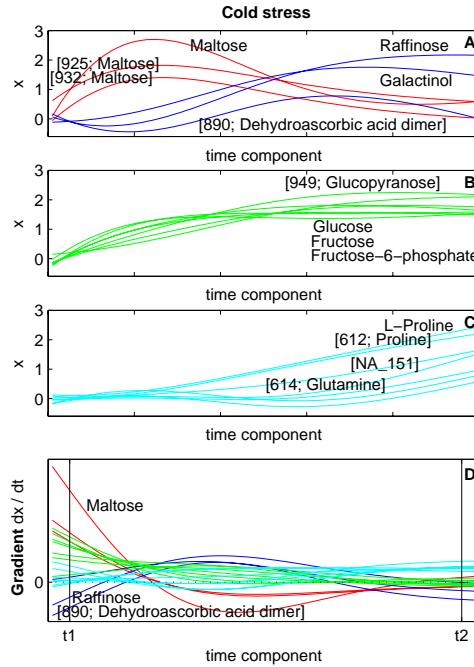


Fig. 2.11. The top three graphs show the different shapes of the approximated metabolite response curves over time. (**A**) Early positive or negative transients, (**B**) increasing metabolite concentrations up to a saturation level, or (**C**) a delayed increase, and still increasing at the last time point. (**D**) The gradient curves represent the influence of the metabolites over time. A high positive or high negative gradient means a strong relative change at the metabolite level. The results show a strong early dynamic, which is quickly moderated, except for some metabolites that are still unstable at the end. The top 20 metabolites of highest gradients are plotted. The metabolite rank order at early time t_1 and late time t_2 is listed in Table 2.2.

To compare different times, we use l_2 -normalised tangents $\tilde{v}_i = v_i / \sqrt{\sum_i |v_i|^2}$ such that $\sum_i (\tilde{v}_i)^2 = 1$. Large absolute values \tilde{v}_i correspond to metabolites of high relative changes on their concentration and hence may be of importance at the considered time. A list of the most important metabolites at an early time point t_1 and a late time point t_2 is given in Table 2.2. The dynamics over time are shown in Figure 2.11D.

2.6.3 Gene Expression Analysis

Many phenomena in biology proceed in a cycle. These include circadian rhythms, the cell cycle, and other regulatory or developmental processes such as the reproductive cycle of the malaria parasite *Plasmodium falciparum* in red blood cells (erythrocytes) which is considered here. Circular PCA is used to analyse this intraerythrocytic developmental cycle (IDC) [9]. The infection and persistence of red blood cells recurs with a periodicity of about 48 hours. The parasite transcriptome is observed by microarrays with a sampling rate of one hour. Two observations, at 23 and 29 hours, are rejected. Thus, the total number of expression profiles is 46, available at <http://malaria.ucsf.edu/> [41, 42]. Each gene is represented by one or more oligonucleotides on this profile. In our analysis, we use the relative expression values of 5,800 oligonucleotides given by the log₂-ratios of individual time hybridisations to a reference pool.

$t_1 \sim 0.5$ hours	$t_2 \sim 96$ hours
\tilde{v} metabolite	\tilde{v} metabolite
0.43 Maltose methoxyamine	0.24 [614; Glutamine]
0.23 [932; Maltose]	-0.20 [890; Dehydroascorbic acid dimer]
0.21 Fructose methoxyamine	0.18 [NA_293]
0.19 [925; Maltose]	0.18 [NA_201]
0.19 Fructose-6-phosphate	0.17 [NA_351]
0.17 Glucose methoxyamine	0.16 [NA_151]
0.17 Glucose-6-phosphate	0.16 L-Arginine
0.16 [674; Glutamine]	0.16 L-Proline
...	...

Table 2.2. Candidate list. The most important metabolites are given for an early time t_1 of about 0.5 hours cold stress and a very late time t_2 of about 96 hours. The metabolites are ranked by the relative change on their concentration levels given by the tangent $\tilde{v}(t)$ at time t on the component curve. As expected, maltose, fructose and glucose show a strong early response to cold stress, however, even after 96 hours there are still some metabolites with significant changes in their levels. Brackets '[...]' denote an unknown metabolite, e.g., [932; Maltose] denotes a metabolite with high mass spectral similarity to maltose.

While a few hundred dimensions of the metabolite data set could still be handled by suitable regularisations in NLPCA, the very high-dimensional data space of 5,800 variables makes it very difficult or even intractable to identify optimal curved components by a given number of only 46 samples. Therefore, the 5,800 variables are linearly reduced to 12 principal components. To handle missing data, this linear PCA transformation was done by a linear neural network with two layers 12-5800 working in inverse mode similar to the nonlinear network in section 2.5. Circular PCA is then applied to the reduced data set of 12 linear components. A network of a 2-5-12 architecture is used with two units in the input layer constrained as circular unit pair (p, q) .

Circular PCA identifies and describes the principal curvature of the cyclic data by a single component, as shown in Figure 2.12. Thus, circular PCA provides a noise-reduced model of the 48 hour time course of the IDC. The nonlinear 2-5-12 network and the linear 12-5800 network together provide a function $\hat{\mathbf{x}} = \Phi_{gen}(\theta)$ which maps any time point, represented by a angular value θ , to the 5,800-dimensional vector $\hat{\mathbf{x}} = (\hat{x}_1, \dots, \hat{x}_{5800})^T$ of corresponding expression values. Again, this model can be used to identify candidate genes at specific times and to interpret the shape of gene expression curves [9].

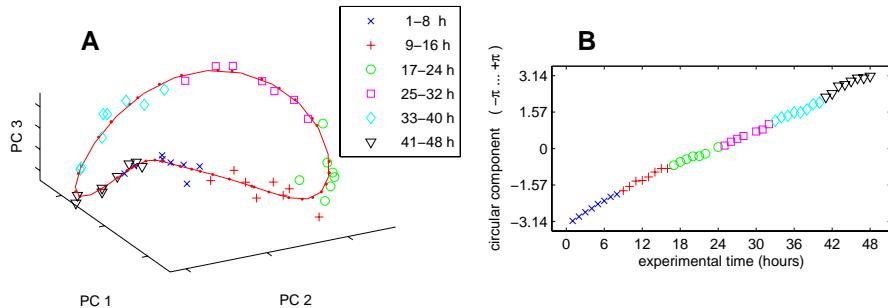


Fig. 2.12. Cyclic gene expression data. **(A)** Circular PCA describes the circular structure of the data by a closed curve – the circular component. The curve represents a one-dimensional component space as a subspace of a 5,800 dimensional data space. Visualised is the component curve in the reduced three dimensional subspace given by the first three components of standard (linear) PCA.
(B) The circular component (corrected by an angular shift) is plotted against the original experimental time. It shows that the main curvature, given by the circular component, explains the trajectory of the IDC over 48 hours.

2.7 Summary

Nonlinear PCA (NLPCA) was described in several variants based on neural networks. This includes the hierarchical, the circular, and the inverse model. While standard NLPCA characterises the desired subspace only as a whole, hierarchical NLPCA enforces to describe this subspace by components arranged in a hierarchical order similar to linear PCA. Hierarchical NLPCA can therefore be seen as a natural nonlinear extension to standard PCA. To describe cyclic or oscillatory phenomena, we need components which describe a closed curve instead of a standard curve with open interval. These circular components can be achieved by circular PCA. In contrast to standard NLPCA which models both the forward component extraction and the inverse data generation, *inverse* NLPCA means to model the inverse mapping alone. Inverse NLPCA is often more efficient and better suited for describing real processes, since it directly models the assumed data generation process. Furthermore, such an inverse model offers the advantage to handle missing data. The idea behind solving the missing data problem was that the criterion of a missing data estimation does not always match the criterion of the subsequent data analysis. Our strategy was therefore to adapt nonlinear PCA to be applicable to incomplete data, instead of estimating the missing values in advance.

Nonlinear PCA was applied to several data sets, in particular to molecular data of experimental time courses. In both applications, the first nonlinear component describes the trajectory over time, thereby confirming our expectations and the quality of the data. Nonlinear PCA provides a noise-reduced model of the investigated biological process. Such computational model can

then be used to interpret the molecular behaviour over time in order to get a better understanding of the biological process.

With the increasing number of time experiments, nonlinear PCA may become more and more important in the field of molecular biology. Furthermore, nonlinearities can also be caused by other continuously observed factors, e.g., a range of temperatures. Even natural phenotypes often take the form of a continuous range [43], where the molecular variation may appear in a nonlinear way.

Availability of Software

A MATLAB® implementation of nonlinear PCA including the hierarchical, the circular, and the inverse model is available at:
<http://www.NLPCA.org/matlab.html>.

Acknowledgement. This work is partially funded by the German Federal Ministry of Education and Research (BMBF) within the programme of *Centres for Innovation Competence* of the BMBF initiative *Entrepreneurial Regions* (Project No. ZIK 011).

References

1. Kramer, M.A.: Nonlinear principal component analysis using auto-associative neural networks. *AIChE Journal*, **37**(2), 233–243 (1991)
2. DeMers, D., Cottrell, G.W.: Nonlinear dimensionality reduction. In: Hanson, D., Cowan, J., Giles, L., eds.: *Advances in Neural Information Processing Systems 5*, San Mateo, CA, Morgan Kaufmann, 580–587 (1993)
3. Hecht-Nielsen, R.: Replicator neural networks for universal optimal source coding. *Science*, **269**, 1860–1863 (1995)
4. Malthouse, E.C.: Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Transactions on Neural Networks*, **9**(1), 165–173 (1998)
5. Kirby, M.J., Miranda, R.: Circular nodes in neural networks. *Neural Computation*, **8**(2), 390–402 (1996)
6. Hsieh, W.W., Wu, A., Shabbar, A.: Nonlinear atmospheric teleconnections. *Geophysical Research Letters*, **33**(7), L07714 (2006)
7. Herman, A.: Nonlinear principal component analysis of the tidal dynamics in a shallow sea. *Geophysical Research Letters*, **34**, L02608 (2007)
8. MacDorman, K., Chalodhorn, R., Asada, M.: Periodic nonlinear principal component neural networks for humanoid motion segmentation, generalization, and generation. In: *Proceedings of the Seventeenth International Conference on Pattern Recognition (ICPR)*, Cambridge, UK, 537–540 (2004)
9. Scholz, M.: Analysing periodic phenomena by circular PCA. In: Hochreiter, M., Wagner, R. (eds.) *Proceedings BIRD conference. LNBI 4414*, Springer-Verlag Berlin Heidelberg, 38–47 (2007)
10. Scholz, M., Vigário, R.: Nonlinear PCA: a new hierarchical approach. In: Verleysen, M., ed.: *Proceedings ESANN*, 439–444 (2002)

11. Hassoun, M.H., Sudjianto, A.: Compression net-free autoencoders. Workshop on Advances in Autoencoder/Autoassociator-Based Computations at the NIPS'97 Conference (1997)
12. Oh, J.H., Seung, H.: Learning generative models with the up-propagation algorithm. In: Jordan, M.I., Kearns, M.J., Solla, S.A., eds.: Advances in Neural Information Processing Systems. Vol. 10., The MIT Press, 605–611 (1998)
13. Lappalainen, H., Honkela, A.: Bayesian nonlinear independent component analysis by multi-layer perceptrons. In: Girolami, M. (ed.) Advances in Independent Component Analysis. Springer-Verlag, 93–121 (2000)
14. Honkela, A., Valpola, H.: Unsupervised variational bayesian learning of nonlinear models. In: Saul, L., Weis, Y., Bottous, L. (eds.) Advances in Neural Information Processing Systems, 17 (NIPS'04), 593–600 (2005)
15. Scholz, M., Kaplan, F., Guy, C., Kopka, J., Selbig, J.: Non-linear PCA: a missing data approach. Bioinformatics, **21**(20), 3887–3895 (2005)
16. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science, **313** (5786), 504–507 (2006)
17. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science, **290** (5500), 2323–2326 (2000)
18. Saul, L.K., Roweis, S.T.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research, **4** (2), 119–155 (2004)
19. Tenenbaum, J., de Silva, V., Langford, J.: A global geometric framework for nonlinear dimensionality reduction. Science, **290** (5500), 2319–2323 (2000)
20. Hastie, T., Stuetzle, W.: Principal curves. Journal of the American Statistical Association, **84**, 502–516 (1989)
21. Kohonen, T.: Self-Organizing Maps. 3rd edn. Springer (2001)
22. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, **10**, 1299–1319 (1998)
23. Mika, S., Schölkopf, B., Smola, A., Müller, K.R., Scholz, M., Rätsch, G.: Kernel PCA and de-noising in feature spaces. In: Kearns, M., Solla, S., Cohn, D., eds.: Advances in Neural Information Processing Systems 11, MIT Press, 536–542 (1999)
24. Harmeling, S., Ziehe, A., Kawanabe, M., Müller, K.R.: Kernel-based nonlinear blind source separation. Neural Computation, **15**, 1089–1124 (2003)
25. Jutten, C., Karhunen, J.: Advances in nonlinear blind source separation. In: Proc. Int. Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003), Nara, Japan, 245–256 (2003)
26. Cichocki, A., Amari, S.: Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications. Wiley, New York (2003)
27. Scholz, M.: Approaches to analyse and interpret biological profile data. PhD thesis, University of Potsdam, Germany (2006) URN: urn:nbn:de:kobv:517-opus-7839, URL: <http://opus.kobv.de/ubp/volltexte/2006/783/>.
28. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
29. Bourlard, H., Kamp, Y.: Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics, **59** (4-5), 291–294, (1988)
30. Scholz, M.: Nonlinear PCA based on neural networks. Master's thesis, Dep. of Computer Science, Humboldt-University Berlin (2002) (in German)

31. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, **49**(6), 409–436 (1952)
32. Little, R.J.A., Rubin, D.B.: *Statistical Analysis with Missing Data*. 2nd edn. John Wiley & Sons, New York (2002)
33. Ghahramani, Z., Jordan, M.: Learning from incomplete data. Technical Report AIM-1509 (1994)
34. Vesanto, J.: Neural network tool for data mining: SOM toolbox. In: *Proceedings of Symposium on Tool Environments and Development Methods for Intelligent Systems (TOOLMET2000)*, Oulu, Finland, Oulun yliopiston paino, 184–196 (2000)
35. Oba, S., Sato, M., Takemasa, I., Monden, M., Matsubara, K., Ishii, S.: A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, **19**(16), 2088–2096 (2003)
36. Bishop, C.: Variational principal components. In: *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, 509–514 (1999)
37. Stock, J., Stock, M.: Quantitative stellar spectral classification. *Revista Mexicana de Astronomia y Astrofisica*, **34**, 143–156 (1999)
38. Webber Jr., C., Zbilut, J.: Dynamical assessment of physiological systems and states using recurrence plot strategies. *Journal of Applied Physiology*, **76**, 965–973 (1994)
39. Mewett, D.T., Reynolds, K.J., Nazeran, H.: Principal components of recurrence quantification analysis of EMG. In: *Proceedings of the 23rd Annual IEEE/EMBS Conference*, Istanbul, Turkey (2001)
40. Kaplan, F., Kopka, J., Haskell, D., Zhao, W., Schiller, K., Gatzke, N., Sung, D., Guy, C.: Exploring the temperature-stress metabolome of *Arabidopsis*. *Plant Physiology*, **136**(4), 4159–4168 (2004)
41. Bozdech, Z., Llinas, M., Pulliam, B., Wong, E., Zhu, J., DeRisi, J.: The transcriptome of the intraerythrocytic developmental cycle of *Plasmodium falciparum*. *PLoS Biology*, **1** (1), E5 (2003)
42. Kissinger, J., Brunk, B., Crabtree, J., Fraunholz, M., Gajria, et al., B.: The plasmodium genome database. *Nature*, **419** (6906), 490–492 (2002)
43. Fridman, E., Carrari, F., Liu, Y.S., Fernie, A., Zamir, D.: Zooming in on a quantitative trait for tomato yield using interspecific introgressions. *Science*, **305** (5691), 1786–1789 (2004)

Learning Nonlinear Principal Manifolds by Self-Organising Maps

Hujun Yin

The University of Manchester, Manchester, M60 1QD, UK,
hujun.yin@manchester.ac.uk

Summary. This chapter provides an overview on the self-organised map (SOM) in the context of manifold mapping. It first reviews the background of the SOM and issues on its cost function and topology measures. Then its variant, the visualisation induced SOM (ViSOM) proposed for preserving local metric on the map, is introduced and reviewed for data visualisation. The relationships among the SOM, ViSOM, multidimensional scaling, and principal curves are analysed and discussed. Both the SOM and ViSOM produce a scaling and dimension-reduction mapping or manifold of the input space. The SOM is shown to be a qualitative scaling method, while the ViSOM is a metric scaling and approximates a discrete principal curve/surface. Examples and applications of extracting data manifolds using SOM-based techniques are presented.

Key words: Self-organising maps, principal curve and surface, data visualisation, topographic mapping

3.1 Introduction

For many years, artificial neural networks have been studied and used to construct information processing systems based on or inspired by natural biological neural structures. They not only provide solutions with improved performance when compared with traditional problem-solving methods, but also give a deeper understanding of human cognitive abilities. Among the various existing neural network architectures and learning algorithms, Kohonen's self-organising map (SOM) [35] is one of most popular neural network models. Developed for an associative memory model, it is an unsupervised learning algorithm with simple structures and computational forms, and is motivated by the retina-cortex mapping. Self-organisation in general is a fundamental pattern recognition process, in which intrinsic inter- and intra-pattern relationships within the data set are learnt without the presence of a potentially biased or subjective external influence. The SOM can provide topologically

preserved mapping from input to output spaces. Although the computational form and structure of the SOM are very simple, numerous researchers have already examined the algorithm and many of its properties, there are still many aspects to be exploited.

In this chapter, we review the background, theories and statistical properties and present recent advances of the SOM. The SOM is an optimal for vector quantisation. Its topographical ordering provides the mapping with enhanced fault and noise tolerant abilities. It also extracts a latent structure of the input space, which is applicable to many applications such as dimensionality reduction, data visualisation, clustering and classification. Various extensions of the SOM have been devised since to extend the mapping as optimal solutions for a wide range of applications. In particular, the SOM has been linked with the principal curve and surface [20] and the recently proposed visualisation induced SOM (ViSOM) [78] has been shown to represent a discrete principal curve/surface [79]. Such an analogy is explored and demonstrated and the advantages and shortly comings examined in the context of other methods such as kernel PCA [66], local linear embedding (LLE) [63] and Isomap [69]. Several examples are presented to highlight the potential of this biologically inspired model in nonlinear, principled data analysis.

3.2 Biological Background

Kohonen's self-organising map (SOM) is an abstract mathematical model of topographic mapping from the (visual) sensory to the cerebral cortex. Modelling and analysing the mapping are important to understanding how the brain perceives, encodes, recognises, and processes the patterns it receives and thus, if somewhat indirectly, is beneficial to machine-based pattern recognition. This section looks into the relevant biological models, from two fundamental phenomena involved, lateral inhibition and Hebbian learning, to Willshaw and von der Malsburg's self-organisation retinotopic model, and then to subsequent Kohonen's simplified and abstracted SOM model. Basic operations and the algorithm of the SOM as well as methods for choosing model parameters are also given.

3.2.1 Lateral Inhibition and Hebbian Learning

Human visual perception and brain make up the most complex cognition system and the most complex of all biological organs. Visual information is processed in both retina and brain, but it is widely believed and verified that most processing is done in the retina, such as extracting lines, angles, curves, contrasts, colours, and motions. The retina then encodes the information and sends through optic nerves and optic chiasma, where some left and right nerves are crossed, to the brain cortex at left or right hemispheres. The retina is a complex neural network. Human retina has over 100 million photosensitive

cells (combining rods and cones) processing in parallel the raw images and codes and renders to just over one million optic nerves to be transmitted to the brain cortex.

The *Perceptron* models some cells in the retina, especially the bipolar and ganglion cells. These cells take inputs from the outputs of cells in the previous layer. To put many units together and connect them into layers, one may hope the resulting network, the *multi-layer perceptron*, will have some functionality similar to the retina (despite some horizontally interconnections are ignored). And indeed such a structure has been demonstrated of capable of certain cognitive and information processing tasks.

Cells in neural networks (either in retina or brain) also connect and interact horizontally. The experiment on limulus by Haldan K. Hartline (1967 Nobel Prize Laureate) and his colleagues in 1960s, has confirmed such a processing on limulus retina. They revealed the so-called *lateral inhibition* activities among the retina cells. That is, there exist both short-range excitatory interaction between close cells and long-range inhibitory interaction between long range cells. This consequently explains the so-called “Mach band” phenomenon on the edges or sharp changes of light intensity. Lateral inhibition tells us that neurons in retina do not just feed the information to upper levels, but also perform an important visual processing task: edge detection and enhancement.

Neural networks present completely different approaches to computing and machine intelligence from traditional symbolic AI. The goal is to emulate the way that natural systems, especially brains, perform on various cognitive tasks. When a network of simple processing units interconnect to each other, there are potentially a massive number of synaptic weights available to be configured and modified such that the network will suit a particular task. This configuration and modification process is carried out by a learning procedure, i.e. *learning or training* algorithm. Traditional pattern recognition approaches usually require solving some well-defined functions or models, such as feature extraction, transformation, and discriminant analysis by a series of processing steps. Neural networks can simply learn from examples. Presented repeatedly with known examples of raw patterns and with an appropriate learning or training algorithm, they are able to extract by themselves the most intrinsic nature of the patterns and are able to perform recognition tasks. They will also have ability to carry out similar recognition tasks not only on trained examples but also on unseen patterns. Learning methods and algorithms, undoubtedly play an important role in building successful neural networks.

Although many learning methods have been proposed, there are two fundamental kinds of learning paradigms: *supervised learning* and *unsupervised learning*. The former is commonly used in most feed-forward neural networks, in which the input-output (or input-target) functions or relationships are built from a set of examples. While the latter resembles a *self-organisation* process in the cortex and seeks inter-relationships and associations among the input.

The most representing supervised learning rule is the *error-correction learning*. When presented an input-output pair, learning takes place when

the error exists between a desired response or target output and the actual output of the network. This learning rule applies an adjustment, proportional to this error, to the weights of the neuron concerned. That is, *learning from errors*. Derivation of such a rule can be often traced back to minimising the mean-square-error function. More details can be found in [21]. A derivative of supervised learning is so-called *reinforcement learning*, which is based trail and error (and reward) [68] and has backings from psychology.

Self-organisation often involves both *competition* and *correlative learning*. When presented with a stimulus, neurons compete among themselves for the possession or ownership of this input. The winners then strengthen their weights or their relationships with this input. *Hebbian learning* is the most common rule for *unsupervised or self-organised learning*. The original Hebb's statement from his book, *The Organization of Behaviour* [22], was “*When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.*”

Mathematically Hebbian learning rule can be directly interpreted as,

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha x_i(t)y_j(t), \quad (3.1)$$

where α is a positive learning rate, $0 < \alpha < 1$, and x and y are the input and output of the neural system respectively, or can also be regarded as the outputs of two neurons. That is, the change of the synaptic weight is proportional to the correlation between an input and its associated output. If the input and output coherent, the weight connecting them is strengthened (xy is positive), otherwise, weakened (xy is either negative or zero).

The Hebbian learning requires some modification before it can be used in practice, otherwise the weight will easily become saturated or unlimited. One solution is to add a forgetting term to prevent weights from increasing/decreasing monotonically as in the SOM (see the next subsection). Alternative is to normalise the weights. For instance, Oja [54] proposed a weight normalisation scheme on all weights. This introduces naturally a forgetting term to the Hebbian rule,

$$\begin{aligned} w_i(t+1) &= \frac{w_i(t) + \alpha x_i(t)y(t)}{\left\{ \sum_{j=1}^n [w_j(t) + \alpha x_j(t)y(t)]^2 \right\}^{1/2}} \\ &\approx w_i(t) + \alpha y(t)[x_i(t) - y(t)w_i(t)] + O(\alpha^2), \end{aligned} \quad (3.2)$$

where $O(\alpha^2)$ represents second- and high-order terms in α , and can be ignored when a small learning rate is used.

The resulting Oja's learning algorithm is a so-called *principal component network*, which learns to extract the most variant directions among the data set. Other variants of Hebbian learning include many algorithms used for *Independent Component Analysis* [55, 26].

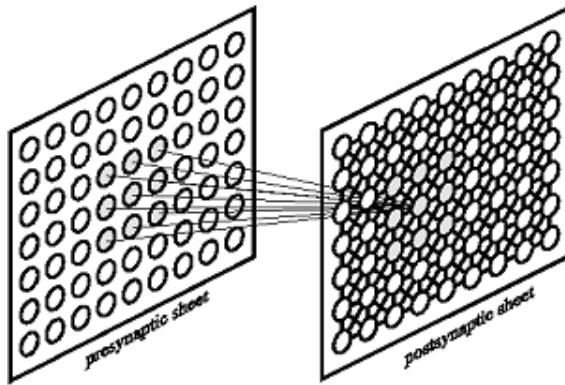


Fig. 3.1. Von der Malsburg's self-organising map model. Local clusters in a presynaptic sheet are connected to local clusters in a postsynaptic sheet. There are lateral interconnections within the postsynaptic sheet (solid lines are used to indicate such connections).

3.2.2 From Von Marsburg and Willshaw's Model to Kohonen's SOM

Stimuli from the outside world are received by various sensory or receptive fields (e.g. visual-, auditory-, motor-, or somato-sensory), coded or abstracted by the living neural networks, and projected through axons onto the cerebral cortex, often to distinct parts of cortex. In other words, the different areas of the cortex (cortical maps) correspond to different sensory inputs. Topographically ordered maps have been widely observed in the cortex. The main structures (primary sensory areas) of the cortical maps are established before birth (cited in [76, 36]), in a predetermined topographically ordered fashion. Other more detailed areas (associative areas), however, are developed through self-organisation gradually during life and in a topographically meaningful order. Therefore studying such topographically ordered projections, which had been ignored during the early period of neural information processing development [37], is clearly important for forming dimension-reduction mapping and for the effective representation of sensory information and feature extraction.

The self-organised learning behaviour of brains has been studied for a long time by many people. Many pioneering works include [2, 7, 17, 22, 35, 52, 74, 75, 76]. von der Malsburg and Willshaw [74, 76] first developed, in mathematical form, self-organising topographical mappings, mainly from two-dimensional presynaptic sheets to two-dimensional postsynaptic sheets, based on retinotopic mapping: the ordered projection of visual retina to visual cortex (see Fig. 3.1).

The model uses short-range excitatory connections between cells so that activity in neighbouring cells becomes mutually reinforced, and uses long-range inhibitory interconnections to prevent activity from spreading too far.

The postsynaptic activities $\{y_j(t), j=1, 2, \dots, N_y\}$, at time t , are expressed by

$$\frac{\partial y_i(t)}{\partial t} + cy_i(t) = \sum_j w_{ij}(t)x_i(t) + \sum_k e_{ik}y_k^*(t) - \sum_{k'} b_{ik'}y_{k'}^*(t), \quad (3.3)$$

where c is the membrane constant, $w_{ij}(t)$ is the synaptic strength between cell i and cell j in pre- and post-synaptic sheets respectively; $\{x_i(t), i=1, 2, \dots, N_x\}$, the state of the presynaptic cells, equal to 1 if cell i is active or 0 otherwise; e_{kj} and b_{kj} are short-range excitation and long-range inhibition constants respectively; and $y_j^*(t)$ is an active cell in postsynaptic sheet at time t . The postsynaptic cells fire if their activity is above a threshold, say,

$$y_j^*(t) = \begin{cases} y_j(t) - \theta, & \text{if } y_j(t) > \theta; \\ 0, & \text{otherwise.} \end{cases} \quad (3.4)$$

The modifiable synaptic weights between pre- and post-synaptic sheets are then facilitated in proportion to the product of activities in the appropriate pre- and postsynaptic cells (direct realisation of Hebbian learning):

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha x_i(t)y_j^*(t), \text{ subject to } \frac{1}{N_x} \sum_i w_{ij} = \text{constant}, \quad (3.5)$$

where α is a small constant representing the learning rate. To prevent the synaptic strengths becoming unstable, the total strength associated with each postsynaptic cell is limited by normalisation to a constant value after each iteration.

Kohonen [35] abstracted the above self-organising learning principles and proposed a much simplified learning mechanism which cleverly incorporates the Hebb's learning rule and lateral interconnection rules and can emulate the self-organising learning effect. Although the resulting SOM algorithm was more or less proposed in a heuristic manner [40], it is a simplified and generalised model of the above self-organisation process.

In Kohonen's model, the postsynaptic activities are similar to Eq. (3.3). To find the solutions of this equation and ensure they are non-negative properties, a sigmoid type of nonlinear function is applied to each postsynaptic activity:

$$y_j(t+1) = \varphi \left(\mathbf{w}_j^T \mathbf{x}(t) + \sum_i h_{ij}y_i(t) \right), \quad (3.6)$$

where h_{kj} is similar to e_{kj} and b_{kj} , the input is described as a vector as the map can be extended to any dimensional input. A typical mapping is shown in Fig. 3.2.

A spatially-bounded cluster or *bubble* will then be formed among the postsynaptic activities and will stabilise at a maximum (without loss of generality which is assumed to be unity) when within the bubble, or a minimum (i.e. zero) otherwise,

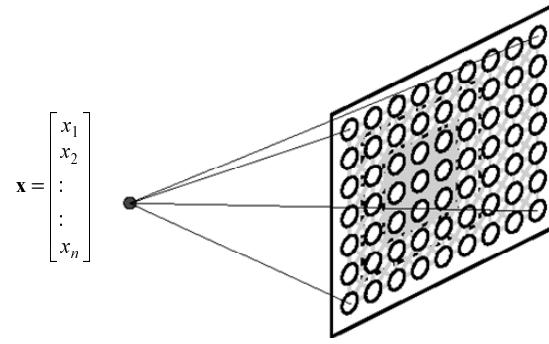


Fig. 3.2. Kohonen’s self-organising map model. The input is connected to every cell in the postsynaptic sheet (the map). The learning makes the map localised, i.e. different local fields will respond to different ranges of inputs. The lateral excitation and inhibition connections are emulated by a mathematical modification, i.e. local sharing, to the learning mechanism. (So there are no actual connections between cells, or in a sense we can say the connections are virtual. Hence grey lines are used to indicate these virtual connections.)

$$y_j(t+1) = \begin{cases} 1, & \text{if neuron } j \text{ is inside the bubble} \\ 0, & \text{otherwise.} \end{cases} \quad (3.7)$$

The bubble is centred on a postsynaptic cell whose synaptic connection with the presynaptic cells is mostly matched with the input or presynaptic state, i.e. the first term in the function in Eq. (3.6) is the highest. The range or size, denoted by $\eta(t)$, of the bubble depends on the ratio of the lateral excitation and inhibition. To modify the Hebbian learning rule, i.e. Eq. (3.5), instead of using normalisation, a forgetting term, $\beta y_j(t)w_{ij}(t)$, is added. Let $\alpha = \beta$, and apply the function (3.7), the synaptic learning rule can then be formulated as

$$\begin{aligned} \frac{\partial w_{ij}(t)}{\partial t} &= \alpha y_j(t)x_i(t) - \beta y_j(t)w_{ij}(t) = \alpha[x_i(t) - w_{ij}(t)]y_j(t) \\ &= \begin{cases} \alpha[x_i(t) - w_{ij}(t)], & \text{if } j \in \eta(t); \\ 0, & \text{if } j \notin \eta(t). \end{cases} \end{aligned} \quad (3.8)$$

At each time step the best matching postsynaptic cell is chosen according to the first term of the function in Eq. (3.6), which is the inner product, or correlation, of the presynaptic input and synaptic weight vectors. When normalisation is applied to the postsynaptic vectors, as it usually is, this matching criterion is similar to the Euclidean distance measure between the weight and input vectors. Therefore the model provides a very simple computational form. The lateral interconnection between neighbouring neurons

and the “Mexican-hat” excitatory or inhibitory rules are simulated (mathematically) by a simple local neighbourhood excitation centred on the winner. Thus the neuron’s lateral interconnections (both excitatory and inhibitory) have been replaced by neighbourhood function adjustment. The neighbourhood function’s width can simulate the control of the exciting and inhibiting scalars. The constrained (with a decaying or forgetting term) Hebbian learning rule has been simplified and becomes a competitive learning model. Most of Kohonen’s work has been in associative memories [32, 33, 34, 35, 36, 37]. In his studies, he has found that the spatially ordered representation of sensory information in the brain is highly related to the memory mechanism, and that the inter-representation and information storage can be implemented simultaneously by an adaptive, massively parallel, and self-organising network [37]. This simulated cortex map, on the one hand can perform a self-organised search for important features among the inputs, and on the other hand can arrange these features in a topographically meaningful order.

3.2.3 The SOM Algorithm

The SOM uses a set of neurons, often arranged in a 2D rectangular or hexagonal grid, to form a discrete topological mapping of an input space, $\mathbf{X} \in \Re^n$. At the start of the learning, all the weights $\{\mathbf{w}_{r1}, \mathbf{w}_{r2}, \dots, \mathbf{w}_{rm}\}$ are initialised to small random numbers. \mathbf{w}_{ri} is the weight vector associated to neuron i and is a vector of the same dimension, n , of the input. m is the total number of neurons. \mathbf{r}_i is the location vector of neuron i on the grid. Then the algorithm repeats the following steps.

- At each time t , present an input, $\mathbf{x}(t)$, select the winner,

$$v(t) = \arg \min_{k \in \Omega} \|\mathbf{x}(t) - \mathbf{w}_k(t)\|. \quad (3.9)$$

- Updating the weights of winner and its neighbours,

$$\Delta \mathbf{w}_k(t) = \alpha(t) \eta(v, k, t) [\mathbf{x}(t) - \mathbf{w}_v(t)]. \quad (3.10)$$

- Repeat until the map converges,

where $\eta(v, k, t)$ is the neighbourhood function and Ω is the set of neuron indexes. Although one can use the original stepped or top-hat type of neighbourhood function (is one when the neuron is within the neighbourhood or zero otherwise), a Gaussian form is often used in practice, i.e. $\eta(v, k, t) = \exp[-\frac{\|v-k\|^2}{2\sigma(t)^2}]$, with σ representing the changing effective range of the neighbourhood.

The coefficients $\{\alpha(t), t \geq 0\}$, termed *adaptation gain*, or *learning rate*, are *scalar-valued, decrease monotonically*, and satisfy [36],

$$(i) \ 0 < \alpha(t) < 1; (ii) \ \lim_{t \rightarrow \infty} \sum \alpha(t) \rightarrow \infty; (iii) \ \lim_{t \rightarrow \infty} \sum \alpha^2(t) < \infty. \quad (3.11)$$

They are the same as to those used in stochastic approximation [62]. The third condition in (11) has been relaxed [60] to a less restrictive one, namely, $\lim_{t \rightarrow \infty} \alpha(t) \rightarrow 0$.

If the inner product similarity measure is adopted as the best matching rule, i.e.

$$v(t) = \arg \min_{k \in \Omega} [\mathbf{w}_k^T(t) \mathbf{x}(t)], \quad (3.12)$$

then the corresponding weight updating should become [39]

$$\mathbf{w}_k(t+1) = \begin{cases} \frac{\mathbf{w}_k(t) + \alpha(t)\mathbf{x}(t)}{\|\mathbf{w}_k(t) + \alpha(t)\mathbf{x}(t)\|}; \\ \mathbf{w}_k(t). \end{cases} \quad (3.13)$$

Such a form is often used in text/document mining applications (e.g. [16]).

3.3 Theories

3.3.1 Convergence and Cost Functions

Although the SOM algorithm has a simple computational form, a formal analysis of it and the associated learning processes and mathematical properties is not easily obtainable. Some important issues still remain unanswered. Self-organisation, or more specifically the ordering process, has been studied in some depth; however a universal conclusion has been difficult to obtain, if not impossible. This section reviews the statistical and convergence properties of the SOM and associated cost functions, the issue that still causes confusions to many even today. Various topology preservation measures will be analysed and explained.

The SOM was proposed to model the sensory to cortex mapping thus the unsupervised associated memory mechanism. Such a mechanism is also related to vector quantisation or vector quantiser (VQ) [46] in coding terms. The SOM has been shown to be an asymptotically optimal VQ [82]. More importantly, with the neighbourhood learning, the SOM is an error tolerant VQ and Bayesian VQ [48, 49, 50].

Convergence and ordering has only formally been proved in trivial one dimensional case. A full proof of both convergence and ordering in multidimensional are still outstanding, though there have been several attempts (e.g. [13, 14, 45, 47, 60, 82]. Especially Erwin, Obermayer and Schulten [13, 14] showed that there was no cost function that the SOM would follow *exactly*. Such an issue is also linked to the claimed lack of an exact cost function that the algorithm is following. Recent work by various researchers has already shed light on this intriguing issue surrounding the SOM. Yin and Allinson [82] extended the Central Limit Theorem and used it to show that when

the neighbourhood is reducing to just winner as in the original SOM, the weight vectors (code references) are asymptotically Gaussian distributed and will converge in mean square sense to the means of the Voronoi cells, i.e. an optimal VQ (with the SOM's nearest distance winning rule),

$$\mathbf{w}_k \rightarrow \frac{1}{P(X_k)} \int_{V_k} \mathbf{x} p(\mathbf{x}) d\mathbf{x}, \quad (3.14)$$

where V_k is the Voronoi cell (the data region) that weight vector \mathbf{w}_k is responsible, and $p(\mathbf{x})$ is the probability density function of the data. In general cases with the effect of the neighbourhood function, the weight vector is a kernel smoothed mean [79],

$$\mathbf{w}_k \rightarrow \frac{\sum_{t=1}^T \eta(v, k, t) \mathbf{x}(t)}{\sum_{t=1}^T \eta(v, k, t)}. \quad (3.15)$$

Yin and Allinson [82] have also proved that the initial state has diminishing effect on the final weights when the learning parameters follow the convergence conditions. Such an effect has been recently verified by de Bolt, Cottrell and Verleysen [10] using Monte-Carlo bootstrap cross validation. The ordering was not considered. (In practice, good initialisation can be used to guide a faster or even better convergence, due to the limited training time and samples, as well as much relaxed learning rates. For example, initialising the map to a principal linear submanifold can reduce the ordering time, if the ordering process is not a key requirement.)

Luttrell [48, 49] first related hierarchical noise tolerant coding theory to the SOM. When the transmission channel noise is considered, a two-stage optimisation has to be done not only to minimise the representation distortion (as in the VQ) but also to minimise the distortion caused by the channel noise. He revealed that the SOM can be interpreted as such a coding algorithm. The neighbourhood function acts as the model for the channel noise distribution and should not go to zero as in the original SOM. Such a noise tolerant VQ has the following objective function [48, 49],

$$D_2 = \int d\mathbf{x} p(\mathbf{x}) \int d\mathbf{n} \pi(\mathbf{n}) \|\mathbf{x} - \mathbf{w}_k\|^2, \quad (3.16)$$

where \mathbf{n} is the noise variable and $\pi(\mathbf{n})$ is the noise distribution. Durbin and Mitchison [12] and Mitchison [53] have also linked the SOM and this noise tolerant VQ with minimal wiring of cortex like maps.

When the code book (the map) is finite, the noise can be considered as discrete, then the cost function can be re-expressed as,

$$D_2 = \sum_i \int_{V_i} \sum_k \pi(i, k) \|\mathbf{x} - \mathbf{w}_k\|^2 p(\mathbf{x}) d\mathbf{x}, \quad (3.17)$$

where V_i is the Voronoi region of cell i . When the channel noise distribution is replaced by a neighbourhood function (analogous to intersymbol dispersion), this gives to the cost function of the SOM algorithm. The neighbourhood function can be interpreted as channel noise model. Such a cost function has been discussed in the SOM community (e.g. [38, 42, 82, 58, 23]). The cost function is therefore [23],

$$E(\mathbf{w}_1, \dots, \mathbf{w}_N) = \sum_i \int_{V_i} \sum_k \eta(i, k) \|\mathbf{x} - \mathbf{w}_k\|^2 p(\mathbf{x}) d\mathbf{x}. \quad (3.18)$$

It leads naturally to the SOM update algorithm using the sample or stochastic gradient descent method [62]. That is, for each Voronoi region, the sub cost function is,

$$E_i(\mathbf{w}_1, \dots, \mathbf{w}_N) = \int_{V_i} \sum_k \eta(i, k) \|\mathbf{x} - \mathbf{w}_k\|^2 p(\mathbf{x}) d\mathbf{x}. \quad (3.19)$$

The optimisation for all weights $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N\}$ can be sought using the sample gradients. The sample gradient for \mathbf{w}_j is,

$$\frac{\partial \widehat{E}_i(\mathbf{w}_1, \dots, \mathbf{w}_N)}{\partial \mathbf{w}_j} = \frac{\partial \sum_k \eta(i, k) \|\mathbf{x} - \mathbf{w}_k\|^2}{\partial \mathbf{w}_j} = 2\eta(i, j)(\mathbf{x} - \mathbf{w}_j), \quad (3.20)$$

which leads to the SOM updating rule, Eq. (3.10). Note, although the neighbourhood function $\eta_{i,k}$ is inexplicitly related to \mathbf{w}_j , it does not contribute to the weight optimisation, nor does the weight optimisation lead to its adaptation (neighbourhood adaptation is often controlled by a pre-specified scheme, unrelated to the weight adaptation); thus the neighbourhood can be omitted from taking partial differentiation. This is the point that has caused problems in interpreting the cost function of the SOM in the past.

It has however been argued that this energy function is violated at boundaries of Voronoi cells where input has exactly the same smallest distance to two neighbouring neurons. Thus this energy function holds mainly for the discrete case where the probability of such boundary input points is close to zero or the local (sample) cost function \widehat{E}_i should be used in deciding the winner [23]. When spatial-invariant neighbourhood function is used as it is often the case, assigning the boundary input to either cells will lead to the same local sample cost (or error), therefore any input data on the boundary can be assigned to either Voronoi cells that have the same smallest distance to it just like as in the ordinary manner (e.g. using the first-come-first-served fashion in the programming). Only when the neurons lie on the borders of the map, such violation occurs as unbalanced neighbourhood of the neurons. The result is a slightly more contraction towards to the centre or inside of the map for the border neurons compared to the common SOM algorithm as shown in [38]. Using either the simple distance or local distortion measure as the winning rule will result in border neurons be contracted towards inside the map,

esp. when the map is not fully converged or when the effective range of the neighbourhood function is large. With the local distortion rule, this boundary effect is heavier as greater local error is incurred for the border neurons due to its few neighbouring neurons than any inside neurons.

To exactly follow the cost function, the winning rule should be modified to follow the local sample cost function \widehat{E}_i (or the local distortion measure) instead of the simplest nearest distance. That is,

$$v = \arg \min_i \sum_k \eta(i, k) \|\mathbf{x} - \mathbf{w}_k\|^2. \quad (3.21)$$

When the neighbourhood function is symmetric as it is often the case and when the data density function is smooth, this local distortion winning rule is the same as to the simplest nearest distance rule for most non-boundary nodes, especially as the number of nodes is large. On the borders of the map, however, the differences exist due to the unbalance of the nodes presented in the neighbourhoods. Such differences become negligible to the majority of the neurons especially when a large map is used and when the neighbourhood function shrinks to its minimum scale.

3.3.2 Topological ordering measures

The ordering to a large extent is still an outstanding and subtle issue, largely due to the fact that there is no clear (or agreed) definition of order [18]. This is the very reason that why a full self-organisation convergence theorem including both the statistical convergence and ordering and the exact cost function are still subject to debate, the fact that has prompted many alternatives such as [67, 19, 5]. The ordering and an ordered map are clearly defined only in 1-D trivial case. Extending to higher dimension proves to be difficult if not impossible. Bauer and Pawelzik [4] have proposed a measure termed topology product to measure the topological ordering of the map,

$$P = \frac{1}{N^2 - N} \sum_i \sum_j \log \left(\prod_{l=1}^j \frac{d^D(\mathbf{w}_i, \mathbf{w}_{\eta^O(l, i)})}{d^D(\mathbf{w}_i, \mathbf{w}_{\eta^D(l, i)})} \frac{d^O(i, \eta^O(l, i))}{d^O(i, \eta^D(l, i))} \right)^{\frac{1}{2k}}, \quad (3.22)$$

where d^D and d^O represent the distance measures in the input or data space and output or map space respectively; $\eta(l, i)$ represents the l -th neighbour of node i in either data (D) or map (O) space.

The first ratio in the product measures the ratio or match of weight distance sequences of a neighbourhood (upto j) on the map and in the data space. The second ratio is the index distance sequences of the neighbourhood on the map and in the data space. The topographic product measures the product of the two ratios of all possible neighbourhoods.

Villmann et al. [73] proposed a topographic function to measure the neighbourhoodness of weight vectors in data space as well as on the lattice. While the neighbourhoodness of the weight vectors is defined by the adjacent Voronoi cells of the weights. The function measures the degree of weight vectors are ordered in the data space as to their indexes on the lattice, as well as how well the indexes are preserved when their weight vectors are neighbours.

Goodhill and Sejnowski [18] proposed the C measure, a correlation between the similarity of stimuli in the data space and the similarity of their prototypes in the map space, to quantify the topological preservation,

$$C = \sum_i \sum_j F(i, j) G[M(i), M(j)], \quad (3.23)$$

where F and G are symmetric similarity measures in the input and map spaces respectively and can be problem specific, and $M(i)$ and $M(j)$ are the mapped points or weight vectors of node i and j respectively.

The C measure directly evaluates the correlation between distance relations across two spaces. Various topographic mapping objectives may be unified under the C measure such as multidimensional scaling, minimal wiring, and travel salesperson problem (TSP), and noise tolerant VQ. It has also been shown that if a mapping that preserves ordering exists then maximising C will find it. Thus the C measure is also the objective function of the mapping, an important property different from other topology preservation measures and definitions.

One can always use the underlying cost function, Eq. (3.18), to measure the goodness of the resulting map including the topology preservation, at least one can use a temporal window to take a sample of it as suggested in [38]. The (final) neighbourhood function specifies the level of topology (ordering) the mapping is likely to achieve or is required. To make an analogy to the above C measure, the neighbourhood function can be interpreted as the G measure used in (3.25) and term $\|\mathbf{x} - \mathbf{w}_k\|^2$ represents the F measure. Indeed, the input \mathbf{x} and weight \mathbf{w}_j are mapped on the map as node index i and j and their G measure is the neighbourhood function such as exponentials. Such an analogy also sheds light on the scaling effect of the SOM. Multidimensional scaling also aims to preserve local similarities on a mapped space.

3.4 SOMs, Multidimensional Scaling and Principal Manifolds

3.4.1 Multidimensional Scaling

The SOM is often associated with VQ and clustering. However it is also associated with data visualisation, dimensionality reduction, nonlinear data projection, and manifold mapping. A brief review on various data projection methods and their relationships has been given before [80].

Multidimensional Scaling

Multidimensional scaling (MDS) is a traditional subject related to dimension reduction and data projection. MDS tries to project data points onto an often two-dimensional sheet by preserving as closely as possible the inter-point metrics [9]. The projection is generally nonlinear and can reveal the overall structure of the data. A general fitness function or the so-called *stress* function is defined as,

$$S = \frac{\sum_{i,j} (d_{ij} - D_{ij})^2}{\sum_{i,j} D_{ij}^2}, \quad (3.24)$$

where d_{ij} represents the proximity (dissimilarity) of data points i and j in the original data space, D_{ij} represents the distance (usually Euclidean) between mapped points i and j in the projected space.,

The MDS relies on an optimisation algorithm to search for a configuration that gives as low stress as possible. A gradient method is commonly used for this purpose. Inevitably, various computational problems such as local minima and divergence may occur to the optimisation process. The methods are also often computationally intensive. The final solution depends on the starting configuration and the parameters used in the algorithm.

Sammon mapping is a well-known example of MDS [65]. In Sammon mapping intermediate normalisation (of original space) is used to preserve good local distributions and at the same time maintain a global structure. The Sammon stress is expressed as,

$$S_{Sammon} = \frac{1}{\sum_{i < j} d_{ij}} \sum_{i < j} \frac{(d_{ij} - D_{ij})^2}{d_{ij}}. \quad (3.25)$$

A second order Newton optimisation method is used to recursively solve the optimal configuration. It converges faster than the simple gradient method, but the computational complexity is even higher. It still has the local minima and inconsistency problems. The Sammon mapping has been shown to be useful for data structure analysis. However, like other MDS methods, the Sammon algorithm is a point-to-point mapping, which does not provide the explicit mapping function and cannot naturally accommodate new data points. It also requires to compute and store all the inter-point distances. This proves difficult or even impossible for many practical applications where data arrives sequentially, the quantity of data is large, and/or memory space for the data is limited.

In addition to being computationally costly, especially for large data sets, and not adaptive, another major drawback of MDS methods including Sammon mapping is lack of an explicit projection function. Thus for any new input data, the mapping has to be recalculated based on all available data. Although some methods have been proposed to accommodate the new arrivals using triangulation [11, 44], the methods are generally not adaptive.

3.4.2 Principal manifolds

Principal component analysis

PCA is a classic linear projection method aiming at finding orthogonal principal directions from a set of data, along which the data exhibit the largest variances. By discarding the minor components, the PCA can effectively reduce data variables and display the dominant ones in a linear, low dimensional subspace. It is the optimal linear projection in the sense of the mean-square-error between original points and projected ones, i.e.,

$$\min_{\mathbf{x}} \sum_{\mathbf{x}} \left[\mathbf{x} - \sum_{j=1}^m (\mathbf{q}_j^T \mathbf{x}) \mathbf{q}_j \right]^2, \quad (3.26)$$

where $\{\mathbf{q}_j, j=1,2, \dots, m, m \leq n\}$ are orthogonal eigenvectors representing principal directions. They are the first m principal eigenvectors of the covariance matrix of the input. The second term in the above bracket is the reconstruction or projection of \mathbf{x} on these eigenvectors. The term $\mathbf{q}_j^T \mathbf{x}$ represents the projection of \mathbf{x} onto the j -th principal dimension. Traditional methods for solving eigenvector problem involve numerical methods. Though fairly efficient and robust, they are not usually adaptive and often require the presentation of the entire data set. Several Hebbian-based learning algorithms and neural networks have been proposed for performing PCA such as, the subspace network [54] and the generalised Hebbian algorithm [64]. The limitation of linear PCA is obvious, as it cannot capture nonlinear relationships defined by higher than the second order statistics. If the input dimension is much higher than two, the projection onto linear principal plane will provide limited visualisation power.

Nonlinear PCA and principal manifolds

The extension to nonlinear PCA (NLPCA) is not unique, due to the lack of a unified mathematical structure and an efficient and reliable algorithm, and in some cases due to excessive freedom in selection of representative basis functions [51, 28]. Several methods have been proposed for nonlinear PCA such as, the five-layer feedforward associative network [41] and the kernel PCA [66]. The first three layers of the associative network project the original data on to a curve or surface, providing an activation value for the bottleneck node. The last three layers define the curve and surface. The weights of the associative NLPCA network are determined by minimising the following objective function,

$$\min_{\mathbf{x}} \sum_{\mathbf{x}} \|\mathbf{x} - \mathbf{f}\{s_f(\mathbf{x})\}\|^2, \quad (3.27)$$

where $\mathbf{f}: \mathbb{R}^1 \rightarrow \mathbb{R}^n$ (or $\mathbb{R}^2 \rightarrow \mathbb{R}^n$), the function modelled by the last three layers, defines a curve (or a surface), $s_f: \mathbb{R}^n \rightarrow \mathbb{R}^1$ (or $\mathbb{R}^n \rightarrow \mathbb{R}^2$), the function modelled by the first three layers, defines the projection index.

The kernel-based PCA uses nonlinear mapping and kernel functions to generalise PCA to NLPCA and has been used for various pattern recognition. The nonlinear function $\Phi(\mathbf{x})$ maps data onto high-dimensional feature space, where the standard linear PCA can be performed via kernel functions: $k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$. The projected covariance matrix is then,

$$Cov = \frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^T. \quad (3.28)$$

The standard linear eigenvalue problem can now be written as $\lambda \mathbf{V} = \mathbf{K} \mathbf{V}$, where the columns of \mathbf{V} are the eigenvectors and \mathbf{K} is a $N \times N$ matrix with elements as kernels $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$.

The principal curves and principal surfaces [20, 43] are the principled nonlinear extension of PCA. The principal curve is defined as a smooth and self-consistency curve, which does not intersect itself. Denote \mathbf{x} as a random vector in \mathbb{R}^n with density p and finite second moment. Let $f(\cdot)$ be a smooth unit-speed curve in \mathbb{R}^n , parameterised by the arc length ρ (from one end of the curve) over $\Lambda \in \mathbb{R}$, a closed interval.

For a data point \mathbf{x} , its projection index on f is defined as

$$\rho_f(\mathbf{x}) = \sup_{\rho \in \Lambda} \{\rho : \|\mathbf{x} - f(\rho)\| = \inf_{\vartheta} \|\mathbf{x} - f(\vartheta)\|\}. \quad (3.29)$$

The curve is called self-consistent or a principal curve of ρ if

$$f(\rho) = E[\mathbf{X} | \rho_f(\mathbf{X}) = \rho]. \quad (3.30)$$

The principal component is a special case of the principal curves if the distribution is ellipsoidal. Although principal curves have been mainly studied, extension to higher dimension, e.g. principal surfaces or manifolds is feasible in principle. However, in practice, a good implementation of principal curves/surfaces relies on an effective and efficient algorithm. The principal curves/surfaces are more of a concept that invites practical implementations. The HS algorithm [20] proposed by Hastie and Stuezle is a nonparametric method, which directly iterates the two steps of the above definition. It is similar to the standard LGB VQ algorithm [46] combined with some smoothing techniques.

HS algorithm:

- Initialisation: Choose the first linear principal component as the initial curve, $f^{(0)}(\mathbf{x})$.
- Projection: Project data points onto the current curve and calculate the projections index, i.e. $\rho^{(t)}(\mathbf{x}) = \rho_{f(t)}(\mathbf{x})$.
- Expectation: For each index, take the mean of data points projected onto it as the new curve point, i.e., $f^{(t+1)}(\rho) = E[\mathbf{X} | \rho_{f(t)}(\mathbf{X}) = \rho]$.

The projection and expectation steps are repeated until a convergence criterion is met, e.g. when the change of the curve between iterations is below a threshold.

For a finite data set, the density p is often unknown, the above expectation is replaced by a smoothing method such as the locally weighted running-line smoother or smoothing splines. For kernel regression, the smoother is,

$$f(\rho) = \frac{\sum_{i=1}^N \mathbf{x}_i \kappa(\rho, \rho_i)}{\sum_{i=1}^N \kappa(\rho, \rho_i)}. \quad (3.31)$$

The arc length is simply computed from the line segments. There are no proofs of convergence of the algorithm, but no convergence problems have been reported, though the algorithm is biased in some cases [20]. Banfield and Raftery [3] have modified the HS algorithm by taking the expectation of the residual of the projections in order to reduce the bias. Kegl et al [31] have proposed an incremental, e.g. segment by segment, and arc length constrained method for practical construction of principal curves.

Tibshirani [70] has introduced a semi-parametric model for the principal curve. A mixture model was used to estimate the noise along the curve; and the expectation and maximisation (EM) method was employed to estimate the parameters. Other options for finding the nonlinear manifold include the GTM [5] and probabilistic principal surfaces [6]. These methods model the data by a means of a latent space. They belong to the semi-parameterised mixture model, although types and orientations of the local distributions vary from method to method.

3.4.3 Visualisation induced SOM (ViSOM)

For scaling and data visualisation, a direct and faithful display of data structure and distribution is highly desirable. ViSOM has been proposed to extend the SOM for directly distance preservation on the map [78], instead of using a colouring scheme such as U-matrix [72], which imprints qualitatively the interneuron distances as colours or grey levels on the map. For the map to capture the data structure naturally and directly, (local) distance quantities must be preserved on the map, along with the topology. The map can be seen as a smooth and graded mesh or manifold embedded into the data space, onto which the data points are mapped and the inter-point distances are approximately preserved.

In order to achieve that, the updating force, $\mathbf{x}(t) - \mathbf{w}_k(t)$, of the SOM algorithm is decomposed into two elements $[\mathbf{x}(t) - \mathbf{w}_v(t)] + [\mathbf{w}_v(t) - \mathbf{w}_k(t)]$. The first term represents the updating force from the winner v to the input $\mathbf{x}(t)$, and is the same to the updating force used by the winner. The second force is a lateral contraction force bringing neighbouring neuron k to the winner v . In the ViSOM, this lateral contraction force is constrained or regulated in order

to help maintain a unified local inter-neuron distance $\|\mathbf{w}_v(t) - \mathbf{w}_k(t)\|$ on the map.

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \alpha(t)\eta(v, k, t)\{[\mathbf{x}(t) - \mathbf{w}_v(t)] + \beta[\mathbf{w}_v(t) - \mathbf{w}_k(t)]\}, \quad (3.32)$$

where the simplest constraint can be $\beta := d_{vk}/(D_{vk}\lambda) - 1$, with d_{vk} the distance of neuron weights in the input space, D_{vk} the distance of neuron indexes on the map, and λ a (required) resolution constant.

The ViSOM regularises the contraction force so that the distances between the nodes on the map are analogous to the distances of their weights in the data space. The aim is to adjust inter-neuron distances on the map in proportion to those in the data space, i.e. $\lambda D_{vk} \propto d_{vk}$. When the data points are eventually projected on a trained map, the distance between point i and j on the map is proportional to that of the original space, subject to the quantisation error (the distance between a data point and its neural representative). This has a similar effect to Sammon mapping, which also aims at achieving this proportionality, $D_{ij} \propto d_{ij}$. The key feature of the ViSOM is that the distances between the neurons (which data are mapped to) on the map (in a neighbourhood) reflect the corresponding distances in the data space. When the map is trained and data points mapped, the distances between mapped data points on the map will resemble approximately those in the original space (subject to the resolution of the map). This makes visualisation more direct, quantitatively measurable, and visually appealing. The map resolution can be enhanced by interpolating a trained map or incorporating local linear projections [81]. The size or covering range of the neighbourhood function can also be decreased from an initially large value to a final smaller one. The final neighbourhood, however, should not contain just the winner. The rigidity or curvature of the map is controlled by the ultimate size of the neighbourhood. The larger of this size the flatter the final map is in the data space. Guidelines for setting these parameters have been given in [79]. An example on data visualisation will be shown in the next section.

Several authors have since introduced improvements and extensions on the ViSOM. For example, in [77], a probabilistic data assignment [19] is used in both the input assignment and the neighbourhood function and an improved second order constraint is adopted. The resulting SOM has a clearer connection to an MDS cost function. In [15] the ViSOM has been extended to arbitrary, neural gas type of map structure. Various existing variants of the SOM such as hierarchical, growing and hierarchical and growing structures are readily extendable to the ViSOM for various application needs.

The SOM has been related to the discrete principal curve/surface algorithm [61]. However the differences remain in both the projection and smoothing processes. In the SOM the data are projected onto the nodes rather than onto the curve. The principal curves perform the smoothing entirely in the data space –see Eq. (3.31). The smoothing process in the SOM and ViSOM, as a convergence criterion, is [79],

$$\mathbf{w}_k = \frac{\sum_{i=1}^L \mathbf{x}_i \eta(v, k, i)}{\sum_{i=1}^L \eta(v, k, i)}. \quad (3.33)$$

The smoothing is governed by the indexes of the neurons in the map space. The kernel regression uses the arc length parameters (ρ, ρ_i) or $\|\rho - \rho_i\|$ exactly, while the neighbourhood function uses the node indexes (k, i) or $\|k - i\|$. Arc lengths reflect the curve distances between the data points. However, node indexes are integer numbers denoting the nodes or the positions on the map grid, not the positions in the input space. So $\|k - i\|$ does not resemble $\|\mathbf{w}_k - \mathbf{w}_i\|$ in the common SOM. In the ViSOM, however, as the local inter-neuron distances on the map represent those in the data space (subject to the resolution of the map), the distances of nodes on the map are in proportion to the difference of their positions in the data space, i.e. $\|k - i\| \sim \|\mathbf{w}_k - \mathbf{w}_i\|$. The smoothing process in the ViSOM resembles that of the principal curves as shown below,

$$\mathbf{w}_k = \frac{\sum_{i=1}^L \mathbf{x}_i \eta(v, k, i)}{\sum_{i=1}^L \eta(v, k, i)} \approx \frac{\sum_{i=1}^L \mathbf{x}_i \eta(\mathbf{w}_v, \mathbf{w}_k, i)}{\sum_{i=1}^L \eta(\mathbf{w}_v, \mathbf{w}_k, i)}. \quad (3.34)$$

It shows that ViSOM is a better approximation to the principal curve than the SOM. The SOM and ViSOM are similar only when the data are uniformly distributed and when the number of nodes becomes very large, in which case both the SOM and ViSOM will closely approximate the principal curve/surface.

3.5 Examples

There have been reported thousands of applications of the SOM and its variants [39, 29, 56] since its introduction, too many to list here. There are a dedicated international Workshop on SOMs (WSOM) and focused sessions in many neural networks conferences. There have also been several special issues dedicated to the advances in the SOM and related topics [1, 27, 8]. Still many new applications are being reported in many relevant journals today. SOMs will remain an active topic in their continued extension, combination and applications in the years to come.

Typical applications include image and video processing and retrieval; density or spectrum profile modelling; nonlinear ICA; classification (LVQ); cross-modal information processing and associations; data visualisations; text and document mining and management systems; gene expression data analysis and discovery; novelty detection; robotics and computer animation. In this section, we take a slice of typical applications and present several examples on high dimensional data visualisation and scaling only.

3.5.1 Data visualisation

Data projection and visualisation has become a major application area for neural networks, in particular for the SOMs [39], as its topology preserving property is unique among other neural models. Good projection and visualisation methods help to identify clustering tendency, to reveal the underlying functions and patterns, and to facilitate decision support. A great deal of research has been devoted to this subject and a number of methods have been proposed.

The SOM has been widely used as a visualisation tool for dimensionality reduction (e.g. [25, 30, 39, 72]). The SOM's unique topology preserving property can be used to visualise the relative mutual relationships among the data. However, the SOM does not directly apply to scaling, which aims to reproduce proximity in (Euclidean) distance on a low visualisation space, as it has to rely on a colouring scheme (e.g. the U-matrix method [72] to imprint the distances crudely on the map. Often the distributions of the data points are distorted on the map. The ViSOM [78, 79] constrains the lateral contraction force between the neurons in the SOM and hence regularises the inter-neuron distances with respect to a scaleable parameter that defines and controls the resolution of the map. It preserves the data structure as well as the topology as faithfully as possible. The ViSOM provides a direct visualisation of both the structure and distribution of the data. An example is shown in Fig. 3.3, where the ViSOM of 100×100 (hexagonal) was used to map the 4-D Iris data set and it gives direct visualisation of data distribution, similar to the sammon mapping. Although, the SOM with colouring can show the gap between iris setosa and the rest, it is impossible to capture the data structure and represent the data proximity on the map.

Usually for a fine mapping, the resolution parameter needs to be set to small value and a large number of nodes, i.e. a large map, is required, as for all discrete mappings. However such computational burden can be greatly reduced by interpolating a trained map [83] or incorporating a local linear projection on a trained low resolution map [81].

A comparison with other mapping methods- PCA, Sammon and LLE - on an "S" shape manifold is also shown in Fig. 3.4.

3.5.2 Document organisation and content management

With drastically increasing amount of unstructured content available electronically within an enterprise or on the web, it is becoming inefficient if not impossible to rely on human operators to manually annotate electronic documents. (Web) content management systems have become an important area of research in many applications such as e-libraries, enterprise portals, e-commerce, software contents management, document management and knowledge discovery. The documents, generated in an enterprise either centrally or locally by employees, are often unstructured or arranged in ad hoc manner

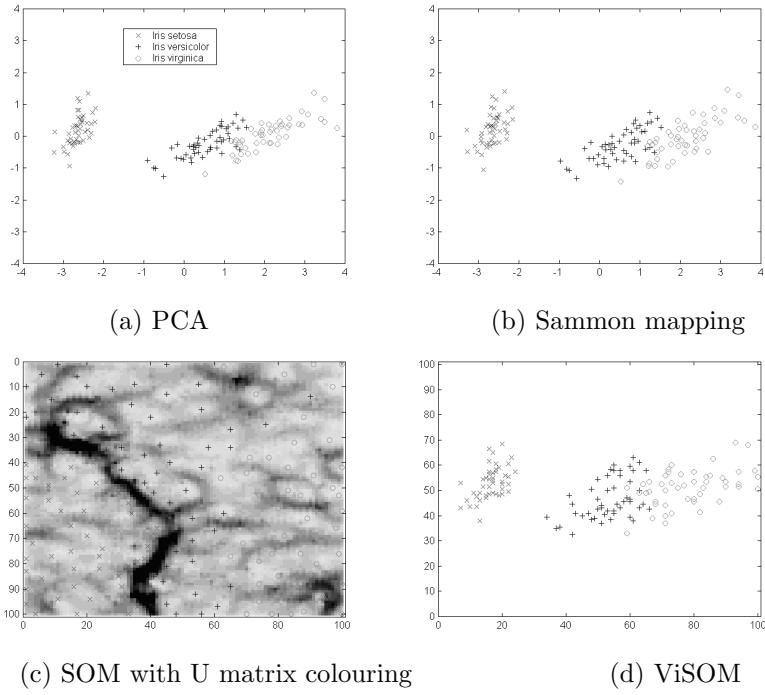


Fig. 3.3. Mapping and visualisation of Iris data set.

(e.g. emails, reports, web pages, presentations). Document management addresses many issues storage, indexing, security, revision control, retrieval and organization of documents. Many existing full-text search engines return a large ranked list of documents, many of which are irrelevant. This is especially true when queries are short and very general words are used. Hence the document organization has become important in information retrieval and content management.

Documents can be treated as feature vectors. There are many methods to extract features such as word frequencies from documents (cf. [16]). The SOM has been applied to organise and visualise vast amount of textual information. Typical examples include the Welfaremap [30] and the WEBSOM [25]. Many variants of the SOM have been proposed and applied to document organization, e.g. TreeGCS [24] and growing hierarchical-SOM (GH-SOM) [57]. The main advantage of the SOM is the topology preservation of input space, which makes similar topics appear closely on the map. Most these applications however are based on 2D maps and grids, which are intuitive for digital library idea. However such a 2D grid presentation of information (mainly document files) is counter to all existing computer file organisers and explorers such as Windows Explorer. A new way of utilising the SOM as a

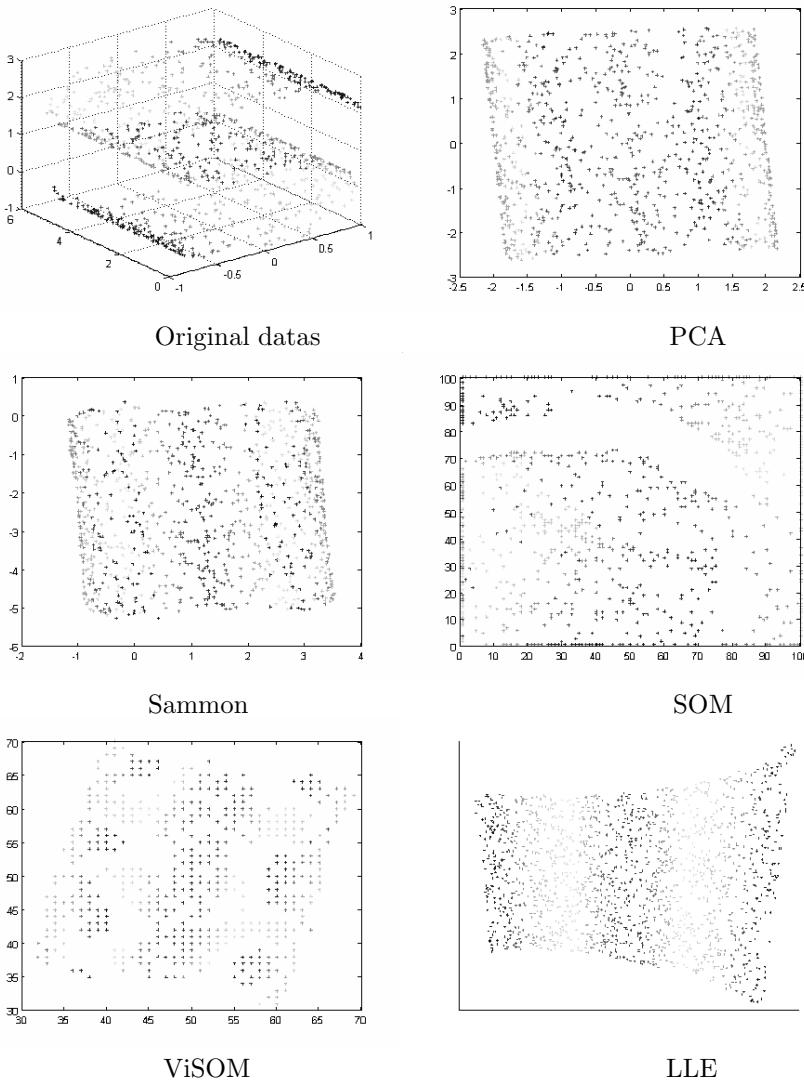


Fig. 3.4. Manifold mapping obtained by various methods.

topology-preserving tree structure for content management and knowledge discovery has been proposed [16]. The method can generate a taxonomy of topics from a set of unannotated, unstructured documents. It consists of a hierarchy of self-organizing growing chains, each of which can develop independently in terms of size and topics. The dynamic development process is validated continuously using a proposed entropy-based Bayesian information

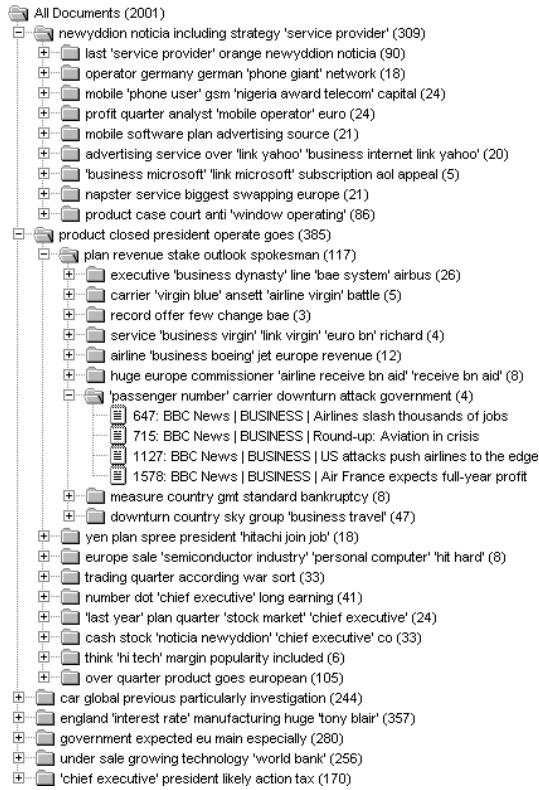


Fig. 3.5. A typical result of topological tree structure for organising documents.

criterion. Each chain meeting the criterion spans child chains, with reduced vocabularies and increased specializations.

This results in a topological tree hierarchy, which can be browsed like a table of contents directory or web portal. A typical tree is shown in Fig. 3.5. This approach has been tested and compared with several existing methods on real world web page datasets. The results have clearly demonstrated the advantages and efficiency in content organization of the proposed method in terms of computational cost and representation. The preserved topology provides a unique, additional feature for retrieving related topics and confining the search space.

An application prototype developed based this method is shown in Fig. 3.5. The left panel displays the generated content tree with various levels and preserved topology on these levels. The right panel shows the details of a selected level or branch or a particular document. The method bears familiar interface to the most popular Windows explorer style.

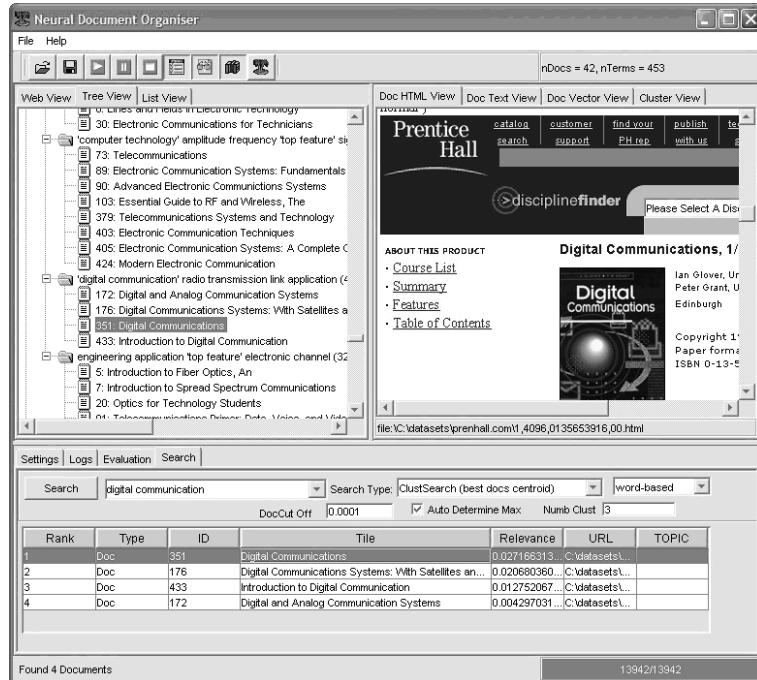


Fig. 3.6. A screen shot of a document management system using topological tree structure.

References

1. Allinson, N.M., Obermayer, K. and Yin, H.: Neural Networks, Special Issue on New Developments in Self-Organising Maps, **15**, 937–1155 (2002)
2. Ameri, S.-I.: Topographic organisation of nerve fields. *Bulletin of Mathematical Biology*, **42**, 339–364 (1980)
3. Banfield, J.D. and Raftery, A.E.: Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *Journal of the American Statistical Association*, **87**, 7–16 (1992)
4. Bauer, H.-U. and Pawelzik, K.R.: Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Trans. Neural Networks*, **3**, 570–579 (1992)
5. Bishop, C.M., Svensén, M., and Williams, C.K.I.: GTM: The generative topographic mapping. *Neural Computation*, **10**, 215–235 (1998)
6. Chang, K.-Y. and Ghosh, J.: A unified model for probabilistic principal surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **23**, 22–41 (2001)
7. Cottrell, M. and Fort, J.C.: A stochastic model of retinotopy: a self-organising process. *Biological Cybernetics*, **53**, 405–411 (1986)
8. Cottrell, M. and Verleysen, M.: Neural Networks, Special Issue on Advances in Self-Organizing Maps, **19**, 721–976 (2006)
9. Cox, T.F. and Cox, M.A.A.: Multidimensional Scaling, Chapman & Hall (1994)
10. de Bolt, E., Cottrell, M., and Verleysen, M.: Statistical tools to assess the reliability of self-organising maps. *Neural Networks*, **15**, 967–978 (2002)

11. De Ridder, D. and Duin, R.P.W.: Sammon mapping using neural networks: a comparison. *Pattern Recognition Letters*, **18**, 1307–1316 (1997)
12. Durbin, R. and Mitchison, G.: A dimension reduction framework for understanding cortical maps. *Nature*, **343**, 644–647 (1990)
13. Erwin, E., Obermayer, K. and Schulten, K.: Self-organising maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, **67**, 47–55 (1992)
14. Erwin, E., Obermayer, K. and Schulten, K.: Self-organising maps: stationary states, metastability and convergence rate. *Biological Cybernetics*, **67**, 35–45 (1992)
15. Estévez, P.A and Figueroa, C.J.: Online data visualization using the neural gas network. *Neural Networks*, **19**, 923–934 (2006)
16. Freeman, R. and Yin, H., Adaptive topological tree structure (ATTS) for document organisation and visualisation. *Neural Networks*, **17**, 1255–1271 (2004)
17. Gaze, R.M.: *The Information of Nerve Connections*, Academic Press (1970)
18. Goodhill, G.J. and Sejnowski, T.: A unifying objective function for topographic mappings. *Neural Computation*, **9** 1291–1303 (1997)
19. Graepel, T., Burger, M. and Obermayer, K.: Phase transitions in stochastic self-organizing maps. *Phys. Rev. E*, **56** 3876–3890 (1997)
20. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association*, **84**, 502–516 (1989)
21. Haykin, S.: *Neural Networks: A Comprehensive Foundation*, second edition, Prentice Hall, (1998)
22. Hebb, D.: *Organisation of Behaviour*, John Wiley & Sons (1949)
23. Heskes, T.: Energy functions for self-organizing maps. *Kohonen Maps*, E. Oja and S. Kaski (eds.), Elsevier, Amsterdam, 303–315 (1999)
24. Hodge, V.J. and Austin, J.: Hierarchical growing cell structures: TreeGCS. *IEEE Trans. Knowledge and Data Engineering*, **13**, 207–218 (2001)
25. Honkela, T., Kaski, S., Lagus, K., and Kohonen, T.: WEBSOM-self-organizing maps of document collections. *Proc. Workshop on Self-Organizing Maps (WSOM'97)*, 310–315 (1997)
26. Hyvärinen, A., Karhunen, J. and Oja, E.: *Independent Component Analysis*. John Wiley & Sons, Inc. (2001)
27. Ishikawa, M., Miikkulainen R. and Ritter, H.: *Neural Network, Special Issue on New Developments in Self-Organizing Systems*, **17**, 1037–1389 (2004)
28. Karhunen, J., and Joutsensalo, J.: Generalisation of principal component analysis, optimisation problems, and neural networks. *Neural Networks*, **8**, 549–562 (1995)
29. Kaski, S., Kangas, J. and Kohonen, T.: Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, **1**, 1–176 (1998)
30. Kaski, S., and Kohonen, T.: Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In: Refenes, A.-P.N., Abu-Mostafa, Y., Moody, J., and Weigend, A. (eds.) *Neural Networks in Financial Engineering*, World Scientific, 498–507 (1996)
31. Kegl, B., Krzyzak, A., Linder, T., and Zeger, K.: A polygonal line algorithm for constructing principal curves. *Neural Information Processing Systems (NIPS'98)*, **11**, 501–507 (1998)
32. Kohonen, T.: Correlation matrix memory. *IEEE Trans. Computers*, **21**, 353–359 (1972)

33. Kohonen, T.: A new model for randomly organised associative memory. *Int. Journal of Neuroscience*, **5**, 27–29 (1973)
34. Kohonen, T.: An adaptive associative memory principle. *IEEE Trans. Computers*, **23**, 444–445 (1974)
35. Kohonen, T.: Self-organised formation of topologically correct feature map. *Biological Cybernetics*, **43**, 56–69 (1982)
36. Kohonen, T.: *Self-organization and associative memory*, Springer (1984)
37. Kohonen, T.: Representation of sensory information in self-organising feature maps, and relation of these maps to distributed memory networks. *Proc. SPIE*, **634**, 248–259 (1986)
38. Kohonen, T.: Self-organizing maps: optimization approaches. In: *Artificial Neural Networks*, vol. 2. North-Holland: Amsterdam, 981–990 (1991)
39. Kohonen, T.: *Self-Organising Maps*, second edition, Springer (1997)
40. Kohonen, T.: Comparison of SOM point densities based on different criteria. *Neural Computation*, **11**, 2081–2095 (1999)
41. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. *AICHE Journal*, **37**, 233–243 (1991)
42. Lampinen, J. and Oja, E.: Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, **2**, 261–272 (1992)
43. LeBlanc, M., and Tibshirani, R. J.: Adaptive principal surfaces. *J. Amer. Statist. Assoc.* **89**, 53–64 (1994)
44. Lee, R.C.T., Slagle, J.R., and Blum, H.: A triangulation method for the sequential mapping of points from n-space to two-space. *IEEE Trans. Computers*, **27**, 288–292 (1977)
45. Lin, S. and Si, J.: Weight-value convergence of the SOM algorithm for discrete input. *Neural Computation*, **10**, 807–814 (1998)
46. Linde, Y., Buzo, A. and Gray, R.M.: An algorithm for vector quantizer design. *IEEE Trans. Communications*, **28**, 84–95 (1980)
47. Lo, Z.P. and Bavarian, B.: On the rate of convergence in topology preserving neural networks. *Biological Cybernetics*, **65**, 55–63 (1991)
48. Luttrell, S.P.: Derivation of a class of training algorithms. *IEEE Trans. Neural Networks*, **1**, 229–232 (1990)
49. Luttrell, S.P.: Code vector density in topographic mappings: Scalar case, *IEEE Trans. Neural Networks*, **2**, 427–436 (1991)
50. Luttrell, S.P. A Bayesian analysis of self-organising maps. *Neural Computation*, **6**, 767–794 (1994)
51. Malthouse, E.C.: Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Trans. Neural Networks*, **9**, 165–173 (1998)
52. Marr, D.: A theory of cerebellar cortex. *Journal of Physiology*, **202**, 437–70 (1969)
53. Mitchison, G.: A type of duality between self-organising maps and minimal wiring, *Neural Computation*, **7**, 25–35 (1995)
54. Oja, E.: Neural networks, principal components, and subspaces. *Int. Journal of Neural Systems*, **1**, 61–68 (1989)
55. Oja, E.: PCA, ICA, and nonlinear Hebbian learning. *Proc. Int. Conf. on Artificial Neural Networks (ICANN'95)*, 89–94 (1995)
56. Oja, M., Kaski, S. and Kohonen, T.: Bibliography of self-organizing map (SOM) papers: 1998–2001 addendum. *Neural Computing Surveys*, **3**: 1–156 (2003)

57. Rauber, A., Merkl, D., and Dittenbach, M.: The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Trans. Neural Networks*, **13** 1331–1341 (2002)
58. Ripley, B.D.: *Pattern Recognition and Neural Networks*, Cambridge University Press (1996)
59. Ritter, H.: Asymptotical level density for class of vector quantisation processes. *IEEE Trans. Neural Networks*, **2**, 173–175 (1991)
60. Ritter, H. and Schulten, K.: Convergence properties of Kohonen's topology conserving maps: fluctuations, stability, and dimension selection. *Biological Cybernetics*, **60**, 59–71 (1988)
61. Ritter, H., Martinetz, T., and Schulten, K.: *Neural Computation and Self-organising Maps: An Introduction*. Addison-Wesley Publishing Company (1992)
62. Robbins, H. and Monro, S.: A stochastic approximation method. *Annals of Math. Statist.*, **22**, 400–407 (1952)
63. Roweis, S. T., and Saul, L. K.: Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**, 2323–2326 (2000)
64. Sanger, T. D.: Optimal unsupervised learning in a single-layer linear feedforward network. *Neural Networks*, **2**, 459–473 (1991)
65. Sammon, J. W.: A nonlinear mapping for data structure analysis. *IEEE Trans. Computer*, **18**, 401–409 (1969)
66. Schölkopf, B., Smola, A., and Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, **10**, 1299–1319 (1998)
67. Sum, J., Leung, C.-S., Chan, L.-W., and Xu, L.: Yet another algorithm which can generate topography map. *IEEE Trans. Neural Networks*, **8**, 1204–1207 (1997)
68. Sutton, R.S., Barto, A.G., and Williams, R.J. (1991). Reinforcement learning is direct adaptive optimal control. *Proc. American Control Conference*, 2143–2146.
69. Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290: 2319–2323.
70. Tibshirani, R.: Principal curves revisited. *Statistics and Computation*, **2**, 183–190 (1992)
71. Törönen, P., Kolehmainen, K., Wong, G., and Castrén, E.: Analysis of gene expression data using self-organising maps. *FEBS Letters*, **451**, 142–146 (1999)
72. Ultsch, A.: Self-organising neural networks for visualisation and classification. In: Opitz, O. Lausen, B., and Klar, R. (eds.) *Information and Classification*, 864–867 (1993)
73. Villmann, T., Der, R., Herrmann, M., and Martinetz, T.M.: Topology preservation in self-organizing feature maps: exactdefinition and measurement. *IEEE Trans. Neural Networks*, **8**, 256–266 (1997)
74. von der Malsburg, C. and Willshaw, D.J.: Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, **4**, 85–100 (1973)
75. Willshaw, D.J., Buneman, O.P., and Longuet-Higgins, H.C.: Non-holographic associative memory. *Nature*, **222**, 960–962 (1969)
76. Willshaw, D.J. and von der Malsburg, C.: How patterned neural connections can be set up by self-organization, *Proc. Royal Society of London. Series B*, **194**, 431–445 (1976)
77. Wu, S. and Chow, T.W.S.: PRSOM: A new visualization method by hybridizing multidimensional scaling and self-organizing map. *IEEE Trans. Neural Networks*, **16**, 1362–1380 (2005)

78. Yin, H.: ViSOM-A novel method for multivariate data projection and structure visualisation. *IEEE Trans. Neural Networks*, **13**, 237–243 (2002)
79. Yin, H.: Data visualisation and manifold mapping using the ViSOM. *Neural Networks*, **15**, 1005–1016 (2002)
80. Yin, H.: Nonlinear multidimensional data projection and visualisation. *Proc. IDEAL'03*, 377–388 (2003)
81. Yin, H.: Resolution enhancement for the ViSOM. *Proc. Workshop on Self-Organizing Maps*, 208–212 (2003)
82. Yin, H. and Allinson, N.M.: On the distribution and convergence of the feature space in self-organising maps. *Neural Computation*, **7**, 1178–1187 (1995)
83. Yin, H. and Allinson, N. M.: Interpolating self-organising map (iSOM). *Electronics Letters*, **35**, 1649–1650 (1999)

Elastic Maps and Nets for Approximating Principal Manifolds and Their Application to Microarray Data Visualization

Alexander N. Gorban^{1,3} and Andrei Y. Zinov'yev^{2,3}

¹ University of Leicester, University Road, Leicester, LE1 7RH, UK,
ag153@le.ac.uk

² Institut Curie, 26, rue d'Ulm, Paris, 75248, France,
andrej.zinov'yev@curie.fr

³ Institute of Computational Modeling of Siberian Branch of Russian Academy of Sciences, Krasnoyarsk, Russia

Summary. Principal manifolds are defined as lines or surfaces passing through “the middle” of data distribution. Linear principal manifolds (Principal Components Analysis) are routinely used for dimension reduction, noise filtering and data visualization. Recently, methods for constructing non-linear principal manifolds were proposed, including our *elastic maps* approach which is based on a physical analogy with elastic membranes. We have developed a general geometric framework for constructing “principal objects” of various dimensions and topologies with the simplest quadratic form of the smoothness penalty which allows very effective parallel implementations. Our approach is implemented in three programming languages (C++, Java and Delphi) with two graphical user interfaces (VidaExpert and ViMiDa applications). In this paper we overview the method of elastic maps and present in detail one of its major applications: the visualization of microarray data in bioinformatics. We show that the method of elastic maps outperforms linear PCA in terms of data approximation, representation of between-point distance structure, preservation of local point neighborhood and representing point classes in low-dimensional spaces.

