A Review of Dimension Reduction Techniques*

Miguel Á. Carreira-Perpiñán

Technical Report CS–96–09
Dept. of Computer Science
University of Sheffield
M.Carreira@dcs.shef.ac.uk

January 27, 1997

Abstract

The problem of dimension reduction is introduced as a way to overcome the curse of the dimensionality when dealing with vector data in high-dimensional spaces and as a modelling tool for such data. It is defined as the search for a low-dimensional manifold that embeds the high-dimensional data. A classification of dimension reduction problems is proposed.

A survey of several techniques for dimension reduction is given, including principal component analysis, projection pursuit and projection pursuit regression, principal curves and methods based on topologically continuous maps, such as Kohonen's maps or the generalised topographic mapping. Neural network implementations for several of these techniques are also reviewed, such as the projection pursuit learning network and the BCM neuron with an objective function.

Several appendices complement the mathematical treatment of the main text.

Contents

*This work has been partially funded by the Spanish Ministry of Education.

List of Figures

List of Tables

1 Introduction to the Problem of Dimension Reduction

1.1 Motivation

Consider an application in which a system processes data (speech signals, images or patterns in general) in the form of a collection of real-valued vectors. Suppose that the system is only effective if the dimension of each individual vector —the number of components— is not too *high*, where *high* depends on the particular application. The problem of dimension reduction appears when the data are in fact of a higher dimension than tolerated. For example, take the following typical cases:

- A face recognition/classification system based on $m \times n$ greyscale images which, by row concatenation, can be transformed into mn-dimensional real vectors. In practice, one could have images of $m = n = 256$, or 65536-dimensional vectors; if, say, a multilayer perceptron was to be used as the classification system, the number of weights would be exceedingly large. Therefore we need to reduce the dimension. A crude solution would be to simply scale down the images to a manageable size. More elaborate approaches exist, e.g. in [31], a first neural network is used to reduce the vector dimension (which actually performs a principal component analysis of the training data) from the original dimension of $63 \times 61 = 3843$ to 80 components and a second one to actually perform the classification.

- A statistical analysis of a multivariate population. Typically there will be a few variables and the analyst is interested in finding clusters or other structure of the population and/or interpreting the variables. To that aim, it is quite convenient to visualise the data, but this will not be reasonably possible for more than 3 dimensions[1]. For example, in [73] data for the three species of flea-beetles *Ch. concinna*, *Ch. heptapotamica* and *Ch. heikertingeri* are given. For each individual, 6 measurements were taken, including head width, front angle of the aedeagus and others. In this case, reduction to a two- or three-dimensional space by projection pursuit techniques can easily show the clustered structure of the data.

- A more straightforward example is simply the estimation of a function of several variables from a finite sample. Due to the curse of the dimensionality (see section 1.4), we can greatly reduce the size of the sample by reducing the number of variables, i.e. the dimension of the data.

Therefore, in a number of occasions it can be useful or even necessary to first reduce the dimension of the data to a manageable size, keeping as much of the original information as possible, and then feed the reduced-dimension data into the system. Figure 1 summarises this situation, showing the dimension reduction as a **preprocessing stage** in the whole system.

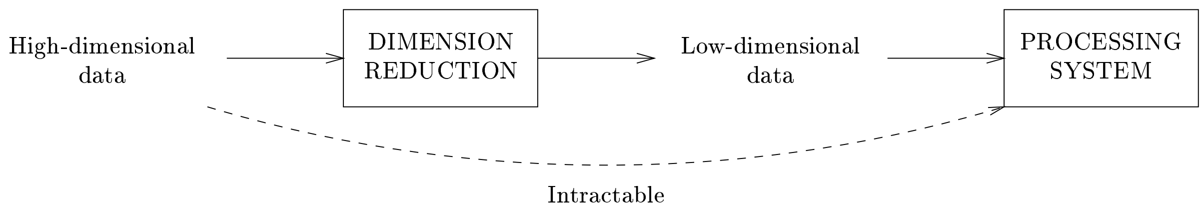

Figure 1: The dimension reduction problem. A given processing system is only effective with vector data of not more than a certain dimension, so data of higher dimension must be reduced before being fed into the system.

Sometimes, a phenomenon which is in appearance high-dimensional, and thus complex, can actually be governed by a few simple variables (sometimes called “hidden causes” or “latent variables” [29, 20, 21, 6, 74]). Dimension reduction can be a powerful **tool for modelling** such phenomena and improve our understanding of them (as often the new variables will have an interpretation). For example:

- Genome sequences modelling. A protein is a sequence of aminoacids (of which there are 20 different ones) with residue lengths varying from tens to tens of thousands. Proteins with the same spatial structure —but often with very different aminoacid sequences— are grouped together in families.

[1] Several representation techniques (see [89] for a review) exist that allow to visualise up to about 5-dimensional data sets, using colours, rotation, stereography, glyphs or other devices, but they lack the appeal of a simple plot; a well-known one is the grand tour [1]. Chernoff faces [13] allow even a few more dimensions, but are difficult to interpret and do not produce a spatial view of the data.

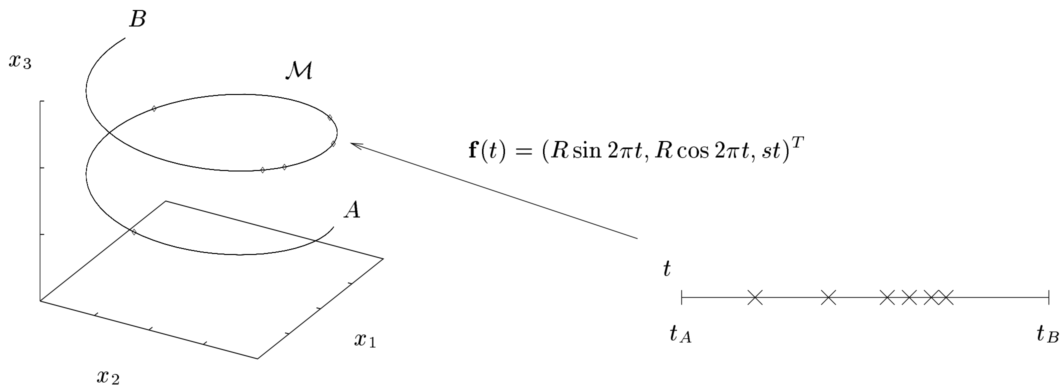

Figure 2: An example of coordinate representation of a one-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^3$ (a curve): the segment of spiral $\mathcal{M} = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{f}(t), t \in [t_A, t_B]\}$ for $\mathbf{f}(t) = (R \sin 2\pi t, R \cos 2\pi t, st)^T$.

A model of protein families can give insight into the properties of particular families and can also help to identify new members of a family or to discover new families. Probabilistic approaches to the investigation of the structure of protein families include hidden Markov models [70] and density networks [74].

- Speech modelling. It has been conjectured[2] that speech recognition, undoubtedly an exceedingly complex process, could be accomplished with only about 5 variables.

1.2 Definition of the problem

More formally, the problem of dimension reduction can be stated as follows. Suppose we have a sample $\{\mathbf{t}_n\}_{i=1}^N$ of D-dimensional real vectors drawn from an unknown probability distribution. The fundamental assumption that justifies the dimension reduction is that the sample actually lies, at least approximately, on a manifold (nonlinear in general) of smaller dimension than the data space. The goal of dimension reduction is to find a representation of that manifold (a coordinate system) that will allow to project the data vectors on it and obtain a low-dimensional, compact representation of the data.

For example, if principal component analysis is employed, a linear manifold (a vector subspace) is obtained and a representation of it is the vector basis formed by the first principal components of the data. In figure 2, a one-dimensional nonlinear manifold in $\mathbb{R}^3$ (a curve, namely a spiral of radius R and step s) is parameterised in terms of the dimensionless parameter t: $\mathcal{M} = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x} = \mathbf{f}(t), t \in [t_A, t_B]\}$ for $\mathbf{f}(t) = (R \sin 2\pi t, R \cos 2\pi t, st)^T$. Notice that the domain of the parameter can be a bounded interval. Given that the arc length of the spiral between points $\mathbf{x}_A = \mathbf{f}(t_A)$ and $\mathbf{x}_B = \mathbf{f}(t_B)$ is

$$\lambda = \int_{\mathbf{x}_A}^{\mathbf{x}_B} \sqrt{\left(\frac{dx_1}{dt}\right)^2 + \left(\frac{dx_2}{dt}\right)^2 + \left(\frac{dx_3}{dt}\right)^2}\, dt = \sqrt{(2\pi R)^2 + s^2}(t_B - t_A)$$

we could reparameterise the manifold in terms of the arc length $\lambda = \sqrt{(2\pi R)^2 + s^2}(t_B - t_A)$ (see section 4 for an example). Yet another parameterisation would be in terms of the angle $\theta = 2\pi t$, etc. This shows that the election of the coordinate system for a given manifold is by no means unique[3].

A more formal definition of k-manifold can be found in appendix H.

1.3 Why is dimension reduction possible?

Often, the original representation of the data will be redundant for several reasons:

- Many of the variables will have a variation smaller than the measurement noise and thus will be irrelevant.

[2] Unfortunately, I don't have a ready reference for this assertion!

[3] And the different coordinate systems do not have to keep a linear relationship. A familiar example are the Cartesian, spherical and cylindrical systems in $\mathbb{R}^3$.

- Many of the variables will be correlated with each other (e.g. through linear combinations or other functional dependence); a new set of incorrelated variables should be found.

Therefore in many situations it should be possible to somehow strip off the redundant information, producing a more economic representation of the data.

1.4 The *curse of the dimensionality* and the *empty space phenomenon*

The *curse of the dimensionality* (term coined by Bellman in 1961 [3]) refers to the fact that, in the absence of simplifying assumptions, the sample size needed to estimate a function of several variables to a given degree of accuracy (i.e. to get a reasonably low-variance estimate) grows exponentially with the number of variables.

For example, most density smoothers base their estimates on some local average of the neighbouring observations (see appendix F); but in order to find enough neighbours in high-dimensional spaces, multivariate smoothers have to reach out farther and the locality can be lost.

A way to avoid the curse of the dimensionality is to reduce the input dimension of the function to be estimated; this is the basis for the use of local objective functions, depending on a small number of variables, in unsupervised methods.

A related fact, responsible for the curse of the dimensionality, is the *empty space phenomenon* (Scott and Thompson [90]): high-dimensional spaces are inherently *sparse*. For example, the probability that a point distributed uniformly in the unit 10-dimensional sphere falls at a distance of 0.9 or less from the centre is only 0.35. This is a difficult problem in multivariate density estimation, as regions of relatively very low density can contain a considerable part of the distribution, whereas regions of apparently high density can be completely devoid of observations in a sample of moderate size [93]. For example, for a one-dimensional standard normal $\mathcal{N}(0,1)$, 70% of the mass is at points contained in a sphere of radius one standard deviation (i.e. the $[-1,1]$ interval); for a 10-dimensional $\mathcal{N}(\mathbf{0},\mathbf{I})$, that same (hyper)sphere contains only 0.02% of the mass and one has to take a radius of more than 3 standard deviations to contain 70%. Therefore, and contrarily to our intuition, in high-dimensional distributions the tails are much more important than in one-dimensional ones.

Another problem caused by the curse of the dimensionality is that, if there are linear correlations in the data (a very likely situation in high dimensions), the optimal mean integrated squared error when estimating the data density will be very large (see appendix E) even if the sample size is arbitrarily large [89].

Appendix I gives more insight about the geometry of high-dimensional spaces.

1.5 The *intrinsic dimension* of a sample

Consider a certain phenomenon governed by L independent variables. In practice, this phenomenon will actually appear as having (perhaps many) more degrees of freedom due to the influence of a variety of factors[4]: noise, imperfection in the measurement system, addition of irrelevant variables, etc. However, provided this influence is not too strong as to completely mask the original structure, we should be able to "filter" it out and recover the original variables or an equivalent set of them. We define the **intrinsic dimension** of a phenomenon as *the number of independent variables that explain satisfactorily that phenomenon.* From a purely geometrical point of view, the intrinsic dimension would be the dimension L of the manifold that embeds a sample of an unknown distribution in D-dimensional space ($D > L$), satisfying certain smoothness constraints.

Incidentally, we know from set theory that $\operatorname{card}\left(\mathbb{R}^D\right) = \operatorname{card}(\mathbb{R})$ $(= \aleph_0)$ for any $D \in \mathbb{N}$, which means that we can map invertibly and continuously $\mathbb{R}^D$ into $\mathbb{R}$, for example using the diagonal Cantor construction[5]. In principle, this would allow to find a (nonlinear) continuous mapping from $\mathbb{R}^D$ into $\mathbb{R}^L$, $L < D$, preserving all information. Of course, due to the finite precision this is of no practical application.

The determination of the intrinsic dimension of a distribution given a sample of it is central to the problem of dimension reduction, because knowing it would eliminate the possibility of over- or underfitting. All the dimension reduction methods known to the author take the intrinsic dimension as a parameter to be given by the user; a trial-and-error process is necessary to obtain a satisfactory value for it (in some practical applications, domain information may give insight into the intrinsic dimension).

Of course, this problem is itself ill-posed, because given a data sample it is possible to make a manifold of any dimension pass through it with negligible error given enough parameters. For example, in fig. 3 a

[4]We assume no knowledge about these factors; otherwise, a simplification of the problem would be possible.

[5]Write each of the components $x_1, \dots, x_D$ in a binary expansion and interleave the expansions to obtain the binary expansion of a number in $\mathbb{R}$.

one-dimensional manifold (the dotted curve) is forced to interpolate a set of points which naturally would lie on the two-dimensional manifold shown (the dotted rectangle). It is necessary to introduce some a priori knowledge about the degree of smoothness of the manifold, perhaps in terms of a regularisation term in a certain objective function.

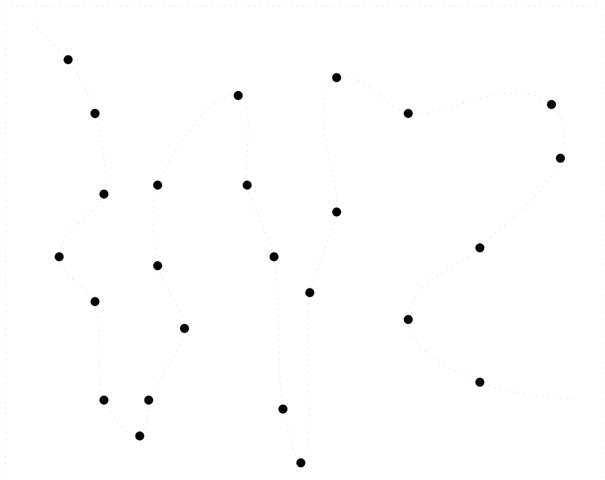

Figure 3: Curve or surface?

1.6 Views of the problem of dimension reduction

Given the basic nature of the curse of the dimensionality, it is not surprising that many different fields are affected by it. We can look at the dimension reduction problem from a number of perspectives:

- Basically, it is nothing else but a projection, i.e. a mapping from a D-dimensional space onto an L-dimensional one, for $D > L$, with its associated change of coordinates.
- In statistics it is related to multivariate density estimation, regression and smoothing techniques.
- From the pattern recognition standpoint, it is equivalent to feature extraction, where the feature vector would be the reduced-dimension one.
- In information theory it is related to the problem of data compression and coding.
- Many visualisation techniques are actually performing some kind of dimension reduction: multidimensional scaling [71], Sammon mapping [87], etc.
- Complexity reduction: if the complexity in time or memory of an algorithm depends on the dimension of its input data as a consequence of the curse of the dimensionality, reducing this will make the algorithm more efficient (at the expense of moving that complexity to the dimension reduction procedure, of course).
- Latent-variable models: in the latent-variable approach, we assume that a small number of hidden *causes* acting in combination gives rise to the apparent complexity of the data [6]. The latent-variable space is the low-dimensional representation or coordinate system mentioned before.