Key words: elastic maps, principal manifolds, elastic functional, data analysis, data visualization, surface modeling

4.1 Introduction and Overview

4.1.1 Fréchet Mean and Principal Objects: K-Means, PCA, what else?

A fruitful development of the statistical theory in the 20th century was an observation made by Fréchet in 1948 [12] about the non-applicability of the

mean value calculation in many important applications when the space of multi-dimensional measurements is not linear. The simple formula $M(x) = (1/N) \sum_{i=1}^N x_i$ where $x = \{x_i\}$ does not work if data points x are defined on a Riemannian manifold (then $M(x)$ can simply not be located on it) or the distance metric defined in the data space is not quadratic.

This led to an important generalization of the mean value notion as a set which minimizes mean-square distance to the set of data samples:

$$M_F(x) = \arg \min_{y \in D} \sum_i (dist(y, x_i))^2, \quad (4.1)$$

where D is the space of data samples and $dist(x, y)$ is a distance function between two points in D .

The existence and uniqueness of the Fréchet mean (4.1) is not generally guaranteed, but in many applications (one of the most important is statistical analysis in the shape space, see [32]) it proves to be constructive. In vector spaces $M_F = M$ only in the case of Euclidean (more generally, quadratic) metrics, whereas (4.1) allows to define a mean value with an alternative choice of metric.

Let us look at Fréchet definition from a different angle. We would like to utilize such spaces D in (4.1) that does not necessarily coincide with the space of data samples. Let us consider a space of objects such that one can measure distance from an object y to a sample x_i . Thus, using (4.1) one can find one (or a set of) “mean” or “principal” object in this space with respect to the dataset X .

Interestingly, simple examples of “principal objects” are equivalent to application of very well-known methods in statistical data analysis.

For example, let us consider D as a space of k -tuples, i.e. k -element sets of vectors in R^m (centroids), where m is the dimension of the data space, and define distance between $y \in D$ and a point $x_i \in R^m$ as

$$dist(y, x_i) = \min_{k=1..k} \|y_k - x_i\|, \quad (4.2)$$

where y_i is the i th vector in the k -tuple y . Then the Fréchet mean corresponds to the optimal positions of centroids in the K -means clustering problem⁴. Detailed discussion of the data approximation (“recovering”) approach with application to clustering is presented in [40].

Another important example is the space D of linear manifolds embedded in R^m . Let us consider a linear manifold y defined in the parametric form $y(t) = a + bt$, where $t \in [-\infty; +\infty]$ is a parameter and $a, b \in R^m$. Let us define the function $dist(y, x)$ as the distance to the orthogonal projection of x on y (i.e., to the closest point on y):

⁴ Usually by K -means one means an iterative algorithm which allows us to calculate the locally optimal position of k centroids. To avoid confusion, we can say that the K -means clustering problem consists in finding a good approximation to the global optimum and the algorithm is one of the possible ways to get close to it.

$$dist(y, x_i) = \min_{t \in [-\infty; +\infty]} \|y(t) - x_i\|. \quad (4.3)$$

Then, in the case of Euclidean metrics $\|y(t) - x\|^2 = (y(t) - x)^2$ the “mean” manifold $y = M_F(x)$ corresponds exactly to the first principal component of X (and this is exactly the first Pearson definition [43]). Analogously, the k -dimensional “mean” linear manifold corresponds to the k -dimensional principal linear manifold.

The initial geometrical definition of the principal component as a line minimizing mean square distance from the data points has certain advantages in application. First, highly effective algorithms for its computation are proposed in the case of data spaces with very high dimension. Second, this definition is easily adapted to the case of incomplete data, as we show below.

In a certain sense, K -means clustering and linear principal manifolds are extreme cases of “principal objects”: the first one is maximally flexible with no particular structure, simply a k -tuple, while the second one presents a rigid linear manifold fitted to data. It is known that linear principal components (including the mean value itself, as a 0-dimensional principal object) are optimal data estimators only for gaussian data distributions. In practice we are often faced with non-gaussianities in the data and then linear PCA is optimal only in the class of *linear* estimators. It is possible to construct a “principal object” in-between these two extreme alternatives (K -Means *vs* PCA), such that it would be flexible enough to adapt to data non-gaussianity and still have some regular structure.

Elastic nets were developed in a series of papers [16, 17, 19, 20, 21, 24]. By their construction, they are system of elastic springs embedded in the data space. This system forms a regular grid such that it can serve for approximation of some low-dimensional manifold. The elastic coefficients of this system allow the switch from completely unstructured K -means clustering (zero elasticity) to the estimators located closely to linear PCA manifolds (very rigid springs). With some intermediate values of the elasticity coefficients, this system effectively approximates non-linear principal manifolds which are briefly reviewed in the next section.

4.1.2 Principal Manifolds

Principal manifolds were introduced in PhD thesis of Hastie [28] and then in the paper of Hastie and Stuetze [29] as lines or surfaces passing through “the middle” of the data distribution. This intuitive definition was supported by a mathematical notion of self-consistency: every point of the principal manifold is a conditional mean of all points that are projected into this point. In the case of datasets only one or zero data points are projected in a typical point of the principal manifold, thus, one has to introduce smoothers that become an essential part of the principal manifold construction algorithms.

Since these pioneering works, many modifications and alternative definitions of principal manifolds have appeared in the literature. Theoretically, existence of self-consistent principal manifolds is not guaranteed for arbitrary probability distributions. Many alternative definitions were introduced (see, for example, [34]) in order to improve the situation and to allow the construction of principal curves (manifolds) for a distribution of points with several finite first moments. A computationally effective and robust algorithmic kernel for principal curve construction, called the Polygonal algorithm, was proposed by Kégl et al. [36]. A variant of this strategy for constructing principal graphs was also formulated in the context of the skeletonization of hand-written digits [35]. “Kernalised” version of PCA was developed in [50]. A general setting for constructing regularized principal manifolds was proposed in [52, 53]. In a practical example [53] of oil flow data it was demonstrated that non-linear principal manifolds reveal point class structure better than the linear ones. Piece-wise construction of principal curves by fitting unconnected line segments was proposed in [57].

Probably, most scientific and industrial applications of principal manifold methodology were implemented using the Kohonen Self-Organizing Maps (SOM) approach developed in the theory of neural networks [33]. These applications are too numerous to be mentioned here (see reviews in [31, 42, 62, 63]): we only mention that SOMs and its numerous improvements, indeed, can provide principal manifold approximations (for example, see [41, 46]) and are computationally effective. The disadvantage of this approach is that it is entirely based on heuristics; also it was shown that in the SOM strategy there does not exist any objective function that is minimized by the training process [11]. These limitations were discussed in [3], and the Generative Topographic Mapping (GTM) approach was proposed, in which a Gaussian mixture model defined on a regular grid and the explicit log likelihood cost function were utilized.

Since 1998 we have developed an approach for constructing principal manifolds based on a physical analogy with elastic membranes modeled by a system of springs. The idea of using the elastic energy functional for principal manifold construction in the context of neural network methodology was proposed in the mid 1990s (see [13, 17] and the bibliography there). Due to the simple quadratic form of the curvature penalty function, the algorithm proposed for minimization of elastic energy proved to be very computationally effective and allowed parallel implementations. Packages in three programming languages (C++, Java and Delphi) were developed [10, 58, 59] together with graphical user interfaces (VidaExpert [58] and ViMiDa [59]). The elastic map approach led to many practical applications, in particular in data visualization and missing data value recovery. It was applied for visualization of economic and sociological tables [20, 21, 22, 64], to visualization of natural [64] and genetic texts [24, 65, 66], recovering missing values in geophysical time series [6]. Modifications of the algorithm and various adaptive optimization strategies were proposed for modeling molecular surfaces and contour extraction in im-

ages [25]. Recently the elastic maps approach was extended for construction of cubic complexes (objects more general than manifolds) in the framework of topological grammars [26]. The simplest type of grammar (“add a node”, “bisect an edge”) leads to construction of branching principal components (principal trees) that are described in the accompanying paper.

4.1.3 Elastic Functional and Elastic Nets

Let G be a simple undirected graph with set of vertices Y and set of edges E . For $k \geq 2$ a k -star in G is a subgraph with $k + 1$ vertices $y_{0,1,\dots,k} \in Y$ and k edges $\{(y_0, y_i) | i = 1, \dots, k\} \subset E$. Suppose that for each $k \geq 2$, a family S_k of k -stars in G has been selected. Then we define an *elastic graph* as a graph with selected families of k -starts S_k and for which for all $E^i \in E$ and $S_k^j \in S_k$, the corresponding elasticity moduli $\lambda_i > 0$ and $\mu_{kj} > 0$ are defined.

Let $E^{(i)}(0)$, $E^{(i)}(1)$ denote two vertices of the graph edge $E^{(i)}$ and $S_k^{(j)}(0), \dots, S_k^{(j)}(k)$ denote vertices of a k -star $S_k^{(j)}$ (where $S_k^{(j)}(0)$ is the central vertex, to which all other vertices are connected). Let us consider a map $\phi : Y \rightarrow R^m$ which describes an embedding of the graph into a multidimensional space. The *elastic energy of the graph embedding* is defined as

$$U^\phi(G) := U_E^\phi(G) + U_R^\phi(G), \quad (4.4)$$

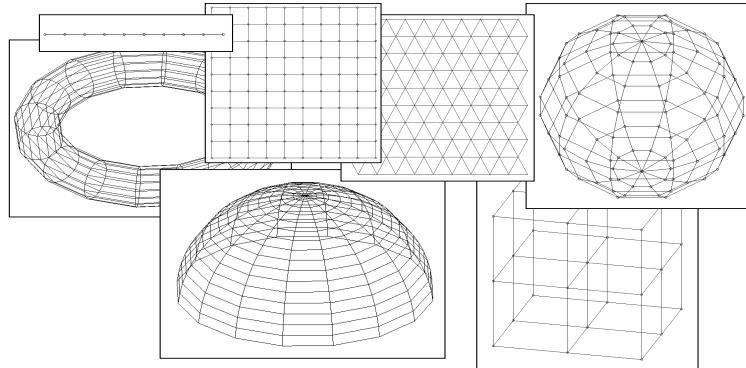
$$U_E^\phi(G) := \sum_{E^{(i)}} \lambda_i \left\| \phi(E^{(i)}(0)) - \phi(E^{(i)}(1)) \right\|^2, \quad (4.5)$$

$$U_R^\phi(G) := \sum_{S_k^{(j)}} \mu_{kj} \left\| \sum_{i=1}^k \phi(S_k^{(j)}(i)) - k\phi(S_k^{(j)}(0)) \right\|^2. \quad (4.6)$$

This general definition of an elastic graph and its energy is used in [26] and in the accompanying paper to define elastic cubic complexes, a new interesting type of principal objects. In this paper we use elastic graphs to construct grid approximations to some regular manifolds. Because of this we limit ourselves to the case $k = 2$ and consider graphs with nodes connected into some regular grids (see Fig. 4.1). In this case we will refer to G as an *elastic net*⁵.

The 2-star is also called a *rib* (see Fig. 4.2). The contribution $\mu_j \left\| \phi(S_2^{(j)}(1)) + \phi(S_2^{(j)}(2)) - 2\phi(S_2^{(j)}(0)) \right\|^2$ (in which one can recognize a discrete approximation of the second derivative squared) of the j th rib to the elastic energy equals zero only if the embedding of the central vertex is the

⁵ To avoid confusion, one should notice that the term elastic net was independently introduced by several groups: in [9] for solving the traveling salesman problem, in [21] in the context of principal manifolds and recently in [30] in the context of regularized regression problem. These three notions have little to do with each other and denote different things.

**Fig. 4.1.** Elastic nets used in practice**Fig. 4.2.** A node, an edge and a rib (2-star).

average of its neighbours. Ribs introduce a simple quadratic penalty term on the non-linearity (in other words, complexity) of graph embedding.

How is the set S_2 chosen for an elastic net? In most of the cases this choice is natural and is dictated by the structure of the net. It is useful to consider another embedding of the graph G in a low-dimensional space, where the graph has its “natural” form. For example, on Fig. 4.1 the regular rectangular grid is embedded into R^2 and S_2 is defined as all triples of adjacent collinear vertices.

We would like to find such a map ϕ that has low elastic energy and simultaneously approximates a set of data points X , where the approximation error is defined as the usual mean-square distance to the graph vertex positions in the data space. For a given map $\phi : Y \rightarrow R^m$ we divide the data set $X = \{x_i\}$ into subsets $\{K^{y_j}\}$, where $y_j \in Y$ is a vertex of the graph G . The set K^{y_j} contains data points for which the node embedding $\phi(y_j)$ is the closest one:

$$K^{y_j} = \{x_i | y_j = \arg \min_{y_k \in Y} \|y_k - x_i\|\}. \quad (4.7)$$

Then the approximation error (“approximation energy”) is defined as

$$U_A^\phi(G, X) = \frac{1}{\sum_{x \in X} w(x)} \sum_{y \in Y} \sum_{x \in K^y} w(x) \|x - \phi(y)\|^2, \quad (4.8)$$

where $w(x) \geq 0$ are the point weights. In the simplest case $w(x) = 1$ but it might be useful to make some points ‘heavier’ or ‘lighter’ in the initial data.

The normalization factor $1/\sum_{x \in X} w(x)$ in (4.8) is needed for the law of large numbers: if X is an i.i.d. sample for a distribution with probability

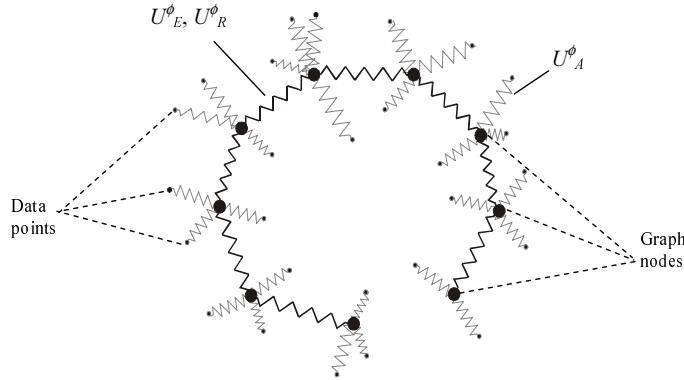


Fig. 4.3. Energy of elastic net

density $\rho(x)$ and finite first two moments then there should exist a limit of $U_A^\phi(G, X)$ for $|X| \rightarrow \infty$. This can be easily proved for $w(x) \equiv 1$ or if $w(x) \geq 0$ is a bounded continuous function of point x . In the limit $|X| \rightarrow \infty$ the energy of approximation is

$$U_A^\phi(G, \rho) = \frac{1}{\mathbf{E}(w)} \int_{x \in V^y} w(x) \rho(x) \|x - \phi(y)\|^2 d^m x, \quad (4.9)$$

where $V^y = \{x \in R^m | y = \arg \min_{z \in Y} \|z - x\|\}$ is the Voronoi cell and $\mathbf{E}(m) = \int w(x) \rho(x) d^m x$ is the expectation.

The optimal map ϕ_{opt} minimizes the total energy function (approximation+elastic energy):

$$U^\phi := U_A^\phi(G, X) + U(G)^\phi. \quad (4.10)$$

To have a visual image of (4.10), one can imagine that every graph node y_j is connected by elastic bonds to its closest data points in K^{y_j} and simultaneously to the adjacent nodes (see Fig. 4.3).

The values λ_i and μ_j are the coefficients of stretching elasticity of every edge $E^{(i)}$ and of bending elasticity of every rib S_2^j . In the simplest case we have

$$\lambda_1 = \lambda_2 = \dots = \lambda_s = \lambda(s), \quad \mu_1 = \mu_2 = \dots = \mu_r = \mu(r),$$

where s and r are the numbers of edges and ribs correspondingly. To obtain $\lambda(s)$ and $\mu(r)$ dependencies we simplify the task and consider the case of a regular, evenly stretched and evenly bent grid. Let us consider a lattice of nodes of “internal” dimension d ($d = 1$ in the case of a polyline, $d = 2$ in case of a rectangular grid, $d = 3$ in the case of a cubical grid and so on). Let the “volume” of the lattice be equal to V . Then the edge length equals $(V/s)^{1/d}$. Having in mind that for typical regular grids $r \approx s$, we can calculate

the smoothing parts of the functional: $U_E \sim \lambda s^{\frac{d-2}{d}}$, $U_R \sim \mu r^{\frac{d-2}{d}}$. Then in the case where we want U_R, U_E to be independent on the grid “resolution”,

$$\lambda = \lambda_0 s^{\frac{2-d}{d}}, \quad \mu = \mu_0 r^{\frac{2-d}{d}}, \quad (4.11)$$

where λ_0, μ_0 are elasticity parameters. This calculation is not applicable, of course, for the general case of any graph. The dimension in this case can not be easily defined and, in practical applications, the λ_i, μ_i are often made different in different parts of a graph according to some adaptation strategy (see below).

The elastic net approximates the cloud of data points and has regular properties. Minimization of the U_A^ϕ term provides approximation, the U_E^ϕ penalizes the total length (or, indirectly, “square”, “volume”, etc.) of the net and U_R^ϕ is a smoothing term, preventing the net from folding and twisting.

Other forms of U_R^ϕ are also used, in particular in the Polygonal algorithm [36]. There exists a simple justification for the quadratic form of U_R^ϕ in (4.4): it is the simplest non-linearity penalty (which corresponds to a simple approximation of the second derivative). All other variants might be more effective in various specific applications but will be more and more close to our choice of U_R^ϕ with increasing grid resolution.

4.2 Optimization of Elastic Nets for Data Approximation

4.2.1 Basic Optimization Algorithm

Finding the globally optimal map ϕ is not an easy problem. In practice we can apply the splitting optimization strategy similar to K -means clustering. The optimization is done iteratively, with two steps at each iteration: 1) Determining $\{K^{y_j}\}$ for a given ϕ and 2) optimizing ϕ for a given $\{K^{y_j}\}$.

In order to perform the map optimization step 2 we derive the system of algebraic linear equations to be solved. Data points are separated in K^{y_j} , $j = 1 \dots p$.

Let us denote

$$\Delta(x, y) = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases},$$

$$\Delta E^{ij} \equiv \Delta(E^{(i)}(0), y^{(j)}) - \Delta(E^{(i)}(1), y^{(j)}),$$

$$\Delta R^{ij} \equiv \Delta(S^{(i)}(2), y^{(j)}) + \Delta(S^{(i)}(1), y^{(j)}) - 2\Delta(S^{(i)}(0), y^{(j)}).$$

That is, $\Delta E^{ij} = 1$ if $y^j = E^{(i)}(0)$, $\Delta E^{ij} = -1$ if $y^j = E^{(i)}(1)$, and $\Delta E^{ij} = 0$ for all other y^j ; $\Delta R^{ij} = 1$ if $y^j = R^{(i)}(1)$ or $y^j = R^{(i)}(2)$, $\Delta R^{ij} = -2$ if

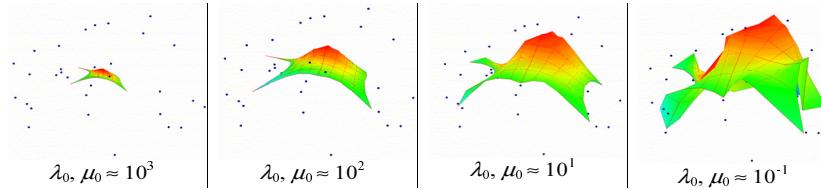


Fig. 4.4. Training elastic net in several epochs (softening)

$y^j = R^{(i)}(0)$, and $\Delta R^{ij} = 0$ for all other y^j . After a short calculation we obtain the system of p linear equations to find new positions of nodes in multidimensional space $\{\phi(y^i), i=1\dots p\}$:

$$\sum_{k=1}^p a_{jk} \phi(y^{(k)}) = \frac{1}{\sum_{x \in D} w(x)} \sum_{x \in K^{y_j}} w(x) x ,$$

where

$$a_{jk} = \frac{n_j \delta_{jk}}{\sum_{x \in D} w(x)} + e_{jk} + r_{jk} , \quad j = 1 \dots p , \quad (4.12)$$

δ_{jk} is Kronecker's δ , and $n_j = \sum_{x \in K^{y_j}} w(x)$, $e_{jk} = \sum_{i=1}^s \lambda_i \Delta E^{ij} \Delta E^{ik}$, $r_{jk} = \sum_{i=1}^r \mu_i \Delta R^{ij} \Delta R^{ik}$. The values of e_{jk} and r_{jk} depend only on the structure of the grid. If the structure does not change then they are constant. Thus only the diagonal elements of the matrix (4.12) depend on the data set. The a matrix has sparse structure for a typical grid used in practice. In practice it is advantageous to calculate directly only non-zero elements of the matrix and this simple procedure was described in [25].

The process of minimization is interrupted if the change in U^ϕ becomes less than a small value ε or after a fixed number of iterations.

As usual, we can only guarantee that the algorithm leads to a local minimum of the functional. Obtaining a solution close to the global minimum can be a non-trivial task, especially in the case where the initial position of the grid is very different from the expected (or unknown) optimal solution. In many practical situations the “softening” strategy can be used to obtain solutions with low energy levels. This strategy starts with “rigid” grids (small length, small bending and large λ, μ coefficients) at the beginning of the learning process and finishes with soft grids (small λ, μ values), Fig. 4.4. Thus, the training goes in several epochs, each epoch with its own grid rigidness. The process of “softening” is one of numerous heuristics that pretend to find the global minimum of the energy U or rather a close configuration.

Nevertheless, for some artificial distributions (like the spiral point distribution, used as a test in many papers on principal curve construction) “softening” starting from any linear configuration of nodes does not lead to the

expected solution. In this case, adaptive strategies, like “growing curve” (an analogue of what was used by Kégl in his polygonal algorithm [36] or “growing surface” can be used to obtain suitable configuration of nodes.

We should mention the problem of good initial approximation for ϕ , used to initialize the algorithm. For principal manifolds, one can start with the plane or the volume of several first principal components calculated for X (as in [3, 16]), but other initializations could be also developed.

Another important remark is the choice of elastic coefficients μ and λ such that they would give good approximation to a hypothetical “ideal” principal manifold. The standard strategies of using a training data subset or cross-validation could be used. For example, we can remove a certain percentage of points from the process of optimization and calculate approximation error on this test set for every “epoch” of manifold softening, thus stopping with the best parameter values. One should notice that despite the fact that we have two parameters, the “bending” elasticity parameter μ is more important for optimizing this “generalization” error, since it determines the curvature of ϕ . The stretching elasticity parameter λ is more important at the initial stage of net optimization since it (together with μ) forces regularity of the node positions, making distances between neighboring nodes almost uniform along the net. For the final stage of the optimization process λ can be even put to zero or some small value.

4.2.2 Missing Data Values

Our geometrical approach to the construction of principal manifolds allows us to deal very naturally with the missing values in the data. In bioinformatics this issue can be very important: in one of the examples below, there are missing values in almost any column and any row of the initial data table. When filling gaps with mean values is not a good solution to the problem, it is better to use principal objects for this purpose (the mean value itself is the 0-dimensional principal object).

The general idea is to be able to project a data point x_i with missing values in some of its coordinates onto the principal object of the whole distribution X . In this case we can not represent x_i as a simple vector anymore. If x_i has p missing values then it is natural to represent it as a linear manifold parallel to the p coordinate axes for which the data values are missing: $x_i = [x_i^0, \dots, t_1, \dots, x_i^k, \dots, t_p, \dots, x_i^m], -\infty \leq t_1, t_2, \dots, t_p \geq \infty$. Then, for a principal manifold y , we can calculate the closest distance (or the intersection point) between two manifolds (the non-linear principal one and the linear one representing x_i) and use the coordinate values of the projected point. Theoretically, the reconstructed values are not unique, when x_i and y intersect in many points or even along lines, which gives several alternative solutions or even continuum for the reconstructed missing values. With increasing data dimension the probability of such intersections decreases.

4.2.3 Adaptive Strategies

The formulation of the method given above allows us to construct different adaptive strategies by playing with a) individual λ_i and μ_j weights; b) the grid connection topology; c) the number of nodes.

This is a way of extending the approach significantly making it suitable for many practical applications.

First of all, let us define a basic operation on the grid, which allows inserting new nodes. Let us denote by \mathbf{N} , \mathbf{S} , \mathbf{R} the sets of all nodes, edges and ribs respectively. Let us denote by $\mathbf{C}(i)$ the set of all nodes which are connected to the i th node by an edge. If one has to insert a new node in the middle of an edge I , connecting two nodes k and l , then the following operations have to be accomplished:

1. Delete from \mathbf{R} those ribs which contain node k or node l ;
2. Delete the edge I from \mathbf{S} ;
3. Put a new node m in \mathbf{N} ;
4. Put in \mathbf{S} two new edges connecting k and m , m and l ;
5. Put in \mathbf{R} new ribs, connecting m , k and all $i \in \mathbf{C}(k)$, and m , l and all $i \in \mathbf{C}(l)$.

At steps 4, 5 one has to assign new weights to the edges and ribs. This choice depends on the task to be solved. If one constructs a “growing” grid, then these weights must be chosen to be the same as they were for the deleted ones. If one constructs a refinement of an already constructed grid, one must choose these weights to be twice bigger than they were for the deleted ones.

The *grow-type strategy* is applicable mainly to grids with planar topology (linear, rectangular, cubic grids). It consists of an iterative determination of those grid parts, which have the largest “load” and doubling the number of nodes in this part of the grid. The load can be defined in different ways. One natural way is to calculate the number of points that are projected onto the nodes. For linear grids the grow-type strategy consists of the following steps:

1. Initializing the grid; it must contain at least two nodes and one edge;
2. Determining the edge which has the largest load, by summing the number of data points (or the sum of their weights) projected to both ends of every edge;
3. Inserting a new node in the middle of the edge, following the operations described above;
4. Optimizing the positions of the nodes.

One stops this process usually when a certain number of nodes in the grid is reached (see, for example, [37]). This number is connected with the total amount of points. In the *elmap* package [10] this is an explicit parameter of the method, allowing a user to implement his own stopping criterion. Because

of this stopping condition the computational complexity is not proportional to the number of data points and, for example, grows like $n^{5/3}$ in the case of the Polygonal Line algorithm. Another form of the stopping condition is when the mean-square error (MSE) does not change more than a small number ε after several insertion/optimization operations.

We should mention here also *growing lump* and *growing flag* strategies used in physical and chemical applications [14, 15]. In the growing lump strategy we add new nodes uniformly at the boundary of the grid using a linear extrapolation of the grid embedding. Then the optimization step follows, and, after that, again the step of growing could be done.

For the principal flag one uses sufficiently regular grids, in which many points are situated on the coordinate lines, planes, etc. First, we build a one-dimensional grid (as a one-dimensional growing lump, for example). Then we add a new coordinate and start growing in new direction by adding lines. After that, we can add the third coordinate, and so on.

The *break*-type adaptive strategy changes individual rib weights in order to adapt the grid to those regions of data space where the “curvature” of data distribution has a break or is very different from the average value. It is particularly useful in applications of principal curves for contour extraction (see [25]). For this purpose the following steps are performed:

1. Collect statistics for the distances from every node i to the mean point of the datapoints that are projected into this node:

$$r_j = \left\| \phi(y_j) - \left(\sum_{x^{(i)} \in K^{y_j}} w_i \right)^{-1} \sum_{x^{(i)} \in K^{y_j}} w_i x^{(i)} \right\|.$$

2. Calculate the mean value and the standard deviation for some power of r : $m = r^\alpha$, $s = \sigma_{r^\alpha}$; where $\alpha > 1$ is a parameter which in our experiments is chosen to be 4.
3. Determine those nodes for which $r_j > m + \beta s$, where $\beta > 0$ is another parameter, equal 2 in our experiments.
4. For every node k determined at the previous step one finds those ribs that have k as their central point and change their weight for $\mu_j^{(new)} = \mu_j^{(old)} \cdot \frac{m}{r_j^\alpha}$.
5. Optimize the node positions.
6. Repeat this process a certain number of times.

The *principal graph* strategy, implemented in the *elmap* package [10] allows us to perform clustering of curvilinear data features along principal curves. Two example applications of this approach are satellite image analysis [2] or hand-written symbol skeletonization [35, 25]. First, notice that the grid we constructed does not have to be a connected graph. The system matrix (6) is not singular if for every connected component of the graph there are

data points that are projected onto one of its nodes. This allows using the same algorithmic kernel to optimize node positions of the unconnected graph. Notice also that if the sets of edges and ribs are empty, then this algorithm acts exactly like the standard K -means clustering algorithm.

To construct a “skeleton” for a two-dimensional point distribution, we apply a variant of local linear principal component analysis first, then connect local components into several connected parts and optimize the node positions afterwards. This procedure is robust and efficient in applications to clustering along curvilinear features and it was implemented as a part of the *elmap* package [10]. The following steps are performed:

1. Make a “grid” from a number of unconnected nodes (sets of edges and ribs are empty at this stage). Optimize the node positions (i.e., do K -means clustering for large k). The number of nodes k is chosen to be a certain proportion of the number of data points. In our experiments we used $k = 5\%$ of the total number of data points.
2. For every node of the grid in position y^i , the local first principal direction is calculated. By local we mean that the principal direction is calculated inside the cluster of datapoints corresponding to the node i . Then this node is substituted by two new nodes in positions $y^{(new1)} = y^i + \alpha s \mathbf{n}$, $y^{(new2)} = y^i - \alpha s \mathbf{n}$, where \mathbf{n} is the unit vector in the principal direction, s is the standard deviation of data points belonging to the node i , α is a parameter, which can be taken to be 1.
3. A collection of edges and ribs is generated, following this simple rule: every node is connected to the node which is closest to this node but not already connected at step 2, and every such connection generates two ribs, consisting of a new edge and one of the edges made at step 2.
4. Weights of the ribs are calculated. A rib is assigned a weight equal to $|\cos(\alpha)|$, where α is an intersection angle of two edges contained in this rib, if $\alpha \geq \frac{\pi}{2}$. Otherwise it is zero (or, equally, the rib is eliminated).
5. The node positions are optimized.

One possible way to improve the resulting graph further is to apply graph simplification rules, analogously to the method in [35]. The idea of this algorithm is close to the k -segments algorithm of Verbeek [57] and, indeed, one possible option is to use k -segment clustering instead of K -means clustering on the first step of the algorithm.

The adaptive strategies: “grow”, “break” and “principal” graphs can be combined and applied one after another. For example, the principal graph strategy can be followed by break-type weight adaptation or by grow-type grid adaptation.

4.3 Elastic Maps

4.3.1 Piecewise Linear Manifolds and Data Projectors

In the process of net optimization we use projection of data into the closest node. This allows us to improve the speed of the data projection step without loosing too much detail when the grid resolution is good enough. The effect of an estimation bias, connected with this type of projection, was observed in [34]. In our approach the bias is indirectly reduced by utilizing the U_E^ϕ term that makes the grid almost isometric (having the same form, the grid will have lesser energy with equal edge lengths). For presentation of data points or for data compression, other projectors can be applied.

In applications we utilize a piece-wise linear projection onto the manifold. A natural way to do this is to introduce a set of simplexes on the grid (line segments for one-dimensional grids, triangles for two-dimensional grids, and tetrahedra for the 3D grids). Then one performs orthogonal projection onto this set. In order to avoid calculation of all distances to all simplexes, one can apply a simplified version of the projector: find the closest node of the grid and then consider only those simplexes that contain this node. This type of projection is used in the examples (Fig. 4.7-4.9).

Since the elastic net has a penalty on its length (and, for higher dimensions, indirectly, area, volume), the result of the optimization procedure is a bounded manifold, embedded in the cloud of data points. Because of this, if the penalty coefficient is big, many points can have projection on the boundary of the manifold. This can be undesirable, for example, in data visualization applications. To avoid this effect, a linear extrapolation of the bounded rectangular manifold (extending it by continuity in all directions) is used as a post-processing step.

Using piece-wise linear interpolation between nodes and linear extrapolation of the manifold is the simplest (hence computationally effective) possibility. It is equivalent to construction of piece-wise linear manifold. Other interpolation and extrapolation methods are also applicable, for example the use of Carleman's formula [1, 14, 15, 18, 19, 6] or using Lagrange polynomials [47], although they are more computationally expensive.

4.3.2 Iterative Data Approximation

An important limitation of many approaches that use grid approximations is that in practice it is feasible to construct only low-dimensional grids (no more than three-dimensional).

This limitation can be overcome in the additive iterative manner. After low-dimensional manifold ϕ is constructed, for every point x , its projection $P^\phi(x)$ onto the manifold is subtracted and new low-dimensional manifold is computed for the set of residues $\{x_i - P^\phi(x_i)\}$. Thus a dataset can be approximated by a number of low-dimensional objects, each of them approximating

the residues obtained from the approximation by the previous one. The norm of the residues at each iteration becomes smaller but is not guaranteed to become zero after a finite number of iterations, as in the case of linear manifolds.

4.4 Principal Manifold as Elastic Membrane

Let us discuss in more detail the central idea of our approach: using the metaphor of elastic membranes in the principal manifold construction algorithm. The system represented in Fig. 4.3 can be modeled as an elastic membrane with external forces applied to the nodes. In this section we consider the question of equivalence between our spring network system and realistic physical systems (evidently, we make comparison in 3D).

Spring meshes are widely used to create physical models of elastic media (for example, [4]). The advantages in comparison with the continuous approaches like the Finite Elements Method (FEM), are evident: computational speed, flexibility, the possibility to solve the inverse elasticity problem easily [56].

Modeling elastic media by spring networks has a number of applications in computer graphics, where, for example, there is a need to create realistic models of soft tissues (human skin, as an example). In [56] it was shown that it is not generally possible to model elastic behavior of a membrane using spring meshes with simple scalar springs. In [61] the authors introduced complex system of penalizing terms to take into account angles between scalar springs as well as shear elasticity terms. This improved the results of modeling and has found applications in subdivision surface design.

In a recent paper [27] a simple but important fact was noticed: every system of elastic finite elements could be represented by a system of springs, if we allow some springs to have negative elasticity coefficients. The energy of a k -star S_k in R^m with y_0 in the center and k endpoints $y_{1,\dots,k}$ is $u_{S_k} = \mu_{S_k}(\sum_{i=1}^k y_i - ky_0)^2$, or, in the spring representation, $u_{S_k} = k\mu_{S_k} \sum_{i=1}^k (y_i - y_0)^2 - \mu_{S_k} \sum_{i>j} (y_i - y_j)^2$. Here we have k positive springs with coefficients $k\mu_{S_k}$ and $k(k-1)/2$ negative springs with coefficients $-\mu_{S_k}$.

Let us slightly reformulate our problem to make it closer to the standard notations in the elasticity theory. We introduce the $m \times p$ -dimensional vector of displacements, stacking all coordinates for every node:

$$u = \{u_1^{(1)}; u_2^{(1)}; \dots; u_m^{(1)}; \dots; u_1^{(p)}; u_2^{(p)}; \dots; u_m^{(p)}\}^T,$$

where m is dimension, p is the number of nodes, $u_i^{(k)}$ is the i th component of the k th node displacement. The absolute positions of nodes are $y^{(k)} = \tilde{y}^{(k)} + u^{(k)}$, where $\tilde{y}^{(k)}$ are equilibrium (relaxed) positions. Then our minimization problem can be stated in the following generalized form:

$$u^T E u + D(u; x) \rightarrow \min, \quad (4.13)$$

where E is a symmetric $(m \times p) \times (m \times p)$ element stiffness matrix. This matrix reflects the elastic properties of the spring network and has the following properties: 1) it is sparse; 2) it is invariant with respect to translations of the whole system (as a result, for any band of m consecutive rows corresponding to a given node k , the sum of the $mm \times m$ off-diagonal blocks should always be equal to the corresponding diagonal block taken with the opposite sign). The $D(u; x)$ term describes how well the set of data x is approximated by the spring network with the node displacement vector u . It can be interpreted as the energy of external forces applied to the nodes of the system. To minimize (4.13) we solve the problem of finding equilibrium between the elastic internal forces of the system (defined by E) and the external forces:

$$Eu = f, \quad f = -\frac{1}{2} \frac{\partial}{\partial u} D(u; x). \quad (4.14)$$

The matrix E is assembled in a similar manner to that described in Subsec. 4.2.1. There is one important point: the springs have zero rest lengths, it means that equilibrium node positions are all in zero: $\tilde{y}^{(k)} = 0$, $k = 1..p$. The system behavior then can be described as “super-elastic”. From the point of view of data analysis it means that we do not impose any pre-defined shape on the data cloud structure.

For $D(u; x)$ we use the usual mean square distance measure, see (4.8): $D(u; x) = U_A^\phi$. The force applied to the j th node equals

$$f_j = \frac{n^{(j)}}{N} (\tilde{x}^{(j)} - u^{(j)}) , \quad (4.15)$$

where

$$\tilde{x}^{(j)} = \frac{\sum_{x^{(i)} \in K^{y_j}} w_i x^{(i)}}{n^{(j)}}, \quad n^{(j)} = \sum_{x^{(i)} \in K^{y_j}} w_i, \quad N = \sum_{x^{(i)}} w_i .$$

It is proportional to the vector connecting the j th node and the weighted average $\tilde{x}^{(j)}$ of the data points in K^{y_j} (i.e., the average of the points that surround the j th node: see (4.7) for definition of K^{y_j}). The proportionality factor is simply the relative size of K^{y_j} . The linear structure of (4.15) allows us to move u in the left part of the equation (4.15). Thus the problem is linear.

Now let us show how we can benefit from the definition (4.14) of the problem. First, we can introduce a pre-defined equilibrium shape of the manifold: this initial shape will be elastically deformed to fit the data. This approach corresponds to introducing a model into the data. Secondly, one can try to change the form (4.15) of the external forces applied to the system. In this way one can utilize other, more sophisticated approximation measures: for example, taking outliers into account.

Third, in three-dimensional applications one can benefit from existing solvers for finding equilibrium forms of elastic membranes. For multidimen-

sional data point distributions one has to adapt them, but this adaptation is mostly formal.

Finally, there is a possibility of a hybrid approach: first, we utilize a “super-elastic” energy functional to find the initial approximation. Then we “fix” the result and define it as the equilibrium. Finally, we utilize a physical elastic functional to find elastic deformation of the equilibrium form to fit the data.

4.5 Method Implementation

For different applications we made several implementations of the *elmap* algorithm in the C++, Java and Delphi programming languages. All of them are available on-line [10, 58, 59] or from the authors by request.

Two graphical user interfaces for constructing elastic maps have also been implemented. The first one is the VidaExpert stand-alone application [58] which provides the possibility not only to construct elastic maps of various topologies but also to utilize linear and non-linear principal manifolds to visualize the results of other classical methods for data analysis (clustering, classification, etc.) VidaExpert has integrated 3D-viewer which allows data manipulations in interactive manner.

A Java implementation was also created [59], it was used to create a Java-applet *ViMiDa* (VIualization of MIcroarray DAta). In the Bioinformatics Laboratory of Institut Curie (Paris) it is integrated with other high-throughput data analysis pipelines and serves as a data visualization user front-end (“one-click” solution). ViMiDa allows remote datasets to be downloaded and their visualization using linear or non-linear (elastic maps) principal manifolds.

4.6 Examples

4.6.1 Test Examples

In Fig. 4.5 we present two examples of 2D-datasets provided by Kégl⁶.

The first dataset called *spiral* is one of the standard ways in the principal curve literature to show that one’s approach has better performance than the initial algorithm provided by Hastie and Stuetzle. As we have already mentioned, this is a bad case for optimization strategies, which start from a linear distribution of nodes and try to optimize all the nodes together in one loop. But the adaptive “growing curve” strategy, though being by order of magnitude slower than “softening”, finds the solution quite stably, with the exception of the region in the neighborhood of zero, where the spiral has very different (compared to the average) curvature.

⁶ <http://www.iro.umontreal.ca/~Kegl/research/pcurves/implementations/Samples/>

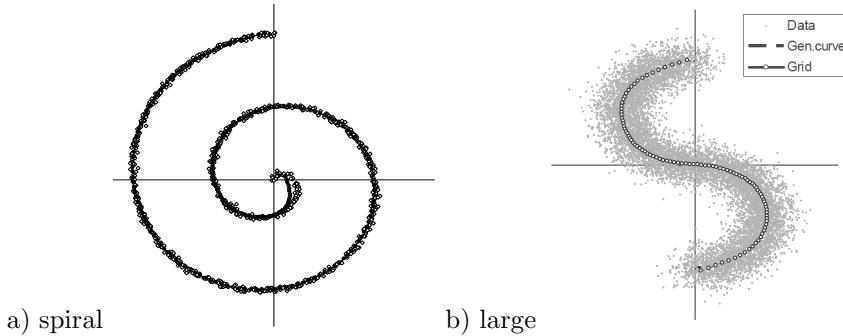


Fig. 4.5. Two-dimensional examples of principal curves construction

The second dataset, called *large* presents a simple situation, despite the fact that it has comparatively large sample size (10000 points). The nature of this simplicity lies in the fact that the initial first principal component based approximation is already effective; the distribution is in fact *quasilinear*, since the principal curve can be unambiguously orthogonally projected onto a line. In Fig. 4.5b it is shown that the generating curve, which was used to produce this dataset, has been discovered almost perfectly and in a very short time.

Some other examples, in particular of application of the *elastic graph* strategy are provided in [25].

4.6.2 Modeling Molecular Surfaces

A molecular surface defines an effective region of space which is occupied by a molecule. For example, the Van-der-Waals molecular surface is formed by surrounding every atom in the molecule by a sphere of radius equal to the characteristic radius of the Van-der-Waals force. After all the interior points are eliminated, this forms a complicated non-smooth surface in 3D. In practice, this surface is sampled by a finite number of points.

Using principal manifolds methodology, we constructed a smooth approximation of such molecular surface for a small piece of a DNA molecule (several nucleotides long). For this we used slightly modified Rasmol source code [49] (available from the authors by request).

First, we have made an approximation of this dataset by a 1D principal curve. Interestingly, this curve followed the backbone of the molecule, forming a spiral (see Fig. 4.6).

Second, we approximated the molecular surface by a 2D manifold. The topology of the surface is expected to be spherical, so we applied spherical topology of the elastic net for fitting. We should notice that since it is impossible to make the lengths of all edges equal for the sphere-like grid, corrections were performed for edge and rib weights during the grid initialization (shorter edges are given with larger weights proportionally and the same for the ribs).

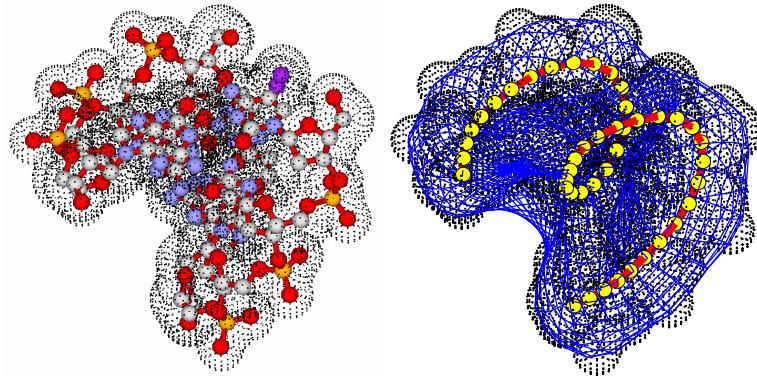


Fig. 4.6. Approximating molecular surface of a simple fragment of DNA by 1D- and 2D-principal manifolds. On the left the molecule with molecular surface is shown, on the right the result of the approximation is shown. The small points are the points on the molecular surface, the big spiral is the 1D principal curve and the mesh is the 2D principal spherical manifold.

As a result one gets a smooth principal manifold with spherical topology, approximating a rather complicated set of points. The result of applying principal manifold construction using the *elmap* [10] package is shown in Fig. 4.6. The manifold allows us to introduce a global spherical coordinate system on the molecular surface. The advantage of this method is its ability to deal not only with star-like shapes as the spherical harmonic functions approach does (see, for example, [5]) but also to model complex forms with cavities as well as non-spherical forms.

4.6.3 Visualization of Microarray Data

DNA microarray data is a rich source of information for molecular biology (for a recent overview, see [39]). This technology found numerous applications in understanding various biological processes including cancer. It allows to screen simultaneously the expression of all genes in a cell exposed to some specific conditions (for example, stress, cancer, normal conditions). Obtaining a sufficient number of observations (chips), one can construct a table of “samples vs genes”, containing logarithms of the expression levels of, typically several thousands (n) of genes, in typically several tens (m) of samples.

The table can be represented as a set of points in two different spaces. Let us call them the “gene space” and the “sample space”. In the gene space every point (vector of dimension m) is a gene, characterized by its expression in m samples. In the sample space every point is a sample (vector of dimension n), characterized by the expression profile of n genes.

One of the basic features of microarray datasets is the fact that $n \gg m$. This creates certain problems when one constructs a classifier in the “sample

space": for example, for standard Linear Discriminant Analysis, too many equivalent solutions for the problem are possible. Thus, some regularization of the classification problem is required (see, for example, [30]) and PCA-like techniques could be also used for such a purpose.

Here we provide results on the visualization of three microarray datasets in the "sample space" using elastic maps. We demonstrate that the two-dimensional principal manifolds outperform the corresponding two-dimensional principal planes, in terms of representation of big and small distances in the initial multidimensional space and visualizing class information.

Datasets used

For the participants of the international workshop "Principal manifolds-2006" which took place in Leicester, UK, in August of 2006, it was proposed to analyze three datasets containing published results of microarray technology application. These datasets can be downloaded from the workshop web-page [45].

Dataset I. Breast cancer microarrays. This dataset was published in [60] and it contains 286 samples and expression values of 17816 genes. We processed the clinical information provided with this dataset and constructed three *ab initio* sample classifications:

GROUP: non-agressive (A) vs agressive (B) cancer;

ER: estrogen-receptor positive (ER+) vs estrogen-receptor negative (ER-) tumors;

TYPE: sub-typing of breast cancer (lumA, lumB, normal, errb2, basal and unclassified), based on the so-called Sorlie breast cancer molecular portraits (see [44]).

Dataset II. Bladder cancer microarrays. This dataset published in [8] contains 40 samples of bladder tumors. There are two *ab initio* sample classifications:

TYPE: clinical classification of tumors (T1, T2+ anf Ta);

GRADE: tumor grade (2,3,4).

Dataset III. Collection of microarrays for normal human tissues. This dataset published in [51] contains gene expression values in 103 samples of normal human tissues. The only *ab initio* sample classification corresponds to the tissue type from which the sample was taken.

Microarray datasets are inevitably incomplete: for example, there is a number of gaps (not reliably measured values) in Dataset III. Datasets I and II do not contain missing values (in Dataset I the missing values are recovered with use of some standard normalization method and in the Dataset II all gaps are filtered out). Dataset III is filtered in such a way that any row (gene) has at least 75% complete values of expression and every column has at least 80% of complete values. It is interesting to mention that missing values in Dataset III are distributed in such a way that almost every column and row contains a missing value. This means that missing value recovery is absolutely necessary for its analysis.

The original expression tables were processed to select 3000 genes with the most variable expression between samples. Thus the sample space has approximately equal dimensions (3000) for all three datasets. For every gene, the expression values were converted to z -values, i.e. every value was centered and divided by its standard deviation.

Principal manifolds for three datasets were approximated by the elastic map algorithm implemented in Java, the package *VDAOEngine* [59] with a 2D rectangular node grid using the softening optimization strategy. The elastic nets were trained in two epochs: first one with $\lambda_0 = 0.1$ and $\mu_0 = 250$ and the second epoch with $\lambda_0 = 0.01$ and $\mu_0 = 30$. No optimization for the final value of μ_0 was done. The resolution of grids was selected 15x15 for Dataset I and Dataset III and 10x10 for Dataset II. For post-processing we used linear extrapolation of the manifold, adding 5 node layers at each border of the manifold. An average time for elastic net computation using an ordinary PC was 1-5 minutes depending on the number of nodes in the net.

For data visualization and analysis we projected data points into their closest points on the manifold obtained by piece-wise linear interpolation between the nodes. The results are presented on Fig. 4.7-4.9. We show also the configuration of nodes projected into the three-dimensional principal linear manifold. This is done only for visualization of the fact that the elastic map is non-linear and usually better approximates the distribution of points in comparison with the linear principal plane. But one should not forget that the non-linear manifold is embedded in all dimensions of the high-dimensional space, not being limited to the space of the first principal components.

Mean-square distance error

By their construction, principal manifolds provide better approximation of data than the less flexible linear PCA. In the Table 4.1 we provide results of comparing the mean square distance error (MSE) of the datasets' approximation by a two-dimensional non-linear elastic map and linear manifolds of various dimensions (1-5-dimensional and 10-dimensional) constructed with using standard PCA. In the case of elastic maps we calculate the distance of the projection to the closest point of the manifold as it is described in the section 4.3.1. The conclusion is that the two-dimensional principal manifolds outperform even four-dimensional linear PCA manifolds in their accuracy of data approximation.

Natural PCA and distance mapping

Our second analysis answers the question of how successfully the structure of distances in the initial highly multidimensional space is reconstructed after projection into the low-dimensional space of the internal co-ordinates of the principal manifold. A possible indicator is the correlation coefficient between all pair-wise distances:

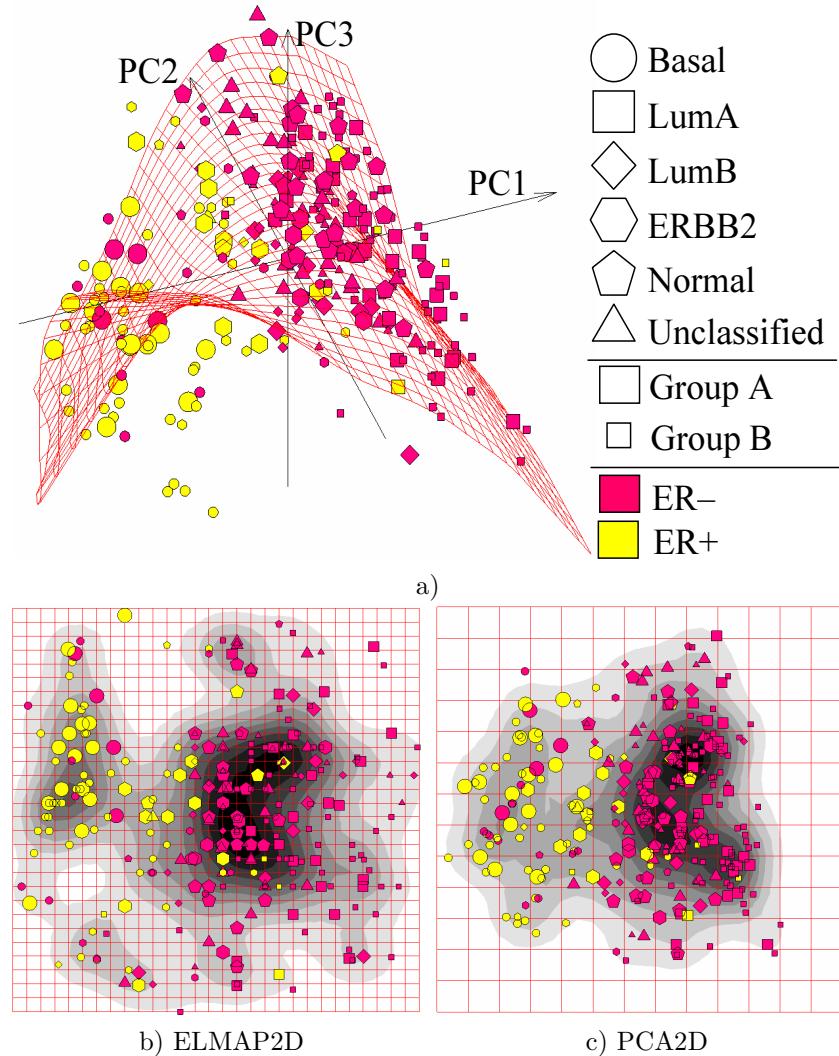


Fig. 4.7. Visualization of Dataset I (breast cancer dataset) using elastic maps. *Ab initio* classifications are shown using points size (ER), shape (GROUP) and color (TYPE). a) configuration of nodes in the three-dimensional principal linear manifold. One clear feature is that the dataset is curved such that it can not be mapped adequately on a two-dimensional principal plane. b) the distribution of points in the internal non-linear manifold coordinates (ELMAP2D) is shown together with an estimation of the two-dimensional density of points. c) the same as b) but for the linear two-dimensional manifold (PCA2D). One can notice that the “basal” breast cancer subtype is visualized more adequately with ELMAP2D and some features of the distribution become better resolved in comparison to PCA2D.

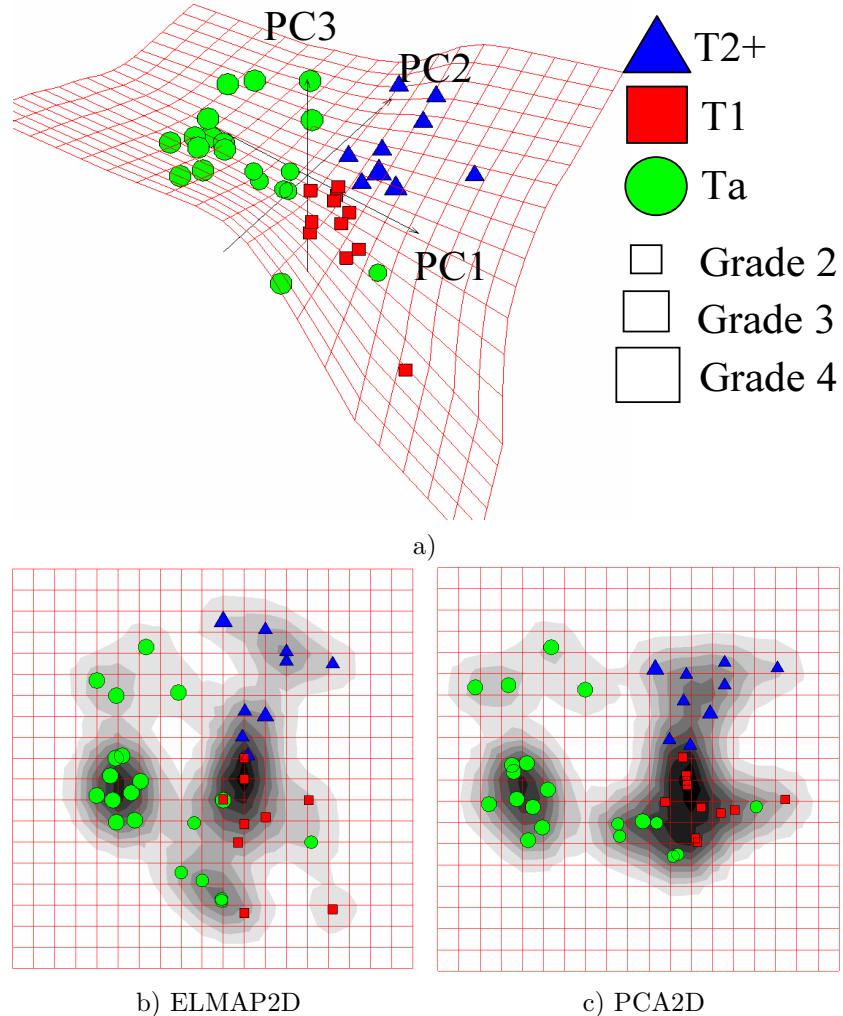
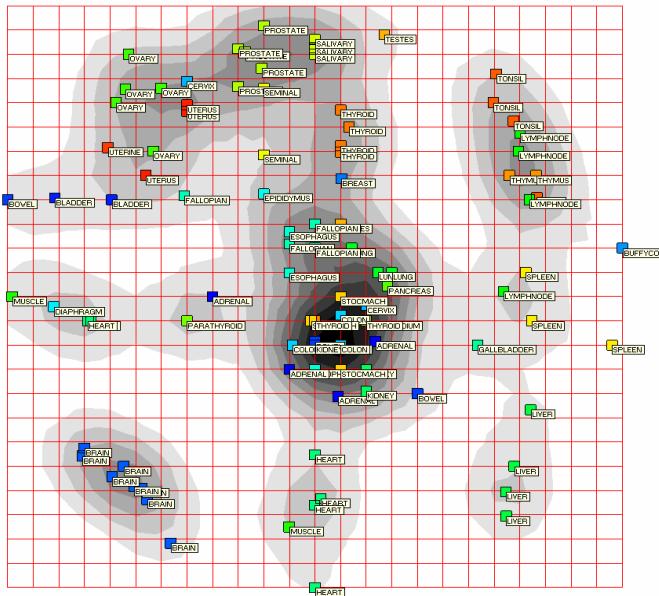
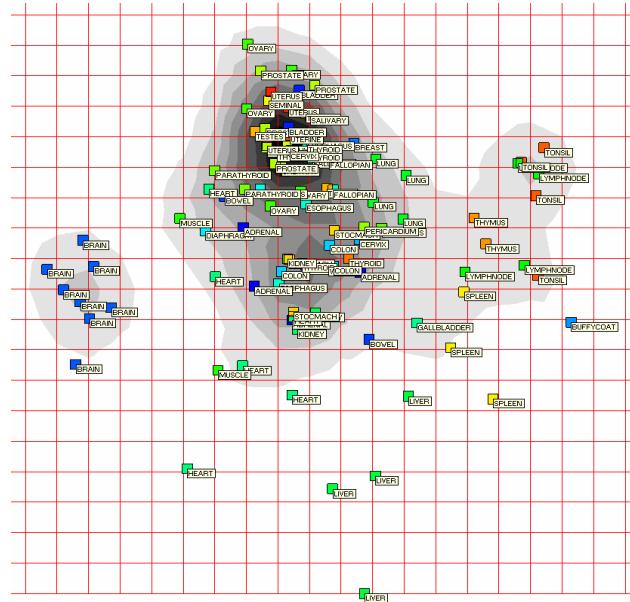


Fig. 4.8. Visualization of Dataset II (bladder cancer dataset) using elastic maps. *Ab initio* classifications are shown using points size (GRADE) and color and shape (TYPE). a) configuration of nodes in the three-dimensional principal linear manifold. b) distribution of points in the internal manifold coordinates is shown together with an estimation of the two-dimensional density of points. From all three datasets, Dataset II is “the least dimensional”, and the configurations of points are quite similar, although ELMAP2D reveals some finer point grouping.



a) ELMAP2D



b) PCA2D

Fig. 4.9. Visualization of Dataset III (normal tissue samples) using elastic maps (a) and two-dimensional PCA (b). Tissue names are shown using point labels. The ELMAP2D image clearly gives a better “resolved” structure of classes, which is confirmed by quantitative measures of “class compactness” (see the text).

Table 4.1. Mean square distance error (MSE) of data approximation by two-dimensional elastic maps (ELMAP2D) and linear PCA manifolds of various dimensions (PC1-PC5, PC10). For linear principal components, the cumulative explained variation is also shown.

Dataset	ELMAP2D	PC1	PC2	PC3	PC4	PC5	PC10
Breast cancer MSE	48.1	52.6	50.7	49.3	48.4	47.6	45.3
Variation explained	-	7.9%	14.3%	19.0%	22.0%	24.6%	31.5%
Bladder cancer MSE	40.6	48.7	45.4	42.8	40.7	39.2	33.0
Variation explained	-	21.0%	31.4%	38.9%	44.8%	48.8%	63.8%
Normal tissues MSE	36.9	48.8	45.5	42.3	40.1	38.5	32.4
Variation explained	-	10.7%	19.1%	26.0%	30.3%	32.2%	40.7%

$$r = \text{corr}(d_{ij}, \hat{d}_{ij}), \quad (4.16)$$

where d_{ij} and \hat{d}_{ij} are the pair-wise distances in the original high-dimensional space and the distances in the “reduced” space after projection onto a low-dimensional object (this object can be principal manifold, linear or non-linear). However, not all of the distances d_{ij} are independent, which creates a certain bias in the estimation of the correlation. We propose a solution for reducing this bias by selecting a subset of “representative” independent distances between the points. This is done by a simple procedure which we call Natural PCA (NatPCA) that is described below. Let M be a finite set of points and $S \in M$ be a subset. We define a distance from a point $i \in M$ to the set S as

$$\text{dist}(i, S) = \min\{d_{ij}, j \in S\}. \quad (4.17)$$

Let us suppose that a point i has the closest point j in S . If there are several closest points, we choose one randomly. We will call *natural principal components* $m - 1$ (m is the number of points) pairs of points $\{i, j\} \in M \times M$ that are selected by the following algorithm:

0. Let S be an empty set.
1. The first component is a pair of the most distant points $\{i_m, j_m\} = \arg \sup_{ij} d_{ij}$. We put i_m and j_m in S .
2. Among all points which are not in S we select a point k_m which is the most distant to S :

$$k_m = \arg \sup_j \{\text{dist}(j, S)\}. \quad (4.18)$$

3. We define next the “natural” component as a pair $\{k_m, p_m\}$, where $p_m \in S$ is the point in S , closest to k_m . We add k_m to S .
4. We repeat steps 2-3 until all points are in S .

As a result we have a set NatPCA of $m - 1$ pairs of points with decreasing distances between them. These distances are independent from each other and

represent all scales in the distribution of data (its diameter, “second” diameter and so on). In other words, the first “natural” component is defined by the most distant points, the second component includes the most distant point from the first pair, and so on. We call this set *natural principal components* because a) for big datasets in vector spaces the vectors defined by the first pairs are usually located close to principal components given that there are no outliers in the data⁷; b) this analysis can be made in a finite metric space, using only a distance matrix as input. NatPCA forms a tree connecting points in the manner which is opposite to neighbour-joining hierarchical clustering.

Here we use NatPCA to define a subset $\text{NatPCA} \in \{d_{ij}\}$ of independent distances in the initial high-dimensional space, which is used to measure an adequacy of distance representations in the reduced space:

$$r = \text{corr}(d_{ij}, \hat{d}_{ij}), \{i, j\} \in \text{NatPCA}. \quad (4.19)$$

In the Table 4.2 we compare the results of calculating r using Pearson and Spearman rank correlation coefficients, r calculated for the two-dimensional principal manifold and for linear manifolds of various dimensions constructed with the use of standard PCA. One clear conclusion is that the two-dimensional elastic maps almost always outperform the linear manifolds of the same dimension, being close in reproducing the distance structure to three-dimensional linear manifolds. This is rather interesting since the principal manifolds are not constructed to solve the problem of distance structure approximation explicitly (unlike Multidimensional Scaling). The results also hold in the case of using all pair-wise distances without the application of NatPCA (data not shown), although NatPCA selection gives less biased correlation values.

Table 4.2. Quality of reproducing distances after dimension reduction by two-dimensional elastic maps (ELMAP2D) and linear PCA manifolds of various dimensions (PC1-PC5, PC10).

Dataset/method	ELMAP2D	PC1	PC2	PC3	PC4	PC5	PC10
Breast cancer/Pearson	0.60	0.40	0.52	0.61	0.65	0.69	0.75
Breast cancer/Spearman	0.40	0.19	0.32	0.36	0.42	0.49	0.56
Bladder cancer/Pearson	0.87	0.82	0.84	0.88	0.89	0.91	0.96
Bladder cancer/Spearman	0.68	0.57	0.60	0.70	0.70	0.75	0.90
Normal tissues/Pearson	0.80	0.68	0.78	0.82	0.86	0.87	0.95
Normal tissues/Spearman	0.68	0.56	0.69	0.79	0.84	0.86	0.94

⁷ By its construction, NatPCA is sensitive to presence of outliers in the data. This problem can be solved by using pre-clustering (adaptive coding) of data with K -means or K -medoids algorithms with relatively big K and performing NatPCA on the centroids (or medoids).

Local neighbourhood preservation

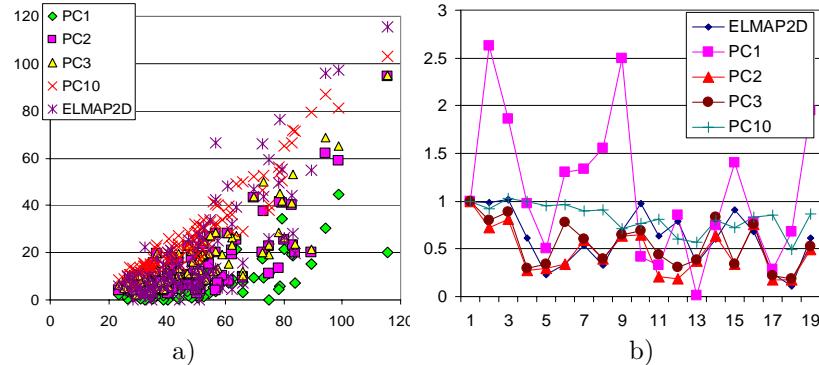


Fig. 4.10. a) Sheppard plot (\hat{d}_{ij} vs d_{ij}) for a subset of pair-wise distances selected by NatPCA, calculated for the normal tissue dataset where \hat{d}_{ij} are calculated for dimension reduction by elastic maps (ELMAP2D), and linear principal components (PC1-PC5, PC10); distances \hat{d}_{ij} calculated in the internal coordinates of the elastic map (ELMAP2D) were scaled such that $\hat{d}_{ij}^{\text{ELMAP2D}}/d_{ij} = 1$ for the first (largest) distance; \hat{d}_{ij} for linear manifolds were taken without re-scaling; b) \hat{d}_{ij}/d_{ij} ratios for the first 20 *NatPCA* components; the ratios are scaled such that $\hat{d}_{ij}/d_{ij} = 1$ for the first (largest) distance.

Dimension reduction using projections onto low-dimensional linear manifolds leads to distance contraction, i.e. all the pair-wise distances become smaller than they were in the initial multidimensional space. The structure of big distances is reproduced relatively well since linear principal components are aligned along the directions of the biggest variation. This is demonstrated on Fig. 4.10, where scatters of $\{d_{ij}, \hat{d}_{ij}\}$ (so-called Sheppard plots) and scaled ratios \hat{d}_{ij}/d_{ij} are shown for distances from NatPCA set. One can expect that k -dimensional PCA is able to reproduce correctly the first k “natural” principal components, and more if the dimension of the data is less than k .

However, preservation of the local structure of small distances is generally not guaranteed, hence it is interesting to measure it. We used a simple test which consists of calculating a set of k neighbors in the “reduced” space for every point, and counting how many of them are also point neighbors in the initial high-dimensional space. The measure of local distance structure preservation is the average ratio of the intersection size of these two sets (k neighbors in the “reduced” and the initial space) over k . The unit value would correspond to perfect neighborhood preservation, while a value close to zero corresponds to strong distortions or a situation where distant regions of dataspace are projected in the same region of the low-dimensional manifold. The results are provided in Table 4.3. The elastic maps perform significantly

better than the two-dimensional linear principal manifold in two cases and as well as two-dimensional PCA for Dataset II. To evaluate the significance of this improvement, we made a random test to calculate the local neighborhood preservation measure obtained by a pure random selection of points.

Table 4.3. Local neighbourhood preservation (defined as an average relative number of neighbours, “preserved” after dimension reduction) for two-dimensional elastic maps (ELMAP2D) and linear PCA manifolds of various dimensions (PC1-PC5,PC10). The last column (RANDOM) corresponds to the value calculated for a random choice of neighbours (10000 permutations are made) which is shown together with its standard deviation.

Dataset	ELMAP2D	PC1	PC2	PC3	PC4	PC5	PC10	RANDOM
Breast cancer (k=10)	0.26	0.13	0.20	0.28	0.31	0.38	0.47	0.04 ± 0.06
Bladder cancer (k=5)	0.53	0.34	0.53	0.61	0.64	0.70	0.80	0.12 ± 0.14
Normal tissues (k=5)	0.49	0.23	0.33	0.43	0.50	0.54	0.69	0.05 ± 0.09

Class compactness

Now let us suppose that the points are marked by labels, thus defining classes of points. In the process of dimension reduction, various distortions of the data space are possible. Distant points can be closely projected, distorting the picture of inter and intra-class distances existing in the multidimensional space. In this section we analyze how principal manifolds preserve class distance relations in comparison with linear principal manifolds of various dimensions using a simple test for “class compactness”.

For a class C , let us define “class compactness” as an average of a proportion of the points of class C among k neighbors of the point. This average is calculated only over the points from the class C .

In Table 4.4, we provide values of “class compactness” for all datasets with different *ab initio* classes defined. One can see that there are many examples when two-dimensional principal manifolds perform as well as four- or five-dimensional linear manifolds in putting together the points from the same class. It works particularly well for the collection of normal tissues. There are cases when neither linear nor non-linear low-dimensional manifolds could put together points of the same class and there are a few examples when linear manifolds perform better. In the latter cases (Breast cancer’s A, B, lumA, lumB and “unclassified” classes, bladder cancer T1 class), almost all class compactness values are close to the estimated random values which means that these classes have big intra-class dispersions or are poorly separated from the others. In this case the value of class compactness becomes unstable (look, for example, at the classes A and B of the breast cancer dataset) and depends on random factors which can not be taken into account in this framework.

The closer class compactness is to unity, the easier one can construct a decision function separating this class from the others. However, in the high-dimensional space, due to many degrees of freedom, the “class compactness” might be compromised and become better after appropriate dimension reduction. In Table 4.4 one can find examples when dimension reduction gives better class compactness in comparison with that calculated in the initial space (breast cancer basal subtype, bladder cancer Grade 2 and T1, T2+ classes). It means that sample classifiers can be regularized by dimension reduction using PCA-like methods.

There are several particular cases (breast cancer basal subtype, bladder cancer T2+, Grade 2 subtypes) when non-linear manifolds give better class compactness than both the multidimensional space and linear principal manifolds of the same dimension. In these cases we can conclude that the dataset in the regions of these classes is naturally “curved” (look, for example, at Fig. 4.7) and the application of non-linear techniques for classification regularization is an appropriate solution.

We can conclude that non-linear principal manifolds provide systematically better or equal resolution of class separation in comparison with linear manifolds of the same dimension. They perform particularly well when there are many small and relatively compact heterogeneous classes (as in the case of normal tissue collection).

4.7 Discussion

Principal Components Analysis already celebrated its 100th anniversary [43]. Nowadays linear principal manifolds are widely used for dimension reduction, noise filtering and data visualization. Recently, methods for constructing non-linear principal manifolds were proposed, including our *elastic maps* approach which is based on a physical analogy with elastic membranes. We have developed a general geometric framework for constructing “principal objects” of various dimensions and topologies with the simplest quadratic form of the smoothness penalty, which allows very effective parallel implementations.

Following the metaphor of elasticity (Fig. 4.3), we introduced two smoothness penalty terms, which are quadratic at the node optimization step. As one can see from (4.4) they are similar to the sum of squared grid approximations of the first and second derivatives⁸. The U_E^ϕ term penalizes the total length (or area, volume) of the principal manifold and, indirectly, makes the grid regular by penalizing non-equidistant distribution of nodes along the grid. The

⁸ The differences should be divided by node-to-node distances in order to be true derivative approximations, but in this case the quadratic structure of the term would be violated. We suppose that the grid is regular with almost equal node-to-node distances, then the dependence of coefficients λ_i, μ_j on the total number of nodes contains this factor.

Table 4.4. “Class compactness” before (ND) and after dimension reduction with use of two-dimensional elastic maps (ElMap2D) and linear principal manifolds (PC1-PC5,PC10). The column “Random” shows a random test when k points are selected by chance (10000 permutations have been made). For normal tissues the number of samples in the class is shown in parentheses (only classes with > 2 samples are shown); “unclas.” stands for unclassified samples.

U_R^ϕ term is a smoothing factor, it penalizes the nonlinearity of the ribs after their embedding into the data space.

Such quadratic penalty allows using standard minimization of quadratic functionals (i.e., solving a system of linear algebraic equations with a sparse matrix), which is considerably more computationally effective than gradient optimization of more complicated cost functions, like the one introduced by Kégl. Moreover, the choice of (4.4) is the simplest smoothness penalty, which is universally applicable.

Minimization of a positive definite quadratic functional can be provided by the sequential one-dimensional minimization for every space coordinate (cyclic). If for a set of coordinates $\{x_i\}_{i \in J}$ terms $x_i x_j$ ($i, j \in J, i \neq j$) do not present in the functional, then for these coordinates the functional can be minimized independently. The quadratic functional we formulate has a sparse structure, it gives us the possibility to use parallel minimization that is expected to be particularly effective in the case of multidimensional data. The application of high-throughput techniques in molecular biology such as DNA microarrays provides such data in large quantities. In the case when $n \gg m$ (the space dimension is much bigger than the number of samples) many existing algorithms for computation of principal manifolds (as GTM [3]) change their performance. In particular, computing the closest nodes with the formula (4.7) is no longer the limiting step with most computations being required for finding the new node positions at each iteration.

Our approach is characterized by a universal and flexible way to describe the connection of nodes in the grid. The same algorithm, given an initial definition of the grid, provides construction of principal manifolds with various dimensions and topologies. It is implemented together with many supporting algorithms (interpolation/extrapolation formulas, adaptation strategies, and so on) in several programming languages [10, 58, 59].

One important application of principal manifolds is dimension reduction and data visualization. In this field they compete with multidimensional scaling methods and recently introduced algorithms of dimension reduction, such as the locally linear embedding (LLE) [48] and ISOMAP [55] algorithms. The difference between these two approaches is that the later ones seek new point coordinates directly and do not use any intermediate geometrical objects. This has several advantages, in particular that a) there is a unique solution for the problem (the methods are not iterative in their nature, there is no problem of grid initialization) and b) there is no problem of choosing a good way to project points onto a non-linear manifold. Another advantage is that the methods are not limited by several first dimensions in dimension reduction (it is difficult in practice to manipulate non-linear manifolds of dimension more than three). However, when needed we can overcome this limitation by using an iterative calculation of low-dimensional objects.

A principal manifold can serve as a non-linear low-dimensional screen to project data. The explicit introduction of such an object gives additional benefits to users. First, the manifold approximates the data and can be used

itself, without applying projection, to visualize different functions defined in data space (for example, density estimation). Also the manifold as a mediator object, “fixing” the structure of a learning dataset, can be used in visualization of data points that were not used in the learning process, for example, for visualization of dataflow “on the fly”. Constructing manifolds does not use a point-to-point distance matrix, this is particularly useful for datasets with large n when there are considerable memory requirements for storing all pair-wise distances. Also using principal manifolds is expected to be less susceptible to additive noise than the methods based on the local properties of point-to-point distances. To conclude this short comparison, LLE and ISOMAP methods are more suitable if the low-dimensional structure in multidimensional data space is complicated but is expected, and if the data points are situated rather tightly on it. Principal manifolds are more applicable for the visualization of real-life noisy observations, appearing in economics, biology, medicine and other sciences, and for constructing data screens showing not only the data but different related functions defined in data space.

In this paper we demonstrated advantages of using non-linear principal manifolds in visualization of DNA microarrays, data massively generated in biological laboratories. The non-linear approach, compared to linear PCA which is routinely used for analysis of this kind of data, gives better representation of distance structure and class information.

References

1. Aizenberg L.: Carleman's Formulas in Complex Analysis: Theory and Applications. Mathematics and its Applications, 244. Kluwer (1993)
2. Banfield, J.D., Raftery, A.E.: Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *Journal of the American Statistical Association* **87** (417), 7–16 (1992)
3. Bishop, C.M., Svensén, M., and Williams, C.K.I.: GTM: The generative topographic mapping. *Neural Computation* **10** (1), 215–234 (1998)
4. Born, M. and Huang, K.: Dynamical theory of crystal lattices. Oxford University Press (1954)
5. Cai, W., Shao, X., and Maigret, B.: Protein-ligand recognition using spherical harmonic molecular surfaces: towards a fast and efficient filter for large virtual throughput screening. *J. Mol. Graph. Model.* **20** (4), 313–28 (2002)
6. Dergachev, V.A., Gorban, A.N., Rossiev, A.A., Karimova, L.M., Kuandykov, E.B., Makarenko, N.G., and Steier, P.: The filling of gaps in geophysical time series by artificial neural networks. *Radiocarbon* **43** 2A, 365–371 (2001)
7. Dongarra, J., Lumsdaine, A., Pozo, R., and Remington, K.: A sparse matrix library in C++ for high performance architectures. In: Proceedings of the Second Object Oriented Numerics Conference, 214–218 (1994)
8. Dyrskjot, L., Thykjaer, T., Kruhoffer, M. et al.: Identifying distinct classes of bladder carcinoma using microarrays. *Nat Genetics* **33** (1), 90–96 (2003)
9. Durbin, R. and Willshaw, D.: An analogue approach to the traveling salesman problem using an elastic net method. *Nature* **326** (6114), 689–691 (1987)

10. Elmap: C++ package available online:
<http://www.ihes.fr/~zinovyev/vidaexpert/elmap>
11. Erwin, E., Obermayer, K., and Schulten, K.: Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics* **67**, 47–55 (1992)
12. Fréchet, M.: Les éléments aléatoires de nature quelconque dans un espace distancié. *Ann. Inst. H. Poincaré* **10**, 215–310 (1948)
13. Gorban A.N. (ed.): Methods of neuroinformatics (in Russian). Krasnoyarsk State University Press (1998)
14. Gorban, A.N., Karlin, I.V., and Zinovyev, A.Yu.: Invariant grids for reaction kinetics. *Physica A* **333**, 106–154. (2004)
15. Gorban, A.N., Karlin, I.V., and Zinovyev, A.Yu.: Constructive methods of invariant manifolds for kinetic problems. *Phys. Reports* **396** (4-6), 197–403 (2004) Preprint online: <http://arxiv.org/abs/cond-mat/0311017>.
16. Gorban, A.N., Pitenco, A.A., Zinov'ev, A.Y., and Wunsch, D.C.: Vizualization of any data using elastic map method. *Smart Engineering System Design* **11**, 363–368 (2001)
17. Gorban, A.N. and Rossiev, A.A.: Neural network iterative method of principal curves for data with gaps. *Journal of Computer and System Sciences International* **38** (5), 825–831 (1999)
18. Gorban, A., Rossiev, A., Makarenko, N., Kuandykov, Y., and Dergachev, V.: Recovering data gaps through neural network methods. *International Journal of Geomagnetism and Aeronomy* **3** (2), 191–197 (2002)
19. Gorban, A.N., Rossiev, A.A., and Wunsch D.C.: Neural network modeling of data with gaps: method of principal curves, Carleman's formula, and other. The talk was given at the USA–NIS Neurocomputing opportunities workshop, Washington DC, July 1999 (Associated with IJCNN'99).
20. Gorban, A.N. and Zinovyev, A.Yu.: Visualization of data by method of elastic maps and its applications in genomics, economics and sociology. Preprint of Institut des Hautes Etudes Scientifiques, M/01/36, 2001.
<http://www.ihes.fr/PREPRINTS/M01/Resu/resu-M01-36.html>
21. Gorban, A.N. and Zinovyev, A.Yu.: Method of elastic maps and its applications in data visualization and data modeling. *International Journal of Computing Anticipatory Systems, CHAOS* **12**, 353–369 (2001)
22. Gorban, A.N., Zinovyev, A.Yu., and Pitenco, A.A.: Visualization of data using method of elastic maps (in Russian). *Informatsionnie technologii* **6**, 26–35 (2000)
23. Gorban, A.N., Zinovyev, A.Yu., and Pitenco, A.A.: Visualization of data. Method of elastic maps (in Russian). *Neurocomputers* **4**, 19–30 (2002)
24. Gorban, A.N., Zinovyev, A.Yu., and Wunsch, D.C.: Application of the method of elastic maps in analysis of genetic texts. In: Proceedings of International Joint Conference on Neural Networks (IJCNN Portland, Oregon, July 20-24) (2003)
25. Gorban, A. and Zinovyev, A.: Elastic Principal Graphs and Manifolds and their Practical Applications. *Computing* **75**, 359 –379 (2005)
26. Gorban, A.N. Sumner, N.R., and Zinovyev A.Y.: Topological grammars for data approximation. *Applied Mathematics Letters* **20** (2007) 382-386 (2006)
27. Gusev, A.: Finite element mapping for spring network representations of the mechanics of solids. *Phys. Rev. Lett.* **93**(2), 034302 (2004)
28. Hastie, T.: Principal curves and surfaces. PhD Thesis, Stanford University (1984)

29. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association* **84** (406), 502–516 (1989)
30. Zou, H. and Hastie, T.: Regularization and variable selection via the elastic net. *J. R. Statist. Soc. B*, **67**, Part 2, 301–320 (2005)
31. Kaski, S., Kangas, J., and Kohonen, T.: Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, **1**, 102–350 (1998)
32. Kendall, D.G.: A Survey of the Statistical Theory of Shape. *Statistical Science*, **4** (2), 87–99 (1989)
33. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics* **43**, 59–69 (1982)
34. Kégl: Principal curves: learning, design, and applications. Ph. D. Thesis, Concordia University, Canada (1999)
35. Kégl, B. and Krzyzak, A.: Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24** (1), 59–74 (2002)
36. Kégl, B., Krzyzak, A., Linder, T., and Zeger, K.: A polygonal line algorithm for constructing principal curves. In: *Neural Information Processing Systems 1998*. MIT Press, 501–507 (1999)
37. Kégl, B., Krzyzak, A., Linder, T., and Zeger, K.: Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(2), 281–297 (2000)
38. LeBlanc, M. and Tibshirani R.: Adaptive principal surfaces. *Journal of the American Statistical Association* **89**, 53–64 (1994)
39. Leung, Y.F. and Cavalieri, D.: Fundamentals of cDNA microarray data analysis. *Trends Genet.* **19** (11), 649–659 (2003)
40. Mirkin, B.: *Clustering for Data Mining: A Data Recovery Approach*. Chapman and Hall, Boca Raton (2005)
41. Mulier, F. and Cherkassky, V.: Self-organization as an iterative kernel smoothing process. *Neural Computation* **7**, 1165–1177 (1995)
42. Oja, M., Kaski, S., and Kohonen, T.: Bibliography of Self-Organizing Map (SOM) Papers: 1998–2001 Addendum. *Neural Computing Surveys*, **3**, 1–156 (2003)
43. Pearson K.: On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, series 6 (2), 559–572 (1901)
44. Perou, C.M., Sorlie, T., Eisen, M.B. et al.: Molecular portraits of human breast tumours. *Nature* **406** (6797), 747–752 (2000)
45. “Principal manifolds for data cartography and dimension reduction”, Leicester, UK, August 2006. A web-page with test microarrays datasets provided for participants of the workshop: <http://www.ihes.fr/zinovyev/princmanif2006>
46. Ritter, H., Martinetz, T., and Schulten, K.: *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley Reading, Massachusetts, 1992.
47. H. Ritter. Parametrized self-organizing maps. In *Proceedings ICANN'93 International Conference on Artificial Neural Networks* Amsterdam, 568–575. Springer (1993)
48. Roweis, S. and Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290** (2000), 2323–2326 (2000)
49. Sayle, R. and Bissell, A.: RasMol: A Program for fast realistic rendering of molecular structures with shadows. In: *Proceedings of the 10th Eurographics UK'92 Conference*, University of Edinburgh, Scotland (1992)

50. Schölkopf, B., Smola, A. and Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10** (5), 1299–1319 (1998)
51. Shyamsundar, R., Kim, Y.H., Higgins, J.P. et al.: A DNA microarray survey of gene expression in normal human tissues. *Genome Biology* **6** R22 (2005)
52. Smola, A.J., Williamson, R.C., Mika, S., and Schölkopf B.: Regularized principal manifolds. EuroCOLT'99, Lecture Notes in Artificial Intelligence 1572, 214-229 (1999)
53. Smola, A.J., Mika, S., Schölkopf, B., and Williamson, R.C.: Regularized Principal Manifolds. *Journal of Machine Learning Research* **1**, 179-209 (2001)
54. Stanford, D. and Raftery, A.E.: Principal curve clustering with noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(6), 601–609 (2000)
55. Tenenbaum, J.B., Silva, V. De, and Langford J.C.: A global geometric framework for nonlinear dimensionality reduction. *Science* **290**, 2319–2323 (2000)
56. Van Gelder, A. and Wilhelms, J.: Simulation of elastic membranes and soft tissue with triangulated spring meshes. Technical Report: UCSC-CRL-97-12 (1997)
57. Verbeek, J.J., Vlassis, N., and Kroese, B.: A k -segments algorithm for finding principal curves. Technical report (2000) (See also *Pattern Recognition Letters*, **23** (8), (2002) 1009-1017 (2002))
58. VidaExpert: Stand-alone application for multidimensional data visualization, available online:
<http://bioinfo.curie.fr/projects/vidaexpert>
59. VIMIDA: Java-applet for Visualisation of MultIdimensional DAta, available online:
<http://bioinfo-out.curie.fr/projects/vimida>
60. Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, A.M., Look, M.P., Yang, F., Tsvantov, D., Timmermans, M., Meijer-van Gelder, M.E., Yu, J. et al.: Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet* **365**, 671-679 (2005)
61. Xie, H. and Qin, H.: A Physics-based framework for subdivision surface design with automatic rules control. In: Proceedings of the Tenth Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2002), IEEE Press, 304–315 (2002)
62. Yin, H.: Data visualisation and manifold mapping using ViSOM. *Neural Networks*, **15**, 1005–1016 (2002)
63. Yin, H.: Nonlinear multidimensional data projection and visualisation. In: Lecture Notes in Computer Science, vol. 2690, 377–388 (2003)
64. Zinovyev A.: Visualization of Multidimensional Data. Krasnoyarsk State University Press Publ. (2000)
65. Zinovyev, A.Yu., Gorban, A.N., and Popova, T.G.: Self-organizing approach for automated gene identification. *Open Systems and Information Dynamics* **10**(4), 321–333 (2003)
66. Zinovyev, A.Yu., Pitenko, A.A., and Popova, T.G.: Practical applications of the method of elastic maps (in Russian). *Neurocomputers* **4**, 31–39 (2002)

Topology-Preserving Mappings for Data Visualisation

Marian Peña, Wesam Barbakh, and Colin Fyfe

Applied Computational Intelligence Research Unit,
The University of Paisley, Scotland,
`{marian.pena,wesam.barbakh,colin.fyfe}@paisley.ac.uk`

Summary. We present a family of topology preserving mappings similar to the Self-Organizing Map (SOM) and the Generative Topographic Map (GTM). These techniques can be considered as a non-linear projection from input or data space to the output or latent space (usually 2D or 3D), plus a clustering technique, that updates the centres. A common frame based on the GTM structure can be used with different clustering techniques, giving new properties to the algorithms.