1.7 Classes of dimension reduction problems

We attempt here a rough classification of the dimension reduction problems:

- *Hard* dimension reduction problems, in which the data have dimension ranging from hundreds to perhaps hundreds of thousands of components, and usually a drastic reduction (possibly of orders of magnitude) is sought. The components are often repeated measures of a certain magnitude in different points of space or in different instants of time. In this class we would find pattern recognition and classification problems involving images (e.g. face recognition, character recognition, etc.) or speech (e.g. auditory models). Principal component analysis is one of the most widespread techniques in most practical cases.

- *Soft* dimension reduction problems, in which the data is not too high-dimensional (less than a few tens of components), and the reduction not very drastic. Typically, the components are observed or measured values of different variables, which have an straightforward interpretation. Most statistical analyses in fields like social sciences, psychology, etc. fall in this class. Typical methods include all the usual multivariate analysis methods [75]: principal component analysis, factor analysis, discriminant analysis, multidimensional scaling, etc.

- *Visualisation* problems, in which the data doesn't normally have a very high dimension in absolute terms, but we need to reduce it to 2, 3 or 4 at most in order to plot it. Lots of applications from many disciplines fall into this category. A number of methods are used in practice, including projection pursuit, principal component analysis, multidimensional scaling and self-organising maps and their variants, among others, as well as interactive programs that allow manual intervention (such as Xgobi [95]).

If we allow the time variable in, we find two further categories: *static dimension reduction* and *time-dependent dimension reduction*. The latter could possibly be useful for vector time series, such as video sequences or continuous speech.

A further categorisation can be done attending to the *discrete* or *continuous* nature of the data vectors. An example of discrete data is found in *electropalatography* (EPG) [44], where each vector is a sequence of about 100 binary values which indicate the presence or absence of tongue-palate contact in coarticulation studies[6]. Another example of discrete data are genome sequences, as mentioned earlier. Images are an example of continuous data, where each pixel value can be scaled to the $[0, 1]$ interval.

1.8 Methods for dimension reduction. Overview of the report

As we will see, when the manifold sought is linear and the criterion is to maximise the directional variance in an incorrelated way, the dimension reduction problem has an exact analytical solution and corresponds to principal component analysis [62]. This, together with its reasonably good results in practice, make it probably the most widespread and well-known of all dimension reduction techniques.

In other cases the problem is harder and a number of techniques approach it from different perspectives: low-dimensional projections of the data (projection pursuit, generalised additive models), regression (principal curves), self-organisation (Kohonen's maps), topologically continuous mappings (generative topographic mapping) and others.

The rest of this report is organised as follows: section 2 briefly presents principal component analysis. Section 3 deals with projection pursuit, a more general statistical technique for dimension reduction (of which PCA and other methods are particular cases); particular emphasis is done on the concept of projection index. Projection pursuit regression, a nonparametric regression approach based on projection pursuit, is introduced in section 3.7, and generalised additive models, a particular case of projection pursuit regression, in section 3.8. Section 4 deals with principal curves and principal surfaces, which appear naturally as a generalisation of regression in the nonlinear, symmetrical case. Section 5 introduces the concept of self-organisation and topological continuity in mappings, and illustrates it with the well-known Kohonen's maps and with the generative topographic mapping (GTM), a method based in MacKay's density networks [74]. Section 6 gives connectionist implementations for several of the techniques previously reviewed and some improvements to them. The report is concluded with a short discussion of the material presented and of further work. Several appendices complement the mathematical part of the main text.

[6] See [43] for a discussion of several ad-hoc dimension reduction methods in use in electropalatography.

2 Principal Component Analysis

Principal component analysis[7] (PCA) is possibly the dimension reduction technique most widely used in practice, perhaps due to its conceptual simplicity and to the fact that relatively efficient algorithms (of polynomial complexity) exist for its computation. In signal processing it is known as the Karhunen-Loève transform.

Let us consider a sample $\{\mathbf{x}_i\}_{i=1}^n$ in $\mathbb{R}^D$ with mean $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$ and covariance matrix $\mathbf{\Sigma} = \mathrm{E}\left\{(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^T\right\}$, with spectral decomposition $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ ($\mathbf{U}$ orthogonal and $\mathbf{\Lambda}$ diagonal). The principal component transformation $\mathbf{y} = \mathbf{U}^T(\mathbf{x}-\overline{\mathbf{x}})$ yields a reference system in which the sample has mean $\mathbf{0}$ and diagonal covariance matrix $\mathbf{\Lambda}$ containing the eigenvalues of $\mathbf{\Sigma}$: the variables are now uncorrelated. One can discard the variables with small variance, i.e. project on the subspace spanned by the first L principal components, and obtain a good approximation (the best linear one in the LS sense) to the original sample. Figure 4 shows an example.

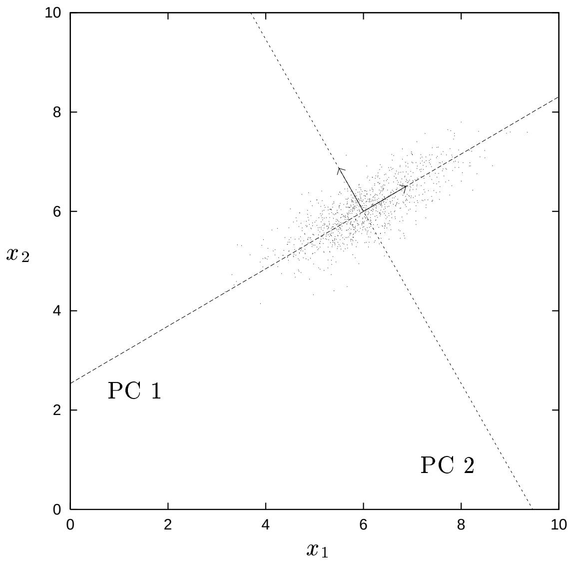

Figure 4: Bidimensional, normal point cloud with its principal components.

Geometrically, the hyperplane spanned by the first L principal components is the regression hyperplane that minimises the orthogonal distances to the data. In this sense, PCA is a *symmetric* regression approach, as opposed to standard linear regression, which points one component as response variable and the rest as predictors (see sections 4 and F.1).

The key property of principal component analysis is that it attains the best linear map $\mathbf{x} \in \mathbb{R}^D \longrightarrow \mathbf{x}^* \in \mathbb{R}^L$ in the senses of:

- Least squared sum of errors of the reconstructed data.
- Maximum mutual information (assuming the data vectors $\mathbf{x}$ distributed normally) between the original vectors $\mathbf{x}$ and their projections $\mathbf{x}^*$: $I(\mathbf{x};\mathbf{x}^*) = \frac{1}{2}\ln\left((2\pi e)^L \lambda_1 \dots \lambda_L\right)$, where $\lambda_1, \dots, \lambda_L$ are the first L eigenvalues of the covariance matrix.

The first principal components are often used as starting points for other algorithms, such as projection pursuit regression, principal curves, Kohonen's maps or the generalised topographic mapping, all of which are reviewed in this report.

There exist several neural network architectures capable to extract principal components; see section 6.2. Also, when the data is clustered, it can be more convenient to apply PCA locally. For example, in piecewise PCA (Kambhatla and Leen [67]), a partition of $\mathbb{R}^D$ is defined by some form of vector quantisation of the data set and PCA applied locally in each region. This approach is fast and has comparable results to autoencoders.

A number of numerical techniques exist for finding all or the first few eigenvalues and eigenvectors of a square, symmetric, semidefinite positive matrix (the covariance matrix) in $\mathcal{O}(D^3)$: singular value

[7] See [27, 61, 62] for a more comprehensive treatment. Also, see section C.2 for a comparison with other transformations of the covariance matrix.

decomposition, Cholesky decomposition, etc.; see [81] or [99]. When the covariance matrix, of order $D \times D$, is too large to be explicitly computed one could use neural network techniques (section 6.2), some of which do not require more memory space other than the one needed for the data vectors and the principal components themselves. Unfortunately, these techniques (usually based on a gradient descent method) are much slower than traditional methods.

The disadvantages of PCA are:

- It is only able to find a linear subspace and thus cannot deal properly with data lying on nonlinear manifolds.
- One does not know how many principal components to keep, although some thumb rules are applied in practice. For example, eliminate components whose eigenvalues are smaller than a fraction of the mean eigenvalue, or keep as many as necessary to explain a certain fraction of the total variance [27].

3 Projection Pursuit

3.1 Introduction

Often, especially during the initial stages, the analysis of a data set is exploratory: one wishes to gain insight about the structure of the data. Projection pursuit[8] is an unsupervised technique that picks *interesting* low-dimensional linear orthogonal projections of a high-dimensional point cloud by optimising a certain objective function called **projection index**. It is typically used to take profit of the human ability to discover patterns in low-dimensional (1- to 3-D) projections: they are visual representations of the projected data density (e.g. histograms or other smoothed density estimates, scatterplots, contour plots) and can be inspected to ascertain the structure of the data (clustering, etc.).

Projections are smoothing operations in that structure can be obscured but never enhanced: any structure seen in a projection is a shadow of an actual structure in the original space. It is of interest to *pursue* the sharpest projections, that will reveal most of the information contained in the high-dimensional data distribution.

A remarkable feature of projection pursuit is that, when applied to regression or density estimation (e.g. projection pursuit regression), it is one of the very few multivariate methods able to bypass the *curse of the dimensionality* to some extent. However, the power of any projection pursuit algorithm to find important structure will still suffer if the sample size is small and the dimension large. Several methods of the classical multivariate analysis are special cases of projection pursuit (for example, principal component analysis).

The disadvantages of projection pursuit are:

- It works with linear projections and therefore it is poorly suited to deal with highly nonlinear structure.
- Projection pursuit methods tend to be computationally intensive.

The (scaled) variable loadings (components of the projection vectors) that define the corresponding solution indicate the relative strength that each variable contributes to the observed effect. Additionally, applying *varimax* rotation[9] or similar procedures to the projections will produce the same picture but with an easier interpretation of the variable loadings.

3.1.1 What is an *interesting* projection?

We consider that a projection is *interesting* if it contains structure. Structure in the data can be:

- Linear: correlations between variables are readily detected by linear regression.
- Nonlinear: clustering or multimodality (the density function presents several peaks), skewness, kurtosis (sharp peaking), discontinuities and in general concentration along nonlinear manifolds.

According to this and the following results:

- For fixed variance, the normal distribution has the least information, in both the senses of Fisher information and negative entropy [16].
- For most high-dimensional clouds, most low-dimensional projections are approximately normal (Diaconis and Freedman [24]).

We will consider the normal distribution as the least structured (or least interesting) density.

For example, figure 5 shows two 2-D projections of a 3-D data set consisting of two clusters. The projection on the plane spanned by $\mathbf{e}_2$ and $\mathbf{e}_3$ is not very informative, as both clusters confuse in one; this projection nearly coincides with the one in the direction of the first principal component, which proves that the projection index of PCA (maximum variance; see section 3.2) is not a good indicator of structure. However, the projection on the plane spanned by $\mathbf{e}_1$ and $\mathbf{e}_2$ clearly shows both clusters.

[8] The term *projection pursuit* was introduced by Friedman and Tukey [38] in 1974, along with the first projection index. Good reviews of projection pursuit can be found in Huber [53] and Jones and Sibson [65].

[9] Varimax rotation [66] is a procedure that, given a subspace or projection, selects a new basis for it that maximises the variance but giving large loadings to as few variables as possible. The projection will be mainly explained by a few variables and thus be easier to interpret.

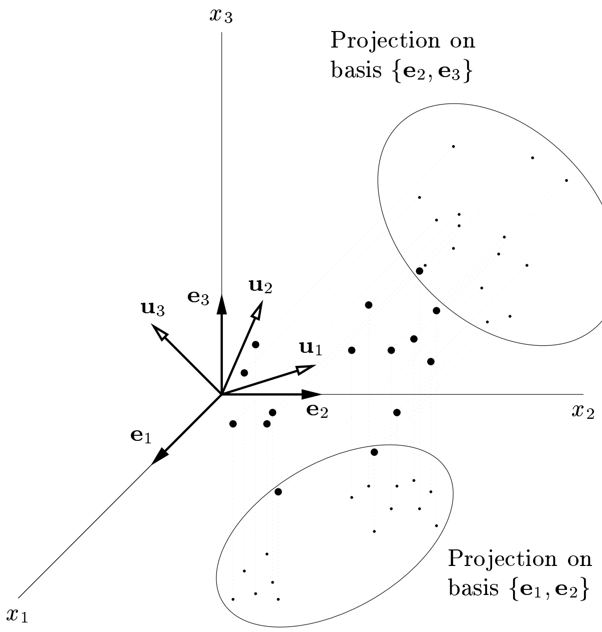

Figure 5: Two-dimensional projections of a three-dimensional data set.

3.2 The projection index

3.2.1 Definition[10]

A **projection index** Q is a real functional on the space of distributions on $\mathbb{R}^k$:

$$Q : f \in L_2(\mathbb{R}^k) \longrightarrow q = Q(f) \in \mathbb{R}$$

Normally, $f = F_{\mathbf{A}}$ will be the distribution of the projection (of matrix $\mathbf{A}$) of a D-dimensional random variable X with distribution F, and will correspond to a k-dimensional random variable $Y = \mathbf{A}^T X$, if $\mathbf{A}$ is $D \times k$ (see section D). Abusing of notation, we will write $Q(X)$ and $Q(\mathbf{A}^T X)$ instead of $Q(F)$ and $Q(F_{\mathbf{A}})$.

Projection pursuit attempts to find projection directions $\mathbf{a}_i$ for a given distribution F which produce local optima of Q. To make the optimisation problem independent of the length of the projection vectors and to obtain uncorrelated directions, the $\mathbf{a}_i$ are constrained to be unit length and mutually orthogonal (i.e. the column vectors of $\mathbf{A}$ must be orthonormal). The optimisation problem is then

$$\text{Optimise} \quad Q(\mathbf{A}) \quad \text{subject to} \quad \mathbf{a}_i^T \mathbf{a}_j = \delta_{ij} \tag{3.1}$$

In this work, we will consider only one-dimensional projections $\mathbf{a}$ —although the treatment is easily extendable to k-dimensional projections $\mathbf{A}$. We will also consider problem (3.1) as a maximisation of the index unless otherwise indicated.

3.2.2 Classification

Let $s \in \mathbb{R}$, $\mathbf{t} \in \mathbb{R}^D$ and X a random variable with values in $\mathbb{R}^D$. Following Huber [53], we distinguish the following classes of indices:

- Class I (location-scale equivariance): $Q_I(sX + \mathbf{t}) = sQ_I(X) + \mathbf{t}$
- Class II (location invariance, scale equivariance): $Q_{II}(sX + \mathbf{t}) = |s|Q_{II}(X)$
- Class III (affine invariance): $Q_{III}(sX + \mathbf{t}) = Q_{III}(X)$, $s \neq 0$

[10] Huber [53] introduces two versions of projection pursuit:
- An *abstract* version, based on D-dimensional probability distributions: X will be a random variable with values in $\mathbb{R}^D$.
- A *practical* version, based on D-dimensional samples or point clouds: $\mathbf{X}_{D \times n}$ will be the matrix representation of a sample of n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ in $\mathbb{R}^D$.

To keep this report short, we will stick to the *abstract* version and the context will show when we actually refer to a sample.

The following equalities hold:

$$|Q'_I - Q''_I| = Q_{II} \qquad \frac{Q'_{II}}{Q''_{II}} = Q_{III}$$

where Q'_I would be a class I index, etc.

3.2.3 Desirable properties

In general, there will be several interesting projections, each showing different insight, which correspond to local optima of the projection index. One way of discovering them is to repeatedly invoke the optimisation procedure, each time removing from consideration the solutions previously found (**structure removal**). Accordingly, a good projection index should:

- Have continuous first (at least) derivative, to allow the use of gradient methods.
- Be rapidly computable —as well as its derivative(s)—, as the optimisation procedure requires evaluating it many times.
- Be invariant to all nonsingular affine transformations in the data space (to discover structure not captured by the correlation), i.e. be of class III.
- Satisfy:

$$Q(X+Y) \le \max(Q(X), Q(Y)) \tag{3.2}$$

  because, by the central limit theorem, $X+Y$ must be more normal (less interesting) than the less normal of X and Y. (3.2) implies $Q(X_1 + \cdots + X_n) \le Q(X)$ if $X_1, \dots, X_n$ are copies of X, and therefore $Q(\mathcal{N}) \le Q(X)$ if $\mathcal{N}$ is normal.

  All known class III indices satisfying (3.2) are of the form $Q_{III}(X) = h(S_1(X)/S_2(X))$, with h monotone increasing and S_1, S_2 are class II satisfying

$$\text{Subadditivity:} \quad S_1^2(X+Y) \le S_1^2(X) + S_1^2(Y)$$

$$\text{Superadditivity:} \quad S_2^2(X+Y) \ge S_2^2(X) + S_2^2(Y)$$

3.2.4 Outliers

The projection pursuit procedure will tend to identify outliers because the presence of the latter in a sample gives it the appearance of nonnormality. This sometimes can obscure the clusters or other interesting structure being sought. Also, the sample covariance matrix is strongly influenced by extreme outliers. Consequently, all methods relying on it (e.g. through data sphering) will not be robust against outliers.

The effect of outliers can be partially tackled by robust sphering, e.g. using a simple multivariate trimming method:

Set d_0 to a convenient threshold distance.
Repeat
  Delete all observations that lie farther than d_0 from the mean.
  Recompute the mean.
Until not more than a certain small fraction of the data are deleted.

3.2.5 Indices based on polynomial moments

Approximating the density by a truncated series of orthogonal polynomials (Legendre, Hermite, etc.; see section E.3.7) allows easy computation of the moments. Indices based on polynomial moments (see eqs. (3.4) and (3.7) for examples):

- Are computationally efficient.
- Don't have to be recomputed at each step of the numerical procedure, as they can be derived for each projection direction from sufficient statistics of the original data set.
- Heavily emphasise departure from normality in the tails of the distribution, thus being oversensitive to outliers; as a result, they perform poorly. Some methods try to address this problem in several ways:

– In EPP (section 3.6), a nonlinear transformation from $\mathbb{R}$ to $[-1,1]$ using a normal distribution function is introduced.

– In the BCM neuron with an objective function (section 6.5.2), a sigmoidal function is applied to the projections.

Second-order polynomials give measures of the mean and variance of the distribution, but not information on multimodality. Higher-order polynomials are required to measure deviation from normality[11].

3.2.6 Projection pursuit and density estimation

Silverman [93] suggests that the maximisation of the projection index I can be carried out by:

1. Finding an estimate of the density of the projected points.
2. Maximising the projection index operating on the estimate.

Step 2 is enormously facilitated if an appropriate (differentiable) density estimate $\hat{f}$ is used; a particular example is the use of orthogonal expansions (section E.3.7). Good results can be obtained using the corresponding optimum smoothing parameter h_{opt}, although the behaviour of the index is not very sensitive to the choice of h (see appendix E).

3.2.7 Examples[12]

Consider a random variable X with density f, expectation $\boldsymbol{\mu} = \mathrm{E}\{X\}$ and covariance matrix $\boldsymbol{\Sigma} = \mathrm{E}\left\{(X-\boldsymbol{\mu})(X-\boldsymbol{\mu})^T\right\}$:

- Average:

$$Q_I(X) = \mathrm{E}\{\cdot\}.$$

  In this case $\max_{\|\mathbf{a}\|=1} Q(\mathbf{a}^T X) = \|\boldsymbol{\mu}\|$ for $\mathbf{a}_0 = \boldsymbol{\mu}/\|\boldsymbol{\mu}\|$.

- Variance:

$$Q_{II}(X) = \mathrm{var}\{\cdot\} = \mathrm{E}\left\{(\mathbf{a}^T(X-\boldsymbol{\mu}))^2\right\}. \tag{3.3}$$

  In this case $\max_{\|\mathbf{a}\|=1} Q(\mathbf{a}^T X) = \lambda_{\max}$ for $\mathbf{a}_0 = \mathbf{u}_{max}$, the largest eigenvalue and normalised principal eigenvector of $\boldsymbol{\Sigma}$, respectively. In other words, this index finds the first principal component.

- Standardised absolute cumulants $k_m(X)$ (defined in eq. (B.2)):

$$Q_{III}(X) = \frac{|k_m(X)|}{k_2(X)^{m/2}} \qquad m > 2. \tag{3.4}$$

- Fisher information:

$$Q_{III}(X) = J(X_\theta) = \mathrm{E}\left\{\frac{\partial}{\partial\theta}\ln f_\theta(x)\right\}^2 = \int \frac{\left(\frac{\partial}{\partial\theta} f_\theta(x)\right)^2}{f_\theta(x)}\,dx \tag{3.5}$$

  where f depends on some parameter θ. $J(X_{\mathrm{var}\{X\}})$ is minimised by the normal pdf (B.5): $J(\mathcal{N}_{\sigma^2}) = 1/\sigma^2$ [53].

- Negative Shannon entropy:

$$Q_{III}(X) = -h(X) = \mathrm{E}\{\ln f(x)\} = \int f(x)\ln f(x)\,dx \tag{3.6}$$

  For fixed variance, $-h(X)$ is minimised by the normal pdf (B.5): $h(\mathcal{N}_{\sigma^2}) = \ln\sqrt{2\pi e\sigma^2}$ [16].

  Jones and Sibson[13] [65] propose two ways to evaluate the entropy index (3.6):

[11] Although many synaptic plasticity models are based on second-order statistics and lead to the extraction of principal components [59].

[12] The visualisation program Xgobi [95, 68] implements several of these indices.

[13] In a variant of projection pursuit they call *projection pursuit exploratory data analysis (PPEDA)*.

– Implementing it as a sample entropy, $\int \hat{f} \ln \hat{f}\, dx$ (by numerical integration) or $\frac{1}{n}\sum_i \ln \hat{f}(\mathbf{a}^T\mathbf{x}_i)$, where $\hat{f}$ is the (univariate) nonparametric density estimate of the projected points $\mathbf{a}^T\mathbf{x}_i$. Both are very slow to compute.
– Approximating f by an expansion in terms of the cumulants; for 1-D projections:

$$\int f \ln f\, dx \approx \frac{1}{12}\left(k_3^2 + \frac{1}{4}k_4^2\right). \tag{3.7}$$

- Original univariate Friedman-Tukey index [38]:

$$Q(X) = \hat{\sigma}_\alpha(X) \sum_{i,j} I_{[0,\infty]}(h - |X_i - X_j|).$$

where $\hat{\sigma}_\alpha(X)$ is the α-trimmed standard deviation (i.e. the standard deviation after $\alpha/2$ of the data are deleted on each end) and h is a parameter. The criterion is large whenever many points are clustered in a neighbourhood of size h.

Huber [53] noted that this index is proportional to $\int f_h^2$, where f_h is the kernel estimate of f with a uniform kernel U$[-0.5, 0.5]$; so, the Friedman-Tukey index is essentially based on $\int f^2$. However, its optimisation is difficult because of the use of both a discontinuous kernel and a discontinuous measure of scale.

Jones [64] found little difference in practice between $\int f \ln f$ and $\int f^2$. However, the integral $\int f \ln f$ is minimised by the normal, while $\int f^2$ is minimised by the Epanechnikov kernel (see table 4).

All the previous class III indices satisfy property (3.2). The following two indices are computed via an orthogonal series estimator of f using Hermite polynomials:

- $I^H = \int (f(x) - \phi(x))^2\, dx$ (Hall [42]).
- $I^N = \int (f(x) - \phi(x))^2 \phi(x)\, dx$ (Cook *et al.* [14]).

where $\phi(x)$ is the normal density of eq. (B.6). Other indices are:

- Eslava and Marriott [28] propose two two-dimensional indices designed to display all clusters:
  – Minimise the *polar nearest neighbour index* $I = \mathrm{E}\left\{\min\left(|\theta_i - \theta_{i-1}|, |\theta_{i+1} - \theta_i|\right)\right\}$, where $\{\theta_i\}_{i=1}^n$ are the polar angles of the projected sample points, ordered increasingly.
  – Maximise the *mean radial distance*, equivalent to minimising the variance of the radial distance for sphered data.
- Posse [78, 79, 80] proposes a two-dimensional index based on radial symmetry.

There is no general unanimity about the merits of the projection indices presented, except that moment indices give poor results than the rest, albeit being faster to compute.

3.2.8 Extension to higher dimensions

Most of the indexes discussed admit a simple extension to two dimensions, depending on their nature:

- Based on density estimation: by estimating the two-dimensional density and integrating.
- Based on moments: by using bivariate moments (e.g. $I^N = \iint (f(x,y) - \phi(x)\phi(y))^2 \phi(x)\phi(y)\, dxdy$ with bivariate Hermite polynomials).

However, extension to high dimensions is analytically very complicated for most indexes.

3.3 Multidimensional projections

If the projections $\mathbf{A}^T X$ are of dimension $k > 1$:

- Computations get harder (optimisation over kD variables instead of only D).
- Optimising Q yields a k-dimensional subspace, but in some cases it can be preferable to get an ordered set of k directions. This can be attained by a recursive approach: find the most interesting direction, remove the structure associated with it, iterate. This leads to projection pursuit density estimation (see section 3.7.3).

Usual starting projections for the optimisation procedure are the principal components or just random starts. Repeated runs with different starting projections are required to explore the local optima and provide with "locally most interesting" views of the data. Optima with small domains of attraction, if such actually occur, are inevitably likely to be missed[14].

3.4 Relation to PCA

From the point of view of projection pursuit, PCA yields a set of directions ordered by decreasing projected variance: the projection index, equation (3.3), is the variance of the projected data and is to be maximised subject to the constraint that the projection directions are orthonormal. PCA is therefore a particular case of projection pursuit.

If the covariance matrix of the data set has rank $L < D$, where D is the dimension of the data space (in a number of cases this happens because the sample size is smaller than D), the projection pursuit has to be restricted to the L largest principal components, because if a subspace contains no variation, it cannot contain any structure. Principal components having a very small variation can be ignored as well —thus further reducing the space dimension— as they will usually be dominated by noise and contain little structure.

3.5 Relation to computer tomography

Huber [53] points out the duality between projection pursuit and computer tomography:

- Projection pursuit is *reduction to projections*: we have some high-dimensional information (a random sample of a density) and we seek a reduced set of projections of that density that best approximate it (additive- or multiplicatively).
- Computer tomography is *reconstruction from projections*: we have a finite (but usually fairly dense and equispaced) set of 1-D (2-D) projections of a 2-D (3-D) density and we seek a density whose projections best agree with those we have.

3.6 Exploratory projection pursuit (EPP)

Exploratory projection pursuit (EPP) (Friedman [34]) is a projection pursuit procedure that provides with a projection index and an optimisation strategy, and thus serves well to illustrate how projection pursuit works in practice.

We consider first one-dimensional EPP. Let Z a centred and sphered random variable and $X = \mathbf{a}^T Z$ a projection. We obtain a new random variable $R = 2\Phi(X) - 1 \in [-1, 1]$, where $\Phi(X)$ is the standard normal cdf, eq. (B.7). Then R is uniform $[-1, 1]$ if X is standard normal. Hence, nonuniformity of R in $[0, 1]$ will mean nonnormality of X in $(-\infty, \infty)$. We take as projection index $I(\mathbf{a})$ the following L_2 measure of nonuniformity of R:

$$I(\mathbf{a}) = \int_{-1}^{1} \left(p_R(r) - \frac{1}{2} \right)^2 dr = \int_{-1}^{1} p_R^2(r)\, dr - \frac{1}{2}$$

where $p_R(r)$ is the pdf of R, but approximated by a Legendre polynomial expansion. The (projected) normal minimises I; our problem is $\max_{\|\mathbf{a}\|=1} I(\mathbf{a})$. I can be rapidly computed using the recursive relations for the Legendre polynomials and their derivatives (eq. (E.1)).

Two-dimensional EPP is analogous with $\mathbf{a}^T\mathbf{b} = 0$ and $\|\mathbf{a}\| = \|\mathbf{b}\| = 1$, i.e. uncorrelated linear combinations with unit variance. Sphering ensures this.

To avoid suboptimal maxima in the maximisation procedure —which can be visualised as high-frequency ripple superimposed on the main variational structure of I— a hybrid optimisation strategy is used:

1. A simple, coarse-stepping optimiser that very rapidly gets close to a substantive maximum (i.e. within its domain of attraction), avoiding pseudomaxima. It maximises along the coordinate axes, which for sphered data are the principal components of the original data; see [34] for details.
2. A gradient method (steepest ascent, quasi-Newton) to quickly converge to the solution.

Like many other methods, EPP gives large weight to fluctuations of f in its tails (because the normal density ϕ is small there) and is therefore sensitive to outliers and scaling.

[14] Some algorithms exist that try to avoid nonglobal optima (e.g. EPP, section 3.6).

3.6.1 Localised EPP

Localised EPP (Intrator [56]) is a nonparametric classification method for high-dimensional spaces. A recursive partitioning method is applied to the high-dimensional space and low-dimensional features are extracted via EPP in each node of the tree using the CART method; an exploratory splitting rule is used, which is potentially less biased to the training data. Implementation of localised EPP in a backpropagation network leads to a modified unsupervised delta rule.

Localised EPP is computationally practical and:

- Less sensitive to the *curse of the dimensionality* due to the feature extraction step.
- Less biased to the training data due to the CART method.

3.7 Projection pursuit regression (PPR)

Projection pursuit regression (PPR) (Friedman and Stuetzle [36]) is a nonparametric regression approach for the multivariate regression problem (see section F.1) based in projection pursuit. It works by **additive composition**, constructing an approximation to the desired response function by means of a sum of low-dimensional smooth functions, called **ridge functions**, that depend on low-dimensional projections through the data[15]:

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{j} g_k(\mathbf{a}_k^T \mathbf{x}) \tag{3.8}$$

where each g_k is constant on hyperplanes. The PPR algorithm determines $\{\mathbf{a}_k, g_k\}_{i=1}^{j}$ as follows:

Starting points: set the projection directions $\mathbf{a}_k$ to the first principal components (or some random vectors). The residuals are, initially, $r_{i0} = y_i$. Set $j = 1$.

Repeat

1. Assuming $\{\mathbf{a}_k, g_k\}_{k=1}^{j-1}$ determined, compute the current residuals:

$$r_{i,j-1} = y_i - \sum_{k=1}^{j-1} g_k(\mathbf{a}_k^T \mathbf{x}_i) = r_{i,j-2} - g_{j-1}(\mathbf{a}_{j-1}^T \mathbf{x}_i) \qquad i = 1, \dots, n$$

2. Fit a nonparametric smooth curve[16] g_j to the residuals $\{r_{i,j-1}\}_{i=1}^{n}$ as a function of $\mathbf{a}^T\mathbf{x}_i$ for any $\mathbf{a} \in \mathbb{R}^D$ with $\|\mathbf{a}\| = 1$.

3. Projection pursuit step: minimise the sum of squared residuals (the L_2-norm) relative to g over $\mathbf{a}$:

$$\mathbf{a}_j = \arg\min_{\|\mathbf{a}\|=1} \sum_{i=1}^{n} (r_{i,j-1} - g(\mathbf{a}^T \mathbf{x}_i))^2 = \arg\min_{\|\mathbf{a}\|=1} \sum_{i=1}^{n} r_{ij}^2. \tag{3.9}$$

4. Insert $\mathbf{a}_j$, g_j as the next term in (3.8).

Until the improvement in (3.9) is small.

Notice that:

- The representation (3.8), if it is exact, will only be unique if f is a polynomial.
- Some functions cannot be represented by a sum of the form (3.8) for finite m; e.g. $f(x_1, x_2) = e^{x_1 x_2}$.

Equation (3.8) can be readily implemented by a multilayer perceptron and this allows a neural network implementation of PPR; see section 6.3 for details.

[15] We consider only one component of the vector function $\mathbf{f}$ of section F.1.

[16] Friedman and Stuetzle [36] use the Friedman's supersmoother [33]; see section E for other smoothers.

3.7.1 Penalty terms in PPR

Expressing the smoothness constraint on the ridge functions g_j by some smoothness measure C, we can merge steps 2 and 3 in the PPR algorithm:

2–3: $(\mathbf{a}_j, g_j) = \arg\min_{g,\|\mathbf{a}\|=1} \sum_{i=1}^{n} r_{ij}^2(\mathbf{x}_i) + C(g)$.