Thus we have the topographic product of experts (ToPoE) with the Product of Experts substituting the Mixture of Experts of the GTM, two versions of the Harmonic Topographic Mapping (HaToM) that utilise the K -Harmonic Means (KHM) clustering, and the faster Topographic Neural Gas (ToNeGas), with the inclusion of Neural Gas in the inner loop. We also present the Inverse-weighted K -means Topology-Preserving Map (IKToM), based on the same structure for non-linear projection, that makes use of a new clustering technique called The Inverse Weighted K -Means. We apply all the algorithms to a high dimensional dataset, and compare it as well with the Self-Organizing Map, in terms of visualisation, clustering and topology preservation.

5.1 Introduction

Topographic mappings are a class of dimensionality reduction techniques that seek to preserve some of the structure of the data in the geometric structure of the mapping. The term “geometric structure” refers to the relationships between distances in data space and the distances in the projection to the topographic map. In some cases all distance relationships between data points are important, which implies a desire for global isometry between the data space and the map space. Alternatively, it may only be considered important that local neighbourhood relationships are maintained, which is referred to as topological ordering [19]. When the topology is preserved, if the projections of two points are close, it is because, in the original high dimensional space, the two points were close. The closeness criterion is usually the Euclidean distance between the data patterns.

One clear example of a topographic mapping is a Mercator projection of the spherical earth into two dimensions; the visualisation is improved, but some of the distances in certain areas are distorted. These projections imply a loss of some of the information which inevitably gives some inaccuracy but they are an invaluable tool for visualisation and data analysis, e.g. for cluster detection. Two previous works in this area have been the Self-Organizing Map (SOM) [13] and the Generative Topographic Map (GTM) [4].

Kohonen's SOM is a neural network which creates a topology-preserving map because there is a topological structure imposed on the nodes in the network. It takes into consideration the physical arrangement of the nodes. Nodes that are “close” together are going to interact differently than nodes that are “far” apart. The GTM was developed by Bishop et al. as a probabilistic version of the SOM, in order to overcome some of the problems of this map, especially the lack of objective function.

Without taking into account the probabilistic aspects of the GTM algorithm, this can be considered as a projection from latent space to dataspace to adapt the nodes to the datapoints (in this chapter called GTM structure), using K -Means with soft responsibilities as clustering technique to update the prototypes in dataspace.

In this chapter we review several topology-preserving maps that make use of the general structure of the GTM. We first review four clustering techniques used in our algorithms in section 5.2. Then we define the common structure based on the GTM in section 5.3.1, and develop the four topology preserving mappings in section 5.3. Finally we compare all the algorithms with the SOM in the experimental section.

5.2 Clustering Techniques

5.2.1 K -Means

K -Means clustering is an algorithm to divide or to group samples \mathbf{x}_i based on attributes/features into K groups. K is a positive integer number that has to be given in advance. The grouping is done by minimizing the sum of squares of distances between data and the corresponding prototypes \mathbf{m}_k .

The performance function for K -Means may be written as

$$J = \sum_{i=1}^N \min_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^2, \quad (5.1)$$

which we wish to minimise by moving the prototypes to the appropriate positions. Note that (5.1) detects only the prototypes closest to data points and then distributes them to give the minimum performance which determines the clustering. Any prototype which is still far from data is not utilised and does

not enter any calculation to determine minimum performance, which may result in dead prototypes, which are never appropriate for any cluster. Thus initializing prototypes appropriately can play a big effect in *K*-Means.

The algorithm has the following steps:

- Step 1. Begin with a decision on the value of $K = \text{number of clusters}$.
- Step 2. Put any initial partition that divides the data into K clusters randomly.
- Step 3. Take each sample in sequence and compute its distance from the prototypes of each of the clusters. If a sample is not currently in the cluster with the closest prototype, switch this sample to that cluster and update the prototype of the cluster gaining the new sample and the cluster losing the sample.
- Step 4. Repeat step 3 until convergence is achieved, that is until a pass through the training samples causes no new assignments.

Considering a general formula for the updating of the prototypes in clustering techniques we may write a general formula

$$\mathbf{m}_k \leftarrow \frac{\sum_{i=1}^N \text{mem}(\mathbf{m}_k / \mathbf{x}_i) * \text{weight}(\mathbf{x}_i) * \mathbf{x}_i}{\sum_{i=1}^N \text{mem}(\mathbf{m}_k / \mathbf{x}_i) * \text{weight}(\mathbf{x}_i)}, \quad (5.2)$$

where

- $\text{weight}(\mathbf{x}_i) > 0$ is the weighting function that defines how much influence a data point \mathbf{x}_i has in recomputing the prototype parameters \mathbf{m}_k in the next iteration.
- $\text{mem}(\mathbf{m}_k / \mathbf{x}_i) \geq 0$ with $\sum_{k=1}^K \text{mem}(\mathbf{m}_k / \mathbf{x}_i) = 1$ the membership function that decides the portion of $\text{weight}(\mathbf{x}_i) * \mathbf{x}_i$ associated with \mathbf{m}_k .

The membership and weight functions for KM are:

$$\begin{aligned} \text{mem}_{KM}(\mathbf{m}_l / \mathbf{x}_i) &= \begin{cases} 1, & \text{if } l = \min_k \|\mathbf{x}_i - \mathbf{m}_k\|, \\ 0, & \text{otherwise;} \end{cases} \\ \text{weight}_{KM}(\mathbf{x}_i) &= 1. \end{aligned} \quad (5.3)$$

The main problem with the *K*-Means algorithm is that, as with the GTM, the initialisation of the parameters can lead to a local minimum. Also the number of prototypes K has to be pre-determined by the user, although this is really one of the objectives of clustering.

5.2.2 K-Harmonic Means

Harmonic Means or Harmonic Averages are defined for spaces of derivatives. For example, if you travel $\frac{1}{2}$ of a journey at 10 km/hour and the other $\frac{1}{2}$ at 20 km/hour, your total time taken is $\frac{d}{10} + \frac{d}{20}$ and so the average speed is

$\frac{2d}{\frac{d}{10} + \frac{d}{20}} = \frac{2}{\frac{1}{10} + \frac{1}{20}}$. In general, the Harmonic Average of K values, a_1, \dots, a_K , is defined as

$$HA(\{a_i, i = 1, \dots, K\}) = \frac{K}{\sum_{k=1}^K \frac{1}{a_k}}. \quad (5.4)$$

Harmonic Means were applied to the K -Means algorithm in [22] to make K -Means a more robust algorithm. The recursive formula to update the prototypes is

$$J = \sum_{i=1}^N \frac{K}{\sum_{k=1}^K \frac{1}{d(\mathbf{x}_i, \mathbf{m}_k)^2}}; \quad (5.5)$$

$$\mathbf{m}_k = \frac{\sum_{i=1}^N \frac{1}{d_{ik}^4 (\sum_{l=1}^K \frac{1}{d_{il}^2})^2} \mathbf{x}_i}{\sum_{i=1}^N \frac{1}{d_{ik}^4 (\sum_{l=1}^K \frac{1}{d_{il}^2})^2}}, \quad (5.6)$$

where d_{ik} is the Euclidean distance between the i^{th} data point and the k^{th} prototype so that $d(\mathbf{x}_i, \mathbf{m}_k) = \|\mathbf{x}_i - \mathbf{m}_k\|$.

In [22] extensive simulations show that this algorithm converges to a better solution (less prone to finding a local minimum because of poor initialisation) than both standard K -Means or a mixture of experts trained using the EM algorithm.

Zhang subsequently developed a generalised version of the algorithm [20, 21] that includes the p^{th} power of the L^2 distance which creates a “dynamic weighting function” that determines how data points participate in the next iteration in the calculation of the new prototypes \mathbf{m}_k . The weight is bigger for data points further away from the prototypes, so that their participation is boosted in the next iteration. This makes the algorithm insensitive to initialisation and also prevents one cluster from taking more than one prototype.

The aim of K-Harmonic Means was to improve the winner-takes-all partitioning strategy of K -Means that gives a very strong relation between each datapoint and its closest prototype, so that the change in membership is not allowed until another prototype is closer. The transition of prototypes between areas of high density is more continuous in K-Harmonic Means due to the distribution of associations between prototypes and datapoints.

The soft membership¹ in the generalised K-Harmonic Means is

$$mem(\mathbf{m}_k / \mathbf{x}_i) = \frac{\|\mathbf{x}_i - \mathbf{m}_k\|^{-p-2}}{\sum_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^{-p-2}} \quad (5.7)$$

allows the data points to belong partly to all prototypes.

¹ Soft membership means that each datapoint can belong to more than one prototype.

The boosting properties for the generalised version of K-Harmonic Means ($p > 2$) are given by the weighting function [9]:

$$\text{weight}(\mathbf{x}_i) = \frac{\sum_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^{-p-2}}{(\sum_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^{-p})^2}, \quad (5.8)$$

where the dynamic function gives a variable influence to data in clustering in a similar way to boosting [6] since the effect of any particular data point on the re-calculation of a prototype is $O(\|\mathbf{x}_i - \mathbf{m}_k\|^{2p-p-2})$, which for $p > 2$ has greatest effect for larger distances.

5.2.3 Neural Gas

Neural Gas (NG) [14] is a vector quantization technique with soft competition between the units; it is called the Neural Gas algorithm because the prototypes of the clusters move around in the data space similar to the Brownian movement of gas molecules in a closed container. In each training step, the squared Euclidean distances

$$d_{ik} = \|\mathbf{x}_i - \mathbf{m}_k\|^2 = (\mathbf{x}_i - \mathbf{m}_k)^T * (\mathbf{x}_i - \mathbf{m}_k) \quad (5.9)$$

between a randomly selected input vector \mathbf{x}_i from the training set and all prototypes \mathbf{m}_k are computed; the vector of these distances is \mathbf{d} . Each prototype k is assigned a rank $r_k(d) = 0, \dots, K-1$, where a rank of 0 indicates the closest and a rank of $K-1$ the most distant prototype to \mathbf{x} . The learning rule is then

$$\mathbf{m}_k = \mathbf{m}_k + \varepsilon * h_\rho[r_k(\mathbf{d})] * (\mathbf{x} - \mathbf{m}_k). \quad (5.10)$$

The function

$$h_\rho(r) = e^{(-r/\rho)} \quad (5.11)$$

is a monotonically decreasing function of the ranking that adapts not only the closest prototype, but all the prototypes, with a factor exponentially decreasing with their rank. The width of this influence is determined by the neighborhood range ρ . The learning rule is also affected by a global learning rate ε . The values of ρ and ε decrease exponentially from an initial positive value $(\rho(0), \varepsilon(0))$ to a smaller final positive value $(\rho(T), \varepsilon(T))$ according to

$$\rho(t) = \rho(0) * [\rho(T)/\rho(0)]^{(t/T)} \quad (5.12)$$

and

$$\varepsilon(t) = \varepsilon(0) * [\varepsilon(T)/\varepsilon(0)]^{(t/T)}, \quad (5.13)$$

where t is the time step and T the total number of training steps, forcing more local changes with time.

5.2.4 Weighted K -Means

This clustering technique was introduced in [3, 2]. We might consider the following performance function:

$$J_A = \sum_{i=1}^N \sum_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^2 , \quad (5.14)$$

which provides a relationship between all the data points and prototypes, but it doesn't provide useful clustering at minimum performance since

$$\frac{\partial J_A}{\partial \mathbf{m}_k} = 0 \implies \mathbf{m}_k = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i, \forall k . \quad (5.15)$$

Minimizing the performance function groups all the prototypes to the centre of the data set regardless of the initial position of the prototypes which is useless for identification of clusters.

We wish to form a performance function with following properties:

- Minimum performance gives an intuitively 'good' clustering.
- It creates a relationship between all data points and all prototypes.

(5.14) provides an attempt to reduce the sensitivity to prototypes' initialization by making a relationship between all data points and all prototypes while (5.1) provides an attempt to cluster data points at the minimum of the performance function. Therefore it may seem that what we want is to combine features of (5.1) and (5.14) to make a performance function such as:

$$J_1 = \sum_{i=1}^N \left[\sum_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\| \right] \min_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^2 . \quad (5.16)$$

As pointed out by a reviewer, there is a potential problem with using $\|\mathbf{x}_i - \mathbf{m}_k\|$ rather than its square in the performance function but in practice, this has not been found to be a problem. We derive the clustering algorithm associated with this performance function by calculating the partial derivatives of (5.16) with respect to the prototypes. We call the resulting algorithm Weighted K -Means (though recognising that other weighted versions of K -Means have been developed in the literature). The partial derivatives are calculated as

$$\frac{\partial J_{1,i}}{\partial \mathbf{m}_r} = -(\mathbf{x}_i - \mathbf{m}_r) \{ \|\mathbf{x}_i - \mathbf{m}_r\| + 2 \sum_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\| \} = -(\mathbf{x}_i - \mathbf{m}_r) a_{ir} , \quad (5.17)$$

when \mathbf{m}_r is the closest prototype to \mathbf{x}_i and

$$\frac{\partial J_{1,i}}{\partial \mathbf{m}_k} = -(\mathbf{x}_i - \mathbf{m}_k) \frac{\|\mathbf{x}_i - \mathbf{m}_r\|^2}{\|\mathbf{x}_i - \mathbf{m}_k\|} = -(\mathbf{x}_i - \mathbf{m}_k) b_{ik} , \quad (5.18)$$

otherwise.

We then solve this by summing over the whole data set and finding the fixed point solution of

$$\frac{\partial J_1}{\partial \mathbf{m}_r} = \sum_{i=1}^N \frac{\partial J_{1,i}}{\partial \mathbf{m}_r} = 0 \quad (5.19)$$

which gives a solution of

$$\mathbf{m}_r = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}}. \quad (5.20)$$

We have given extensive analysis and simulations in [3, 2] showing that this algorithm will cluster the data with the prototypes which are closest to the data points being positioned in such a way that the clusters can be identified. However there are some potential prototypes which are not sufficiently responsive to the data and so never move to identify a cluster. In fact, these points move to (a weighted) prototype of the data set. This may be an advantage in some cases in that we can easily identify redundancy in the prototypes however it does waste computational resources unnecessarily.

5.2.5 The Inverse Weighted K -Means

Consider the performance algorithm

$$J_2 = \sum_{i=1}^N \left[\sum_{k=1}^K \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \right] \min_{k=1}^K \|\mathbf{x}_i - \mathbf{m}_k\|^n. \quad (5.21)$$

Let \mathbf{m}_r be the closest prototype to \mathbf{x}_i . Then

$$\begin{aligned} J_2(\mathbf{x}_i) &= \left[\sum_{k=1}^K \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \right] \|\mathbf{x}_i - \mathbf{m}_r\|^n \\ &= \|\mathbf{x}_i - \mathbf{m}_r\|^{n-p} + \sum_{j \neq r} \frac{\|\mathbf{x}_i - \mathbf{m}_r\|^n}{\|\mathbf{x}_i - \mathbf{m}_k\|^p}. \end{aligned} \quad (5.22)$$

Therefore

$$\begin{aligned} \frac{\partial J_2(\mathbf{x}_i)}{\partial \mathbf{m}_r} &= -(n-p)(\mathbf{x}_i - \mathbf{m}_r)\|\mathbf{x}_i - \mathbf{m}_r\|^{n-p-2} \\ &\quad - n(\mathbf{x}_i - \mathbf{m}_r)\|\mathbf{x}_i - \mathbf{m}_r\|^{n-2} \sum_{j \neq r} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \\ &= (\mathbf{x}_i - \mathbf{m}_r)a_{ir}, \end{aligned} \quad (5.23)$$

$$\frac{\partial J_2(\mathbf{x}_i)}{\partial \mathbf{m}_k} = p(\mathbf{x}_i - \mathbf{m}_k) \frac{\|\mathbf{x}_i - \mathbf{m}_r\|^n}{\|\mathbf{x}_i - \mathbf{m}_k\|^{p+2}} = (\mathbf{x}_i - \mathbf{m}_k)b_{ik}. \quad (5.24)$$

At convergence, $E(\frac{\partial J_2}{\partial \mathbf{m}_r}) = 0$ where the expectation is taken over the data set. If we denote by V_k the set of points, \mathbf{x} for which \mathbf{m}_k is the closest, we have

$$\begin{aligned} \frac{\partial J_2}{\partial \mathbf{m}_r} = 0 \iff & \int_{\mathbf{x} \in V_r} \{(n-p)(\mathbf{x}_i - \mathbf{m}_r) \|\mathbf{x}_i - \mathbf{m}_r\|^{n-p-2} \\ & + n(\mathbf{x} - \mathbf{m}_r) \|\mathbf{x} - \mathbf{m}_r\|^{n-2} \sum_{j \neq r} \frac{1}{\|\mathbf{x} - \mathbf{m}_j\|^p} P(\mathbf{x})\} d\mathbf{x} \\ & + \sum_{k \neq r} \int_{\mathbf{x} \in V_k} p(\mathbf{x} - \mathbf{m}_k) \frac{\|\mathbf{x} - \mathbf{m}_r\|^n}{\|\mathbf{x} - \mathbf{m}_k\|^{p+2}} P(\mathbf{x}) d\mathbf{x} = 0, \quad (5.25) \end{aligned}$$

where $P(\mathbf{x})$ is the probability measure associated with the data set. This is, in general, a very difficult set of equations to solve. However it is readily seen that, for example, in the special case that there are the same number of prototypes as there are data points, that one solution is to locate each prototype at each data point (at which time $\frac{\partial J_2}{\partial \mathbf{m}_r} = 0$). Again solving this over all the data set results in

$$\mathbf{m}_r = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}}. \quad (5.26)$$

From (5.25), we see that $n \geq p$ if the direction of the first term is to be correct and $n \leq p+2$ to ensure stability in all parts of that equation. In practice, we have found that a viable algorithm may be found by using (5.24) for all prototypes (and thus never using (5.23) for the closest prototype). We will call this the Inverse Weighted K -Means Algorithm.

5.3 Topology Preserving Mappings

5.3.1 Generative Topographic Map

The Generative Topographic Mapping (GTM) is a non-linear latent variable model, intended for modeling continuous, intrinsically low-dimensional probability distributions, embedded in high-dimensional spaces. It provides a principled alternative to the self-organizing map resolving many of its associated theoretical problems. An important, potential application of the GTM is visualization of high-dimensional data. Since the GTM is non-linear, the relationship between data and its visual representation may be far from trivial, but a better understanding of this relationship can be gained by computing the so-called magnification factors [5].

There are two principal limitations of the basic GTM model. The computational effort required will grow exponentially with the intrinsic dimensionality of the density model. However, if the intended application is visualization, this will typically not be a problem. The other limitation is the initialisation of the parameters, that can lead the algorithm to a local optimum.

The GTM defines a non-linear, parametric mapping $\mathbf{y}(\mathbf{x}; W)$ from a q -dimensional latent space to a d -dimensional data space $\mathbf{x} \in R^d$, where normally $q < d$. The mapping is defined to be continuous and differentiable. $\mathbf{y}(\mathbf{t}; W)$ maps every point in the latent space to a point in the data space. Since the latent space is q -dimensional, these points will be confined to a q -dimensional manifold non-linearly embedded into the d -dimensional data space. If we define a probability distribution over the latent space, $p(\mathbf{t})$, this will induce a corresponding probability distribution into the data space. Strictly confined to the q -dimensional manifold, this distribution would be singular, so it is convolved with an isotropic Gaussian noise distribution, given by

$$p(\mathbf{x}|\mathbf{t}, W, \beta) = \left(\frac{\beta}{2\pi} \right)^{d/2} \exp \left\{ -\frac{\beta}{2} \sum_{d=1}^d (\mathbf{x}_d - y_d(\mathbf{t}, W))^2 \right\}, \quad (5.27)$$

where \mathbf{x} is a point in the data space and β denotes the noise variance. By integrating out the latent variable, we get the probability distribution in the data space expressed as a function of the parameters β and W ,

$$p(\mathbf{x}|W, \beta) = \int p(\mathbf{x}|\mathbf{t}, W, \beta) p(\mathbf{t}) d\mathbf{t}. \quad (5.28)$$

Choosing $p(\mathbf{t})$ as a set of K equally weighted delta functions on a regular grid,

$$p(\mathbf{t}) = \frac{1}{K} \sum_{k=1}^K \delta(\mathbf{t} - \mathbf{t}_k), \quad (5.29)$$

the integral in (5.28) becomes a sum,

$$p(\mathbf{x}|W, \beta) = \frac{1}{K} \sum_{k=1}^K p(\mathbf{x}|\mathbf{t}_k, W, \beta). \quad (5.30)$$

Each delta function centre maps into the centre of a Gaussian which lies in the manifold embedded in the data space. This algorithm defines a constrained mixture of Gaussians[11, 12], since the centres of the mixture components can not move independently of each other, but all depend on the mapping $\mathbf{y}(\mathbf{t}; W)$. Moreover, all components of the mixture share the same variance, and the mixing coefficients are all fixed at $1/K$. Given a finite set of independent and identically distributed (i.i.d.) data points, $\{\mathbf{x}_{i=1}^N\}$, the log-likelihood function of this model is maximized by means of the Expectation Maximisation algorithm with respect to the parameters of the mixture, namely W and β . The form of the mapping $\mathbf{y}(\mathbf{t}; w)$ is defined as a generalized linear regression model $\mathbf{y}(\mathbf{t}; W) = \phi(\mathbf{t})W$ where the elements of $\phi(\mathbf{t})$ consist of M fixed basis functions $\phi_i(\mathbf{t})_{i=1}^M$, and W is a $d \times M$ matrix.

If we strip out the probabilistic underpinnings of the GTM method, the algorithm can be considered as a non-linear model structure, to which a clustering technique is applied in data space to update the prototypes, in this case the *K*-Means algorithm. In the next sections we present four algorithms that share this model structure.

5.3.2 Topographic Product of Experts ToPoE

Hinton [10] investigated a product of K experts with

$$p(\mathbf{x}_i|\Theta) \propto \prod_{k=1}^K p(\mathbf{x}_i|k), \quad (5.31)$$

where Θ is the set of current parameters in the model. Hinton notes that using Gaussians alone does not allow us to model e.g. multi-modal distributions, however the Gaussian is ideal for our purposes. Thus the base model is

$$p(\mathbf{x}_i|\Theta) \propto \prod_{k=1}^K \left(\frac{\beta}{2\pi} \right)^{\frac{D}{2}} \exp \left(-\frac{\beta}{2} \|\mathbf{m}_k - \mathbf{x}_i\|^2 \right). \quad (5.32)$$

To fit this model to the data we can define a cost function as the negative logarithm of the probabilities of the data so that

$$J = \sum_{i=1}^N \sum_{k=1}^K \frac{\beta}{2} \|\mathbf{m}_k - \mathbf{x}_i\|^2. \quad (5.33)$$

In [7] the Product of Gaussian model was extended by allowing latent points² to have different responsibilities depending on the data point presented:

$$p(\mathbf{x}_i|\Theta) \propto \prod_{k=1}^K \left(\frac{\beta}{2\pi} \right)^{\frac{D}{2}} \exp \left(-\frac{\beta}{2} \|\mathbf{m}_k - \mathbf{x}_i\|^2 r_{ik} \right), \quad (5.34)$$

where r_{ik} is the responsibility of the k^{th} expert for the data point, \mathbf{x}_i . Thus all the experts are acting in concert to create the data points but some will take more responsibility than others. Note how crucial the responsibilities are in this model: if an expert has no responsibility for a particular data point, it is in essence saying that the data point could have a high probability as far as it is concerned. We do not allow a situation to develop where no expert accepts responsibility for a data point; if no expert accepts responsibility for a data point, they all are given equal responsibility for that data point (see below).

² The latent points, \mathbf{t}_k , generate the \mathbf{m}_k prototypes, which are the latent points' projections in data space; thus there is a bijection between the latent points and the prototypes. \mathbf{m}_k act as prototypes of the clusters.

We wish to maximise the likelihood of the data set $X = \{\mathbf{x}_i : i = 1, \dots, N\}$ under this model. The ToPoE learning rule (5.36) is derived from the minimisation of $-\log(p(\mathbf{x}_i|\Theta))$ with respect to a set of parameters which generate the \mathbf{m}_k .

We may update \mathbf{W} either in batch mode or with online learning. To change \mathbf{W} in online learning, we randomly select a data point, say \mathbf{x}_i . We calculate the current responsibility of the k^{th} latent point for this data point,

$$r_{ik} = \frac{\exp(-\gamma d_{ik}^2)}{\sum_{k=1}^K \exp(-\gamma d_{ik}^2)}, \quad (5.35)$$

where $d_{pq} = \|\mathbf{x}_p - \mathbf{m}_q\|$, the Euclidean distance between the p^{th} data point and the projection of the q^{th} latent point (through the basis functions and then multiplied by \mathbf{W}). If no prototypes are close to the data point (the denominator of (5.35) is zero), we set $r_{ik} = \frac{1}{K}, \forall k$. γ is known as the width of the responsibilities and is usually set to 20.

We wish to maximise (5.34) so that the data is most likely under this model. We do this by minimising the $-\log()$ of that probability: define $m_d^{(k)} = \sum_{\omega=1}^M w_{\omega d} \phi_{k\omega}$, i.e. $m_d^{(k)}$ is the projection of the k^{th} latent point on the d^{th} dimension in data space. Similarly let $x_d^{(n)}$ be the d^{th} coordinate of \mathbf{x}_i . These are used in the update rule

$$\Delta_i w_{\omega d} = \sum_{k=1}^K \eta \phi_{k\omega} (x_d^{(i)} - m_d^{(k)}) r_{ik}, \quad (5.36)$$

where we have used Δ_i to signify the change due to the presentation of the i^{th} data point, \mathbf{x}_i , so that we are summing the changes due to each latent point's response to the data points. Note that, for the basic model, we do not change the Φ matrix during training at all.

5.3.3 The Harmonic Topographic Map

The HaToM has the same structure as the GTM, with K latent points that are mapped to a feature space by M Gaussian basis functions, and then into the data space by a matrix of weights \mathbf{W} . In HaToM the initialisation problems of GTM are overcome replacing the arithmetic means of K -Means algorithm with harmonic means, i.e. using K-Harmonic Means [22].

The basic batch algorithm often exhibited twists, such as are well-known in the Self-organizing Map (SOM) [13], so we developed a growing method that prevents the mapping from developing these twists. The latent points are arranged in a square grid in a similar manner to the SOM grid.

We developed two versions of the algorithm [17]. The main structure for the data-driven HaToM or D-HaToM is as follows:

1. Initialise K to 2. Initialise the W weights randomly and spread the centres of the M basis functions uniformly in latent space.
2. Initialise the K latent points uniformly in latent space.
3. Calculate the projection of the latent points to data space. This gives the K prototypes, \mathbf{m}_k .
 - a) count=0
 - b) For every data point, \mathbf{x}_i , calculate $d_{ik} = \|\mathbf{x}_i - \mathbf{m}_k\|$.
 - c) Recalculate prototypes, \mathbf{m}_k , using (5.6).
 - d) If count<MAXCOUNT, count= count +1 and return to 3b
4. Recalculate W using $(\Phi^T \Phi + \delta I)^{-1} \Phi^T \Xi$ where Ξ is the matrix containing the K prototypes, I is identity matrix and δ is a small constant^a, necessary because initially $K < M$ and so the matrix $\Phi^T \Phi$ is singular.
5. If $K < K_{\max}$, $K = K + 1$ and return to 2.

^a usually 0.001 but other values gave similar results

We do not randomise W each time we augment K , but we use the value from the previous iteration to update the prototypes \mathbf{m}_k with the increased number of latent points.

If we wish to use the mapping for visualisation, we must map data points into latent space. We define the responsibility as in (5.35), and the i^{th} data point is represented by \mathbf{y}_i where

$$\mathbf{y}_i = \sum_{k=1}^K r_{ik} \mathbf{t}_k , \quad (5.37)$$

where \mathbf{t}_k is the position of the k^{th} latent point in latent space.

In the model-driven HaToM or M-HaToM, we give greater credence to the model by recalculating W and hence the prototypes, \mathbf{m}_k , within the central loop each time. Thus we are explicitly forcing the structure of the M-HaToM model on the data. The visualisation of the \mathbf{y}_i values in latent space is the same as above.

In [17], we showed that this version had several advantages over the D-HaToM: in particular, the M-HaToM creates tighter clusters of data points and finds an underlying data manifold smoothly no matter how many latent points are used in creating the manifold. The D-HaToM, on the other hand, is too responsive to the data (too influenced by the noise), but this quality makes it more suitable for outlier detection.

Generalised Harmonic Topographic Map (G-HaToM)

The generalised version of K-Harmonic Means can be applied also to the HaToM algorithm. The advantage of this generalisation is the utilisation of a “p” value that, when bigger than 2, gives a boosting-like property to the

updating of the prototypes. The recalculation of the prototypes in this case is:

$$\mathbf{m}_k = \frac{\sum_{i=1}^N \frac{1}{d_{ik}^p (\sum_{l=1}^K \frac{1}{d_{il}^2})^{p-2}} \mathbf{x}_i}{\sum_{i=1}^N \frac{1}{d_{ik}^p (\sum_{l=1}^K \frac{1}{d_{il}^2})^{p-2}}} , \quad (5.38)$$

so that p determines the power of the L^2 distance used in the algorithm.

This generalised version of the algorithm includes the p^{th} power of the L^2 distance which creates a “dynamic weighting function” [20] that determines how data points participate in the next iteration to calculate the new prototypes \mathbf{m}_k . The weight is bigger for data points further away from the prototypes, so that their participation is boosted in the next iteration. This makes the algorithm insensitive to initialisation and also prevents one cluster from taking more than one prototype.

Some results for the generalised version of HaToM can be seen in [16].

5.3.4 Topographic Neural Gas

Topographic Neural Gas (ToNeGas) [18] unifies the underlying structure in GTM for topology preservation, with the technique of Neural Gas (NG). The prototypes in data space are then clustered using the NG algorithm. The algorithm has been implemented based on the Neural Gas algorithm code included in the SOM Toolbox for Matlab [15].

We have used the same growing method as with HaToM but have found that, with the NG learning, we can increment the number of latent points by e.g. 10 each time we augment the map whereas with HaToM, the increase can only be one at a time to get a valid mapping. One of the advantages of this algorithm is that the Neural Gas part is independent of the non-linear projection, thus the clustering efficiency is not limited by the topology preservation restriction.

5.3.5 Inverse-Weighted K -Means Topology-Preserving Map

As with KHM and NG, it is possible to extend the IWKM clustering algorithm to provide a new algorithm for visualization and topology-preserving mappings, by using IWKM with the GTM structure. We called the new algorithm Inverse-weighted K -Means Topology-Preserving Map (IKToM).

5.4 Experiments

We use a dataset containing results of a high-throughput experimental technology application in molecular biology (microarray data [8])³. The datasets

³ <http://www.ihes.fr/~zinovьев/princmanif2006/> Dataset II - ”Three types of bladder cancer”.

contains only 40 observations of high-dimensional data (3036 dimensions) and the data is drawn from three types of bladder cancer: T1, T2+ and Ta. The data can be used in the gene space (40 rows of 3036 variables), or in the sample space (3036 rows of 40 variables), where each sample contains the profiles of the 40 genes. In these experiments we consider the first case. The dataset has been preprocessed to have zero mean; also, in the original dataset some data was missing and these values have been filtered out.

We use the same number of neurons for all the mappings, a 12*12 grid. We used a value of $p = 3$ for HaToM, and $p = 7$ for IKTOM. For several of the results below we have utilised the SOMtoolbox [15] with default values.

5.4.1 Projections in Latent Space

The four algorithms are able to properly separate the three types of cancer in the projection (see Figs. 5.1 and 5.2), but ToPoE requires to run for 100,000 iterations while the others do it with only 20 passes or less.

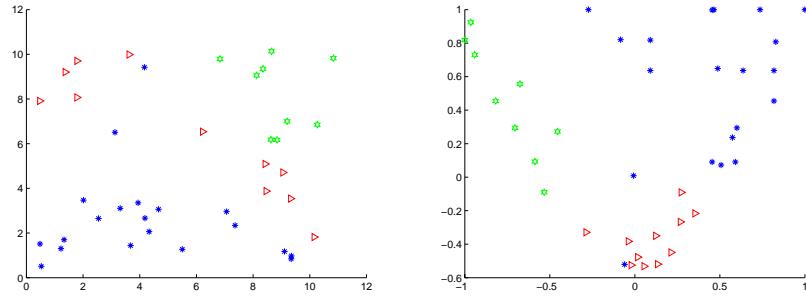


Fig. 5.1. The ToPoE (left) and HaToM (right) projection for the gene data with $p=6$. T1 in triangles (red), T2+ in circles (green) and Ta in stars (blue).

5.4.2 Responsibilities

The responsibilities of each latent point for each datapoint are shown in Fig. 5.3.

5.4.3 U-matrix, Hit Histograms and Distance Matrix

The U-Matrix assigns to each cell the average distance to all of its neighbors. This enables the identification of regions of similarity using different colors for different ranges of distances.

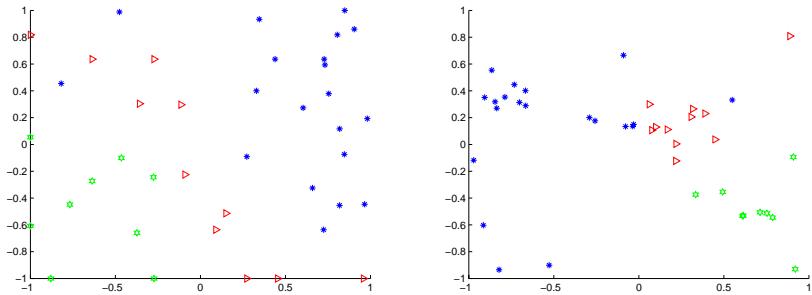


Fig. 5.2. The ToNeGas (left) and IKToM (right) projection for the gene data. T1 in triangles (red), T2+ in circles (green) and Ta in stars (blue).

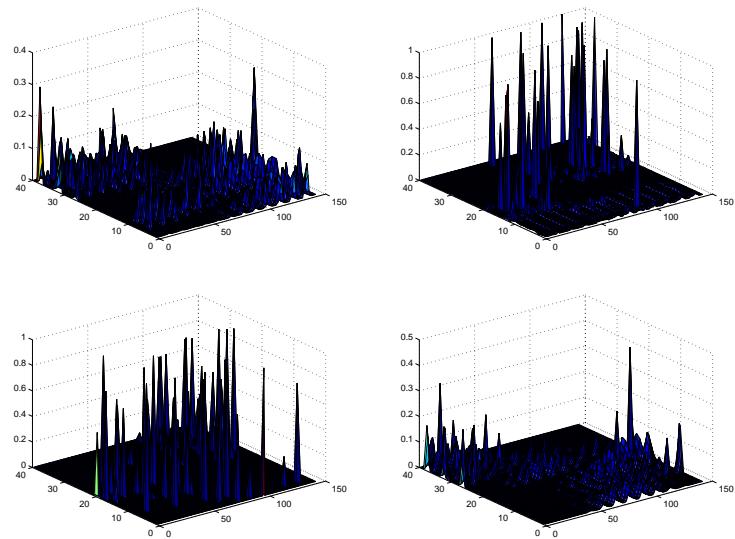


Fig. 5.3. Responsibilities for ToPoE, HaToM, ToNeGas and IKToM. In each diagram the latent points are on the right axis, the data points on the left and the vertical axis measures the responsibilities.

The hit histogram are formed by finding the Best Matching Unit (BMU) of each data sample from the map, and increasing a counter in a map unit each time it is the BMU. The hit histogram shows the distribution of the data set on the map. Here, the hit histogram for the whole data set is calculated and visualized with the U-matrix (Figs. 5.4, 5.5, 5.6).

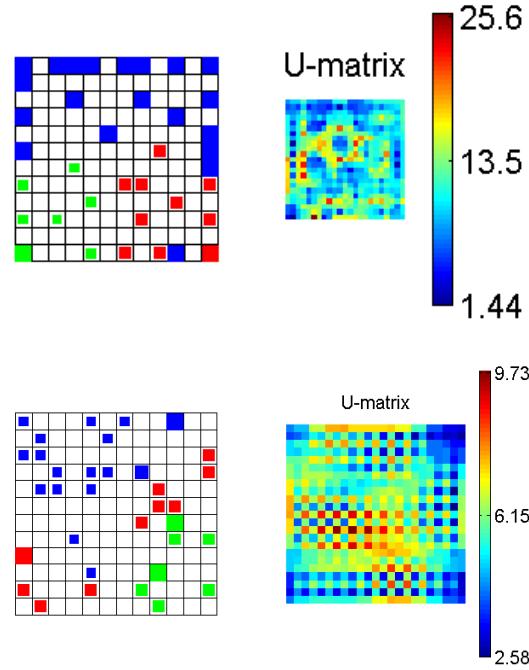


Fig. 5.4. Hit histogram and U-matrix for SOM (top) and ToPoE (bottom).

The hits histograms show that all SOM, ToPoE, HaToM, ToNeGas and IKToM have separate areas in the grids that are responsible for the different classes of genes (with higher distances in between clusters shown in the U-matrix), with only one blue point that appears as outlier for both of them.

Surface plot of distance matrix (Fig. 5.7): both color and vertical-coordinate indicate average distance to neighboring map units. This is closely related to the U-matrix.

The distance matrix is similar to the U-matrix and therefore gives similar results.

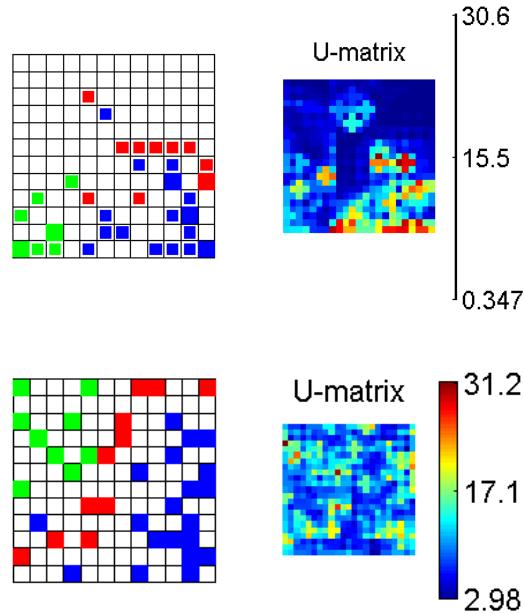


Fig. 5.5. Hit histogram and U-matrix for HaToM (top) and ToNeGas (bottom).

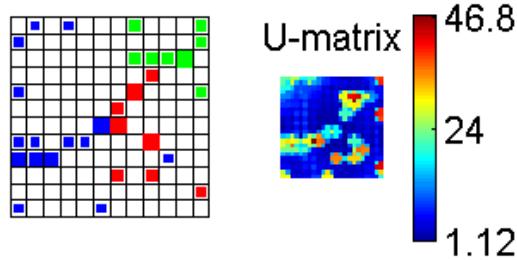


Fig. 5.6. Hit histogram and U-matrix for IKToM.

5.4.4 The Quality of The Map

Any topology preserving map requires a few parameters (such as size and topology of the map or the learning parameters) to be chosen a priori, and this influences the goodness of the mapping. Typically two evaluation criteria are used: resolution and topology preservation. If the dimension of the data set is higher than the dimension of the map grid, these usually become contradictory goals.

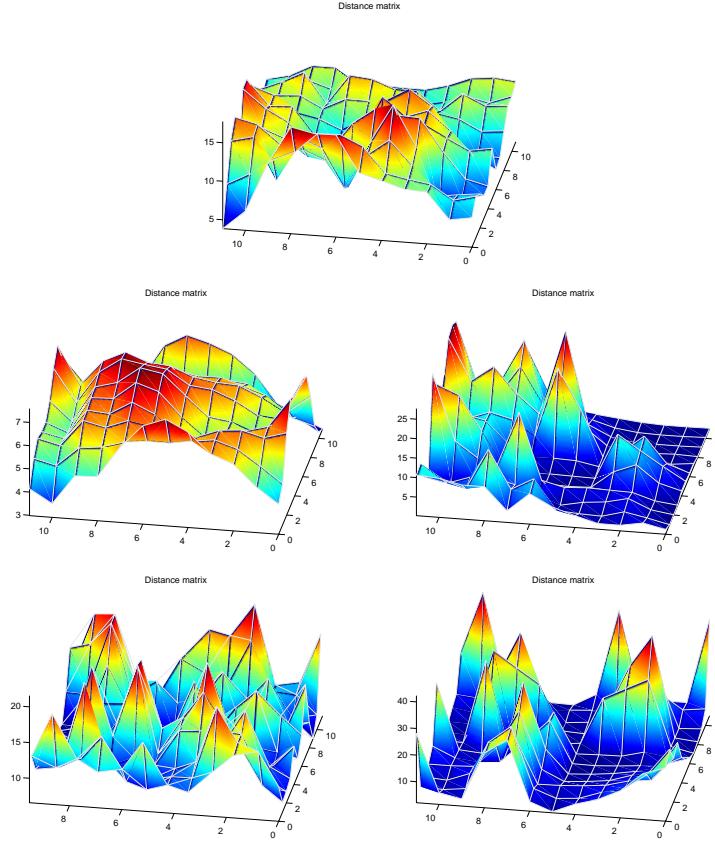


Fig. 5.7. Distance matrix for SOM, ToPoE, HaToM, ToNeGas, and IKToM.

We first analyze the quantization error for each datapoint with the distance to its Best Matching Unit (BMU). The mean quantization error q_e is the average distance between each data vector and its BMU; it measures then the resolution of the mapping.

$$q_e = \frac{1}{N} \sum_{i=1}^N \| \mathbf{x}_i - (\text{BMU}(i), k) \| . \quad (5.39)$$

ToPoE and HaToM are much worse than the other three mappings suggesting that their prototypes are further from the data.

The distortion measure which measures the deviation between the data and the quantizers is defined as:

$$E = \sum_{i=1}^N \sum_{k=1}^K h(\text{BMU}(i), k) \| \mathbf{m}_k - \mathbf{x}_i \|^2 . \quad (5.40)$$

We first calculate the total distortion for each unit, and then average for the total number of neurons.

Another important measure of the quality of the mapping is the topology preservation. In this case we calculate the topographic error, t_e , i.e. the proportion of all data vectors for which first and second BMUs are not adjacent units.

$$t_e = \frac{1}{N} \sum_{i=1}^N u(\mathbf{x}_i) \quad (5.41)$$

where $u(\mathbf{x}_i)$ is equal to 1 if first and second BMU are adjacent and 0 otherwise.

This t_e does not consider diagonal neighbors, thus the hexagonal case in SOM always gives lower values of t_e due to its six neighbors for each unit in comparison to the four in the rectangular mapping used for the other three algorithms. We may use a different topographic error, such as the Alfa error [1] which considers also the diagonal neighbors in the rectangular case (though now the rectangular mappings have an advantage in that they have 8 neighbours). The formula for the alpha error is as follows:

$$Alfa = \frac{1}{N} \sum_{i=1}^N \alpha(\mathbf{x}_i) \quad (5.42)$$

where $\alpha(\mathbf{x}_i)$ is equal to 1 if first and second BMU are adjacent or diagonals and 0 otherwise.

Table 5.1. Quantization error and topology preservation error with topology-preserving Mappings for the gene data.

Algorithm	SOM	ToPoE	HaToM	ToNeGas	IKToM
Map Size	(12*12)	(12*12)	(12*12)	(12*12)	(12*12)
Mean Quantization Error	11.8813	22.4106	22.3085	8.8433	13.7959
Average total distortion for each unit (e+003)	0.597	0.8924	1.3571	0.8514	1.1074
Topology preservation error	0.0250	0.2500	0.7500	0.2000	0.4500
Alfa error	0	0.0250	0.6750	0.1000	0.4000

The lower clustering errors are for ToNeGas, closely followed by SOM. The topology is completely preserved in SOM, but also quite well in ToPoE and ToNeGas. The other two have higher topology errors in this experiment.

5.5 Conclusions

We have developed four new Topology preserving mappings based on the Generative Topographic Mapping. Each algorithm applies a different clustering technique, providing the whole with particular advantages: HaToM can be

used in a data-driven or model-driven version, which are useful for different situations; both HaToM and ToNeGas overcome the problem of local minima with their clustering technique. ToPoE does not require a growing mode, while ToNeGas can apply the growing faster. Finally the Inverse Weighted K -Means has proven to improve situations where the initialisation of the prototypes are not randomly positioned. All four algorithms were applied to a high dimensional dataset, and all properly separated the clusters in the projection space.

References

1. Arsuaga-Uriarte, E. and Daz-Martn, F.: Topology preservation in SOM. *Transactions On Engineering, Computing And Technology*, **15**, 1305–5313 (2006)
2. Barbakh, W., Crowe, M., and Fyfe, C.: A family of novel clustering algorithms. In: 7th international conference on intelligent data engineering and automated learning, IDEAL2006 (2006)
3. Barbakh, W. and Fyfe, C.: Performance functions and clustering algorithms. *Computing and Information Systems*, **10** (2), 2–8 (2006) ISSN 1352-9404.
4. Bishop, C.M., Svensen, M., and Williams, C.K.I.: Gtm: The generative topographic mapping. *Neural Computation*, **10** (1), 215–234 (1997)
5. Bishop, C.M., Svensen, M., and Williams, C.K.I.: Magnification Factors for the GTM Algorithm. In: Proceedings of the IEE 5th International Conference on Artificial Neural Networks, Cambridge, U.K., 64–69P (1997)
6. Friedman, J., Hastie, T., and Tibshirani, R.: Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, **28**, 337–374 (2000)
7. Fyfe, C.: Two topographic maps for data visualization. *Mining and Knowledge Discovery* (2007)
8. Dyrskjot, L., Thykjaer, T., Kruhoffer, M. et al.: Identifying distinct classes of bladder carcinoma using microarrays. *Nat Genetics* **33** (1), 90–96 (2003)
9. Hamerly, G. and Elkan, C.: Alternatives to the k -means algorithm that find better clusterings. In: CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management, pages 600–607, New York, NY, USA, ACM Press (2002)
10. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Computational Neuroscience Unit, University College, London (2000)
11. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E.: Adaptive mixtures of local experts. *Neural Computation*, **3**, 79–87 (1991)
12. Jordan, M.I. and Jacobs, R.A.: Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, **6**, 181–214 (1994)
13. Kohonen, T.: Self-Organising Maps. Springer (1995)
14. Martinetz, T.M., Berkovich, S.G., and Schulten, K.J.: 'neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, **4** (4), 558–569 (1993)
15. Neural Networks Research Centre, Helsinki University of Technology. Som toolbox. www.cis.hut.fi/projects/somtoolbox.
16. Peña, M. and Fyfe, C.: Developments of the generalised harmonic topographic mapping. *WSEAS Transactions On Computers*, **4** (11), 1548–1555 (2005)

17. Peña, M. and Fyfe, C.: Model- and data-driven harmonic topographic maps. *WSEAS Transactions On Computers*, **4** (9), 1033–1044, (2005)
18. Peña, M. and Fyfe, C.: The topographic neural gas. In: *7th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL06.*, pages 241–249 (2006)
19. Tipping, M.E.: Topographic Mappings and Feed-Forward Neural Networks. PhD thesis, The University of Aston in Birmingham (1996)
20. Zhang, B.: Generalized k-harmonic means – boosting in unsupervised learning. Technical Report HPL-2000-137, HP Laboratories, Palo Alto, October (2000)
21. Zhang, B.: Generalized k-harmonic means– dynamic weighting of data in unsupervised learning. In: First SIAM international Conference on Data Mining (2001)
22. Zhang, B., Hsu, M., and Dayal, U.: K-harmonic means - a data clustering algorithm. Technical Report HPL-1999-124, HP Laboratories, Palo Alto, October (1999)

The Iterative Extraction Approach to Clustering

Boris Mirkin

School of Computer Science and Information Systems,
Birkbeck, University of London, UK,
mirkin@dcs.bbk.ac.uk

Summary. The Iterative Extraction approach (ITEX) extends the one-by-one extraction techniques in Principal Component Analysis to other additive data models. We describe additive models for clustering entity-to-feature and similarity data and apply ITEX for deriving computationally feasible clustering solutions. Specifically, two ITEX derived clustering methods, iK-Means and ADDI-S, are presented as well as update results on theoretical, experimental and applicational aspects of these methods.

6.1 Introduction

The iterative extraction approach emerged within the Principal Component Analysis (PCA) framework. The PCA builds aggregated features, “hidden factors” to score hidden capabilities of entities, relying on the Singular Value Decomposition of the data matrix and its mathematical properties: The singular vectors, that underlie the principal components, are mutually orthogonal, thus can be found and extracted one by one in the descending order of singular values. Moreover, the square of a singular value represents the share of the data scatter taken into account by the corresponding principal component.

The author extended this approach to clustering in [30, 31, 32, 33, 34] by extending the bilinear Singular Value Decomposition model to that of clustering. Specifically, the “scoring” principal components sought are compulsory restricted to be binary, thus representing cluster memberships rather than scores. This extension proved to be reasonable since it encompasses many a popular method including K -Means clustering. An analogous extension, of the spectral decomposition of a square symmetric semi-positive matrix applied to similarity data also brings forward clusters that are provably tight.

The ITEX approach applied in this setting finds clusters one by one, which has a number of advantages as well as drawbacks. Among advantages are the following: (i) computational efficiency, (ii) the possibility of the additive decomposition of the data scatter in such a way that the

contribution of each cluster, as well as of its elements, can be clearly evaluated, (iii) the easiness of getting overlapping clusters by not bothering to take into account the clusters already found. This approach will be abbreviated further on as ITEX; in [32, 33] it was referred to as SEFIT. Some methods emerging within ITEX, such as iK-Means and ADDI-S, are described in the further text. The data scatter decompositions proved effective in extending clustering methods to categorical and mixed scale data as well as for interpretation of the results (see [34]). Here we concentrate on reviewing methods and some applications, and do not consider usage of the decompositions.

The paper is organized as follows. Section 2 describes the ITEX approach for the entity-to- feature data and methods of Anomalous Pattern and iK-Means clustering following from it. Method iK-Means can be considered as a version of the conventional K -Means in which the number of clusters nor cluster seeds are not pre-specified but found sequentially by extracting “principal” clusters one by one. An experiment over generated data is described involving a number of different approaches to choosing the “right” number of clusters published in the literature. The experiment demonstrates that the iK-Means is superior to other methods, especially if supplemented with Hartigan’s rule for choosing the cluster “discarding threshold.” Section 3 describes the ITEX applied to additive structuring and clustering models over similarity data. The material clearly demonstrates that there have been a bunch of heuristic similarity clustering methods proposed that nicely fit into the framework. A concept that appears to be crucial in this models is the intercept that also can be interpreted as a similarity scale shift or the similarity threshold. Within the ITEX, its value becomes as important as the number of clusters in conventional approaches. With the scale shift specified to be either user-defined or optimal, the ITEX leads to intuitive and fast clustering methods. The final method, ADDI-S, in which all parameters are least-squares adjusted, produces provably tight clusters involving a variable similarity threshold, equal to half the average similarity within the cluster. This method proved useful in applications, three of which are described in brief.

6.2 Clustering Entity-to-feature Data

6.2.1 Principal Component Analysis

The method of iterative extraction extends the process of a major data analysis tool, the Principal Component Analysis (PCA). Typically, PCA is presented as a heuristic data extraction method [8, 22]. Observed data such as marks of students $i \in I$ at academic disciplines labelled by $v = 1, \dots, V$ constitute a data matrix $X = (x_{iv})$. This matrix is pre-processed by centering and rescaling its columns, after which a normed $|I|$ -dimensional “scoring” vector z is sought such that the linear combination $c = X^T z$ takes into account the maximum possible share of the data scatter $Tr(X^T X)$. This problem reduces

to finding the maximum eigenvalue λ_1 and corresponding normed eigenvector c_1 of the covariance matrix $S = X^T X$, $Sc_1 = \lambda_1 c_1$, so that the solution is $z_1 = Xc_1/\sqrt{\lambda_1}$ and λ_1 is the share of the data scatter taken into account by it. Vectors z_1 and c_1 form what is referred to the first Principal Component “scoring” and “loading” vectors, respectively. The second component is found with the same process, but applied to the residual matrix $S' = S - \lambda_1 c_1 c_1^T$ rather than matrix $S = X^T X$ itself. The second Principal Component corresponds to the second largest eigenvalue, its eigenvector being orthogonal to the first one. The process can be reiterated to find the third, the fourth, etc. mutually orthogonal components. The Principal Components are interpreted as “better” features, bearing the maximum possible share of the data scatter, so that a few of them can approximate all the data.

In this narrative, PCA is but a heuristic method for the iterative extraction of the principal components. In fact, as is rather well known, PCA can be considered a method for fitting the model presented in equation (6.1) below.

Assume that each entry x_{iv} reflects the i -th entity scores (the student’s hidden abilities) z_{ik} ($i \in I$) along with feature v impact coefficients c_{kv} , over a number of hidden factors $k = 1, \dots, K$, so that

$$x_{iv} = c_{1v} z_{i1} + \dots + c_{Kv} z_{iK} + e_{iv} \quad (6.1)$$

for all $i \in I$ and $v = 1, \dots, V$, or, in the matrix algebra notation,

$$X = Z_K C_K + E. \quad (6.2)$$

We are interested in finding the least squares solution to equation (6.1) – (6.2), that is, matrices Z_K and C_K minimizing the sum of squared elements of the residual matrix E . What is nice in this formulation is that the components are not defined as linear combinations of X columns, nor they are supposed to be normed; and the criterion is but the conventional statistical approximation of the observed data by the “ideal” data produced by the model.

The least-squares solution is defined only up to a K -dimensional linear subspace of the space of N -dimensional vectors, whose base is formed by columns of matrix Z_K . The optimal linear subspace can be specified in terms of the so-called singular value decomposition (SVD) of matrix X , typically after it is standardized. In fact, matrices Z_K and C_K , whose columns are the first K singular vectors of X , form orthonormal bases of the least-squares optimal subspaces. They can be found by iterative application of the same process of obtaining just one principal component of matrix X by least-squares fitting the one-factor model

$$x_{iv} = c_v z_i + e_{iv} \quad (6.3)$$

with respect to unknown vectors c and z . At each k -th step of the process, matrix X is substituted by the residual data matrix calculated by subtraction of the current component matrix $\mu_k z_k^T c_k$ from the previous X . (The traditional assumptions of SVD are assumed here: μ_k is k -th singular value of X ; z_k , c_k are the normed versions of the singular vectors corresponding to μ_k .)

The conventional process of extracting eigenvectors from $S = X^T X$ described above can be considered an implementation of this method since computation of singular vectors can be performed not necessarily with the matrix X but also with its derivative matrices: $V \times V$ matrix $X^T X$ or $|I| \times |I|$ matrix XX^T , because z_k is an eigen vector of XX^T and c_k an eigen vector of $X^T X$ corresponding to their respective k -th eigen-values, both equal to μ_k^2 ($k = 1, 2, \dots, K$).

6.2.2 Additive Clustering Model and ITEX

Assuming that the score vectors z_1, z_2, \dots, z_K are restricted to be 0/1 binary, the model (6.1) can be reinterpreted as a clustering model [31, 32]. According to this model, binary vector z_k is the membership vector for cluster $S_k \subseteq I$ so that $z_{ik} = 1$ if $i \in S_k$ and $z_{ik} = 0$ if $i \notin S_k$. Vector c_k is a representation of cluster k in the feature space so that every data row $x_i = (x_{iv})$ approximately equals the sum of representative vectors c_k over all such k that $i \in S_k$. In the case when cluster sets S_k are mutually disjoint, that is, vectors z_k are mutually orthogonal, any row x_i approximates just one representative vector c_k .

The clustering problem according to model (6.1) is similar to that of PCA: given matrix X and number K , find binary z_k and real c_k minimizing a prespecified monotonely growing function of the residuals e_{iv} .

For the least-squares criterion, the one-by-one extracting strategy ITEX here builds clusters S_k one by one, each time minimizing one-cluster criterion of the model (6.3):

$$l = \sum_{i \in I} \sum_{v \in V} (x_{iv} - c_v z_i)^2 \quad (6.4)$$

over unknown c_v and binary z_i , index k being omitted. The membership vector z is characterized by subset $S = \{i : z_i = 1\}$. Criterion (6.4) can be rewritten in terms of S :

$$W(S, c) = \sum_{i \in S} d(x_i, c) + \sum_{i \notin S} d(x_i, 0), \quad (6.5)$$

where d is the Euclidean distance squared, $d(x, y) = \sum_j (x_j - y_j)^2$, and x_i is i -th row of the residual data matrix.

This is a conventional clustering square error criterion [20, 34] for a partition consisting of two clusters, S and its complement $\bar{S} = I - S$, with regard to their respective centroids, c and 0. However, in contrast to conventional clustering formulations, the centroid 0 here is not the gravity center of the complementary set $I - S$, but is being kept constant and does not change when S and $I - S$ change.

Given S , the optimal c in (6.5) is obviously the center of gravity of S because the first sum is minimum at that c and the second sum does not depend on c . Given c , a subset S to minimize (6.5) must include every $i \in I$ such that $d(x_i, c) < d(x_i, 0)$. These properties immediately give rise to the following

implementation of the alternating minimization algorithm for criterion (6.5) [34].

1. *Pre-processing.* Specify a reference point $a = (a_1, \dots, a_n)$ (this can be the data grand mean) and standardize the original data table using the reference point coordinates as shift parameters a_v . (This way, the space origin is shifted into a .) In the case when feature scales significantly differ, the standardization should also involve rescaling the feature scales (see details in [34]).
2. *Initial setting.* Put a tentative centroid, c , as an entity which is the most distant from the origin, 0. [This minimizes (6.5) with respect to all singleton clusters.]
3. *Cluster update.* Determine cluster list S around c against the only other “centroid” 0 with the Minimum distance rule so that y_i is assigned to S if $d(y_i, c) < d(y_i, 0)$.
4. *Centroid update.* Calculate the within S mean c' and check whether it differs from the previous centroid c . If c' and c do differ, update the centroid by assigning $c \leftarrow c'$ and return to Step 3. Otherwise, go to 5.
5. *Output.* Output list S and centroid c , with accompanying interpretation aids, as the most anomalous pattern.

The process is illustrated on Figure 6.1.

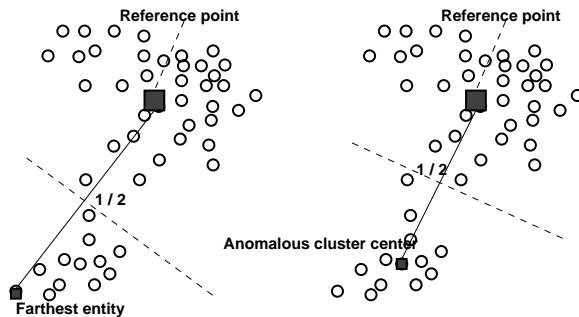


Fig. 6.1. Extracting an “Anomalous Pattern” cluster with the reference point in the gravity center: the initial iteration is on the left and the final one on the right.

The AP method is a reference-point based version of the popular clustering method K -Means in which:

- (i) the number of clusters K is 2;
- (ii) centroid of one of the clusters is forcibly kept at the 0 reference point through all the iterations;
- (iii) the initial centroid of the anomalous cluster is taken as an entity point which is the most distant from 0.

6.2.3 Overlapping and Fuzzy Clustering Case

The ITEX can be easily applied to the case of overlapping clusters. After one cluster has been found, the next one can be sought by applying the same AP method to the matrix of residuals $X - zc^T$, in a manner similar to that in PCA.

When clusters in the feature space are well separated from each other or the cluster structure can be thought of as a set of differently contributing clusters, the clusters can be found with the iterative application of Anomalous Pattern algorithm that would mitigate the need for pre-setting the number of clusters and their initial centroids.

Consider, for example, the setting in Table 6.1 which represent a set of entities of which one fourth are ideally located in $c1$, five eighths in $c2$, and the remaining one eighth is assigned to the summary point $c1 + c2$. These represent two clusters comprising $3n$ and $6n$ entities, respectively, in such a way that their intersection consists of n entities. This setting, designed according to [7], perfectly fits into model (6.1) with all residuals being zero. The first three rows of the Table 6.1 can be considered a shortcut for a data table comprising $N = 8n$ entities and 2 features that may have only patterns presented in its lines one ($2n$ entities), two ($5n$ entities) and three (n entities); the fourth line represents the feature values at the grand mean of the data.

Table 6.1. Two overlapping ideal clusters presented by their centroids: $c1$, the first cluster; $c2$, the second cluster; $c1 + c2$, their overlap; c , the grand mean. The column on the right represents the relative numbers of entities 2:5:1, in the first cluster short of the overlap, the second cluster short of the overlap, and the overlap, respectively.

Notation	Feature I	Feature II	Quantity
$c1$	12	2	$2n$
$c2$	-1	-2	$5n$
$c1 + c2$	11	0	n
Mean c	2.75	-0.75	

To apply ITEX clustering to this data, let us pre-process each column by subtracting the corresponding component of the grand mean c and dividing by the range (13 for feature I, 4 for feature II) afterwards. This transforms $2n$ entities at $c1$ to $c1' = (0.64, 0.69)$, $5n$ entities at $c2$ to $c2' = (-0.36, -0.31)$ and n entities at $c1 + c2$ to $c3' = (0.56, 0.19)$, which is not the sum of the former two anymore. Still $c3'$ lies not too far from $c1'$, which is picked up by AP as the anomalous starting point. These two make the first AP cluster, which exactly coincides with Cluster 1 in Table 6.1. The next cluster can be found using the residual data matrix that can be computed by finding the centroid of the first AP cluster, $c' = (2c1' + c3')/3 = (0.61, 0.52)$ and subtracting it from all the elements in the AP cluster, thus leading to $c1'' = c1' - c' = (0.03, 0.17)$ and $c3'' = c3' - c' = (-0.05, -0.33)$. Now the anomalous seed is $c2'$ that has been remained intact. Of the two residual centroids, $c1''$ and $c3''$, the former is closer to 0 while the latter to $c2'$. This brings Cluster 2 as the second AP,

and leaves all the entries in the next residual matrix close to zero, accounting for less than 5% of the initial standardized data scatter.

Unfortunately, when cluster contributions are less different, ITEX may be not as successful and fail to properly identify the clusters, even in a similarly “ideal” situation when all residuals are zero [7]. For example, with the same data entries as in Table 6.1 but proportions of the entities being equal to 1:1:1 rather than 2:5:1 as in the Table, the ITEX with AP clustering will find not two but three clusters, corresponding to each of the three parts under consideration: the overlapping part of two clusters and the clusters’ parts short of the overlap.

In general, the ITEX AP may tend to produce non-overlapping parts of clusters rather than those in the data structure. This may suggest that this method should be used for disjoint rather than overlapping clustering, which is described in next section. Nonetheless, even in the overlapping case, the ITEX method can provide a useful initialization for other algorithmic strategies such as based on the follow-up iterations of centroid and cluster updates [7]. On the other hand, one may think that the situation may be alleviated if fuzzy rather than crisp belongingness vectors z_k are involved. Indeed, this would make the summary centroid more similar to the averaged one, since the individual cluster centroids would be weighted by memberships that should sum up to unity over the set of centroids. Such a possibility was considered by Mirkin and Satarov [35, 32]. The model (6.1) considered with fuzzy vectors z_k leads to an interesting concept of ideal types. Indeed, with the condition that vectors z_k are not negative and sum up to unity ($k = 1, 2, \dots, K$), all entity points should be convex combinations of centroids c_1, c_2, \dots, c_K , which are thus supposed to be the extreme points of a convex polytope covering the entity set rather than within cluster averages. Applied as is, this may lead to rather wild “ideal type” centroids possibly located quite far away from the entities. A reasonable modification of model (6.1) leading to better fitting fuzzy clusters was proposed and experimentally explored in [42]; this however, utilizes finding all clusters in parallel rather than sequentially. As shown in [42], the modified model retains, in a weaker form, the “ideal type” property, which can be useful in some applications.

6.2.4 *K*-Means and iK-Means Clustering

K-Means is one of the most popular clustering methods. It iteratively updates a set of cluster centroids by assigning to them their closest entities and finding the gravity centers of thus obtained clusters. An issue of *K*-Means is a mandatory setting of the *K* and initial seeds for the centroids. Properties of the Anomalous Pattern algorithm mitigate the issue of determining appropriate initial seeds, which allows using it for finding an initial setting for *K*-Means.

Some other potentially useful features of the method relate to its flexibility with regard to dealing with outliers and the “swamp” of inexpressive, ordinary, entities situated around the grand mean.

iK-Means

0. *Setting.* Put $k = 1$ and I_k the original entity set. Specify a threshold of resolution to discard all AP clusters whose cardinalities are less than the threshold.
1. *Anomalous pattern.* Apply AP to I_k to find S_k and c_k .
2. *Control.* If Stop-condition (see below) does not hold, put

$$I_{k+1} \leftarrow I_k - S_k \text{ and } k \leftarrow k + 1,$$

and go to Step 1.

3. *Removal of small clusters.* Remove all of the found clusters that are smaller than a pre-specified cluster *discarding threshold* for the cluster size. Denote the number of remaining clusters by K and their centroids by c_1, \dots, c_K .
4. *K-Means.* Do K -Means with c_1, \dots, c_K as initial seeds.

The Stop-condition in this method can be any or all of the following:

1. **All clustered.** $S_K = I_K$ so that there are no unclustered entities left.
2. **Large cumulative contribution.** The total contribution of the first K AP clusters to the data scatter has reached a pre-specified threshold such as 60 %.
3. **Small cluster contribution.** Contribution of k -th AP cluster is too small, say, compared to the order of average contribution of a single entity, $1/N$, where N denotes the total number of entities.
4. **Number of clusters reached.** Number of clusters, k , has reached a pre-specified value K .

The first condition is natural if the data consists of “natural” clusters, that indeed differ in their contributions to the data scatter. The second and third conditions can be considered as imposing certain degrees of resolution, reflected in the contribution thresholds, with which the user looks at the data.

At step 4, K -Means can be applied to either the entire dataset or to the set from which the smaller clusters have been removed. This may depend on the application domain: in some problems, such as structuring of a set of settlements for better planning or monitoring, no entity should be left out of the consideration, whereas in other problems, such as developing synoptic descriptions for text corpora, some deviant texts should be left out of the coverage.

An extensive set of experiments to test how well iK-Means recovers the “true” number of clusters have been described in [28, 29]. These experiments involved the data generated according to a mixture of Gaussians distributions,

several other methods for finding the “right” number of clusters, and a number of evaluation criteria, that are briefly described below.

1. Data generation.

The Gaussian mixture distribution data are generated as random samples using the functions in Neural Network NetLab, which are applied as implemented in a MATLAB Toolbox freely available on the web [13]. Our sampling scheme is based on a modified version of that proposed in Wasito and Mirkin (2006) utilizing two geometrically meaningful parameters: the clusters spread (the average distance between cluster centroids) and spatial cluster size which is proportional to the norm of the covariance matrix. We use either of two types of covariance structure: the ordinary spherical shape or the probabilistic principal component analysis (PPCA) shape [49]. The spatial cluster sizes are taken constant at the spherical shape, and variant at the PPCA shape. We maintain two types of the spatial cluster size by scaling covariance matrices using factors that are either proportional to the clusters index k (the linear distribution of sizes) or its square k^2 (the quadratic distribution of sizes) ($k = 1, 2, \dots, K$). Cluster centroids are generated randomly from a normal $N(0, 1)$ distribution with the follow-up scaling them by a factor expressing spread of the clusters. In the experiments, we generate data sets of size 1000×15 with the hidden dimension 6 involving two K settings (7 and 9 clusters), two spreads (small and large) and three types of distributions of cluster “diameters” (equal, k -proportional and k^2 proportional).

2. Methods involved.

We distinguish between four different approaches to the problem of determining the “right” number of clusters of which one is our ITEX approach and the other three involve statistics calculated at random trials of K -Means at different K in a prespecified range, typically from $K = 2$ to $K = 12 - 20$. Given K value, a random initialization of K centroids is generated (within the features’ ranges) and K -Means applies until convergence. After a number of runs, set to 100 in our experiments, for a specified statistic s , the statistic’s value s_K is calculated. Then the best K , according to the statistic, is selected. The three approaches we found in the literature involve statistics based on variance, structure and consensus as follows.

a) Variance based statistics.

These are formulated in terms of the clustering criterion

$$W = \sum_{k=1}^K \sum_{i \in S_k} d(i, c_k),$$

where S_k is the k -th cluster, c_k its centroid and $d(i, c_k)$ the Euclidean distance squared between i -th row of the data matrix and c_k . Obviously, W is the summary weighted within-cluster variance of the features.

Probably the earliest was Hartigan's criterion $H = (W_K/W_{K+1} - 1)(N - K - 1)$, where W_K is the minimum value of W at the results of a number of runs of the algorithm at given K and random initial settings. The criterion is based on the following intuition. Let there be a natural clustering of K_0 clusters in data. Then, at $K < K_0$, K -Means would tend to produce a clustering that aggregates some of the 'natural' clusters so that increasing K by 1 would just separate a cluster or cluster aggregate from the K -cluster clustering so that the relative change $(W_K - W_{K+1})/W_{K+1}$ would be relatively small. If however $K > K_0$, then the clusterings will tend to be the K 'natural' clusters randomly subdivided into smaller chunks. Thus, the very first K at which the relative difference becomes large enough, making $H \geq 10$ [16], should be the right number K_0 . Somewhat similar reasoning lies behind the Calinski and Harabasz's Fisher-like criterion [6]. We also utilize the so-called Jump statistic based on the maximum jump in value of $M_K^{V/2}$ where M_K is K -Means criterion computed according to Mahalanobis distances and divided by V ; this is proven to be the case if the data are generated according to a mixture of Gaussian distributions (see [48]).

b) *Structure based statistic.*

We utilize the popular average silhouette width introduced in [24] to reflect, for every entity, the difference between its within cluster distances and distances to the closest of other clusters.

c) *Consensus based statistics.*

In contrast to the other approaches, this one utilizes results of all, say M , runs of K -Means from random initialisations at a given K , not just the best of them. Monti et al. [40] proposed using the distribution of entries in the so-called consensus matrix that can be defined after a set of M runs of K -Means. This is an entity-to-entity similarity matrix, whose (i, j) -th entry is the number of those of the M clusterings in which i and j belong to the same cluster. In the ideal case, all clusterings coincide, which would make the distribution of the consensus matrix entries to be binomial. We use the jumps of two related indexes, one measuring the area under the cumulative distribution function proposed in [40] and the other, the average distance between clusterings that can be expressed in terms of the distribution's variance [34].

The ITEX clustering, in our experiments, was represented by two algorithms, one based on the least-squares criterion described above, the other based on the least modules criterion, thus differing in that the median is taken instead of the average, and the citi-block distance instead of the Euclidean squared distance. The cluster discarding threshold is taken to be 1.

3. Evaluation criteria.

The number of clusters generated is fixed at $K = 7$ or 9 , which is easy to compare with the number of clusters found at any method utilized. We also evaluate two aspects of a clustering, the intensional and extensional ones, by measuring the differences between cluster centroids and cluster contents.

To compare centroids of found clusters with those generated, in the situation when the numbers of clusters may differ, we utilize the following strategy. Let the generated clustering have K clusters and that found one K' clusters so that $K < K'$. We first one-to-one assign each of the K clusters with the closest one from the K' clusters, using the between-centroid distance as the criterion; after that we assign each of the remaining $K' - K$ clusters to that of the K clusters that are closest in terms of the between-centroid distances. Then we calculate the average distance between linked centroids either weighted by the cluster cardinalities or not. The weighted distance, in our experiments, appears to be orthogonal to other evaluation measures and thus, dropped off as an unworthy one.

To compare cluster contents between two partitions, we utilize four different measures of (dis)similarity between partitions: the distance [34], the adjusted Rand index [18], the Tchouproff coefficient and the averaged relative cluster-to-cluster overlap [34]. The four are highly correlated and they all support the general findings in [28, 29].

According to the experiments, the Hartigan statistics based method shows the best performance in terms of the number of clusters, though not in terms of centroids and cluster contents. In terms of the similarities between generated and found centroids and partitions, in most cases, the ITEX based methods performed better than the rest, and the least-squares ITEX somewhat better than that least-modules ITEX. Further analysis suggests that, most likely, ITEX results are inferior to those by other methods (typically, these are the silhouette width or jump statistics based methods that can be superior sometimes) in the cases in which ITEX based methods produce too many clusters.

This has led us to the following adjustment of the ITEX clustering methods. First, produce the H based evaluation of the number of clusters K_H . Second, if iK-Means leads to much more clusters than K_H , increase the cluster discarding threshold until the number of iK-Means clusters becomes reasonably close to K_H . In further experiments, with this adjustment, iK-Means clustering results on average were superior to all other methods in consideration [29].

6.3 ITEX Structuring and Clustering for Similarity Data

6.3.1 Similarity Clustering: a Review

The following review of the subject is reminiscent to that in [38].

Let $A = (a_{ij})$ be a symmetric matrix of similarities (or, synonymously, proximities or interactions) between entities $i, j \in I$. The greater the value of

a_{ij} , the greater is the similarity between i and j . A cluster is a set of highly similar entities whose similarity to entities outside of the cluster is low.

Similarity clustering emerged quite early in graph theory, probably before the discipline of clustering itself. A graph may be thought of as a structural expression of similarity data, its nodes corresponding to entities with edges joining similar nodes. Cluster related graph-theoretic concepts include: (a) *connected component* (a maximal subset of nodes in which every pair of nodes is connected by a path), (b) *bicomponent* (a maximal subset of nodes in which each pair of nodes belongs to a cycle), and (c) *clique* (a subset of nodes in which each pair of nodes is connected by an edge).

Other early clustering concepts include the B-coefficient method for clustering variables using their correlation matrix [17] and the Wroclaw taxonomy [9]. These are precursors to the ADDI and ADDI-S methods [31], described later, and the single linkage method [15, 14], respectively.

Two more recent graph-theoretic concepts are also relevant: *maximum density subgraph* [11] and *min-multi-cut* in a weighted graph [12].

The density $g(S)$ of a subgraph $S \subset I$ is the ratio of the number of edges in S to the cardinality of S . For an edge weighted graph with weights specified by the matrix $A = (a_{ij})$, the density $g(S)$ is equal to the *Raleigh quotient* $s^T As / s^T s$, where $s = (s_i)$ is the characteristic vector of S , viz. $s_i = 1$ if $i \in S$ and $s_i = 0$ otherwise. A subgraph of maximum density represents a cluster. After removing such a cluster from the graph, a maximum density subgraph of the remaining graph can be found. This may be repeated until no “significant” clusters remain. Such an incomplete clustering procedure is natural for many types of data, including protein interaction networks. However, to our knowledge, this method has never been applied to such problems, probably because it involves rather extensive computations. A heuristic analogue can be found in [2]. We consider that the maximum density subgraph problem is of interest because it is a reasonable relaxation of the maximum clique problem and fits well into data recovery clustering (see section 6.3.3). The maximum value of the Raleigh quotient of a symmetric matrix over any real vector s is equal to the maximum eigenvalue and is attained at an eigenvector corresponding to this eigenvalue. This gives rise to *spectral clustering*, a method of clustering based on first finding a maximum eigenvector s^* and then defining the spectral cluster by $s_i = 1$ if $s_i^* > t$ and $s_i = 0$ otherwise, for some threshold t . This method may have computational advantages when A is sparse. Unfortunately, it does not necessarily produce an optimal cluster [33], but empirically it produces good clusters in most cases.

The concept of min-multi-cut is an extension of the max-flow min-cut concept in capacitated networks, and essentially seeks a partition of nodes into classes having minimum summary similarities between classes or, equivalently, maximum summary similarities within classes. When similarities are non-negative, this criterion may often lead to a highly unbalanced partition with one huge class and a number of singleton classes. This line of research has led to using the *normalized cut*, proposed in [45], as a meaningful clustering

criterion. The normalized cut criterion assumes that the set I should be split into two parts, S and \bar{S} , so that the normalized cut

$$nc(S) = a(S, \bar{S})/a(S, I) + a(S, \bar{S})/a(\bar{S}, I)$$

is minimized. Here $a(S, T)$ denotes the summary similarity between subsets S and T . The criterion $nc(S)$ can be expressed as a Raleigh quotient for a generalized eigenvalue problem [45], so the spectral clustering approach may be applied to minimizing the normalized cut too.

It should be noted that the user typically finds it meaningful, in the framework of domain knowledge, to define a similarity threshold α , such that entities i and j should be aggregated if $a_{ij} > \alpha$ but not if $a_{ij} < \alpha$. When this is the case, the data should be pre-processed to take the threshold into account. There are two different ways of implementing this idea: (i) by zeroing all similarities a_{ij} that are less than α , or (ii) by shifting the zero similarity to α by subtracting α from each similarity a_{ij} . The former is popular, for example, in image analysis because it makes the similarity data sharper and sparser. However, we favour the latter as better fitting in with the additive structure recovery models presented later. In fact, the similarity shift originated from these models (see, for example, [30, 31]).

6.3.2 The additive structuring model and ITEX

To represent a set of structures assumed to underly the similarity matrix A , we use the terminology of binary relations. A binary relation on the set I can be defined by a $(0,1)$ matrix $R = (r_{ij})$ such that $r_{ij} = 1$ if i and j are related and $r_{ij} = 0$ otherwise. Partitions, rankings and subsets can be represented by equivalence, order and square relations, respectively. A quantitative expression of the intensity of a relation can be modelled by a real value λ . So a relation of intensity λ is represented by the product λR .

Given a set of binary relations \mathcal{R} defined by a general property (for example, equivalence or order relations), an additive structuring model for a given $N \times N$ matrix $A = (a_{ij})$ is defined by the equations

$$a_{ij} = \sum_{k=0}^K \lambda_k r_{ij}^k + e_{ij}, \quad \text{for all } i, j \in I, \quad (6.6)$$

where $R^k = (r_{ij}^k) \in \mathcal{R}$ and λ_k is the intensity of R^k ; the number of relations $K+1$ in (6.6) is typically assumed to be much smaller than $|I|$, the cardinality of I . The goal is to minimize the residuals e_{ij} with respect to the unknown relations R^k and intensities λ_k . In some problems, the intensities λ_k may be given, based on substantive or model considerations.

To minimize the residuals in (6.6), the least-squares criterion can be applied again. This criterion brings in an important property. The data matrix A is not necessarily symmetric. However, in the situations in which all relations

in \mathcal{R} are symmetric the matrix A can be equivalently substituted by symmetric matrix $\tilde{A} = (A + A^T)/2$ where A^T denotes A transposed. Indeed, for any given set $R^k \in \mathcal{R}$ ($k = 1, \dots, K$) let us take any pair $i, j \in I$ and denote the sum in (6.6) by λ . Then the contribution of the pair i, j to the sum of squared residuals, $\sum e_{ij}^2$, will be equal to $(a_{ij} - \lambda)^2 + (a_{ji} - \lambda)^2 = f - 2\lambda(a_{ij} + a_{ji}) + \lambda^2$ where $f = a_{ij}^2 + a_{ji}^2$. If we change a_{ij} in (6.6) for $\tilde{a}_{ij} = (a_{ij} + a_{ji})/2$, then the contribution of i, j to the summary quadratic criterion will be $2(\tilde{a}_{ij} - \lambda)^2 = \tilde{f} - 4\lambda\tilde{a}_{ij} + \lambda^2$ where $\tilde{f} = 2\tilde{a}_{ij}^2 \leq f$. Since $\tilde{a}_{ij} = (a_{ij} + a_{ji})/2$, both expressions have the same variable parts, which proves that any least-squares solution to (6.6) remains a least-squares solution after a_{ij} is changed for \tilde{a}_{ij} at all $i, j \in I$.

In certain cases, we may require one of the relations R^k to be the universal relation, for which $r_{ij}^k = 1$ for all $i, j \in I$. The corresponding intensity λ_k then plays a role in the model (6.6) similar to that of the intercept in linear regression. Conventionally, we relabel the universal relation as R^0 and denote its matrix by $\mathbf{1}$. The intercept value λ_0 may be interpreted as a similarity shift, with the shifted similarity matrix $A' = (a'_{ij})$ defined by $a'_{ij} = a_{ij} - \lambda_0$. Equation (6.6) for the shifted model has a'_{ij} on the left and the sum on the right starting from $k = 1$.

With the least-squares criterion, we can employ again the greedy heuristic of extracting the relations R^k one by one in order to reduce the amount of computation and have a useful decomposition of the data scatter over found relations. This may be particularly useful if the relations R^k contribute very unequally to the data, for example, when the λ_k vary significantly. At step k , for $k = 0, 1, 2, \dots, K$, we find R^k using an algorithm for minimizing

$$L^2(R) = \sum_{i,j \in I} (a_{ij}^k - \lambda r_{ij})^2 \quad (6.7)$$

over $R \in \mathcal{R}$ and λ (unless pre-specified). The residual similarity matrix $A^k = (a_{ij}^k)$ is updated after step k by subtracting $\lambda_k R^k$ from it. At the start, $A^0 = A$ and, at the end, $A^{K+1} = (e_{ij})$, the matrix of residuals.

Given R , the optimal value of λ is equal to the average similarity a_{ij}^k over all related pairs (i, j) , i.e. those for which $r_{ij} = 1$. The complexity of this minimization problem depends on the type of relations in \mathcal{R} . Therefore, in some cases, we only find a local minimum of (6.7).

When the λ_k are not pre-specified, then, at each step, the residual similarity matrix is orthogonal to the relation extracted. This implies the following Pythagorean decomposition [32, 33]:

$$\sum_{i,j \in I} (a_{ij})^2 = \sum_{k=0}^K \lambda_k^2 \sum_{i,j \in I} r_{ij}^k + \sum_{i,j \in I} e_{ij}^2. \quad (6.8)$$

This equation additively decomposes the data scatter into the contributions of the extracted relations R^k (“explained” by the model) and the minimised residual square error (the “unexplained” part). The decomposition (6.8)

makes it possible to prove that the residual part converges to zero under relatively mild and easily checked assumptions on the solutions found at each iteration [32, 33].