This shows that the estimation of the nonparametric ridge functions g_j is coupled to the estimation of the projection directions $\mathbf{a}_j$. Therefore, if overfitting occurs when estimating one of the ridge functions (which is very likely at the beginning), the search for optimal projections will not yield good results. This can be addressed in several ways:

- Choosing the ridge functions from a very small family of functions (e.g. sigmoids with variable bias). Then, there is no need to estimate a nonparametric ridge function, but the complexity of the architecture is increased. This approach is widely used in neural networks.
- Concurrently (instead of sequentially) estimating a fixed number of ridge functions and projection directions, provided that the ridge functions are taken from a very limited set of functions. It presents a small additional computational burden and is also widely used in neural networks.
- Partially decoupling the estimation of the ridge functions from the estimation of the projections.

Intrator [58] proposes a combination of all of the above: minimise

$$\sum_{i=1}^{n} r_{ij}^2(\mathbf{x}_i) + C(g_1, \dots, g_l) + I(\mathbf{a}_1, \dots, \mathbf{a}_l)$$

where

- The partial decoupling of the search of $\mathbf{a}_j$ from that of g_j is achieved by a penalty term based on a projection index $I(\mathbf{a})$.
- The concurrent minimisation over several $\mathbf{a}_j$ and g_j is expressed in a general way by a penalty term $B(\hat{f}) = C(g_1, \dots, g_l) + I(\mathbf{a}_1, \dots, \mathbf{a}_l)$.

A neural network implementation of these ideas can be an MLP in which the number of hidden units is much smaller than that of input units, with a penalty added to the hidden layer; the learning rule takes the form:

$$\dot{w}_{ij} = -\eta \left(\frac{\partial E}{\partial w_{ij}} + \frac{\partial I(\mathbf{w})}{\partial w_{ij}} + \text{contribution of cost/complexity terms} \right)$$

where E is the error of the network (for an example, see equation (6.2) for the BCM network with inhibiting connections).

3.7.2 Projection pursuit density approximation (PPDA)

If f is not just any arbitrary function in $\mathbb{R}^D$ but a probability density, additive decompositions are awkward because the approximating sums are not densities (they do not integrate to 1 in general). **Multiplicative decompositions** are better (cf. eq. (3.8)):

$$\hat{f}(\mathbf{x}) = \prod_{k=1}^{j} h_k(\mathbf{a}_k^T \mathbf{x}) \tag{3.10}$$

If $j < D$, (3.10) is not integrable; then we take

$$\hat{f}(x) = f_0(\mathbf{x}) \prod_{k=1}^{j} h_k(\mathbf{a}_k^T \mathbf{x})$$

where $f_0(\mathbf{x})$ is a density in $\mathbb{R}^D$ with the same mean and covariance as f.

The construction algorithm is, as in PPR, stepwise, and can be of two types: *synthetic*, where by successive modifications to a starting simple density we build up the structure of f; or *analytic*, where by successive modifications to f we strip away its structure.

The quality of approximation of the estimate to f can be measured with various criteria: relative entropy (B.8), Hellinger distance (B.9), etc. For the relative entropy $D(f\|g)$, Huber [53] proposes the following algorithms:

- Synthetic: start with a Gaussian distribution of the same mean and covariance matrix as f. In each step, find a new projection direction $\mathbf{a}$ maximising $D(f_{\mathbf{a}}||\hat{f}_{\mathbf{a}})$, where $\hat{f}$ is the current estimate and the subindex $\mathbf{a}$ indicates one-dimensional marginalisation of $\mathbf{a}^T\mathbf{x}$ under the corresponding density. Replace the current estimate by $\hat{f} f_{\mathbf{a}}/\hat{f}_{\mathbf{a}}$.

- Analytic: start with f. In each step, replace the current estimate by $\hat{f}\phi_{\mathbf{a}}/\hat{f}_{\mathbf{a}}$, where ϕ is a Gaussian distribution of the same mean and covariance matrix as f. Continue until $j = D$ or $D(f||g) = 0$, when $\hat{f}$ will be the Gaussian density.

3.7.3 Projection pursuit density estimation (PPDE)

Projection pursuit density estimation (PPDE; Friedman, Stuetzle and Schroeder [37]) is appropriate when the variation of densities is concentrated in a linear manifold of the high-dimensional space. Given the data sample in $\mathbb{R}^D$, it operates as follows:

1. Sphere the data.

2. Take as starting estimate the standard normal in D dimensions, $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (B.4).

3. Apply the synthetic PPDA.

3.8 Generalised additive models

A *generalised additive model* (GAM) (Hastie and Tibshirani [49]) for a D-dimensional density $f(\mathbf{x})$ is:

$$\hat{f}(\mathbf{x}) = \alpha + \sum_{k=1}^{D} g_k(x_k) \tag{3.11}$$

GAMs are a particular case of PPR (and as such, they have a neural network implementation; see section 6.1) with $\mathbf{a}_k = \mathbf{e}_k$ and $\mathrm{E}\{g_k(x_k)\} = 0$, $k = 1, \ldots, d$, in eq. (3.8), i.e. the projection directions are fixed to the axis directions and one has to determine the zero-mean ridge functions $\{g_k\}_{i=1}^D$. It is therefore less general because there is no interaction between input variables (e.g. the function $x_1 x_2$ cannot be modelled), but it is more easily interpretable (the functions $g_k(x_k)$ can be plotted); see [47, 48] for some applications. One could add cross-terms of the form $g_{kl}(x_k, x_l)$ to achieve greater flexibility but the combinatorial explosion quickly sets in.

3.8.1 Backfitting

The ridge functions $\{g_k\}_{i=1}^D$ are estimated nonparametrically by the **backfitting** algorithm:

Start with some initial estimates of $\{g_k\}_{i=1}^D$, perhaps obtained parametrically.

Repeat

For $j = 1, \ldots, D$:

1. Compute the current residuals: $r_{i,j} = y_i - \sum_{k \neq j} g_k(x_{k,i})$, $i = 1, \ldots, n$.
2. Fit a nonparametric smooth curve[17] g_j to the residuals $\{r_{i,j}\}_{i=1}^n$ as a function of x_k.

Until some stopping criterion is satisfied.

That is, one keeps iterating over each of the D variables, smoothing the residuals not explained by the $D - 1$ predictors remaining.

The GAM itself is fitted by the local scoring algorithm, a natural generalisation of the iterative least squares procedure for the linear case of (3.11), which applies backfitting.

[17] Hastie and Tibshirani use the running line smoother.

3.8.2 Multivariate adaptive regression splines (MARS)

Multivariate adaptive regression splines (MARS) (Friedman [35]) are an extension to GAMs to allow interactions between variables:

$$\hat{f}(\mathbf{x}) = \alpha + \sum_{k=1}^{d} \prod_{l=1}^{L_k} g_{kl}(x_{\nu(k,l)}) \tag{3.12}$$

In this case, the k-th basis function $g_k = \prod_{l=1}^{L_k} g_{kl}(x_{\nu(k,l)})$ is the product of L_k one-dimensional spline functions g_{kl} each depending on one variable x_ν. The number of factors L_k, the labels $\nu(k,l)$ and the knots for the one-dimensional splines are determined from the data. Basis functions are added incrementally during learning by sequential forward selection.

4 Principal Curves and Principal Surfaces

Principal curves (Hastie and Stuetzle [46]) are smooth 1-D curves that pass through the middle of a p-dimensional data set, providing a nonlinear summary of it. They are estimated in a nonparametric way, i.e. their shape is suggested by the data.

Principal curves can be considered as a generalisation of regression (see fig 6):

- Linear regression minimises the sum of squared deviations in the response variable $y = ax + b$ (i.e. in the vertical direction in an X-Y graph). Thus changing the roles produces a different regression line (dotted line).
- The first principal component is a *regression* line symmetrical with all the variables, minimising orthogonal deviation to that line.
- Nonlinear regression can use a variety of methods (see appendix F.3) to produce a curve that attempts to minimise the vertical deviations (the sum of squared deviations in the response variable) subject to some form of smoothness constraint.
- Principal curves are a natural generalisation for nonlinear, symmetric regression: they attempt to minimise the sum of squared deviations in all the variables (i.e. the orthogonal or shortest distance from the curve to the points) subject to smoothness constraints.

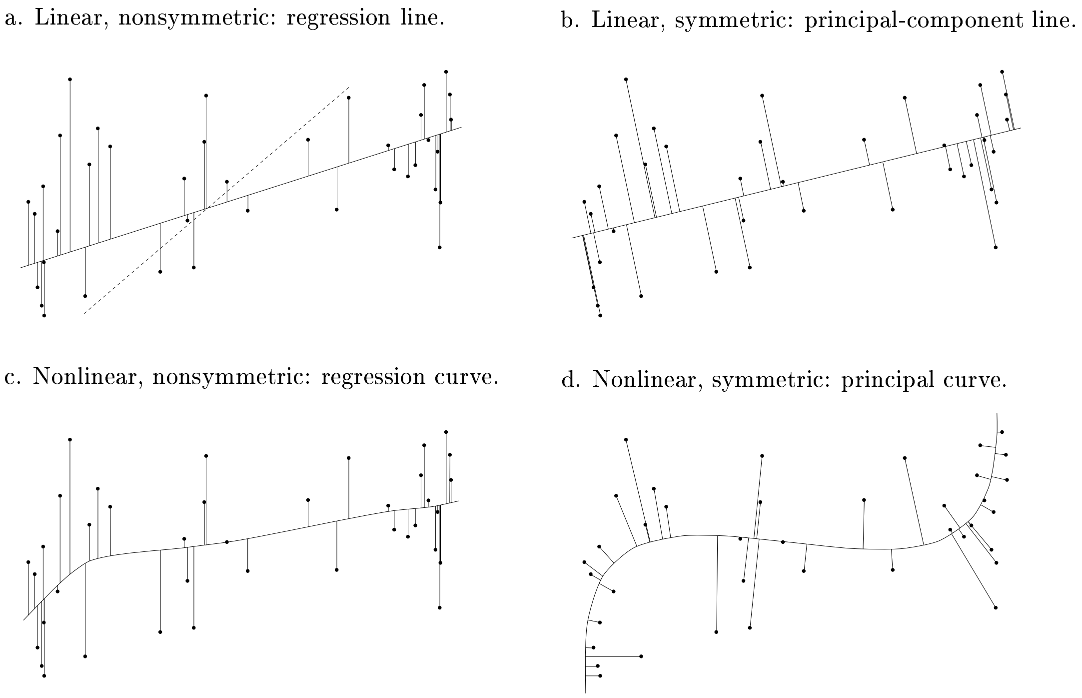

Figure 6: Principal curves as generalised (nonlinear, symmetric) regression. (a) The linear regression line minimises the sum of squared deviations in the response variable (or in the independent one, for the dashed line). (b) The principal-component line minimises the sum of squared deviations in all the variables. (c) The smooth regression curve minimises the sum of squared deviations in the response variable, subject to smoothness constraints. (d) The principal curve minimises the sum of squared deviations in all the variables, subject to smoothness constraints. From Hastie and Stuetzle [46].

We say that a curve is *self-consistent* with respect to a dataset if the average of all data points that project onto a given point on the curve coincides with the point on the curve. More formally, let $\mathbf{f}(\lambda)$ be a smooth curve in $\mathbb{R}^D$ parameterised by its arc length $\lambda \in \mathbb{R}$. For any data point $\mathbf{x}_0 \in \mathbb{R}^D$ we seek the nearest point $\lambda(\mathbf{x}_0) = \lambda_0$ in the curve in Euclidean distance[18], i.e. its orthogonal projection on the curve. If $\mathrm{E}\left\{\mathbf{x}|\lambda(\mathbf{x}) = \lambda_0\right\} = \mathbf{f}(\lambda_0)$ then $\mathbf{f}$ is self-consistent for the distribution $\mathbf{x}$. We can then say that principal curves pass through the middle of the data in a smooth way and are self-consistent for that dataset. This definition poses several questions so far unanswered in the general case:

[18] For definiteness, in the exceptional case where there are several nearest points, we take the one with largest λ.

- For what kinds of distributions do principal curves exist?
- How many different principal curves exist for a given distribution?
- What are their properties?

These questions can be answered for some particular cases:

- For ellipsoidal distributions the principal components are principal curves.
- For spherically symmetric distributions any line through the mean is a principal curve.
- For 2-D spherically symmetric distributions a circle with centre at the mean and radius $\mathrm{E}\{\|\mathbf{x}\|\}$ is a principal curve.

The following properties of principal curves are known:

- For a model of the form $\mathbf{x} = \mathbf{f}(\lambda) + \boldsymbol{\epsilon}$, with $\mathbf{f}$ smooth and $\mathrm{E}\{\boldsymbol{\epsilon}\} = 0$, $\mathbf{f}$ seems not to be a principal curve in general. This means that the principal curve is biased for the functional model, although the bias seems to be small and to decrease to 0 as the variance of the errors gets small relative to the radius of curvature of $\mathbf{f}$.
- If a straight line $\mathbf{f}(\lambda) = \mathbf{u}_0 + \lambda\mathbf{v}_0$ is self-consistent, then it is a principal component (i.e. $\mathbf{u}_0 = 0$ and $\mathbf{v}_0$ is an eigenvector of the covariance matrix). In other words, linear principal curves are principal components.

Principal curves depend critically on the scaling of the features, as all projection techniques do. Current algorithms also depend on the degree of smoothing.

4.1 Construction algorithm

1. Let $j \leftarrow 0$ the iteration index and start with some smooth prior summary of the data $\mathbf{f}_0(\lambda)$ (typically the first principal component).
2. *Projection step*: project the distribution onto the candidate curve $\mathbf{f}_j(\lambda)$.
3. *Averaging step*[19]: let $\mathbf{f}_{j+1}(\lambda_0) = \mathrm{E}\{\mathbf{x}|\lambda(\mathbf{x}) = \lambda_0\}$, the conditional expectation, and reparameterise this in terms of arc length λ.
4. If $\mathbf{f}_j(\lambda(\mathbf{x})) = \mathbf{f}_{j+1}(\lambda(\mathbf{x}))$ for all points $\mathbf{x}$, the curve is self-consistent and thus it is a principal curve; finish.

   Otherwise, take as $\mathbf{f}_{j+1}(\lambda)$ a smooth or local average of the p-dimensional points, where the definition of local is based on the distance in arc length from a fixed point (or any other parameterisation of a 1-D curve) of the projections of the points onto the previous curve; goto 2.

Observe that:

- The construction algorithm converges to the first principal component if conditional expectations are replaced by least-squares straight lines. Principal curves are then local minima of the distance function (sum of squared distances).
- For probability distributions, both operations —projection and average or conditional expectation— reduce the expected distance from the points to the curve; for discrete data sets this is unknown.
- The construction algorithm has not been proven to converge in general.

4.2 Extension to several dimensions: Principal surfaces

Principal curves can be naturally extended to several dimensions and they are then called *principal surfaces*. However, once again the curse of the dimensionality makes smoothing in several dimensions hard unless data are abundant. Further investigation is required to see whether principal surfaces are of interest for very high-dimensional problems.

[19] For discrete data sets, the averaging step has to be estimated (by means of a scatterplot smoother).

5 Topologically Continuous Maps

This term includes several related techniques mainly used for visualisation of high-dimensional data, of which the most well-known are Kohonen's self-organising maps. The objective of these techniques is to learn in an unsupervised manner a mapping from a space of fixed dimension (sometimes called lattice or latent space) onto the high-dimensional data space that embeds the data distribution. Therefore, the common element in them is the concept of *topological* or *topographic* map, which basically means a *continuous* mapping, i.e. a mapping that assigns nearby images in the codomain to nearby points in the domain.

The term "topologically continuous maps" is not very appropriate, because it has some undesirable connotations in the mathematical theory of topological spaces; the similar term "topographic maps" is not very satisfactory either. "Self-organising maps," although somewhat vague, would probably be a better election, but unfortunately it is too linked to Kohonen's maps in the literature.

5.1 Kohonen's self-organising maps

Let $\{\mathbf{t}_n\}_{n=1}^N$ a sample in the data space $\mathbb{R}^D$. Kohonen's self-organising maps (SOMs) [69] can be considered as a form of dimension reduction in the sense that they learn, in an unsupervised way, a mapping between a 2-D lattice[20] and the data space. The mapping preserves the two-dimensional topology of the lattice when adapting to the manifold spanned by the sample. One can visualise the learning process as a plane sheet that twists around itself in D dimensions to resemble as much as possible the distribution of the data vectors.

5.1.1 The neighbourhood function

In a SOM, like in vector quantisation [41], we have a set of **reference** or **codebook vectors** $\{\boldsymbol{\mu}_i\}_{i=1}^M$ in data space $\mathbb{R}^D$, initially distributed at random[21], but each of them is associated to a node i in a 2-D lattice —unlike in vector quantisation, in which no topology exists. Assume we have defined two distances (typically Euclidean) d_D and d_L in the data space and in the lattice, respectively. The topology of the lattice is determined by the **neighbourhood function** h_{ij}. This is a symmetric function with values in $[0, 1]$ that behaves as an "inverse" distance in the lattice: given a node i, $h_{ii} = 1$ for any node i and for any other node j, h_{ij} is smaller the farther apart node j is from node i in the lattice. The neighbourhood of node i is composed of those nodes for which h_{ij} is not negligibly small. In practice, usually $h_{ij} = \exp(-d_L^2(i, j)/2\sigma^2)$, where σ would quantify the range of the neighbourhood.

5.1.2 Kohonen learning[22]

A competitive learning rule is applied iteratively over all data vectors until convergence is achieved. Given a data vector $\mathbf{t}_n$, let $\boldsymbol{\mu}_{i^*}$ be the reference vector closest to $\mathbf{t}_n$ in data space:

$$i^* = \arg \min_{j \in \text{lattice}} d_D(\boldsymbol{\mu}_j, \mathbf{t}_n).$$

Learning occurs as follows (where t is the iteration index and $\alpha^{(t)} \in [0, 1]$ is the learning rate):

$$\boldsymbol{\mu}_i^{\text{new}} = \boldsymbol{\mu}_i^{\text{old}} + \alpha^{(t)} h_{i^* i}^{(t)} (\mathbf{t}_n - \boldsymbol{\mu}_i^{\text{old}}) = (1 - \rho)\boldsymbol{\mu}_i^{\text{old}} + \rho \mathbf{t}_n \tag{5.1}$$

i.e. reference vector $\boldsymbol{\mu}_i$ is drawn a distance $\rho = \alpha^{(t)} h_{i^* i}^{(t)}$ toward data vector $\mathbf{t}_n$. The update affects only vectors whose associated nodes lie in the neighbourhood of the winner i^* and its intensity decreases with the iteration index t because both $\alpha^{(t)}$ and the range of $h^{(t)}$ must decrease with t (for convergence considerations).

Intuitively one sees that the reference vectors $\boldsymbol{\mu}_i$ will become dense in regions of $\mathbb{R}^D$ where the $\mathbf{t}_n$ are common and sparse where the $\mathbf{t}_n$ are uncommon, thus replicating the distribution of the data vectors.

5.1.3 Summary

Kohonen learning creates an L-dimensional arrangement such that:

[20] In the typical case, but the idea is valid for L-dimensional topological arrangements.
[21] Or perhaps one can take a random set of the data vectors, or the first principal components.
[22] This is online learning. A batch version also exists.

- The number density of reference vectors in data space is approximately proportional to the data probability density.
- The mapping from the L-dimensional arrangement into data space is topologically continuous.

5.1.4 Disadvantages

KSOMs have proven successful in many practical applications, particularly in visualisation. However, due to their heuristic nature they have a number of shortcomings:

- No cost function to optimise can be defined.
- No schedules for selecting $\alpha^{(t)}$ and $h^{(t)}$ exist that guarantee convergence in general.
- No general proofs of convergence exist.
- No probability distribution function (generative model for the data) is obtained. The L-dimensional manifold in data space is defined indirectly by the location of the reference vectors; for intermediate points one has to interpolate.

5.2 Generative modelling: density networks

In **generative modelling**, all observables in the problem are assigned a probability distribution to which the Bayesian machinery is applied. Density networks (MacKay [74]) are a form of Bayesian learning that attempts to model data in terms of latent variables [29]. First we will introduce Bayesian neural networks, then the density networks themselves and we will conclude with GTM, a particular model based in density networks.

5.2.1 Bayesian neural networks

Assume we have data $\mathcal{D}$ which we want to model using parameters $\mathbf{w}$ and define the likelihood $L(\mathbf{w}) = p(\mathcal{D}|\mathbf{w})$ as the probability of the data given the parameters. Learning can be classified into [76]:

- **Traditional (frequentist)**, e.g. the MLP: no distribution over the parameters $\mathbf{w}$ is assumed; we are interested in a single value $\mathbf{w}^*$ which is often found as a maximum likelihood estimator:

  $$\mathbf{w}^* = \arg\max_{\mathbf{w}} \{\ln L(\mathbf{w}) + R(\mathbf{w})\} \tag{5.2}$$

  where $R(\mathbf{w})$ is a regularisation term, such as the quadratic regulariser with (hyper)parameter α

  $$R(\mathbf{w}) = \frac{1}{2}\alpha \sum_i \mathbf{w}_i^2. \tag{5.3}$$

  New data $\mathcal{D}'$ is predicted as $p(\mathcal{D}'|\mathbf{w}^*)$.

- **Bayesian**, e.g. density networks: a probability distribution over the model parameters is obtained, based in a *prior* distribution $p(\mathbf{w})$ that expresses our initial belief about their values, before any data has arrived. Given data $\mathcal{D}$, we update this prior to a *posterior* distribution using Bayes' rule:

  $$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \propto L(\mathbf{w})p(\mathbf{w}). \tag{5.4}$$

  New data $\mathcal{D}'$ is predicted as

  $$p(\mathcal{D}'|\mathcal{D}) = \int p(\mathcal{D}'|\mathbf{w})p(\mathbf{w}|\mathcal{D})\, d\mathbf{w}$$

Traditional learning can be viewed as a maximum a posteriori probability (MAP) estimate of Bayesian learning (cf. eq (5.2)):

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \ln\{L(\mathbf{w})p(\mathbf{w})\} = \arg\max_{\mathbf{w}} \{\ln L(\mathbf{w}) + \ln p(\mathbf{w})\}$$

with a prior $p(\mathbf{w}) = \exp(R(\mathbf{w}))$. For the quadratic regulariser (5.3) the prior would be proportional to a Gaussian density with variance $1/\alpha$.

The Bayesian approach presents the advantage over the frequentist one of finding a full distribution for the parameters. However, this is earned at the expense of introducing the prior, whose selection is often criticised as being arbitrary.

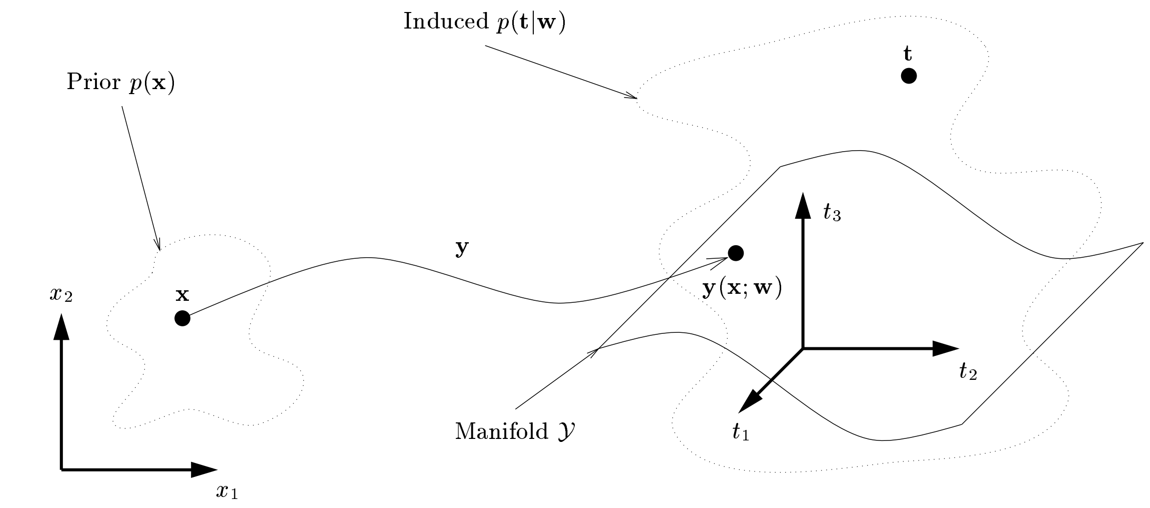

Figure 7: "Jump" from the latent space into the data space.

5.2.2 Density networks

We want to model a certain distribution $p(\mathbf{t})$ in data space $\mathbb{R}^D$, given a sample $\{\mathbf{t}_n\}_{n=1}^N$ drawn independently from it, in terms of a small number L of latent variables. The likelihood and log-likelihood (MacKay calls them *evidence* and *log-evidence*) are:

$$L(\mathbf{w}) = p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{t}_n|\mathbf{w}) \qquad l(\mathbf{w}) = \ln p(\mathcal{D}|\mathbf{w}) = \sum_{n=1}^{N} \ln p(\mathbf{t}_n|\mathbf{w}).$$

We define the following functions in some convenient way for the problem being considered:

- A prior distribution on L-dimensional latent space $\mathbb{R}^L$: $p(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^L$.
- A smooth mapping $\mathbf{y}$ from the latent space onto an L-dimensional[23] manifold $\mathcal{Y}$ in data space, with parameters $\mathbf{w}$ (for example, if $\mathbf{y}$ is an MLP, $\mathbf{w}$ would be the weights and biases):

$$\begin{array}{rccc} \mathbf{y}: & \mathbb{R}^L & \longrightarrow & \mathcal{Y} \subset \mathbb{R}^D \\ & \mathbf{x} & \longmapsto & \mathbf{y}(\mathbf{x};\mathbf{w}) \end{array}$$

  This jump from L to D dimensions is the key for the dimension reduction.

- The error functions $G_n(\mathbf{x};\mathbf{w}) = \ln p(\mathbf{t}_n|\mathbf{x},\mathbf{w}) = \ln p(\mathbf{t}_n|\mathbf{y})$. For example:
  - The *softmax* classifier for a classification problem with I classes:

$$p(t = i|\mathbf{x},\mathbf{w}) = y_i(\mathbf{x};\mathbf{w}) = \frac{e^{f_i(\mathbf{x};\mathbf{w})}}{\sum_{j=1}^{I} e^{f_j(\mathbf{x};\mathbf{w})}}$$

    where $\{f_i\}_{i=1}^I$ are functions of $\mathbf{x}$ parameterised by $\mathbf{w}$.
  - The linear logistic model (single sigmoidal neuron) in a binary classification problem:

$$p(t = 1|\mathbf{x},\mathbf{w}) = y_1(\mathbf{x};\mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}}}$$

    which is a particular case of the softmax classifier.
  - A function of the Euclidean squared distance $\|\mathbf{t}_n - \mathbf{y}\|^2$, as in the GTM model, eq. (5.9).

Using Bayes' rule, we can compute the posterior in latent space:

$$p(\mathbf{x}|\mathbf{t}_n,\mathbf{w}) = \frac{p(\mathbf{t}_n|\mathbf{x},\mathbf{w})p(\mathbf{x})}{p(\mathbf{t}_n|\mathbf{w})} \tag{5.5}$$

[23] Perhaps less than L-dimensional if the mapping is singular.

with normalisation constant

$$p(\mathbf{t}_n|\mathbf{w}) = \int p(\mathbf{t}_n|\mathbf{x},\mathbf{w})p(\mathbf{x})\,d\mathbf{x}. \tag{5.6}$$

Applying Bayes' rule again (5.4), we find the posterior in parameter space:

$$p(\mathbf{w}|\{\mathbf{t}_n\}_{n=1}^N) = \frac{p(\{\mathbf{t}_n\}_{n=1}^N|\mathbf{w})p(\mathbf{w})}{p(\{\mathbf{t}_n\}_{n=1}^N)}$$

with $p(\{\mathbf{t}_n\}_{n=1}^N|\mathbf{w}) = \prod_{n=1}^N p(\mathbf{t}_n|\mathbf{w}) = L(\mathbf{w})$. Learning of the parameters $\mathbf{w}$ by maximum likelihood can take place using gradient descent on the log-likelihood; from eq. (5.6):

$$\nabla_{\mathbf{w}}\{\ln p(\mathbf{t}_n|\mathbf{w})\} = \frac{1}{p(\mathbf{t}_n|\mathbf{w})}\int e^{G_n(\mathbf{x};\mathbf{w})}p(\mathbf{x})\nabla_{\mathbf{w}}G_n(\mathbf{x};\mathbf{w})\,d\mathbf{x} = \\ \int p(\mathbf{x}|\mathbf{t}_n,\mathbf{w})\nabla_{\mathbf{w}}G_n(\mathbf{x};\mathbf{w})\,d\mathbf{x} = \mathrm{E}_{p(\mathbf{x}|\mathbf{t}_n,\mathbf{w})}\left\{\nabla_{\mathbf{w}}G_n(\mathbf{x};\mathbf{w})\right\} \tag{5.7}$$

i.e. the gradient of the log-likelihood is the expectation of the traditional backpropagation gradient $\nabla_{\mathbf{w}}G_n(\mathbf{x};\mathbf{w})$ over the posterior $p(\mathbf{x}|\mathbf{t}_n,\mathbf{w})$ of eq. (5.5).

From the computational point of view, integral (5.6) is analytically difficult for most priors $p(\mathbf{x})$. The log-likelihood and its derivatives can be approximated by Monte Carlo sampling[24]:

$$l(\mathbf{w}) = \sum_{n=1}^N \ln\int e^{G_n(\mathbf{x};\mathbf{w})}p(\mathbf{x})\,d\mathbf{x} \approx \sum_{n=1}^N \ln\frac{1}{R}\sum_{r=1}^R e^{G_n(\mathbf{x}_r;\mathbf{w})} \tag{5.8a}$$

$$\nabla_{\mathbf{w}}l(\mathbf{w}) \approx \sum_{n=1}^N \frac{\sum_{r=1}^R e^{G_n(\mathbf{x}_r;\mathbf{w})}\nabla_{\mathbf{w}}G_n(\mathbf{x};\mathbf{w})}{\sum_{r=1}^R e^{G_n(\mathbf{x}_r;\mathbf{w})}} \tag{5.8b}$$

where $\{\mathbf{x}_r\}_{r=1}^R$ would be random samples from $p(\mathbf{x})$, but still this is a very costly process because the sampling depends exponentially on the dimension of the integral, L.

Summary Density networks provide a framework for generative modelling which can be adapted to specific problems by conveniently selecting:

- The dimension of the latent space L.
- The prior in latent space $p(\mathbf{x})$, as simple as possible to allow easy integration of the error function.
- The smooth mapping $\mathbf{y}(\mathbf{x};\mathbf{w})$, if possible differentiable on $\mathbf{w}$ to allow using standard optimisation techniques.
- The error function $p(\mathbf{t}|\mathbf{x};\mathbf{w})$.
- The optimisation algorithm for maximising the posterior in parameter space $p(\mathbf{w}|\{\mathbf{t}_n\}_{n=1}^N)$.

Relationship with MLPs and autoassociators In MLPs, the outputs are conditioned over the values of the input variables and there is no density model of the input variables. Conversely, in density modelling (or generative modelling) a density of all observable quantities is constructed. Target outputs are specified, but not inputs.

Density networks can be considered as the *generative* half of an autoassociator (from the bottleneck to the output). The *recognition* mapping (from the input to the bottleneck), responsible for the feature extraction, plays no role in the probabilistic model (see fig. 10).

5.2.3 Generative topographic mapping (GTM)

The generative topographic mapping (GTM), put forward by Bishop, Svensén and Williams [7] as a principled view of Kohonen's SOMs, is a density network based on a constrained Gaussian mixture model and trained with the EM algorithm. No biological motivation is intended.