Obviously, our convention implies that, when the ITEX is to be applied to the shifted model, the universal relation R^0 must be extracted first. In this case, the optimal value of λ_0 will be equal to \bar{a} , the average of the similarities in A .

Also, the property that matrix A can be equivalently substituted by its symmetrized version $\tilde{A} = (A + A^T)/2$ if \mathcal{R} consists of symmetric relations, holds when the ITEX is utilized.

6.3.3 Additive clustering model

A square relation $r = (r_{ij})$ is defined by a subset $S \subseteq I$ in such a way that $r_{ij} = 1$ if both i and j belong to S and $r_{ij} = 0$, otherwise. In other words, $r_{ij} = s_i s_j$ where $s = (s_i)$ is the membership vector so that for any $i \in I$, s_i is 1 or 0 depending on whether $i \in S$ or not.

By restricting \mathcal{R} to consist of all square relations, the shifted version of the model (6.6) becomes what is referred to as the additive clustering model [44]. The universal relation $R^0 = \mathbf{1}$, used in the shifted model, is the square relation corresponding to the universal cluster I .

When we assume that the similarities in A are generated by a set of “additive clusters” $S^k \subseteq I$, $k = 0, 1, \dots, K$, in such a way that each a_{ij} approximates the sum of the intensities of those clusters that contain both i and j , the shifted version of (6.6) becomes:

$$a_{ij} = \sum_{k=1}^K \lambda_k s_i^k s_j^k + \lambda_0 + e_{ij}, \quad (6.9)$$

where $s^k = (s_i^k)$ are the membership vectors of the unknown clusters S^k , $k = 1, 2, \dots, K$, and e_{ij} are the residuals to be minimised. In this model, introduced in [44], the intensities λ_k , $k = 1, 2, \dots, K$, and the shift λ_0 also have to be optimally determined. In the more general formulation of the “categorical factor analysis” [30, 31], these values may be user specified.

We note that the role of the intercept λ_0 in (6.9) is three-fold: it can be considered as

1. an intercept of the bilinear model, similar to that in the linear regression
or
2. the intensity of the universal cluster I or
3. a ‘soft’ similarity threshold in the sense that it is the shifted similarity matrix a'_{ij} , rather than the original A , is used to determine the clusters S^k , $k = 1, 2, \dots, K$. This role is of a special interest when λ_0 is user specified.

When the one-by-one ITEX strategy is applied to fitting (6.9) with none of the λ s pre-specified, the data scatter decomposition (6.8) holds for the

optimal values of λ_k . In this case, λ_k is equal to \bar{a}_k , the average of the residual similarities a_{ij}^k for $i, j \in S^k$. Substituting $s_i^k s_j^k$ for r_{ij}^k and \bar{a}_k for λ_k , (6.8) can be written in the form:

$$(A, A) = \sum_{k=0}^K [s^{kT} A^k s^k / s^{kT} s^k]^2 + (E, E). \quad (6.10)$$

The inner products (A, A) and (E, E) denote the sums of the squares of the elements of the matrices, considering A and E as vectors; these are conventionally expressed as the traces (sums of diagonal elements) of the products $A^T A$ and $E^T E$, respectively.

6.3.4 Approximate partitioning

In this section, we restrict the additive clustering model to nonoverlapping clusters.

If clusters S^k , $k = 1, \dots, K$, are mutually disjoint (so the membership vectors s^k are mutually orthogonal), the optimal intensity λ_k depends only on the elements a'_{ij} , $i, j \in S^k$, of the shifted matrix $A' = A - \lambda_0 \mathbf{1}$ and not on the residual matrix A^k . The following decomposition of A' corresponding to (6.10) then holds and is independent of the the order of the clusters.

$$(A', A') = \sum_{k=1}^K [s^{kT} A' s^k / s^{kT} s^k]^2 + (E, E). \quad (6.11)$$

Although similar in form to the decomposition for A in (6.10), this decomposition for A' differs in that: (i) the terms in the summation involve the original matrix A' , not the residual matrix, and (ii) the summation starts from 1, not 0.

Since $A' = A - \lambda_0 \mathbf{1}$, it follows that

$$(A, A) = 2\lambda_0(\bar{a} - \lambda_0/2)(\mathbf{1}, \mathbf{1}) + \sum_{k=1}^K [s^{kT} A' s^k / s^{kT} s^k]^2 + (E, E). \quad (6.12)$$

When λ_0 is not pre-specified and must be found according to the least-squares criterion, its optimal value, found by differentiating (6.12) with respect to λ_0 , is:

$$\lambda_0 = \frac{\sum_{i,j \in I} a_{ij}(1 - s_{ij})}{\sum_{i,j \in I}(1 - s_{ij})}, \quad (6.13)$$

where $s_{ij} = \sum_{k=1}^K s_i^k s_j^k$ (so $s_{ij} = 1$ if both i and j belong to S^k for some $k = 1, 2, \dots, K$ and $s_{ij} = 0$ otherwise).

Thus, the optimal λ_0 is the average of similarities a_{ij} for i and j belonging to different clusters.

Equation (6.11) is analogous to the representation of the trace of $A'^T A'$ as the sum of the squares of the eigenvalues of A' because the terms are squares of the Raleigh quotients

$$g(s^k) = s^{kT} A' s^k / s^{kT} s^k . \quad (6.14)$$

which are attained at zero/one rather than arbitrary vectors s^k .

According to (6.11), an optimal partition with weights λ_k adjusted according to the least-squares criterion must maximize the sum of the cluster contributions $g^2(s^k)$, that is,

$$\sum_{k=1}^K g^2(s^k) = \sum_{k=1}^K \left(\sum_{i,j \in S^k} a'_{ij} / N_k \right)^2 , \quad (6.15)$$

where $N_k = |S^k|$, the cardinality of S^k .

An “unsquared” version of this criterion comes from applying the data recovery approach to an entity-to-feature data matrix in section 1, which leads to

$$\sum_{k=1}^K g(S^k) = \sum_{k=1}^K \sum_{i,j \in S^k} a_{ij} / N_k , \quad (6.16)$$

as the contribution of the clusters to the entity-to-feature data scatter. The similarity a_{ij} is defined, in this approach, as the inner product of the feature vectors corresponding to entities i and j . In matrix terms, if Y is an entity-to-feature data matrix then A is defined as $A = YY^T$. The difference between criteria (6.15) and (6.16) is somewhat similar to that between the spectral decomposition of $A = YY^T$ and singular-value decomposition of Y .

In contrast to (6.16), criterion (6.15) has never been analysed, neither theoretically nor experimentally.

To illustrate the difference between preset and optimal values of the shift λ_0 when model (6.9) is used for approximate partitioning, let us consider the similarity data between eight entities in Table 6.2.

For $\lambda_0 = 2$, the only positive values of $a'_{ij} = a_{ij} - \lambda_0$ are within clusters 1-2-3, 4-5, and 6-7-8 plus similarities between entity 4 and both 6 and 7. These positive extra-cluster similarities lead to differences in the clustering if λ_0 is changed. At the average similarity shift $\lambda_0 = \bar{a} = 1.49$, these three clusters with respective intensities 3.46, 3.13 and 3.70 form the optimal partition. This partition contributes 37.1% to the original data scatter. For the globally optimal partition, the $\lambda_0 = 0.49$ and entity 4 joins the cluster 6-7-8. The optimal partition then consists of clusters 1-2-3 (with intensity 4.47), 4-6-7-8 (with intensity 3.17), and singleton 5 (since self-similarity is not defined, the intensity has no meaning). This contributes 65.6% of the data scatter. The rather large difference between the two contributions to the data scatter is mainly due to the difference in the first term on the right-hand side of (6.12) involving λ_0 .

Table 6.2. Illustrative similarities between eight entities; self-similarity is not defined.

Entity	1	2	3	4	5	6	7	8
1	-	4.33	5.60	-0.20	-0.16	-0.21	-0.49	0.17
2	4.33	-	4.93	0.79	0.06	1.22	-0.10	-0.45
3	5.60	4.93	-	0.21	0.79	-1.20	-0.15	0.80
4	-0.20	0.79	0.21	-	4.62	3.29	2.80	0.32
5	-0.16	0.06	0.79	4.62	-	-1.00	0.25	-0.08
6	-0.21	1.22	-1.20	3.29	-1.00	-	5.96	4.38
7	-0.49	-0.10	-0.15	2.80	0.25	5.96	-	5.23
8	0.17	-0.45	0.80	0.32	-0.08	4.38	5.23	-

6.3.5 One cluster clustering

Applying ITEX to the additive clustering involves extracting a single cluster from, possibly residual, similarity data presented in the form of a symmetric matrix A , assuming that any required shift λ_0 has already been made. As noted above, if A is not symmetric, it can be equivalently changed for symmetric $\tilde{A} = (A + A^T)/2$. For the sake of simplicity, in this section, we assume that the diagonal entries a_{ii} are all zero.

Pre-specified intensity

We first consider the case in which the intensity λ of the cluster to be found is pre-specified. Noting that $s_i^2 = s_i$ for any 0/1 variable s_i , criterion (6.7) can be expressed as

$$L^2(S) = \sum_{i,j \in I} (a_{ij} - \lambda s_i s_j)^2 = \sum_{i,j \in I} a_{ij}^2 - 2\lambda \sum_{i,j \in I} (a_{ij} - \lambda/2) s_i s_j. \quad (6.17)$$

Since $\sum_{i,j} a_{ij}^2$ is constant, for $\lambda > 0$, minimizing (6.17) is equivalent to maximizing the summary within-cluster similarity after subtracting the threshold value $\pi = \lambda/2$:

$$f(S, \pi) = \sum_{i,j \in I} (a_{ij} - \pi) s_i s_j = \sum_{i,j \in S} (a_{ij} - \pi). \quad (6.18)$$

This criterion implies that, for an entity i to be added to or removed from the S under consideration, the difference between the value of (6.18) for the resulting set and its value for S , $f(S \pm i, \pi) - f(S, \pi)$, is equal to $\pm 2f(i, S, \pi)$ where

$$f(i, S, \pi) = \sum_{j \in S} (a_{ij} - \pi) = \sum_{j \in S} a_{ij} - \pi|S|.$$

This gives rise to a local search algorithm for maximizing (6.18): start with $S = \{i^*, j^*\}$ such that $a_{i^* j^*}$ is maximum element in A , provided that $a_{i^* j^*} > \pi$.

An element $i \notin S$ may be added to S if $f(i, S, \pi) > 0$; similarly, an element $i \in S$ may be removed from S if $f(i, S, \pi) < 0$. The greedy procedure ADDI [31] iteratively finds an $i \notin S$ maximizing $+f(i, S, \pi)$ and an $i \in S$ maximizing $-f(i, S, \pi)$, and takes the i giving the larger value. The iterations stop when this larger value is negative. The resulting S is returned along with its contribution to the data scatter, $4\pi \sum_{i \in S} f(i, S, \pi)$. The following version of ADDI reducing the dependence on the initial S proved successful in experiments. The computations here start from the singleton $S = \{i\}$, for each $i \in I$, so that N ADDI based results are generated; of these, that cluster S is selected that contributes most to the data scatter, i.e., that minimizes the square error $L^2(S)$ (6.17). In fact, the set of resulting clusters should be of interest on its own since many of them coincide or almost coincide and the structure of not coinciding clusters represents an overlapping structure of the similarity data.

The heuristic algorithm CAST [3], popular in bioinformatics, is in fact a version of the ADDI algorithm, because it uses the same iterative process of adding or removing an entity by utilizing criterion $\sum_{j \in S} a_{ij} > \pi|S|$, for the case of adding, with the $\sum_{j \in S} a_{ij}$ referred to as the affinity of i to S – which is equivalent to criterion $f(i, S, \pi) > 0$.

Another property of the criterion is that $f(i, S, \pi) > 0$ if and only if the average similarity between a given $i \in I$ and the elements of S is greater than π , which means that the final cluster S produced by ADDI/CAST is rather tight: the average similarities between $i \in I$ and S is at least π if $i \in S$ and no greater than π if $i \notin S$ [31].

Intuitively, changing the threshold π should lead to corresponding changes in the optimal S . Indeed, it has been proven that the greater π is, the smaller S will be [31].

Optimal intensity

When λ in (6.17) is not fixed but chosen to further minimize the criterion, it is not difficult to prove that

$$L^2(S) = (A, A) - [s^T As / s^T s]^2, \quad (6.19)$$

in line with the decomposition (6.11), with $K = 1$ and $L^2(S) = (E, E)$. The proof is based on the fact that the optimal λ is the average similarity $a(S)$ within S , i.e.,

$$\lambda = a(S) = s^T As / [s^T s]^2, \quad (6.20)$$

since $s^T s = |S|$.

The decomposition (6.19) implies that the optimal cluster S must maximize the criterion

$$g^2(S) = [s^T As / s^T s]^2 = a^2(S)|S|^2. \quad (6.21)$$

According to (6.21), the maximum of $g^2(S)$ may correspond to either positive or negative value of $a(S)$. The latter case may emerge when the similarity

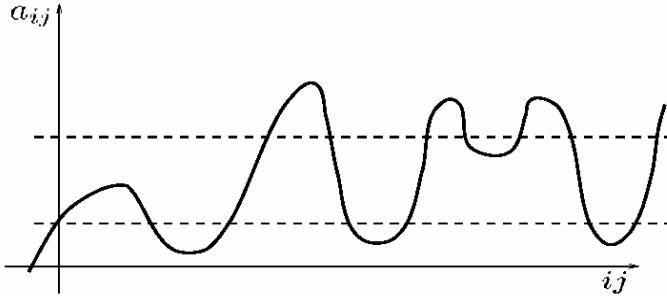


Fig. 6.2. A pattern of clustering depending on the subtracted similarity shift λ_0 . shift λ_0 is large and corresponds to S being the so-called *anti-cluster* [33]. In this paper, we do not consider this case, but focus on maximizing (6.21) for positive $a(S)$ only. This is equivalent to maximizing the Raleigh quotient,

$$g(S) = s^T As / s^T s = a(S)|S|. \quad (6.22)$$

To maximize $g(S)$, one may utilize the ADDI-S algorithm [31], which is the same as the algorithm ADDI/CAST, described above, except that the threshold π is recalculated after each step as $\pi = a(S)/2$, corresponding to the optimal λ in (6.20).

A property of the resulting cluster S , similar to that for the constant threshold case, holds: the average similarity between i and S is at least half the within-cluster average similarity $a(S)/2$ if $i \in S$, and at most $a(S)/2$ if $i \notin S$.

To obtain a set of (not necessarily disjoint) clusters within the framework of the additive clustering model, one may use ITEX by repeatedly extracting a cluster S using ADDI-S and then replacing A by the residual matrix $A - a(S)ss^T$.

We can apply this method to the partitioning problem, by repeatedly using ADDI-S to find a cluster S and then removing from consideration all the entities in S . The process stops when the similarity matrix on the remaining entities has no positive entries. The result is a set of non-overlapping clusters S_k , $k = 1, \dots, K$, each assigned with its intensity $a(S_k)$ and contribution to the data scatter $g^2(S_k)$, and also the remaining unclustered entities in I .

ADDI-S utilizes no ad hoc parameters, so the number of clusters is determined by the process of clustering itself. However, changing the similarity shift λ_0 may affect the clustering results, which can be of advantage in contrasting within- and between- cluster similarities. Figure 6.2 demonstrates the effect of changing a positive similarity a_{ij} to $a'_{ij} = a_{ij} - \lambda_0$ for $\lambda_0 > 0$; small similarities $a_{ij} < \lambda_0$ are transformed into negative similarities a'_{ij} .

6.3.6 Some applications

For the similarity data, the ITEX may lead to relevant overlapping clusters using similarity data. In our experience, the ITEX produced meaningful over-

lapping clusters in the situations in which the model of additive similarities was applicable. Here we briefly review three applications.

Elementary meanings in sorting experiments

A sorting experiment, in psycho-linguistics, goes as follows [43]. A number of nouns expressing concepts related to an aspect of the real world, such as the “kin” or “kitchenware” are put down on a separate card each, and respondents are requested to sort the cards into groups according to subjective similarity of the concepts. The extent of similarity between two concepts is expressed then by the number of respondents who put the concepts into the same group (“consensus” similarity). These similarities reflect the semantic similarities within the community represented by the respondents. Considering that an elementary meaning can be expressed as a cluster of certain intensity, it is reasonable to suggest that the similarity between two concepts should be equal to the sum of the intensities of those clusters that contain both of them. For example, the similarity between kinship concepts “son” and “father” should sum up intensities of elementary meanings such as “Nuclear family” and “Male relatives”. Therefore, the additive clustering model should be applicable here.

The iterative one-by-one extraction with ADDI-S algorithm has been applied for finding out additive clusters underlying a sorting consensus similarity matrix several times. In the analysis of similarities between 72 kitchenware terms, it was found that none of the clusters reflected logical or structural similarities between the kitchenware items; all the clusters related to the usage. Specifically, three types of communality were represented by the clusters: (i) a common process, such as frying or boiling; (ii) a common consumption use, such as drinking or eating, and (iii) a common situation such as a banquet [10].

Kim and Rosenberg data of sorting 15 kinship terms, observed six times [43], have been analyzed with the iterative ADDI-S adapted to the three-way data type in [33], p. 223. The clusters, in general, supported previously published findings such as clusters of “male relatives” or “female relatives”, but also added more subtle groupings such as “aunt, uncle”. The group “brother, daughter, father, mother, sister, son” that had been interpreted as “nuclear family” was further divided into “daughter, father, mother, son” and “brother, sister”, which some might view as more elementary groupings expressing the concepts of “nuclear family” and “siblings”, respectively.

Subject clusters in profiling a research organization

Profiling is a relatively new activity in computation, related to finding such features in data that are relevant to a pre-specified list of properties. The data may be rather unstructured, such as a text or set of texts. Profiling can be done rather conveniently with respect to a taxonomy or ontology in which all properties of interest are clearly delineated and well structured. The

paper [39] specifically refers to the ACM classification of Computer Sciences (ACMC) [1] that can be used for profiling a Computer Science department. The ACM classification is organized as a three layer tree. The first layer items are: A. General Literature, B. Hardware, C. Computer Systems Organization, D. Software, E. Data, F. Theory of Computation, G. Mathematics of Computing, H. Information Systems, I. Computing Methodologies, J. Computer Applications, and K. Computing Milieux. They are further subdivided into the second layer items such as I.5 Pattern Recognition. The third layer comprises further divisions such as I.5.3. Clustering.

The method proposed in [39] maps the set of subjects that are investigated in a research department onto the ACM classification and involves:

1. Selecting the level of classification to be used as the baseline.
2. Measuring similarities between selected ACMC topics according to the research activities of the department in question.
3. Decomposing the similarity structure by finding topic clusters that are not necessarily disjoint. Here the additive cluster model seems appropriate, thus ADDI-S applicable.
4. Mapping topic clusters to the ACM classification and highlighting the *head subjects*, *offshoots* and *gaps* revealed. A head subject of a topic cluster is the ACMC subject of a higher layer, whose “children” in the classification tree belong to the cluster, with gaps being those children that do not belong to the topic cluster.

As an example, we considered all 59 specific topic items of the second layer of the ACM classification, of which 26 have been covered by the research going on in the department under consideration [39]. The similarity between two topics was measured as just the number of academics pursuing both topics in their research. Application of ADDI-S to the similarity matrix leads to six clusters with contributions not less than $1/N = 1/26 = 4\%$. Five of the clusters mainly fall within the corresponding five head subjects, with very few gaps and offshoots to other ACMC nodes. One of the clusters, however, covers two of the head subjects which come on top of two other subject clusters, each pertaining to just one of the head subjects, D. Software or H. Information Systems. This can be interpreted as an indication that the two-headed cluster signifies a new direction in Computer Sciences, combining D and H into a single new direction, which seems to be a feature of the current developments in Computer Sciences unifying software and information systems indeed.

Aggregate protein families

This is an example in which a partition, not a set of potentially overlapping clusters, is sought, with the concept of similarity used to analyze complex objects, such as protein families. Proteins belonging to different organisms are combined into a family if they perform the same function and are considered as orthologous, that is, inherited from a common ancestor and being similar

because of that. Two features of general interest should be noted in the relation to this application: (i) using similarity between neighbourhoods rather than the objects themselves, and (ii) interplay with knowledge domain for getting a “right” similarity shift value [38].

The usage of neighbourhoods is convenient for complex objects. Similarity between complex objects is relatively straightforward to measure when they are similar indeed: the differences can be captured by superpositioning one object over another. When the similarity decreases, however, finding a correct superposition becomes rather tricky and subject to local search and arbitrary parameter values. This is why it is convenient to represent a complex object by its neighbourhood, which is the set of entities that are similar to the object (the idea first proposed in [47]). When the neighbourhoods are defined, two objects, $i, j \in I$, can be compared by comparing their neighbourhoods $L(i)$ and $L(j)$ with a convenient similarity measure between sets. The most popular between-set similarity measure is the Jaccard coefficient, sometimes referred to as Tanimoto’s coefficient, equal to $n/(n_1 + n_2 - n)$ where n_1 , n_2 and n are cardinalities of the neighbourhoods and their intersection, respectively. The ratio relates the cardinalities of the overlap and the set theoretic union of the neighbourhoods. This coefficient, however, suffers from an intrinsic flaw of systematically underestimating the similarity [36].

The most natural indexes would be the relative sizes of the overlap $\frac{n}{n_1}$ and $\frac{n}{n_2}$, but they are not symmetric and are avoided by the researchers because of this. However, in the context of additive clustering, these can be used anyway because they can be equivalently converted to their average, $mbc = \frac{1}{2}(\frac{n}{n_1} + \frac{n}{n_2})$, as proven in section 3.2. The use of mbc index alleviates the issues of Jaccard-Tanimoto’s coefficient [36].

To define an appropriate similarity shift value, families with known function have been selected and, of those, family pairs have been put into two categories: (I) those whose function is clearly the same (86 pairs), and (II) those whose function clearly differ (279 pairs) [37, 38]. The pair-wise similarities should be high in the category (I) and low in the category (II), so that any intermediate value could be taken as the scale shift. It appears, the two distributions of similarities in this application are not disjoint; some proteins with the same function have very weak similarities. Therefore, two most likely shift values have been chosen: (i) that at which the distributions overlap (0.42) thus minimizing the rate of error in deciding which pairs should have positive and which negative similarity, and (ii) that at which none of the proteins have different functions (0.67). The final decision is made by comparing scenarios of evolution of the aggregate families with the knowledge of gene arrangement. It appears, both the shift values lead to similar clusterings, but the shift of 0.42 provides a cluster whose reconstructed history is more consistent with other knowledge than does the shift of 0.67.

Conclusion

Iterative extraction in clustering is a powerful approach from both theoretical and practical points of view. We tried to demonstrate that with two data formats, quantitative entity-to-feature data and similarity data, and two cluster structures, overlapping clusters and non-overlapping clusters.

The ITEX approach has been applied to other data formats, such as co-occurrence or mixed scale data, too; and a number of other cluster structures were utilized in different applications; the hierarchical cluster structures have been shown to be treatable with ITEX as well [33]. Applying the approach to other, yet not tried, data formats such as temporal and/or spatial data could be of interest.

The Pythagorean decomposition of the data scatter into explained and unexplained parts, pro-intuitive cluster properties, and fast computation are among advantages of the ITEX. Its shortcomings are related to: (i) the compulsory additive structure of the underlying model and (ii) unequal contributions of the underlying clusters. The former is probably behind the lack of intersection among ITEX clusters at the entity-to-feature data. Indeed, the idea of summing up centroids to represent intersections of corresponding clusters may be somewhat odd in some contexts.

The mentioned shortcomings suggest a number of directions for the theoretical and experimental investigation into the ITEX: (i) extending it to non-additive clustering models, (ii) characterization of data structures that are reliably treatable with the ITEX, (iii) development of different criteria for the ITEX, not necessarily based on minimizing residuals in the one-cluster models.

References

1. The ACM Computing Classification System (1998), <http://www.acm.org/class/1998/ccs98.html>.
2. Bader, G.D. and Hogue, C.W.V.: An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2 (2003)
3. Ben-Dor, A., Shamir, R., and Yakhini, Z.: Clustering gene expression patterns. *Journal of Computational Biology*, 6, 281-297 (1999)
4. Benzecri, J.P.: Correspondence Analysis Handbook. Marcel Dekker, New York (1992)
5. Brohée, S. and van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks *BMC Bioinformatics*, 7:488 (2006) (<http://www.biomedcentral.com/1471-2105/7/488>).
6. Calinski, T. and Harabasz, J.: A Dendrite Method for Cluster Analysis. *Communications in Statistics*, 3 (1), 1-27 (1974)
7. Depril, D., Van Mechelen, I., and Mirkin, B.: Algorithms for additive clustering of rectangular data tables submitted. (2007)

8. Everitt, B.S. and Dunn, G.: Applied Multivariate Data Analysis. Arnold, London (2001)
9. Florek, K., Lukaszewicz, J., Perkal, H., Steinhaus, H., and Zubrzycki, S.: Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicum*, **2**, 282–285 (1951)
10. Frumkina, R.M., Mikheev, A.V.: Meaning and Categorization. Nova Science Publishers, Commack, N.Y. (1996)
11. Gallo, G., Grigoriadis M.D., and Tarjan R.E.: A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, **18**, 30–55 (1989)
12. Garg, N., Vazirani, V.V., and Yannakakis, M.: Approximate Max-Flow Min-(Multi) Cut theorems and their applications. *SIAM Journal on Computing*, **25** (2), 235–251 (1996)
13. Generation of Gaussian mixture distributed data 2006, NETLAB neural network software, <http://www.ncrg.aston.ac.uk/netlab>.
14. Gower, J.C. and Ross, G.J.S.: Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, **18**, 54–64 (1969)
15. Hartigan, J.A.: Representation of similarity matrices by trees: *J. Amer. Stat. Assoc.*, **62**, 1140–1158 (1967)
16. Hartigan, J.A.: *Clustering Algorithms*, J. Wiley & Sons, New York (1975)
17. Holzinger, K.J. and Harman, H.H.: *Factor Analysis*, University of Chicago Press, Chicago (1941)
18. Hubert, L.J. and Arabie, P.: Comparing Partitions. *Journal of Classification*, **2**, 193–218 (1985)
19. Inkpen, D., and Desilts, A.: Semantic similarity for detecting recognition errors in automatic speech transcripts. In: Conference on Empirical Methods in Natural Language Processing. Vancouver, Canada (2005)
20. Jain, A.K. and Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, NJ (1988)
21. Jarvis, R.A. and Patrick E.A.: Clustering using a similarity measure based on shared nearest neighbors. *IEEE Trans. Comput.*, **22**, 1025–1034 (1973)
22. Jolliffe, I.T.: Principal Component Analysis. Springer, New York, (1986)
23. Kawaji, H., Takenaka, Y., and Matsuda, H.: Graph-based clustering for finding distant relationships in a large set of protein sequences, *Bioinformatics*, **20** (2), 243–252 (2004)
24. Kaufman, L. and Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley & Sons, New York (1990)
25. Krzanowski, W. and Lai, Y.: A criterion for determining the number of groups in a dataset using sum of squares clustering. *Biometrics*, **44**, 23–34 (1985)
26. McLachlan, G. and Basford, K.: Mixture Models: Inference and Applications to Clustering. Marcel Dekker, New York (1988)
27. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Fifth Berkeley Symposium on Mathematical Statistics and Probability, II, 281–297 (1967)
28. Chiang, M. M.-T. and Mirkin, B.: Determining the number of clusters in the straight K -Means: Experimental comparison of eight options. In: Proceedings of United Kingdom Computational Intelligence Workshop, Leeds, 143–150 (2006)
29. Chiang, M. M.-T. and Mirkin, B.: Initializing K -Means clustering: An experimental study, submitted (2007)
30. Mirkin, B.: Analysis of Categorical Features, Finansy i Statistika Publishers, Moscow (1976) (In Russian)

31. Mirkin, B.: Additive clustering and qualitative factor analysis methods for similarity matrices. *Journal of Classification*, **4**, 7–31 (1987); Erratum. **6**, 271–272 (1989)
32. Mirkin, B.: A sequential fitting procedure for linear data analysis models. *Journal of Classification*, **7**, 167–195 (1990)
33. Mirkin, B.: *Mathematical Classification and Clustering*. Kluwer Academic Press, Dordrecht (1996)
34. Mirkin, B.: *Clustering for Data Mining: A Data Recovery Approach*. Chapman and Hall, Boca Raton (2005)
35. Mirkin, B. and Satarov, G.: Method of fuzzy additive types for analysis of multidimensional data. *Autom. Remote Control*, I, II, **51** (5, 6), 683–688 (1990)
36. Mirkin, B. and Koonin, E.: A top-down method for building genome classification trees with linear binary hierarchies. In: Janowitz, M., Lapointe, J.-F., McMorris, F., Mirkin, B., and Roberts, F. (eds.) *Bioconsensus*, DIMACS Series, vol. 61, AMS, Providence: 97–112 (2003)
37. Mirkin, B., Camargo, R., Fenner, T., Loizou, G., and Kellam, P.: Aggregating homologous protein families in evolutionary reconstructions of herpesviruses. In: Ashlock, D. (ed.) *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. Piscataway NJ, 255–262 (2006)
38. Mirkin, B., Camargo, R., Fenner, T., Loizou, G., and Kellam, P.: Using domain knowledge and similarity scale shift in clustering, submitted (2007)
39. Mirkin, B., Nascimento, S., Moniz Pereira, L.: Using ACM classification for profiling a research organisation, submitted (2007)
40. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus Clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, **52**, 91–118 (2003)
41. Murtagh, F.: *Correspondence Analysis and Data Coding with JAVA and R*. Chapman & Hall/CRC, Boca Raton, FL (2005)
42. Nascimento, S., Mirkin, B., and Moura-Pires, F.: Modeling proportional membership in fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, **11** (2), 173–186 (2003)
43. Rosenberg, S.: The method of sorting in multivariate research with applications selected from cognitive psychology and person perception. In: Hirschberg, N. and Humphreys, L.G. (eds.) *Multivariate Applications in the Social Sciences*, University of Illinois at Urbana-Champaign: L. Erlbaum Assoc., 117–142. (1982)
44. Shepard, R.N. and Arabie, P.: Additive clustering: representation of similarities as combinations of overlapping properties. *Psychological Review*, **86**, 87–123 (1979)
45. Shi, J. and Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22** (8), 888–905 (2000)
46. Smid, M., Dorssers, L.C.J., and Jenster, G.: Venn Mapping: clustering of heterologous microarray data based on the number of co-occurring differentially expressed genes. *Bioinformatics*, **19** (16), 2065–2071 (2003)
47. Small, H.: Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, **24**, 265–269 (1973)
48. Sugar, C.A. and James, G.M.: Finding the number of clusters in a data set: An information-theoretic approach. *Journal of American Statistical Association*, **98** (463), 750–778 (2003)

49. Tipping, M.E. and Bishop, C.M.: Probabilistic principal component analysis. *J. Roy. Statist. Soc. Ser. B* **61**, 611–622 (1999)
50. Wasito, I. and Mirkin, B.: Nearest neighbours in least-squares data imputation algorithms with different missing patterns, *Computational Statistics & Data Analysis* **50**, 926–949 (2006)

Representing Complex Data Using Localized Principal Components with Application to Astronomical Data

Jochen Einbeck¹, Ludger Evers², and Coryn Bailer-Jones³

¹ Department of Mathematical Sciences, Durham University, Science Laboratories, South Road, Durham DH1 3LE, United Kingdom,
`jochen.einbeck@durham.ac.uk`

² Department of Mathematics, University of Bristol, University Walk, BS8 1TW, United Kingdom,
`l.evers@bris.ac.uk`

³ Max-Planck-Institut für Astronomie, Königstuhl 17, 69117 Heidelberg, Germany,
`calj@mpia-hd.mpg.de`

Summary. Often the relation between the variables constituting a multivariate data space might be characterized by one or more of the terms: “nonlinear”, “branched”, “disconnected”, “bended”, “curved”, “heterogeneous”, or, more general, “complex”. In these cases, simple principal component analysis (PCA) as a tool for dimension reduction can fail badly. Of the many alternative approaches proposed so far, local approximations of PCA are among the most promising. This paper will give a short review of localized versions of PCA, focusing on local principal curves and local partitioning algorithms. Furthermore we discuss projections other than the local principal components. When performing local dimension reduction for regression or classification problems it is important to focus not only on the manifold structure of the covariates, but also on the response variable(s). Local principal components only achieve the former, whereas localized regression approaches concentrate on the latter. Local projection directions derived from the partial least squares (PLS) algorithm offer an interesting trade-off between these two objectives.

We apply these methods to several real data sets. In particular, we consider simulated astrophysical data from the future Galactic survey mission Gaia.

7.1 Introduction

Principal component analysis is a well established tool for dimension reduction. For data $\mathbf{X} = (\mathbf{x}'_1, \dots, \mathbf{x}'_n)'$ with $\mathbf{x}_i \in \mathbb{R}^p$ the principal components provide a sequence of best linear approximations to it. Let $\boldsymbol{\Sigma}$ be the empirical covariance matrix of \mathbf{X} , then the principal components are given by the eigen decomposition

$$\boldsymbol{\Sigma} = \boldsymbol{\Gamma} \boldsymbol{\Lambda} \boldsymbol{\Gamma}' , \quad (7.1)$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p)$ is a diagonal matrix containing the ordered eigenvalues of Σ , with $\lambda_1 \geq \dots \geq \lambda_p$, and Γ is an orthogonal matrix. The columns of $\Gamma = (\gamma_1, \dots, \gamma_p)$ are the eigenvectors of Σ and are called *principal component loadings*. The first loading γ_1 maximizes the variance of the scores $\mathbf{X}\gamma$ over all $\gamma \in \mathbb{R}^p$ with $\|\gamma\| = 1$, the second loading γ_2 maximizes the variance of $\mathbf{X}\gamma$ over all $\gamma \in \mathbb{R}^p$ with $\|\gamma\| = 1$ which are orthogonal to γ_1 , and so on.

As an example, we consider a speed-flow diagram recorded on the freeway 4-W in Contra Costa County, California (figure 7.1(a)).⁴ The displayed points correspond to the average speed and flow over 5 minute intervals. The data were collected on 10th December 2006; this was a Sunday, so the traffic flow was quite low, not exceeding 1300 vehicles per hour. Most cars went at a speed close to the speed limit. One can easily imagine a first principal component line $\mathbf{g}(\eta) = \bar{\mathbf{x}} + \eta \cdot \gamma_1$ (with overall mean $\bar{\mathbf{x}}$) fitted through the data cloud, capturing the main part (here: 99.54%) of the variance in this data set (figure 7.1(a)). Clearly, the projection indices $\eta_i = (\mathbf{x}_i - \bar{\mathbf{x}})^T \gamma_1$ of the data projected onto this line are good indicators for the positions of the data points within the data cloud.

However, the application of principal component analysis postulates implicitly some form of linearity. More precisely, one assumes that the data cloud is directed, and that the data points can be well approximated by their projections onto the line \mathbf{g} corresponding to the first principal component (or in general the affine hyperplane corresponding to the first d principal components). This implicit assumption of linear PCA can already be violated in very simple situations. As an example, consider the speed-flow diagram recorded three days earlier with the same detector at the same location. This was a Thursday and traffic was busy, leading to occasional congestion. As a consequence, cars often had to reduce their speed at higher flows, resulting in the data cloud shown in figure 7.1(b). This kind of pattern has frequently been reported in the transportation science literature, see e.g. [27].

These data are somewhat half-moon shaped and it does not seem to make much sense to speak of some principal direction for this data. The projection indices onto any straight line like the first principal component (figure 7.1(b)) would be uninformative with respect to the position of the data within this bent data cloud. This will certainly get worse if the the data cloud is strongly twisted, if it has crossings, if it consists of several branches, if the curvature permanently changes, or if there are several disconnected clouds, and so on.

Solutions to problems of this kind are readily available, and fall into two major categories: *Nonlinear principal component analysis* (hereafter: NLPCA) and *principal curves* (or their multivariate extension, *principal manifolds*). The dividing lines between these two approaches are often rather fuzzy, but the following vague rule can be used to distinguish between them. Principal curves is a nonparametric extension of linear PCA, while NLPCA is a parametric (or

⁴ The data are taken from the database PemS [45].

semi-parametric), but nonlinear version of PCA. Most, but not all algorithms belonging to either of the categories accomplish their task in two steps:

Projection Define a dimension reducing transformation $\mathbf{f}(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^d$

Reconstruction Find a mapping back to the data space $\mathbf{g}(\boldsymbol{\eta}) : \mathbb{R}^d \rightarrow \mathbb{R}^p$

Here, the vector $\boldsymbol{\eta}$ represents a d -dimensional parameterization of the p -dimensional data space. Summarizing, the reconstructed curve is given by $\mathbf{g}(\mathbf{f}(\mathbf{x}))$, which is mostly found by (implicitly or explicitly) minimizing the reconstruction error

$$\mathbb{E}\|\mathbf{x} - \mathbf{g}(\mathbf{f}(\mathbf{x}))\|^2, \quad (7.2)$$

or its empirical counterpart

$$\sum_{i=1}^n \|\mathbf{x}_i - \mathbf{g}(\mathbf{f}(\mathbf{x}_i))\|^2. \quad (7.3)$$

PCA can be seen as a special case of the above algorithm. It follows directly from the extremal properties of the eigenvectors that the first d principal components define the *linear*, d -dimensional functions minimizing equations (7.2) or (7.3). In the special case of PCA $\mathbf{f}(\mathbf{x}) = (\boldsymbol{\gamma}_1 \cdots \boldsymbol{\gamma}_d)'(\mathbf{x} - \boldsymbol{\mu})$, $\mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\mu} + (\boldsymbol{\gamma}_1 \cdots \boldsymbol{\gamma}_d)\boldsymbol{\eta}$, and $(\mathbf{g} \circ \mathbf{f})(\mathbf{x}) = \boldsymbol{\mu} + (\boldsymbol{\gamma}_1 \cdots \boldsymbol{\gamma}_d)(\boldsymbol{\gamma}_1 \cdots \boldsymbol{\gamma}_d)'(\mathbf{x} - \boldsymbol{\mu})$, where $\boldsymbol{\mu} = \mathbb{E}(\mathbf{x})$. If all principal components are used (i.e. $d = p$) then it is easy to see that $\mathbf{g} \circ \mathbf{f}$ is the identity and the reconstruction error vanishes.

A further difference between principal curves and NLP PCA is that the projection \mathbf{f} in NLP PCA has to be continuous whereas the projection in principal curves can be discontinuous [37]. An important concept for NLP PCA as well as for principal curves is that of self-consistency, implying that the reconstructed curve is the mean over all data with equal projection, i.e $\mathbf{g}(\boldsymbol{\eta}) = \mathbb{E}(\mathbf{x}|\mathbf{f}(\mathbf{x}) = \boldsymbol{\eta})$. This idea is the cornerstone of the original principal curve approach by Hastie and Stuetzle [28] (hereafter: HS) as well as for the NLP PCA by Bolton et al. [5], who also consider all possible combinations of linear and nonlinear projection and reconstruction. A comparison of NLP PCA and HS principal curves can be found in [39].

A special type of principal curves, a so-called *local principal curve* ([17], see Section 7.2.2), is shown for the second of the two data sets in figure 7.1.

This article is concerned with extensions of PCA based on *localization*. In Section 2 we review such methods and also present a new approach based on local partitioning. In Section 3 we discuss how principal component regression can be localized. Furthermore we discuss projection directions other than the principal components. Using the projection directions used in the partial least squares (PLS) algorithm allows for taking both the inherent structure of the covariates and the response variable into account. In Section 4 we apply these methods to simulated data produced in order to develop classification and parametrization methods for the upcoming Gaia astronomical survey mission. This survey will obtain spectroscopic data (i.e. high dimensional vectors) on over one billion (10^9) stars, from which we wish to estimate intrinsic astrophysical parameters (temperature, abundance, surface gravity).

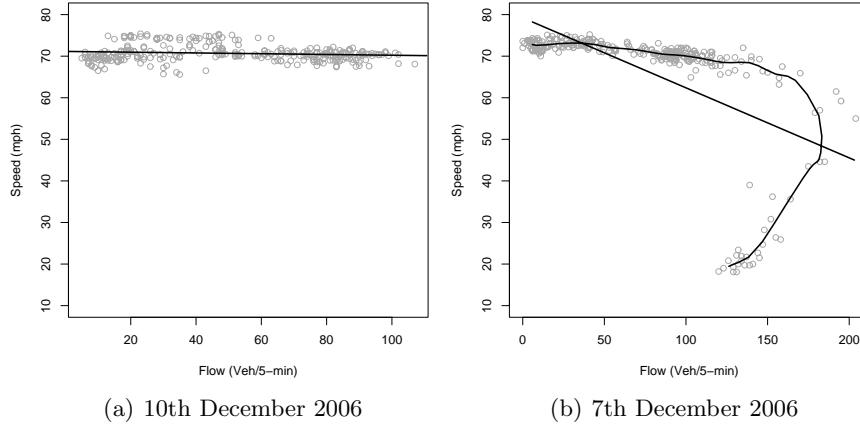


Fig. 7.1. Speed flow diagrams for Freeway 4-W on 10th (a) and 7th (b) of December 2006, with principal component lines (straight lines) and a principal curve through the latter data set (smooth curve).

7.2 Localized Principal Component Analysis

The term “local(ized) PCA” was coined decades ago in the literature on dimension reduction and feature extraction. However, it has not been used in a unique way; there are several ways of interpreting the term *local(ized)*. In this section, we will give a brief review of types of local PCA and illustrate some of them by applying them to the second speed-flow diagram presented in Section 1. In addition, we consider another artificial data set (corresponding to figure 3d in [17]), representing a spiral-like data cloud, with which principal curve / NLPCA algorithms generally tend to have problems.

7.2.1 Cluster-wise PCA

Historically the term *localized PCA* was used for cluster-wise PCA. Instead of fitting principal components through the whole data set, cluster-wise PCA partitions the data cloud into clusters, within which the principal component analysis is carried out “locally”.

Originally cluster-wise PCA was proposed as a tool for exploratory data analysis [6, 24]. It was rediscovered by the neural network community in the 1990s in the context of non-linear signal extraction as an alternative to auto-associative neural networks [33]. Five-layer auto-associative neural networks were successfully employed for (global) nonlinear PCA [40], using input and output layers with p units, and a hidden layer with $d < p$ units. Auto-associative neural networks possess many desirable theoretical properties in

terms of signal approximation [30]. However neural networks typically yield a non-convex optimization problem that is burdensome to solve, and the algorithm is — without suitable regularization — prone to getting trapped in poor local optima. Empirical evidence [33] suggests that cluster-wise PCA is at least on a par with auto-associative neural networks.

The cluster-wise PCA algorithm can be seen as a generalization of the Generalized Lloyd algorithm (“ K -Means”) [25]. In the K -Means algorithm the cluster centers are points, whereas in localized PCA, the cluster centers are hyperplane segments. The outline of the the cluster-wise PCA algorithm is as follows [14]:

Algorithm 1: Cluster-wise (“local”) PCA

1. Choose a target dimension d ($d = 1$ e.g. yields a line), a number of clusters Q , and an initial partitioning of the input space into Q disjoint regions $R^{(1)} \cup \dots \cup R^{(Q)} = \mathbb{R}^p$.

2. Iterate ...

- i. For each partition $R^{(q)}$ ($q = 1, \dots, Q$) compute the “local” covariance matrices

$$\boldsymbol{\Sigma}^{(q)} = \frac{1}{n^{(q)}} \sum_{i \in R^{(q)}} (\mathbf{x}_i - \bar{\mathbf{x}}^{(q)})(\mathbf{x}_i - \bar{\mathbf{x}}^{(q)})'$$

with $\bar{\mathbf{x}}^{(q)} = \frac{1}{n^{(q)}} \sum_{i \in R^{(q)}} \mathbf{x}_i$ and $n^{(q)}$ being the number of observations in partition $R^{(q)}$. Obtain the local principal components $\gamma_1^{(q)}, \dots, \gamma_d^{(q)}$ from the eigen decomposition of $\boldsymbol{\Sigma}^{(q)}$ (with corresponding eigenvalues $\lambda_1^{(q)} \geq \dots \geq \lambda_d^{(q)} \geq \dots$).

- ii. Update the partitioning $R^{(1)}, \dots, R^{(Q)}$ by allocating each observation \mathbf{x}_i to the “nearest” partition, i.e. the partition whose hyperplane segment is closest to \mathbf{x}_i .
-

Note that it is important in step 2.ii. to only consider the hyperplane segment $\left\{ \bar{\mathbf{x}}^{(q)} + \sum_{j=1}^d \xi_j \gamma_j^{(q)} : \xi_1, \dots, \xi_d \in \mathbb{R} \right\} \cap R^{(q)}$ instead of the whole hyperplane. Originally the partitioning was fixed a priori and thus there was no need to iterate steps i. and ii. This much simpler setup however is typically detrimental to the performance of the algorithm. A variant of algorithm 1 featuring automatic selection of clusters, and of the number of principal components within clusters, was proposed by Liu et al. [36] and used for star/galaxy classification.

It is easy to see that in complete analogy to the K -Means algorithm the sum of squared distances (7.3) is never increased by any step of the above algorithm. However as with K -Means or auto-associative neural networks, there is no guarantee that the global optimum is found. The most important drawback of this method is that its results are highly dependent on the initial choice of the partitioning.

This gives rise to the idea of slowly “building up” the partitions recursively akin to classification and regression trees (CARTs) [7]. Starting with a single partition, each partition is recursively “split up” into two partitions if the data can locally be better approximated by two hyperplane segments instead of a single one.

In order to evaluate whether such a split is necessary, every partition $R^{(q)}$ is split at the mean of the partition $\bar{\mathbf{x}}^{(q)}$ orthogonally to the first principal component $\gamma^{(q)}$ of the data in partition q , yielding two partitions $R^{(l)}$ and $R^{(r)}$. Next a small number of “k-segments” steps (steps 2.i. and 2.ii. from algorithm 1) are carried out for the partitions $R^{(l)}$ and $R^{(r)}$, however only using the data initially belonging to the partition $R^{(q)}$. The split is retained if

$$\frac{\lambda_1^{(q)} + \dots + \lambda_d^{(q)}}{\lambda_1^{(q)} + \dots + \lambda_p^{(q)}} < C \cdot \left(\frac{n^{(l)}}{n^{(q)}} \cdot \frac{\lambda_1^{(l)} + \dots + \lambda_d^{(l)}}{\lambda_1^{(l)} + \dots + \lambda_p^{(l)}} + \frac{n^{(r)}}{n^{(q)}} \cdot \frac{\lambda_1^{(r)} + \dots + \lambda_d^{(r)}}{\lambda_1^{(r)} + \dots + \lambda_p^{(r)}} \right), \quad (7.4)$$

where $n^{(q)}$ is the number of observations in partition $R^{(q)}$, $\lambda_k^{(q)}$ the k -th largest eigenvalue of the covariance of the data in partition $R^{(q)}$, and C is a constant which is typically chosen to be 1. Note that, contrary to CARTs, only a single split is considered for each partition. Considering every possible split of the partition would be computationally wasteful and in most cases not necessary as the split is subsequently optimized using a few “k-segments” steps.

In addition to testing whether partitions should be split, one can test whether neighboring partitions should be joined. This allows the algorithm to get around some of the suboptimal local extrema. In our experience it is beneficial to use a rather loose definition of neighborhood. Two partitions $R^{(l)}$ and $R^{(r)}$ are hereby considered to be neighbors if for at least for one observation \mathbf{x}_i , both $R^{(l)}$ and $R^{(r)}$ are amongst the $s > 2$ “closest” partitions. If the inequality (7.4) does not hold for two neighboring partitions $R^{(l)}$ and $R^{(r)}$, these are joined forming a new partition $R^{(q)} = R^{(l)} \cup R^{(r)}$.

This yields the following new algorithm:

Algorithm 2: Recursive local PCA

1. Start with a single partition $R^{(1)}$ containing all the data.
 2. Iterate ...
 - i. Test for each partition $R^{(q)}$ whether it should be split (using the criterion (7.4)).
 - ii. Carry out a fixed number of “k segments” steps (steps 2.i. and 2.ii. of algorithm 1) updating all partitions.
 - iii. Test for each neighboring pair of partitions whether the pair should be joined (using the criterion (7.4)).
 - ... until there is no change in the allocation of observations to partitions.
-

Note that this algorithm does not require the choice of an initial partitioning. It can be beneficial to “enforce” a certain number of splits during the

first few iterations, i.e. initially carry out every possible split irrespective of whether it meets criterion (7.4) and initially skip step 2.ii.

A similar idea has been proposed by Verbeek et al. [46]. Their method is however based on splitting off “zero length” segments corresponding to each observation.

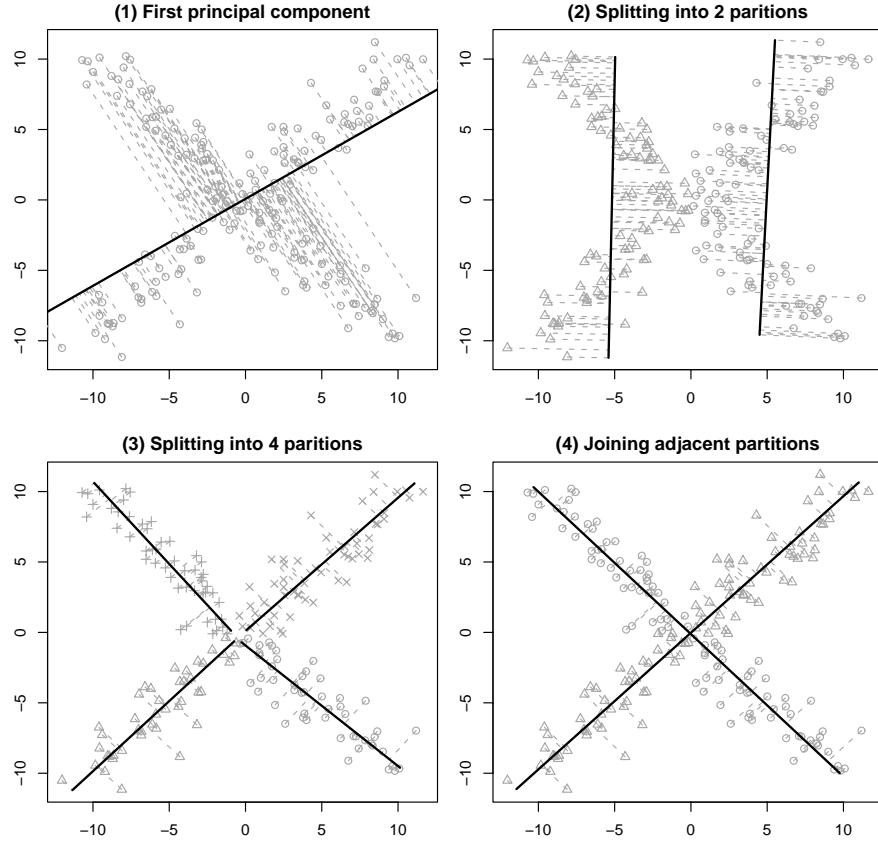


Fig. 7.2. Example illustrating the recursive local PCA algorithm.

Figure 7.2 illustrates the algorithm using a simple example. The data cannot be summarized by its first principal component (top left panel). Whilst it can be summarized by two line segments, the two line segments found in the second step are no truthful representation of the data (top right panel). The four line segments found in the third step correspond to the structure behind the data (bottom right panel). Finally the adjacent segments are joined leading to two partitions only (bottom right panel).

Note that the algorithm does not yield a principal curve or manifold, but merely disconnected line or hyperplane segments. It can be seen as finding tangent approximations to the principal curve or manifold. Thus the algorithm can by design cope with disconnected or branching principal curves or manifolds. Note that in the case of principal curves ($d = 1$) one can join the line segments to build a polygonal line [46].

7.2.2 Principal Curves

Principal curves were firstly introduced by Hastie and Stuetzle [28]. Their definition is based on the concept of self-consistency. A curve \mathbf{g} is called a Hastie-Stuetzle (HS) principal curve if $\mathbf{g}(\eta) = \mathbb{E}(\mathbf{x}|\eta_{\mathbf{g}}(\mathbf{x}) = \eta)$, with projection index $\eta_{\mathbf{g}}(\mathbf{x}) := \arg \min_{\eta} \|\mathbf{x} - \mathbf{g}(\eta)\|$. Hastie and Stuetzle show that a HS principal curve is a critical point of the distance function (7.2). Hastie and Stuetzle's definition however has a number of shortcomings. Whilst the principal curve can be shown to be a critical point of the distance function, one can show under fairly general conditions that it is just a saddle point of the distance function [15]. Further, when the data \mathbf{x} are generated by adding noise ϵ with $\mathbb{E}\epsilon = \mathbf{0}$ to a curve \mathbf{g} , i.e. $\mathbf{x} = \mathbf{g}(\eta) + \epsilon$, \mathbf{g} is typically not the principal curve. Kégl et al. [34] overcome the former problem by considering principal curves of fixed length. Tibshirani's definition of principal curves [43] overcomes the latter shortcoming by defining principal curves using a generative model.

The algorithms proposed by Hastie and Stuetzle, Tibshirani, and Kégl can all be described as “top-down” (or “global”) algorithms. All start with the first principal component and then iteratively “bend” it. The algorithm by Hastie and Stuetzle can be summarized as follows.

Algorithm 3: “Top-down” principal curves (HS)

1. Initialize the principal curve as the first principal component line $\mathbf{g}(\eta) = \bar{\mathbf{x}} + \eta\gamma_1$.
 2. Iterate between ...
 - i. **Projection** Project the data points \mathbf{x}_i onto the principal curve \mathbf{g} yielding projection indexes $\eta_{\mathbf{g}}(\mathbf{x}_i) = \eta_i$.
 - ii. **Reconstruction** Fit the data points \mathbf{x}_i component-wise against the projection indices η_i using a scatterplot smoother.

... until the change of some measure of goodness-of-fit falls below a certain threshold. The curve $\mathbf{g} : \mathbb{R} \longrightarrow \mathbb{R}^p$ reconstructed in the last iteration is the estimated principal curve.
-

The reconstruction step requires some form of smoothing, otherwise all data points would just be interpolated. It is implemented by using splines or local smoothers such as LOESS [10]. Note that this algorithm entails that the order of the projections η_i is maintained in each iteration. Tibshirani's

approach yields a similar algorithm. As Tibshirani's definition does not assume errors that are orthogonal to the principal curve the observations cannot be directly associated to a single point on the curve. Rather the algorithm is based on weights ("responsibilities") that correspond to the likelihood that a certain observation was generated by a certain point on the curve. From this point of view Tibshirani's algorithm is very similar to the EM algorithm for Gaussian mixture models. Kégl et al.'s algorithm uses polygonal lines with an increasing number of nodes.

Though all the three algorithms were shown to give satisfying results in a variety of circumstances, they all suffer from the problem of being highly dependent on the initialization. An unsuitable initialization of the projection indices η_i cannot be corrected later on. This is particularly obvious for spiral-like data, see figure 7.3 (top).

Hence, instead of starting with a *global* initial line, it often seems more appropriate to construct the principal curve in a "bottom-up" manner by considering in every step only data in a *local* neighborhood of the currently considered point. Delicado [12] proposed the first principal curve approach which can be assigned to this family, the *principal curves of oriented points (PCOP)*. A second approach in this direction was recently made by Einbeck et al. [17] and is known as *local principal curves (LPC)*. These two algorithms differ from all other existing NLPCA / principal curve algorithms in several crucial points:

- They do not start with an initial globally constructed line like the first principal component.
- They do not dissect into the stages of projection and reconstruction. The entire fitting process is carried out in the data space only.
- They do not maximize/minimize a global fitting criterion.

We illustrate this family of methods by providing the LPC algorithm explicitly. To fix terms, let $K_{\mathbf{H}}(\cdot)$ be a p -dimensional kernel function, \mathbf{H} a multivariate $(p \times p)$ bandwidth matrix, $w_i^{\mathbf{x}} = K_{\mathbf{H}}(\mathbf{x}_i - \mathbf{x}) / \sum_{i=1}^n K_{\mathbf{H}}(\mathbf{x}_i - \mathbf{x})$, and $\mathbf{x}'_i = (x_{i1}, \dots, x_{ip})$, $i = 1, \dots, n$.

Algorithm 4: Local principal curves (LPC)

1. Select a starting point \mathbf{x}_0 and a step size t_0 . Set $\mathbf{x} = \mathbf{x}_0$.
2. Calculate the local center of mass $\boldsymbol{\mu}^{\mathbf{x}} = \sum_{i=1}^n w_i^{\mathbf{x}} \mathbf{x}_i$ at \mathbf{x} . Denote by $\mu_j^{\mathbf{x}}$ the j -th component of $\boldsymbol{\mu}^{\mathbf{x}}$.
3. Estimate the local covariance matrix $\boldsymbol{\Sigma}^{\mathbf{x}} = (\sigma_{jk}^{\mathbf{x}})$ at \mathbf{x} via

$$\sigma_{jk}^{\mathbf{x}} = \sum_{i=1}^n w_i^{\mathbf{x}} (x_{ij} - \mu_j^{\mathbf{x}})(x_{ik} - \mu_k^{\mathbf{x}}).$$

Let $\boldsymbol{\gamma}^{\mathbf{x}}$ be the first column of the loadings matrix $\boldsymbol{\Gamma}^{\mathbf{x}}$ computed locally at \mathbf{x} in analogy to (7.1).

4. Setting $\mathbf{x} := \boldsymbol{\mu}^{\mathbf{x}} + t_0 \boldsymbol{\gamma}^{\mathbf{x}}$, one finds the updated value of \mathbf{x} .

5. Repeat steps 2 to 4 until the sequence of μ^x remains approximately constant (implying that the end of the data cloud is reached). Then set again $x = x_0$, set $\gamma^x := -\gamma^x$ and continue with step 4.

The step size t_0 is recommended to be set equal to h if $H = \text{diag}(h, \dots, h)$. The starting point x_0 may or may not be a member of the data cloud, and can be selected by hand or at random. Depending on the particular data set, it can have a rather crucial influence on the fitted curve. (In the examples provided in figure 7.3 (middle), the spiral fit is quite independent of x_0 , but it may require some attempts to get the fit through the traffic data right. A successful (randomly selected) choice was here e.g. $x_0 = (93, 71)$, using $h = t_0 = 7$.) For branched or disconnected data clouds it is often useful to work with multiple initializations [16, 17] and to compose the principal curve from the individual parts resulting from each starting point. In order to improve stability it is beneficial to replace the first local principal component γ^x by the weighted average between the current local principal component γ^x and the previous local principal component $\gamma^{x_{\text{old}}}$, i.e. use $\alpha\gamma^x + (1 - \alpha)\gamma^{x_{\text{old}}}$ for a suitable $\alpha \in (0, 1]$ (“angle penalization”, see [17]).

To make the difference to Delicado’s algorithm clear, let $\mathcal{H}(x, b)$ be the hyperplane which contains x and is orthogonal to a vector b . Then, for given x , Delicado defines the *principal direction* $b^*(x)$ as the vector minimizing the total variance of all observations lying on this hyperplane, and $\mu^*(x)$ as the expectation of all points lying on $\mathcal{H}(x, b^*(x))$. The set of fixed points of μ^* defines the set $\mathcal{P}(X)$ of *principal oriented points* (POPs), and any curve $\alpha : \mathbb{R} \supset I \longrightarrow \mathbb{R}^p$ with $\{\alpha(s) : s \in I\} \in \mathcal{P}(X)$ is a PCOP.

This is a probabilistic definition, hence estimates are required in practice, and this is where localization enters into Delicado’s algorithm in two different ways. Firstly, one considers a certain neighborhood of the current hyperplane \mathcal{H} , and data nearer to \mathcal{H} are associated with higher weights. Secondly, a cluster analysis is performed on \mathcal{H} , and only data in the local cluster are used for averaging. In contrast to the LPCs, where the local center of mass is found in a one-step approximation, Delicado’s analogue to step 2 requires iteration until convergence. Summarizing the essential difference slightly simplified, whereas LPC is based on calculating alternately a local first principal component γ_1^x and a local center of mass μ^x , in Delicado’s algorithm these two components are replaced by the estimated principal direction and the estimated fixed points (POPs), respectively. While Delicado’s algorithm is based on an elaborated and sound probabilistic theory, it is quite burdensome from a computational point of view, as his principal directions are not obtainable through an eigen decomposition [17], and a large number of cluster analyses has to be run. On the other hand, the computationally faster LPCs are a purely empirical concept, but they can be shown to be an approximation of Delicado’s algorithm for small cluster sizes [17]. For either algorithm, extensions to branched principal curves using the notion of *local second prin-*

cipal curves have been suggested [12, 16] and automatic bandwidth selection routines have been proposed [13, 17].

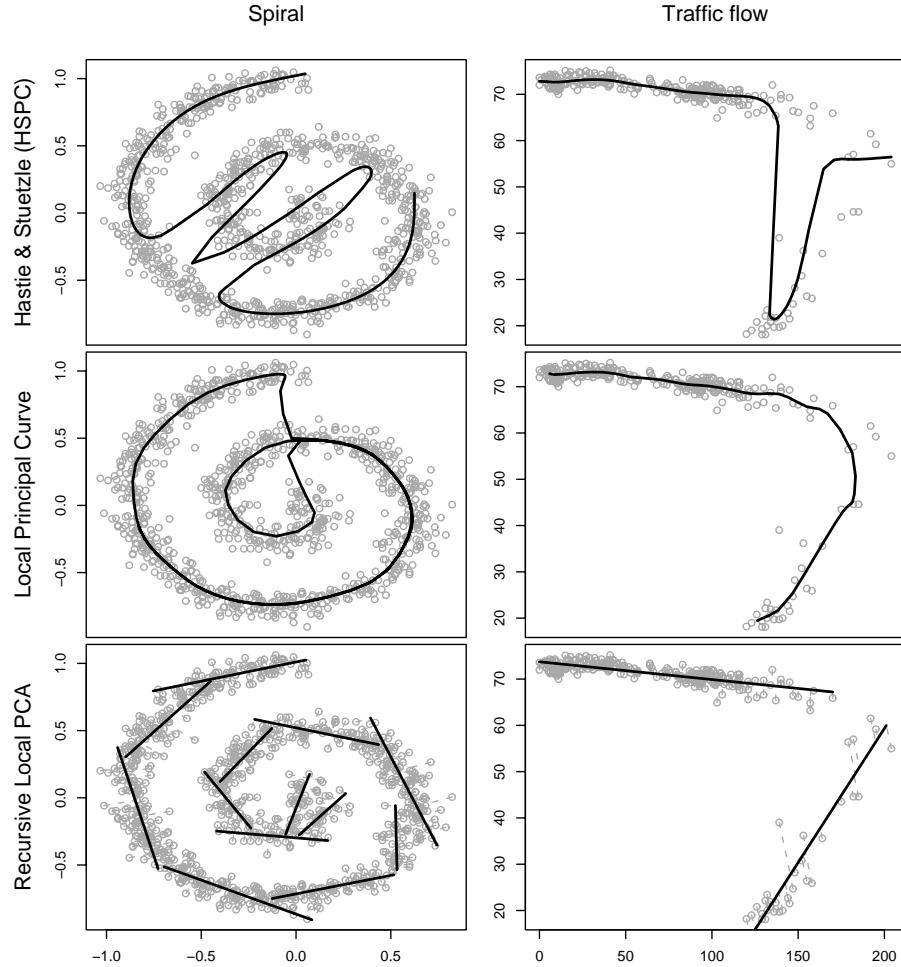


Fig. 7.3. Principal curves obtained for the spiral data and the non-linear traffic flow data using algorithm 3 (top row), algorithm 4 (middle row), and algorithm 2 (bottom row)

7.2.3 Further Approaches

The term local PCA has also been used for other techniques and in other contexts, which we briefly summarize here. Firstly, one can extend the algo-

rithms presented in Section 2.1 by allowing for smoothness over clusters using mixture models [38]. This means, rather than allocating an observation \mathbf{x}_i to the nearest partition q , one considers the posterior probabilities $\pi_{i,q}$ that \mathbf{x}_i is generated by partition q , and defines *for each observation* the weighted (first) local principal component $\gamma_{1,i} = \sum_q \pi_{i,q} \gamma_1^{(q)}$. This method requires knowledge of the mixture density itself and produces eigenvectors which are biased towards adjacent components, which motivated [38] to provide a modified algorithm using Fisher’s linear discriminant (FLD) instead of PCA.

Secondly, there are approaches from the NLPCA family using local methods. While most NLPCA algorithms work by applying some nonlinear, but parametric, transformation in the projection and/or the reconstruction step ([47], [8], among others), Bolten et al. [5] allowed for a nonparametric reconstruction \mathbf{g} . Using in the projection step a nonlinear transformation followed by a linear mapping, i.e. $\mathbf{f}(\mathbf{x}) = \mathbf{W}\phi(\mathbf{x})$, $\phi : \mathbb{R}^p \rightarrow \mathbb{R}^l$, $\mathbf{W} \in \mathbb{R}^{d \times l}$, the reconstructed curve takes the form $\mathbf{g}(\mathbf{W}\phi(\mathbf{x}))$, which is estimated using projection pursuit regression [23]. The actual smoothing method employed is the so-called supersmooth [21], which is a running line smoother using a local neighborhood of the target point with variable span. We do not consider NLPCA approaches further, as the contribution by U. Kruger with co-authors in this volume is explicitly dedicated to them.

Thirdly, the term “local principal component analysis” was used by Aluja-Banet and Nonell-Torrent [2] for PCA using contiguity relations based on a non-directed graph expressing binary relations between the individuals. Their motivation is to control for the effect of a latent or third variable (e.g. geographical position) by virtually eliminating its influence on the principal component analysis. This falls somewhat out of the context of the present paper and we do not consider it further.

7.3 Combining Principal Curves and Regression

7.3.1 Principal Component Regression and its Shortcomings

Using principal components for dimension reduction prior to carrying out further analysis is an old paradigm in statistics. The combination of principal component analysis and linear regression, known as principal component regression (PCR), was proposed already in the 1950s by Kendall [35] and Hotelling [31]. PCR consists of two steps. As a first step the first d principal components are extracted from the covariates \mathbf{X} . In the second step, the first d principal component scores $\mathbf{T}_d := (t_1 \cdots t_d) = \mathbf{X} \cdot (\gamma^{(1)} \cdots \gamma^{(d)})$ are used as covariates in a linear regression model. Geometrically, this corresponds to projecting the covariates onto the affine hyperplane spanned by the first d principal components and using the projection indices as covariates in the regression.

The purpose of the projection step is to eliminate directions in which the covariates have little variance. The rationale behind this is that directions with little variance correspond to directions in which there is little information in the covariates and thus are prone to leading to an overfit.

Being built upon principal component analysis, PCR suffers from the same shortcomings as PCA. If the structure of the covariates cannot be suitably approximated by an affine hyperplane of low dimension, PCA and thus PCR are likely to fail. In the following we will propose a new regression-tree like algorithm, based on algorithm 2, which can be used for local dimension reduction of the covariate space of a regression model.

7.3.2 The Generalization to Principal Curves

Instead of projecting the covariates onto the principal component surface we can project them onto a principal curve or manifold. In complete analogy to PCR we can then use the projection indices in the regression model.

This requires a unique parameterization of the principal manifold (e.g. a suitable unit speed parameterization). Whilst some of the algorithms presented in Section 2 compute the projection indices, other principal curve or manifold algorithms like algorithm 4 only yield a set of points on the curve or manifold.

The problem of having to compute the projection indices onto a non-linear curve or manifold can be circumvented by using tangent approximations to the principal curve or manifold as provided by algorithm 2, because the projections are then onto hyperplane segments and not onto non-linear manifolds. The projections onto the hyperplane segment corresponding to partition $R^{(q)}$ are

$$(t_{i1}^{(q)}, \dots, t_{id}^{(q)}) = \arg \min_{\substack{t_{i1}^{(q)}, \dots, t_{id}^{(q)} \in \mathbb{R} \\ \bar{\mathbf{x}}^{(q)} + \sum_{j=1}^d t_{ij}^{(q)} \boldsymbol{\gamma}_j^{(q)} \in R^{(q)}}} \left\| \mathbf{x}_i - \bar{\mathbf{x}}^{(q)} - \sum_{j=1}^d t_{ij}^{(q)} \boldsymbol{\gamma}_j^{(q)} \right\|^2.$$

The $(t_{i1}^{(q)}, \dots, t_{id}^{(q)})$ are easy to compute as they are either the orthogonal projection of the covariates onto the hyperplane or a point on the boundary of the hyperplane segment.

In analogy to algorithm 2 one can fit a regression model separately in each partition. This, however, leads to a couple of disadvantages. First of all, the prediction would be discontinuous at the boundaries between the partitions. Furthermore, keeping the “hard partitioning” can lead to a large variance of the predictions, especially if the number of partitions is large. In order to avoid these detrimental effects the partitioning is “softened” by the introduction of weights that reflect how far an observation \mathbf{x}_i is from a certain hyperplane segment. In order to achieve this all observations are projected onto all of the hyperplane segments. For each partition $R^{(q)}$ weights $\omega_i^{(q)}$ are

associated with each observation \mathbf{x}_i . The weight $\omega_i^{(q)}$ decreases with increasing distance between the observation and the hyperplane, i.e. $\omega_i^{(q)} := w(\delta_i^{(q)})$, where $\delta_i^{(q)} = \left\| \mathbf{x}_i - \bar{\mathbf{x}}^{(q)} - \sum_{j=1}^d t_{ij}^{(q)} \gamma_j^{(q)} \right\|^2$ is the squared distance between the i -th observation and its projection onto the hyperplane segment belonging to the partition $R^{(q)}$, and $w : \mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$ is a decreasing function, such as $w(\delta) = e^{-C \cdot \delta}$. Figure 7.4 illustrates this idea. The steps of the proposed algorithm are as follows:

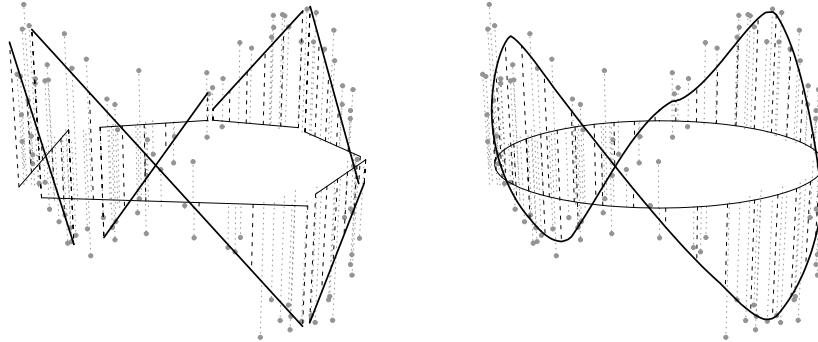


Fig. 7.4. Local regression lines in the partitions and resulting regression function for covariates lying on a circle in the two-dimensional plain. The vertical dimension corresponds to the response. Left: “hard partitioning”; right: “softened”.

Algorithm 5: Projection-based regression trees

1. Carry out algorithm 2 yielding a set of partitions $R^{(1)}, \dots, R^{(Q)}$ with the corresponding hyperplane segments.
2. Compute the squared distances $\delta_i^{(q)} = \left\| \mathbf{x}_i - \bar{\mathbf{x}}^{(q)} - \sum_{j=1}^d t_{ij}^{(q)} \gamma_j^{(q)} \right\|^2$ and weights $\omega_i^{(q)} := w(\delta_i^{(q)})$.
3. For each partition $R^{(1)}, \dots, R^{(Q)}$:
Fit a regression model using the projections of the data $\mathbf{T}^{(q)} := (\mathbf{t}_1^{(q)} \dots \mathbf{t}_d^{(q)})$ as covariates, \mathbf{y} as response, and $\omega_i^{(q)}$ as observation weights.
4. Compute the predictions as weighted means

$$\hat{y}_i = \frac{\sum_{q=1}^Q \omega_i^{(q)} \hat{y}_i^{(q)}}{\sum_{q=1}^Q \omega_i^{(q)}},$$

where $\hat{y}_i^{(q)}$ is the prediction obtained from the regression model in partition $R^{(q)}$.

Note that a wide variety of regression methods can be used in step 3. If the regression method does not support weights one can alternatively sample from all observations using the $\omega_i^{(q)}$ as sampling weights.

Algorithm 5 is a prediction algorithm that is like CARTs based on recursive partitioning. In contrast to CARTs, which split up the covariate space by a series of (orthogonal) “cuts”, the method proposed here partitions the covariate space according to a tessellation defined by the hyperplane segments. Figure 7.5 illustrates this comparison. As algorithm 5 takes the structure of the covariates into account, it provides some sort of regularization. Thus it tends to have a smaller variance than CARTs, however at the price of a potentially increased bias. In a boosting [20] context, this makes the method proposed here an interesting candidate for a weak learner.

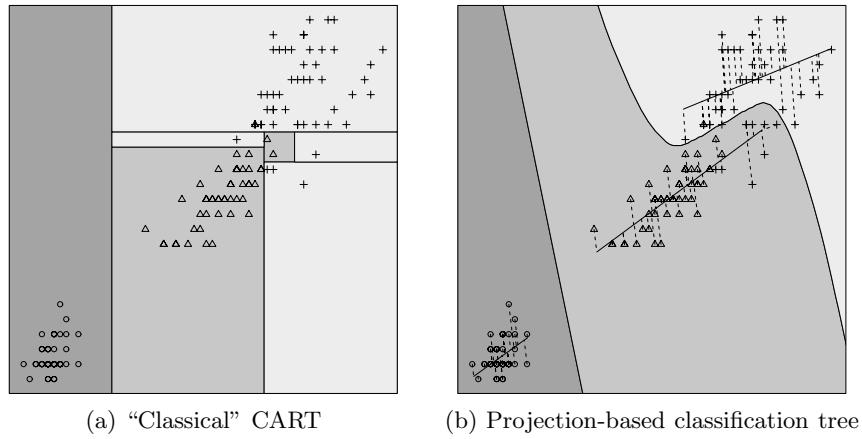


Fig. 7.5. Comparison between a classification tree (CART) and a projection-based classification tree

The algorithm presented here is similar to cluster-wise linear regression [42], however there are a few major differences. Most importantly, a dimension reduction step is carried out in each cluster and the clusters are not constructed around a single point, the cluster center, but around hyperplane segments. Furthermore the boundaries between the partitions are softened by the exponential weights. Note that the algorithm considered so far does not take the response variable y into account when constructing the partitions.

7.3.3 Using Directions Other than the Local Principal Components

The rationale behind projecting the data onto the principal component plane or a principal manifold was based on equating the variance of the covariates

to the information the covariates provide. But this does not necessarily imply that the directions with the largest variance necessarily provide the *relevant* information necessary to predict the response \mathbf{y} .⁵ In other words, it might be beneficial to consider both the structure of the covariates and the response variable.

We propose to modify algorithm 2 to take both these aspects into account. The core step of algorithm 2 were the “k-segments steps” used already in algorithm 1. In step 2.i. the principal components are computed in each partition. The core property of the principal components is that they maximize the variance of the projections $\text{Var}(\mathbf{X}^{(q)}\boldsymbol{\gamma})$. Clearly, this does not take the response $\mathbf{y}^{(q)}$ into account at all.⁶

If we were to choose the direction that predicts the response $\mathbf{y}^{(q)}$ best, we would choose the least squares regression estimate $\hat{\boldsymbol{\beta}}^{(q)} = (\mathbf{X}^{(q)'} \mathbf{X}^{(q)})^{-1} \mathbf{X}^{(q)'} \mathbf{y}^{(q)}$, which maximizes the squared correlation coefficient $\rho^2(\mathbf{y}^{(q)}, \mathbf{X}^{(q)'} \boldsymbol{\beta})$. Note that the latter criterion does not consider the variance of the covariates $\mathbf{X}^{(q)}$ at all.

It seems to be a suitable compromise to choose a projection direction $\boldsymbol{\gamma}$ that maximizes the product of the two aforementioned criteria, which is equivalent to maximizing the squared covariance between the $\mathbf{X}^{(q)}\boldsymbol{\gamma}$ and $\mathbf{y}^{(q)}$, i.e.

$$\text{Cov}^2(\mathbf{y}^{(q)}, \mathbf{X}^{(q)}\boldsymbol{\gamma}) = \rho^2(\mathbf{y}^{(q)}, \mathbf{X}^{(q)}\boldsymbol{\gamma}) \cdot \text{Var}(\mathbf{X}^{(q)}\boldsymbol{\gamma}) \cdot \text{Var}(\mathbf{y}^{(q)}).$$

This covariance is maximised by the projection direction of the PLS algorithm [49, 11]. This alternative projection direction can easily be incorporated into algorithms 1 and 2. Step 2.i. of algorithm 1 simply has to be replaced by the computation of the PLS projection direction obtained by carrying out a PLS regression with the data belonging to partition $R^{(q)}$. Using this modification typically improves the predictive performance. The following paragraph gives an example for this.

7.3.4 A Simple Example

In the following we will consider a data set taken from [9] to illustrate the algorithm proposed in this section. The data consists of measurements of the radial velocity of a spiral galaxy taken at 323 points in the area of sky which it covers. The different points in the area of the sky are determined by their north-south and east-west coordinates. The data is visualized in figure 7.6. It is easy to see that the measurements lie within a small number of slots crossing the origin.

The data set was randomly split into a training set of 162, and a test set of 161 observations. Table 7.1 gives the L_2 error (and its standard deviation) obtained when predicting the radial velocity using different methods

⁵ See [32] for a more detailed discussion.

⁶ $\mathbf{X}^{(q)}$ contains hereby all covariates $\mathbf{x}^{(q)}$ from the partition $R^{(q)}$, i.e. the $\mathbf{x}_i \in R^{(q)}$. $\mathbf{y}^{(q)}$ is defined analogously.

based on 1000 replications. The methods considered were algorithm 5 once with the principal components and once with the PLS projection direction, a linear model, generalized additive models (GAM) [29], multivariate adaptive regression splines (MARS) [22], and projection pursuit regression (PPR) [23]. The results show that the algorithm proposed here clearly outperforms the other methods. This is mostly due to the fact that it can successfully exploit the structure of the covariates. The PLS based variant of the algorithm gives much better results than the principal-component based variant.

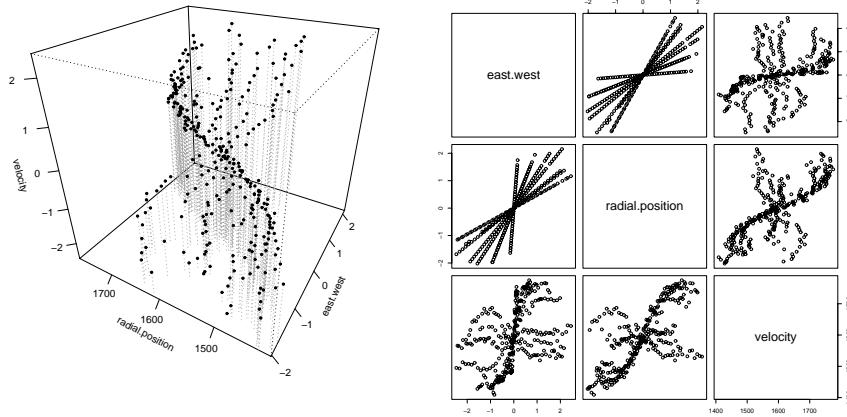


Fig. 7.6. Visualization of the galaxy data.

Table 7.1. Average L_2 error (and standard deviation of the error) obtained for the galaxy example for the different methods compared.

	Training set	Test set
Using PC directions	1642.00 (578.4)	1758.44 (615.5)
Using PLS directions	511.42 (79.3)	577.05 (101.9)
Linear model	7420.45 (352.9)	7591.19 (377.5)
MARS	2965.64 (334.7)	3738.76 (494.7)
GAM	3027.14 (321.6)	3554.26 (385.5)
PPR	2207.73 (622.0)	3317.94 (820.7)

7.4 Application to the Gaia Survey Mission

Gaia is an astrophysics mission of the European Space Agency (ESA) which will undertake a detailed survey of over 10^9 stars in our Galaxy and extragalactic objects. An important part of the scientific analysis of these data is the classification of all the objects as well as the estimation of stellar astrophysical parameters. Below we give a brief overview of the mission and the data before describing the results obtained using the techniques presented in the preceding section.

7.4.1 The Astrophysical Data

After launch in 2011, Gaia will study the composition and origin of our Galaxy by determining the properties of stars in different populations across our entire Galaxy [18, 41]. One of its major contributions will be to measure stellar distances to much higher accuracy than has hitherto been possible (and will do it for a vast numbers of stars). Gaia will also measure the three-dimensional space motions of stars in exquisite detail. These will be used together in dynamical models to map out the distribution of matter, and can be used to answer fundamental questions concerning galaxy formation.⁷

Much of the Gaia astrometric (3D position, 3D velocity) data would be of little value if we did not know the intrinsic properties of the stars observed, quantities such as the temperature, mass, chemical compositions, radius etc. (collectively referred to as *Astrophysical Parameters*, or APs; see [3]). For this reason, Gaia is equipped with a low resolution spectrograph to sample the spectral energy distribution at 96 points across the optical and near-infrared wavelength range (330–1000 nm). The measurements themselves are photon counts (energy flux). Each object can therefore be represented as a point in a 96-dimensional data space.

For those objects which are stars, the astrophysical parameters of most interest are the following four: (1) *effective temperature*, which roughly corresponds to the temperature of the observable part of the stellar atmosphere; (2) the *surface gravity*, which is the acceleration due to gravity at the surface of the star; (3) the *metallicity* or *abundance*, a single measurement of the chemical composition of the star relative to that of the Sun; (4) the interstellar extinction, which measures how much of the star’s light has been absorbed or scattered by interstellar dust lying between us and the star. In practice there is additional “cosmic variance” due to other APs, but these four are the main ones of interest. (In the rest of this paper we examine a simpler case in which there is no variance due to interstellar extinction.)

After a century of progress in astrophysics, we now have sophisticated models of stellar structure and from these we can generate synthetic spectra which reproduce real stellar spectra reasonably well. Therefore, we can construct libraries of template spectra with known APs and use these to train

⁷ For more information see <http://www.rssd.esa.int/Gaia>

supervised regression models in order to estimate stellar APs. This has received quite a lot of attention in the astronomical literature, with methods based on nearest neighbors (e.g. [1]), neural networks (e.g. [4], [48]) and support vector machines (e.g. [?])

In its simplest form, the problem of estimating APs with Gaia is one of finding the optimal mapping between the 96-dimensional data space and the 3 or more dimensional AP space. In theory this can be solved directly with regularized nonlinear regression, with the mapping inferred from simulated data. In practice, however, it is more complicated. For example, the mapping is not guaranteed to be unique, so some kind of partitioning of the data space may be appropriate. Furthermore, the intrinsic dimensionality of the data space is much lower than 96, so we could probably benefit from dimensionality reduction. Standard PCA has been applied to such data (e.g. [4], [19]). It produces more robust models, but possibly at the cost of filtering out low amplitude features which are nonetheless relevant for determining the “weaker” APs (see below). Here we investigate local reduction techniques as an alternative.

7.4.2 Principal Manifold Based Approach

In this section we study how the methods discussed in Sections 2 and 3 can be applied to Gaia spectral data. The data comprise several thousand spectra showing variance in the three astrophysical parameters temperature (in Kelvin), metallicity and gravity; the latter two variables are on a logarithmic scale.⁸ Temperature is a “strong” parameter, meaning it accounts for most of the variance across the data set. Gravity and metallicity, in contrast, are weak. The parameters have a correlated impact on the data, e.g. at high temperatures, varying the metallicity has a much smaller impact on spectra than it does at low temperatures. The data used here are simulated with no noise added.

The plot of the first three principal components of the covariates, the photon counts, shows clearly that these possess some low-dimensional structure (figure 7.7), which cannot be *linearly* approximated. This suggests employing principal-manifolds based methods.

The low-dimensional structure of the photon counts can be exploited by the projection-based regression tree algorithm (algorithm 5). Recall that the algorithm is based on the idea that the response is mainly determined by the projection of the covariates onto the (tangent to the) principal manifold. This however does not need to be the case; it might well be that the *relevant* information is not captured by the principal manifold.

In the following we will compare whether exploiting the manifold structure of the data allows for obtaining better predictions of the APs. Given

⁸ In the present example the spectra actually have a dimensionality of just 16, rather than 96 as mentioned above, because when we carried out this work the Gaia instruments were still being developed. Nonetheless, the results we present are illustrative of problems typical in observational astronomy.