In GTM, the dimension of the latent space is assumed small (usually 2) and the various features of the general framework of density networks are selected as follows:

[24] $\int Q(x)p(x)\,dx \approx \frac{1}{K}\sum_{i=1}^K Q(x_i)$ for a sample $\{x_i\}_{i=1}^K$ of the pdf $p(x)$.

- Error functions $G_n(\mathbf{x}; \mathbf{w})$: $p(\mathbf{t}|\mathbf{x}, \mathbf{w})$ is a spherical Gaussian $\mathcal{N}(\mathbf{y}(\mathbf{x}; \mathbf{W}), \beta^{-1}\mathbf{I})$ centred on $\mathbf{y}(\mathbf{x}; \mathbf{W})$ with variance β^{-1} (for convenience of notation, we separate the parameters $\mathbf{w}$ into the mapping parameters $\mathbf{W}$ and the variance parameter β):

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta) = \left(\frac{\beta}{2\pi}\right)^{\frac{D}{2}} \exp\left(-\frac{\beta}{2}\|\mathbf{y}(\mathbf{x}; \mathbf{W}) - \mathbf{t}\|^2\right) \tag{5.9}$$

which behaves as a noise model for $\mathbf{y}(\mathbf{x}; \mathbf{W})$ that *extends* the manifold $\mathcal{Y}$ to $\mathbb{R}^D$: a given data vector $\mathbf{t}$ could have been generated by any point $\mathbf{x}$ with probability $p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \beta)$.

- Prior in latent space:

$$p(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} \delta(\mathbf{x} - \mathbf{x}_i) \tag{5.10}$$

where $\{\mathbf{x}_i\}_{i=1}^{K}$ stand on a regular grid[25] in latent space. This prior choice is equivalent to a Monte Carlo approximation of (5.6) for an arbitrary $p(\mathbf{x})$. Then, from eq. (5.6):

$$p(\mathbf{t}|\mathbf{W}, \beta) = \frac{1}{K}\sum_{i=1}^{K} p(\mathbf{t}|\mathbf{x}_i, \mathbf{W}, \beta) \tag{5.11}$$

is a constrained mixture of Gaussians (because the Gaussians lie on the L-dimensional manifold $\mathcal{Y}$ in data space), and

$$l(\mathbf{W}, \beta) = \sum_{n=1}^{N} \ln\left\{\frac{1}{K}\sum_{i=1}^{K} p(\mathbf{t}_n|\mathbf{x}_i, \mathbf{W}, \beta)\right\}$$

which is just eq. (5.8a). If $\mathbf{y}$ was linear and $p(\mathbf{x})$ Gaussian, then $p(\mathbf{t})$ would also be Gaussian and GTM would reduce to a particular case of *factor analysis* in which all the variances are equal (to β^{-1}).

- Mapping selection: *generalised linear model* $\mathbf{y}(\mathbf{x}; \mathbf{W}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{x})$ with $\mathbf{W}$ a $D \times M$ matrix of weights and $\boldsymbol{\phi}$ an $M \times 1$ vector of basis functions[26] (see appendix G). This turns the M step of the optimisation into a matrix equation, see (5.13).

  Typically, Gaussians with explicitly set parameters are used as basis functions:

  - Centres drawn from the grid in latent space.
  - Variances chosen accordingly, based on the known distance between centres.

  Although the actual shape of the basis functions is probably unimportant as long as they are localised.

  Generalised linear networks are universal approximators, but their use limits the dimension L of the latent space because the number of basis functions M required to attain a given accuracy grows exponentially with L. This can be a serious disadvantage in some cases.

  The number of sample points $\mathbf{x}_i$, K, must be bigger than M in order to obtain a smooth mapping $\mathbf{y}$; in [7] it is suggested to take $\frac{K}{M} = \mathcal{O}(100)$. Because K is not a parameter of the model, increasing it does not favour overfitting.

- The model parameters $\mathbf{W}$ and β are determined by maximum likelihood, which provides us with the following objective function:

$$l(\mathbf{W}, \beta) = \ln \prod_{n=1}^{N} p(\mathbf{t}_n|\mathbf{W}, \beta) = \sum_{n=1}^{N} \ln p(\mathbf{t}_n|\mathbf{W}, \beta) \tag{5.12}$$

to which a regularisation term (a prior on the parameters $\mathbf{W}$) could be added to control the mapping $\mathbf{y}$. The mixture distribution (5.11) suggests using the EM algorithm to maximise the log-likelihood:

[25] The reason to position the points $\mathbf{x}_i$ in the nodes of a regular grid is to facilitate visualisation of the posterior (5.15) in a computer screen.

[26] An additional column with biases w_{k0} can be added to $\mathbf{W}$, and consequently an additional fixed component $\phi_0(\mathbf{x}_i) = 1$ to $\boldsymbol{\phi}$.

E step: computation of responsibilities R_{in}, see eq. (5.14).

M step[27]: taking partial derivatives of L w.r.t. the parameters $\mathbf{W}$ and β one obtains:

– A matrix equation[28,29] for $\mathbf{W}$:

$$\mathbf{\Phi}^T \mathbf{G}^{\text{old}} \mathbf{\Phi} (\mathbf{W}^{\text{new}})^T = \mathbf{\Phi}^T \mathbf{R}^{\text{old}} \mathbf{T} \tag{5.13}$$

solvable for $\mathbf{W}^{\text{new}}$ using standard matrix inversion techniques.

– A reestimation formula for β:

$$\frac{1}{\beta} = \frac{1}{ND} \sum_{n=1}^{N} \sum_{i=1}^{K} R_{in}(\mathbf{W}, \beta) \| \mathbf{y}(\mathbf{x}_i; \mathbf{W}) - \mathbf{t}_n \|^2.$$

Where:

– $\mathbf{\Phi} = (\phi_{ij})$ is a $K \times M$ matrix of constants: $\phi_{ij} = \phi_j(\mathbf{x}_i)$.

– $\mathbf{T} = (\mathbf{t}_1 \dots \mathbf{t}_N)^T$ is an $N \times D$ matrix of constants.

– $\mathbf{R} = (R_{in})$ is a $K \times N$ matrix of *posterior probabilities* or *responsibilities*:

$$R_{in}(\mathbf{W}, \beta) = p(\mathbf{x}_i | \mathbf{t}_n, \mathbf{W}, \beta) = \frac{p(\mathbf{t}_n | \mathbf{x}_i, \mathbf{W}, \beta)}{\sum_{i'=1}^{K} p(\mathbf{t}_n | \mathbf{x}_{i'}, \mathbf{W}, \beta)} \tag{5.14}$$

.

– $\mathbf{G}$ is a diagonal $K \times K$ matrix with elements $g_{ii}(\mathbf{W}, \beta) = \sum_{n=1}^{N} R_{in}(\mathbf{W}, \beta)$.

Because the EM algorithm increases the log-likelihood monotonically [23], the convergence of GTM is guaranteed. According to [7], convergence is usually achieved after a few tens of iterations. As initial weights one can take the first L principal components of the sample data $\{\mathbf{t}_n\}$. After convergence, the value of $1/\beta^*$ should be small for the approximation to be good.

An *online version* of the M step is obtained by decomposing the objective function over the data points, $l = \sum_{n=1}^{N} l_n$, and using the Robbins-Monro procedure [85]:

$$w_{kj}^{(t+1)} = w_{kj}^{(t)} + \alpha^{(t)} \left(\frac{\partial l_n}{\partial w_{kj}} \right)^{(t)} \qquad \beta^{(t+1)} = \beta^{(t)} + \alpha^{(t)} \left(\frac{\partial l_n}{\partial \beta} \right)^{(t)}.$$

If the learning rate $\alpha^{(t)}$ is an appropriately decreasing function of the iteration step t, convergence to an extremum of l is assured.

Posterior probabilities and visualisation Fixed $\mathbf{W}^*$, β^* and given a point $\mathbf{t}_n$ in data space, eq. 5.4 gives the posterior distribution for $\mathbf{x}_i$, which can be considered as the "inverse" mapping of $\mathbf{y}$:

$$p(\mathbf{x}_i | \mathbf{t}_n, \mathbf{W}^*, \beta^*) = \frac{p(\mathbf{t}_n | \mathbf{x}_i, \mathbf{W}^*, \beta^*)}{\sum_{i'=1}^{K} p(\mathbf{t}_n | \mathbf{x}_{i'} \mathbf{W}^*, \beta^*)} = R_{in} \tag{5.15}$$

i.e. the responsibility of $\mathbf{x}_i$ given $\mathbf{t}_n$. Thus, while many visualisation techniques provide just a single *responsible* point $\mathbf{x}_i$ for $\mathbf{t}_n$, GTM provides a full posterior distribution $p(\mathbf{x}_i | \mathbf{t}_n)$, as a result of the Bayesian approach. This can be a considerable amount of data; to simplify it, one can summarise this distribution by its mean, but this can be misleading if it is multimodal ([7] gives actual examples of this situation).

Magnification factors The local magnification factor, i.e. the relation between the volume elements in the manifold $\mathcal{Y}$ and in the latent space, can be computed analytically for GTM [8]:

$$\frac{dV'}{dV} = \sqrt{\det(\mathbf{\Psi}^T \mathbf{W}^T \mathbf{W} \mathbf{\Psi})} \qquad \psi_{jk} = \frac{\partial \phi_j}{\partial x_k} \qquad dV = \prod_{k=1}^{D} dx_k$$

The local magnification factor may contain information on the clustering properties of the data, as it seems to be small near the center of a cluster and high in the cluster boundaries. Therefore it could be useful to highlight boundaries.

[28] Or, using the centroid $K \times D$ matrix $\mathbf{M} = (\mu_{ik})$ instead of the responsibility matrix, one has $\mathbf{RT} = \mathbf{GM}$. The *centroids* are defined as:

$$\boldsymbol{\mu}_i = \frac{\sum_{n=1}^{N} R_{in} \mathbf{t}_n}{\sum_{n=1}^{N} R_{in}}.$$

[29] If a quadratic regulariser (5.3) is added to $l(\mathbf{W}, \beta)$, equation (5.13) becomes:

$$(\mathbf{\Phi}^T \mathbf{G}^{\text{old}} \mathbf{\Phi} - \frac{\alpha}{\beta} \mathbf{I})(\mathbf{W}^{\text{new}})^T = \mathbf{\Phi}^T \mathbf{R}^{\text{old}} \mathbf{T}$$

for a constant regularisation (hyper)parameter α. Notice that equation (15) in [7] is incorrect.

	SOM	GTM
Internal representation of manifold	Nodes $\{i\}$ in L-dimensional array, held together by neighbourhood function h	Point grid $\{\mathbf{x}_i\}$ in L-dimensional latent space that keeps its topology through smooth mapping $\mathbf{y}$
Definition of manifold in data space	Indirectly by locations of reference vectors	By mapping $\mathbf{y}$
Objective function	No	Yes, based on log-likelihood
Self-organisation	Difficult to quantify	Smooth mapping $\mathbf{y}$ preserves topology
Convergence	Not guaranteed	Yes. Batch: EM algorithm; online: Robbins-Monro
Smoothness of manifold	Depends on $\alpha^{(t)}$ and $h^{(t)}$	Depends on basis functions parameters and prior distribution $p(\mathbf{x})$
Generative model	No; hence no density function	Yes, $p(\mathbf{t}\|\mathbf{W}, \beta)$
Additional parameters to select	$\alpha^{(t)}$, $h^{(t)}$ arbitrarily	$\alpha^{(t)}$ annealed by Robbins-Monro schedules (in online version)
Speed of training	Comparable (according to [7])	
Magnification factors	Approximated by the difference between reference vectors	Exactly computable anywhere

Table 1: Comparison between GTM and Kohonen's SOM.

Extensions to GTM The basic generative model of GTM can be extended to:

- Deal with missing values (unobserved components in the data vectors) if they are missing at random, because the objective function L can be obtained by integrating out the unobserved values.
- Mixtures of GTM models of the form $p(\mathbf{t}) = \sum_l p(l)p(\mathbf{t}|l)$, where $p(\mathbf{t}|l)$ is the lth model (with its independent set of parameters) and $p(l)$ a discrete distribution that provides the mixing coefficients of the model. The mixture is, again, trainable by maximum likelihood using the EM algorithm.

Conclusions GTM is effective for visualisation purposes, and has the advantage over Kohonen's SOMs of having a solid theoretical basis (see table 1 for a comparison between GTM and KSOMs). But as it stands, it doesn't seem suitable for *hard* dimension reduction problems, in which the dimension of the latent space will be large enough as to make the number of basis functions M exceedingly big. Perhaps a different choice of the mapping able to cope with the curse of the dimensionality could overcome this difficulty.

And, as with any dimension reduction technique, there is no insight about what the dimension L of the latent space should be. In GTM, the choice of the latent space is driven by:

- Prior knowledge about the problem. For example, in the problem of multiphase flows in oil pipelines to which Bishop *et al.* apply GTM [6], it is known in advance that the points in data space have been generated by a random process with only two degrees of freedom —even though the dimension of the data space is 12.
- Other constraints, e.g. the fact that the dimension of the latent space must not exceed 2 or 3 for visualisation.

6 Neural Network Implementation of Some Statistical Models

In this section, we will see that many of the algorithms of the previous sections can be implemented in a natural way by neural networks, with a clever election of the architecture and learning algorithm. We will also present the implementation of some new projection indices (e.g. Intrator's BCM neuron with objective function [59, 58, 57]).

6.1 Architectures based on a two-layer perceptron

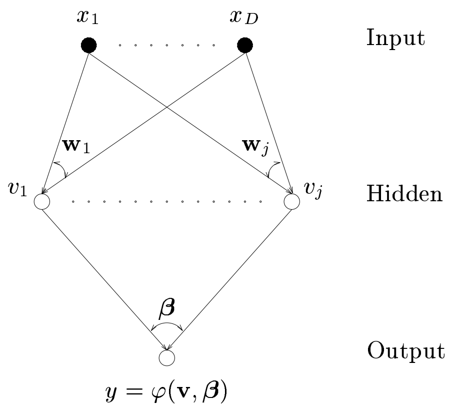

Figure 8: Two-layer perceptron with one output unit.

Consider the two-layer (nonlinear) perceptron of fig. 8. It has D input units, j hidden units and only one output unit. The activation function for each unit, g_k, is (in general) different. The output is a general function of the activation of the hidden layer, $\mathbf{v}$, and the weights of the second layer, $\boldsymbol{\beta}$: $y = f(\varphi(\mathbf{v}, \boldsymbol{\beta}))$. The following particular cases of this network implement several of the techniques previously mentioned [12]:

- Projection pursuit regression (cf. eq. (3.8)):

$$\varphi(\mathbf{v}, \boldsymbol{\beta}) = \mathbf{v}^T \mathbf{1}, \quad v_k = g_k(\mathbf{w}_k^T \mathbf{x}) \quad \Rightarrow \quad y = \sum_{k=1}^{j} g_k(\mathbf{w}_k^T \mathbf{x}).$$

  The activation functions g_k are determined from the data during training; $\mathbf{w}_k$ represent the projection directions.

  Incidentally, this shows that continuous functions g_k can be uniformly approximated by a sigmoidal MLP of one input. Therefore, the approximation capabilities of MLPs and PP are very similar [25, 63].

  This architecture admits generalisations to several output variables [84] depending on whether the output share the common "basis functions" g_k and, if not, whether the separate g_k share common projection directions $\mathbf{w}_k$.

- Generalised additive models:

$$j = D, \quad w_{k0} = 0, \quad \mathbf{w}_k = \mathbf{e}_k, \quad \varphi(\mathbf{v}, \boldsymbol{\beta}) = \mathbf{v}^T \mathbf{1} + w_0 \quad \Rightarrow \quad y = w_0 + \sum_{k=1}^{D} g_k(x_k)$$

- Kernel estimation (section E.3.3), using radial basis functions:

$$y = w_0 + \sum_{k=1}^{j} w_k \tau^{-D} \varphi\left(\frac{\|\mathbf{x} - \boldsymbol{\mu}_k\|}{\tau}\right)$$

  where φ is a radial basis function (spherically symmetric, e.g. a multivariate Gaussian density) centred in $\boldsymbol{\mu}_k \in \mathbb{R}^D$ and with scale parameter τ. There are variations based on regularisation and wavelets.

6.2 Principal component analysis networks

There exist several neural network architectures capable to extract PCs (see fig. 9), which can be classified in (see [26] or [11] for more details):

- *Autoassociators* (also called autoencoders, bottlenecks or n-h-n networks), which are linear two-layer perceptrons with n inputs, h hidden units and n outputs, trained to replicate the input in the output layer minimising the squared sum of errors, and typically trained with backpropagation. Bourlard and Kamp [9] and Baldi and Hornik [2] showed that this network finds a basis of the subspace spanned by the first h PCs, not necessarily coincident with them[30]; see [11, 15] for applications.
- Networks based in Oja's rule [77] with some kind of decorrelating device (e.g. Kung and Diamantaras' APEX [72], Földiák's network [32], Sanger's Generalised Hebbian Algorithm [88]).

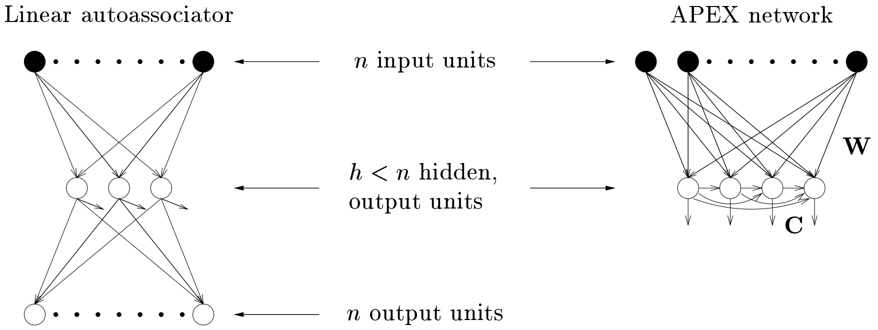

Figure 9: Two examples of neural networks able to perform a principal component analysis of its training set: left, a linear autoassociator, trained by backpropagation; right, the APEX network, with Hebbian weights $\mathbf{W}$ and anti-Hebbian, decorrelating weights $\mathbf{C}$. In both cases, the number of hidden units determines how many principal components are kept.

Linear autoassociators can be extended by including nonlinear activation functions and several layers (see fig. 10). Surprisingly, this approach has found little success, possibly due to the difficulty to train this network.

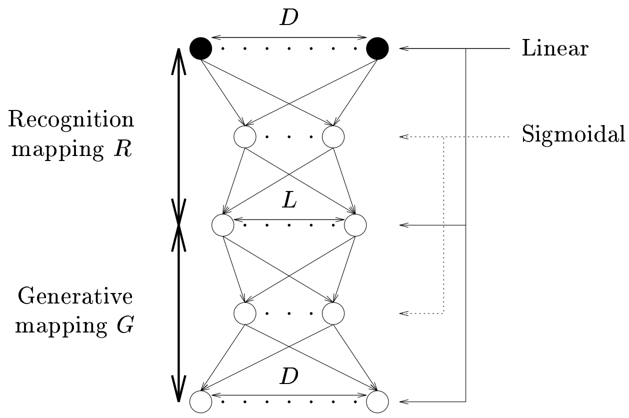

Figure 10: Autoencoder, implemented as a four-layer nonlinear perceptron where $L < D$ and $\hat{\mathbf{x}} = G(R(\mathbf{x}))$.

6.3 Projection pursuit learning network (PPLN)

Hwang *et al.* [54] propose the two-layer perceptron depicted in fig. 11 with a projection pursuit learning (PPL) algorithm to solve the multivariate nonparametric regression problem of section F.1. The outputs

[30] If necessary, the true first principal directions can be obtained from the projections of the data on this basis by other method (e.g. numerical methods, or any of the other PCA networks). The computational effort for this second step is now much smaller, as the number of dimensions should have decreased substantially compared to the original one. However, in many applications, any basis —not necessarily the principal components one— will be enough for practical purposes.

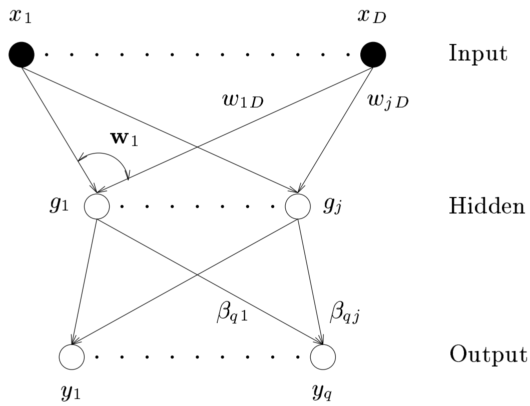

Figure 11: Two-layer perceptron with several output units.

of this perceptron can be expressed as:

$$y_l = \sum_{k=1}^{j} \beta_{lk} g_k(\mathbf{x}^T \mathbf{w}_k - w_{k0}) = \hat{g}_l(\mathbf{x}) = \sum_{k=1}^{j} \beta_{lk} g_k \left(\frac{\mathbf{x}^T \mathbf{a}_k - \boldsymbol{\mu}_k}{s_k} \right), \quad l = 1, \dots, q$$

where $\mathbf{a}_k = \frac{\mathbf{w}_k}{\|\mathbf{w}_k\|}$ and w_{k0} is the bias of the k-th hidden unit. The argument of g_k, $\frac{\mathbf{x}^T \mathbf{a}_k - \boldsymbol{\mu}_k}{s_k}$, is a translated ($\boldsymbol{\mu}_k$) and scaled ($s_k$) one-dimensional projection.

In backprojection learning (BPL), the g_k are usually sigmoids and all the weights in the network are estimated simultaneously by gradient descent. Hwang *et al.* propose a projection pursuit learning algorithm based on PPR (see section 3.7) that trains the weights *sequentially* (i.e. it can be considered that neurons are added to the hidden layer one at a time):

- $\hat{y}_l = \mathrm{E}\{y_l|\mathbf{x}\} = \mathrm{E}\{y_l\} + \sum_{k=1}^{j} \beta_{lk} g_k(\mathbf{x}^T \mathbf{a}_k)$ with $\mathrm{E}\{y_l\} = \frac{1}{n} \sum_{i=1}^{n} y_{il}$, for $l = 1, \dots, q$, and $\mathrm{E}\{g_k\} = 0$, $\mathrm{E}\{g_k^2\} = 1$ and $\|\mathbf{a}_k\| = 1$, for $k = 1, \dots, j$. $\{g_k\}_{k=1}^{j}$ are unknown smooth functions.

- **Estimate** projection directions $\{\mathbf{a}_k\}$, projection strengths $\{\beta_{lk}\}$ and unknown smooth functions $\{g_k\}$ via LS minimisation of the loss function

$$L_2(\{\mathbf{a}_k\}, \{\beta_{lk}\}, \{g_k\}) = \sum_{l=1}^{q} W_l \, \mathrm{E}\left\{(y_l - \hat{y}_l)^2\right\} = \sum_{l=1}^{q} W_l \, \mathrm{E}\left\{ \left(y_l - \mathrm{E}\{y_l\} - \sum_{k=1}^{j} \beta_{lk} g_k(\mathbf{x}^T \mathbf{a}_k) \right)^2 \right\}$$

  where $\{W_l\}_{l=1}^{q}$ are response weightings to specify relative contributions of each output (typically $W_l = 1/\operatorname{var}\{y_l\}$, where the variances are either known or, more frequently, estimated from the data). Expressing L_2 as function of the residual functions $R_{l(k)} = y_l - \mathrm{E}\{y_l\} - \sum_{k \neq m}^{j} \beta_{lm} g_m(\mathbf{x}^T \mathbf{a}_m)$:

$$L_2 = \sum_{l=1}^{q} W_l \, \mathrm{E}\left\{(R_{l(k)} - \beta_{lk} g_k(\mathbf{x}^T \mathbf{a}_k))^2\right\}$$

- **Learn** unit by unit and layer by layer cyclically after all patterns are presented according to the following algorithm:

For each unit $k = 1, \dots, j$:
  Assign initial guesses to $\mathbf{a}_k$, g_k and $\{\beta_{lk}\}_{l=1}^{q}$.
  Repeat
    Repeat "several times"
      Estimate iteratively $\mathbf{a}_k$.
      Given the new value of $\mathbf{a}_k$, estimate g_k.
    Given the most recent values for $\mathbf{a}_k$ and g_k, estimate $\{\beta_{lk}\}_{l=1}^{q}$.
  Until L_2 is minimised according to a certain stopping criterion.

This training algorithm presents a clear difference with conventional training of neural networks, in which all weights are updated at the same time.

Hwang *et al.* use as stopping criterion $\frac{|L_2^{\mathrm{new}} - L_2^{\mathrm{old}}|}{L_2^{\mathrm{old}}} < \xi = 0.005$.

The estimation of the different sets of parameters is as follows:

– $\{\beta_{lk}\}_{l=1}^{q}$ given $\mathbf{a}_k$ and g_k: L_2 is quadratic in β_{lk} for fixed g_m, β_{im} and $\mathbf{a}_m$, $m \neq k$; then, a linear LS minimisation $\partial L_2 / \partial \beta_{lk} = 0$ yields:

$$\hat{\beta}_{lk} = \frac{\mathrm{E}\left\{R_{l(k)} f_k(\mathbf{x}^T \mathbf{a}_k)\right\}}{\mathrm{E}\left\{f_k^2(\mathbf{x}^T \mathbf{a}_k)\right\}}, \qquad l = 1, \dots, q$$

– g_k given $\mathbf{a}_k$ and $\{\beta_{lk}\}_{l=1}^{q}$: use a 1-D data smoother:

1. Construct an unconstrained nonsmooth estimate g_k^* satisfying

$$\min_{\gamma_{ki}} L_2 = \sum_{l=1}^{q} W_l \,\mathrm{E}\left\{\left(R_{l(k)} - \beta_{lk}\gamma_{ki}\right)^2\right\}$$

with $\gamma_{ki} = g_k(\mathbf{x}_i^T \mathbf{a}_k)$:

$$g_k^*(\mathbf{x}_i^T \mathbf{a}_k) = \frac{\sum_{l=1}^{q} W_l \beta_{lk} R_{l(k)}}{\sum_{l=1}^{q} W_l \beta_{lk}^2}$$

2. Find a smooth curve $\hat{g}_k$ that best represents the scatterplot $\{(z_{ki}, g_k^*(z_{ki}))\}_{i=1}^{n}$ with $z_{ki} = \mathbf{x}_i^T \mathbf{a}_k$. Hwang *et al.* propose the use of the supersmoother (section F.3.4) or a smoother based on Hermite functions (section E.3.7).

– $\mathbf{a}_k$ given $\{\beta_{lk}\}_{l=1}^{q}$ and g_k: by Gauss-Newton nonlinear LS method, because L_2 is not quadratic in $\mathbf{a}_k$.

- **Forward growing procedure:** hidden neurons are added one at a time. After the parameters of the hidden neuron k are estimated, the parameters of previous neurons are fine-tuned by backfitting (see section 3.8.1), which is fast and easy because the output units are linear.

- A **backward pruning procedure** could also be implemented, in which overfitting neurons are removed one at a time by fitting all models of decreasing size $\tilde{\jmath} = j^* - 1, j^* - 2, \dots, j$ for $j^* > j$ (if j^* neurons have already been obtained).

6.3.1 Performance of PPL compared to that of BPL

Although approximation theorems of the form "any noise-free square-integrable function can be approximated to an arbitrary degree of accuracy" exist for both BPL [18, 50, 51, 52] and PPL (see section 6.1), nothing is said about the number of neurons required in practice. Hwang *et al.* report some empirical results (based on a simulation with $D = 2$, $q = 1$):

- PPL (with Hermite polynomials or with the supersmoother) outperforms BPL in learning accuracy (MSE of estimation on test data set for equal number of hidden neurons), learning parsimony (number of hidden neurons required to attain a fixed MSE) and —unless Gauss-Newton BPL is used— particularly in learning speed (CPU time employed for training with fixed MSE).

- In PPL, the Hermite polynomials normally outperform the supersmoother.

6.3.2 Parametric PPR

Backfitting in (nonparametric) PPR presents two problems:

- It is difficult to choose the smoothing parameter.

- It either converges very slowly or has a large bias even with noise-free data.

Zhao and Atkeson [101] propose *parametric PPR with direct training* to achieve improved training speed and accuracy: $\hat{f}(\mathbf{x}) = \sum_{k=1}^{j} \sum_{l=1}^{P} c_{lk} G(\mathbf{x}^T \mathbf{a}_k, t_l)$, where $\mathbf{a}_k$ are the direction projections, t_l are equispaced knots in the range of the projections of the n data points and are fixed for all directions, and $G(s,t)$ are one-dimensional weight functions usually depending on $|s-t|$. The ideal G is very similar to a cubic spline smoother, as its asymptotic form for large N is:

$$G(s,t) = \frac{1}{f(t)h(t)} K\left(\frac{s-t}{h(t)}\right) \qquad h(t) = \left(\frac{\lambda}{Nf(t)}\right)^{1/4} \qquad K(u) = \frac{1}{2} e^{-\frac{|u|}{\sqrt{2}}} \sin\left(\frac{|u|}{\sqrt{2}} + \frac{\pi}{4}\right)$$

where t_l has local density $f(t)$, $h(t)$ is the local bandwidth and $K(u)$ is the kernel function of eq. (F.2), depicted in fig. 12.

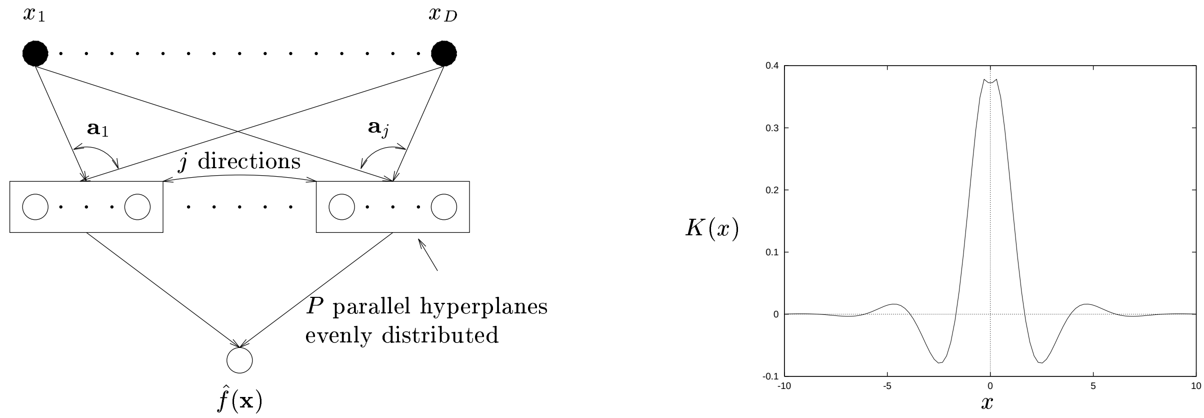

Figure 12: Kernel function for parametric PPR.

$\hat{f}(\mathbf{x})$ has jP linear parameters c_{lk} and jD nonlinear parameters for the D-dimensional directions $\mathbf{a}_k$. We can consider $\hat{f}$ as a two-layer perceptron (fig. 12) with activation function K, fixed location parameters t_l and nodes grouped in j directions, trainable as an MLP and that achieves dimension reduction if $j < D$.

Zhao and Atkeson propose an efficient training algorithm with direct search for all parameters and good initial values:

Repeat "several times"

1. Optimise simultaneously c_{lk} and $\mathbf{a}_k$ by a nonlinear LS method, stopping before convergence.
2. LS fit for c_{lk} with $\mathbf{a}_k$ fixed as in 1.

Good initial estimates for the directions $\mathbf{a}_k$ can be obtained if the first and second derivatives of the target functions at the data points are known (difficult). Otherwise, backfitting with a simple smoother can provide approximate initial directions.

Grouping hidden units (i.e. number of hyperplanes) from $H = jP$ into j groups of P reduces the number of parameters of the net by a fraction of j but keeps constant (asymptotically) the number of cells in D-dimensional space (the hidden layer partitions the space in cells produced by the intersecting hyperplanes $\mathbf{x}^T\mathbf{a}_k$). This means the accuracy is the same as for a two-layer perceptron but with fewer neurons, so that training is easier and generalisation ability better.