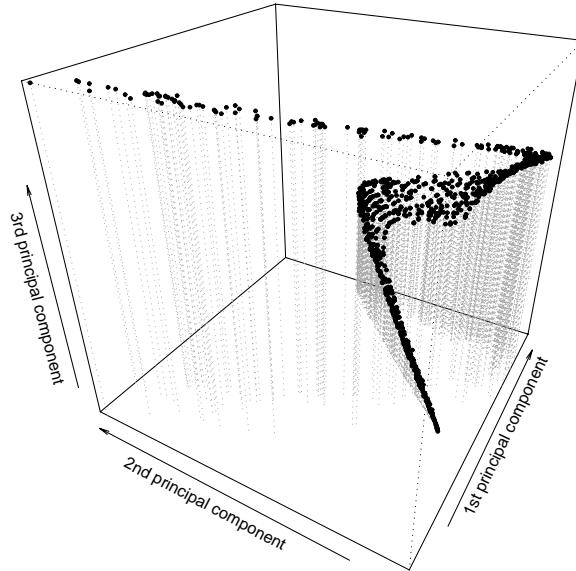


Fig. 7.7. First three principal components of the photon counts

the highly nonlinear structure of the data we use support vector regression machines (with a Gaussian kernel) in each partition.⁹ The partitions are determined by the PLS-based version of algorithm 5 using 6-dimensional (gravity, metallicity) and 4-dimensional (temperature) hyperplane segments. The results obtained are compared to two further support vector regression machines: one using the original data space of 16 photon counts directly, and one using the first 6 (gravity, metallicity) / 4 (temperature) global principal components as covariates.

For this purpose, a training set of 3,000 observations, a calibration set of 1,000 observations, and a test set of 1,000 observations is repeatedly drawn from the simulated Gaia data (100 replications). Table 7.2 gives the results. As support vector regression machines minimize the L_1 distance, we use the L_1 error $\sum_i |y_i - \hat{y}_i|/n$. The relative error reported is $\sum_i |y_i - \hat{y}_i|/\sum_i |y_i - \bar{y}|$.

The results obtained for the temperature and the metallicity show that using the manifold structure of the photon counts allows for a significant improvement of the predictive performance. The hyperplanes extracted in algorithm 5 seem to capture the information that is relevant to predicting the temperature and the metallicity whilst some of the noise is discarded in the projection step, which facilitates the prediction. This explains why algorithm 5

⁹ The optimal cost parameter and the optimal kernel width are determined for each partition individually using a calibration set of 1,000 observations.

Table 7.2. Average L_1 error and relative error (and standard deviation of the error) obtained for the Gaia data.

(a) Results obtained for the temperature

	Training set		Test set	
	L_1 error	Relative err.	L_1 error	Relative err.
Algorithm 5	252.9 (75.1)	0.042 (0.012)	258.9 (75.1)	0.043 (0.013)
SVR (all counts)	408.6 (11.6)	0.068 (0.002)	432.1 (16.0)	0.074 (0.004)
SVR (P.Comps.)	404.3 (19.6)	0.067 (0.003)	412.8 (21.8)	0.070 (0.004)

(b) Results obtained for the gravity

	Training set		Test set	
	L_1 error	Relative err.	L_1 error	Relative err.
Algorithm 5	0.083 (0.003)	0.069 (0.002)	0.109 (0.008)	0.090 (0.005)
SVR (all counts)	0.080 (0.001)	0.067 (0.001)	0.104 (0.005)	0.087 (0.005)
SVR (P.Comps.)	0.091 (0.002)	0.076 (0.001)	0.146 (0.009)	0.118 (0.007)

(c) Results obtained for the metallicity

	Training set		Test set	
	L_1 error	Relative err.	L_1 error	Relative err.
Algorithm 5	0.193 (0.018)	0.134 (0.012)	0.269 (0.016)	0.189 (0.011)
SVR (all counts)	0.279 (0.005)	0.193 (0.003)	0.363 (0.013)	0.253 (0.008)
SVR (P.Comps.)	0.256 (0.007)	0.177 (0.004)	0.389 (0.014)	0.269 (0.011)

outperforms the support vector regression machine using all photon counts. The principal components are less able to capture the relevant information; the performance of the support vector regression machine using the first 6 (or 4, respectively) global principal components is clearly worse.

The results obtained for the gravity, the weakest of the APs, however give a different picture. Using the manifold structure does *not* allow for improved predictions of the gravity. The information relevant for predicting the gravity seems to be “orthogonal” to the extracted hyperplane segments. Thus the support vector regression machine using all the photon counts performs better than the ones based on lower-dimensional projections.

7.5 Conclusion

We have reviewed several approaches and algorithms for the representation of high-dimensional complex data structures through lower-dimensional curves

or manifolds, which share the property of being based — by some means or other — on carrying out principal component analysis “locally”. We have focused on localized versions of principal curves (algorithm 4), and on an “intelligent” partitioning algorithm (algorithm 2) avoiding the necessity to specify an initial partitioning as with usual cluster-wise (“local”) PCA.

We have demonstrated, using the latter of the two algorithms, how localized principal components can be used to reduce the dimension of the predictor space in a non-linear high-dimensional regression problem. The information relevant to predicting the response variable can be elegantly taken into account by maximizing the covariance between the response variable and the extracted projection directions akin to partial least squares (PLS). We have applied this technique successfully to photon count data from the Gaia mission, where we were able to improve the predictive performance for some of the response variables significantly. The framework presented here however does not guarantee an improved prediction, especially if the information relevant to the prediction problem at hand cannot be captured by the extracted low-dimensional structure — as it was the case for the gravity.

It would be desirable to be able to also apply local principal curves (algorithm 4) to such problems. In contrast to algorithm 2, LPCs have the advantage of representing the covariate space through a proper curve (or manifold) instead of disconnected line (or hyperplane) segments. Currently, algorithm 4 can only extract one-dimensional curves, which it approximates by a sequence of points. Generalizing this idea to higher dimensions, one can approximate a d -dimensional principal manifold by a d -dimensional mesh, very much like the elastic net algorithm proposed by Gorban and Zinov'yev [26]. The (basic) elastic net algorithm however is a “top-down” method that iteratively bends a mesh of points, starting with a given topology. The generalization of algorithm 4 would use a “bottom-up” approach, i.e. learn the local topology from the data requiring no initialization.

However, generalizing algorithm 4 to d -dimensional manifolds poses a number of challenges. Firstly, the angle penalization needs to be modified that it can be applied to local principal components of higher order. Further one has to make sure that the different branches meet, forming a proper mesh of points, which will require keeping the distance between subsequent μ^x constant.

References

1. Allende Prieto, C.: An automated system to classify stellar spectra – I. Monthly Notices of the Royal Astronomical Society, **339**, 1111–1116 (2003)
2. Aluja-Banet, T. and Nonell-Torrent, R.: Local principal component analysis. *Qüestioó*, **3**, 267–278 (1991)
3. Bailer-Jones, C. A. L.: Determination of stellar parameters with GAIA. *Astrophysics and Space Science*, **280**, 21–29 (2002)

4. Bailer-Jones, C. A. L., Irwin, M., and von Hippel, T.: Automated classification of stellar spectra. II: Two-dimensional classification with neural networks and principal components analysis. *Monthly Notices of the Royal Astronomical Society*, **298**, 361–377 (1998)
5. Bolton, R. J., Hand, D. J., and Webb, A. R.: Projection techniques for nonlinear principal component analysis. *Statistics and Computing*, **13**, 267–276 (2003)
6. Braverman, E.M.: Methods of extremal grouping of parameters and problem of apportionment of essential factors, *Automation and Remote Control* (1) 108–116 (1970)
7. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J.: *Classification and Regression Trees*. Wadsworth and Brooks/Cole, Monterey (1984)
8. Chalmoud, B. and Girard, S. C.: Nonlinear modelling of scattered data and its application to shape change. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**, 422–432 (1999)
9. Chambers, J. M. and Hastie, T. J. (eds.): *Statistical Models* in S. CRC Press, Inc., Boca Raton (1991)
10. Cleveland, W. S.: Robust locally weighted regression and smoothing scatter-plots. *Journal of the American Statistical Association*, **74**, 829–836 (1979)
11. de Jong, S.: SIMPLS. An alternative approach to partial least squares regression. *Chemometric and Intelligent Laboratory Systems*, **18**, 251–263 (1993)
12. Delicado P.: Another look at principal curves and surfaces. *Journal of Multivariate Analysis*, **77**, 84–116 (2001)
13. Delicado, P. and Huerta, M.: Principal curves of oriented points: Theoretical and computational improvements. *Computational Statistics*, **18**, 293–315 (2003)
14. Diday, E.: Optimisation en classification automatique, Tome 1.,2.. INRIA, Rocquencourt (in French) (1979)
15. Duchamps, T. and Stuetzle, W.: Extremal properties of principal curves in the plane. *Annals of Statistics*, **24** (4), 1511–1520 (1996)
16. Einbeck, J., Tutz, G., and Evers, L.: Exploring multivariate data structures with local principal curves. In: Weihs, C. and Gaul, W. (eds.) *Classification - The Ubiquitous Challenge*, pages 257–263, Springer, Heidelberg (2005)
17. Einbeck, J., Tutz, G., and Evers, L.: Local principal curves. *Statistics and Computing*, **15**, 301–313 (2005)
18. ESA. Gaia: Composition, formation and evolution of the galaxy. Technical Report ESA-SCI 4, ESA (2000)
19. Fiorentin, P. Re, Bailer-Jones, C. A. L., Lee, Y. S., Beers, T. C., and Sivarani, T.: Estimating stellar atmospheric parameters from SDSS/SEGUE spectra. *Astronomy & Astrophysics*, submitted, (2007)
20. Freund, Y. and Schapire, R. E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pages 148–156 (1996)
21. Friedman, J. H.: A variable span scatterplot smoother. Technical Report No. 5, Laboratory for Computational Statistics, Stanford University (1984)
22. Friedman J. H.: Multivariate adaptive regression splines. *The Annals of Statistics*, **19** (1), 1–67 (1991)
23. Friedman, J. H. and Stuetzle, W.: Projection pursuit regression. *Journal of the American Statistical Association*, **76** (376), 817–823 (1981)
24. Fukunaga, K. and Olsen, D.: An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, **20** (2), 176–183 (1971)
25. Gersho, A. and Gray, R. M.: *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Amsterdam (1992)

26. Gorban, A. and Zinovyev, A.: Elastic principal graphs and manifolds and their practical applications. *Computing*, **75**, 359–379 (2005)
27. Hall, F. L., Hurdle, V. F., and Banks J. M.: Synthesis of recent work on the nature of speed-flow and flow-occupancy (or density) relations on freeways. *Transportation Research Record*, 1365, 12–17, (1992)
28. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association*, **84**, 502–516 (1989)
29. Hastie, T. and Tibshirani, R.: Generalized additive models. *Statistical Science*, **1**, 297–318 (1986)
30. Hornik, K., Stinchcombe, M., and White, H.: Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366 (1989)
31. Hotelling, H.: The relations of the newer multivariate statistical methods to factor analysis. *British Journal of Statistical Psychology*, **10**, 69–79 (1957)
32. Jolliffe, I. T.: A note on the use of principal components in regression. *Applied Statistics*, **31** (3), 300–303 (1982)
33. Kambhatla, N. and Leen, T. K.: Dimension reduction by local PCA. *Neural Computation*, **9**, 1493–1516 (1997)
34. Kégl, B., Krzyżak, A., Linder, T., and Zeger, K.: Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, 281–297 (2000)
35. Kendall, M. G.: *A Course in Multivariate Analysis*. Griffin, London (1957)
36. Liu, Z.-Y., Chiu, K.-C., and Xu, L.: Improved system for object detection and star/galaxy classification via local subspace analysis. *Neural Networks*, **16**, 437–451 (2003)
37. Malthouse, E. C.: Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Transactions on Neural Networks*, **9**, 165–173 (1998)
38. Marchette, D.J. and Poston, W.L.: Local dimensionality reduction using normal mixtures. *Computational Statistics*, **14**, 469–489 (1999)
39. Monaghan, A. H.: Nonlinear principal component analysis by neural networks: theory and application to the Lorenz system. *Journal of Climate*, **13**, 821–835 (2000)
40. Oja, E.: Data compression, feature extraction and autoassociation in feedforward neural networks. In: Kohonen, T., Makkisara, K., Simula, O., and Kangas, J. (eds.) *Artificial neural networks*, pages 737–745. Elsevier, Amsterdam (1991)
41. Perryman, M. A. C., de Boer, K. S., Gilmore, G., Høg, E., Lattanzi, M. G., Lindegren, L., Luri, X., Mignard, F., Pace, O., and de Zeeuw, P. T.: Gaia: Composition, formation and evolution of the galaxy. *Astronomy and Astrophysics*, **369**, 339–363 (2001)
42. Späth, H.: Algorithm 39: Clusterwise linear regression. *Computing*, **22**, 367–373 (1979)
43. Tibshirani, R.: Principal curves revisited. *Statistics and Computing*, **2**, 183–190 (1992)
44. Tsalmantza, P., Kontizas, M., Bailer-Jones, C. A. L., Rocca-Volmerange, B., Korakitis, R., Kontizas, E., Livanou, E., Dapergolas, A., Bellas-Velidis, I., Vallenari, A., and Fioc, M.: Towards a library of synthetic galaxy spectra and preliminary results of the classification and parametrization of unresolved galaxies from Gaia. *Astronomy and Astrophysics*, submitted (2007)
45. Varaiya, P.: Freeway performance measurement system (PeMS) version 4. Technical Report UCB-ITS-PRR-2004-31, California Partners for Advanced Transit and Highways (PATH) (2004)

46. Verbeek, J. J., Vlassis, N., and Kröse, B.: A k -segments algorithm for finding principal curves. Technical report, Computer Science Institute, University of Amsterdam (2000) IAS-UVA-00-11.
47. Webb, A. R.: An approach to non-linear principal component analysis using radially symmetric basis functions. *Statistics and Computing*, **6**, 159–168 (1996)
48. Willemse, P. G., Hilker, M., Kayser, A., and Bailer-Jones, C. A. L.: Analysis of medium resolution spectra by automated methods: Application to M 55 and ω Centauri. *Astronomy and Astrophysics*, **436**, 379–390 (2005)
49. Wold, H.: Soft modeling. The basic design and some extensions. In: Wold, H. and Jøreskog, K. G. (eds.) *Systems under Indirect Observation. Causality - Structure - Prediction*, volume 1, pages 1–53. North Holland (1982)

Auto-Associative Models, Nonlinear Principal Component Analysis, Manifolds and Projection Pursuit

Stéphane Girard¹ and Serge Iovleff²

¹ INRIA Rhône-Alpes, projet Mistis, Inovallée, 655, av. de l'Europe, Montbonnot, 38334 Saint-Ismier cedex, France,

`Stephane.Girard@inrialpes.fr`

² Laboratoire Paul Painlevé, 59655 Villeneuve d'Ascq Cedex, France,
`serge.iovleff@univ-lille1.fr`

Summary. Auto-associative models have been introduced as a new tool for building nonlinear Principal component analysis (PCA) methods. Such models rely on successive approximations of a dataset by manifolds of increasing dimensions. In this chapter, we propose a precise theoretical comparison between PCA and auto-associative models. We also highlight the links between auto-associative models, projection pursuit algorithms, and some neural network approaches. Numerical results are presented on simulated and real datasets.

8.1 Introduction

Principal component analysis (PCA) is a well-known method for extracting linear structures from high-dimensional datasets. It computes the subspace best approaching the dataset from the Euclidean point of view. This method benefits from efficient implementations based either on solving an eigenvalue problem or on iterative algorithms. We refer to [27] for details. In a similar fashion, multi-dimensional scaling [3, 35, 44] addresses the problem of finding the linear subspace best preserving the pairwise distances. More recently, new algorithms have been proposed to compute low dimensional embeddings of high dimensional data. For instance, Isomap [46], LLE (Locally linear embedding) [42] and CDA (Curvilinear distance analysis) [9] aim at reproducing in the projection space the structure of the initial local neighborhood. These methods are mainly dedicated to visualization purposes. They cannot produce an analytic form of the transformation function, making it difficult to map new points into the dimensionality-reduced space. Besides, since they rely on local properties of pairwise distances, these methods are sensitive to noise and outliers. We refer to [38] for a comparison between Isomap and CDA and to [48] for a comparison between some features of LLE and Isomap.

Finding nonlinear structures is a challenging problem. An important family of methods focuses on self-consistent structures. The self-consistency concept is precisely defined in [45]. Geometrically speaking, it means that each point of the structure is the mean of all points that project orthogonally onto it. For instance, it can be shown that the *K*-Means algorithm [23] converges to a set of k self-consistent points. Principal curves and surfaces [8, 24, 37, 47] are examples of one-dimensional and two-dimensional self-consistent structures. Their practical computation requires to solve a nonlinear optimization problem. The solution is usually non robust and suffers from a high estimation bias. In [31], a polygonal algorithm is proposed to reduce this bias. Higher dimensional self-consistent structures are often referred to as self-consistent manifolds even though their existence is not guaranteed for arbitrary datasets. An estimation algorithm based on a grid approximation is proposed in [19]. The fitting criterion involves two smoothness penalty terms describing the elastic properties of the manifold.

In this paper, auto-associative models are proposed as candidates to the generalization of PCA. We show in paragraph 8.2.1 that these models are dedicated to the approximation of the dataset by a manifold. Here, the word "manifold" refers to the topology properties of the structure [39]. The approximating manifold is built by a projection pursuit algorithm presented in paragraph 8.2.2. At each step of the algorithm, the dimension of the manifold is incremented. Some theoretical properties are provided in paragraph 8.2.3. In particular, we can show that, at each step of the algorithm, the mean residuals norm is not increased. Moreover, it is also established that the algorithm converges in a finite number of steps. Section 8.3 is devoted to the presentation of some particular auto-associative models. They are compared to the classical PCA and some neural networks models. Implementation aspects are discussed in Section 8.4. We show that, in numerous cases, no optimization procedure is required. Some illustrations on simulated and real data are presented in Section 8.5.

8.2 Auto-Associative Models

In this chapter, for each unit vector $a \in \mathbb{R}^p$, we denote by $P_a(\cdot) = \langle a, \cdot \rangle$ the linear projection from \mathbb{R}^p to \mathbb{R} . Besides, for all set E , the identity function $E \rightarrow E$ is denoted by Id_E .

8.2.1 Approximation by Manifolds

A function $F^d: \mathbb{R}^p \rightarrow \mathbb{R}^p$ is a d -dimensional auto-associative function if there exist d unit orthogonal vectors a^k , called principal directions, and d continuously differentiable functions $s^k: \mathbb{R} \rightarrow \mathbb{R}^p$, called regression functions, such that

$$P_{a^j} \circ s^k = \delta_{j,k} Id_{\mathbb{R}} \quad \text{for all } 1 \leq j \leq k \leq d , \quad (8.1)$$

where $\delta_{j,k}$ is the Kronecker symbol and

$$F^d = (Id_{\mathbb{R}^p} - s^d \circ P_{a^d}) \circ \dots \circ (Id_{\mathbb{R}^p} - s^1 \circ P_{a^1}) = \prod_{k=d}^1 (Id_{\mathbb{R}^p} - s^k \circ P_{a^k}). \quad (8.2)$$

The main feature of auto-associative functions is mainly a consequence of (8.1):

Theorem 1. *The equation $F^d(x) = 0$, $x \in \mathbb{R}^p$ defines a differentiable d -dimensional manifold of \mathbb{R}^p .*

We refer to [16] for a proof. Thus, the equation $F^d(x) = 0$ defines a space in which every point has a neighborhood which resembles the Euclidean space \mathbb{R}^d , but in which the global structure may be more complicated. As an example, on a 1-dimensional manifold, every point has a neighborhood that resembles a line. In a 2-manifold, every point has a neighborhood that looks like a plane. Examples include the sphere or the surface of a torus.

Now, let X be a square integrable random vector of \mathbb{R}^p . Assume, without loss of generality, that X is centered and introduce $\sigma^2(X) \stackrel{\text{def}}{=} \mathbb{E}[\|X\|^2]$. For all auto-associative function F^d , let us consider $\varepsilon = F^d(X)$. Note that, from the results of Subsection 8.2.3 below, ε is necessarily a centered random vector. In this context, $\sigma^2(\varepsilon)$ is called the residual variance. Geometrically speaking, the realizations of the random vector X are approximated by the manifold $F^d(x) = 0$, $x \in \mathbb{R}^p$ and $\sigma^2(\varepsilon)$ represents the variance of X "outside" the manifold.

Of course, such random vector X always satisfies a 0-dimensional auto-associative model with $F^0 = Id_{\mathbb{R}^p}$ and $\sigma^2(\varepsilon) = \sigma^2(X)$. Similarly, X always satisfies a p -dimensional auto-associative model with $F^p = 0$ and $\sigma^2(\varepsilon) = 0$. In practice, it is important to find a balance between these two extreme cases by constructing a d -dimensional model with $d \ll p$ and $\sigma^2(\varepsilon) \ll \sigma^2(X)$. For instance, in the case where the covariance matrix Σ of X is of rank d , then X is located on a d -dimensional linear subspace defined by the equation $F_{PCA}^d(x) = 0$ with

$$F_{PCA}^d(x) = x - \sum_{k=1}^d P_{a^k}(x)a^k, \quad (8.3)$$

and where a^k , $k = 1, \dots, d$ are the eigenvectors of Σ associated to the positive eigenvalues. A little algebra shows that (8.3) can be rewritten as $F^d(x) = 0$, where F^d is a d -dimensional auto-associative function with linear regression functions $a^k(t) = ta^k$ for $k = 1, \dots, d$. Moreover, we have $\sigma^2(\varepsilon) = 0$. Since (8.3) is the model produced by a PCA, it straightforwardly follows that PCA is a special (linear) case of auto-associative models. In the next section, we propose an algorithm to build auto-associative models with non necessarily linear regression functions, small dimension and small residual variance. Such

models could also be called "semi-linear" or "semi-parametric" since they include a linear/parametric part through the use of linear projection operators and a non-linear/non-parametric part through the regression functions.

8.2.2 A Projection Pursuit Algorithm

Let us recall that, given an unit vector $a \in \mathbb{R}^p$, an index $I: \mathbb{R} \rightarrow \mathbb{R}$ is a functional measuring the interest of the projection $P_a(X)$ with a non negative real number. The meaning of the word "interest" depends on the considered data analysis problem. For instance, a possible choice of I is the projected variance $I \circ P_a(\cdot) = \text{Var}[P_a(\cdot)]$. Some other examples are presented in Section 8.4.2. Thus, the maximization of $I \circ P_a(X)$ with respect to a yields the most interesting direction for this given criteria. An algorithm performing such an optimization is called a projection pursuit algorithm. We refer to [26] and [28] for a review on this topic.

Let $d \in \{0, \dots, p\}$, and consider the following algorithm which consists in applying iteratively the following steps: [A] computation of the Axes, [P] Projection, [R] Regression and [U] Update:

Algorithm 1 Define $R^0 = X$.

For $k = 1, \dots, d$:

- [A] Determine $a^k = \arg \max_{x \in \mathbb{R}^p} I \circ P_x(R^{k-1})$ s.t. $\|x\| = 1$, $P_{a^j}(x) = 0$, $1 \leq j < k$.
- [P] Compute $Y^k = P_{a^k}(R^{k-1})$.
- [R] Estimate $s^k(t) = \mathbb{E}[R^{k-1}|Y^k = t]$,
- [U] Compute $R^k = R^{k-1} - s^k(Y^k)$.

The random variables Y^k are called principal variables and the random vectors R^k residuals. Step [A] consists in computing an axis orthogonal to the previous ones and maximizing a given index I . Step [P] consists in projecting the residuals on this axis to determine the principal variables, and step [R] is devoted to the estimation of the regression function of the principal variables best approximating the residuals. Step [U] simply consists in updating the residuals. Thus, Algorithm 1 can be seen as a projection pursuit regression algorithm [14, 32] since it combines a projection pursuit step [A] and a regression step [R]. The main problem of such approaches is to define an efficient way to iterate from k to $k+1$. Here, the key property is that the residuals R^k are orthogonal to the axis a^k since

$$\begin{aligned} P_{a^k}(R^k) &= P_{a^k}(R^{k-1}) - P_{a^k} \circ s^k(Y^k) \\ &= P_{a^k}(R^{k-1}) - \mathbb{E}[P_{a^k}(R^{k-1})|Y^k] \\ &= Y^k - \mathbb{E}[Y^k|Y^k] \\ &= 0. \end{aligned} \tag{8.4}$$

Thus, it is natural to iterate the model construction in the subspace orthogonal to a^k , see the orthogonality constraint in step [A]. The theoretical results provided in the next paragraph are mainly consequences of this property.

8.2.3 Theoretical Results

Basing on (8.4), it is easily shown by induction that both the residuals and the regression functions computed at the iteration k are almost surely (a.s.) orthogonal to the axes computed before. More precisely, one has

$$\langle a^j, R^k \rangle = 0, \text{ a.s. for all } 1 \leq j \leq k \leq d, \quad (8.5)$$

$$\langle a^j, s^k(Y^k) \rangle = 0, \text{ a.s. for all } 1 \leq j < k \leq d. \quad (8.6)$$

Besides, the residuals, principal variables and regression functions are centered:

$$\mathbb{E}[R^k] = \mathbb{E}[Y^k] = \mathbb{E}[s^k(Y^k)] = 0,$$

for all $1 \leq k \leq d$. Our main result is the following:

Theorem 2. *Algorithm 1 builds a d -dimensional auto-associative model with principal directions $\{a^1, \dots, a^d\}$, regression functions $\{s^1, \dots, s^d\}$ and residual $\varepsilon = R^d$. Moreover, one has the expansion*

$$X = \sum_{k=1}^d s^k(Y^k) + R^d, \quad (8.7)$$

where the principal variables Y^k and Y^{k+1} are centered and non-correlated for $k = 1, \dots, d-1$.

The proof is a direct consequence of the orthogonality properties (8.5) and (8.6). Let us highlight that, for $d = p$, expansion (8.7) yields an exact expansion of the random vector X as:

$$X = \sum_{k=1}^p s^k(Y^k),$$

since $R^p = 0$ (a.s.) in view of (8.5). Finally, note that the approximation properties of the conditional expectation entails that the sequence of the residual norms is almost surely non increasing. As a consequence, the following corollary will prove useful to select the model dimension similarly to the PCA case.

Corollary 1. *Let Q_d be the information ratio represented by the d -dimensional auto-associative model:*

$$Q_d = 1 - \sigma^2(R^d) / \sigma^2(X).$$

Then, $Q_0 = 0$, $Q_p = 1$ and the sequence (Q_d) is non decreasing.

Note that all these properties are quite general, since they do not depend either on the index I , nor on the estimation method for the conditional expectation. In the next section, we show how, in particular cases, additional properties can be obtained.

8.3 Examples

We first focus on the auto-associative models which can be obtained using linear estimators of the regression functions. The existing links with PCA are highlighted. Second, we introduce the intermediate class of additive auto-associative models and compare it to some neural network approaches.

8.3.1 Linear Auto-Associative Models and PCA

Here, we limit ourselves to linear estimators of the conditional expectation in step [R]. At iteration k , we thus assume

$$s^k(t) = tb^k, \quad t \in \mathbb{R}, \quad b^k \in \mathbb{R}^p.$$

Standard optimization arguments (see [18], Proposition 2) shows that, necessarily, the regression function obtained at step [R] is located on the axis

$$b^k = \Sigma_{k-1} a^k / ({}^t a^k \Sigma_{k-1} a^k), \quad (8.8)$$

with Σ_{k-1} the covariance matrix of R^{k-1} :

$$\Sigma_{k-1} = \mathbb{E}[R^{k-1} {}^t R^{k-1}], \quad (8.9)$$

and where, for all matrix M , the transposed matrix is denoted by ${}^t M$. As a consequence of Theorem 2, we have the following linear expansion:

$$X = \sum_{k=1}^d \frac{Y^k \Sigma_{k-1} a^k}{{}^t a^k \Sigma_{k-1} a^k} + R^d.$$

As an interesting additional property of these so-called linear auto-associative models, we have $\mathbb{E}[Y_j Y_k] = 0$ for all $1 \leq j < k \leq d$. This property is established in [18], Proposition 2. Therefore, the limitation to a family of linear functions in step [R] allows to recover an important property of PCA models: the non-correlation of the principal variables. It is now shown that Algorithm 1 can also compute a PCA model for a well suited choice of the index.

Proposition 1. *If the index in step [A] is the projected variance, i.e.*

$$I \circ P_x(R^{k-1}) = \text{Var}[P_x(R^{k-1})],$$

and step [R] is given by (8.8) then Algorithm 1 computes the PCA model of X .

Indeed, the solution a^k of step [A] is the eigenvector associated to the maximum eigenvalue of Σ_{k-1} . From (8.8) it follows that $b^k = a^k$. Replacing in (8.2), we obtain, for orthogonality reasons, $F^d = F_{PCA}^d$.

8.3.2 Additive Auto-Associative Models and Neural Networks

A d -dimensional auto-associative function is called additive if (8.2) can be rewritten as

$$F^d = Id_{\mathbb{R}^d} - \sum_{k=1}^d s^k \circ P_{a^k} . \quad (8.10)$$

In [17], the following characterization of additive auto-associative functions is provided. A d -dimensional auto-associative function is additive if and only if

$$P_{a^j} \circ s^k = \delta_{j,k} Id_{\mathbb{R}} \text{ for all } (j, k) \in \{1, \dots, d\}^2 .$$

As a consequence, we have:

Theorem 3. *In the linear subspace spanned by $\{a^1, \dots, a^d\}$, every d -dimensional additive auto-associative model reduces to the PCA model.*

A similar result can be established for the nonlinear PCA based on a neural network and introduced in [29]. The proposed model is obtained by introducing a nonlinear function $g : \mathbb{R} \rightarrow \mathbb{R}$, called activation function, in the PCA model (8.3) to obtain

$$F_{KJ}^d(x) = x - \sum_{k=1}^d g \circ P_{a^k}(x) a^k . \quad (8.11)$$

Note that (8.11) is an additive auto-associative model as defined in (8.10) if and only if $g = Id_{\mathbb{R}}$, i.e. if and only if it reduces to the PCA model in the linear subspace spanned by $\{a^1, \dots, a^d\}$. Moreover, in all cases, we have

$$\{F_{KJ}^d(x) = 0, x \in \mathbb{R}^p\} \subset \{F_{PCA}^d(x) = 0, x \in \mathbb{R}^p\} ,$$

which means that this model is included in the PCA one. More generally, the auto-associative Perceptron with one hidden layer [7] is based on multidimensional activation functions $\sigma^k : \mathbb{R} \rightarrow \mathbb{R}^p$:

$$F_{AAP}^d(x) = x - \sum_{k=1}^d \sigma^k \circ P_{a^k}(x) . \quad (8.12)$$

Unfortunately, it can be shown [10] that a single hidden layer is not sufficient. Linear activation functions (leading to a PCA) already yield the best approximation of the data. In other words, the nonlinearity introduced in (8.12) has no significant effect on the final approximation of the dataset. Besides, determining a^k , $k = 1, \dots, d$ is a highly nonlinear problem with numerous local minima, and thus very dependent on the initialization.

8.4 Implementation Aspects

In this section, we focus on the implementation aspects associated to Algorithm 1. Starting from a n -sample $\{X_1, \dots, X_n\}$, two problems are addressed. In Subsection 8.4.1, we propose some simple methods to estimate the regression functions s^k appearing in step [R]. In Subsection 8.4.2, the choice of the index in step [A] is discussed. In particular, we propose a contiguity index whose maximization is explicit.

8.4.1 Estimation of the Regression Functions

Linear auto-associative models

To estimate the regression functions, the simplest solution is to use a linear approach leading to a linear auto-associative model. In this case, the regression axis is explicit, see (8.8), and it suffices to replace Σ_{k-1} defined in (8.9) by its empirical counterpart

$$V_{k-1} = \frac{1}{n} \sum_{i=1}^n R_i^{k-1} {}^t R_i^{k-1}, \quad (8.13)$$

where R_i^{k-1} is the residual associated to X_i at iteration $k - 1$.

Nonlinear auto-associative models

Let us now focus on nonlinear estimators of the conditional expectation $s^k(t) = \mathbb{E}[R^{k-1}|Y^k = t]$, $t \in \mathbb{R}$. Let us highlight that s^k is a univariate function and thus its estimation does not suffer from the curse of dimensionality [1]. This important property is a consequence of the "bottleneck" trick used in (8.2) and, more generally, in neural networks approaches. The key point is that, even though $s^k \circ P_{a^k}$ is a p -variate function, its construction only requires the nonparametric estimation of a univariate function thanks to the projection operator.

For the sake of simplicity, we propose to work in the orthogonal basis B^k of \mathbb{R}^p obtained by completing $\{a^1, \dots, a^k\}$. Let us denote by R_j^{k-1} the j -th coordinate of R^{k-1} in B^k . In view of (8.5), $R_j^{k-1} = 0$ for $j = 1, \dots, k - 1$. Besides, from step [P], $R_k^{k-1} = Y^k$. Thus, the estimation of $s^k(t)$ reduces to the estimation of $p - k$ functions

$$s_j^k(t) = \mathbb{E}[R_j^{k-1}|Y^k = t], \quad j = k + 1, \dots, p.$$

This standard problem [22, 12] can be tackled either by kernel [2] or projection [20] estimates.

Kernel estimates

Each coordinate $j \in \{k+1, \dots, p\}$ of the estimator can be written in the basis B^j as:

$$\hat{s}_j^k(t) = \sum_{i=1}^n R_{j,i}^{k-1} K\left(\frac{t - Y_i^k}{h}\right) \Bigg/ \sum_{i=1}^n K\left(\frac{t - Y_i^k}{h}\right), \quad (8.14)$$

where $R_{j,i}^{k-1}$ represents the j -th coordinate of the residual associated to the observation X_i at the $(k-1)$ -th iteration in the basis B^k , Y_i^k is the value of the k -th principal variable for the observation X_i and K is a Parzen-Rosenblatt kernel, that is to say a bounded real function, integrating to one and such that $tK(t) \rightarrow 0$ as $|t| \rightarrow \infty$. For instance, one may use a standard Gaussian density. The parameter h is a positive number called window in this context. In fact, $\hat{s}_j^k(t)$ can be seen as a weighted mean of the residuals $R_{j,i}^{k-1}$ which are close to t :

$$\hat{s}_j^k(t) = \sum_{i=1}^n R_{j,i}^{k-1} w_i^k(t),$$

where the weights are defined by

$$w_i^k(t) = K\left(\frac{t - Y_i^k}{h}\right) \Bigg/ \sum_{i=1}^n K\left(\frac{t - Y_i^k}{h}\right),$$

and are summing to one:

$$\sum_{i=1}^n w_i^k(t) = 1.$$

The amplitude of the smoothing is tuned by h . In the case of a kernel with bounded support, for instance if $\text{supp}(K) = [-1, 1]$, the smoothing is performed on an interval of length $2h$. For an automatic choice of the smoothing parameter h , we refer to [25], Chapter 6.

Projection estimates

Each coordinate $j \in \{k+1, \dots, p\}$ of the estimator is expanded on a basis of L real functions $\{b_\ell(t), \ell = 1, \dots, L\}$ as:

$$\tilde{s}_j^k(t) = \sum_{\ell=1}^L \tilde{\alpha}_{j,\ell}^k b_\ell(t).$$

The coefficients $\tilde{\alpha}_{j,\ell}^k$ appearing in the linear combination of basis functions are determined such that $\tilde{s}_j^k(Y_i^k) \simeq R_{j,i}^{k-1}$ for $i = 1, \dots, n$. More precisely,

$$\tilde{\alpha}_{j,.}^k = \arg \min_{\alpha_{j,.}^k} \sum_{i=1}^n \left(\sum_{\ell=1}^L \alpha_{j,\ell}^k b_\ell(Y_i^k) - R_{j,i}^{k-1} \right)^2,$$

and it is well-known that this least-square problem benefits from an explicit solution which can be matricially written as

$$\tilde{\alpha}_{j,.}^k = (^t B^k B^k)^{-1} {}^t B^k R_{j,.}^{k-1}, \quad (8.15)$$

where B^k is the $n \times L$ matrix with coefficients $B_{i,\ell}^k = b_\ell(Y_i^k)$, $i = 1, \dots, n$, $\ell = 1, \dots, L$. Note that this matrix does not depend on the coordinate j . Thus, the matrix inversion in (8.15) is performed only once at each iteration k . Besides, the size of this matrix is $L \times L$ and thus does not depend either on the dimension of the space p , nor on the sample size n . As an example, one can use a basis of cubic splines [11]. In this case, the parameter L is directly linked to N the number of knots: $L = N + 4$. Remark that, in this case, condition $N + 4 \leq n$ is required so that the matrix is ${}^t B^k B^k$ is regular.

8.4.2 Computation of Principal Directions

The choice of the index I is the key point of any projection pursuit problem where it is needed to find "interesting" directions. We refer to [26] and [28] for a review on this topic. Let us recall that the meaning of the word "interesting" depends on the considered data analysis problem. As mentioned in Subsection 8.2.2, the most popular index is the projected variance

$$I_{PCA} \circ P_x(R^{k-1}) = \frac{1}{n} \sum_{i=1}^n P_x^2(R_i^{k-1}) \quad (8.16)$$

used in PCA. Remarking that this index can be rewritten as

$$I_{PCA} \circ P_x(R^{k-1}) = \frac{1}{2n^2} \sum_{i=1}^n \sum_{j \neq i} P_x^2(R_i^{k-1} - R_j^{k-1}),$$

it appears that the "optimal" axis maximizes the mean distance between the projected points. An attractive feature of the index (8.16) is that its maximization benefits from an explicit solution in terms of the eigenvectors of the empirical covariance matrix V_{k-1} defined in (8.13). Friedman *et al* [15, 13], and more recently Hall [21], proposed an index to find clusters or use deviation from the normality measures to reveal more complex structures of the scatter-plot. An alternative approach can be found in [4] where a particular metric is introduced in PCA so as to detect clusters. We can also mention indices dedicated to outliers detection [40]. Similar problems occur in the neural networks context where the focus is on the construction of nonlinear mappings to unfold the manifold. It is usually required that such a mapping preserves that local topology of the dataset. In this aim, Demartines and Herault [9] introduce an index to detect the directions in which the nonlinear projection approximatively preserves distances. Such an index can be adapted to our framework by restricting ourselves to linear projections:

$$I_{DH} \circ P_x(R^{k-1}) = \sum_{i=1}^n \sum_{j \neq i} (\|R_i^{k-1} - R_j^{k-1}\| - |P_x|(R_i^{k-1} - R_j^{k-1}))^2 \\ H \circ |P_x|(R_i^{k-1} - R_j^{k-1}).$$

The function H is assumed to be positive and non increasing in order to favor the local topology preservation. According the authors, the application of this function to the outputs $P_x R_i^{k-1}$ instead of the inputs R_i^{k-1} allows to obtain better performances than the Kohonen's self-organizing maps [33, 34]. Similarly, the criterion introduced in [43] yields in our case

$$I_S \circ P_x(R^{k-1}) = \sum_{i=1}^n \sum_{j \neq i} (\|R_i^{k-1} - R_j^{k-1}\| - |P_x|(R_i^{k-1} - R_j^{k-1}))^2 \\ \sqrt{\sum_{i=1}^n \sum_{j \neq i} P_x^2(R_i^{k-1} - R_j^{k-1})}.$$

However, in both cases, the resulting functions are nonlinear and thus difficult to optimize with respect to x .

Our approach is similar to Lebart one's [36]. It consists in defining a contiguity coefficient whose minimization allows to unfold nonlinear structures. At each iteration k , the following Rayleigh quotient [41] is maximized with respect to x :

$$I \circ P_x(R^{k-1}) = \sum_{i=1}^n P_x^2(R_i^{k-1}) \left/ \sum_{i=1}^n \sum_{j=1}^n m_{i,j}^{k-1} P_x^2(R_i^{k-1} - R_j^{k-1}) \right. . \quad (8.17)$$

The matrix $M^{k-1} = (m_{i,j}^{k-1})$ is a first order contiguity matrix, whose value is 1 when R_j^{k-1} is the nearest neighbor of R_i^{k-1} , 0 otherwise. The upper part of (8.17) is proportional to the usual projected variance, see (8.16). The lower part is the distance between the projection of points which are nearest neighbor in \mathbb{R}^p . Then, the maximization of (8.17) should reveal directions in which the projection best preserves the first order neighborhood structure (see Figure 8.1). In this sense, the index (8.17) can be seen as a first order approximation of the index proposed in [6]. Thanks to this approximation, the maximization step benefits from an explicit solution: The resulting principal direction a^k is the eigenvector associated to the maximum eigenvalue of $(V_{k-1}^*)^{-1} V_{k-1}$ where

$$V_{k-1}^* = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n m_{i,j}^{k-1} (R_i^{k-1} - R_j^{k-1})^t (R_i^{k-1} - R_j^{k-1})$$

is proportional to the local covariance matrix. $(V_{k-1}^*)^{-1}$ should be read as the generalized inverse of the singular matrix V_{k-1}^* . Indeed, since R^{k-1} is orthogonal to $\{a^1, \dots, a^{k-1}\}$ from (8.5), V_{k-1}^* is, at most, of rank $p - k + 1$. Note that this approach is equivalent to Lebart's one when the contiguity matrix M is symmetric.

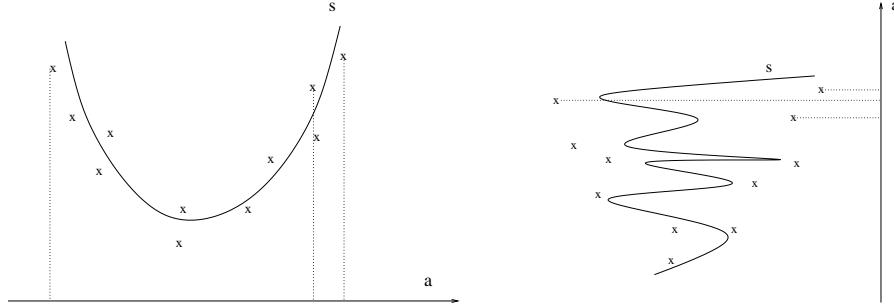


Fig. 8.1. Left: axis a such that the associated projection P_a preserves the first-order neighborhood structure. The regression function s correctly fits the dataset. Right: axis a for which P_a does not preserve the first-order neighborhood structure. The regression function s cannot yield a good approximation of the dataset.

8.5 Illustration on Real and Simulated Data

Our first illustration is done on the "DistortedSShape" simulated dataset introduced in [30], paragraph 5.2.1 and available on-line³. The dataset consists of 100 data points in \mathbb{R}^2 and located around a one-dimensional curve (solid line on Figure 8.2). The bold dashed curve is the one-dimensional manifold estimated by the principal curves approach [24]. The estimated curve fails to follow the shape of the original curve. Using the auto-associative model, the estimated one-dimensional manifold (dashed curve) is closer to the original one. In this experiment, we used one iteration of Algorithm 1 with the continuity index (8.17) in combination with a projection estimate of the regression functions. A basis of $N = 4$ cubic splines was used to compute the projection.

Our second illustration is performed on the "dataset I - Five types of breast cancer" provided to us by the organizers of the "Principal Manifolds-2006" workshop. The dataset [49] is available on-line⁴. It consists of microarray data containing logarithms of expression levels of $p = 17816$ genes in $n = 286$ samples. The data is divided into five types of breast cancer (lumA, lumB, normal, errb2 and basal) plus an unclassified group. Before all, let us note that, since n points are necessarily located on a linear subspace of dimension $n - 1$, the covariance matrix is at most of rank $n - 1 = 285$. Thus, as a preprocessing step, the dimension of the data is reduced to 285 by a classical PCA, and this, without any loss of information. Forgetting the labels, *i.e.* without using the initial classification into five types of breast cancer, the information ratio Q_d (see Corollary 1) obtained by the classical PCA and the generalized one (basing on auto-associative models), are compared. Figure 8.3 illustrates the behavior of Q_d as the dimension d of the model

³ <http://www.iro.umontreal.ca/~kegl/research/pcurves>

⁴ <http://www.ihes.fr/~zinovyev/princmanif2006/>

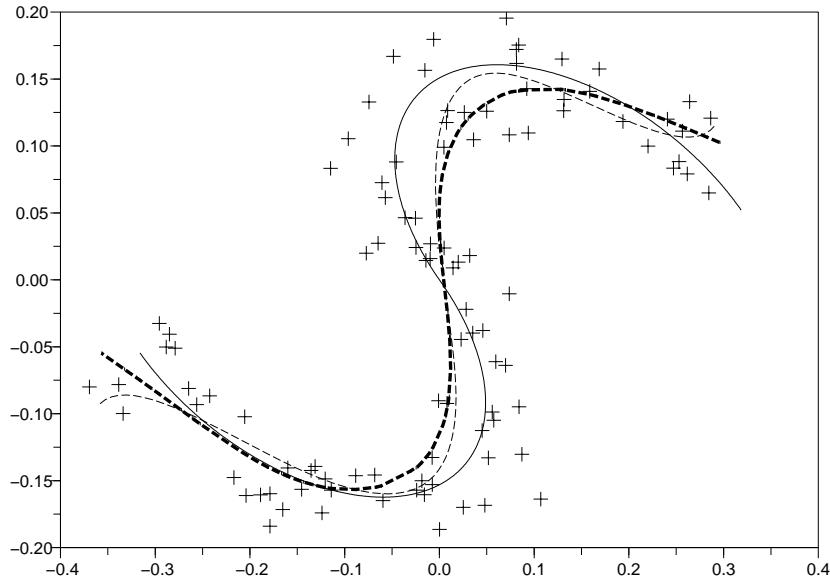


Fig. 8.2. Comparison of one-dimensional estimated manifolds on a simulated dataset. solid line: original curve, dashed line: curve estimated from the auto-associative model approach, bold dashed line: principal curve estimated by the approach proposed in [24].

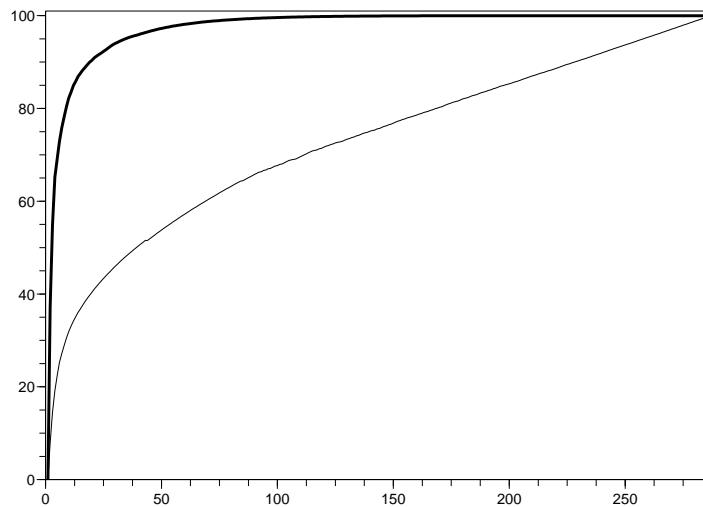


Fig. 8.3. Forgetting the labels, information ratio Q_d as a function of d on a real dataset. solid line: classical PCA, bold line: generalized PCA based on auto-associative models.

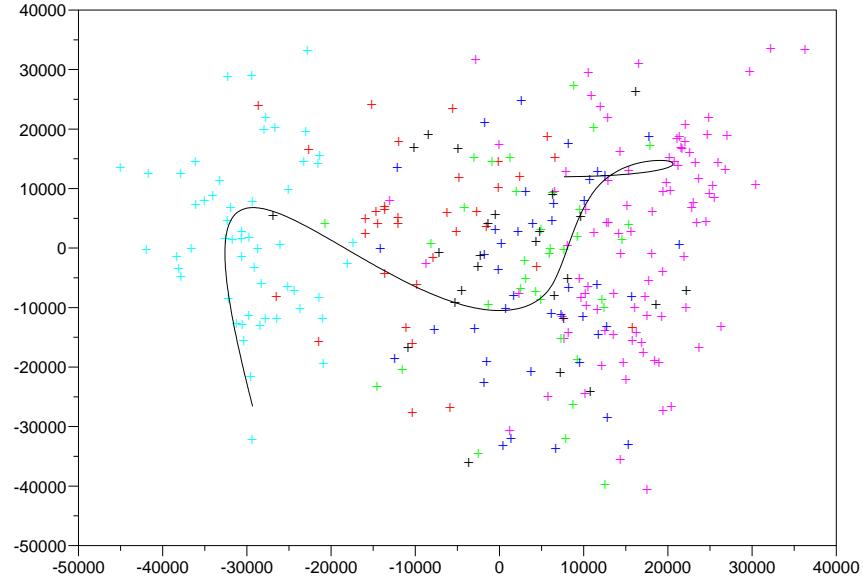


Fig. 8.4. One-dimensional manifold estimated on a real dataset with the auto-associative models approach and projected on the principal plane.

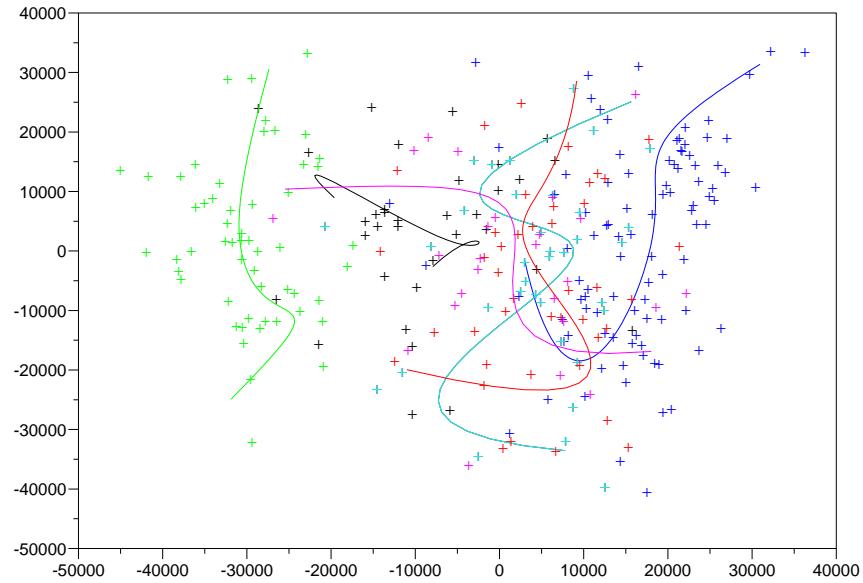


Fig. 8.5. One-dimensional manifolds estimated on each type of cancer of the real dataset with the auto-associative models approach, and projected on the principal plane.

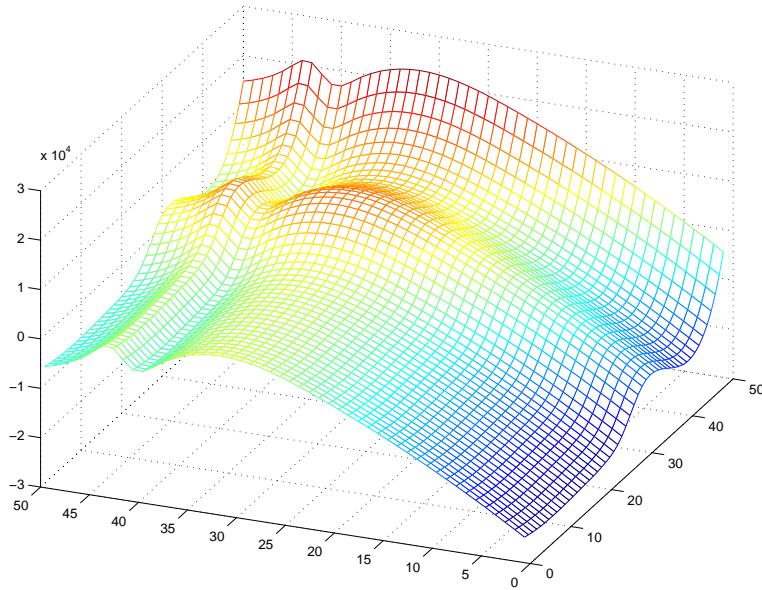


Fig. 8.6. Two-dimensional manifold estimated on a real dataset with the auto-associative models approach and projected on the three first principal axes.

increases. The bold curve, corresponding to the auto-associate model, was computed with the contiguity index (8.17) in combination with a projection estimate of the regression functions. A basis of $N = 2$ cubic splines was used to compute the projection. One can see that the generalized PCA yields far better approximation results than the classical one.

As an illustration, the one-dimensional manifold is superimposed to the dataset on Figure 8.4. Each class is represented with a different gray level. For the sake of the visualization, the dataset as well as the manifold are projected on the principal plane. Similarly, the two-dimensional manifold is represented on Figure 8.6 on the linear space spanned by the three first principal axes. Taking into account the labels, it is also possible to compute the one-dimensional manifold associated to each type of cancer and to the unclassified points, see Figure 8.5. Each manifold then represents a kind of skeleton of the corresponding dataset.

Other illustrations can be found in [5], Chapter 4, where auto-associative models are applied to some image analysis problems.

References

1. Bellman, R.: Dynamic programming. Princeton university press, (1957)
2. Bosq, D. and Lecoutre, J.P.: Théorie de l'estimation fonctionnelle. Economie et Statistiques avancées. Economica, Paris (1987)
3. Carroll, J.D. and Arabie P.: Multidimensionnal scaling. Annual Rev. of Psychology, **31**, 607–649 (1980)
4. Caussinus, H. and Ruiz-Gazen, A.: Metrics for finding typical structures by means of principal component analysis. In: Data science and its Applications, pages 177–192. Harcourt Brace, Japan (1995)
5. Chalmond, B.: Modeling and inverse problems in image analysis. In: Applied Mathematics Science series, volume 155. Springer, New-York (2002)
6. Chalmond, B. and Girard, S.: Nonlinear modeling of scattered multivariate data and its application to shape change. IEEE Pattern Analysis and Machine Intelligence, **21** (5), 422–432 (1999)
7. Cheng, B. and Titterington, D.M.: Neural networks: A review from a statistical perspective. Statistical Science, **9** (1), 2–54 (1994)
8. Delicado, P.: Another look at principal curves and surfaces. Journal of Multivariate Analysis, **77**, 84–116 (2001)
9. Demartines, P. and Hérault, J.: Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. IEEE Trans. on Neural Networks, **8** (1), 148–154 (1997)
10. Diamantaras, K.L. and Kung, S.Y.: Principal component neural networks. Wiley, New-York (1996)
11. Eubank, R.L.: Spline smoothing and non-parametric regression. Decker (1990)
12. Ferraty, F. and Vieu, P.: Nonparametric modelling for functional data. Springer (2005)
13. Friedman, J.H.: Exploratory projection pursuit. Journal of the American Statistical Association, 82(397), 249–266 (1987)
14. Friedman, J.H. and Stuetzle, W.: Projection pursuit regression. Journal of the American Statistical Association, **76** (376), 817–823 (1981)
15. Friedman, J.H. and Tukey, J.W.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. on Computers, **C23** (9), 881–890 (1974)
16. Girard, S.: A nonlinear PCA based on manifold approximation. Computational Statistics, **15** (2), 145–167 (2000)
17. Girard, S., Chalmond, B., and Dinten, J.-M.: Position of principal component analysis among auto-associative composite models. Comptes-Rendus de l'Académie des Sciences, Série I, **326**, 763–768 (1998)
18. Girard, S. and Iovleff, S.: Auto-associative models and generalized principal component analysis. Journal of Multivariate Analysis, **93** (1), 21–39 (2005)
19. Gorban, A. and Zinovyev, A.: Elastic principal graphs and manifolds and their practical applications. Computing, **75** (4), 359–379 (2005)
20. Green, P.J. and Silverman, B.W.: Non-parametric regression and generalized linear models. Chapman and Hall, London (1994)
21. Hall, P.: On polynomial-based projection indices for exploratory projection pursuit. The Annals of Statistics, **17** (2), 589–605 (1990)
22. Härdle, W.: Applied nonparametric regression. Cambridge University Press, Cambridge (1990)
23. Hartigan, J.A.: Clustering algorithms. Wiley, New-York (1995)

24. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association*, **84** (406), 502–516 (1989)
25. Hastie, T., Tibshirani, R., and Friedman, J.: The elements of statistical learning. In: Springer Series in Statistics. Springer (2001)
26. Huber, P.J.: Projection pursuit. *The Annals of Statistics*, **13** (2), 435–475 (1985)
27. Jolliffe, I.: Principal Component Analysis. Springer-Verlag, New York (1986)
28. Jones, M.C. and Sibson R.: What is projection pursuit? *Journal of the Royal Statistical Society A*, **150**, 1–36 (1987)
29. Karhunen, J. and Joutsensalo, J.: Generalizations of principal component analysis, optimization problems and neural networks. *Neural Networks*, **8**, 549–562 (1995)
30. Kégl, B.: "Principal curves: learning, design, and applications". PhD thesis, Concordia University, Canada (1999)
31. Kégl, B., Krzyzak, A., Linder, T., and Zeger, K.: A polygonal line algorithm for constructing principal curves. In: Proceedings of 12h NIPS, pages 501–507, Denver, Colorado (1998)
32. Klinke, S. and Grassmann, J. Projection pursuit regression. In: Wiley Series in Probability and Statistics, pages 471–496. Wiley (2000)
33. Kohonen, T.: Self-organization of topologically correct feature maps. *Biological cybernetics*, **43**, 135–140 (1982)
34. Kohonen, T.: Self-organization and associative memory, 3rd edition. Springer-Verlag, Berlin (1989)
35. Kruskal, J.B. and Wish, M.: Multidimensional scaling. Sage, Beverly Hills (1978)
36. Lebart L.: Contiguity analysis and classification. In: Opitz O. Gaul W. and Schader M. (eds.) Data Analysis, pages 233–244. Springer, Berlin (2000)
37. LeBlanc, M. and Tibshirani, R.: Adaptive principal surfaces. *Journal of the American Statistical Association*, **89** (425), 53–64 (1994)
38. Lee, J.A., Lendasse, A., and Verleysen, M.: Curvilinear distance analysis versus isomap. In: European Symposium on Artificial Neural Networks, pages 185–192. Bruges, Belgium (2002)
39. Milnor, J.: Topology from the differentiable point of view. University press of Virginia, Charlottesville (1965)
40. Pan, J-X., Fung, W-K., and Fang, K-T.: Multiple outlier detection in multivariate data using projection pursuit techniques. *Journal of Statistical Planning and Inference*, **83** (1), 153–167 (2000)
41. Parlett, B. N.: The symmetric eigenvalue problem. In: Classics in Applied Mathematics, vol. 20. SIAM, Philadelphia (1997)
42. Roweis, S.T. and Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**, 2323–2326 (2000)
43. Sammon, J.W.: A nonlinear mapping algorithm for data structure analysis. *IEEE Transactions on Computer*, **18** (5), 401–409 (1969)
44. Shepard, R.N. and Carroll, J.D.: Parametric representation of nonlinear data structures. In: Krishnaiah, P.R. (ed.) Int. Symp. on Multivariate Analysis, pages 561–592. Academic-Press (1965)
45. Tarpey, T. and Flury, B.: Self-consistency: A fundamental concept in statistics. *Statistical Science*, **11** (3), 229–243 (1996)
46. Tenenbaum, J.B., de Silva, V., and Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**, 2319–2323 (2000)

47. Tibshirani, R.: Principal surfaces revisited. *Statistics and computing*, **2**, 183–190 (1992)
48. Vlachos, M., Domeniconi, C., Gunopulos, D., Kollios, G., and Koudas, N.: Non-linear dimensionality reduction techniques for classification and visualization. In: *Proceedings of 8th SIGKDD*, pages 23–26. Edmonton, Canada (2002)
49. Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, et al.: Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet* **365**, 671–679 (2005)

Beyond The Concept of Manifolds: Principal Trees, Metro Maps, and Elastic Cubic Complexes

Alexander N. Gorban^{1,3}, Neil R. Sumner¹, and Andrei Y. Zinovyev^{2,3}

¹ University of Leicester, University Road, Leicester, LE1 7RH, UK,
`{ag153,nrs7}@le.ac.uk`

² Institut Curie, 26, rue d'Ulm, Paris, 75248, France,
`andrei.zinovyev@curie.fr`

³ Institute of Computational Modeling of Siberian Branch of Russian Academy of Sciences, Krasnoyarsk, Russia

Summary. Multidimensional data distributions can have complex topologies and variable local dimensions. To approximate complex data, we propose a new type of low-dimensional “principal object”: a *principal cubic complex*. This complex is a generalization of linear and non-linear principal manifolds and includes them as a particular case. To construct such an object, we combine a method of *topological grammars* with the minimization of an elastic energy defined for its embedment into multidimensional data space. The whole complex is presented as a system of nodes and springs and as a product of one-dimensional continua (represented by graphs), and the grammars describe how these continua transform during the process of optimal complex construction.

The simplest case of a topological grammar (“add a node”, “bisect an edge”) is equivalent to the construction of “principal trees”, an object useful in many practical applications. We demonstrate how it can be applied to the analysis of bacterial genomes and for visualization of cDNA microarray data using the “metro map” representation.

Key words: principal trees, topological grammars, principal manifolds, elastic functional, data visualization

9.1 Introduction and Overview

In this paper, we discuss a classical problem: how to approximate a finite set D in R^m for relatively large m by a finite subset of a regular low-dimensional object in R^m . In application, this finite set is a dataset, and this problem arises in many areas: from data visualization to fluid dynamics.

The first hypothesis we have to check is: whether the dataset D is situated near a low-dimensional affine manifold (plane) in R^m . If we look for a point,

straight line, plane, ... that minimizes the average squared distance to the datapoints, we immediately come to Principal Component Analysis (PCA). PCA is one of the most seminal inventions in data analysis. Now it is textbook material and celebrated the 100th anniversary [26]. Nonlinear generalization of PCA is a great challenge, and many attempts have been made to answer it. Two of them are especially important for our consideration: Kohonen's Self-Organizing Maps (SOM) and principal manifolds.

With the *SOM* algorithm [19] we take a finite metric space Y with metric ρ and try to map it into R^m with (a) the best preservation of initial structure in the image of Y and (b) the best approximation of the dataset D . The SOM algorithm has several setup variables to regulate the compromise between these goals. We start from some initial approximation of the map, $\phi_1 : Y \rightarrow R^m$. On each (k -th) step of the algorithm we have a datapoint $x \in D$ and a current approximation $\phi_k : Y \rightarrow R^m$. For these x and ϕ_k we define an "owner" of x in Y : $y_x = \operatorname{argmin}_{y \in Y} \|x - \phi_k(y)\|$. The next approximation, ϕ_{k+1} , is

$$\phi_{k+1}(y) = h_k w(\rho(y, y_x))(x - \phi_k(y)). \quad (9.1)$$

Here h_k is a step size, $0 \leq w(\rho(y, y_x)) \leq 1$ is a monotonically decreasing neighbourhood function. There are many ways to combine steps (9.1) in the whole algorithm. The idea of SOM is flexible and seminal, it has plenty of applications and generalizations, but, strictly speaking, we don't know what we are looking for. We have the algorithm, but no independent definition: SOM is a result of the algorithm at work. The attempts to define SOM as solution of a minimization problem for some energy functional were not very successful [5], however, this led to the development of the optimization-based Generative Topographic Mapping (GTM) method [1].

For a known probability distribution, *principal manifolds* were introduced as lines or surfaces passing through "the middle" of the data distribution [17]. This intuitive vision was transformed into the mathematical notion of *self-consistency*: every point x of the principal manifold M is a conditional expectation of all points z that are projected into x . Neither manifold, nor projection need to be linear: just a differentiable projection π of the data space (usually it is R^m or a domain in R^m) onto the manifold M with the self-consistency requirement for conditional expectations: $x = \mathbf{E}(z|\pi(z) = x)$. For a finite dataset D , only one or zero datapoints are typically projected into a point of the principal manifold. In order to avoid overfitting, we have to introduce smoothers that become an essential part of the principal manifold construction algorithms.

SOMs give the most popular approximations for principal manifolds: we can take for Y a fragment of a regular k -dimensional grid and consider the resulting SOM as the approximation to the k -dimensional principal manifold (see, for example, [24, 29]). Several original algorithms for construction of principal curves [18] and surfaces for finite datasets were developed during last decade, as well as many applications of this idea. The recently proposed

idea of local principal curves [4] allows to approximate data with nonlinear, branched, and disconnected one-dimensional continua.

In 1996, in a discussion about SOM at the 5th Russian National Seminar in Neuroinformatics, a method of multidimensional data approximation based on elastic energy minimization was proposed (see [8, 30, 14] and the bibliography there). This method is based on the analogy between the principal manifold and an elastic membrane (and plate). Following the metaphor of elasticity, we introduce two quadratic smoothness penalty terms. This allows one to apply standard minimization of quadratic functionals (i.e., solving a system of linear algebraic equations with a sparse matrix). The elastic map approach led to many practical applications, in particular in data visualization and missing data values recovery. It was applied for visualization of economic and socio-logical tables [10, 11, 13, 30], to visualization of natural [30] and genetic texts [12, 31], and to recovering missing values in geophysical time series [3]. Modifications of the algorithm and various adaptive optimization strategies were proposed for modeling molecular surfaces and contour extraction in images [14].

9.1.1 Elastic Principal Graphs

Let G be a simple undirected graph with set of vertices Y and set of edges E . For $k \geq 2$ a k -star in G is a subgraph with $k + 1$ vertices $y_{0,1,\dots,k} \in Y$ and k edges $\{(y_0, y_i) \mid i = 1, \dots, k\} \subset E$. Suppose for each $k \geq 2$, a family S_k of k -stars in G has been selected. We call a graph G with selected families of k -stars S_k an *elastic graph* if, for all $E^{(i)} \in E$ and $S_k^{(j)} \in S_k$, the correspondent elasticity moduli $\lambda_i > 0$ and $\mu_{kj} > 0$ are defined. Let $E^{(i)}(0), E^{(i)}(1)$ be vertices of an edge $E^{(i)}$ and $S_k^{(j)}(0), \dots, S_k^{(j)}(k)$ be vertices of a k -star $S_k^{(j)}$ (among them, $S_k^{(j)}(0)$ is the central vertex). For any map $\phi : Y \rightarrow R^m$ the *energy of the graph* is defined as

$$\begin{aligned} U^\phi(G) := & \sum_{E^{(i)}} \lambda_i \left\| \phi(E^{(i)}(0)) - \phi(E^{(i)}(1)) \right\|^2 \\ & + \sum_{S_k^{(j)}} \mu_{kj} \left\| \sum_{i=1}^k \phi(S_k^{(j)}(i)) - k\phi(S_k^{(j)}(0)) \right\|^2. \end{aligned} \quad (9.2)$$

Very recently, a simple but important fact was noticed [16]: every system of elastic finite elements could be represented by a system of springs, if we allow some springs to have negative elasticity coefficients. The energy of a k -star s_k in R^m with y_0 in the center and k endpoints $y_{1,\dots,k}$ is $u_{s_k} = \mu_{s_k} (\sum_{i=1}^k |y_i - ky_0|^2)$, or, in the spring representation, $u_{s_k} = k\mu_{s_k} \sum_{i=1}^k (y_i - y_0)^2 - \mu_{s_k} \sum_{i>j} (y_i - y_j)^2$. Here we have k positive springs with coefficients $k\mu_{s_k}$ and $k(k-1)/1$ negative springs with coefficients $-\mu_{s_k}$.

For a given map $\phi : Y \rightarrow R^m$ we divide the dataset D into subsets K^y , $y \in Y$. The set K^y contains the data points for which the node $\phi(y)$ is the closest one in $\phi(Y)$:

$$K^{y_j} = \{x_i | y_j = \arg \min_{y_k \in Y} \|y_k - x_i\|\} . \quad (9.3)$$

The *energy of approximation* is:

$$U_A^\phi(G, D) = \frac{1}{\sum_{x \in D} w(x)} \sum_{y \in Y} \sum_{x \in K^y} w(x) \|x - \phi(y)\|^2 , \quad (9.4)$$

where $w(x) \geq 0$ are the point weights. In the simplest case $w(x) = 1$ but it might be useful to make some points ‘heavier’ or ‘lighter’ in the initial data. The normalization factor $1/\sum_{x \in D} w(x)$ in (9.4) is needed for the law of large numbers⁴.

9.2 Optimization of Elastic Graphs for Data Approximation

9.2.1 Elastic Functional Optimization

The simple algorithm for minimization of the energy $U^\phi = U_A^\phi(G, D) + U^\phi(G)$ is the splitting algorithm, in the spirit of classical K -means clustering:

1. For a given system of sets $\{K^y | y \in Y\}$ we minimize U^ϕ (it is the minimization of a positive quadratic functional). This is done by solving a system of linear algebraic equations for finding new positions of nodes $\{\phi(y_i)\}$:

$$\sum_{k=1}^p a_{jk} \phi(y_k) = \frac{1}{\sum_{x \in D} w(x)} \sum_{x \in K^{y_j}} w(x) x . \quad (9.5)$$

2. For a given ϕ we find new $\{K^y\}$ (9.3).
3. Go to step 1 and so on; stop when there are no significant changes in ϕ .

Here,

$$a_{jk} = \frac{n_j \delta_{jk}}{\sum_{x \in D} w(x)} + e_{jk} + s_{jk}, \quad n_j = \sum_{x \in K^{y_j}} w(x) \quad (j = 1 \dots p) , \quad (9.6)$$

⁴ For more details see Gorban & Zinovyev paper in this volume.

δ_{jk} is Kronecker's δ , and matrices e_{jk} and s_{jk} depend only on elasticity modules and on the content of the sets $\{E^{(i)}\}$ and $\{S_k^{(i)}\}$, thus they need not be recomputed if the structure of the graph was not changed.

Matrix a_{jk} is sparse. In practical computations it is easier to compute only non-zero entries of the e_{jk} and s_{jk} matrices. This can be done using the following scheme:

1. Initialize the s_{ij} matrix to zero.
2. For each k -star $S_k^{(i)}$ with weight μ_i , outer nodes y^{N_1}, \dots, y^{N_k} and central node y^{N_0} , the s_{ij} matrix is updated as follows ($1 \leq l, m \leq k$):

$$\begin{aligned} s'_{N_0 N_0} &= s_{N_0 N_0} + k^2 \mu_i, & s'_{N_l N_m} &= s_{N_l N_m} + \mu_i, \\ s'_{N_0 N_l} &= s_{N_0 N_l} - k \mu_i, & s'_{N_l N_0} &= s_{N_l N_0} - k \mu_i. \end{aligned}$$

3. Initialize the e_{ij} matrix to zero.
4. For each edge $E^{(i)}$ with weight λ_i , one vertex y^{k1} and the other vertex y^{k2} , the e_{jk} matrix is updated as follows:

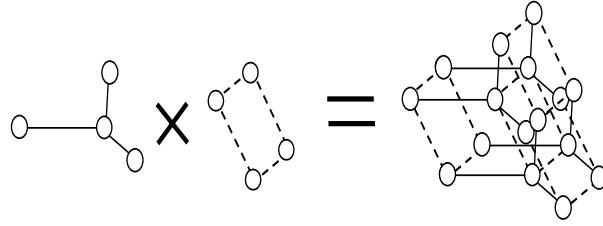
$$\begin{aligned} e_{k_1 k_1} &= e_{k_1 k_1} + \lambda_i, & e_{k_2 k_2} &= e_{k_2 k_2} + \lambda_i, \\ e_{k_1 k_2} &= e_{k_1 k_2} - \lambda_i, & e_{k_2 k_1} &= e_{k_2 k_1} - \lambda_i. \end{aligned}$$

This algorithm gives a local minimum, and the global minimization problem arises. There are many methods for improving the situation, but without guarantee of finding the global minimization (see, for example, accompanying paper [15]).

9.2.2 Optimal Application of Graph Grammars

The next problem is the elastic graph construction. Here we should find a compromise between simplicity of graph topology, simplicity of geometrical form for a given topology, and accuracy of approximation. Geometrical complexity is measured by the graph energy $U^\phi(G)$, and the error of approximation is measured by the energy of approximation $U_A^\phi(G, D)$. Both are included in the energy U^ϕ . Topological complexity will be represented by means of elementary transformations: it is the length of the energetically optimal chain of elementary transformation from a given set applied to the initial simple graph.

The graph grammars [28, 21] provide a well-developed formalism for the description of elementary transformations. An elastic graph grammar is presented as a set of production (or substitution) rules. Each rule has a form $A \rightarrow B$, where A and B are elastic graphs. When this rule is applied to an elastic graph, a copy of A is removed from the graph together with all its incident edges and is replaced with a copy of B with edges that connect B to the graph. For a full description of this language we need the notion of a

**Fig. 9.1.** Cartesian product of graphs.

labeled graph. Labels are necessary to provide the proper connection between B and the graph.

A link in the energetically optimal transformation chain is constructed by finding a transformation application that gives the largest energy descent (after an optimization step), then the next link, and so on, until we achieve the desirable accuracy of approximation, or the limit number of transformations (some other termination criteria are also possible). The selection of an energetically optimal application of transformations by the trial optimization steps is time-consuming. There exist alternative approaches. The preselection of applications for a production rule $A \rightarrow B$ can be done through the comparison of the energy of copies of A with its incident edges and stars in the transformed graph G .

9.2.3 Factorization and Transformation of Factors

If we approximate multidimensional data by a k -dimensional object, the number of points (or, more generally, elements) in this object grows with k exponentially. This is an obstacle for grammar-based algorithms even for modest k , because for analysis of the rule $A \rightarrow B$ applications we should investigate all isomorphic copies of A in G . The natural way to avoid this obstacle is the principal object factorization. Let us represent an elastic graph as a Cartesian product of graphs (Fig. 9.1).

The Cartesian product $G_1 \times \dots \times G_r$ of elastic graphs G_1, \dots, G_r is an elastic graph with vertex set $V_1 \times \dots \times V_r$. Let $1 \leq i \leq r$ and $v_j \in V_j$ ($j \neq i$). For this set of vertices, $\{v_j\}_{j \neq i}$, a copy of G_i in $G_1 \times \dots \times G_r$ is defined with vertices $(v_1, \dots, v_{i-1}, v, v_{i+1}, \dots, v_r)$ ($v \in V_i$), edges

$$((v_1, \dots, v_{i-1}, v, v_{i+1}, \dots, v_r), (v_1, \dots, v_{i-1}, v', v_{i+1}, \dots, v_r)), \quad (v, v') \in E_i,$$

and, similarly, k -stars of the form $(v_1, \dots, v_{i-1}, S_k, v_{i+1}, \dots, v_r)$, where S_k is a k -star in G_i . For any G_i there are $\prod_{j,j \neq i} |V_j|$ copies of G_i in G . Sets of edges and k -stars for Cartesian product are unions of that set through all copies of all factors. A map $\phi : V_1 \times \dots \times V_r \rightarrow R^m$ maps all the copies of factors into R^m too. *The energy of the elastic graph product is the energy sum of all factor copies.* It is, of course, a quadratic functional of ϕ .

The only difference between the construction of general elastic graphs and factorized graphs is in the application of the transformations. For factorized graphs, we apply them to factors. This approach significantly reduces the amount of trials in selection of the optimal application. The simple grammar with two rules, “add a node to a node, or bisect an edge,” is also powerful here, it produces products of primitive elastic trees. For such a product, the elastic structure is defined by the topology of the factors.

9.3 Principal Trees (Branching Principal Curves)

9.3.1 Simple Graph Grammar (“Add a Node”, “Bisect an Edge”)

As a simple (but already rather powerful) example we use a system of two transformations: “add a node to a node” and “bisect an edge.” These transformations act on a class of *primitive elastic graphs*: all non-terminal nodes with k edges are centers of elastic k -stars, which form all the k -stars of the graph. For a primitive elastic graph, the number of stars is equal to the number of non-terminal nodes – the graph topology prescribes the elastic structure.

The transformation “*add a node*” can be applied to any vertex y of G : add a new node z and a new edge (y, z) . The transformation “*bisect an edge*” is applicable to any pair of graph vertices y, y' connected by an edge (y, y') : Delete edge (y, y') , add a vertex z and two edges, (y, z) and (z, y') . The transformation of the elastic structure (change in the star list) is induced by the change of topology, because the elastic graph is primitive. This two-transformation grammar with energy minimization builds *principal trees* (and principal curves, as a particular case) for datasets. A couple of examples are presented on Fig. 9.3. For applications, it is useful to associate one-dimensional continua with these principal trees. Such a continuum consists of node images $\phi(y)$ and of pieces of straight lines that connect images of linked nodes.

9.3.2 Visualization of Data Using “Metro Map” Two-Dimensional Tree Layout

A principal tree is embedded into a multidimensional data space. It approximates the data so that one can project points from the multidimensional space into the closest node of the tree (other projectors are also possible, for example, see the accompanying paper [15]). The tree by its construction is a one-dimensional object, so this projection performs dimension reduction of the multidimensional data. The question is how to represent the result of this projection? For example, how to produce a tree layout on the two-dimensional surface of paper sheet? Of course, there are many ways to layout a tree on a plane. It is always possible to find a tree layout without edge intersection. But it would be very nice if both some tree properties and global distance relations would be represented using the layout. We can require that

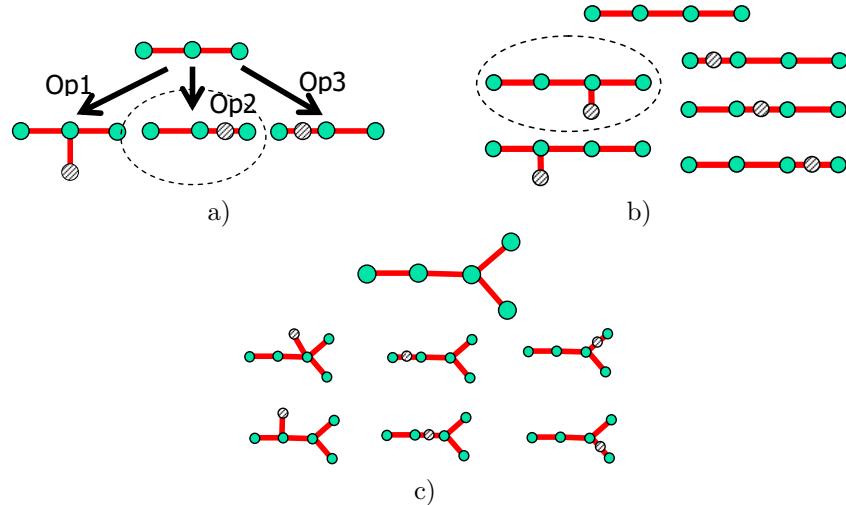


Fig. 9.2. Illustration of the simple “add node to a node” or “bisect an edge” graph grammar application. a) We start with a simple 2-star from which one can generate three distinct graphs shown. The “Op1” operation is adding a node to a node, operations “Op2” and “Op3” are edge bisections (here they are topologically equivalent to adding a node to a terminal node of the initial 2-star). For illustration let us suppose that the “Op2” operation gives the biggest elastic energy decrement, thus it is the “optimal” operation. b) From the graph obtained one can generate 5 distinct graphs and choose the optimal one. c) The process is continued until a definite number of nodes is inserted.

1) In a two-dimensional layout, all k -stars should be represented equiangular, because the ideal configuration of a k -star with small energy is equiangular and with equal edge lengths.

2) The edge lengths should be proportional to their length in the multi-dimensional embedding; thus one can represent between-node distances.

This defines a tree layout up to global rotation and scaling and also up to changing the order of leaves in every k -star. We can change this order to eliminate edge intersections, but the result can not be guaranteed. In order to represent the global distance structure, we found that a good approximation for the order of k -star leaves can be taken from the projection of every k -star on the linear principal plane calculated for all data points, or on the local principal plane in the vicinity of the k -star, calculated only for the points close to this star.

In the current implementation of the method, after defining the initial approximation, a user manually modifies the layout switching the order of k -star leaves to get rid of edge intersections. The edge lengths are also modifiable. Usually it is possible to avoid edge intersections in the layout after a small number of initial layout modifications. This process could be also fully auto-

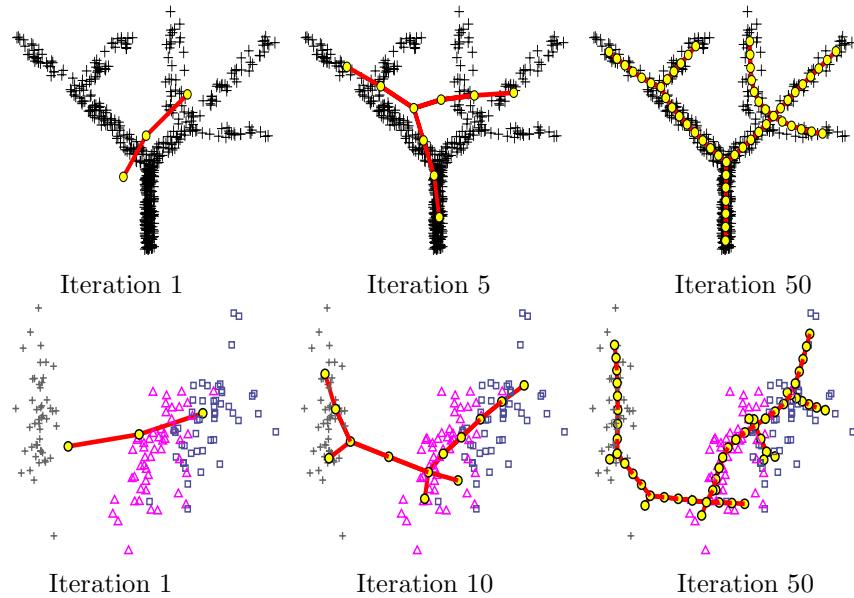


Fig. 9.3. Applying a simple “add a node to a node or bisect an edge” grammar to construct principal elastic trees (one node is added per iteration). Upper row: an example of two-dimensional branching distribution of points. Lower row: the classical benchmark, the “iris” four-dimensional dataset (point shapes distinguish three classes of points), the dataset and principal tree are presented in projection onto the plane of the first two principal components.

mated, by a greedy optimization algorithm, for example, but this possibility is not yet implemented.

The point projections are then represented as pie diagrams, where the size of the diagram reflects the number of points projected into the corresponding tree node. The sectors of the diagram allow us to show proportions of points of different classes projected into the node (see an example on Fig. 9.4).

We call this type of visualization a “metro map” since it is a schematic and “idealized” representation of the tree and the data distribution with inevitable distortions made to produce a nice 2D layout, but using this map one can still estimate the distance from a point (tree node) to a point passing through other points. This map is inherently unrooted (as a real metro map). It is useful to compare this metaphor with trees produced by hierarchical clustering where the metaphor is closer to a “genealogy tree”.

9.3.3 Example of Principal Cubic Complex: Product of Principal Trees

Principal trees are one-dimensional objects. To illustrate the idea of a d -dimensional principal cubic complex that is a cartesian product of graphs (see

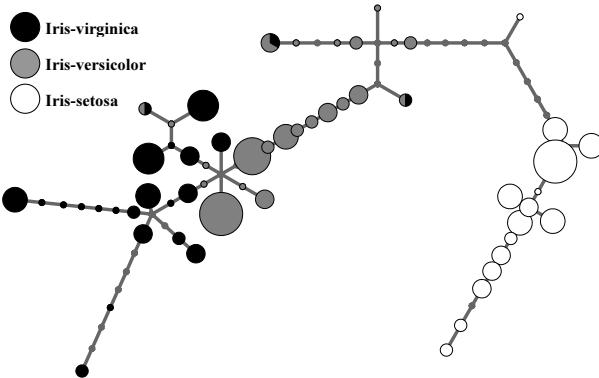


Fig. 9.4. Two-dimensional layout of the principal tree constructed for the iris dataset. In this layout all stars are equiangular and the edge lengths are proportional to the real edge lengths in the multidimensional space. The data points were projected into the closest tree nodes. The circle radii are proportional to the number of points projected into each node. Three different colors (or gray tints on the gray version of the image) denote three point classes. If points of different classes were projected into the same tree node then the number of points of every class is visualized by the pie diagram.

Fig. 9.1), we implemented an algorithm, constructing products of principal trees. The pseudocode for this algorithm is provided below:

1. Initialize one factor graph consisting of one edge connecting two nodes positioned half a standard deviation around the data mean.
2. Optimize the node positions.
3. Test the addition of a node to each star in each factor, optimizing the node postions of the Cartesian product.
4. Test the bisection of each edge in each factor, optimising the node positions of the Cartesian product.
5. Test the initialisation of another factor graph consisting of one edge using the scheme:
 - a) For each node use the k-means algorithm with $k=2$ on the data class associated with the node, intializing using the mean of the data class and the node position.
 - b) Normalize the vector between the 2 centres thus obtained and scale it to the size of the mean of the edge lengths already incident with the node.
 - c) Optimize the node positions.
6. Choose the transformation from steps (3) to (5) which gives the greatest energy descent.
7. Until the stopping criteria are met repeat steps (3) - (6).

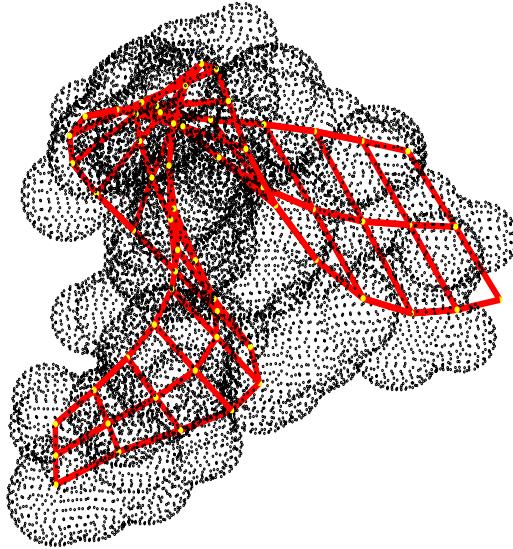


Fig. 9.5. A cubic complex (2-dimensional, “add a node, bisect an edge” graph grammar) constructed for a distribution of points on the Van-der-Waals surface of a fragment of a DNA molecule (dots). The result is the product of two unbranched trees which “discovers” the double-helical DNA structure.

Thus, the structure of a principal cubic complex is defined by its dimension and the graph grammar applied for its construction. A simple example of 2-dimensional principal tree, i.e. cubic complex constructed with the simplest “add a node or bisect an edge” grammar (see Fig. 9.2) is given in Fig. 9.5. Here the cubic complex is constructed for a distribution of points on the molecular surface of a fragment of a DNA molecule (compare with the application of the method of elastic maps to the same dataset, given in the accompanying paper [15]). The method of topological grammars [9] “discovers” the double-helical structure of the DNA molecule. Note that in this example the energy optimization gave no branching in both factors (trees) and as a result we obtained a product of two simple poly-lines. In other situations, the resulting cubic complex could be more complicated, with branching in one or both factors.

9.4 Analysis of the Universal 7-Cluster Structure of Bacterial Genomes

In this section we describe the application of the method of topological grammars to the analysis of the cluster structure of bacterial genomes. This struc-

ture appears as a result of projecting a genome sequence into a multidimensional space of short word frequencies [6, 7]. In particular, we show that a one-dimensional principal tree can reveal a signal invisible when three-dimensional PCA is applied.

9.4.1 Brief Introduction

One of the most exciting problems in modern science is to understand the organization of living matter by reading genomic sequences. The information that is needed for a living cell to function is encoded in a long molecule of DNA. It can be presented as a text that has only four letters A, C, G and T.

One distinctive message in a genomic sequence is a piece of text, called a *gene*. Genes can be oriented in the sequence in the forward and backward directions. In bacterial genomes genes are always continuous from their start to the stop signal.

It was one of many great discoveries of the twentieth century that biological information is encoded in genes by means of triplets of letters, called *codons* in the biological literature. In the famous paper by Crick *et al.* [2], this fact was proven by genetic experiments carried out on bacteria mutants.

In nature, there is a special mechanism that is designed to read genes. It is evident that as the information is encoded by non-overlapping triplets, it is critical for this mechanism to start reading a gene without a shift, from the first letter of the first codon to the last one; otherwise, the information decoded will be completely corrupted.

A *word* is any continuous piece of text that contains several subsequent letters. As there are no spaces in the text, separation into words is not unique.

The method we use to “decipher” genomic sequences is the following. We clip the whole text into fragments of 300 letters in length and calculate the frequencies of short words (of length 1–4) inside every fragment. This gives a description of the text in the form of a numerical table (word frequency vs fragment number).

As there are only four letters, there are four possible words of length 1 (singlets), $16 = 4^2$ possible words of length 2 (duplicates), $64 = 4^3$ possible words of length 3 (triplets) and $256 = 4^4$ possible words of length 4 (quadruplets). The first table contains four columns (frequency of every singlet) and the number of rows equals the number of fragments. The second table has 16 columns and the same number of rows, and so on.

These tables can be visualized by means of standard PCA. The result of such visualization is given on Fig. 9.6. As one can see from PCA plots, counting triplets gives an interesting flower-like pattern (described in details in [6, 7]), which can be interpreted as the existence of non-overlapping triplet code.

The triplet picture evidently contains 7 clusters, and it is more structured in the space than 1,2- and 4-tuples. To understand the 7-cluster structure, let us make some explanations.

Let us blindly cut the text into fragments. Any fragment can contain: (a) piece of gene in the forward direction; (b) piece of gene in the backward direction; (c) no genes (non-coding part); (d) a mixture of coding and non-coding.

Consider case (a). The fragment can overlap with a gene in three possible ways, with three possible shifts. If we start to read the information one triplet after another starting from the first letter of the fragment then we can read the gene correctly only if the fragment overlaps it with a correct shift. In general, if the start of the fragment is chosen randomly then we can read the gene in three possible ways. Thus, case a) generates three possible frequency distributions, “shifted” one with respect to another.

Case (b) is quite analogous and also gives three possible triplet distributions. They are not quite independent from the ones obtained at the step (a) for the following reason. The frequency of triplets is in fact the same as in the case (a), the difference is the triplets are read “from the end to the beginning” which produces a kind of mirror reflection of the triplet distributions from the case (a).

Case (c) will produce only one distribution which will be symmetrical with respect to the “shifts” (or rotations) in the first two cases, and there is a hypothesis that this is a result of genomic sequence evolution. Let us explain it.

The vitality of a bacterium depends on the correct functioning of all biological mechanisms. All these mechanisms are encoded in genes, and if something wrong happens with gene sequences (for example there is an error when DNA is duplicated), then the organism risks becoming non-vital. Nothing is perfect in our world and errors happen all the time, and in the DNA duplication process as well. These errors are called *mutations*.

The most dangerous mutations are those which change the reading frame, i.e. letter deletions or insertions. If such a mutation happens in the middle of a gene sequence, the rest of the gene becomes corrupted: the reading mechanism (which reads the triplets one by one and does not know about the mutation) will read it with a shift. Because of this the organisms with such mutations often die without leaving their off-spring. Conversely, if such a mutation happens in the non-coding part, where there are no genes, this does not lead to problems, and the organism leaves off-spring. Thus such mutations are constantly accumulated in the non-coding part making all three phase-specific distributions identical. The (d) case also produces mix of triplet distributions.

As a result, we have three distributions for case (a), three for case (b) and one, symmetrical for the “non-coding” fragments (case (c)). Because of natural statistical deviations and other reasons we have 7 clusters of points in the multidimensional space of triplet frequencies.

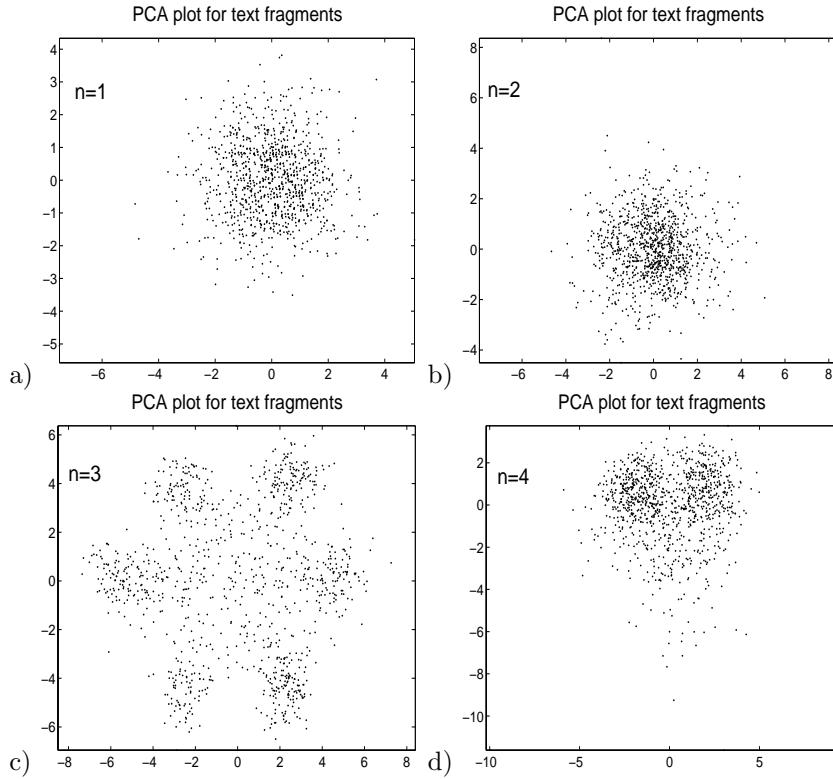


Fig. 9.6. PCA plots of word frequencies of different length. In c) one can see the most structured distribution. The structure is interpreted as the existence of a non-overlapping triplet code.

9.4.2 Visualization of the 7-Cluster Structure

It happens that the flower-like pattern of the 7-cluster structure is only one of several possible [6, 7] when we observe many bacterial genomes. Four “typical” configurations of 7-clusters observed in bacterial genomes are shown on Fig. 9.7.

Among these four typical configurations, there is one called “degenerative” (*Ercherichia coli* in Fig. 9.7). In this configuration three clusters corresponding to reading genes in the backward direction (reddish clusters) overlap with three clusters corresponding to reading genes in the forward direction (greenish clusters), when the distribution is projected in the three-dimensional space of the first principal components. It allows us to make a hypothesis that the usage of triplets is symmetrical with respect to the operation of “complementary reversal”.

However, for a real genome of *Ercherichia coli*, we can observe, using the “metro map” representation, that the clusters are in fact rather well separated

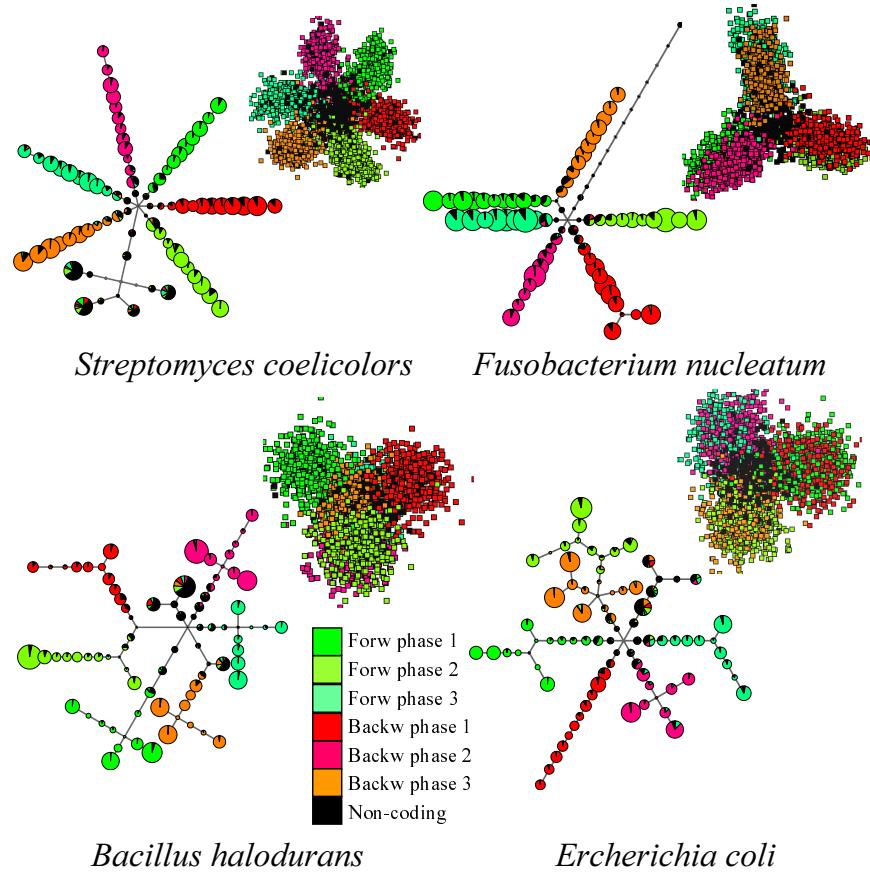


Fig. 9.7. Seven cluster structures presented for 4 selected genomes. A genome is represented as a collection of points (text fragments represented by their triplet frequencies) in a multidimensional space. Color codes correspond to 6 possible frameshifts when a random fragment overlaps with a gene (3 in the forward and 3 in the backward direction of the gene), and the black color corresponds to non-coding regions. For every genome a principal tree (“metro map” layout) is shown together with 2D PCA projection of the data distribution. Note that the clusters that are mixed in the PCA plot for *Escherichia coli* (they remain mixed in 3D PCA as well, see [32]) are well separated on the “metro map”.

in space. This signal is completely hidden in the PCA plot. This is even more interesting since we are comparing data approximation and visualization by a one-dimensional object (principal tree) with one made by a three-dimensional linear manifold (PCA).

9.5 Visualization of Microarray Data

9.5.1 Dataset Used

DNA microarray data is a rich source of information for molecular biology (for a recent overview, read [20]). This technology found numerous applications in understanding various biological processes including cancer. It allows screening of the expression of all genes simultaneously in a cell exposed to some specific conditions (for example, stress, cancer, normal conditions). Obtaining a sufficient number of observations (chips), one can construct a table of "samples vs genes", containing logarithms of the expression levels of typically several thousands (n) of genes in typically several tens (m) of samples.

We use data published in [27] containing gene expression values for 10401 genes in 103 samples of normal human tissues. The sample labels correspond to the tissue type from which the sample was taken. This dataset was proposed for analysis for the participants of the international workshop "Principal manifolds-2006" which took place in Leicester, UK, in August of 2006. It can be downloaded from the workshop web-page [25].

9.5.2 Principal Tree of Human Tissues

On Fig. 9.8 a metro map representation of the principal tree calculated for the human tissue data is shown. To reduce the computation time we first calculated a new spatial basis by calculating 103 linear principal components and projected samples from the full-dimensional space into this basis. The missing values in the dataset were treated as described in the accompanying paper [15] (a data point with missing value(s) is represented as a line or a (hyper)plane parallel to the corresponding coordinate axes, for which we have missing information, and then projected into the closest point on the linear manifold). The principal tree was then constructed using the *vdaengine* Java package available from the authors by request. We stopped construction of the optimal principal tree when 70 nodes were added to the tree.

One can see from the figure that most of the tissues are correctly clustered on the tree. Moreover, tissues of similar origin are grouped closely.

9.6 Discussion

In the continuum representation, factors are one-dimensional continua, hence, a product of r factors is represented as an r -dimensional *cubic complex* [23] that is glued together from r -dimensional parallelepipeds ("cubes"). Thus, the factorized principal elastic graphs generate a new and, as we can estimate now, useful construction: a principal cubic complex. One of the obvious benefits from this construction is adaptive dimension: the grammar approach with energy optimization develops the necessary number of non-trivial factors, and

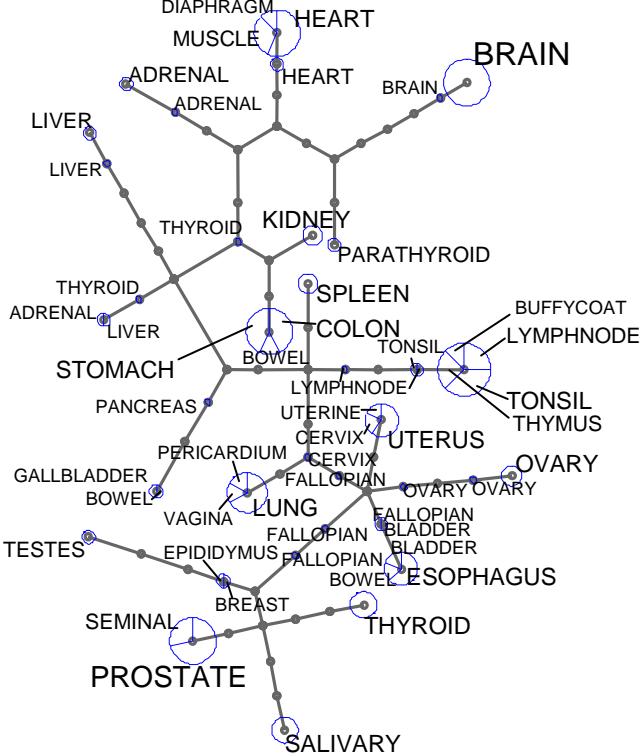


Fig. 9.8. Principal tree of human tissues, constructed from the gene expression microarray data. The size of the circles corresponds to the number of points projected into this node. The sectors show the proportion of different tissue types projected into a node.

not more. These complexes can approximate multidimensional datasets with complex, but still low-dimensional topology. The topology of the complex is not prescribed, but adaptive. In that sense, they are even more flexible than SOMs. The whole approach can be interpreted as an intermediate between absolutely flexible *neural gas* [22] and significantly more restrictive *elastic map* [14]. It includes as simple limit cases the k -means clustering algorithm (low elasticity moduli) and classical PCA (high μ for S_2 and $\mu \rightarrow \infty$ for S_k , $k > 2$).

We demonstrated how application of the simplest “add a node, bisect an edge” grammar leads to the construction of a useful “principal tree” object (more precisely, branching principal curve) which can be advantageous over the application of customary linear PCA. Of course, more work is required to evaluate all advantages and disadvantages which this approach gives in comparison with existing and widely used techniques (for example, with hierarchical clustering). However, it is clear that the principal tree approach

accompanied by the “metro map” representation of data, can provide additional insights into understanding the structure of complex data distributions and can be most suitable in some particular applications.

References

1. Bishop, C.M., Svensén, M., and Williams, C.K.I.: GTM: The generative topographic mapping. *Neural Computation* **10** (1), 215–234 (1998)
2. Crick, F.H.C., Barnett, L., Brenner, S., and Watts-Tobin, R.J.: General nature of the genetic code for proteins. *Nature*, **192**, 1227–1232 (1961)
3. Dergachev, V.A., Gorban, A.N., Rossiev, A.A., Karimova, L.M., Kuandykov, E.B., Makarenko, N.G., and Steier, P.: The filling of gaps in geophysical time series by artificial neural networks *Radiocarbon* **43**, 2A, 365–371 (2001)
4. Einbeck, J., Tutz, G., and Evers, L.: Local principal curves. *Statistics and Computing*, **15**, 301–313 (2005)
5. Erwin, E., Obermayer, K., and Schulten, K.: Self-organizing maps: ordering, convergence properties and energy functions. *Biological Cybernetics* **67**, 47–55 (1992)
6. Gorban, A.N., Popova, T.G., and Zinovyev, A.Yu.: Codon usage trajectories and 7-cluster structure of 143 complete bacterial genomic sequences. *Physica A: Statistical and Theoretical Physics*, **353**, 365–387 (2005)
7. Gorban, A.N., Zinovyev, A.Yu., and Popova, T.G.: Four basic symmetry types in the universal 7-cluster structure of 143 complete bacterial genomic sequences. *In Silico Biology* **5** 0025 (2005)
8. Gorban, A.N. and Rossiev, A.A.: Neural network iterative method of principal curves for data with gaps. *Journal of Computer and System Sciences International* **38** (5), 825–831 (1999)
9. Gorban, A.N., Sumner, N.R. and Zinovyev, A.Y.: Topological grammars for data approximation, *Applied Mathematics Letters* **20** (4), 382–386 (2005)
10. Gorban, A.N. and Zinovyev, A.Y.: Visualization of data by method of elastic maps and its applications in genomics, economics and sociology Preprint of Institut des Hautes Etudes Scientifiques, M/01/36 (2001)
<http://www.ihes.fr/PREPRINTS/M01/Resu/resu-M01-36.html>
11. Gorban, A.N. and Zinovyev, A.Y.: Method of elastic maps and its applications in data visualization and data modeling. *International Journal of Computing Anticipatory Systems, CHAOS*, **12**, 353–369 (2001)
12. Gorban, A.N., Zinovyev, A.Yu. and Wunsch, D.C.: Application of the method of elastic maps in analysis of genetic texts. In: *Proceedings of International Joint Conference on Neural Networks (IJCNN)*. Portland, Oregon (2003).
13. Gorban, A.N., Zinovyev, A.Yu. and Pitenco, A.A.: Visualization of data using method of elastic maps (in Russian). *Informationsnne technologii* **6**, 26–35 (2000)
14. Gorban, A. and Zinovyev, A.: Elastic Principal Graphs and Manifolds and their Practical Applications. *Computing* **75**, 359–379 (2005)
15. Gorban, A.N. and Zinovyev, A.Y.: Elastic maps and nets for approximating principal manifolds and their application to microarray data visualization. In this book.
16. Gusev, A.: Finite element mapping for spring network representations of the mechanics of solids. *Phys. Rev. Lett.* **93** (2), 034302 (2004)

17. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association* **84** (406) (1989), 502–516 (1989)
18. Kégl, B. and Krzyzak, A.: Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24** (1), 59–74 (2002)
19. Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics* **43**, 59–69 (1982)
20. Leung, Y.F. and Cavalieri, D.: Fundamentals of cDNA microarray data analysis. *Trends Genet.* **19** (11), 649–659 (2003)
21. Löwe, M.: Algebraic approach to single-pushout graph transformation. *Theor. Comp. Sci.* **109**, 181–224 (1993)
22. Martinetz, T.M., Berkovich, S.G., and Schulten K.J.: Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, **4** 4, 558–569 (1993)
23. Matveev, S. and Polyak, M.: Cubic complexes and finite type invariants. In: *Geometry & Topology Monographs*, Vol. 4: Invariants of knots and 3-manifolds. Kyoto, 215–233 (2001)
24. Mulier, F. and Cherkassky, V.: Self-organization as an iterative kernel smoothing process. *Neural Computation* **7**, 1165–1177 (1995)
25. “Principal manifolds for data cartography and dimension reduction”, Leicester, UK, August 2006. A web-page with test microarrays datasets provided for participants of the workshop: <http://www.ihes.fr/~zinovyev/princmanif2006>
26. Pearson, K.: On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, series 6 (2), 559–572 (1901)
27. Shyamsundar, R., Kim, Y.H., Higgins, J.P. et al.: A DNA microarray survey of gene expression in normal human tissues. *Genome Biology*, **6**, R22 (2005)
28. Nagl, M.: Formal languages of labelled graphs: Computing, **16**, 113–137 (1976)
29. Ritter, H., Martinetz, T. and Schulten, K.: *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley Reading, Massachusetts (1992)
30. Zinovyev, A.: *Visualization of Multidimensional Data*. Krasnoyarsk State University Press Publ. (2000)
31. Zinovyev, A.Yu., Gorban, A.N. and Popova, T.G.: Self-organizing approach for automated gene identification. *Open Systems and Information Dynamics* **10** (4), 321–333 (2003)
32. Cluster structures in genomic word frequency distributions. Web-site with supplementary materials. <http://www.ihes.fr/~zinovyev/7clusters/index.htm>

Diffusion Maps - a Probabilistic Interpretation for Spectral Embedding and Clustering Algorithms

Boaz Nadler¹, Stephane Lafon^{2,3}, Ronald Coifman³, and
Ioannis G. Kevrekidis⁴

¹ Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, 76100, Israel,
boaz.nadler@weizmann.ac.il

² Google, Inc.

³ Department of Mathematics, Yale University, New Haven, CT, 06520-8283, USA,
coifman@math.yale.edu

⁴ Department of Chemical Engineering and Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544, USA,
yannis@princeton.edu

Summary. Spectral embedding and spectral clustering are common methods for non-linear dimensionality reduction and clustering of complex high dimensional datasets. In this paper we provide a diffusion based probabilistic analysis of algorithms that use the normalized graph Laplacian. Given the pairwise adjacency matrix of all points in a dataset, we define a random walk on the graph of points and a *diffusion distance* between any two points. We show that the diffusion distance is equal to the Euclidean distance in the embedded space with all eigenvectors of the normalized graph Laplacian. This identity shows that *characteristic relaxation times and processes* of the random walk on the graph are the key concept that governs the properties of these spectral clustering and spectral embedding algorithms. Specifically, for spectral clustering to succeed, a necessary condition is that the mean exit times from each cluster need to be significantly larger than the largest (slowest) of all relaxation times inside all of the individual clusters. For complex, multiscale data, this condition may not hold and multiscale methods need to be developed to handle such situations.

10.1 Introduction

Clustering and low dimensional representation of high dimensional data sets are important problems in many diverse fields. In recent years various spectral methods to perform these tasks, based on the eigenvectors of adjacency matrices of graphs on the data have been developed, see for example [1]-[12] and references therein. In the simplest version, known as the normalized graph

Laplacian, given n data points $\{\mathbf{x}_i\}_{i=1}^n$ where each $\mathbf{x}_i \in \mathbb{R}^p$ (or some other normed vector space), we define a pairwise similarity matrix between points, for example using a Gaussian kernel with width σ^2 ,

$$W_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right), \quad (10.1)$$

and a diagonal normalization matrix $D_{ii} = \sum_j W_{ij}$. Many works propose to use the first few eigenvectors of the normalized eigenvalue problem $W\phi = \lambda D\phi$, or equivalently of the matrix

$$M = D^{-1}W, \quad (10.2)$$

either as a basis for the low dimensional representation of data or as good coordinates for clustering purposes. Although eq. (1) is based on a Gaussian kernel, other kernels are possible, and for actual datasets the choice of a kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ can be crucial to the method's success.

The use of the first few eigenvectors of M as good coordinates is typically justified with heuristic arguments or as a relaxation of a discrete clustering problem [3]. In [6, 7] Belkin and Niyogi showed that, when data is uniformly sampled from a low dimensional manifold of \mathbb{R}^p , the first few eigenvectors of M are discrete approximations of the eigenfunctions of the Laplace-Beltrami operator on the manifold, thus providing a mathematical justification for their use in this case. We remark that a compact embedding of a manifold into a Hilbert space via the eigenfunctions of the Laplace-Beltrami operator was suggested in differential geometry, and used to define distances between manifolds [13]. A different theoretical analysis of the eigenvectors of the matrix M , based on the fact that M is a stochastic matrix representing a random walk on the graph was described by Meilă and Shi [14], who considered the case of piecewise constant eigenvectors for specific lumpable matrix structures. Additional notable works that considered the random walk aspects of spectral clustering are [10, 15], where the authors suggest clustering based on the average commute time between points, [16, 17] which considered the relaxation process of this random walk, and [18, 19] which suggested random walk based agglomerative clustering algorithms.

In this paper we present a unified probabilistic framework for the analysis of spectral clustering and spectral embedding algorithms based on the normalized graph Laplacian. First, in section 10.2 we define a distance function between any two points based on the random walk on the graph, which we naturally denote the *diffusion distance*. The diffusion distance depends on a time parameter t , whereby different structures of the graph are revealed at different times. We then show that the non-linear embedding of the nodes of the graph onto the eigenvector coordinates of the normalized graph Laplacian, which we denote as the *diffusion map*, converts the diffusion distance between the nodes into Euclidean distance in the embedded space. This identity provides a probabilistic interpretation for such non-linear embedding algorithms.

It also provides the key concept that governs the properties of these methods, the characteristic relaxation times and processes of the random walk on a graph. Properties of spectral embedding and spectral clustering algorithms in light of these characteristic relaxation times are discussed in sections 10.3 and 10.4. We conclude with summary and discussion in section 10.5. The main results of this paper were first presented in [20] and [24].

10.2 Diffusion Distances and Diffusion Maps

The starting point of our analysis, as also noted in other works, is the observation that the matrix M is adjoint to a symmetric matrix

$$M_s = D^{1/2}MD^{-1/2}. \quad (10.3)$$

Thus, the two matrices M and M_s share the same eigenvalues. Moreover, since M_s is symmetric it is diagonalizable and has a set of n real eigenvalues $\{\lambda_j\}_{j=0}^{n-1}$ whose corresponding eigenvectors $\{\mathbf{v}_j\}$ form an orthonormal basis of \mathbb{R}^n . We sort the eigenvalues in decreasing order in absolute value, $|\lambda_0| \geq |\lambda_1| \geq \dots \geq |\lambda_{n-1}|$. The left and right eigenvectors of M , denoted ϕ_j and ψ_j are related to those of M_s according to

$$\phi_j = \mathbf{v}_j D^{1/2}, \quad \psi_j = \mathbf{v}_j D^{-1/2}. \quad (10.4)$$

Since the eigenvectors \mathbf{v}_j are orthonormal under the standard dot product in \mathbb{R}^n , it follows that the vectors ϕ_i and ψ_j are bi-orthonormal

$$\langle \phi_i, \psi_j \rangle = \delta_{ij}, \quad (10.5)$$

where $\langle \mathbf{u}, \mathbf{v} \rangle$ is the standard dot product between two vectors in \mathbb{R}^n . We now utilize the fact that by construction M is a stochastic matrix with all row sums equal to one, and can thus be interpreted as defining a random walk on the graph. Under this view, M_{ij} denotes the transition probability from the point \mathbf{x}_i to the point \mathbf{x}_j in one time step,

$$\Pr\{\mathbf{x}(t+1) = \mathbf{x}_j | \mathbf{x}(t) = \mathbf{x}_i\} = M_{ij}. \quad (10.6)$$

We denote by $p(t, \mathbf{y} | \mathbf{x})$ the probability distribution of a random walk landing at location \mathbf{y} at time t , given a starting location \mathbf{x} at time $t = 0$. In terms of the matrix M , this transition probability is given by $p(t, \mathbf{y} | \mathbf{x}_i) = \mathbf{e}_i M^t$, where \mathbf{e}_i is a row vector of zeros with a single entry equal to one at the i -th coordinate.

For ε large enough all points in the graph are connected, so that M is an irreducible and aperiodic Markov chain. It has a unique eigenvalue equal to 1, with the other eigenvalues strictly smaller than one in absolute value. Then, regardless of the initial starting point \mathbf{x} ,

$$\lim_{t \rightarrow \infty} p(t, \mathbf{y} | \mathbf{x}) = \phi_0(\mathbf{y}), \quad (10.7)$$

where ϕ_0 is the left eigenvector of M with eigenvalue $\lambda_0 = 1$, explicitly given by

$$\phi_0(\mathbf{x}_i) = \frac{D_{ii}}{\sum_j D_{jj}}. \quad (10.8)$$

This eigenvector has a dual interpretation. The first is that ϕ_0 is the stationary probability distribution on the graph, while the second is that $\phi_0(\mathbf{x})$ is a density estimate at the point \mathbf{x} . Note that for a general shift invariant kernel $K(\mathbf{x} - \mathbf{y})$ and for the Gaussian kernel in particular, ϕ_0 is simply the well known Parzen window density estimator [21].

For any finite time t , we decompose the probability distribution in the eigenbasis $\{\phi_j\}$

$$p(t, \mathbf{y} | \mathbf{x}) = \phi_0(\mathbf{y}) + \sum_{j \geq 1} a_j(\mathbf{x}) \lambda_j^t \phi_j(\mathbf{y}), \quad (10.9)$$

where the coefficients a_j depend on the initial location \mathbf{x} . The bi-orthonormality condition (10.5) gives $a_j(\mathbf{x}) = \psi_j(\mathbf{x})$, with $a_0(\mathbf{x}) = \psi_0(\mathbf{x}) = 1$ already implicit in (10.9).

Given the definition of the random walk on the graph, it is only natural to quantify the similarity between any two points according to the evolution of probability distributions initialized as delta functions on these points. Specifically, we consider the following distance measure at time t ,

$$\begin{aligned} D_t^2(\mathbf{x}_0, \mathbf{x}_1) &= \|p(t, \mathbf{y} | \mathbf{x}_0) - p(t, \mathbf{y} | \mathbf{x}_1)\|_w^2 \\ &= \sum_{\mathbf{y}} (p(t, \mathbf{y} | \mathbf{x}_0) - p(t, \mathbf{y} | \mathbf{x}_1))^2 w(\mathbf{y}) \end{aligned} \quad (10.10)$$

with the specific choice $w(\mathbf{y}) = 1/\phi_0(\mathbf{y})$ for the weight function, which takes into account the (empirical) local density of the points, and puts more weight on low density points.

Since this distance depends on the random walk on the graph, we quite naturally denote it as the *diffusion distance* at time t . We also denote the mapping between the original space and the first k eigenvectors as the *diffusion map* at time t

$$\Psi_t(\mathbf{x}) = (\lambda_1^t \psi_1(\mathbf{x}), \lambda_2^t \psi_2(\mathbf{x}), \dots, \lambda_k^t \psi_k(\mathbf{x})). \quad (10.11)$$

The following theorem relates the diffusion distance and the diffusion map.

Theorem: *The diffusion distance (10.10) is equal to Euclidean distance in the diffusion map space with all $(n - 1)$ eigenvectors.*

$$D_t^2(\mathbf{x}_0, \mathbf{x}_1) = \sum_{j \geq 1} \lambda_j^{2t} (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2 = \|\Psi_t(\mathbf{x}_0) - \Psi_t(\mathbf{x}_1)\|^2. \quad (10.12)$$

Proof: Combining (10.9) and (10.10) gives

$$D_t^2(\mathbf{x}_0, \mathbf{x}_1) = \sum_{\mathbf{y}} \left(\sum_j \lambda_j^t (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1)) \phi_j(\mathbf{y}) \right)^2 / \phi_0(\mathbf{y}). \quad (10.13)$$

Expanding the brackets and changing the order of summation gives

$$\begin{aligned} D_t^2(\mathbf{x}_0, \mathbf{x}_1) \\ = \sum_{j,k} \lambda_j^t (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1)) \lambda_k^t (\psi_k(\mathbf{x}_0) - \psi_k(\mathbf{x}_1)) \sum_{\mathbf{y}} \frac{\phi_j(\mathbf{y}) \phi_k(\mathbf{y})}{\phi_0(\mathbf{y})}. \end{aligned}$$

From relation (10.4) it follows that $\phi_k/\phi_0 = \psi_k$. Moreover, according to (10.5) the vectors ϕ_j and ψ_k are bi-orthonormal. Therefore, the inner summation over \mathbf{y} gives δ_{jk} , and overall the required formula (10.12). Note that in (10.12) summation starts from $j \geq 1$ since $\psi_0(\mathbf{x}) = 1$. \square

This theorem provides a probabilistic interpretation to the non-linear embedding of points \mathbf{x}_i from the original space (say \mathbb{R}^p) to the diffusion map space \mathbb{R}^{n-1} . Therefore, geometry in diffusion space is meaningful, and can be interpreted in terms of the Markov chain. The advantage of this distance measure over the standard distance between points in the original space is clear. While the original distance between any pair of points is independent of the location of all other points in the dataset, the diffusion distance between a pair of points depends on all possible paths connecting them, including those that pass through other points in the dataset. The diffusion distance thus measures the dynamical proximity between points on the graph, according to their connectivity.

Both the diffusion distance and the diffusion map depend on the time parameter t . For very short times, all points in the diffusion map space are far apart, whereas as time increases to infinity, all pairwise distances converge to zero, since $p(t, \mathbf{y}|\mathbf{x})$ converges to the stationary distribution. It is in the intermediate regime, where at different times different structures of the graph are revealed [11].

The identity (10.12) shows that the eigenvalues and eigenvectors $\{\lambda_j, \psi_j\}_{j \geq 1}$ capture the characteristic relaxation times and processes of the random walk on the graph. On a connected graph with n points, there are $n - 1$ possible time scales. However, most of them capture fine detail structure and only the first few largest eigenvalues capture the coarse global structures of the graph. In cases where the matrix M has a *spectral gap* with only a few eigenvalues close to one and all remaining eigenvalues much smaller than one, the diffusion distance at a large enough time t can be well approximated by only the first few k eigenvectors $\psi_1(\mathbf{x}), \dots, \psi_k(\mathbf{x})$, with a negligible error. Furthermore, as shown in [22], quantizing this diffusion space is equivalent to lumping the random walk, retaining only its slowest relaxation processes. The following lemma bounds the error of a k -term approximation of the diffusion distance.

Lemma: For all times $t \geq 0$, the error in a k -term approximation of the diffusion distance is bounded by

$$|D_t^2(\mathbf{x}_0, \mathbf{x}_1) - \sum_{j=1}^k \lambda_j^{2t} (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2| \leq \lambda_{k+1}^{2t} \left(\frac{1}{\phi_0(\mathbf{x}_0)} + \frac{1}{\phi_0(\mathbf{x}_1)} \right). \quad (10.14)$$

Proof: From the spectral decomposition (10.12)

$$\begin{aligned} |D_t^2(\mathbf{x}_0, \mathbf{x}_1) - \sum_{j=1}^k \lambda_j^{2t} (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2| &= \sum_{j=k+1}^{n-1} \lambda_j^{2t} (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2 \\ &\leq \lambda_{k+1}^{2t} \sum_{j=0}^{n-1} (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2. \end{aligned} \quad (10.15)$$

In addition, at time $t = 0$, we get that

$$D_0^2(\mathbf{x}_0, \mathbf{x}_1) = \sum_{j=0}^{n-1} (\psi_j(\mathbf{x}_0) - \psi_j(\mathbf{x}_1))^2.$$

However, from the definition of the diffusion distance (10.10), we have that at time $t = 0$

$$D_0^2(\mathbf{x}_0, \mathbf{x}_1) = \|p(0, \mathbf{y}|\mathbf{x}_0) - p(0, \mathbf{y}|\mathbf{x}_1)\|_w^2 = \left(\frac{1}{\phi_0(\mathbf{x}_0)} + \frac{1}{\phi_0(\mathbf{x}_1)} \right).$$

Combining the last three equations proves the lemma. \square .

Remark: This lemma shows that the error in computing an approximate diffusion distance with only k eigenvectors decays exponentially fast as a function of time. As the number of points $n \rightarrow \infty$, Eq. (10.14) is not informative since the steady state probabilities of individual points decay to zero at least as fast as $1/n$. However, for a very large number of points it makes more sense to consider the diffusion distance between regions of space instead of between individual points. Let Ω_1, Ω_2 be two such subsets of points. We then define

$$D_t^2(\Omega_1, \Omega_2) = \sum_{\mathbf{x}} \frac{(p(\mathbf{x}, t|\Omega_1) - p(\mathbf{x}, t|\Omega_2))^2}{\phi_0(\mathbf{x})}, \quad (10.16)$$

where $p(\mathbf{x}, t|\Omega_1)$ is the transition probability at time t , starting from the region Ω_1 . As initial conditions inside Ω_i , we choose the steady state distribution, conditional on the random walk starting inside this region,

$$p(\mathbf{x}, 0|\Omega_i) = p_i(\mathbf{x}) = \begin{cases} \frac{\phi_0(\mathbf{x})}{\phi_0(\Omega_i)}, & \text{if } \mathbf{x} \in \Omega_i; \\ 0, & \text{if } \mathbf{x} \notin \Omega_i, \end{cases} \quad (10.17)$$

where

$$\phi_0(\Omega_i) = \sum_{\mathbf{y} \in \Omega_i} \phi_0(\mathbf{y}) . \quad (10.18)$$

Eq. (10.16) can then be written as

$$D_t^2(\Omega_1, \Omega_2) = \sum_{\mathbf{x} \in \Omega_1, \mathbf{y} \in \Omega_2} D_t^2(\mathbf{x}, \mathbf{y}) p_1(\mathbf{x}) p_2(\mathbf{y}) . \quad (10.19)$$

Similar to the proof of the lemma, it follows that

$$\left| D_t^2(\Omega_1, \Omega_2) - \sum_{\mathbf{x}, \mathbf{y}} D_{t,k}^2(\mathbf{x}, \mathbf{y}) p_1(\mathbf{x}) p_2(\mathbf{y}) \right| \leq \lambda_{k+1}^{2t} \left[\frac{1}{\phi_0(\Omega_1)} + \frac{1}{\phi_0(\Omega_2)} \right] , \quad (10.20)$$

where $D_{t,k}(\mathbf{x}, \mathbf{y})$ is the approximation to the diffusion distance at time t by the first k eigenvectors.

Therefore, if we take regions Ω_i with non negligible steady state probabilities that are bounded from below by some constant, $\phi_0(\Omega_i) > \alpha$, for times $t \gg |\log(\lambda_{k+1})/\log(\alpha)|$, the approximation error of the k -term expansion is negligible. This observation provides a probabilistic interpretation as to what information is lost and retained in dimensional reduction with these eigenvectors.

In addition, the following theorem shows that this k -dimensional approximation is *optimal* under a certain mean squared error criterion.

Theorem: *Out of all k -dimensional approximations of the form*

$$\hat{p}_k(t, \mathbf{y}|\mathbf{x}) = \phi_0(\mathbf{y}) + \sum_{j=1}^k a_j(t, \mathbf{x}) \mathbf{w}_j(\mathbf{y})$$

for the probability distribution at time t , the one that minimizes the mean squared error

$$\mathbb{E}_{\mathbf{x}} \{ \|p(t, \mathbf{y}|\mathbf{x}) - \hat{p}_k(t, \mathbf{y}|\mathbf{x})\|_w^2 \} ,$$

where averaging over initial points \mathbf{x} is with respect to the stationary density $\phi_0(\mathbf{x})$, is given by $\mathbf{w}_j(\mathbf{y}) = \phi_j(\mathbf{y})$ and $a_j(t, \mathbf{x}) = \lambda_j^t \psi_j(\mathbf{x})$. Therefore, the optimal k -dimensional approximation is given by the truncated sum

$$\hat{p}_k(\mathbf{y}, t|\mathbf{x}) = \phi_0(\mathbf{y}) + \sum_{j=1}^k \lambda_j^t \psi_j(\mathbf{x}) \phi_j(\mathbf{y}) . \quad (10.21)$$

Proof: The proof is a consequence of weighted principal component analysis applied to the matrix M , taking into account the bi-orthogonality of the left and right eigenvectors.

We note that the first few eigenvectors are also optimal under other criteria, for example for data sampled from a manifold as in [6], or for multiclass spectral clustering [23].

10.2.1 Asymptotics of the Diffusion Map

Further insight into the properties of spectral clustering can be gained by considering the limit as the number of samples converges to infinity, and as the width of the kernel approaches zero. This has been the subject of intensive research over the past few years by various authors [6, 26, 29, 11, 24, 25, 27, 28]. Here we present the main results without detailed mathematical proofs and refer the reader to the above works.

The starting point for the analysis of this limit is the introduction of a statistical model in which the data points $\{\mathbf{x}_i\}$ are i.i.d. random samples from a smooth probability density $p(\mathbf{x})$ confined to a compact connected subset $\Omega \subset \mathbb{R}^p$ with smooth boundary $\partial\Omega$. Following the statistical physics notation, we write the density in Boltzmann form, $p(\mathbf{x}) = e^{-U(\mathbf{x})}$, where $U(\mathbf{x})$ is the (dimensionless) potential or energy of the configuration \mathbf{x} .

For each eigenvector \mathbf{v}_j of the discrete matrix M with corresponding eigenvalue $\lambda_j \neq 0$, we extend its definition to any $\mathbf{x} \in \Omega$ as follows

$$\psi_j^{(n)}(\mathbf{x}) = \frac{1}{\lambda_j} \sum_i \frac{k(\mathbf{x}, \mathbf{x}_i)}{D(\mathbf{x})} \mathbf{v}_j(\mathbf{x}_i) \quad (10.22)$$

with $D(\mathbf{x}) = \sum_i k(\mathbf{x}, \mathbf{x}_i)$. Note that this definition retains the values at the sampled points, e.g., $\psi_j^{(n)}(\mathbf{x}_i) = v_j(\mathbf{x}_i)$ for all $i = 1, \dots, n$.

As shown in [24], in the limit $n \rightarrow \infty$ the random walk on the discrete graph converges to a random walk on the continuous space Ω . Then, it is possible to define an integral operator T as follows,

$$T[\psi](\mathbf{x}) = \int_{\Omega} M(\mathbf{y}|\mathbf{x})\psi(\mathbf{y})p(\mathbf{y}) d\mathbf{y} ,$$

where $M(\mathbf{x}|\mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\sigma^2)/D_{\sigma}(\mathbf{y})$ is the transition probability from \mathbf{y} to \mathbf{x} in time ε , and $D_{\sigma}(\mathbf{y}) = \int_{\Omega} \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\sigma^2)p(\mathbf{x}) d\mathbf{x}$. In the limit $n \rightarrow \infty$, the eigenvalues λ_j and the extensions $\psi_j^{(n)}$ of the discrete eigenvectors v_j converge to the eigenvalues and eigenfunctions of the integral operator T .

Further, in the limit $\sigma \rightarrow 0$, the random walk on the space Ω , upon scaling of time, converges to a diffusion process in a potential $2U(\mathbf{x})$,

$$\dot{\mathbf{x}}(t) = -\nabla(2U) + \sqrt{2}\dot{\mathbf{w}}(t) , \quad (10.23)$$

where $U(\mathbf{x}) = -\log(p(\mathbf{x}))$ and $\mathbf{w}(t)$ is standard Brownian motion in p dimensions. In this limit, the eigenfunctions of the integral operator T converge to those of the infinitesimal generator of this diffusion process, given by the following Fokker-Planck (FP) operator,

$$\mathcal{H}\psi = \Delta\psi - 2\nabla\psi \cdot \nabla U . \quad (10.24)$$

Table 10.1. Random Walks and Diffusion Processes

Case	Operator	Stochastic Process
$\sigma > 0$	finite $n \times n$ matrix M	R.W. discrete in space discrete in time
$n \rightarrow \infty$	integral operator T	R.W. in continuous space discrete in time
$\sigma \rightarrow 0$	infinitesimal generator \mathcal{H}	diffusion process continuous in time & space

The Langevin equation (10.23) is the standard model to describe stochastic dynamical systems in physics, chemistry and biology [30, 31]. As such, its characteristics as well as those of the corresponding FP equation have been extensively studied, see [30, 31, 32] and references therein. The term $\nabla\psi \cdot \nabla U$ in (10.24) is interpreted as a *drift* term towards low energy (high-density) regions, and plays a crucial part in the definition of clusters.

Note that when data is *uniformly sampled* from Ω , $\nabla U = 0$ so the drift term vanishes and we recover the Laplace-Beltrami operator on Ω .

Finally, when the density p has compact support on a domain Ω , the operator \mathcal{H} is defined only inside Ω . Its eigenvalues and eigenfunctions thus depend on the boundary conditions at $\partial\Omega$. As shown in [11], in the limit $\sigma \rightarrow 0$ the random walk satisfies reflecting boundary conditions on $\partial\Omega$, which translate into

$$\left. \frac{\partial\psi(\mathbf{x})}{\partial\mathbf{n}} \right|_{\partial\Omega} = 0 , \quad (10.25)$$

where \mathbf{n} is a unit normal vector at the point $\mathbf{x} \in \partial\Omega$.

To conclude, the right eigenvectors of the finite matrix M can be viewed as discrete approximations to those of the operator T , which in turn can be viewed as approximations to those of \mathcal{H} . Therefore, if there are enough data points for accurate statistical sampling, the structure and characteristics of the eigenvalues and eigenfunctions of \mathcal{H} are similar to the corresponding eigenvalues and discrete eigenvectors of M . In the next sections we show how this relation can be used to explain the characteristics of spectral clustering and dimensional reduction algorithms. The three different stochastic processes are summarized in table 1.

10.3 Spectral Embedding of Low Dimensional Manifolds

Let $\{\mathbf{x}_i\}$ denote points sampled (uniformly for simplicity) from a low dimensional manifold embedded in a high dimensional space. Eq. (10.14) shows that by retaining only the first k coordinates of the diffusion map, the reconstruction error of the long time random walk transition probabilities is

negligible. However, this is not necessarily the correct criterion for an embedding algorithm. Broadly speaking, assuming that data is indeed sampled from a manifold, a low dimensional embedding should preserve (e.g. uncover) the information about the global (coarse) structure of this manifold, while throwing out information about its fine details. A crucial question is then under what conditions does spectral embedding indeed satisfy these requirements, and perhaps more generally, what are its characteristics.

The manifold learning problem of a low dimensional embedding can be formulated as follows: Let $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^n \subset \mathbb{R}^q$ denote a set of points randomly sampled from some smooth probability density defined in a compact domain of \mathbb{R}^q (the coordinate space). However, we are given the set of points $\mathcal{X} = f(\mathcal{Y})$ where $f : \mathbb{R}^q \rightarrow \mathbb{R}^p$ is a smooth mapping with $p > q$. Therefore, assuming that the points \mathcal{Y} are not themselves on a lower dimensional manifold than \mathbb{R}^q , then the points \mathcal{X} lie on a manifold of dimension q in \mathbb{R}^p . Given $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the problem is to estimate the dimensionality q and the coordinate points $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$. Obviously, this problem is ill-posed and various degrees of freedom, such as translation, rotation, reflection and scaling cannot be determined.

While a general theory of manifold learning is not yet fully developed, in this section we would like to provide a glimpse into the properties of spectral embeddings, based on the probabilistic interpretation of section 10.2. We prove that in certain cases spectral embedding works, in the sense that it finds a reasonable embedding of the data, while in other cases modifications to the basic scheme are needed.

We start from the simplest example of a one dimensional curve embedded in a higher dimensional space. In this case, a successful low dimensional embedding should uncover the one-dimensionality of the data and give a representation of the arclength of the curve. We prove that spectral embedding succeeds in this task:

Theorem: *Consider data sampled uniformly from a non-intersecting smooth 1-D curve embedded in a high dimensional space. Then, in the limit of a large number of samples and small kernel width the first diffusion map coordinate gives a one-to-one parametrization of the curve. Further, in the case of a closed curve, the first two diffusion map coordinates map the curve into the circle.*

Proof: Let $\Gamma : [0, 1] \rightarrow \mathbb{R}^p$ denote a constant speed parametrization s of the curve ($\|d\Gamma(s)/ds\| = \text{const}$). As $n \rightarrow \infty, \varepsilon \rightarrow 0$, the diffusion map coordinates (eigenvectors of M) converge to the eigenfunctions of the corresponding FP operator. In the case of a non-intersecting 1-D curve, the Fokker-Planck operator is

$$\mathcal{H}\psi = \frac{d^2\psi}{ds^2}, \quad (10.26)$$

where s is an arc-length along Γ , with Neumann boundary conditions at the edges $s = 0, 1$. The first two non-trivial eigenfunctions are $\psi_1 = \cos(\pi s)$ and $\psi_2 = \cos(2\pi s)$. The first eigenfunction thus gives a one-to-one *parametrization* of the curve, and can thus be used to embed it into \mathbb{R}^1 . The second eigenfunction $\psi_2 = 2\psi_1^2 - 1$ is a quadratic function of the first. This relation (together with estimates on the local density of the points) can be used to verify that for a given dataset, at a coarse scale its data points indeed lie on a 1-D manifold.

Consider now a *closed* curve in \mathbb{R}^p . In this case there are no boundary conditions for the operator and we seek periodic eigenfunctions. The first two non-constant eigenfunctions are $\sin(\pi s + \theta)$ and $\cos(\pi s + \theta)$ where θ is an arbitrary rotation angle. These two eigenfunctions map data points on the curve to the circle in \mathbb{R}^2 , see [11]. \square

Example 1: Consider a set of 400 points in three-dimensions, sampled uniformly from a spiral curve. In figure 10.1 the points and the first two eigenvectors are plotted. As expected, the first eigenvector provides a parametrization of the curve, whereas the second one is a quadratic function of the first.

Example 2: The analysis above can also be applied to images. Consider a dataset of images of a single object taken from different horizontal rotation angles. These images, although residing in a high dimensional space, are all on a 1-d manifold defined by the rotation angle. The diffusion map can uncover this underlying one dimensional manifold on which the images reside and organize the images according to it. An example is shown in figure 10.2, where the first two diffusion map coordinates computed on a dataset of 37 images of a truck taken at uniform angles of $0, 5, \dots, 175, 180$ degrees are plotted one against the other. All computations were done using a Gaussian kernel with standard Euclidean distance between all images. The data is courtesy of Ronen Basri [33].

We remark that if data is sampled from a 1-D curve or more generally from a low dimensional manifold, but not in a uniform manner, the standard normalized graph Laplacian converges to the FP operator (10.24) which contains a drift term. Therefore its eigenfunctions depend both on the geometry of the manifold and on the probability density on it. However, replacing the isotropic kernel $\exp(-\|\mathbf{x} - \mathbf{y}\|^2/4\varepsilon)$ by the anisotropic one $\exp(-\|\mathbf{x} - \mathbf{y}\|/4\varepsilon)/D(\mathbf{x})D(\mathbf{y})$ asymptotically removes the effects of density and retains only those of geometry. With this kernel, the normalized graph Laplacian converges to the Laplace-Beltrami operator on the manifold [11].

We now consider the characteristics of spectral embedding on the “swiss roll” dataset, which has been used as a synthetic benchmark in many papers, see [7, 34] and refs. therein. The swiss roll is a 2-D manifold embedded in \mathbb{R}^3 . A set of n points $\mathbf{x}_i \in \mathbb{R}^3$ are generated according to $\mathbf{x} = (t \cos(t), h, t \sin(t))$, where $t \sim U[3\pi/2, 9\pi/2]$, and $h \sim U[0, H]$. By unfolding the roll, we obtain a rectangle of length L and width H , where in our example,

$$L = \int_{3\pi/2}^{9\pi/2} \sqrt{\left(\frac{d}{dt} t \sin t\right)^2 + \left(\frac{d}{dt} t \cos t\right)^2} dt \approx 90 .$$

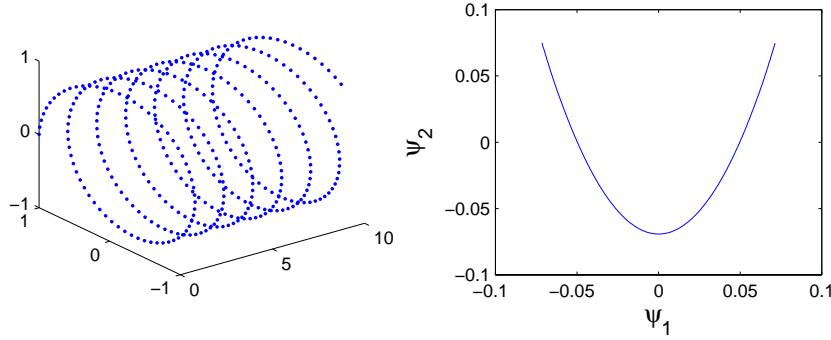


Fig. 10.1. 400 points uniformly sampled from a spiral in 3-D (left). First two non-trivial eigenfunctions. The first eigenfunction ψ_1 provides a parametrization of the curve. The second one is a quadratic function of the first.

For points uniformly distributed on this manifold, in the limit $n \rightarrow \infty, \varepsilon \rightarrow 0$, the FP operator is

$$\mathcal{H}\psi = \frac{d^2\psi}{dt^2} + \frac{d^2\psi}{dh^2}$$

with Neumann boundary conditions at the boundaries of the rectangle. Its eigenvalues and eigenfunctions are

$$\begin{aligned} \mu_{j,k} &= \pi^2 \left(\frac{j^2}{L^2} + \frac{k^2}{H^2} \right), \quad j, k \geq 0 \\ \psi(t, h) &= \cos\left(\frac{j\pi t}{L}\right) \cos\left(\frac{k\pi h}{H}\right). \end{aligned} \quad (10.27)$$

First we consider a reasonably wide swiss roll, with $H = 50$. In this case, the length and width of the roll are similar and so upon ordering the eigenvalues $\mu_{j,k}$ in increasing order, the first two eigenfunctions after the constant one are $\cos(\pi t/L)$ and $\cos(\pi h/H)$. In this case spectral embedding via the first two diffusion map coordinates gives a reasonably nice parametrization of the manifold, uncovering its 2-d nature, see fig. 10.3.

However, consider now the same swiss roll but with a slightly smaller width $H = 30$. Now the roll is roughly three times as long as it is wide. In this case, the first eigenfunction $\cos(\pi t/L)$ gives a one-to-one parametrization of the parameter t . However, the next two eigenfunctions, $\cos(2\pi t/L)$ and $\cos(3\pi t/L)$, are functions of ψ_1 , and thus provide no further useful information for the low dimensional representation of the manifold. It is only the 4th eigenfunction that reveals its two dimensional nature, see fig. 10.4. We remark that in both figures we do not obtain perfect rectangles in the embedded space. This is due to the non-uniform density of points on the manifold, with points more densely sampled in the inward spiral than in the outward one.

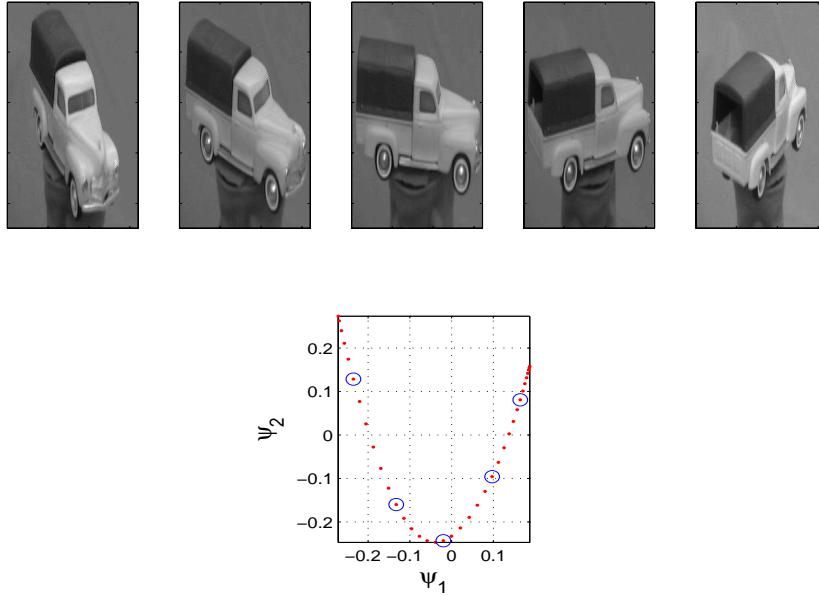


Fig. 10.2. Figures of a truck taken at five different horizontal angles (top). The mapping of the 37 images into the first two eigenvectors, based on a Gaussian kernel with standard Euclidean distance between the images as the underlying metric (bottom). The blue circles correspond to the five specific images shown above.

This example shows a fundamental difference between (linear) low dimensional embedding by principal component analysis, vs. nonlinear spectral methods. In PCA once the variance in a specific direction has been captured, all further projections are orthogonal to it. In non-linear spectral methods, the situation is fundamentally different. For example, even for points on a one dimensional (linear) line segment, there are N different eigenvectors that capture the various relaxation processes on it, all with non-zero eigenvalues. Therefore, several eigenvectors may encode for the same geometrical or spatial “direction” of a manifold. To obtain a sensible low dimensional representation, an analysis of the relations between the different eigenvectors is required to remove this redundancy.

10.4 Spectral Clustering of a Mixture of Gaussians

A second common application of spectral embedding methods is for the purpose of clustering. Given a set of n points $\{\mathbf{x}_i\}_{i=1}^n$ and a corresponding similarity matrix W_{ij} , many works suggest to use the first few coordinates of the normalized graph Laplacian as an embedding into a new space, where standard clustering algorithms such as k-means can be employed. Most methods

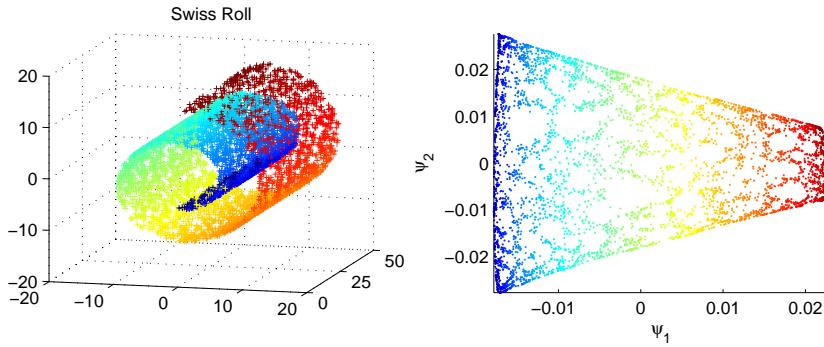


Fig. 10.3. 5000 points sampled from a wide swiss roll and embedding into the first two diffusion map coordinates.

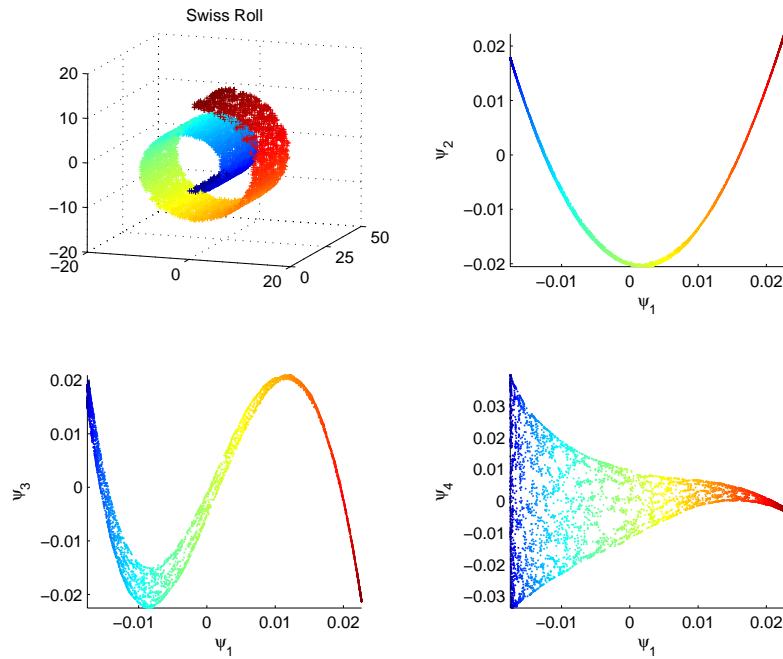


Fig. 10.4. 5000 points sampled from a narrow swiss roll and embedding into various diffusion map coordinates.

suggest to use the first $k - 1$ non-trivial eigenvectors after the constant one to find k clusters in a dataset. The various methods differ by the exact normalization of the matrix for which the eigenvectors are computed and the specific clustering algorithm applied after the embedding into the new space. Note

that if the original space had dimensionality $p < k$, then the embedding actually *increases* the dimension of the data for clustering purposes. An interesting question is then under what conditions are these spectral embedding followed by standard clustering methods expected to yield successful clustering results.

Two ingredients are needed to analyze this question. The first is a generative model for clustered data, and the second is an explicit definition of what is considered a good clustering result.

A standard generative model for data in general and for clustered data in particular is the *mixture of Gaussians* model. In this setting, data points $\{\mathbf{x}_i\}$ are i.i.d. samples from a density composed of a mixture of K Gaussians,

$$p(\mathbf{x}) = \sum_{i=1}^K w_i N(\boldsymbol{\mu}_i, \Sigma_i) \quad (10.28)$$

with means $\boldsymbol{\mu}_i$, covariance matrices Σ_i and respective weights w_i . We say that data from such a model is clusterable into K clusters if all the different Gaussian clouds are well separated from each other. This can be translated into the condition that

$$\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2 > 2 \min[\lambda_{\max}(\Sigma_i), \lambda_{\max}(\Sigma_j)] \quad \forall i, j, \quad (10.29)$$

where $\lambda_{\max}(\Sigma)$ is the largest eigenvalue of a covariance matrix Σ .

Let $\{\mathbf{x}_i\}$ denote a dataset from a mixture that satisfies these conditions, and let $S_1 \cup S_2 \cup \dots \cup S_K$ denote the partition of space into K disjoint regions, where each region S_j is defined to contain all points $\mathbf{x} \in \mathbb{R}^p$ whose probability to have been sampled from the j -th Gaussian is the largest. We consider the output of a clustering algorithm to be successful if its K regions have a high overlap to these optimal Bayes regions S_j .

We now analyze the performance of spectral clustering in this setting. We assume that we have a very large number of points and do not consider the important issue of finite sample size effects. Furthermore, we do not consider a specific spectral clustering algorithm, but rather give general statements regarding their possible success given the structure of the embedding coordinates.

In our analysis, we employ the intimate connection between the diffusion distance and the characteristic time scales and relaxation processes of the random walk on the graph of points, combined with matrix perturbation theory. A similar analysis can be made using the properties of the eigenvalues and eigenfunctions of the limiting FP operator.

Consider then n data points $\{\mathbf{x}_i\}_{i=1}^n$ sampled from a mixture of K reasonably separated Gaussians, and let $S_1 \cup S_2 \cup \dots \cup S_K$ denote a partition of space into K disjoint cluster regions as defined above. Then, by definition, each cluster region S_j contains the majority of points of each respective Gaussian. Consider the similarity matrix W computed on this discrete dataset, where we sort the points according to which cluster region they belong to. Since the

Gaussians are partially overlapping, the similarity matrix W does not have a perfect block structure (with the blocks being the sets S_j), but rather has small non zero weights between points of different cluster regions. To analyze the possible behavior of the eigenvalues and eigenvectors of such matrices, we introduce the following quantities. For each point $\mathbf{x}_i \in S_j$ we define

$$a_i = \sum_{\mathbf{x}_k \notin S_j} W_{ik} \quad (10.30)$$

and

$$b_i = \sum_{\mathbf{x}_k \in S_j} W_{ik} . \quad (10.31)$$

The quantity a_i measures the amount of connectivity of the point \mathbf{x}_i to points outside its cluster, whereas b_i measures the amount of connectivity to points in the same cluster. Further, we introduce a family of similarity matrices depending on a parameter ε , as follows:

$$W(\varepsilon) = (1 - \varepsilon) \text{diag} \left(\frac{a_i}{b_i} \right) W_0 + \varepsilon W_1 , \quad (10.32)$$

where

$$W_0(i, j) = \begin{cases} W_{ij} , & \text{if } \mathbf{x}_i, \mathbf{x}_j \in S_k , i \neq j ; \\ 0 , & \text{otherwise ,} \end{cases} \quad (10.33)$$

and

$$W_1(i, j) = \begin{cases} W_{ij} , & \text{if } \mathbf{x}_i \in S_\alpha , \mathbf{x}_j \in S_\beta , \alpha \neq \beta ; \\ 0 , & \text{otherwise .} \end{cases} \quad (10.34)$$

The matrix W_0 is therefore a block matrix with K blocks, which contains all intra-cluster connections, while the matrix W_1 contains all the inter-cluster connections. Note that in the representation (10.32), for each point \mathbf{x}_i , $D(\mathbf{x}_i) = \sum W_{ij}(\varepsilon)$ is independent of ε . Therefore, for the symmetric matrix $M_s(\varepsilon)$ similar to the Markov matrix, we can write

$$M_s(\varepsilon) = D^{1/2} W(\varepsilon) D^{1/2} = M_s(0) + \varepsilon M_1 . \quad (10.35)$$

When $\varepsilon = 0$, $W(\varepsilon) = W_0$ is a block matrix and so the matrix $M_s(0)$ corresponds to a *reducible* Markov chain with K components. When $\varepsilon = 1$ we obtain the original Markov matrix on the dataset, whose eigenvectors will be used to cluster the data. The parameter ε can thus be viewed as controlling the strength of the inter-cluster connections. Our aim is to relate the eigenvalues and eigenvectors of $M_s(0)$ to those of $M_s(1)$, while viewing the matrix εM_1 as a small perturbation.

Since $M_s(0)$ corresponds to a Markov chain with K disconnected components, the eigenvalue $\lambda = 1$ has multiplicity K . Further, we denote by $\lambda_1^R, \dots, \lambda_K^R$ the next largest eigenvalue in each of the K blocks. These eigenvalues correspond to the characteristic relaxation times in each of the K clusters, (denoted as spurious eigenvalues in [14]). As ε is increased from zero, the

eigenvalue $\lambda = 1$ with multiplicity K splits into K different branches. Since $M_s(\varepsilon)$ is a Markov matrix for all $0 \leq \varepsilon \leq 1$ and becomes connected for $\varepsilon > 0$, exactly one of the K eigenvalues stays fixed at $\lambda = 1$, whereas the remaining $K - 1$ decrease below one. These slightly smaller than one eigenvalues capture the mean exit times from the now weakly connected clusters.

According to Kato [35], [Theorem 6.1, page 120], the eigenvalues and eigenvectors of $M(\varepsilon)$ are *analytic* functions of ε on the real line. The point $\varepsilon = 0$, where $\lambda = 1$ has multiplicity $K > 1$ is called an *exceptional point*. Further, (see Kato [35], page 124) if we sort the eigenvalues in decreasing order, then the graph of each eigenvalue as a function of ε is a continuous function, which may cross other eigenvalues at various exceptional points ε_j . At each one of these values of ε , the graph of the eigenvalue as a function of ε jumps from one smooth curve to another. The corresponding eigenvectors, however, change abruptly at these crossing points as they move from one eigenvector to a different one.

We now relate these results to spectral clustering. A set of points is considered clusterable by these spectral methods if the corresponding perturbation matrix M_1 is small, that is, if there are no exceptional points or eigenvalue crossings for all values $\varepsilon \in (0, 1)$. This means that the fastest exit time from either one of the clusters is significantly slower than the slowest relaxation time in each one of the clusters. In this case, the first $K - 1$ eigenvectors of the Markov matrix M are approximately piecewise constant inside each of the K clusters. The next eigenvectors capture relaxation processes inside individual clusters and so each of them is approximately zero in all clusters but one. Due to their weighted bi-orthogonality of all eigenvectors (see section 10.2), clustering the points according to the sign structure of the first $K - 1$ eigenvectors approximately recovers the K clusters. This is the setting in which we expect spectral clustering algorithms to succeed.

However, now consider the case where relaxation times of some clusters are larger than the mean exit times from other clusters. Then there exists at least one exceptional point $\varepsilon < 1$, where a crossing of eigenvalues occurs. In this case, crucial information required for successful clustering is lost in the first $K - 1$ eigenvectors, since at least one of them now captures the relaxation process inside a large cluster. In this case, regardless of the specific clustering algorithm employed on these spectral embedding coordinates, it is not possible to distinguish one of the small clusters from others.

Example: We illustrate the results of this analysis on a simple example. Consider $n = 1000$ points generated from a mixture of three Gaussians in two dimensions. The centers of the Gaussians are

$$\mu_1 = (-6, 0), \mu_2 = (0, 0), \mu_3 = (x_R, 0),$$

where x_R is a parameter. The two rightmost Gaussians are spherical with standard deviation $\sigma_2 = \sigma_3 = 0.5$. The leftmost cluster has a diagonal covariance matrix

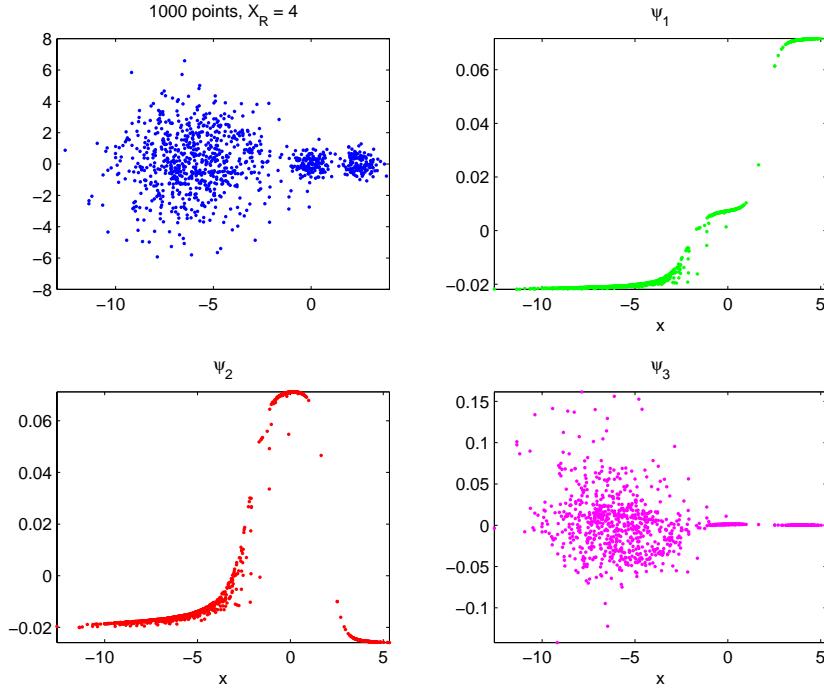


Fig. 10.5. Top left - 1000 points from three Gaussians. The three other panels show the first three non-trivial eigenvectors as a function of the x -coordinate.

$$\Sigma_1 = \begin{pmatrix} 2.0 & 0 \\ 0 & 2.4 \end{pmatrix}.$$

The weights of the three clusters are $(w_1, w_2, w_3) = (0.7, 0.15, 0.15)$. In figure 10.5 we present the dataset of 1000 points sampled from this mixture with $x_R = 4$, and the resulting first three non-trivial eigenvectors, ψ_1, ψ_2, ψ_3 as a function of the x -axis. All computations were done with a Gaussian kernel with width $\sigma = 1.0$. As seen in the figure, the three clusters are well separated and thus the first two non-trivial eigenvectors are piecewise constant in each cluster, while the third eigenvector captures the relaxation along the y -axis in the leftmost Gaussian and is thus not a function of the x -coordinate. We expect spectral clustering that uses only ψ_1 and ψ_2 to succeed in this case.

Now consider a very similar dataset, only that the center x_R of the rightmost cluster is slowly decreased from $x_R = 4$ towards $x = 0$. The dependence of the top six eigenvalues on x_R is shown in figure 10.6. As seen from the top panel, the first eigenvalue crossing occurs at the exceptional point $x_R = 2.65$, and then additional crossings occur at $x_R = 2.4, 2.3$ and at 2.15 .

Therefore, as long as $x_R > 2.65$ the mean exit time from the rightmost cluster is slower than the relaxation time in the large cluster, and spectral

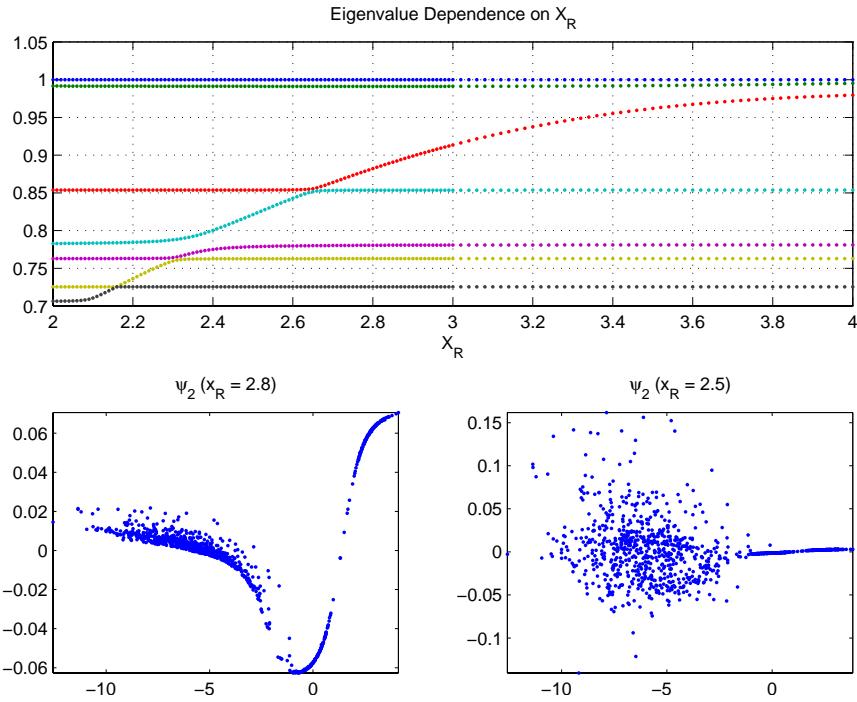


Fig. 10.6. Dependence of six largest eigenvalues on location of right cluster center (Top). The second largest non-trivial eigenvector as a function of the x -coordinate when $x_R = 2.8$ (Bottom left) and when $x_R = 2.5$ (Bottom right).

clustering using ψ_1, ψ_2 should be successful. However, for $x_R < 2.65$ the information distinguishing the two small clusters is not present any more in ψ_1, ψ_2 and thus spectral clustering will not be able to distinguish between these two clusters. An example of this sharp transition in the shape of the second eigenvector ψ_2 is shown in fig. 10.6 at the bottom left and right panels. For $x_R = 2.8 > 2.65$ the second eigenvector is approximately piecewise constant with two different constants in the two small clusters, whereas for $x_R = 2.5 < 2.65$ the second eigenvector captures the relaxation process in the large cluster and is approximately zero on both of the small ones. In this case ψ_3 captures the difference between these two smaller clusters.

As x_R is decreased further, additional eigenvalue crossings occur. In fig. 10.7 we show the first five non-trivial eigenvectors as a function of the x -coordinate for $x_R = 2.25$. Here, due to multiple eigenvalue crossings only ψ_5 is able to distinguish between the two rightmost Gaussians.

Our analysis shows that while spectral clustering may not work on multi-scale data, the comparison of relaxation times inside one set of points vs. the mean first passage time between two sets of points plays a natural role in the

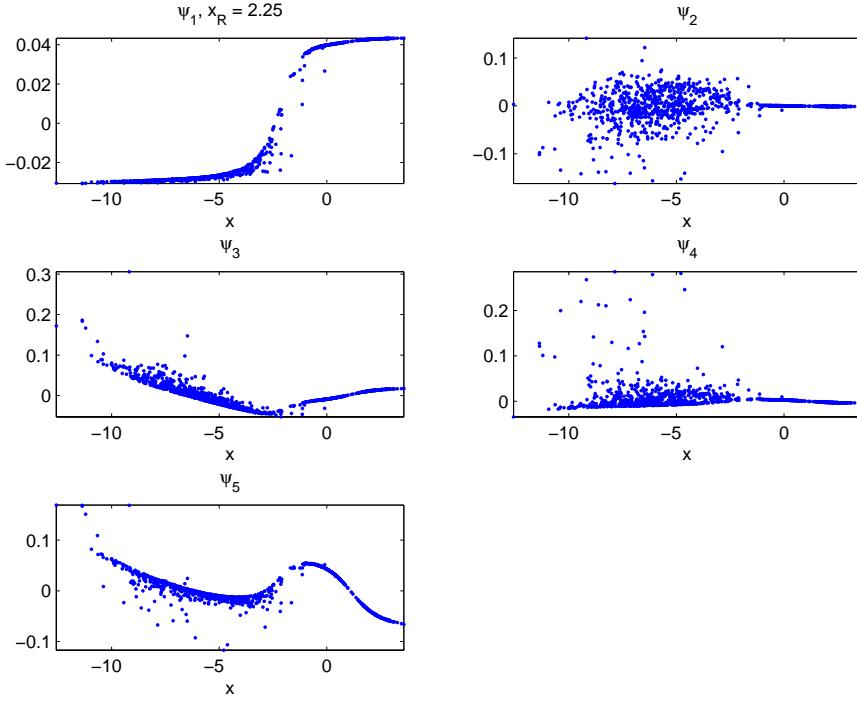


Fig. 10.7. The first five non-trivial eigenvectors as a function of the x -coordinate when the rightmost cluster is centered at $x_R = 2.25$.

definition of clusters. This leads to a multi-scale approach to clustering, based on a relaxation time coherence measure for the determination of the coherence of a group of points as all belonging to a single cluster, see [36]. Such an approach is able to successfully cluster this example even when $x_R = 2.25$, and has also been applied to image segmentation problems.

Finally, we would like to mention a simple analogy between spectral clustering where the goal is the uncovering of clusters, and the uncovering of signals in (linear) principal component analysis. Consider a setting where we are given n observations of the type “signals + noise”. A standard method to detect the signals is to compute the covariance matrix C of the observations and project the observations onto the first few leading eigenvectors of C . In this setting, if the signals lie in a low dimensional hyperspace of dimension k , and the noise has variance smaller than the smallest variance of the signals in this subspace, then PCA is successful at recovery of the signals. If, however, noise has variance larger than the smallest variance in this subspace, then at least one of the first k eigenvectors points in a direction orthogonal from this subspace, dictated by the direction with largest noise variance, and it is not possible to uncover all signals by PCA. Furthermore there is a sharp transi-

tion in the direction of this eigenvector, as noise strength is increased between being smaller than signal strength to larger than it [37]. As described above, in our case a similar sharp *phase transition phenomenon* occurs, only that the signal and the noise are replaced by other quantities: The “signals” are the mean exit times from the individual clusters, while the “noises” are the mean relaxation times inside them.

10.5 Summary and Discussion

In this paper we presented a probabilistic interpretation of spectral clustering and dimensionality reduction algorithms. We showed that the mapping of points from the feature space to the diffusion map space of eigenvectors of the normalized graph Laplacian has a well defined probabilistic meaning in terms of the diffusion distance. This distance, in turn, depends on both the geometry and density of the dataset. The key concepts in the analysis of these methods, that incorporates the density and geometry of a dataset, are the characteristic relaxation times and processes of the random walk on the graph. This provides novel insight into spectral clustering algorithms, and the starting point for the development of multiscale algorithms [36]. A similar analysis can also be applied to semi-supervised learning based on spectral methods [38]. Finally, these eigenvectors may be used to design better search and data collection protocols [39].

Acknowledgement. This work was partially supported by DARPA through AFOSR, and by the US department of Energy, CMPD (IGK). The research of BN is supported by the Israel Science Foundation (grant 432/06) and by the Hana and Julius Rosen fund.

References

1. Schölkopf, B. and Smola, A. J., and Müller, K.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, **10** (5), 1299–1319 (1998)
2. Weiss, Y.: Segmentation using eigenvectors: a unifying view. *ICCV* (1999)
3. Shi, J. and Malik, J.: Normalized cuts and image segmentation. *PAMI*, **22** (8), 888–905, (2000)
4. Ding, C., He, X., Zha, H., Gu, M., and Simon, H.: A min-max cut algorithm for graph partitioning and data clustering. In: Proc. IEEE International Conf. Data Mining, 107–114, (2001)
5. Cristianini, N., Shawe-Taylor, J., and Kandola, J.: Spectral kernel methods for clustering. *NIPS*, **14** (2002)
6. Belkin, M. and Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS*, **14** (2002)

7. Belkin, M. and Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* **15**, 1373–1396 (2003)
8. Ng, A.Y., Jordan, M., and Weiss, Y.: On spectral clustering, analysis and an algorithm. *NIPS*, **14** (2002)
9. Zhu, X., Ghahramani, Z., and Lafferty J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th international conference on machine learning (2003)
10. Saerens, M., Fouss, F., Yen L., and Dupont, P.: The principal component analysis of a graph and its relationships to spectral clustering. In: Proceedings of the 15th European Conference on Machine Learning, ECML, 371–383 (2004)
11. Coifman, R.R., Lafon, S.: Diffusion Maps. *Appl. Comp. Harm. Anal.*, **21**, 5–30 (2006)
12. Coifman, R.R., Lafon, S., Lee, A.B., Maggioni, M., Nadler, B., Warner, F., and Zucker S.: Geometric diffusion as a tool for harmonic analysis and structure definition of data, parts I and II. *Proc. Nat. Acad. Sci.*, **102** (21), 7426–7437 (2005)
13. Berard, P., Besson, G., Gallot, S.: Embedding Riemannian manifolds by their heat kernel. *Geometric and Functional Analysis*, **4** (1994)
14. Meila, M., Shi, J.: A random walks view of spectral segmentation. *AI and Statistics* (2001)
15. Yen, L., Vanvyve, D., Wouters, F., Fouss, F., Verleysen M., and Saerens, M.: Clustering using a random-walk based distance measure. In: Proceedings of the 13th Symposium on Artificial Neural Networks, ESANN, 317–324 (2005)
16. Tishby, N. and Slonim, N.: Data Clustering by Markovian Relaxation and the information bottleneck method. *NIPS* (2000)
17. Chennubhotla, C. and Jepson, A.J.: Half-lives of eigenflows for spectral clustering. *NIPS* (2002)
18. Harel, D. and Koren, Y.: Clustering spatial data using random walks. In: Proceedings of the 7th ACM Int. Conference on Knowledge Discovery and Data Mining, 281–286. ACM Press (2001)
19. Pons, P. and Latapy, M.: Computing Communities in Large Networks Using Random Walks. In: 20th International Symposium on Computer and Information Sciences (ISCIS'05). LNCS 3733 (2005)
20. Nadler, B., Lafon, S., Coifman, R.R., and Kevrekidis, I.G.: Diffusion maps spectral clustering and eigenfunctions of Fokker-Planck operators. *NIPS* (2005)
21. Parzen, E.: On estimation of a probability density function and mode. *Ann. Math. Stat.* **33**, 1065–1076 (1962)
22. Lafon, S. and Lee, A.B.: Diffusion maps: A unified framework for dimension reduction, data partitioning and graph subsampling. *IEEE Trans. Patt. Anal. Mach. Int.*, **28** (9), 1393–1403 (2006)
23. Yu, S. and Shi, J.: Multiclass spectral clustering. *ICCV* (2003)
24. Nadler, B., Lafon, S., Coifman, R.R., and Kevrekidis, I.G.: Diffusion maps, spectral clustering, and the reaction coordinates of dynamical systems. *Appl. Comp. Harm. Anal.*, **21**, 113–127 (2006)
25. von Luxburg, U., Bousquet, O., and Belkin, M.: On the convergence of spectral clustering on random samples: the normalized case. *NIPS* (2004)
26. Belkin, M. and Niyogi, P.: Towards a theoretical foundation for Laplacian-based manifold methods. *COLT* (2005)
27. Hein, M., Audibert, J., and von Luxburg, U.: From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. *COLT* (2005)

28. Singer, A.: From graph to manifold Laplacian: the convergence rate. *Applied and Computational Harmonic Analysis*, **21** (1), 135–144 (2006)
29. Belkin, M. and Niyogi, P.: Convergence of Laplacian eigenmaps. *NIPS* (2006)
30. Gardiner, C.W.: *Handbook of Stochastic Methods*, 3rd edition. Springer, NY (2004)
31. Risken, H.: *The Fokker Planck equation*, 2nd edition. Springer NY (1999)
32. Matkowsky, B.J. and Schuss, Z.: Eigenvalues of the Fokker-Planck operator and the approach to equilibrium for diffusions in potential fields. *SIAM J. App. Math.* **40** (2), 242–254 (1981)
33. Basri, R., Roth, D., and Jacobs, D.: Clustering appearances of 3D objects. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR-98)*, 414–420 (1998)
34. Roweis, S.T. and Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**, 2323–2326 (2000)
35. Kato, T.: *Perturbation Theory for Linear Operators*, 2nd edition. Springer (1980)
36. Nadler, B. and Galun, M.: Fundamental limitations of spectral clustering. *NIPS*, **19** (2006)
37. Nadler, B.: Finite Sample Convergence Results for Principal Component Analysis: A Matrix Perturbation Approach, submitted.
38. Zhou, D., Bousquet, O., Navin Lal, T., Weston J., and Scholkopf, B.: Learning with local and global consistency. *NIPS*, **16** (2004)
39. Kevrekidis, I.G., Gear, C.W., Hummer, G.: Equation-free: The computer-aided analysis of complex multiscale systems. *AICHE J.* **50** 1346–1355 (2004)

On Bounds for Diffusion, Discrepancy and Fill Distance Metrics

Steven B. Damelin

Department of Mathematical Sciences, Georgia Southern University, PO Box 8093,
Statesboro, GA, 30460, USA,
damelin@georgiasouthern.edu

Summary. Criteria for optimally *discretizing* measurable sets in Euclidean space is a difficult and old problem which relates directly to the problem of good numerical integration rules or finding points of low discrepancy. On the other hand, *learning* meaningful descriptions of a finite number of given points in a measure space is an exploding area of research with applications as diverse as dimension reduction, data analysis, computer vision, critical infrastructure, complex networks, clustering, imaging neural and sensor networks, wireless communications, financial marketing and dynamic programming. The purpose of this paper is to show that a general notion of extremal energy as defined and studied recently by Damelin, Hickernell and Zeng on measurable sets \mathcal{X} in Euclidean space, defines a *diffusion metric* on \mathcal{X} which is equivalent to a *discrepancy* on \mathcal{X} and at the same time bounds the *fill distance* on \mathcal{X} for suitable measures with discrete support. The diffusion metric is used to *learn* via normalized graph Laplacian dimension reduction and the discrepancy is used to *discretize*.

Key words: Affinity Matrix, Diffusion Distance, Discrepancy, Distribution, Discretization, Dimension Reduction, Eigenfunction Expansion, Energy, Equilibrium measure, Euclidean space, Fill Distance, Graph, Laplacian, Kernels, Mesh Norm, Numerical Integration, Orthonormal Basis, Positive Definite

11.1 Introduction

Criteria for optimally *discretizing* measurable sets in Euclidean space is a difficult and old problem which relates directly to the problem of good numerical integration rules or finding points of low discrepancy. Indeed, let us consider the problem of uniformly distributing points on spheres, more generally, on compact sets in $d \geq 1$ dimensional Euclidean space. It is folklore, that such problems were discussed already by Carl Friedrich Gauss in his famous *Disquisitiones arithmaticae*, although it is most likely that similar problems appeared in mathematical writings even before that time. For $d \geq 2$, let S^{d-1} denote the d -dimensional unit sphere in \mathbb{R}^d , given by

$$x_1^2 + \cdots + x_d^2 = 1.$$

For $d = 2$, the problem is reduced to uniformly distributing $n \geq 1$ points on a circle, and equidistant points provide an obvious answer. For $d \geq 3$, the problem becomes much more difficult; in fact, there are numerous criteria for uniformity, resulting in different optimal configurations on the sphere. See [6, 3, 8, 9, 10, 12, 11] and the references cited therein.

On the other hand, *learning* meaningful descriptions of a finite number of given points in a measure space (set learning) is an exploding area of research with applications as diverse as dimension reduction, data analysis, computer vision, critical infrastructure, complex networks, clustering, imaging neural and sensor networks, wireless communications, financial marketing and dynamic programming. See [1]-[25] and the references cited therein.

The purpose of this paper is to show that a general notion of extremal energy as defined and studied recently by Damelin, Hickernell and Zeng on measurable sets \mathcal{X} in Euclidean space, defines a *diffusion metric* on \mathcal{X} which is equivalent to a *discrepancy* on \mathcal{X} and at the same time bounds the *fill distance* on \mathcal{X} for suitable measures with discrete support. The diffusion metric is used to *learn* via normalized graph Laplacian dimension reduction and the discrepancy is used to *discretize*.

The remainder of this paper is structured as follows. In Sec. 2, we introduce needed ideas of energy, discrepancy, integration and distance on measurable sets in Euclidean space. In Sec. 3, we discuss set learning via normalized Laplacian dimension reduction and diffusion distance. Finally, Sec. 4 is devoted to our main result on bounds for discrepancy, diffusion and fill distance metrics.

11.2 Energy, Discrepancy, Distance and Integration on Measurable Sets in Euclidean Space

Here and throughout, let \mathcal{X} be a measurable subset of $d \geq 1$ Euclidean space \mathbb{R}^d and let $\mathcal{M}(\mathcal{X})$ denote the space of all (non zero), finite signed measures (distributions) μ on \mathcal{X} so that $Q(\mu) := \int_{\mathcal{X}} d\mu$ exists and is finite. We will henceforth call $Q(\mu)$ the total charge (mass) of μ . If the space $\mathcal{M}(\mathcal{X})$ is endowed with a norm $\|\cdot\|_{\mathcal{M}(\mathcal{X})}$, then the discrepancy problem measures the difference between any two measures in $\mathcal{M}(\mathcal{X})$ in the norm $\|\cdot\|_{\mathcal{M}(\mathcal{X})}$. Following, Damelin, Hickernell and Zeng, [6], let $K(\mathbf{x}, \mathbf{y})$ be a positive definite function on $\mathcal{X} \times \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^d$. This means that $\int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{x}) d\mu(\mathbf{y})$ exists, is finite and is positive for $\mu \in \mathcal{M}(\mathcal{X})$. Also we assume that K is symmetric, i.e., $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and $\int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{y})$ exists. We call K an *energy kernel*, which means that the *potential field* $\phi_{K,\mu}$ induced by the *charge distribution* μ on \mathcal{X} is

$$\phi_{K,\mu}(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{y}), \mathbf{x} \in \mathcal{X}.$$

The *energy* of a charge distribution $\mu \in \mathcal{M}(\mathcal{X})$ is

$$E_K(\mu) = \int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{x}) d\mu(\mathbf{y}),$$

and the *energy* of the charge distribution μ in the field

$$f_{K,\mu}(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\mu_f(\mathbf{y})$$

induced by the charge distribution μ_f is

$$E_K(\mu, \mu_f) = \int_{\mathcal{X}} f(\mathbf{x}) d\mu(\mathbf{x}) = \int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{x}) d\mu_f(\mathbf{y}) = \langle \mu, \mu_f \rangle_{\mathcal{M}} .$$

Here we see that $E_K(\mu, \mu_f)$ defines an inner product on the space \mathcal{M} of signed measures (charge distributions) for which the energy is finite. Sometimes we need only assume that K is *conditionally positive definite*, meaning

$$\int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{x}) d\mu(\mathbf{y}) > 0 \quad \forall \mu \neq 0 \text{ with } Q(\mu) = 0 .$$

For conditionally positive definite kernels the energy $E_K(\mu)$ may be negative. The energy $E_K(\mu, \mu_f)$ may be of either sign in general, even for positive definite kernels. In what follows, $|\cdot|$ and $|\cdot|_2^2$ denotes the Euclidean and squared L_2 norms in \mathbb{R}^d respectively.

We also call K the *reproducing kernel* of a Hilbert space, $H(K)$ which is a Hilbert space of functions $f : \mathcal{X} \rightarrow \mathbb{R}$. This means that $K(\cdot, \mathbf{y})$ is the representer of the linear functional that evaluates $f \in H(K)$ at \mathbf{y} :

$$f(\mathbf{y}) = \langle K(\cdot, \mathbf{y}), f \rangle_{H(K)} \quad \forall f \in H(K), \mathbf{y} \in \mathcal{X} .$$

For any $f, g \in H(K)$ with

$$f(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\mu_f(\mathbf{y}) \text{ and } g(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\mu_g(\mathbf{y})$$

it follows that their inner product is the energy of the two corresponding charge distributions:

$$\langle f, g \rangle_{H(K)} = E_K(\mu_f, \mu_g) = \int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\mu_f(\mathbf{x}) d\mu_g(\mathbf{y}) = \langle \mu_f, \mu_g \rangle_{\mathcal{M}} .$$

Note that a crucial feature of the function space $H(K)$ is that it depends directly on the kernel K .

In *electrostatics*, a positive charge μ placed upon a conductor (compact set) will distribute itself so as to minimize its energy. Equilibrium will be reached when the total energy is minimal amongst all possible possible charge distributions on \mathcal{X} . One would thus expect that the potential field $\int_{\mathcal{X}} K(x, y) d\mu(y)$ should be *constant* on compact \mathcal{X} most of the time for otherwise charge would flow from one point of \mathcal{X} to the next disturbing the equilibrium. The kernel K describes the interaction of the electrons (positive charges) on the conductor \mathcal{X} and a charge distribution which gives a constant potential field is an example of an equilibrium measure. More precisely, we have, see [6]:

Theorem 1. *Let K be an energy kernel. Then*

$$E_K(\mu) = \int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\mu(\mathbf{x}) d\mu(\mathbf{y}) \geq \frac{[Q(\mu)]^2}{C_K(\mathcal{X})}, \mu \in \mathcal{M}(\mathcal{X})$$

for a constant $C_K(\mathcal{X})$ depending only on \mathcal{X} and K (called the capacity of \mathcal{X}) and with equality holding for any equilibrium charge distribution or equilibrium measure,

$\mu_{e,K}$, defined as one that induces a constant field,¹

$$\phi_{K,\mu_{e,K}}(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\mu_{e,K}(\mathbf{y}) = \frac{Q(\mu_{e,K})}{C_K(\mathcal{X})} \quad \forall \mathbf{x} \in \mathcal{X}.$$

Examples of energy kernels are:

- (a) The weighted Riesz kernel on compact subsets of $d \geq 1$ Euclidean space \mathbb{R}^d

$$K_{s,w}(\mathbf{x}, \mathbf{y}) = \begin{cases} w(\mathbf{x}, \mathbf{y}) |\mathbf{x} - \mathbf{y}|^{-s}, & 0 < s < d, \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}, \\ -w(\mathbf{x}, \mathbf{y}) \log |\mathbf{x} - \mathbf{y}|, & s = 0, \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}, \\ w(\mathbf{x}, \mathbf{y})(c - |\mathbf{x} - \mathbf{y}|^{-s}), & -1 \leq s < 0, \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}, \end{cases}$$

where $w : \mathcal{X} \times \mathcal{X} \rightarrow (0, \infty)$ is chosen to be lower semi-continuous such that K is an energy kernel. Such kernels (in the case $w \equiv 1, s > 0$), arise naturally in describing the distributions of electrons on rectifiable manifolds such as the sphere S^d . The case when w is active, comes about for example in problems in computer modeling in which points are not necessarily uniformly distributed on \mathcal{X} . See [3, 8, 9, 10, 12]. The case when $-1 \leq s < 0$ appears more frequently in discrepancy theory. Here c is chosen so that the kernel is positive definite.

- (b) If $\rho(\text{dist}_K(\mathbf{x}, \mathbf{y}))$ is conditionally negative semi-definite and $\rho(0) = 0$, then $K(\mathbf{x}, \mathbf{y}) = \Psi(\rho(\text{dist}_K(\mathbf{x}, \mathbf{y})))$ is an energy kernel for any non constant, completely monotonic function Ψ on \mathcal{X} where dist_K is a metric on \mathcal{X}^2 which is defined by way of (11.2) below. For example, typical examples of such kernels are the heat kernel $\exp(-c|\mathbf{x} - \mathbf{y}|_2^2)$, $c > 0$ on \mathcal{X} and Hamming distance kernel used in the construction of linear codes when well defined.

In what follows, we will assume that our kernels are finite. Good numerical integration rules or good *discretizations* of \mathcal{X} are obtained generally by optimizing the placement of points and weights assigned to function values on \mathcal{X} . In the classical setting, the domain of integration is the interval $[-1, 1]$ and as is known, the nodes of the celebrated Gaussian quadrature formula, may be uniquely determined by the following characteristic property of the nodes of an $n \geq 1$ point Gauss quadrature: The n nodes are the zeros of the unique monic polynomial of minimal mean-square deviation on a real interval. In other words, the nodes are the zeros of the unique solution of an extremal problem. In particular, recent work of Damelin and his collaborators, see [3, 6, 8, 9] has shown that minimal energy points are good for integrating functions with respect to equilibrium distributions. More precisely, in numerical integration or cubature we approximate the integral of a function $f \in H(K)$,

$$I(f; \tilde{\mu}) = \int_{\mathcal{X}} f(\mathbf{x}) d\tilde{\mu}(\mathbf{x}) = E_K(\tilde{\mu}, \mu_f) = \langle \tilde{\mu}, \mu_f \rangle_{\mathcal{M}} = \langle \phi_{\tilde{\mu}}, f \rangle_{H(K)}$$

by the cubature rule

¹ Note that the capacity depends of course on the dimension d but since d is fixed when \mathcal{X} is chosen, this dependence is suppressed for notational convenience. Also the equilibrium measure depends on \mathcal{X} and d but again it is supported on \mathcal{X} by definition so we suppress this notation again. We adopt the same convention for other qualities as well throughout this paper which will be clear to the reader.

$$\begin{aligned} I(f; \hat{\mu}) &= \int_{\mathcal{X}} f(\mathbf{x}) d\hat{\mu}(\mathbf{x}) \\ &= \sum_{i=1}^n c_i f(\mathbf{x}_i) = E_K(\hat{\mu}, \mu_f) = \langle \hat{\mu}, \mu_f \rangle_{\mathcal{M}} = \langle \phi_{\hat{\mu}}, f \rangle_{H(K)} , \end{aligned}$$

where $\hat{\mu}$ is the charge distribution (signed measure) with support on the set of n points, $\mathbf{x}_1, \dots, \mathbf{x}_n$ and charge c_i at each point f_i . Moreover,

$$\phi_{\tilde{\mu}}(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\tilde{\mu}(\mathbf{y})$$

is the representer of the integration functional, and

$$\phi_{\hat{\mu}}(\mathbf{x}) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\hat{\mu}(\mathbf{y}) = \sum_{i=1}^n c_i K(\mathbf{x}, \mathbf{x}_i)$$

is the representer of the cubature rule functional. The error of this numerical approximation is

$$\begin{aligned} \int_{\mathcal{X}} f(\mathbf{x}) d\tilde{\mu}(\mathbf{x}) - \sum_{i=1}^n c_i f(\mathbf{x}_i) &= I(f; \tilde{\mu}) - I(f; \hat{\mu}) = \int_{\mathcal{X}} f(\mathbf{x}) d[\tilde{\mu} - \hat{\mu}](\mathbf{x}) \\ &= E_K(\tilde{\mu} - \hat{\mu}, \mu_f) = \langle \tilde{\mu} - \hat{\mu}, \mu_f \rangle_{\mathcal{M}} = \langle \phi_{\tilde{\mu}} - \phi_{\hat{\mu}}, f \rangle_{H(K)} . \end{aligned}$$

The *worst-case integration error* is defined as the largest absolute value of this error for integrands, f , with unit norm. By the Cauchy-Schwartz inequality we see that this occurs when f is parallel to $\phi_{\tilde{\mu}} - \phi_{\hat{\mu}}$, or equivalently, μ_f is parallel to $\tilde{\mu} - \hat{\mu}$. Thus, we have, see [6]:

Theorem 2.

$$\begin{aligned} D_K(\tilde{\mu}, \hat{\mu}) &:= \min_{\|f\|_{H(K)} \leq 1} \left| \int_{\mathcal{X}} f(\mathbf{x}) d\tilde{\mu}(\mathbf{x}) - \sum_{i=1}^n c_i f(\mathbf{x}_i) \right| \quad (11.1) \\ &= \sqrt{E_K(\tilde{\mu} - \hat{\mu})} = \|\tilde{\mu} - \hat{\mu}\|_{\mathcal{M}} = \|\phi_{\tilde{\mu}} - \phi_{\hat{\mu}}\|_{H(K)} \\ &= \left\{ \int_{\mathcal{X}^2} K(\mathbf{x}, \mathbf{y}) d\tilde{\mu}(\mathbf{x}) d\hat{\mu}(\mathbf{y}) - 2 \sum_{i=1}^n c_i \int_{\mathcal{X}} K(\mathbf{x}_i, \mathbf{y}) d\tilde{\mu}(\mathbf{y}) \right. \\ &\quad \left. + \sum_{i,k=1}^n c_i c_k K(\mathbf{x}_i, \mathbf{x}_k) \right\}^{1/2} . \end{aligned}$$

The quantity $D_K(\tilde{\mu}, \hat{\mu})$, defined by (1) which depends both on the placement and magnitude of the point charges defining $\hat{\mu}$, is called the *discrepancy*. We see that it is equivalent to the square root of an energy.

For a fixed choice of points $\mathcal{Y} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the best cubature rule, i.e., the choice of c_i that minimizes the discrepancy, is obtained by choosing the potential induced by $\hat{\mu}$ to match the potential induced by $\tilde{\mu}$ on \mathcal{Y} , i.e.,

$$\phi_{\hat{\mu}}(\mathbf{x}_i) = \phi_{\tilde{\mu}}(\mathbf{x}_i), \quad i = 1, \dots, n.$$

In this case

$$D_K(\tilde{\mu}, \hat{\mu}) = \{E_K(\tilde{\mu}) - E_K(\hat{\mu})\}^{1/2}.$$

The best choice of locations and magnitude of the charges is to find the set \mathcal{Y} consisting of n points that has *maximum energy* under the given constraint.

It is now possible to define a distance on \mathcal{X} by way of:

$$\text{dist}_K(\mathbf{x}, \mathbf{y}) := \sqrt{K(\mathbf{x}, \mathbf{x}) - 2K(\mathbf{x}, \mathbf{y}) + K(\mathbf{y}, \mathbf{y})}, \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}. \quad (11.2)$$

In the next section, we will show that the distance defined in (11.2), for certain energy kernels, arises as a diffusion metric which may be used to *learn* \mathcal{X} via normalized graph Laplacian dimension reduction. Our main result in Sec. 4 shows that this later distance is essentially equivalent to the discrepancy defined by way of (11.2).

11.3 Set Learning via Normalized Laplacian Dimension Reduction and Diffusion Distance

In this section, we show that the distance defined by way (2) arises in a natural way in set learning. See [2]. In what follows, we will need to define a class of energy kernels which we will call *admissible* in the sense below. Suppose first that K is non-negative. Next, suppose that K is zonal in the sense that it depends only on the Euclidean distance between two points \mathbf{x} and \mathbf{y} in \mathcal{X} . For example, Damelin, Levesley and Sun have shown in [8] that if \mathcal{X} is a compact d dimensional homogenous reflexive manifold with invariant group G , then if K is invariant under actions of G , it is zonal. Suppose that \mathcal{X} is a finite collection of points which are the vertices of an oriented graph and if $b(\mathbf{x}, \mathbf{y})$ denotes the associated adjacency matrix, then we assume that $b(\mathbf{x}, \mathbf{y}) = 1$ if there is an edge going from \mathbf{x} to \mathbf{y} and $b(\mathbf{x}, \mathbf{y}) = 0$ otherwise. Let $\hat{\mu}$ be a positive counting measure with support \mathcal{X} . K is now scaled in the following way. First, normalize K to be stochastic (to have sum 1 along its rows) by setting

$$v^2(x) = \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) d\hat{\mu}(\mathbf{y}), \quad \mathbf{x} \in \mathcal{X}$$

and defining

$$a(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x}, \mathbf{y})}{v(\mathbf{x})v(\mathbf{y})}, \quad \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

Also define $A : L_2(\hat{\mu}) \rightarrow L_2(\hat{\mu})$ by

$$Af(\mathbf{x}) := \int_{\mathcal{X}} a(\mathbf{x}, \mathbf{y})f(\mathbf{y}) d\hat{\mu}(\mathbf{y}), \quad \mathbf{x} \in \mathcal{X}.$$

Then A is bounded, has norm 1, and positive definite. As is well known, we may also assume that A is symmetric. Assuming that A is compact, we may also write

$$a(\mathbf{x}, \mathbf{y}) = \sum_{j \geq 0} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathcal{X},$$

where

$$A\phi_j(\mathbf{x}) = \lambda_j \phi_j(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X},$$

where the λ_j are discrete, non-increasing and non-negative. Given $m \geq 1$, following [23], we define $A^{(m)}(x, y)$ to be the kernel for A^m so that

$$a^{(m)}(\mathbf{x}, \mathbf{y}) = \sum_{j \geq 0} \lambda_j^m \phi_j(\mathbf{x}) \phi_j(\mathbf{y}), \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

At the level of the points in \mathcal{X} , $a^{(m)}$ denotes the probability for a Markov chain with transition matrix a to reach \mathbf{y} from \mathbf{x} in m steps. The mapping

$$\Psi(\mathbf{x}) := (\psi_0(\mathbf{x}), \psi_1(\mathbf{x}), \psi_2(\mathbf{x}), \dots)^T$$

maps \mathcal{X} into an Euclidean space and each eigenfunction is a coordinate on this set. Thus the mapping provides a dimension controlled representation of the points in \mathcal{X} in Euclidean space.

Define for each m on \mathcal{X} , $\text{dist}_{a^{(m)}}(\cdot)$ as given by (11.2). Then $\text{dist}_{a^{(m)}}(\cdot)$ is a diffusion distance which measures the rate of connectivity between points on the data set \mathcal{X} .

11.4 Main Result: Bounds for Discrepancy, Diffusion and Fill Distance Metrics

Following is our main result.

Theorem 3. Let \mathcal{X} be a finite measure space in \mathbb{R}^d for some $d \geq 1$ consisting of a finite collection of $n \geq 1$ points which we label as \mathbf{x}_i , $i = 1, \dots, n$. Let $\hat{\mu}$ be a positive counting measure with support \mathcal{X} and let $\tilde{\mu}$ be any measure in $\mathcal{M}(\mathcal{X})$. Let $m \geq 1$, let $K : \mathcal{X}^2 \rightarrow \mathbb{R}$ be admissible and define $\text{dist}_{a^{(m)}}(\cdot)$ as given by (2) and Sec. 3. Then the following hold:

(a)

$$\begin{aligned} D_{a^{(m)}}^2(\tilde{\mu}, \hat{\mu}) &= -\frac{1}{2} \int_{\mathcal{X}^2} \text{dist}_{a^{(m)}}^2(\mathbf{x}, \mathbf{y}) d\tilde{\mu}(\mathbf{x}) d\tilde{\mu}(\mathbf{y}) \\ &\quad + \frac{1}{n} \sum_{i=1}^n \int_{\mathcal{X}} \text{dist}_{a^{(m)}}^2(\mathbf{x}_i, \mathbf{x}) d\tilde{\mu}(\mathbf{x}) - \frac{1}{2n^2} \sum_{i,j=1}^n \text{dist}_{a^{(m)}}^2(\mathbf{x}_i, \mathbf{x}_j) \\ &\leq \text{fill}_{a^{(m)}}^2(\hat{\mu}), \end{aligned} \tag{11.3}$$

where

$$\text{fill}_{a^{(m)}}^2(\hat{\mu}) := \sup_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{X}} (\text{dist}_{a^{(m)}}(\mathbf{y}, \mathbf{x}))$$

is the fill distance or mesh norm of \mathcal{X} .

(b)

$$\text{dist}_{a^{(m)}}^2(\mathbf{x}, \mathbf{y}) = \sum_{j \geq 0} \lambda_j^m (\phi_j(\mathbf{x}) - \phi_j(\mathbf{y}))^2, \mathbf{x}, \mathbf{y} \in \mathcal{X}.$$

Part (a) of the theorem gives an equivalence between discrepancy on \mathcal{X} and diffusion distance. Notice that $\tilde{\mu}$ can be arbitrary in $\mathcal{M}(\mathcal{X})$. Part (b) says that discrepancy or diffusion metrics can be computed as weighted Euclidean distance in the embedded space of Ψ , the weights being precisely the eigenvalues of the operator A .

Proof Part (a) of the theorem follows from Theorem(2), the definition of A and (11.2). Part (b) follows from (11.2). \square

Acknowledgement. S. B. Damelin was supported, in part, by EP/C000285, NSF-DMS-0439734 and NSF-DMS-0555839. This work began while S. B. Damelin was on sabbatical at the Institute for Mathematics and Applications, University of Minnesota during the 2005-2006 academic year. The author wishes to thank the workshop organisers for a wonderful meeting.

References

1. Bajnok, B., Damelin, S.B., Li, J., and Mullen, G.: A constructive finite field method for scattering points on the surface of a d -dimensional sphere. *Computing*, **68**, 97–109 (2002)
2. Chung F.: Spectral Graph Theory. In: CBNS-AMS, **92**. AMS Publications, Providence, RI (1997)
3. Damelin, S.B. and Grabner, P.: Numerical integration, energy and asymptotic equidistribution on the sphere. *Journal of Complexity*, **19**, 231–246 (2003)
4. Damelin, S.B.: Marcinkiewicz-Zygmund inequalities and the Numerical approximation of singular integrals for exponential weights: Methods, Results and Open Problems, some new, some old. *Journal of Complexity*, **19**, 406–415 (2003)
5. Damelin, S.B., Hero, A., and Wang, S.J.: Kernels, hitting time metrics and diffusion in clustering and data analysis, in preparation.
6. Damelin, S.B., Hickernell F., and Zeng, X.: On the equivalence of discrepancy and energy on closed subsets of Euclidean space, in preparation.
7. Damelin, S.B. and Kuijlaars, A.: The support of the extremal measure for monomial external fields on $[-1, 1]$. *Trans. Amer. Math. Soc.* **351**, 4561–4584 (1999)
8. Damelin, S.B., Levesley, J., and Sun, X.: Energies, Group Invariant Kernels and Numerical Integration on Compact Manifolds. *Journal of Complexity*, submitted.
9. Damelin, S.B., Levesley, J., and Sun, X.: Energy estimates and the Weyl criterion on compact homogeneous manifolds. In: Algorithms for Approximation V. Springer, to appear.
10. Damelin S.B. and Maymeskul, V.: On point energies, separation radius and mesh norm for s -extremal configurations on compact sets in \mathbb{R}^n . *Journal of Complexity*, **21** (6), 845–863 (2005)
11. Damelin S.B. and Maymeskul, V.: Minimal discrete energy problems and numerical integration on compact sets in Euclidean spaces. In: Algorithms for Approximation V. Springer, to appear.
12. Damelin S.B. and Maymeskul, V.: On regularity and dislocation properties of minimal energy configurations on compact sets, submitted.
13. Du, Q., Faber, V., and Gunzburger, M.: Centroidal Voronoi tessellations: applications and algorithms. *SIAM Rev.* **41** (4), 637–676 (1999)
14. Helgason, S. Differential Geometry, Lie Groups, and Symmetric Spaces. (Graduate Studies in Mathematics.) American Mathematical Society (2001)
15. Hickernell, F.J.: Goodness of fit statistics, discrepancies and robust designs. *Statist. Probab. Lett.* **44** (1), 73–78 (1999)
16. Hickernell, F.J.: What affects the accuracy of quasi-Monte Carlo quadrature? In: Niederreiter, H. and Spanier, J. (eds.) Monte Carlo and Quasi-Monte Carlo Methods. Springer-Verlag, Berlin, 16–55 (2000)

17. Hickernell, F.J.: A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, **67** (221), 299–322 (1998)
18. Hickernell, F.J.: An algorithm driven approach to error analysis for multidimensional integration, submitted.
19. Hickernell, F.J.: My dream quadrature rule. *J. Complexity*, **19**, 420–427 (2003)
20. Huffman, W.C. and Pless, V.: *Fundamentals of error-correcting codes*, Cambridge University Press, Cambridge (2003)
21. Johnson, M.E., Moore, L.M., and Ylvisaker, D.: Minimax and maxmin distance designs. *J. Statist. Plann. Inference* **26**, 131–148 (1990)
22. Kuipers, L. and Niederreiter, H.: *Uniform Distribution of Sequences*. Wiley-Interscience, New York (1974)
23. Lafon, S. and Coifman, R.R.: Diffusion maps. *Applied and Computational Harmonic Analysis*, **21**, 5–30 (2006)
24. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. Volume 63 of SIAM CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia (1992)
25. Lubotzky, A., Phillips, R., and Sarnak, P.: Hecke operators and distributing points on the sphere (I-II). *Comm. Pure App. Math.* **39-40**, 148–186, 401–420 (1986, 1987)

Geometric Optimization Methods for the Analysis of Gene Expression Data

Michel Journée¹, Andrew E. Teschendorff², Pierre-Antoine Absil³, Simon Tavaré²,
and Rodolphe Sepulchre¹

¹ Department of Electrical Engineering and Computer Science, University of Liège, Belgium.

² Breast Cancer Functional Genomics Program, Cancer Research UK Cambridge Research Institute, Department of Oncology, University of Cambridge, Robinson Way, Cambridge CB2 0RE, UK.

³ Department of Mathematical Engineering, Université catholique de Louvain, Belgium

MJ and AET contributed equally to this work.

Summary. DNA microarrays provide such a huge amount of data that unsupervised methods are required to reduce the dimension of the data set and to extract meaningful biological information. This work shows that Independent Component Analysis (ICA) is a promising approach for the analysis of genome-wide transcriptomic data. The paper first presents an overview of the most popular algorithms to perform ICA. These algorithms are then applied on a microarray breast-cancer data set. Some issues about the application of ICA and the evaluation of biological relevance of the results are discussed. This study indicates that ICA significantly outperforms Principal Component Analysis (PCA).

12.1 Introduction

The transcriptome is the set of all mRNA molecules in a given cell. Unlike the genome, which is roughly similar for all the cells of an organism, the transcriptome may vary from one cell to another according to the biological functions of that cell as well as to the external stimuli. The transcriptome reflects the activity of all the genes within the cell. The quantity of a given mRNA is determined by a complex interaction between cooperative and counteracting biological processes. Understanding the intricate mechanism that induces the mRNA expression is an important step in elucidating the relation between the transcriptome of a cell and its phenotype. Microarray technology provides a quantitative measure of the concentration of mRNA molecules in a cell for the whole transcriptome in a systematic way. This measure is called the expression level of a gene. We refer to [1] and references therein for more details

about microarrays.

Microarray technology provides a huge amount of data, typically related to several thousand genes over a few hundred experiments, which correspond, e.g., to different patients, tissues or environmental conditions. Gene expression data sets are so large that a direct interpretation of them is usually infeasible. Unsupervised methods are required to reduce the dimension of the data set and to provide some biological insight in an automatic way. A typical approach is Principal Component Analysis (PCA) that generates a linear representation of the data in terms of components that are uncorrelated [2]. This linear decorrelation can reveal interesting biological information. Nevertheless, mapping the data to independent biological processes should provide a more realistic model.

The present paper proposes the use of ICA to help the biological interpretation of a gene expression data set that is related to breast cancer [3]. The ICA model is well-suited for the analysis of data sets that enclose an independence assumption within the data generation process. Intuitively, gene expression results from several biological processes (here we call them “expression modes”) that take place independently. Each of these biological processes involves various biological functions and rely on the activation or inhibition of a subset of genes. Several studies have already shown the value of ICA in the gene expression context, notably Liebermeister [4], who was the first to apply ICA to gene expression data. Important results on some bacterial and human databases are also detailed in [5, 6, 7]. These studies identified independent components and used the Gene Ontology (add citation from our other paper!) framework to evaluate their biological significance. The present paper extends the results from these previous studies by evaluating ICA in the framework of biological and cancer-related pathways, which constitutes the more relevant validation framework since genes with different GO-terms may be involved in the same pathway or biological process. Specifically, it tests the ICA-model for gene expression by showing that ICA significantly outperforms a non-ICA based method (PCA). Furthermore, it discusses some issues about the way to apply ICA and to evaluate the results. These issues are disregarded in most of the previous studies.

This paper starts with a review of some standard algorithms to perform ICA. The link between the ICA approach and the theory of geometric optimization is first outlined (Section 2). The three fundamental components of each ICA algorithm are then discussed, namely the contrast function (Section 3), the matrix manifold (Section 4) and the optimization process (Section 5). The last part of the paper (Section 6) illustrates the application of ICA to the analysis of gene expression data.

12.2 ICA as a Geometric Optimization Problem

The ICA approach was originally dedicated to the blind source separation problem, which recovers independent source signals from linear mixtures of them. As in the original paper of Comon [8], a linear instantaneous mixture model will be considered here,

$$X = AS, \quad (12.1)$$

where X , A and S are respectively matrices in $\mathbb{R}^{n \times N}$, $\mathbb{R}^{n \times p}$ and $\mathbb{R}^{p \times N}$ with p less or equal than n . The rows of S are assumed to stand for samples of independent random variables. Thus, ICA provides a linear representation of the data X in terms of components S that are statistically independent. Random variables are independent if the value of any one variable does not carry any information on the value of any other variable. By definition, p statistically independent random variables s_i have a joint probability distribution that equals the product of their marginal distributions, i.e.,

$$p(s_1, \dots, s_p) = p(s_1) \dots p(s_p).$$

Each ICA algorithm is based on the inverse of the mixing model (12.1), namely

$$Z = W^T X,$$

where Z and W are respectively matrices of $\mathbb{R}^{p \times N}$ and $\mathbb{R}^{n \times p}$. The rows of Z should represent random variables that are statistically independent. Unfortunately, the number of degrees of freedom available through the matrix W is usually insufficient for an exact independence. Thus, the rows of Z are just expected to represent random variables that are as independent as possible, such that ICA can be treated as an optimization problem.

Given an $n \times N$ data matrix X , an ICA algorithm aims at computing an optimum of a *contrast function*

$$\gamma : \mathbb{R}^{n \times p} \rightarrow \mathbb{R} : W \mapsto \gamma(W),$$

that estimates the statistical independence of the p random variables whose samples are given in the p rows of the matrix $Z = W^T X$.

It is important to note that the integer n is fixed by the chosen dataset X , while the integer p defines the number of components the user wants to compute. This number is, in most applications, smaller than n . Some contrast functions are able to deal with rectangular demixing matrices. Nevertheless, most of them are defined for square matrices only, such that the observations have to be preprocessed by means of prewhitening. Prewhitening is equivalent to Principal Component Analysis (PCA), which performs a Singular Value Decomposition of the matrix of the observations. The reduction of the

dimension is achieved by retaining the dominant p -dimensional subspace, that is, the subspace related to the p largest singular values. The matrix optimization is then applied on the prewhitened observations and identifies a square matrix of dimension p . Hence, the demixing matrix results from the product of two matrices. The first belongs to $\mathbb{R}^{n \times p}$ and is identified by prewhitening, while the second belongs to $\mathbb{R}^{p \times p}$ and results from the optimization of the contrast function. Most ICA algorithms present these two successive steps. They are denoted *prewhitening-based*.

Another important issue is to deal with the inherent symmetries of contrast functions. Symmetries are present because the measure of dependence between random variables must not be altered by scaling or by permutation of these variables. Optimizing a function with symmetries entails difficulties of theoretical (convergence analysis) and practical nature unless some constraints are introduced.

In the case of prewhitening-based ICA, where the whitening matrix X satisfies $XX^T = I$, it is common practice to restrict the matrix W to be orthonormal, i.e., $W^TW = I$. This implies that the sample covariance matrix ZZ^T is the identity matrix. Two options are conceivable to deal with the orthogonality constraint on W . First is to perform constrained optimization over a Euclidean space, i.e.,

$$\min_{W \in \mathbb{R}^{p \times p}} \gamma(W) \quad \text{such that} \quad W^TW = I.$$

This paper favors the second alternative which incorporates the constraints directly into the search space and performs unconstrained optimization over a nonlinear matrix manifold, i.e.,

$$\min_{W \in \mathcal{O}_p} \gamma(W) \quad \text{with} \quad \mathcal{O}_p = \{W \in \mathbb{R}^{p \times p} | W^TW = I\}. \quad (12.2)$$

Most classical unconstrained optimization methods - such as gradient-descent, Newton, trust-region and conjugate gradient methods - have been generalized to the optimization over matrix manifolds, in particular over the orthogonal group \mathcal{O}_p . Developing efficient matrix algorithms that perform optimization on a matrix manifold is a topic of active research (see the monograph [9] and references therein).

12.3 Contrast Functions

This section discusses the function γ of the general problem statement (12.2). This function measures the statistical independence of random variables. It presents thus a global minimum at the solution of the ICA problem. Many contrast functions have been proposed in the literature, and we review some of them here.

12.3.1 Mutual Information [8, 10]

Mutual information is a central notion of information theory [11, 12] that characterizes statistical independence. The mutual information $I(Z)$ of the multivariate random variable $Z = (z_1, \dots, z_p)$ is defined as the Kullback-Leibler divergence between the joint distribution and the product of the marginal distributions,

$$I(Z) = \int p(z_1, \dots, z_p) \log \frac{p(z_1, \dots, z_p)}{p(z_1) \dots p(z_p)} dz_1 \dots dz_p .$$

The mutual information presents all the required properties for a contrast function: it is non negative and equals zero if and only if the variables Z are statistically independent. Hence, its global minimum corresponds to the solution of the ICA problem.

Several approaches to compute efficiently an approximation to this quantity can be found in the literature. These approaches expand the mutual information in a sum of integrals that are expected to be more easily evaluated, namely the differential entropy and the negentropy. The differential entropy $S(z)$ and the negentropy $J(z)$ of a random variable z are respectively defined by

$$S(z) = \int p(z) \log(p(z)) dz, \quad \text{and} \quad J(z) = S(g) - S(z) ,$$

where g stands for a gaussian variable with same mean and variance as z . The mutual information can be expressed in terms of differential entropies as follows,

$$I(Z) = \sum_{i=1}^p S(z_i) - S(z_1, \dots, z_p) .$$

A similar expansion in terms of negentropies is given by

$$I(Z) = J(z_1, \dots, z_p) - \sum_{i=1}^p J(z_i) + \frac{1}{2} \log \frac{\prod C_{ii}^Z}{|C^Z|} ,$$

where C^Z denotes the covariance matrix of Z , $\prod C_{ii}^Z$ the product of its diagonal elements and $|\cdot|$ the determinant.

A contrast defined over the space of the demixing matrices is obtained once the demixing model $Z = W^T X$ is introduced within these two expansions, i.e.,

$$\gamma(W) = \sum_{i=1}^p S(e_i^T W^T X) - \log(|W|) - S(x_1, \dots, x_p) , \quad (12.3)$$

and

$$\gamma(W) = J(x_1, \dots, x_p) - \sum_{i=1}^p J(e_i^T W^T X) + \frac{1}{2} \log \frac{\prod C_{ii}^{W^T X}}{|C^{W^T X}|}, \quad (12.4)$$

where e_i is the i th basis vector. More details about the derivation of these expressions can be found in [10] for (12.3) and in [8] for (12.4). It should be noted that, once the observations are prewhitened and the demixing matrix is restricted to be orthogonal, both terms $\log(|W|)$ and $\frac{1}{2} \log \frac{\prod C_{ii}^{W^T X}}{|C^{W^T X}|}$ cancel.

At this point, statistical estimators are required to evaluate efficiently the differential entropy as well as the negentropy for a one-dimensional variable. Comon suggests using the Edgeworth expansion of a probability function in order to estimate the negentropy [8]. A truncated expansion up to statistics of fourth order leads to the following approximation,

$$J(z) \approx \frac{1}{12} \kappa_3^2 + \frac{1}{48} \kappa_4^2 + \frac{7}{48} \kappa_3^4 - \frac{1}{8} \kappa_3^2 \kappa_4,$$

where κ_i denotes the cumulant of order i of the standardized one-dimensional random variable z .

An efficient estimator of the differential entropy was derived by considering order statistics [10]. Given a one-dimensional variable z defined by its samples, the order statistics of z is the set of samples $\{z^1, \dots, z^N\}$ rearranged in non-decreasing order, i.e., $z^1 \leq \dots \leq z^N$. The differential entropy of a one-dimensional variable z defined by its order statistics $\{z^1, \dots, z^N\}$ can be estimated by a simple formula,

$$\hat{S}(z) = \frac{1}{N-m} \sum_{j=1}^{N-m} \log \left(\frac{N+1}{m} (z^{(j+m)} - z^{(j)}) \right), \quad (12.5)$$

where m is typically set to \sqrt{N} . This expression is derived from an estimator originally due to Vasicek [13]. The contrast of the RADICAL algorithm [10] is actually the function (12.3) where the differential entropies are evaluated with the estimator (12.5),

$$\gamma(W) = \sum_{i=1}^p \hat{S}(e_i^T W^T X) - \log(|W|) - S(x_1, \dots, x_p). \quad (12.6)$$

12.3.2 \mathcal{F} -Correlation [14]

This contrast is based on a generalization of the Pearson correlation coefficient, called the \mathcal{F} -correlation. It is proven in [14] that two random variables z_1 and z_2 are statistically independent if and only if the \mathcal{F} -correlation $\rho_{\mathcal{F}}$ vanishes, with $\rho_{\mathcal{F}}$ being defined by

$$\rho_{\mathcal{F}} = \max_{f_1, f_2 \in \mathcal{F}} \text{corr}(f_1(z_1) f_2(z_2)) ,$$

where $\text{corr}(x, y)$ is the Pearson correlation coefficient between the random variables x and y and \mathcal{F} is a vector space of functions from \mathbb{R} to \mathbb{R} . A contrast for the two-dimensional ICA problem is thus given by

$$\gamma(W) = \max_{f_1, f_2 \in \mathcal{F}} \text{corr}(f_1(w_1^T X) f_2(w_2^T X)) , \quad (12.7)$$

where w_i is the i -th column of the matrix W . This quantity seems complex to evaluate since it involves an optimization over a space of infinite dimension. Nevertheless, the authors of [14] showed that, by means of kernel methods [15, 16], this evaluation can be approximated by the solution of an eigenvalue problem of finite dimension. The \mathcal{F} -correlation $\rho_{\mathcal{F}}$ is estimated by the largest eigenvalue of a generalized eigenvalue problem of dimension $2N$ where N stands for the number of samples.

The contrast function (12.7) allows the identification of only two independent components. The paper [14] proposes a generalization to higher dimensions. The contrast remains the largest eigenvalue of a generalized eigenvalue problem, but of dimension pN , with p being the number of components. This contrast is the core of the KernelICA algorithm [14].

12.3.3 Non-Gaussianity [17]

Informally speaking, the central limit theorem states that the sum of independent random variables converges (in distribution) to a Gaussian variable as the number of terms tends to infinity. Thus, each linear combination of random variables is expected to be more gaussian than the original ones, which should be the most non-gaussian. A whole range of contrast functions is based on measuring the gaussianity (or non-gaussianity) of a one-dimensional random variable. The most intuitive expression for such a measure is given by

$$J(z) = (E[G(z)] - E[G(g)])^2 , \quad (12.8)$$

where $E[\cdot]$ is the expectation operator, G is a smooth function and g is a gaussian variable with same mean and variance as z . A quadratic function G enables to reveal features related to statistics up to the second order only, whereas non-gaussianity involves higher order statistics. For this reason, G should be non-quadratic. Some suggestions for the function G are given in [17]. For the particular choice of $G(z) = \frac{1}{4}z^4$, the distance to gaussianity becomes the square of the kurtosis $\kappa(z)$,

$$J(z) = \kappa(z)^2 .$$

The kurtosis is a classical measure of non-gaussianity since it equals zero for gaussian variables and has a large absolute value for non-gaussian ones.

The link between non-gaussianity and statistical independence is rigorously proven in the particular case of the kurtosis. A theorem in [18] states that the kurtosis of the sum of two independent variables z_1 and z_2 presents a smaller absolute value than the largest absolute value of the kurtosis among these variables, i.e.,

$$|\kappa(z_1 + z_2)| \leq \max(|\kappa(z_1)|, |\kappa(z_2)|) .$$

A contrast function is easily obtained from these measures of gaussianity by introducing the ICA model $z = w^T X$ in equation (12.8),

$$\gamma(w) = (E[G(w^T X)] - E[G(g)])^2 , \quad (12.9)$$

where w is a vector of \mathbb{R}^n . The maximization of this contrast results in the FastICA algorithm [17], probably the most popular ICA algorithm. It is important to note that this contrast is one-unit based, i.e., it is defined for one column of the demixing matrix W . The notion of statistical independence becomes meaningful once the optimization is performed several times from different initial conditions and identifies different sources z .

12.3.4 Joint Diagonalization of Cumulant Matrices [19]

Independent random variables are also characterized by the diagonality of some statistically motivated matrices. A necessary condition for statistically independent random variables is, for example, given by a diagonal covariance matrix. However, identifying the transformation W that diagonalizes the covariance matrix of $Z = W^T X$ for some observations X will only identify components that are uncorrelated but not independent.

In case of a zero-mean random variable, the covariance matrix can be considered as a particular case of the concept of *cumulant tensors*. The covariance matrix is then simply the cumulant tensor of second order. We refer to [20] for the definition and properties of these tensors. The essential issue for the present review is that cumulant tensors related to statistically independent variables are diagonal at any order. A particular matrix is derived from the fourth order cumulant tensor that presents the desired property of being diagonal in case of independent variables. If \mathcal{Q}_X^4 denotes the fourth order cumulant tensor of the p -variate random variable X , the *cumulant matrix* $Q_X(M)$ related to a matrix M is defined elementwise by

$$Q_X(M)|_{ij} = \sum_{k,l=1}^p \mathcal{Q}_X^4|_{ijkl} M_{kl} ,$$

where $\mathcal{Q}_X^4|_{ijkl}$ denotes the element at position (i, j, k, l) of the fourth order cumulant tensor. The cumulant matrix can be efficiently evaluated without the computation of the whole tensor \mathcal{Q}_X^4 thanks to the following expression,

$$\begin{aligned} Q_X(M) &= E[(X^T M X) X X^T] \\ &\quad - E[XX^T] \text{tr}(M E[XX^T]) - E[XX^T](M + M^T)E[XX^T], \end{aligned}$$

where $\text{tr}(\cdot)$ denotes the trace. This expression is valid only if the variable X has a zero mean.

An important property of the cumulant matrix is that the orthogonal transform $Z = W^T X$ results in a similarity transform for $Q(M)$, i.e.,

$$Q_Z(M) = W^T Q_X(M) W,$$

whatever the matrix M .

A set of matrices that are simultaneously diagonal can be constructed by picking several matrices M . The maximal set of cumulant matrices is obtained whenever the matrices M form an orthogonal basis for the linear space of $p \times p$ symmetric matrices. The matrix W that diagonalizes simultaneously all the cumulant matrices performs a projection of the observations X on random variables Z that are statistically independent. However, it is usually impossible to identify a joint diagonalizer W that diagonalizes exactly all these matrices. This suggests defining ICA algorithms that optimize the joint diagonalization of them.

Several cost functions for this approximate joint diagonalization are conceivable, but a frequently encountered one is the following,

$$\gamma(W) = \sum_i \|\text{off}(W^T Q_X(M_i) W)\|_F^2, \quad (12.10)$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\text{off}(A)$ is a matrix with entries identical to those of A except on the diagonal, which contains only zero-valued elements. Optimizing (12.10) means that one minimizes the sum of the squares of all non-diagonal elements of the cumulant matrices. This is consistent with performing the best approximate joint diagonalization of these matrices. Thanks to the diagonal property of the cumulant matrices in case of independent variables, this cost function is at the same time a contrast for ICA.

This contrast is used by the famous JADE algorithm [19]. We mention that other type of matrices have been defined in the literature that are also diagonal in the case of independent random variables. The SOBI algorithm (Second Order Based Identification), for example, performs ICA by the approximate joint diagonalization of matrices that involve only second order statistics [21].

12.4 Matrix Manifolds for ICA

Each contrast function considered in the previous section possesses two symmetries. First, since the statistical independence of random variables is not affected by a scaling, the following equality holds,

$$\gamma(W) = \gamma(WA),$$

whenever A is a diagonal matrix. Likewise, the permutation of random variables does not affect their statistical independence, i.e.,

$$\gamma(W) = \gamma(WP),$$

where P is a permutation matrix. Because of these symmetries, the global minimum of the contrast is not a single point but a set of subspaces of $\mathbb{R}^{p \times p}$. Optimizing cost functions that present continuous symmetries is a difficult task. Therefore, constraints have to be added to restrict these subspaces to a set of discrete points. In the present case, the scaling symmetry disappears if each column of W is set to a fixed norm, for example to a unit-norm. This constraint set defines the so-called oblique manifold [22],

$$\mathcal{OB}_p = \{W \in \text{GL}(p) : \text{diag}(W^T W) = I\},$$

where $\text{GL}(p)$ is the set of all invertible $p \times p$ matrices. This manifold gets rid of the continuous scaling symmetry. The permutation symmetry defines the global optimum as a set of discrete points. This is suitable to most optimization algorithms. Hence, the oblique manifold is the most general manifold to perform ICA in an efficient way.

Usually, some further constraints are imposed. Prewhitening-based ICA algorithms preprocess the data by means of Principal Component Analysis (PCA). It is shown in [8] that this allows us to restrict the ICA optimization to the orthogonal group,

$$\mathcal{O}_p = \{W \in \mathbb{R}^{p \times p} | W^T W = I_p\}.$$

Furthermore, this manifold ensures a good conditioning of the optimization algorithms.

As mentioned in the previous section, the FastICA contrast is defined for one column of the demixing matrix W , while most of the other contrasts are defined on a matrix space. In order to remove the scaling symmetry, the optimization of that contrast is performed on the sphere,

$$S^{n-1} = \{w \in \mathbb{R}^n | w^T w = 1\}.$$

Some implementations of the FastICA algorithm perform in parallel several optimizations of the contrast (12.9) starting from different initial conditions

[17]. In order to prevent convergence toward identical minima, the parallel iterates are reorthogonalized at each iteration, so that the algorithm simultaneously identifies p independent columns of the demixing matrix. Since these columns are orthogonal, the data must be preprocessed by PCA, but the square constraint $p = n$ disappears. These implementations are equivalent to an optimization over the Stiefel manifold,

$$\text{St}(n, p) = \{W \in \mathbb{R}^{n \times p} | W^T W = I_p\}.$$

12.5 Optimization Algorithms

12.5.1 Line-Search Algorithms

Many optimization algorithms on a Euclidean space are based on the following update formula,

$$x_{k+1} = x_k + t_k \eta_k. \quad (12.11)$$

which consists to move from the current iterate x_k in the search direction η_k with a certain step size t_k to identify the next iterate x_{k+1} . The search direction and the step size are chosen such that the cost function decreases sufficiently at each iteration. The search direction is usually set to the opposite of the gradient of the cost function γ at the current iterate, i.e.,

$$\eta_k = -\text{grad}\gamma(x_k).$$

The iterate is thus moving in the direction of steepest-descent. The step size has then to be selected in order to induce a significant decrease of the cost function. Iteration (12.11) is, however, valid only in case the iterates belong to a Euclidean space.

On non-Euclidean manifolds, the update formula (12.11) is generalized to

$$W_{k+1} = R_{W_k}(t_k \eta_k),$$

where W_k and W_{k+1} are two successive iterates on the manifold \mathcal{M} , t_k is a scalar and η_k belongs to $T_{W_k}\mathcal{M}$, the tangent space to \mathcal{M} at W_k . The retraction $R_W(\eta)$ is a mapping from the tangent plane to the manifold. More details about this concept can be found in [9].

The search direction η_k is, as above, set to the opposite of the gradient of the cost function γ at W_k ,

$$\eta_k = -\text{grad}\gamma(W_k).$$

In case of a manifold \mathcal{M} embedded in a Euclidean space $\bar{\mathcal{M}}$, the gradient at $W \in \mathcal{M}$ is simply computed as the orthogonal projection onto $T_W\mathcal{M}$ of the gradient in the embedding space, i.e.,

$$\text{grad}\gamma(W) = P_W \text{grad}\bar{\gamma}(W) ,$$

where $\bar{\gamma}$ denotes a smooth extension on $\bar{\mathcal{M}}$ of the cost function γ , i.e.,

$$\bar{\gamma}(W) = \gamma(W), \quad \forall W \in \mathcal{M} .$$

We refer to [9] for more details about the orthogonal projector P_W .

The gradient in the embedding Euclidean space is defined, as usual, from the directional derivative,

$$\langle \text{grad}\bar{\gamma}(W), \zeta \rangle = D\bar{\gamma}(W)[\zeta] = \lim_{t \rightarrow 0} \frac{\bar{\gamma}(W + t\zeta) - \bar{\gamma}(W)}{t} ,$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product.

To complete the description of the gradient-descent algorithm, the choice of the step size t_k has to be discussed. Several alternatives are conceivable. First, an exact line-search identifies the minimum in the direction of search. Such an optimization is usually tricky and requires a huge computational effort. A good alternative is given by the Armijo step size t^A . This step size is defined by

$$t^A = \beta^m \alpha ,$$

with the scalars $\alpha > 0$, $\beta \in (0, 1)$ and m being the first nonnegative integer such that,

$$\gamma(W) - \gamma(R_W(\beta^m \alpha)) \geq -\sigma \langle \text{grad}\gamma(W), \beta^m \alpha \rangle_W ,$$

where W is the current iterate on \mathcal{M} , $\sigma \in (0, 1)$, $R_W(\eta)$ is a retraction.

The KernelICA algorithm performs a gradient-descent optimization of the \mathcal{F} -correlation $\rho_{\mathcal{F}}$ on the orthogonal group \mathcal{O}_p [14]. The projection operator and the retraction are respectively given by

$$P_W(\eta) = (I - WW^T)\eta \quad \text{and} \quad R_W(\eta) = W \exp(W^T \eta) .$$

We refer to the original paper [14] for the details about the derivation of the gradient.

A gradient-descent algorithm for the minimization over the orthogonal group of the contrast function (12.6), originally dedicated to the RADICAL algorithm [10], was recently proposed by us in [23].

Not only line-search methods generalize on nonlinear manifolds. We mention, among others, trust-regions and conjugate gradient algorithms. More details about these methods can be found in [9].

12.5.2 FastICA

FastICA algorithms perform the optimization of the cost function γ in one direction at the time,

$$\max_{w \in \mathbb{R}^n} \gamma(w), \quad \text{such that } w^T w = 1, \quad (12.12)$$

where w is one column of the demixing matrix W . In this case, the orthonormal constraint $W^T W = I_n$ reduces to a spherical constraint $w^T w = 1$.

The FastICA approach then exploits standard constrained optimization schemes. The solution of the problem (12.12) has to satisfy the Kuhn-Tucker condition

$$\frac{\partial \gamma(w)}{\partial w} - \beta w = 0,$$

where $w \in \mathbb{R}^n$ and β is a Lagrange multiplier. A Newton method to solve this equation results in the iteration,

$$w^+ = w - \left(\frac{\partial^2 \gamma(w)}{\partial w^2} - \beta I \right)^{-1} \cdot \left(\frac{\partial \gamma(w)}{\partial w} - \beta w \right), \quad (12.13)$$

where w^+ denotes the new iterate. The central point of the FastICA algorithm is to approximate the matrix inversion of the Hessian by a simple scalar inversion. It is shown in [17] that once the data is prewhitened, the Hessian of the contrast is close to a scalar matrix,

$$\frac{\partial^2 \gamma(w)}{\partial w^2} \approx \frac{\partial^2 \tilde{\gamma}(z)}{\partial z^2} I,$$

with

$$\tilde{\gamma} : \mathbb{R} \rightarrow \mathbb{R} \quad \text{such that } \tilde{\gamma}(w^T X) = \gamma(w),$$

where X is the data. Hence, iteration (12.13) can be approximated by

$$w^+ = w - \frac{1}{\frac{\partial^2 \tilde{\gamma}(z)}{\partial z^2} - \beta} \cdot \left(\frac{\partial \gamma(w)}{\partial w} - \beta w \right). \quad (12.14)$$

Multiplying both sides of (12.14) by $(\beta - \frac{\partial^2 \tilde{\gamma}(z)}{\partial z^2})$ results in,

$$w^+ = \left(\beta - \frac{\partial^2 \tilde{\gamma}(z)}{\partial z^2} \right) \cdot w + \left(\frac{\partial \gamma(w)}{\partial w} - \beta w \right) = \frac{\partial \gamma(w)}{\partial w} - \frac{\partial^2 \tilde{\gamma}(z)}{\partial z^2} w.$$

The new iterate is normalized at each iteration to a unit-norm to ensure the stability of the algorithm,

$$w^+ = \frac{w^+}{\|w^+\|}.$$

Hence, the FastICA algorithm consists in repeating the iteration

$$\begin{cases} w^+ = \frac{\partial \gamma(w)}{\partial w} - \frac{\partial^2 \tilde{\gamma}(z)}{\partial(z)^2} w, & \text{with } z = w^T X, \\ w^+ = \frac{w^+}{\|w^+\|}. \end{cases} \quad (12.15)$$

More details about this algorithm can be found in [17].

The algorithm (12.15) identifies one column only of the demixing matrix W . Nevertheless, it can be used to reconstruct several columns of that matrix by means of a deflation technique. Assuming prewhitening of the data, the demixing matrix is orthogonal. Suppose that p columns $\{w_1, \dots, w_p\}$ of the whole matrix W have been computed. The one-unit algorithm will converge to a new vector w_{p+1} that is orthogonal to the already known directions if, after each iteration, the projections of these directions are subtracted,

$$w_{p+1}^+ = w_{p+1} - \sum_{j=1}^p w_{p+1}^T w_j w_j.$$

The drawback of any deflation scheme is that small computational errors are amplified by the computation of new vectors. Thus, the first computed directions should be accurately estimated. The last ones should be expected to be tagged with larger errors.

An alternative is the symmetric orthogonalization, which performs all the computations in parallel without favoring some directions. Each one-unit algorithm is randomly initialized and the iterates are reorthogonalized after each iteration according to,

$$W^+ = (WW^T)^{-\frac{1}{2}}W.$$

The matrix square root can be avoided by means of an iterative algorithm that is described in [17].

12.5.3 Jacobi Rotations

Jacobi rotations provide a classical optimization method on the orthogonal group [24, 8, 19]. The iterates are constrained on the orthogonal group by successive multiplication by special orthogonal matrices W_k containing a single parameter,

$$W_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos(\alpha) & \cdots & \sin(\alpha) \\ & & & \ddots & \\ & & \vdots & & 1 \\ & & & & \ddots \\ & & & & & \cos(\alpha) \\ & & & & & -\sin(\alpha) \\ & & & & & \vdots \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix}.$$

Such matrices achieve a planar rotation of angle α in the subspace spanned by the directions (i) and (j) . At each iteration, a new subspace is selected and the best rotation is computed to maximize the cost function. Hence, only a one-dimensional optimization problem has to be solved at each iteration. In case of the JADE algorithm [19], this task is performed analytically such that an explicit expression of the optimal angle α is available. The RADICAL algorithm on the other hand [10] performs the global minimization over that parameter by exhaustive search on $[0, 2\pi]$, or more precisely on $[0, \frac{\pi}{2}]$ because of the permutation symmetry of the contrast function.

12.6 Analysis of Gene Expression Data by ICA

12.6.1 Some Issues About the Application of ICA

To fix the notations, let us define the gene expression matrix X such that its element (i, j) corresponds to the expression level of gene i in the j th experiment. X is thus a $n \times N$ matrix, where n is the number of analyzed genes and N is the number of experiments. Note that n is usually much larger than N . The breast cancer database considered for the following analysis is related to $n = 17816$ genes and $N = 286$ patients. The number n is typical for genome-wide expression data while $N = 286$ is fairly large in comparison with most other profiling studies in breast cancer.

Two modelling hypothesis underlie the application of ICA to gene expression data. First is that the expression level of a gene is a linear superposition of biological processes, some of which try to express it, while others try to repress it. Specifically, ICA performs an approximate decomposition of the gene expression matrix into two smaller matrices A and B that are respectively $n \times p$ and $p \times N$ with $p < N$, i.e.,

$$X \approx AB. \quad (12.16)$$

Figure 12.1 provides a clearer idea of this decomposition by representing matrices with Hinton diagrams. In such diagrams, each value is displayed by a square whose size is an image of the magnitude.

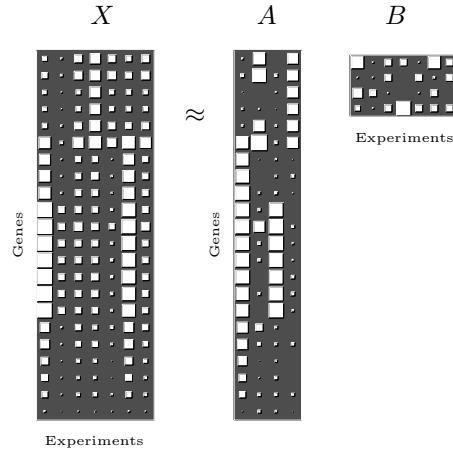


Fig. 12.1. ICA decomposition of a gene expression database X .

Each column of A is a vector in the gene space that is called an *expression mode*. In the same spirit, each row of B is a vector in the samples space that stands for the *activity* of the corresponding expression mode. This means that the expression of one particular gene across a set of samples (e.g. different patients) is modelled as the superposition of a restricted number of expression modes (typically about 10). The activity is the relative weight of an expression mode across the samples. The matrices A and B are selected to minimize the error between X and AB in the least-square sense. The second hypothesis imposes a notion of independence somewhere in the underlying biology. Hence, the matrices A and B have to maximize a certain measure of statistical independence.

Two alternatives are conceivable for the statistical independence assumption within the model (12.16). Either the independence is set in sample space, i.e., the rows of B stand for independent random variables (model I), or it is set in gene space, i.e., the columns of A stand for independent random variables (model II). We next describe the underlying and distinct motivations for the two models.

In the case of model I, the algorithm seeks independent components across samples, which amounts to finding non-gaussian distributions for the rows of B . It is expected that these non-gaussian projections separate samples into

biologically relevant subtypes, although there is no requirement that this be so. As such, the algorithm performs a kind of unsupervised projection pursuit.

In contrast model II seeks independent components across genes. There is a strong biological motivation for this model, since it is most natural to assume that the measured gene expression levels represent the net effect of many independent biological processes, which may or may not have different activity levels across samples. Furthermore, it is natural to assume that most genes in an expression mode do not play an active role in it, and that it is only a small percentage of genes which are relevant to any given expression mode. This is tantamount to assuming that the distribution of weights in the columns of A are supergaussian (leptokurtic), which fits in well with the ICA model II.

An issue of importance deals with the estimation of p , the number of independent components that underlie the gene expression data set. Some approaches compute the probability of observing the data X if the underlying model contains p components. This enables a rough estimation of the number of independent components. The Bayesian Information Criterion (BIC) in a maximum likelihood framework [25] or using the evidence bound in a variational Bayesian approach [26] are examples of such methods. In the present study, we choose a fixed number of components for all algorithms. The correct estimation of the number of components seems difficult because of the small number N of samples available.

12.6.2 Evaluation of the Biological Relevance of the Expression Modes

The general objective of PCA and ICA methods is to identify a small set of variables, in terms of which the data can be more easily interpreted. Previous applications of ICA to gene expression data have evaluated the results by means of the Gene Ontology (GO) framework. However, this does not provide the best framework in which to validate these methods, since genes with the same GO-annotation may not necessarily be part of the same biological pathway, and vice versa, genes that are part of the same pathway may have quite distinct biological functions. Instead of GO, we propose to use the framework of biological pathways, since it is the alteration pattern of specific pathways that underlies the cancer-cell phenotype. The independent components derived from ICA are expected to summarize the net effect of independent altered transcriptional programs in cancer, and as such, should map closer to aberrations in existing biological and cancer-related pathways. While research in cancer biology is still at the stage of trying to elucidate all the pathways that may be involved, several efforts are underway in building up pathway databases. Some of these pathways have been curated from various sources, while others were determined by specific experiments. Each pathway in these databases is essentially a list of genes that are known to participate

together when a certain biological function is required. In this work, we evaluate the PCA and ICA methods against their ability to correlate expression modes with known pathways. To our knowledge, the PCA and ICA methods were never evaluated in the explicit context of biological pathways.

Let us specify more concretely the concept of mapping between a pathway and an expression mode. Each expression mode is a list of all the genes with an associated weight. Since a pathway is just a list of some specific genes that are known to present a linked activity, the expression modes require some post-processing that consists in selecting the genes with major weights. To identify the genes that are differentially activated, it is common to impose a threshold of typically 2 or 3 standard deviations from the mean of the distribution of the inferred weights. A gene list that looks like a pathway is thus obtained by selecting the genes with an absolute weight that exceeds this threshold. A stringent threshold at three standard deviations was chosen in the present study to reveal the most relevant pathways captured by each of the expression modes.

If the application of ICA/PCA in the gene expression context is biologically well-founded, each expression mode should be strongly tagged by a specific pathway or by a superposition of some distinct pathways that are highly dependent. A quantitative measure of the enrichment level of a given expression mode i in a given pathway p has to be defined. Let n_i denote the number of genes in the expression mode and n_p denote the number of genes in the pathway. Further, let d_i denote the number of genes selected in the expression mode i and t_{ip} the number of genes from pathway p among the selected d_i genes. Under the null-hypothesis, where the selected genes are chosen randomly, the number t_{ip} follows a hypergeometric distribution [27]. Specifically, the probability distribution is

$$\begin{aligned} P(t) &= \binom{d_i}{t} \prod_{j=0}^{t-1} \frac{n_p - j}{n_i - j} \prod_{j=0}^{j=d_i-t-1} \frac{n_i - n_p - j}{n_i - t - j} \\ &= \frac{\binom{n_p}{t} \binom{n_i - n_p}{d_i - t}}{\binom{n_i}{d_i}}. \end{aligned}$$

A probability can thus be computed as $P(t > t_{ip})$. This quantity enables to estimate for each mode-pathway pair how the mode is enriched in terms of genes from that particular pathway. This mode-pathway association will be said significant if the P-value $P(t > t_{ip})$ is less than a certain threshold. To evaluate the biological significance of the ICA approach, a large set of pathways is selected and a pathway enrichment index (PEI) is defined as the fraction of biological pathways that are found to be enriched in at least one expression mode.

12.6.3 Results Obtained on the Breast Cancer Microarray Data Set

PCA as well as the four ICA methods detailed in the first sections of this paper (i.e., JADE [19], RADICAL [10], KernelICA [14] and FastICA [17]) have been applied to one of the largest breast cancer microarray data set available [3]. The analysis was performed for both models I and II. The number of components p was fixed to 10 in each study. Since the four ICA algorithms are all prewhitening-based, the reduction of the dimensions of the problem from N , the number of experiments, to p is simply done by Singular Value Decomposition (SVD) during the prewhitening step. The ICA step computes thereafter a square demixing matrix of dimensions $p \times p$.

To evaluate the biological significance of the results, we compiled a list of 536 pathways that are known to be directly or indirectly involved in cancer biology. 522 of these pathways come from the Molecular Signature Database MSigDB [28]. The others are known oncogenic pathways recently derived in [29] and cancer-signalling pathways coming from the resource NETPATH (www.netpath.org). Figure 12.2 shows the pathway enrichment index (PEI) based on these 536 pathways for the five methods for both models I and II.

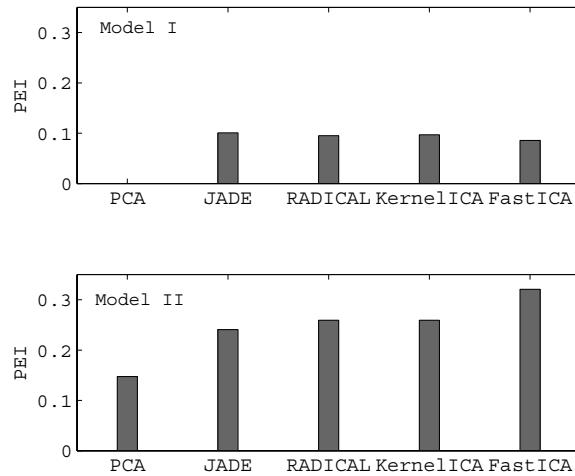


Fig. 12.2. PEI based on a set of 536 pathways for both models I and II.

The same kind of analysis was performed on the reduced set of the oncogenic pathways and the cancer-signalling pathways of NETPATH. Since these 14 pathways are frequently altered in cancer, many of them are expected to

be captured by the expression modes. The PEI related to them are illustrated on Figure 12.3.

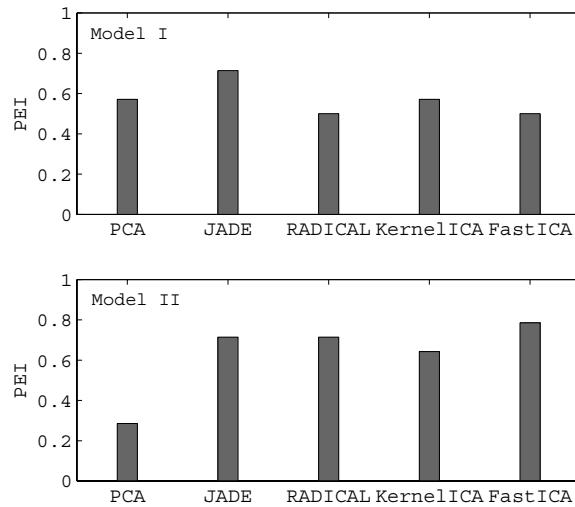


Fig. 12.3. PEI based on a set of 14 cancer-signalling and oncogenic pathways for both models I and II.

Both Figures 12.2 and 12.3 indicate that ICA achieves a more realistic representation of the gene expression data than PCA. Furthermore, the PEI values are clearly higher for model II than for model I. Hence, the ICA-model with the independence assumption stated in the gene space seems to be the most efficient approach to unveil the biological significance of the gene expression data. It is however difficult to discriminate between different ICA algorithms.

The present work is part of a larger project that investigates the assets of the ICA approach for the biological interpretation of microarray databases. A deeper analysis of the ICA decomposition has been performed on a total of nine microarray data sets related to breast cancer, gastric cancer and lymphoma. This large study also favors the use of ICA with model II for the analysis of gene expression data. It highlights that ICA algorithms are able to extract biological information not detected by PCA methods. More details about this study can be found in [30].

12.7 Conclusion

DNA microarrays enable new perspectives in biomedical research and especially in the understanding of some cellular mechanisms. This is of outmost interest for the treatment of cancer diseases. This emerging technology provides such a huge amount of data that unsupervised algorithms are required to automatically unveil the biological processes that have led to the observed transcriptome. The present paper reviews some standard algorithms to perform Independent Component Analysis and emphasizes their common feature, namely the optimization of a measure of statistical independence (the contrast) over a matrix manifold. The paper then illustrates the way to use these algorithms in the context of gene expression data. Even if the application of ICA to gene expression data sets is not a new idea, the evaluation of the results in the explicit context of biological pathways, has never been performed before. The main conclusion of this study is the significant outperformance of the ICA approach against Principal Component Analysis (PCA). The ICA model, with the statistical independence assumption stated in the gene space, seems to be a realistic representation of the mechanisms that determine the gene expression levels. ICA shows significant promise for the analysis of DNA microarray databases.

Acknowledgement. This work was supported by the Belgian National Fund for Scientific Research (FNRS) through a Research Fellowship at the University of Liège (MJ), by Microsoft Research through a Research Fellowship at Peterhouse, Cambridge (PAA), by a grant from Cancer Research UK and by a grant from the Isaac Newton Trust to Simon Tavaré (AET). This paper presents research results of the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office. The scientific responsibility rests with its authors.

References

1. Riva, A., Carpentier, A.-S., Torrésani, B., and Hénaut A.: Comments on selected fundamental aspects of microarray analysis. *Computational Biology and Chemistry*, **29** (5), 319–336 (2005)
2. Alter, O., Brown, P. O., and Botstein, D.: Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms. *Proc Natl Acad Sci USA*, **100** (6), 3351–3356, March (2003)
3. Wang, Y., Klijn, J. G., Zhang, Y., Sieuwerts, A. M., Look, M. P., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M. E., Yu, J., Jatkoe, T., Berns, E. M., Atkins, D., and Foekens, J. A.: Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet*, **365** (9460), 671–679, February (2005)
4. Liebermeister, W.: Linear modes of gene expression determined by independent component analysis. *Bioinformatics*, **18**, 51–60 (2002)

5. Martoglio, A.-M., Miskin, J. W., Smith, S. K., and MacKay, D. J. C.: A decomposition model to track gene expression signatures: preview on observer-independent classification of ovarian cancer. *Bioinformatics*, **18** (12), 1617–1624 (2002)
6. Lee, S.-I. and Batzoglou, S.: Application of independent component analysis to microarrays. *Genome Biology*, **4**, R76 (2003)
7. Saidi, S. A., Holland, C. M., Kreil, D. P., MacKay, D. J. C., Charnock-Jones, D. S., Print, C. G., and Smith S. K.: Independent component analysis of microarray data in the study of endometrial cancer. *Oncogene*, **23** (39), 6677–6683 (2003)
8. Comon, P.: Independent Component Analysis, a new concept ? *Signal Processing, (Special issue on Higher-Order Statistics)*, **36** (3), 287–314, April (1994)
9. Absil, P.A., Mahony, R., and Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, To appear.
10. Learned-Miller, E.G. and Fisher, J. W. III.: ICA using spacings estimates of entropy. *Journal of Machine Learning Research*, **4**, 1271–1295 (2003)
11. Mackay, D. J. C.: Information Theory, Inference & Learning Algorithms. Cambridge University Press (2002)
12. Cover, T. M. and Thomas, J. A.: Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience (2006)
13. Vasicek, O.: A test for normality based on sample entropy. *Journal of the Royal Statistical Society, Series B*, **38**, 54–59 (1976)
14. Bach, F. R. and Jordan, M. I.: Kernel independent component analysis. *Journal of Machine Learning Research*, **3**, 1–48 (2003)
15. Saitoh, S.: Theory of Reproducing Kernels and its Applications. Longman Scientific & Technical, Harlow, England (1988)
16. Scholkopf, B. and Smola, A. J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2001)
17. Hyvärinen, A., Karhunen, J., and Oja, E.: Independent Component Analysis. John Wiley & Sons (2001)
18. Mathis, H.: Nonlinear Functions for Blind Separation and Equalization. PhD thesis, Swiss Federal Institute of Technology, Zrich, Switzerland (2001)
19. Cardoso, J.-F.: High-order contrasts for independent component analysis. *Neural Computation*, **11** (1), 157–192 (1999)
20. De Lathauwer, L. and Vandewalle, J.: Dimensionality reduction in higher-order signal processing and rank- (R_1, R_2, \dots, R_n) reduction in multilinear algebra. *Lin. Alg. Appl.*, **391**, 31–55 (2004)
21. Belouchrani, A., Abed-Meraim, K., Cardoso, J.-F., and Moulines E.: A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, **45**, 434–444, February (1997)
22. Absil, P.-A. and Gallivan, K.A.: Joint diagonalization on the oblique manifold for independent component analysis. In: Proceedings of ICASSP2006 (2006)
23. Journée, M., Teschendorff, A. E., Absil, P.-A., and R. Sepulchre: Geometric optimization methods for independent component analysis applied on gene expression data. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007), April (2007)
24. Golub, G. H. and Van Loan, C. F.: Matrix Computations. The Johns Hopkins University Press (1996)

25. Hansen, L., Larsen, J., and Kolenda, T.: Blind detection of independent dynamic components. In: Proceedings of ICASSP'2001, Salt Lake City, Utah, USA, SAM-P8.10, vol. 5 (2001)
26. Minka, T. P.: Automatic choice of dimensionality for PCA. In: NIPS, pages 598–604 (2000)
27. Beißbarth, T., and Speed, T. P.: GOstat: find statistically overrepresented gene ontologies within a group of genes. *Bioinformatics*, **20** (9), 1464–1465 (2004)
28. Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., and Mesirov J. P.: From the cover: Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS*, **102** (43), 15545–15550, October (2005)
29. Bild, A. H., Yao, G., Chang, J. T., Wang, Q., Potti, A., Chasse, D., Joshi, M.-B., Harpole, D., Lancaster, J. M., Berchuck, A., Olson, J. A., Marks, J. R., Dressman, H. K., West, M., and Nevins J. R.: Oncogenic pathway signatures in human cancers as a guide to targeted therapies. *Nature*, **439**, 353–357 (2006)
30. Teschendorff, A. E., Journée, M., Absil, P.-A., Sepulchre, R., and Caldas, C.: Elucidating the altered transcriptional programs in breast cancer using independent component analysis. Submitted to PLoS Biology (2007)

13

Dimensionality Reduction and Microarray data

David A. Elizondo, Benjamin N. Passow, Ralph Birkenhead, and Andreas Huemer

Centre for Computational Intelligence, School of Computing, Faculty of Computing Sciences and Engineering, De Montfort University, Leicester, UK,{elizondo,passow,rab,ahuemer}@dmu.ac.uk

Summary. Microarrays are being currently used for the expression levels of thousands of genes simultaneously. They present new analytical challenges because they have a very high input dimension and a very low sample size. It is highly complex to analyse multi-dimensional data with complex geometry and to identify low-dimensional “principal objects” that relate to the optimal projection while losing the least amount of information. Several methods have been proposed for dimensionality reduction of microarray data. Some of these methods include principal component analysis and principal manifolds. This article presents a comparison study of the performance of the linear principal component analysis and the non linear local tangent space alignment principal manifold methods on such a problem. Two microarray data sets will be used in this study. A classification model will be created using fully dimensional and dimensionality reduced data sets. To measure the amount of information lost with the two dimensionality reduction methods, the level of performance of each of the methods will be measured in terms of level of generalisation obtained by the classification models on previously unseen data sets. These results will be compared with the ones obtained using the fully dimensional data sets.

Key words: Microarray data; Principal Manifolds; Principal Component Analysis; Local Tangent Space Alignment; Linear Separability; Neural Networks.

13.1 Introduction

Microarray data is arrived at by using a high-throughput experimental technology in molecular biology. This data is used for parallel analysis of genes which may be involved in a particular disease. This high dimensional data is characterised by a very large variable/sample ratio. Typically, they contain a large number (up to tens of thousands) of genes, each expressed as a number. The number of samples, for each of these genes is relatively small (several tens). The high dimensionality of this data has two main consequences. On

the one hand, it makes its analysis challenging. On the other hand, intuitively, it might increase the likelihood that the data will be linearly separable.

The problem of high dimensionality can be approached with the use of dimensionality reduction methods. Principal component analysis and principal manifolds are commonly used methods for the analysis of high dimensional data. Principal manifolds were introduced by Hastie and Stuetze in 1989 as lines or surfaces passing through the middle of the data distribution [8]. This intuitive definition was supported by the mathematical notion of self-consistency: every point of the principal manifold is a conditional mean of all points that are projected into this point. In the case of datasets only one or zero data points are projected in a typical point of the principal manifold, thus, one has to introduce smoothers that become an essential part of the principal manifold construction algorithms.

One important application of principal manifolds is dimension reduction. In this field they compete with multidimensional scaling methods and the recently introduced advanced algorithms of dimension reduction, such as locally linear embedding (LLE) [12] and ISOMAP [19] algorithms. The difference between the two approaches is that the later ones seek new point coordinates directly and do not use any intermediate geometrical objects. This has several advantages, in particular that a) there is a unique solution to the problem (the methods are not iterative in their nature, there is no problem of grid initialisation) and b) there is no problem of choosing a good way to project points onto a non-linear manifold. This paper will use the principal manifold non-linear dimension reduction algorithm based on local tangent space alignment introduced in [23]. This method has been previously used to reduce the dimensionality of microarray data with good results [20]. The local tangent space alignment is a novel and interesting method which is available as a *ready to go* Matlab tool box [10] that has already been tested and verified.

Other work related to the analysis of Microarray data using dimensionality reduction techniques include [15], [11] and [21]. In [15] a semi-parametric approach is used to produce generalised linear models reducing the dimension, [11] uses graph theoretical methods to aid the search for models of reduced dimension and [21] uses discriminant partial least squares to provide models with more explanation of the response variables than might arise from the standard PCA method.

It is important to analyze the amount of information that is lost by the dimensionality reduction methods. This is why this article proposes the development of linear classifiers, using the fully dimensional and dimensionally reduced data sets, as a way to measure and compare the effects on the data caused by reducing the dimensionality. In the case of linearly separable data sets, several methods can be used to provide a separating hyperplane [3, 18]. When the sets are not linearly separable, a linear neural network such as the Recursive Deterministic Perceptron [5, 16, 17] or a Backpropagation Neural Network [13, 14] can be used.

In this study, two microarray data sets are used. The first data set is classified into three classification criteria. These include: five types of breast cancer type, positive or negative Estrogen-Receptor, and aggressive or non aggressive cancer. The second data set is classified into three types of bladder cancer.

This paper is divided into five sections. Some background information about dimension reduction and about the notion of linear separability are given in section two. This includes the introduction of a linear and a nonlinear dimensionality reduction methods, Principal Component Analysis and Local Tangent Space Alignment respectively. In section three, the procedure used to compare the two dimensionality reduction methods is presented with the use of two microarray data sets. Section four presents some results and discussion. A summary and some conclusions are presented in section five.

13.2 Background

In this section, some of the standard notions used throughout this chapter are introduced, together with some definitions and properties.

13.2.1 Microarray data

In biological and medical research, DNA microarray technology is widely used to study gene expression in cells for example in the diagnosis of diseases including cancer. Therefore, this technology is a very important and widely used method in research and diagnosis. Unfortunately, the data produced by this method is highly dimensional. High dimensionality could mean tens or tens of thousands of dimensions, depending on the circumstances and experiment setup on which this data is produced. In this study, two microarray data sets, provided at the first workshop in principal manifolds¹ held in Leicester in 2006, were used. The first data set [22], here after referred to as D1, was initially used in breast cancer research to identify patterns of breast cancer gene expressions that could be used to predict the patients disease progression. The data set consists of 17816 gene expressions. As a standard procedure, the data is preprocessed in such a way that the absolute average expression level is zero. This is done because only the difference in expression between samples, as opposed to the overall gene expression, contains useful biological meaning . This data set contains a total of 286 samples which correspond to the number of patients from which the samples were taken.

The second data set used in this study [6], here after referred to as data set D2, was originally used in bladder cancer research. The data set consists of 3036 gene expressions, also preprocessed to have zero mean as in the case of the D1 data set. The number of samples in this data set is 40.

¹ <http://www.ihes.fr/~zinovьев/princmanif2006/>

One of the features of the data sets is that their dimensionality is much higher than the sample size. This makes the analysis extremely difficult if no dimensionality reduction method is applied beforehand. Therefore, this kind of data is mostly reduced into a denser representation, keeping only the “most important” aspects of the data. The number of available methods that can be used for this reduction is growing, especially because there is no “correct” solution possible due to the fact that some information is always lost in the process.

13.2.2 Methods for Dimension Reduction

Several methods for the analysis of high dimensional data have been proposed including principal components analysis (linear) and principal manifolds (non linear). In this study, the level of performance of the principal component analysis (PCA) and the local tangent space alignment (LTSA) non linear principal manifold methods is studied for dimensionality reduction of high dimensional microarray data.

Principal Component Analysis

The PCA dimensionality reduction method [9, 7] is a linear dimensionality reduction method. It works by projecting a number of correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. The algorithm solves for the eigenvalues and eigenvectors of a square symmetric matrix with sums of squares and cross products. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component. The sum of the eigenvalues equals the trace of the square matrix and the maximum number of eigenvectors equals the number of rows (or columns) of this matrix.

Local Tangent Space Alignment

The Local Tangent Space Alignment algorithm was introduced in [23]. The LTSA is a nonlinear dimensionality reduction method that aims to find a global coordinate system within a low dimensional space that best characterises the high dimensional data set. It finds an approximation to the tangent space at each point using a neighbourhood and then aligns these tangent spaces in a process of constructing the coordinate system for the non-linear manifold. The computation speed is affected by the choice of size of neighbourhood due to the search cost for the nearest neighbours. There was an interaction between the LTSA output and the perceptron algorithm used for finding

the separating plane. The perceptron algorithm took longer to converge in cases where the LTSA was applied with small neighbourhoods, requiring a trade off to get improved performance.

Strengths and Limitations of the PCA and LTSA Methods

Currently there is no general method available to distinguish the nature of data as being linear or non linear [20]. The high dimensional space and low number of samples found in microarray data sets makes it appear highly likely to be linearly separable data and hides away any non linear structures. This is why the study of both linear and non linear dimensionality reduction methods is interesting.

PCA is a well established method that has been used over many years and frequently on Microarray data. According to [20] the PCA is fast to compute and easy to implement. Its complexity of the PCA algorithm is $O(n_s \times n)$ [2] where n_s represents the sample size, and n the original dimensionality. This method is guaranteed to find a lower dimensional representation of the data on a linear subspace if such representation exists. However, as mention in [1] the PCA method can only identify gross variability as opposed to distinguishing among and within groups variability.

In [20], the LTSA method has been used to reduce the dimensionality of microarray data with good results and it outperformed the linear PCA method in some aspects. The LTSA is a fairly recent method that reduces the high dimensionality on data sets by using tangent space produced by fitting an affine subspace in the proximity of each data sample. The LTSA algorithm is using a k-nearest neighbours search that can be computational expensive for large k and large input matrices. A small neighbourhood can result in less accurate dimensionality reduction. Also, the computation of the smallest eigenvectors of the alignment matrix used by the algorithm, is computationally expensive. Therefore, the PCA is more computationally efficient than the LTSA. Nevertheless, the non linear structures intrinsic in the data can not be efficiently exploited by using the linear PCA method.

13.2.3 Linear Separability

Preliminaries

The following standard notions are used: Let $\mathbf{p}_1, \mathbf{p}_2$ be the standard position vectors representing two points P_1 and P_2 in \mathbb{R}^d ,

- The set $\{t\mathbf{p}_1 + (1 - t)\mathbf{p}_2 \mid 0 \leq t \leq 1\}$ is called the segment between $\mathbf{p}_1, \mathbf{p}_2$ and is denoted by $[\mathbf{p}_1, \mathbf{p}_2]$.
- The dot product of two vectors $\mathbf{u} = (u_1, \dots, u_d), \mathbf{v} = (v_1, \dots, v_d)$ is defined as $\mathbf{u}^T \mathbf{v} = u_1 v_1 + \dots + u_d v_d$. $Adj(\mathbf{u}, r) = (u_1, \dots, u_d, r)$ and by extension $Adj(S, r) = \{Adj(\mathbf{x}, r) \mid \mathbf{x} \in S\}$.

- $\mathcal{P}(\mathbf{w}, t)$ stands for the hyperplane $\{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{w}^T \mathbf{x} + t = 0\}$ of \mathbb{R}^d . \mathbf{w} is the normal (i.e. is perpendicular), to the hyperplane \mathcal{P} . The threshold t is proportional to the distance from the origin to \mathcal{P} . \mathcal{P} will stand for the set of all hyperplanes of \mathbb{R}^d .

Two sub-sets X and Y of \mathbb{R}^d are said to be linearly separable (LS) if there exists a hyperplane P of \mathbb{R}^d such that the elements of X and those of Y lie on opposite sides of it. Figure (13.1) shows an example of both a LS (a) and a NLS (b) set of points. Squares and circles denote the two classes.

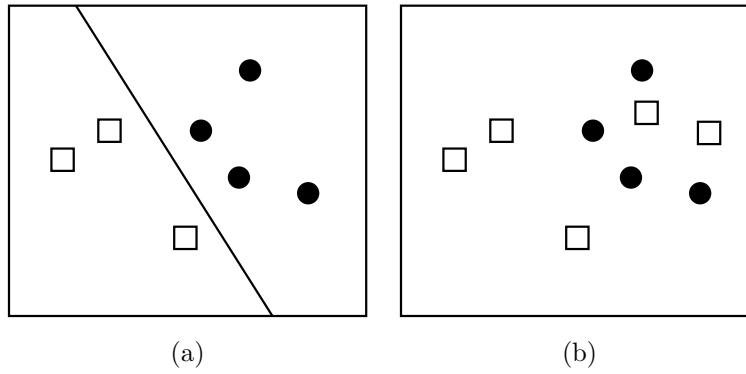


Fig. 13.1. LS (a) and a non-LS (b) set of points.

Methods for testing linear separability

The methods for testing linear separability between two classes can be divided into five groups:

- **The methods based on solving systems of linear equations.** These methods include: the Fourier-Kuhn elimination algorithm, and the Simplex algorithm. The original classification problem is represented as a set of constrained linear equations. If the two classes are LS, the two algorithms provide a solution to these equations.
- **The methods based on computational geometry techniques.** The principal methods include the convex hull algorithm and the class of linear separability method. If two classes are LS, the intersection of the convex hulls of the set of points that represent the two classes is empty. The class of linear separability method consists in characterising the set of points P of \mathbb{R}^d by which it passes a hyperplane that linearly separates two sets of points X and Y .

- **The methods based on neural networks.** The perceptron neural network is one of most commonly used methods for testing linear separability. If the two classes are LS, the perceptron algorithm is guaranteed to converge, after a finite number of steps, and will find a hyperplane that separates them.
- **The methods based on quadratic programming.** These methods can find a hyperplane that linearly separates two classes by solving a quadratic optimisation problem. This is the case for the Support Vector Machines.
- **The Fisher Linear Discriminant method.** This method tries to find a linear combination of input variables, $w \times x$, which maximises the average separation of the projections of the points belonging to the two classes C_1 and C_2 while minimising the within class variance of the projections of those points.

These methods are described in detail in [4]. Several heuristic methods, to reduce the calculation time while testing for linear separability, are presented in [3]. Table 13.1 describes the perceptron algorithm which is one of the most commonly used algorithms for testing linear separability. A faster more efficient algorithm for testing linear separability on relatively small data sets is the Simplex. However, due to the high dimensionality of the microarray data sets, the use of the Simplex algorithm becomes impractical. This is why the selected method for testing linear separability, used for this work, is the Perceptron algorithm.

Table 13.1. The Perceptron Learning algorithm.

PERCEPTRON(S, w)

```

– data: a set of vectors  $S$  constructed from the classes  $X$  and  $Y$  we wish to
distinguish
– result: a weight vector  $w$  which separates the two classes if they are LS
 $w_0 := x_0; (x_0 \in I\!\!R^{d+1})$ 
 $k := 0;$ 
while ( $\exists x_k \in S$ ) such that ( $w_k^T x_k \leq 0$ ) do
  Begin
     $w_{k+1} := w_k + x_k;$ 
     $k := k + 1;$ 
  End

```

The problem of finding a hyperplane $\mathcal{P}(w, t)$ which separates sets X and Y in $I\!\!R^d$ [i.e. finding w, t where $w^T x + t > 0$ and $w^T y + t < 0$ for all

$\mathbf{x} \in X, \mathbf{y} \in Y]$ is equivalent to finding $\mathbf{w}_1 \in \mathbb{R}^{d+1}$ for which $(\mathbf{w}_1^T \mathbf{s} > 0 \ \forall \mathbf{s} \in S)$ where $S = Adj(X, -1) \cup -Adj(Y, -1)$. [Given \mathbf{w}_1 which solves the “S problem” in \mathbb{R}^{d+1} , the separability hyperplane in \mathbb{R}^d is $\mathcal{P}(\mathbf{w}, t)$ where $\mathbf{w}_1 = Adj(\mathbf{w}, -t)$. It can be seen that $\mathbf{w}^T \mathbf{x} + t > 0$ and $-\mathbf{w}^T \mathbf{y} - t > 0 \Rightarrow \mathbf{w}^T \mathbf{y} + t < 0$]. The presented perceptron algorithm is implemented in \mathbb{R}^{d+1} solving the problem of S . The solution to the original problem for X and Y is then obtained.

13.3 Comparison Procedure

The comparison performed in this study was realised using microarray data sets together with their given classifications. These sets were split up into training and testing data sets, 60% for training and 40% for testing. For the classification model a neural network was first developed with the training set and then tested with the testing set. This procedure was also done with two dimensionality reduction methods before the data set was split and the classification model was used. The dimensions of the data sets were reduced by using principal component analysis (PCA) and the local tangent space alignment (LTSA) manifolds introduced in section two. The level of performance of the dimensionality reduction methods was measured in terms of the level of generalisation obtained and the time to compute needed by the classification models on previously unseen data sets. The system used for this comparison was a modern standard PC with Matlab installed. All computation and implementation of algorithms was realised in Matlab using standard toolboxes as well as the toolbox_dimreduc - a toolbox for dimension reduction methods [10] . The following section will explain the complete comparison procedure in more detail.

13.3.1 Data sets

Microarray data set D1 was initially used to identify patterns of breast cancer gene expressions that could be used to predict the patients disease progression. It is a collection of 17816 gene expressions with a sample size of only 286. Three ab initio sample classifications are available with the data set:

- **Group** non-aggressive (A) vs aggressive (B) breast cancer
- **ER** estrogen-receptor positive (ER+) vs negative (ER-) tumours
- **Type** lumA, lumB, normal, errb2, basal and unclassified breast cancer types

The table 13.2 gives an overview of the classes, the number of members of each class as well as the number of members of other classes. As a regular perceptron was used in this work to classify the sets, only one class versus the rest of the classes could be handled at one time. Therefore the type classification was done by training and testing each of the six classes separately.

The other data set used for the comparison task is the data set D2, identified in section 2. The data set has 3036 gene expressions with a sample size of only 40. It was originally analyzed to gather knowledge on bladder cancer. The available classification used in this work is the clinical categorisation of the tumour into three classes.

- Type T1, T2+ and Ta tumour types

Table 13.2. DNA Microarray Data Set D1 Classes

Data Set	Classification	Classes	Samples per class	Rest
D1	Group	non-aggressive (A)	193	93
		aggressive (B)	93	193
	Estrogen-Receptor	positive (ER+)	209	77
		negative (ER-)	77	209
	Cancer type	lumA	95	191
		lumB	25	261
		normal	35	251
		erb2	34	252
		basal	55	231
		unclassified	42	244
D2	Tumour type	T1	11	29
		T2+	9	31
		Ta	20	20

Cross Validation

In this study the technique of cross validation was applied to split the data into training and testing data sets. In detail, ten equally sized classification data sets were made, sixty percent of the samples were used for training the neural networks and the remaining forty percent were used for testing purposes. The ten sets were made out of the original data set by moving a window 10 percent for each new set. The resulting ten sets each include all data from the original set but with the training and testing sets varying. Thus, for each of the two dimensionality reduction methods, ten different neural networks were developed and tested using different combinations of test sets that were picked up from the ten similar sample sets. Without this method the single test result could be interpreted in a wrong way due to e.g. an uneven distribution of samples in the set. If the method of cross validation is applied, one can use statistical methods of the multiple results to provide a more general interpretation of the results.

Sample Distribution

The microarray data set D2 was originally sorted with respect to their membership to the classes. Simply splitting up the data set into training and testing sets would result in sets missing a complete class or containing samples of only a single class. In order to solve this problem the data was randomised into sets with the constraint that at least three members of any class were present in each train and test data set.

13.3.2 Dimensionality Reduction

In general, high dimensional data, especially combined with a low sample size, can make further analysis with the raw data computational expensive and difficult. For comparison purposes in this work, two dimensionality methods were chosen, the principal component analysis and the local tangent space alignment. These two methods represent a well known linear approach and a nonlinear manifold learning approach.

Principal Component Analysis (PCA)

PCA is a method well known and frequently used for dimensionality reduction. It projects the high dimensional data onto a new coordinate system with fewer dimensions. The highest amount of the variance of the original data is taken care of by the first coordinate or principal component, the second highest amount by the second principal component and so on. The matlab function *princomp* was used for this study. Due to the very high dimensionality, the flag *econ* had to be used when executing the *princomp* function. This flag restricts the computation and output to only include the eigenvalues of the covariance matrix of the input data that are not necessarily zero. Without this flag set, the returned principal component coefficients matrix itself would have taken 2.4 GB of RAM, exceeding the memory available to Matlab. In order to compare the dimensionality reduction methods the execution time for the *princomp* function was measured and noted. After the dimensionality reduction took place, the chosen number of dimensions to be used for further computation correspond to 80 percent of the total variance of the components, the eigenvalues of the covariance matrix. For the data set D1, 80 percent correspond to 150 dimensions. For the data set D2, 80 percent correspond to 19 dimensions. The preprocessed data returned by *princomp* was then split into the training and test sets that were subsequently used to train and test the perceptron.

Local Tangent Space Alignment (LTSA)

The LTSA technique used in this work is a nonlinear manifold learning method. The algorithm is part of the toolbox *dimreduc* available for Matlab.

Using this implementation, two parameters need to be specified next to the input matrix: the dimensionality of the output matrix as well as the number of nearest neighbours to take into account. The correct choice of parameters is important to achieve a good result within a reasonable time frame.

A careful choice of the number of output dimensions is important too. The parameters used for the data sets and classifications are listed in table 13.3. Due to the huge size of the D1 data set, the parameters for it were chosen by hand using trial and error. The parameters for the D2 data set on the other hand were searched for by a script checking a large number of sensible parameter combinations. In the end the best solution was chosen. As part of the comparison process, the time needed for the LTSA to execute was measured and noted. Thereafter the resulting data was split up into training and testing data sets for the perceptron.

Table 13.3. LTSA parameters used

Data set	Classification	Dimensions	Neighbours
D1	Group	114	133
	Estrogen - Receptor	80	99
	Cancer type	90	131
D2	Tumor type	18	31

13.3.3 Perceptron Models

The classification model is using a simple feedforward artificial neural network, a perceptron. It is capable of solving any linear separable classification problem in a limited time frame. The perceptron takes the two classes from the training data set and develops the network weights and bias accordingly. The iteration limit for the implemented perceptron was set to 10,000. The time the network needed to calculate the weights was measured and noted. Next, the network was tested using the testing data set. The level of generalisation, the percentage of successful classifications by the sum of all classifications, was computed and noted as well.

This procedure was used to compare the generalisation and the time needed to compute the data of the original high dimensional microarray data and the data sets reduced in dimensionality with the PCA and the LTSA methods beforehand. This process is repeated for all of the ten cross validation data sets, for all classes of all classifications, and the two data sets.

13.4 Results

This section presents results on the comparison of the convergence time and the level of generalisation obtained with the classification models created using

raw and dimensionality reduced microarray data sets. Since both in high and low dimensions the data sets are linearly separable, a single layer Perceptron neural network was used to create the classification models. The technique of cross validation was applied to split the microarray data sets into training and testing data sets. Table 13.4 shows the times needed for each manifold method to reduce the dimensionality of the data sets. As seen before, the PCA method produces more dimensions than the LTSA. However, the convergence time of the PCA method is less than 10% of that of the LTSA one. The PCA convergence times are similar for each data set. The LTSA method shows convergence time differences due to the choice of different dimensions and nearest neighbours taken into account (see table 13.3).

Table 13.4. Dimensionality reduction time taken by each manifold method (seconds)

Data Set	Class	PCA	LTSA
D1	Type	9.96	165.78
	Group	10.16	169.80
	ER+/-	9.86	138.90
D2	Type	0.109	0.697

Table 13.5 shows the average convergence time (across ten cross validation data sets), required to train the Perceptron neural networks using raw and PCA and LTSA preprocessed microarray data sets. It can be clearly seen that both PCA and LTSA give a dramatic improvement in the construction of the Perceptron classification neural networks for the D1 data set with mean differences ranging from 4.8 up to 45 times faster. This is expected as the original number of dimensions was dramatically cut down using the two dimensionality reduction methods. Overall, the average smaller convergence times were obtained with the PCA preprocessed data sets. For the D2 data sets, this difference is not as conclusive with convergence times ranging from 0.15 to 2 times faster. The D2 data set has only 24 training samples compared to 172 for the D1 data set. This could lead, depending on the distribution of the samples in the space, to more cycles of the perceptron algorithm for the D1 data set compared to the D2 one.

Although the times of convergence obtained with the perceptron neural network using the PCA and the LTSA dimensionality reduced data sets are not as clear in difference, the time to reduce the dimensionality using these two methods is large, with the PCA being, on average, 15 times faster for the D1 data set, and 7 times faster for the D2 data set as seen in table 13.4. The LTSA reduces the dimensions of the original data sets to less than the PCA method. This is due to the choice of dimensions and nearest neighbour parameters for the LTSA method as introduced previously in table 13.3. Better results with the perceptron neural network were obtained by fine tuning these parameters.

For the data set D1 it can be seen that it is more time efficient to reduce the dimensionality with a PCA beforehand training the perceptron. The overall time to reduce the dimensions with the PCA and train the perceptron is less than training the perceptron on the original data set. This is due to the many fewer dimensions the perceptron needs to take into account when finding the hyperplane which linearly separates the different classes. For the data set D1, the dimensions left, after applying the PCA, are less than one percent of the original data set.

The LTSA dimensionality reduction is more time consuming than the PCA method. However, when classifying multiple classes of the D1 data set, applying the LTSA dimension reduction method, prior to training the perceptron, is more time efficient as well.

Because of the smaller dimensionality of the D2 data set, the results, in terms of time, are not as conclusive as the ones obtained with the D1 data set.

Table 13.5. Results obtained in terms of time needed to train (convergence time) the Perceptron neural network. All values in seconds.

Data set	No Dim. Reduction				PCA				LTSA			
	Min	Mean	Max	Mode	Min	Mean	Max	Mode	Min	Mean	Max	Mode
D1 - Type:												
Class 1 vs Rest	12.43	18.15	29.81	12.43	0.37	0.46	0.55	0.37	2.69	3.78	4.44	2.69
Class 2 vs Rest	71.83	100.15	148.82	71.83	1.23	2.30	2.86	1.23	0.53	0.97	1.69	0.53
Class 3 vs Rest	62.69	85.03	122.01	62.69	1.29	2.66	4.32	1.29	1.21	3.04	4.05	1.21
Class 4 vs Rest	77.87	86.19	97.47	77.87	1.26	1.89	2.77	1.26	0.82	1.38	2.28	0.82
Class 5 vs Rest	10.94	16.14	24.70	10.94	0.35	0.43	0.53	0.35	0.65	0.93	1.30	0.65
D1 - ER+/-:												
Class 1 vs Class 2	23.07	32.48	45.26	23.07	0.63	0.89	1.14	0.68	3.29	16.28	32.71	3.29
D1 - Group:												
Class 1 vs Class 2	58.66	69.14	87.39	58.66	1.38	1.97	2.91	1.38	3.10	10.19	17.90	3.10
D2 - Type:												
Class 1 vs Rest	0.035	0.055	0.065	0.035	0.019	0.028	0.033	0.019	0.016	0.028	0.043	0.016
Class 2 vs Rest	0.054	0.070	0.084	0.054	0.027	0.067	0.178	0.027	0.020	0.072	0.272	0.020
Class 3 vs Rest	0.056	0.068	0.081	0.056	0.066	0.170	0.492	0.066	0.222	0.406	0.651	0.222

Table 13.6 shows the level of generalisation, in terms of percentage of well classified samples, obtained using the raw, and PCA/LTSA preprocessed data sets. The generalisation results are presented in terms of the mean, mode (the most frequently occurring value), min, and max obtained over the ten data subsets generated with the cross validation approach.

This table shows that the classification results obtained with the dimensionality reduction methods are generally better than the ones obtained with the raw data sets. The results obtained with the PCA preprocessed data sets are significantly higher than the ones obtained with the LTSA reduction method. However, in terms of the number of dimensions, the LTSA performance is better than the one obtained with the PCA method. For example, in the data set D1, the variance of the PCA transformed data accomplishes for 80%. In the case of the D1 cancer type classification, 90 dimensions correspond to 67% of the total variance with regards to the PCA method. Overall

slightly better results are obtained with the LTSA with respect to the raw data sets.

The best generalisation results are obtained with classes 1 and 5 of the D1 data set. These two classes have the highest number of samples compared to the other classes (see table 13.2). Therefore, more data was available to train the perceptron linear classification models. The level of generalisation obtained with the LTSA preprocessed data on class 5 is surprisingly low in comparison with both raw and PCA preprocessed data sets. This is the only class where such large differences, in terms of the level of generalisation, can be seen. This might suggest that the LTSA parameters have to be fine tuned independently for each of the classes.

All methods obtained the same maximum level of generalisation for class 1 of the D2 data set. LTSA gives higher results for class 2. The raw data provides the best maximum generalisation level for class 3. Overall, the PCA provides better results closely followed by those obtained with the LTSA method.

Table 13.6. Results obtained with the Perceptron in terms of the level of generalisation

Class	No Dim. Reduction				PCA				LTSA			
	Min	Mean	Max	Mode	Min	Mean	Max	Mode	Min	Mean	Max	Mode
D1 - Type:												
Class 1 vs Rest	70.18	74.30	79.82	75.44	72.81	76.75	79.82	75.44	59.65	74.47	84.21	77.19
Class 2 vs Rest	7.89	16.93	24.56	18.42	26.32	34.04	37.72	36.84	9.65	16.40	28.95	9.65
Class 3 vs Rest	12.28	20.26	28.95	20.18	28.95	33.86	39.47	33.33	20.18	23.86	29.82	21.93
Class 4 vs Rest	15.79	21.93	28.07	19.30	32.46	38.07	43.86	34.21	16.67	23.33	30.70	24.56
Class 5 vs Rest	75.44	83.60	89.47	86.84	77.19	83.33	89.47	80.70	19.30	22.81	26.32	21.93
D1 - ER +/-:												
Class 1 vs Class 2	58.77	65.00	73.68	66.67	59.65	68.25	79.82	71.05	43.86	52.72	60.53	43.86
D1 - Group:												
Class 1 vs Class 2	42.11	48.86	56.14	43.86	49.12	54.91	60.53	53.51	45.61	53.77	63.16	54.39
D2 - Type:												
Class 1 vs Rest	25.00	54.38	81.25	56.25	50.00	65.63	81.25	62.50	31.25	57.50	81.25	37.50
Class 2 vs Rest	18.75	32.50	56.25	25.00	31.25	44.38	56.25	37.50	18.75	40.00	62.50	37.50
Class 3 vs Rest	31.25	46.88	68.75	43.75	25.00	40.00	56.25	43.75	50.00	54.38	62.50	50.00

13.5 Conclusions

A comparison study of the performance of the linear principal component analysis and the non linear local tangent space alignment principal manifold methods was presented. Two microarray data sets were used in this study. The first data set contained five, two and two classes. The second data set contained three classes. Linear classification models were created using fully dimensional and dimensionality reduced data sets. To measure the amount of information lost with the two dimensionality reduction methods, the level of performance of each of the methods was measured in terms of level of generalisation obtained by the classification models on previously unseen data sets.

In terms of convergence time, the benefit offered by the dimensionality reduction methods is clear. Using the PCA dimensionality reduction method,

prior to the development of the classification models, is over all faster than developing the models with raw, fully dimensional, data. For the LTSA, the time benefit is less significant. Nevertheless, for training multiple classification models, the LTSA is also a time beneficial alternative. This was shown to be the case for both microarray data sets.

In terms of generalisation, the linear classification models, built using both PCA and LTSA dimensionality reduction methods, frequently outperform the ones developed using raw data. The models developed by using the PCA method, give more consistent results than the ones using the LTSA. However, for this microarray data sets, the LTSA method produces more compact reduced data sets.

In conclusion, the results obtained with this study, do not allow to clearly measure the amount of information lost by the dimensionality reduction methods. Nevertheless, the results obtained are interesting, demonstrating conclusively that the level of generalisation was better when using dimensionality reduction methods. The reason for this might be related to the level of noise in the data. Most of the variance in the original 17816 dimensional data is provided by only about 150 dimensions. Due to the the high number of irrelevant dimensions, the inherited linear separability of these data sets might hide away non linear structures in the data.

Further studies could include the use of other manifold methods (linear and non linear). The use of non linear methods for building the classification methods might also be of interest since the current linear separability of the original data might be related to the few samples in a large input space. Data sets containing more samples will probably result in better classification models.

References

1. Barker, M. and Rayens, W.: Partial least squares for discrimination. *Journal of Chemometrics*, **17**, 166–173 (2002)
2. Bollacker, K.D. and Ghosh, J.: Linear feature extractors based on mutual information. In: Proceedings of the 13th International Conference on Pattern Recognition, volume 2, pages 720–724, Vienna, Austria, (1996)
3. Elizondo, D.: Searching for linearly separable subsets using the class of linear separability method. In: IEEE-IJCNN'04, pages 955–960 (2004)
4. Elizondo, D., The linear separability problem: Some testing methods. Accepted for Publication: IEEE TNN (2006)
5. Elizondo, D., Birkenhead, R., and Taillard, E.: Generalisation and the recursive deterministic perceptron. In: IEEE IJCNN'06, (2006)
6. Dyrskjot, L., Thykjaer, T., Kruhoffer, M. et al.: Identifying distinct classes of bladder carcinoma using microarrays. *Nat Genetics* **33** (1), 90–96 (2003).
7. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Elsevier (1990)
8. Hastie, T. and Stuetzle, W.: Principal curves. *Journal of the American Statistical Association*, **84** (1989)

9. Pearson, K.: On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, **2** (6), 559–572 (1901)
10. Peyré, G.: <http://www.mathworks.com/matlabcentral/fileexchange/>. Dimension reduction toolbox (2006)
11. Castelo, R. and Roverato, A.: A robust procedure for gaussian graphical model search from microarray data with p larger than n . *Journal of Machine Learning Research*, **7**, 2621–2650 (2006)
12. Rowels, S. and Saul, L.: Non linear dimensionality reduction by locally linear embedding. *Science*, **290** (2000)
13. Rumelhart, D. E., McClelland, J. L., and the PDP Research Group: Parallel Distributed Processing, vol. 1. The MIT Press, Cambridge, MA (1986)
14. Rumelhart, D. E., McClelland, J. L., and the PDP Research Group: Parallel Distributed Processing, volume 2. The MIT Press, Cambridge, Massachusetts (1986)
15. Lambert-Lacroix, S. and Peyre, J.: Local likelihood regression in generalized linear single-index models with applications to microarray data. *Computational Statistics and Data Analysis*, **51** (3), 2091–2113 (2006)
16. Tajine, M. and Elizondo, D.: The recursive deterministic perceptron neural network. *Neural Networks*, **11**, 1571–1588 (1997)
17. Tajine, M. and Elizondo, D.: Growing methods for constructing recursive deterministic perceptron neural networks and knowledge extraction. *Artificial Intelligence*, **102**, 295–322 (1998)
18. Tajine, M. and Elizondo, D.: New methods for testing linear separability. *Neurocomputing*, **47**, 161–188 (2002)
19. Tenenbaum, J. B., de Silva, V., and Langford, J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science*, **290** (5500), 2319–2323, (2000)
20. Teng, L., Li, H., Fu, X., Chen, W., and Shen, I.: Dimension reduction of microarray data based on local tangent space alignment. In: Fourth IEEE Conference on Cognitive Informatics, pages 154–159. University of California, Irvine, USA, August (2005)
21. Tan, Y., Shi, L., Tong, W., Hwang, G.T.G., and Wang C.: Multi-class tumor classification by discriminant partial least squares using microarray gene expression data and assessment of classification models. *Computational Biology and Chemistry*, **28** (3), 235–244 (2004)
22. Wang, Y., Klijn, J.G., Zhang, Y., Sieuwerts, A.M., Look, M.P., Yang, F., Tsvetkov, D., Timmermans, M., Meijer-van Gelder, M.E., Yu, J. et al.: Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet* **365**, 671–679 (2005)
23. Zhang, Z. and Zha, H.: Principal manifolds and non-linear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, **26** (1), 313–338 (2004)

PCA and K-Means Decipher Genome

Alexander N. Gorban^{1,3} and Andrei Y. Zinov'yev^{2,3}

¹ University of Leicester, University Road, Leicester, LE1 7RH, UK,
ag153@le.ac.uk

² Institut Curie, 26, rue d'Ulm, Paris, 75248, France,
andrei.zinovyev@curie.fr

³ Institute Of Computational Modeling of Siberian Branch of Russian Academy of Science, Krasnoyarsk, Russia

Summary. In this paper, we aim to give a tutorial for undergraduate students studying statistical methods and/or bioinformatics. The students will learn how data visualization can help in genomic sequence analysis. Students start with a fragment of genetic text of a bacterial genome and analyze its structure. By means of principal component analysis they “discover” that the information in the genome is encoded by non-overlapping triplets. Next, they learn how to find gene positions. This exercise on PCA and K-Means clustering enables active study of the basic bioinformatics notions. The Appendix contains program listings that go along with this exercise.

Key words: Bioinformatics; Data visualization; Cryptography; Clustering; Principal component analysis

14.1 Introduction

When it is claimed in newspapers that a new genome is deciphered, it usually means that the sequence of the genome has been read only, so that a long sequence using four genetic letters: A, C, G and T is known. The first step in complete deciphering of the genome is identification of the positions of elementary messages in this text or *detecting genes* in the biological language. This is imperative before trying to understand what these messages mean for a living cell.

Bioinformatics – and genomic sequence analysis, in particular – is one of the hottest topics in modern science. The usefulness of statistical techniques in this field cannot be underestimated. In particular, a successful approach to the identification of gene positions is achieved by statistical analysis of genetic text composition.

In this exercise, we use Matlab to convert genetic text into a table of short word frequencies and to visualize this table using principal component analy-

sis (PCA). Students can find, by themselves, that the sequence of letters is not random and that the information in the text is encoded by non-overlapping triplets. Using the simplest K-Means clustering method from the Matlab Statistical toolbox, it is possible to detect positions of genes in the genome and even to predict their direction.

14.2 Required Materials

To follow this exercise, it is necessary to prepare a genomic sequence. We provide a fragment of the genomic sequence of *Caulobacter Crescentus*. Other sequences can be downloaded from the Genbank FTP-site [1]. Our procedures work with the files in the *Fasta* format (the corresponding files have a *.fa* extension) and are limited to analyzing fragments of 400–500 kb in length.

Five simple functions are used:

<i>LoadFreq.m</i>	loads Fasta-file into a Matlab string
<i>CalcFreq.m</i>	converts a text into a numerical table of short word frequencies
<i>PCAFreq.m</i>	visualizes a numerical table using principal component analysis
<i>ClustFreq.m</i>	is used to perform clustering with the K-Means algorithm
<i>GenBrowser.m</i>	is used to visualize the results of clustering on the text

All sequence files and the m-files should be placed into the current Matlab working folder. The sequence of Matlab commands for this exercise is the following:

```
str = LoadSeq('ccrescentus.fa');
xx1 = CalcFreq(str,1,300);
xx2 = CalcFreq(str,2,300);
xx3 = CalcFreq(str,3,300);
xx4 = CalcFreq(str,4,300);
PCAFreq(xx1);
PCAFreq(xx2);
PCAFreq(xx3);
PCAFreq(xx4);
fragn = ClustFreq(xx3,7);
GenBrowser(str,300,fragn,13000);
```

All the required materials can be downloaded from [2].

14.3 Genomic Sequence

14.3.1 Background

The information that is needed for a living cell functioning is encoded in a long molecule of DNA. It can be presented as a text with an alphabet that has only four letters A, C, G and T. The diversity of living organisms and their complex properties is hidden in their genomic sequences. One of the most exciting problems in modern science is to understand the organization of living matter by reading genomic sequences.

One distinctive message in a genomic sequence is a piece of text, called *a gene*. Genes can be oriented in the sequence in the forward and backward directions (see Fig. 14.1). This simplified picture with unbroken genes is close to reality for bacteria. In the highest organisms (humans, for example), the notion of a gene is more complex.

It was one of many great discoveries of the twentieth century that biological information is encoded in genes by means of triplets of letters, called *codons* in the biological literature. In the famous paper by Crick *et al.* [3], this fact was proven by genetic experiments carried out on bacteria mutants. In this exercise, we will analyze this by using the genetic sequence only.

In nature, there is a special mechanism that is designed to read genes. It is evident that as the information is encoded by non-overlapping triplets, it is important for this mechanism to start reading a gene without a shift, from the first letter of the first codon to the last one; otherwise, the information decoded will be completely corrupted.

An easy introduction to modern molecular biology can be found in [4].

14.3.2 Sequences for the Analysis

The work starts with a fragment of genomic sequence of the *Caulobacter Crescentus* bacterium. A short biological description of this bacterium can be found in [5]. The sequence is given as a long text file (300 kb), and the students are asked to look at the file and ensure that the text uses the alphabet of four letters (*a*, *c*, *g* and *t*) and that these letters are used without spaces. It is noticeable that, although the text seems to be random, it is well organized, but we cannot understand it without special tools. Statistical methods can help us understand the text organization.

The sequence can be loaded in the Matlab environment by *LoadSeq* function:

```
str = LoadSeq('ccrescentus.fa');
```

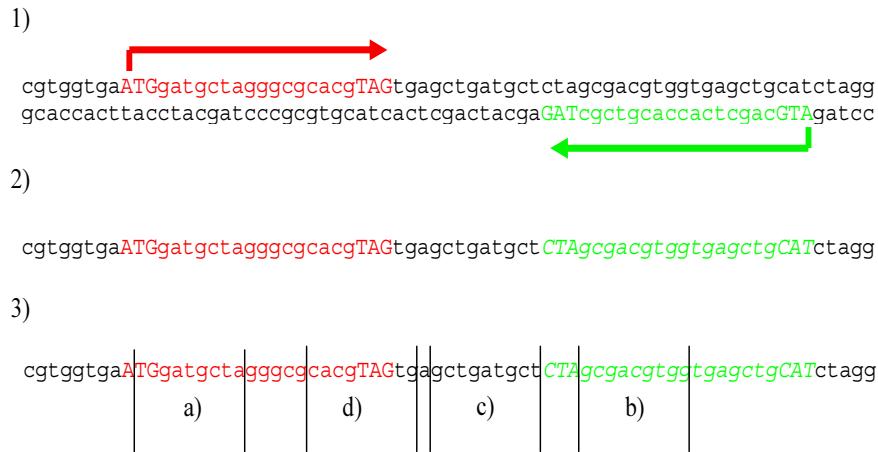


Fig. 14.1. 1) DNA can be represented as two complementary text strings. “Complementary” here means that instead of any letter “a”, “t”, “g” and “c” in one string there stands “t”, “a”, “c” and “g”, respectively, in the other string. Elementary messages or genes can be located in both strings, but in the lower one they are read from right to the left. Genes usually start with “atg” and end with “tag” or “taa” or “tga” words. 2) In databases one obtains only the upper “forward” string. Genes from the lower “backward” string can be reflected on it, but should be read in the opposite direction and changing the letters, accordingly to the complementary rules. 3) If we take a randomly chosen fragment, it can be of one of three types: a) entirely in a gene from the forward string; b) entirely in a gene from the backward string; c) entirely outside genes; d) partially in genes and partially outside genes.

14.4 Converting Text to a Numerical Table

A *word* is any continuous piece of text that contains several subsequent letters. As there are no spaces in the text, separation into words is not unique.

The method we use is as follows. We clip the whole text into fragments of 300 letters⁴ in length and calculate the frequencies of short words (of length 1–4) inside every fragment. This will give us a description of the text in the form of a numerical table. There will be four such tables for every short word length from 1 to 4.

As there are only four letters, there are four possible words of length 1 (singlets), $16 = 4^2$ possible words of length 2 (duplets), $64 = 4^3$ possible words of length 3 (triplets) and $256 = 4^4$ possible words of length 4 (quadruplets). The first table contains four columns (frequency of every singlet) and the number of rows equals the number of fragments. The second table has 16 columns and the same number of rows, and so on.

⁴ Mean gene size in bacteria is about 1000 genetic letters, the fragment length in 300 letters corresponds well to detect genes on this scale, with some resolution

To calculate the tables, students use the *CalcFreq.m* function. The first input argument for the function *CalcFreq* is the string containing the text, the second input argument is the length of the words to be counted, and the third argument is the fragment length. The output argument is the resulting table of frequencies. Students use the following set of commands to generate tables corresponding to four different word lengths:

```
xx1 = CalcFreq(str,1,300);
xx2 = CalcFreq(str,2,300);
xx3 = CalcFreq(str,3,300);
xx4 = CalcFreq(str,4,300);
```

14.5 Data Visualization

14.5.1 Visualization

PCAFreq.m function has only one input argument, the table obtained from the previous step. It produces a PCA plot for this table (PCA plot shows distribution of points on a principal plane, with the *x* axis corresponding to the projection of the point on the first principal component and the *y* axis corresponding to the projection on the second one). For an introduction to PCA, see [6].

By typing the commands below, students produce four plots for the word lengths from 1 to 4 (see Fig. 14.2).

```
PCAFreq(xx1);
PCAFreq(xx2);
PCAFreq(xx3);
PCAFreq(xx4);
```

14.5.2 Understanding Plots

The main message in these four pictures is that the genomic text contains information that is encoded by non-overlapping *triplets*, because the plot corresponding to the triplets is evidently highly structured as opposed to the pictures of singlets, duplets and quadruplets. The triplet picture evidently contains 7 clusters.

It is important to explain to students how 7-cluster structure in Fig. 14.1 occurs in nature.

Let us suppose that the text contains genes and that information is encoded by non-overlapping subsequent triplets (codons), but we do not know where the genes start and end or what the frequency distribution of codons is.

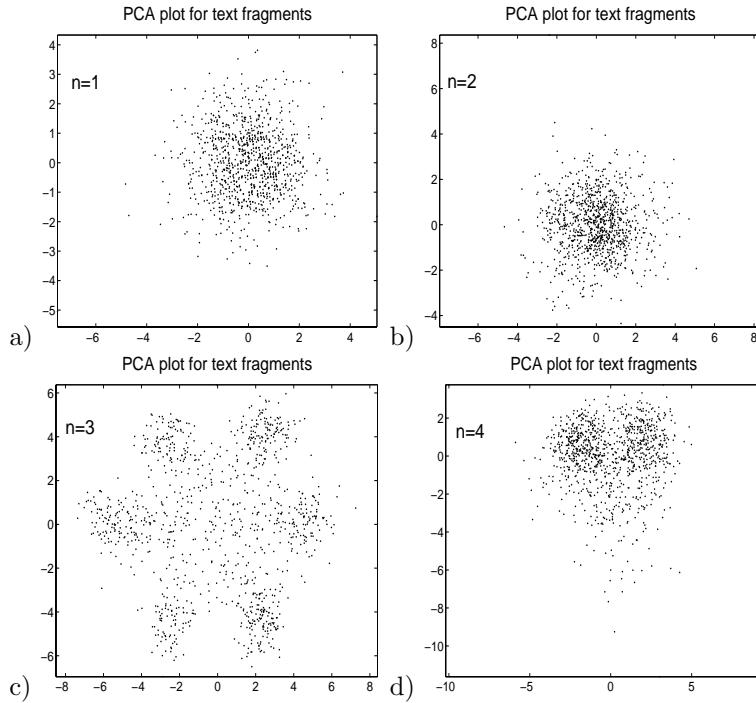


Fig. 14.2. PCA plots of word frequencies of different length. c) Shows the most structured distribution. The structure can be interpreted as the existence of a non-overlapping triplet code.

Let us blindly cut the text into fragments. Any fragment can contain: a) a piece of a gene in the forward direction; b) a piece of a gene in the backward direction; c) no genes (non-coding part); d) or a mix of coding and non-coding parts.

Consider the first case (a). The fragment can overlap with a gene in three possible ways, with three possible shifts ($\text{mod}3$) of the first letter of the fragment with respect to the start of the gene. If we enumerate the letters in the gene from the first one, 1-2-3-4-..., then the first letter of the fragment can be in the sequence 1-4-7-10-... ($=1(\text{mod}(3))$) (a “correct” shift), the sequence 2-5-8-11-... ($=2(\text{mod}(3))$), or 3-6-9-12-... ($=0(\text{mod}(3))$). If we start to read the information triplet by triplet, starting from the first letter of the fragment, we can read the gene correctly only if the fragment overlaps with it with a correct shift (see Fig. 14.1). In general, if the start of the fragment is not chosen deliberately, then we can read the gene in three possible ways. Therefore, this first case (a) generates three possible frequency distributions, each one of which is “shifted” in relation to another.

The second case (b) is analogous and also gives three possible triplet distributions. They are not independent of the ones obtained in the first cases

for the following reason. The difference is the triplets are read “from the end to the beginning” which produces a kind of mirror reflection of the triplet distributions from the first case (a).

The third case (c) produces only one distribution, which is symmetrical with respect to the ‘shifts’ (or rotations) in the first two cases, and there is a hypothesis that this is a result of genomic sequence evolution. This can be explained as follows.

Vitality of a bacterium depends on the correct functioning of all biological mechanisms. These mechanisms are encoded in genes, and if something wrong happens with gene sequences (for example there is an error when DNA is duplicated), then the organism risks becoming non-vital. Nothing is perfect in our world and the errors happen all the time, including during DNA duplication. These errors are called *mutations*.

The most dangerous mutations are those that change the reading frame, i.e. letter deletions or insertions. If such a mutation happens inside a gene sequence, the rest of the gene becomes corrupted: the reading mechanism (which reads the triplets one by one and does not know about the mutation) will read it with a shift. Because of this the organisms with such mutations often die before producing offspring. On the other hand, if such a mutation happens in the non-coding part (where no genes are present) this does not lead to a serious problem, and the organism produces offspring. Therefore, such mutations are constantly accumulated in the non-coding part and three shifted triplet distributions are mixed into one. The fourth case (d) also produces a mix of triplet distributions.

As a result, we have three distributions for the first case (a), three for the second case (b) and one, symmetrical distribution for the ‘non-coding’ fragments (third case (c)). Because of natural statistical deviations and other reasons, we have 7 clusters of points in the multidimensional space of triplet frequencies.

For more illustrative material see [7, 8, 11, 15, 12].

14.6 Clustering and Visualizing Results

The next step is clustering the fragments into 7 clusters. This can be explained to the students that as a classification of fragments of text by similarity in their triplet distributions. This is an unsupervised classification (the cluster centers are not known in advance).

Clustering is performed by the K-Means algorithm using the Matlab Statistical toolbox. It is implemented in the *ClustFreq.m* file. The first argument is the frequency table name and the second argument is the number of clusters proposed. As we visually identified 7 clusters, in this activity we put 7 as the value of the second argument:

```
fragn = ClustFreq(xx3,7);
```

The function assigns different colors onto the cluster points. The cluster that is the closest to the center of the picture is automatically colored in black (see Fig. 14.3).

After clustering, every fragment of the text is assigned a cluster label (a color). The *GenBrowser* function puts this color back to the text and then visualizes it. The input arguments of this function are a string with the genetic text, the fragment size, a vector with cluster labels and a position in the genetic text to read from (we use a value 13000, which can be changed for any other position):

```
GenBrowser(str,300,fragn,13000);
```

This function implements a simple *genome browser* (a program for visualizing genome sequence together with other properties) with information about gene positions.

It is explained to the students that clustering of fragments corresponds to segmentation of the whole sequence into homogeneous parts. The homogeneity is understood as similarity of the short word frequency distributions for the fragments of the same cluster (class). For example, for the following text

```
aaaaaaaaatataaaaaattttatTTTttttttgggggggggaagaggggccccgcctcccccc
```

one can try to apply the frequency dictionary of length 1 for a fragment size around 5-10 to separate the text into four homogeneous parts.

14.7 Task List and Further Information

Interested students can continue this activity. They can modify the Matlab functions in order to solve the following problems (the first three of them are rather difficult and require programming):

Determine a cluster corresponding to the correct shift. As it was explained in the “Understanding plots” section, some fragments overlap with genes in such a way that the information can be read with the correct shift, whereas others contain the “shifted” information. The problem is to detect the cluster corresponding to the correct shift. To give a hint, we can say that the correct triplet distribution (probably) will contain the lowest frequency of the stop codons TAA, TAG and TGA (see [16, 17]). Stop codon can appear only once in a gene because it terminates its transcription.

Measure information content for every phase. The information of a triplet distribution with respect to a letter distribution is $I = \sum_{ijk} f_{ijk} \ln \frac{f_{ijk}}{p_i p_j p_k}$, where p_i is a frequency of letter i (a genetic text is characterized by four such frequencies), and f_{ijk} is the frequency of triplet ijk . Each cluster is a collection of fragments. Each fragment F can be divided on triplets starting

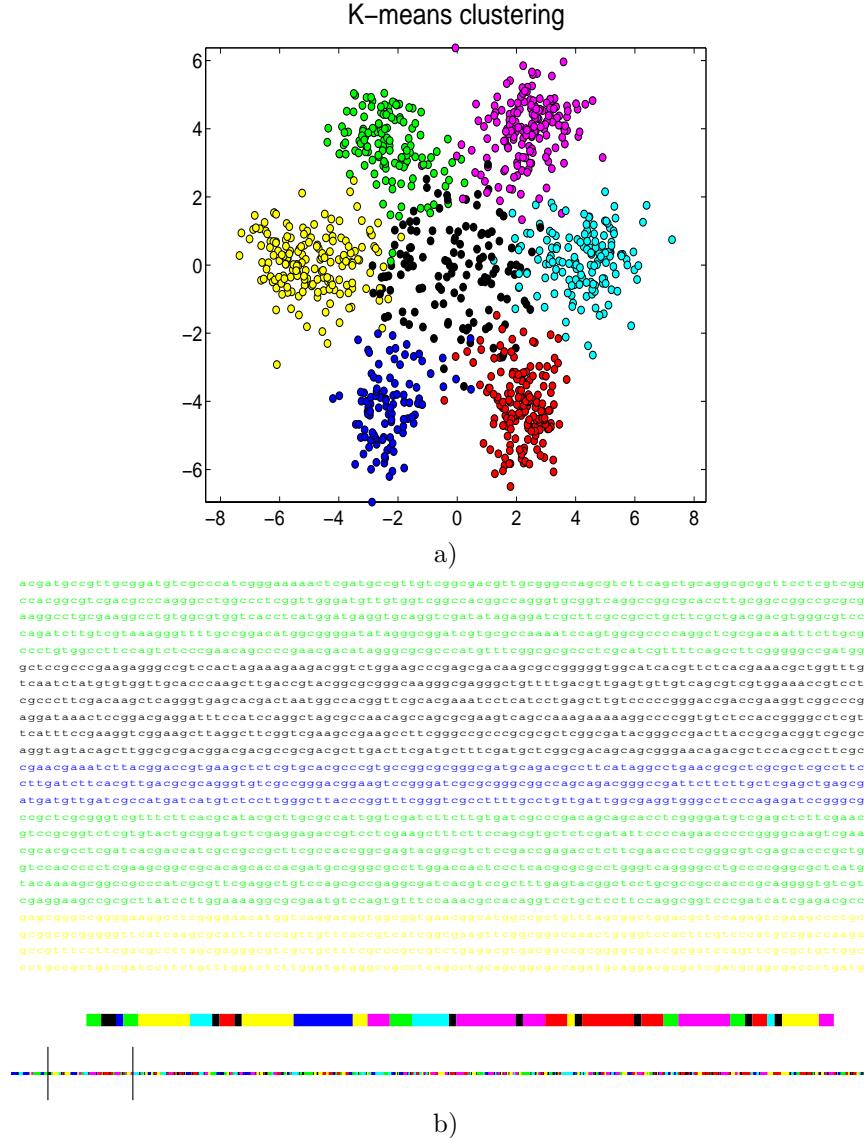


Fig. 14.3. K-Means clustering of triplet frequencies (a) and visualizing the clustering results onto the text (b). There are three scales in the browser. The bottom color code represents the genetic text as a whole. The second line shows colors of 100 fragments starting from a position in the text specified as an argument of the *GenBrowser.m* function. The letter color code shows 2400 symbols, starting from the same position. The black color corresponds to the fragments with non-coding information (the central cluster); other colors correspond to locations of coding information in different directions and with different shifts with respect to randomly chosen division of the text into fragments.

from the first letter. We can calculate the information value of this triplet distribution $I(F)$ for each afragment F . Is the information of fragments in the cluster with a correct shift significantly different from the information of fragments in other clusters? Is the *mean* information value in the cluster with a correct shift significantly different from the *mean* information value of fragments in other clusters? Could you verify the hypothesis that in the cluster with a correct shift the mean information value is higher than in other clusters?

Increase resolution of determining gene positions. In this exercise, we use the same set of fragments to calculate the cluster centers and to annotate the genome. As a result the gene positions are determined with precision that is equal to the fragment size. In fact, cluster centers can be calculated with non-overlapping fragments and then the genome can be scanned again with a sliding window (this will produce a set of overlapping fragments). Following this, triplet frequencies for each window can be calculated and the corresponding cluster in the 64-dimensional space can be determined. The position in the text is assigned a color corresponding to the content of the fragment centered in this position. For further details of this process, see [9, 10, 11, 13].

Precise start and end positions of genes. The biological mechanism for reading genes (the polymerase molecule) identifies the beginning and the end of a gene using special signals, known as specialized codons. Therefore, almost all genes start with “ATG” start codon and end with “TAG”, “TAA” or “TGA” stop codons. Try to use this information to find the beginning and end of every gene.

Play with natural texts. The *CalcFreq* function is not designed specifically for the four-letter alphabet text of genome sequences; it can work with any text. For natural text, it is also possible to construct local frequency dictionaries and segment it into “homogeneous” (in short word frequencies) parts. But be careful with longer frequency dictionaries, for an English text one has $(26 + 1)^2 = 729$ possible duplets (including space as an extra letter)!

Students can also look at visualizations of 143 bacterial genomic sequences at [14]. All possible types of the 7-cluster structure have been described in [15]. Nonlinear principal manifolds were utilized for visualization of the 7-cluster structure in [17]. Principal trees are applied for visualization of the same structure in [18].

14.8 Conclusion

In this exercise on applying PCA and K -means to the analysis of a genomic sequence, the students learn basic bioinformatics notions and train to apply PCA to the visualization of local frequency dictionaries in genetic text.

References

1. Genbank FTP-site: <ftp://ftp.ncbi.nih.gov/genbank/genomes>
2. An http-folder with all materials required for the tutorial: <http://www.ihes.fr/~zinovyev/pcadg/>
3. Crick, F.H.C., Barnett, L., Brenner, S., and Watts-Tobin, R.J.: General nature of the genetic code for proteins. *Nature*, **192**, 1227–1232 (1961)
4. Clark, D. and Russel, L.: Molecular Biology Made Simple and Fun. Cache River Press (2000)
5. Caulobacter crescentus short introduction at <http://caulo.stanford.edu/caulo/> .
6. Jackson, J.: A User's Guide to Principal Components (Wiley Series in Probability and Statistics). Wiley-Interscience, (2003)
7. Zinovyev A.: Hierarchical Cluster Structures and Symmetries in Genomic Sequences. Colloquium talk at the Centre for Mathematical Modelling, University of Leicester. December, (2004) (PowerPoint presentation at <http://www.ihes.fr/~zinovyev/presentations/7clusters.ppt>)
8. Zinovyev, A.: Visualizing the spatial structure of triplet distributions in genetic texts. HES Preprint, M/02/28 (2003) Online: <http://www.ihes.fr/PREPRINTS/M02/Resu/resu-M02-28.html>
9. Gorban, A.N., Zinovyev, A.Yu., and Popova, T.G.: Statistical approaches to the automated gene identification without teacher, Institut des Hautes Etudes Scientifiques. IHES Preprint, M/01/34 (2001) Online: <http://www.ihes.fr> web-site. (See also e-print: <http://arxiv.org/abs/physics/0108016>)
10. Zinovyev, A.Yu., Gorban, A.N., and Popova, T.G.: Self-Organizing Approach for Automated Gene Identification. *Open Systems and Information Dynamics*, **10**(4), 321–333 (2003)
11. Gorban, A., Zinovyev, A., and Popova, T.: Seven clusters in genomic triplet distributions In Silico Biology, **3** 0039 (2003) (Online: <http://arxiv.org/abs/cond-mat/0305681> and <http://cogprints.ecs.soton.ac.uk/archive/00003077/>)
12. Gorban, A.N., Popova, T.G., and Zinovyev, A.Yu.: Codon usage trajectories and 7-cluster structure of 143 complete bacterial genomic sequences. *Physica A*, **353**, 365-387 (2005)
13. Ou, H.Y., Guo, F.B., and Zhang, C.T.: Analysis of nucleotide distribution in the genome of *Streptomyces coelicolor* A3(2) using the Z curve method, *FEBS Lett.* **540** (1-3), 188–194 (2003)
14. Cluster structures in genomic word frequency distributions. Web-site with supplementary materials. <http://www.ihes.fr/~zinovyev/7clusters/index.htm>
15. Gorban, A.N., Zinovyev, A.Yu., and Popova, T.G.: Four basic symmetry types in the universal 7-cluster structure of 143 complete bacterial genomic sequences. *In Silico Biology* **5** (2005) 0025. On-line: <http://www.bioinfo.de/isb/2005/05/0025/>
16. Staden, R. and McLachlan, A.D.: Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Res* **10** (1), 141-56 (1982)
17. Gorban, A.N., Zinovyev, A.Y., and Wunsch, D.C.: Application of The Method of Elastic Maps In Analysis of Genetic Texts, In Proceedings of International Joint Conference on Neural Networks (IJCNN'03), Portland, Oregon (2003)
18. Gorban, A.N., Sumner, N.R., and Zinovyev,A.Y.: Elastic maps and nets for approximating principal manifolds and their application to microarray data visualization, In this book.

Appendix. Program listings

Function LoadSeq Listing

```

function str=LoadSeq(fafile)
fid = fopen(fafile); i=1; str = '';
disp('Reading fasta-file...');

while 1
    if round(i/200)==i/200
        disp(strcat(int2str(i), ' lines'));
    end
    tline = fgetl(fid);
    if ischar(tline), break, end;
    if(size(tline) ==0)
        if(strcmp(tline(1), '>')==0)
            str = strcat(str,tline);
        end;
    end;
    i=i+1;
end
nn = size(str); n = nn(2);
disp(strcat('Length of the string: ',int2str(n)));

```

Function PCAFreq Listing

```

function PCAFreq(xx)
% standard normalization
nn = size(xx); n = nn(1); mn = mean(xx);
mas = xx - repmat(mn,n,1); stdr = std(mas);
mas = mas./repmat(stdr,n,1);
% creating PCA plot
[pc,dat] = princomp(mas);
plot(dat(:,1),dat(:,2),'k.');
hold on;
set(gca,'FontSize',16);
axis equal;
title('PCA plot for text fragments','FontSize',22);
set(gcf,'Position',[232 256 461 422]);
hold off;

```

Function CalcFreq Listing

```

function xx=CalcFreq(str,len,wid)
disp('Cutting in fragments...');

i=1; k=1;nn = size(str);
while i+wid<nn(2)
    if round(k/200)==k/200
        disp(strcat(int2str(k), ' fragments'));
    end
    frag = str(i:i+wid-1); vf(k) = calcf(frag,len);
    i = i+wid; k=k+1;
end
disp('Merging into table...');

names = java.util.Vector; n = 0;
for i=1:size(vf)
    if size(vf(i)) ==0
        keys = vf(i).keys;
        while keys.hasMoreElements
            key = keys.nextElement;
            if names.indexOf(key)==-1
                names.add(key);
            end
        end, n=n+1; end, end
xx = zeros(n,size(names));
for i=1:size(vf)
    if size(vf(i)) ==0
        if round(i/200)==i/200
            disp(strcat(int2str(i), ' points'));
        end
        for j=1:size(names)
            xx(i,j) = getwf(names.elementAt(j-1),vf(i));
        end, end, end

function vf=calcf(str,num)
vf = java.util.Hashtable; i = 1; nn = size(str);
while i+num<nn(2)
    wrd = str(i:i+num-1); i = i+num; addwf(wrd,vf,1);
end

function addwf(word,hash,fr)
wf = hash.get(word);
if size(wf)==0 hash.put(word,fr); else
    hash.put(word,fr+wf); end

function fr=getwf(word,hash)
wf = hash.get(word);
if size(wf)==0 r=0; else fr=wf; end

```

Function ClustFreq Listing

```

function fragn = ClustFreq(xx,k)
% centralization and normalization
nn = size(xx); n = nn(1)
mn = mean(xx);
mas = xx - repmat(mn,n,1);
stdr = std(mas);
mas = mas./repmat(stdr,n,1);
% calculating principal components
[pc,dat] = princomp(mas);
% k-means clustering
[fragn,C] = kmeans(mas,k);
% projecting cluster centers into the PCA basis
XTP = C; temp = size(XTP); nums = temp(1);
X1c = XTP-repmat(mn,nums,1);
X1r = X1c./repmat(stdr,nums,1);
X1P = pc'*X1r'; X1P = X1P';
% marking the central cluster black
cnames = ['k','r','g','b','m','c','y'];
for i=1:k no(i) = norm(X1P(i,1:3)); end
[m,mi] = min(no);
for i=1:size(fragn)
    if fragn(i)==mi fragn(i)=1;
    elseif fragn(i)==1 fragn(i)=mi; end
end
% plotting the result using PCA
for i=1:n
    plot(dat(i,1),dat(i,2),'ko',
        'MarkerEdgeColor',[0 0 0],'MarkerFaceColor',
        cnames(fragn(i)));
    hold on;
end
set(gca,'FontSize',16); axis equal;
title('K-means clustering','FontSize',22);
set(gcf,'Position',[232 256 461 422]);

```

Function GenBrowser Listing

```

function GenBrowser(str,wid,fragn,startp)
% we will show 100 fragments in the detailed view
endp = startp+wid*100; nn = size(fragn); n = nn(1);
xr1 = startp/(n*wid); xr2 = endp/(n*wid);
cnames = ['k','r','g','b','m','c','y'];
subplot('Position',[0 0 1 0.1]);
for i=1:size(fragn)
    plot(i/n,0,strcat(cnames(fragn(i)),',s'),'MarkerSize',2);
    hold on;
end
plot([xr1 xr1],[-1 1],',k'); hold on;
plot([xr2 xr2],[-1 1],',k'); axis off;
subplot('Position',[0 0.1 1 0.1]);
for i=floor(startp/wid)+1:floor(endp/wid)+1
    plot([(i-0.5)*wid (i+0.5)*wid],[0 0],
        strcat(cnames(fragn(i)),'-'),'LineWidth',5);
    hold on;
end
axis off;
subplot('Position',[0 0.25 0.98 0.75]);
xlim([0,1]); ylim([0,1]); twid = 100; nlin = 24;k=startp;
for j=1:nlin
    for i=1:twid
        col = cnames(fragn(floor(k/wid)+1));
        h=text(i/twid,1-j/nlin,str(k),'FontSize',8,
            'FontName','FixedWidth');
        set(h,'Color',col);
        k=k+1;
    end end
    axis off;
    set(gcf,'Position',[64 356 879 195]);

```

Index

- adaptive strategies, 107
- additive decomposition, 153
- admissible kernel, 270
- Arabidopsis thaliana*, 61
- autoassociative network, 24, 48
- backward propagation, 53
- blind inverse problem, 53
- break-type adaptive strategy, 108
- cancer, 116, 144, 216, 288
- capacity, 267
- cluster-wise PCA, 183
- code book, 78
- codon, 234
- compactness, 124
- contrast function, 276
- cost function, 78
- cross validation, 305
- cross-validation, 8
- cumulant matrix, 281
- Delicado's algorithm, 189
- diffusion distance, 245
- diffusion map, 245
- dimensionality reduction, 49
- discretization, 268
- elastic energy, 101
- elastic graph, 101, 225
- elastic map, 110
- elastic net, 99
- elasticity coefficients, 103
- electrostatics, 267
- entropy, 278
- Ercherichia coli*, 236
- expression mode, 289
- extraction function, 48
- factorization, 228
- feature extraction, 49
- Fokker-Planck operator, 249
- forward propagation, 52
- Fréchet mean, 98
- fuzzy clustering, 158
- Gaia, 197
- galaxy, 195
- Genbank, 314
- genome, 234, 315
- graph grammar, 227
- grid approximation, 101
- growing flag, 108
- growing lump, 108
- harmonic means, 135
- harmonic topographic map, 142
- Hartigan's rule, 154
- Hebbian learning, 72
- hidden factors, 153
- iK-Means, 154
- independent components, 276
- inverse problem, 53
- Jacobi rotation, 287
- kernel estimates, 213

- kurtosis, 280
- Langevin equation, 250
- Laplace-Beltrami operator, 250
- linear auto-associative model, 212
- linear separability, 301
- local PCA, 32, 183
- local principal curve, 182
- local tangent space alignment, 300
- Matlab, 314
- microarray, 115, 144, 216, 275, 288, 297
- missing data, 56, 106
- molecular surface, 114
- multi-layer perceptron, 48
- multidimensional scaling, 82
- mutations, 235, 319
- mutual information, 278
- natural PCA, 117
- natural principal components, 121
- negentropy, 278
- neighbourhood function, 76
- neural gas, 136
- nonlinear auto-associative model, 212
- nonlinear PCA (NLPCA), VIII
- dynamic extensions, 37
 - hierarchical, 49
 - inverse, 53
- nonlinearity test, 6
- optimal transformation, 228
- Pearson correlation, 279
- penalty, 102, 110
- perceptron, 303
- phase transition, 262
- Plasmodium falciparum, 47
- prewhitening, 277
- principal component regression, 191
- principal curve, VIII, 16, 187
- principal flag, 108
- principal manifold, VIII, 85, 99
- principal object, 99
- principal tree, 229
- product of experts, 141
- production rule, 227
- projection pursuit, 208
- Rand index, 163
- reconstruction error, 55
- relaxation time, 246
- reproducing kernel, 267
- rib, 101
- Riesz kernel, 268
- Sammon stress, 82
- scalability, 49
- self-consistent curve, 21
- set learning, 270
- singular value decomposition, 5, 153
- skeleton, 109
- softening, 105
- SOM algorithm, 76
- spectral embedding, 251
- splitting algorithm, 226
- stiffness matrix, 112
- swiss roll, 252
- tissue, 116, 238
- topographic map, 133
- topographic neural gas, 144
- Voronoi cell, 78, 103

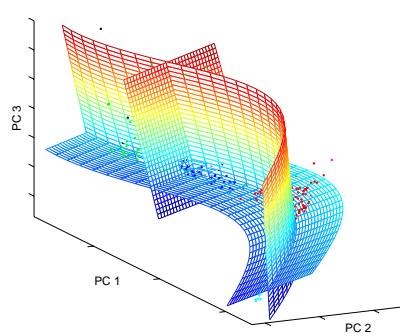
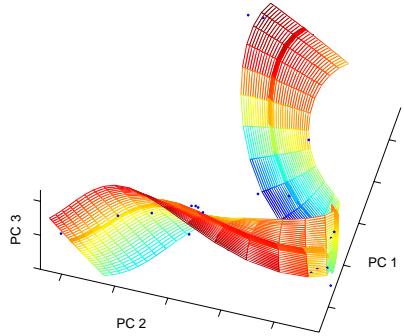
Astronomical data**EMG data**

Fig. 2.8. Hierarchical nonlinear PCA is applied to a star spectral data set and to electromyographic (EMG) recordings. Both data sets show a clear nonlinear behaviour. The first three nonlinear components are visualised in the space of the first three PCA components. The grids represent the new coordinate system of the component space. Each grid is spanned by two of the three components while the third is set to zero.

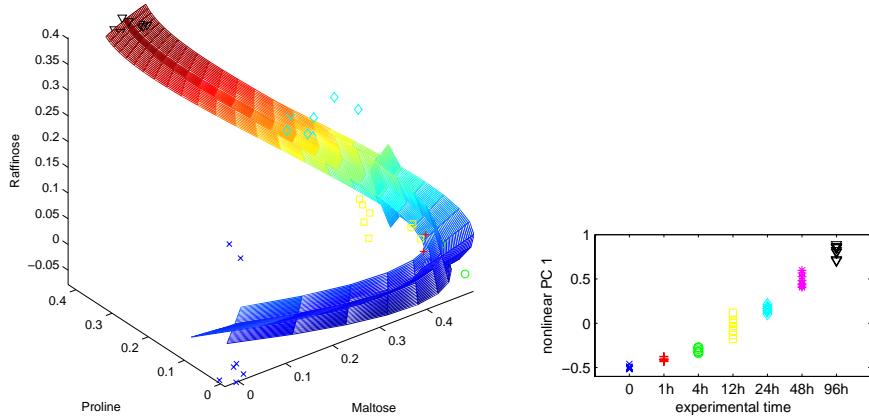


Fig. 2.9. Cold stress metabolite data. Left: The first three extracted nonlinear components are plotted into the data space given by the top three metabolites of highest variance. The grid represents the new coordinate system of the component space. The principal curvature given by the first nonlinear component represents the trajectory over time in the cold stress experiment as shown on the right by plotting the first component against the experimental time.

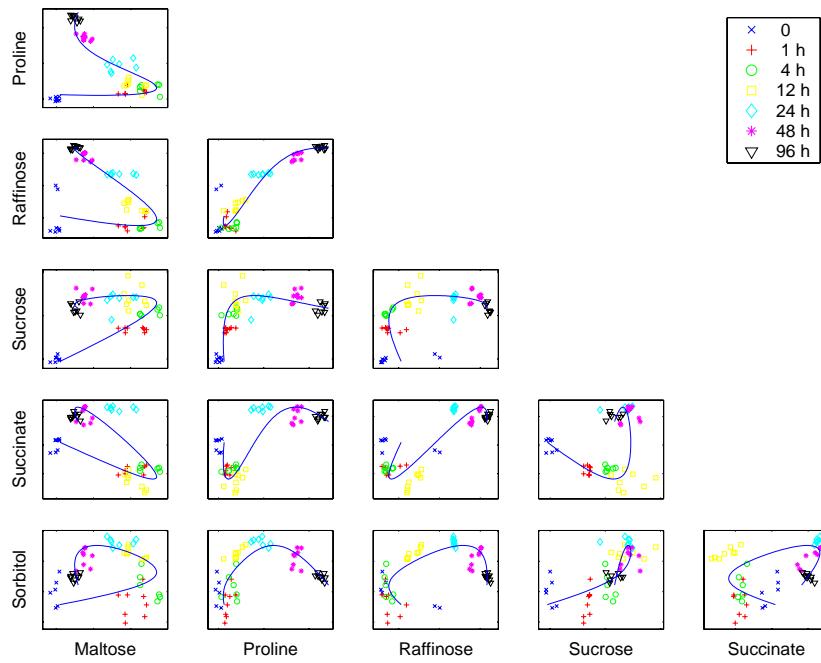


Fig. 2.10. Time trajectory. Scatter plots of pair-wise metabolite combinations of the top six metabolites of highest relative variance. The extracted time component (nonlinear PC 1), marked by a line, shows a strong nonlinear behaviour.

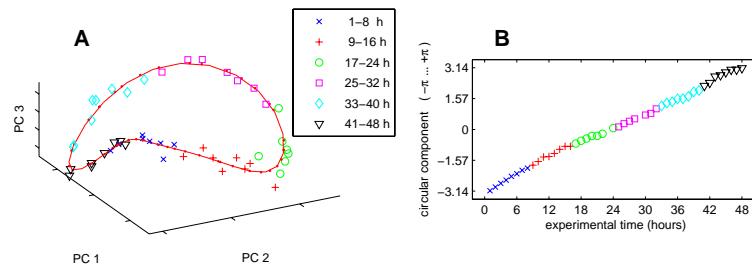


Fig. 2.11. Cyclic gene expression data. (A) Circular PCA describes the circular structure of the data by a closed curve – the circular component. The curve represents a one-dimensional component space as a subspace of a 5,800 dimensional data space. Visualised is the component curve in the reduced three dimensional subspace given by the first three components of standard (linear) PCA.
(B) The circular component (corrected by an angular shift) is plotted against the original experimental time. It shows that the main curvature, given by the circular component, explains the trajectory of the IDC over 48 hours.

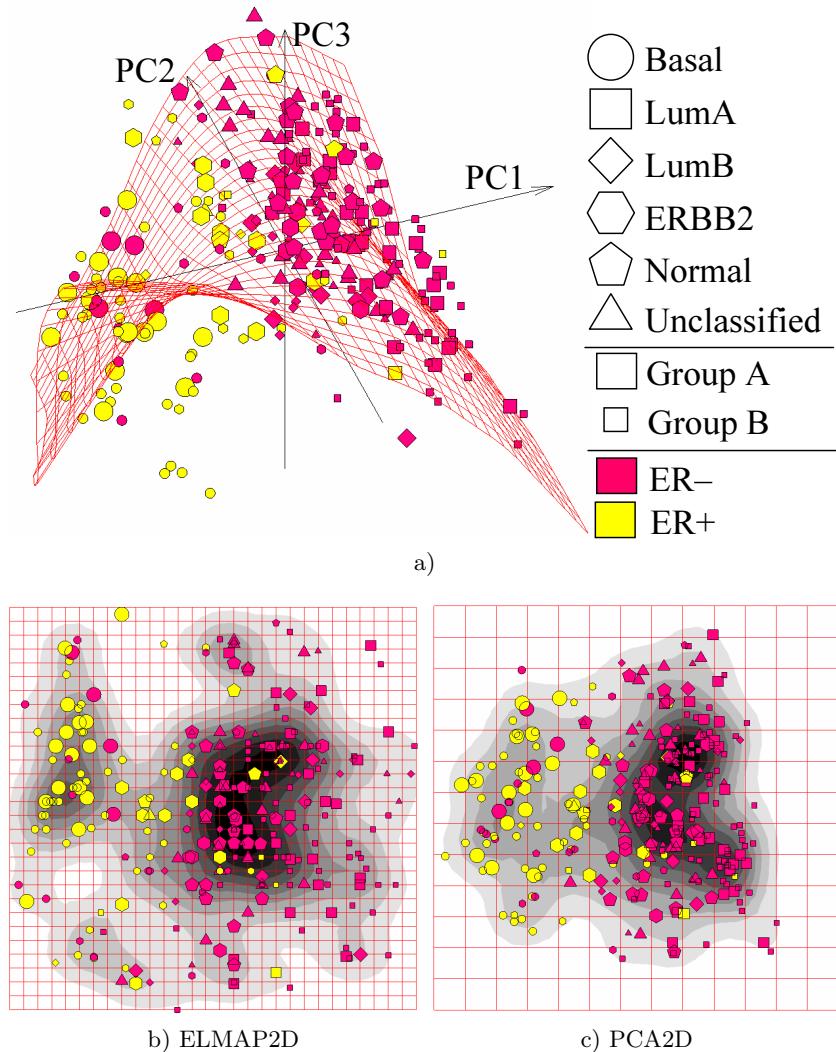


Fig. 4.7. Visualization of Dataset I (breast cancer dataset) using elastic maps. *Ab initio* classifications are shown using points size (ER), shape (GROUP) and color (TYPE). a) configuration of nodes in the three-dimensional principal linear manifold. One clear feature is that the dataset is curved such that it can not be mapped adequately on a two-dimensional principal plane. b) the distribution of points in the internal non-linear manifold coordinates (ELMAP2D) is shown together with an estimation of the two-dimensional density of points. c) the same as b) but for the linear two-dimensional manifold (PCA2D). One can notice that the “basal” breast cancer subtype is visualized more adequately with ELMAP2D and some features of the distribution become better resolved in comparison to PCA2D.

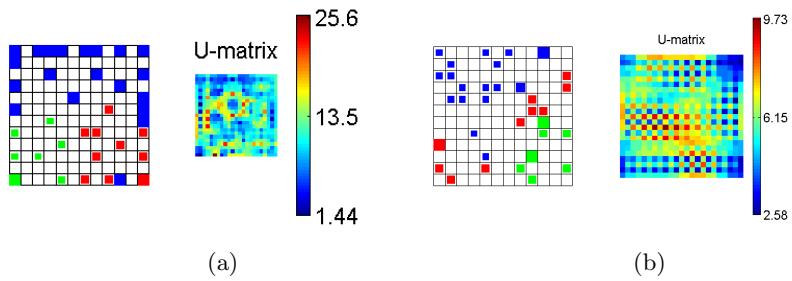


Fig. 5.5. Hit histogram and U-matrix for SOM (a) and ToPoE (b).

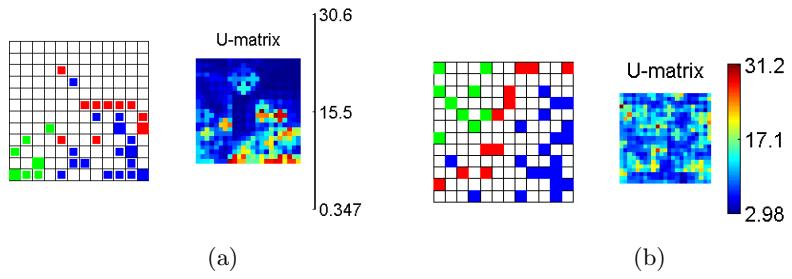


Fig. 5.6. Hit histogram and U-matrix for HaToM (a) and ToNeGas (b).

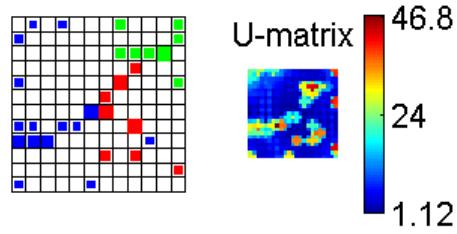


Fig. 5.7. Hit histogram and U-matrix for IKToM.

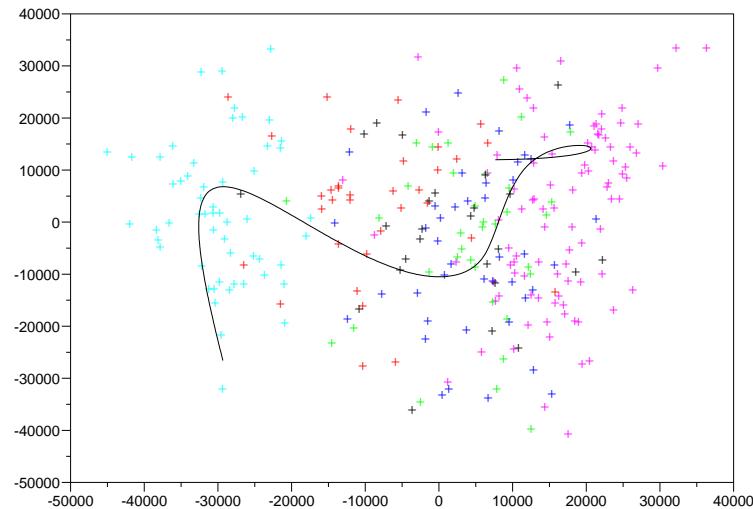


Fig. 8.4. One-dimensional manifold estimated on a real dataset with the auto-associative models approach and projected on the principal plane.

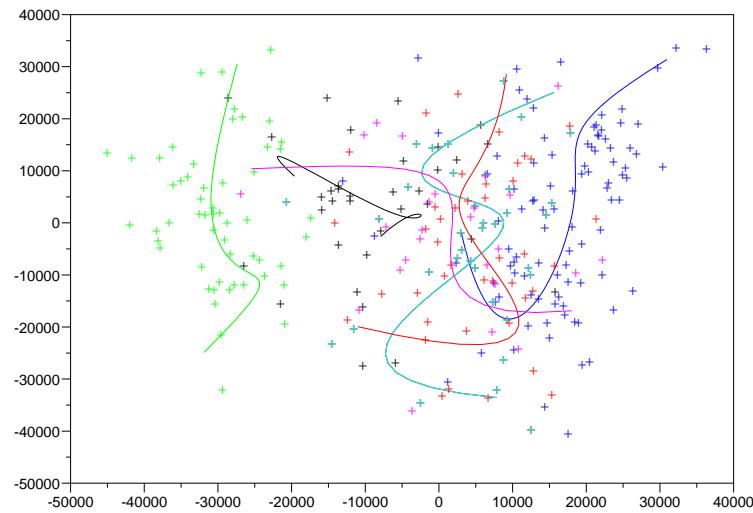


Fig. 8.5. One-dimensional manifolds estimated on each type of cancer of the real dataset with the auto-associative models approach, and projected on the principal plane.

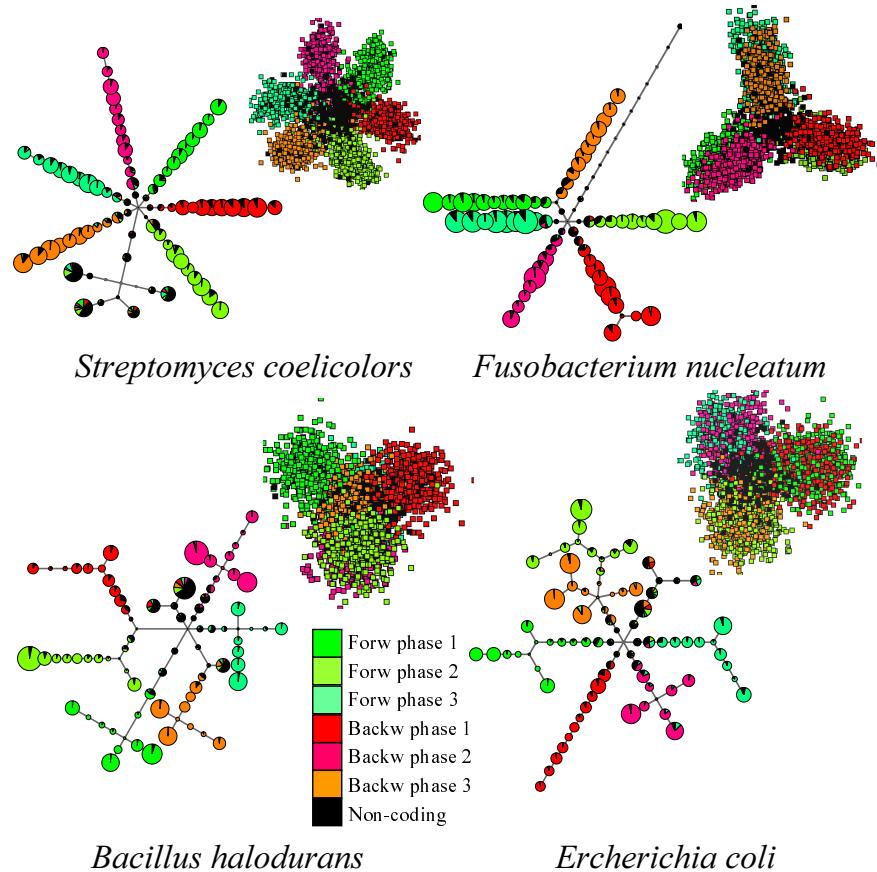


Fig. 9.7. Seven cluster structures presented for 4 selected genomes. A genome is represented as a collection of points (text fragments represented by their triplet frequencies) in a multidimensional space. Color codes (color online) correspond to 6 possible frameshifts when a random fragment overlaps with a gene (3 in the forward and 3 in the backward direction of the gene), and the black color corresponds to non-coding regions. For every genome a principal tree (“metro map” layout) is shown together with 2D PCA projection of the data distribution. Note that the clusters that are mixed in the PCA plot for *Escherichia coli* (they remain mixed in 3D PCA as well) are well separated on the “metro map”.

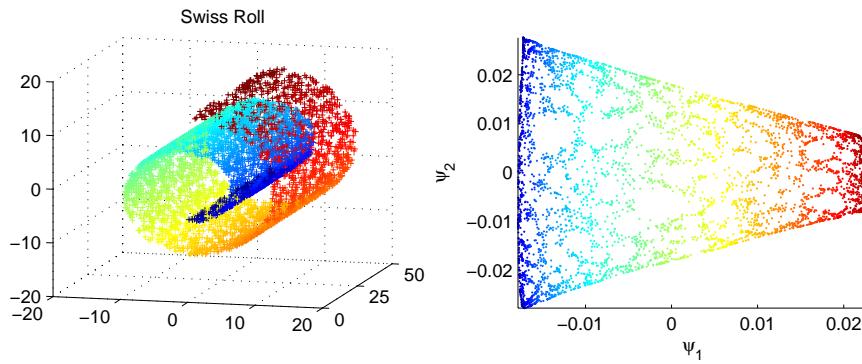


Fig. 10.3. 5000 points sampled from a wide swiss roll and embedding into the first two diffusion map coordinates.

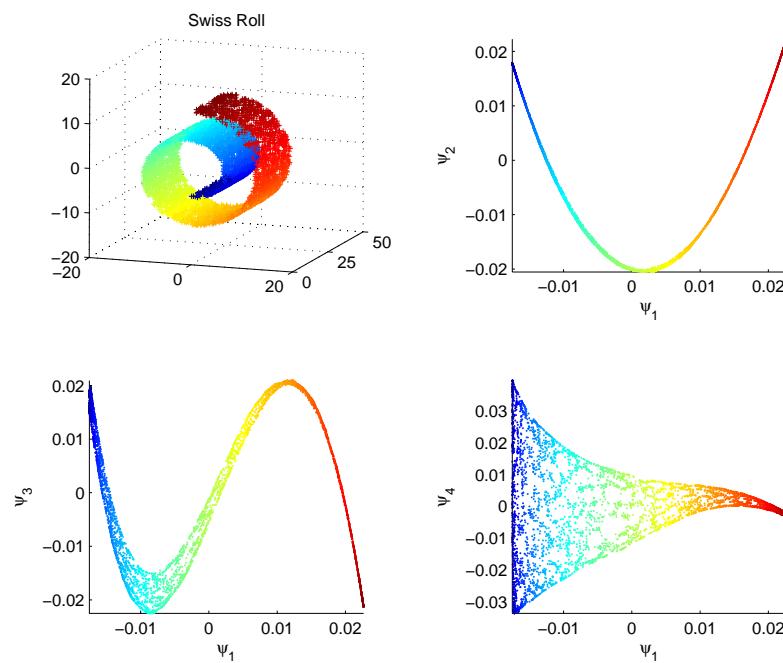


Fig. 10.4. 5000 points sampled from a narrow swiss roll and embedding into various diffusion map coordinates.

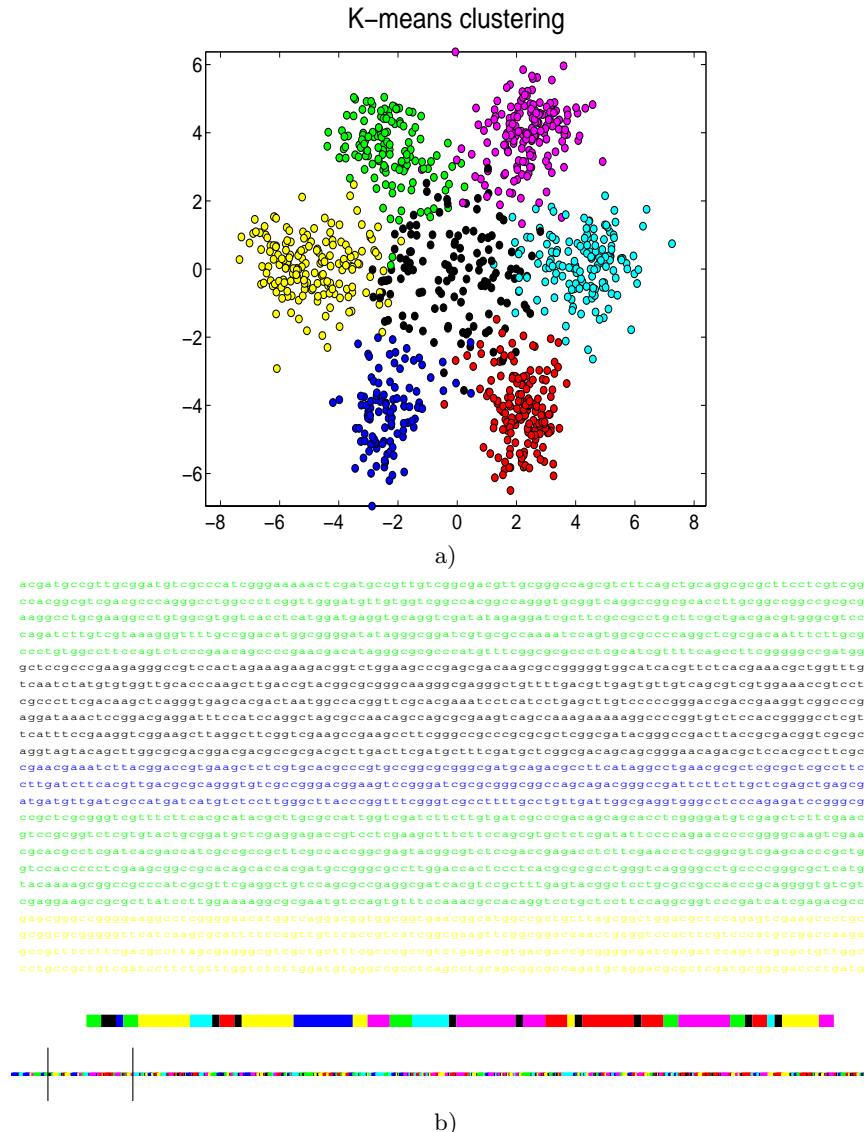


Fig. 14.3. *K*-Means clustering of triplet frequencies (a) and visualizing the clustering results onto the text (b). There are three scales in the browser. The bottom color code represents the genetic text as a whole. The second line shows colors of 100 fragments starting from a position in the text specified as an argument of the *GenBrowser.m* function. The letter color code shows 2400 symbols, starting from the same position. The black color corresponds to the fragments with non-coding information (the central cluster); other colors correspond to locations of coding information in different directions and with different shifts with respect to randomly chosen division of the text into fragments.