Since $H \approx DC^{1/D}$, where C is the number of cells, the number of hidden units grows linearly with the dimension for fixed accuracy: the *curse of dimensionality* has been reduced.

6.4 Cascade correlation learning network (CCLN)

Figure 13 shows a cascade correlation learning network (CCLN). CCLN [30] is a supervised learning architecture that dynamically grows layers of hidden neurons with fixed nonlinear activations (e.g. sigmoids), so that the network topology (size, length) can be efficiently determined. The Cascor training algorithm is as follows: given a new candidate unit h_k,

1. Train input-to-hidden (hidden layer) units using criterion of maximum correlation R between the candidate unit's value h_k and all the residual output errors $e_i = y_i - \hat{y}_i$:

$$\Delta w = \eta \frac{\partial R}{\partial w} \qquad \max R = \sum_{i=1}^{q} \mathrm{E}\left\{(h_k - \overline{h}_k)(e_i - \overline{e}_i)\right\} = \frac{1}{n}\sum_{l=1}^{n}\sum_{i=1}^{q} \mathrm{E}\left\{(h_k^{(l)} - \overline{h}_k)(e_i^{(l)} - \overline{e}_i)\right\}$$

where $\overline{h}_k$ and $\overline{e}_i$ are the averages over all n training patterns.

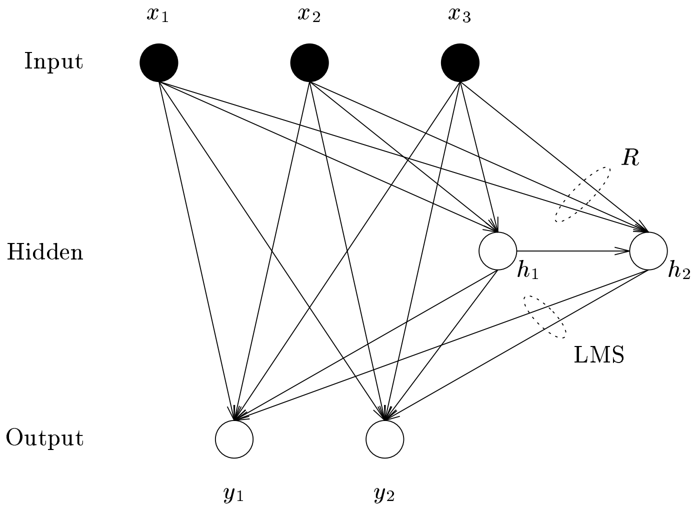

Figure 13: Cascade correlation learning network (CCLN).

	CCLN	PPLN
Inputs to the new hidden unit	From input units and all previous hidden units: increasing dimension	From input units only: fixed dimension
High-order nonlinearity employed in residual error approximation	Input connections	Input connections and nonparametric, trainable activation functions
Activation function	Fixed	Variable: trainable, nonparametric
Suitable for	Classification (not always)	Regression and classification
Retraining of the hidden layer	No	Yes (backfitting)

Table 2: CCLNs vs PPLNs.

2. (Re)train output-to-hidden (output layer) units from the new and all the previous hidden units using MSE criterion and *speedy quickprop*.

Hwang *et al.* [55] give a comparison between CCLN and PPLN:

- Each new hidden unit (from a pool of candidate units) receives connections from the input layer but also from all the previous hidden units, which provide with high-order linearity in approximating the residual error.
- Therefore, the input dimension of the candidate hidden unit increases but the nonlinear activation remains fixed, thus making training more and more difficult (weight search in high-dimensional space).
- The maximum correlation criterion pushes the hidden units to their saturated extreme values instead of the active region, making the decision boundary (classification) or the interpolated surface (regression) very zigzag. Thus, it is not suitable for regression and sometimes not for classification either.
- Fast retraining of output layer using MSE and *speedy quickprop*, as there is no need to backpropagate errors through hidden units (hidden layer not retrained).

Table 2 compares CCLNs with PPLNs.

6.5 BCM neuron using an objective function

6.5.1 The BCM neuron

Figure 14 shows the BCM neuron (Bienenstock, Cooper and Munro [4]). Let $\mathbf{x} \in \mathbb{R}^D$ be the input to the neuron and $c = \mathbf{x}^T\mathbf{w} \in \mathbb{R}$ its output. We define the threshold $\theta_\mathbf{w} = \mathrm{E}\left\{(\mathbf{x}^T\mathbf{w})^2\right\} = \mathrm{E}\left\{c^2\right\}$ and the functions[31] $\hat{\phi}(c,\theta_{\mathbf{w}}) = c^2 - \frac{1}{2}c\theta_{\mathbf{w}}$ and $\phi(c,\theta_{\mathbf{w}}) = c^2 - c\theta_{\mathbf{w}}$. ϕ has been suggested as a biologically plausible synaptic modification function to explain visual cortical plasticity. Because the threshold $\theta_{\mathbf{w}}$ is dynamic and depends on the projections in a nonlinear way, it will always move itself to a position such that the distribution will never be concentrated at only one of its sides (see fig. 14). Thus, the distribution has part of its mass on both sides of $\theta_{\mathbf{w}}$, which makes it a plausible projection index that seeks multimodalities.

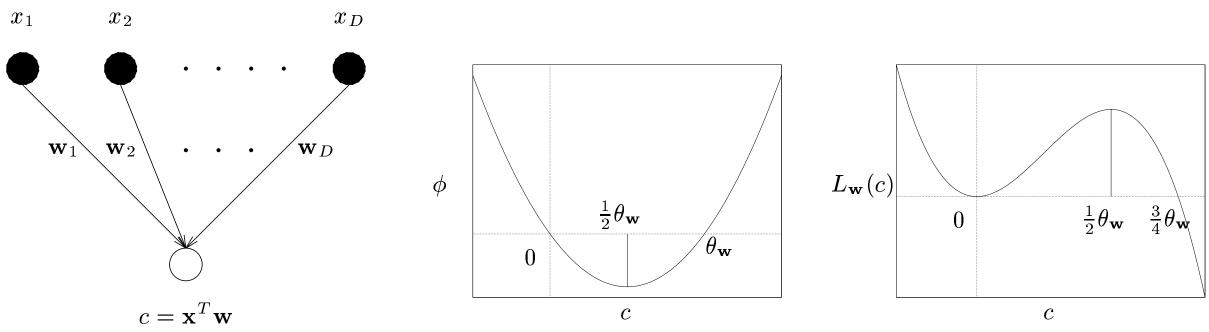

Figure 14: The BCM neuron and its associated functions ϕ (synaptic) and $L_{\mathbf{w}}$ (loss).

A loss function is attached to the BCM neuron, so that learning of the weights seeks to minimise the average loss or *risk*. If the risk is a smooth function its minimisation can be accomplished by gradient descent. Different loss functions will yield different learning procedures.

Let us consider the family of loss functions

$$L_{\mathbf{w}}(\mathbf{x}) = \int_0^{\mathbf{x}^T\mathbf{w}} \hat{\phi}(s,\theta_{\mathbf{w}}) = \frac{1}{3}c^3 - \frac{1}{4}\operatorname{E}\left\{c^2\right\}c^2 = c^2\left(\frac{c}{3} - \frac{\theta_{\mathbf{w}}}{4}\right) = L_{\mathbf{w}}(c)$$

and the risk

$$R_{\mathbf{w}} = \operatorname{E}\left\{L_{\mathbf{w}}(\mathbf{x})\right\} = \frac{1}{3}\operatorname{E}\left\{c^3\right\} - \frac{1}{4}\operatorname{E}\left\{c^2\right\}^2$$

Then, the learning rule is:

$$\frac{dw_i}{dt} = -\mu\frac{\partial R_{\mathbf{w}}}{\partial w_i} = \mu\left(\operatorname{E}\left\{c^2 x_i\right\} - \operatorname{E}\left\{c^2\right\}\operatorname{E}\left\{c x_i\right\}\right) = \mu\operatorname{E}\left\{\phi(c,\theta_{\mathbf{w}})x_i\right\} \Longrightarrow$$

$$\frac{d\mathbf{w}}{dt} = -\mu\nabla_{\mathbf{w}}R_{\mathbf{w}} = \mu\operatorname{E}\left\{\phi(c,\theta_{\mathbf{w}})\mathbf{x}\right\} \tag{6.1}$$

where μ is a time-decaying learning rate.

6.5.2 Extension to nonlinear neuron

Turning the linear BCM neuron into a nonlinear one we achieve:

- Insensitivity to outliers by means of a *sigmoidal activation function* (cf. eq. (B.1)): $c = \sigma(\mathbf{x}^T\mathbf{w})$, as $\sigma' \approx 0$ for outliers (points away from the mean lie in the flat part of the sigmoid).
- Ability to shift the distribution so that the part of it that satisfies $c < \theta_{\mathbf{w}}$ will have its mode at 0, by means of a *bias*: $c = \sigma(\mathbf{x}^T\mathbf{w} + w_0)$. From a biological viewpoint, the bias w_0 can be identified as spontaneous activity.

Therefore

$$L_{\mathbf{w}}(\mathbf{x}) = \sigma^2(c)\left(\frac{\sigma(c)}{3} - \frac{\theta_{\mathbf{w}}}{4}\right) \qquad -\nabla_{\mathbf{w}}R_{\mathbf{w}} = \operatorname{E}\left\{\phi(\sigma(c),\theta_{\mathbf{w}})\sigma'(c)\mathbf{x}\right\}.$$

6.5.3 Extension to network with feedforward inhibition

Adding inhibition connections between j BCM neurons (see fig. 15), the inhibited activity for neuron k is $\overline{c}_k = c_k - \eta\sum_{l\neq k} c_l$, the threshold $\overline{\theta}^k_{\mathbf{w}} = \operatorname{E}\left\{\overline{c}^2_k\right\}$ and the risk $R_k = \frac{1}{3}\operatorname{E}\left\{\overline{c}^3_k\right\} - \frac{1}{4}\operatorname{E}\left\{\overline{c}^2_k\right\}^2$, with a total

[31] Intrator uses a different definition of these functions in some papers (e.g. [57]): $\hat{\phi}(c,\theta_{\mathbf{w}}) = c^2 - \frac{2}{3}c\theta_{\mathbf{w}}$, $\phi(c,\theta_{\mathbf{w}}) = c^2 - \frac{4}{3}c\theta_{\mathbf{w}}$.

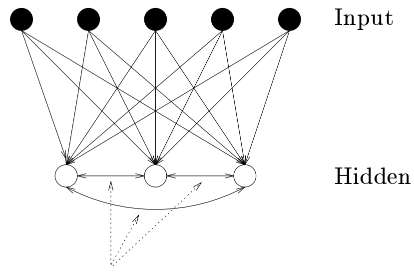

Figure 15: BCM neurons with inhibiting connections. Compare this figure with figure 9.

risk $R = \sum_{k=1}^{j} R_k$. The learning rule for linear neurons is:

$$\dot{\mathbf{w}}_k = -\mu \nabla_{\mathbf{w}_k} R = -\mu \frac{\partial R}{\partial \mathbf{w}_k} = -\mu \left(\mathrm{E}\left\{ \phi(\overline{c}_k, \overline{\theta}_{\mathbf{w}}^k)\mathbf{x} \right\} - \eta \sum_{l \neq k} \mathrm{E}\left\{ \phi(\overline{c}_l, \overline{\theta}_{\mathbf{w}}^l)\mathbf{x} \right\} \right)$$

and for nonlinear neurons with activation $\overline{c}_k = \sigma\left(c_k - \eta \sum_{l \neq k} c_l\right)$ is:

$$\begin{aligned} \dot{\mathbf{w}}_k = -\mu \left(\mathrm{E}\left\{ \phi(\overline{c}_k, \overline{\theta}_{\mathbf{w}}^k)\sigma'(\overline{c}_k)\mathbf{x} \right\} - \eta \sum_{l \neq k} \mathrm{E}\left\{ \phi(\overline{c}_l, \overline{\theta}_{\mathbf{w}}^l)\sigma'(\overline{c}_l)\mathbf{x} \right\} \right) = \\ -\mu\, \mathrm{E}\left\{ \phi(\overline{c}_k, \overline{\theta}_{\mathbf{w}}^k)\sigma'(\overline{c}_k)\mathbf{x} - \eta \sum_{l \neq k} \phi(\overline{c}_l, \overline{\theta}_{\mathbf{w}}^l)\sigma'(\overline{c}_l)\mathbf{x} \right\} \end{aligned} \tag{6.2}$$

Intrator used a BCM feedforward inhibition network with speech data [57] and facial images [60].

6.5.4 Conclusions

- Uses low-order polynomial moments, which are efficiently computable and not very sensible to outliers, due to the addition of the sigmoidal activation function.
- Natural extension to multidimensional projection pursuit using the feedforward inhibition network.
- The gradient optimisation is $\mathcal{O}(Nn)$, i.e. linear with dimension and number of projections sought.

7 Conclusions and Future Work

We have defined the problem of dimension reduction as the search for an economic coordinate representation of a submanifold of a high-dimensional Euclidean space, a problem so far not yet solved in a satisfactory and general way. We have then given an overview of several well-known techniques for dimension reduction, as well as straightforward implementations of some of them using neural networks.

The two major issues that remain open are:

- To overcome the curse of the dimensionality, which demands huge sample sizes to obtain reasonable results. Most of the techniques reviewed still suffer of this to an extent.
- To determine the intrinsic dimension of a distribution given a sample of it. This is central to the problem of dimension reduction, because knowing it would eliminate the possibility of over- or underfitting.

Finally, we would like to conclude by mentioning a number of further techniques related to dimension reduction that have not been included in this work due to lack of time. These would include the Helmholtz machine [19, 20], some variations of self-organising maps (growing neural gas [39, 40], Bayesian approaches [96, 97], etc.), population codes [100], and curvilinear component analysis [22], among others.

A Glossary

- Backfitting algorithm: an iterative method to fit additive models, by fitting each term to the residuals given the rest (see section 3.8.1). It is a version of the Gauss-Seidel methods of numerical linear algebra.
- BCM: Bienenstock-Cooper-Munro model of neuron selectivity.
- BPL: Backpropagation Learning.
- CART: Classification and Regression Trees.
- CCLN: Cascade Correlation Learning Network.
- cdf, $F(x)$: cumulative distribution function.
- EPP: Exploratory Projection Pursuit.
- Equivariant procedure: one that correspondingly transforms its answer under a transformation of its input: $\phi = g(\theta) \Rightarrow \hat{\phi} = g(\hat{\theta})$.

  Invariant procedure: one that does not transform its answer under a transformation of its input: $\phi = g(\theta) \Rightarrow \phi = g(\hat{\theta})$.
- GAM: Generalised Additive Model.
- HMM: Hidden Markov Models.
- iid: independent identically distributed.
- LS: Least Squares.
- MAP: Maximum a posteriori Probability.
- MARS: Multivariate Adaptive Regression Splines.
- MDS: Multidimensional Scaling.
- MLP: Multilayer Perceptron.
- Needleplot: plot of a sample $\{X_i\}_{i=1}^n$ so that each point gets the same part of the density mass, i.e. we put a *needle* at the value of each observation X_i to get an impression of the distribution of the sample. It is a very noisy estimate of the density. It is equivalent to a sum of discrete Dirac deltas located at the observations.
- PC: Principal Component.
- PCA: Principal Component Analysis.
- pdf, $f(x)$: probability distribution function or density.
- pmf, f_i: probability mass function.
- PP: Projection Pursuit.
- PPDA: Projection Pursuit Density Approximation.
- PPDE: Projection Pursuit Density Estimation.
- PPL: Projection Pursuit Learning.
- PPLN: Projection Pursuit Learning Network.
- PPR: Projection Pursuit Regression.
- RBF: Radial Basis Function.

B Notation

- Kronecker delta:
$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

- Indicator function:
$$I_{\mathcal{M}}(x) = \begin{cases} 1 & x \in \mathcal{M} \\ 0 & \text{otherwise.} \end{cases}$$

- Sigmoid function:
$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{B.1}$$

- L_p-norm of a vector $\mathbf{v}$: $\|\mathbf{v}\|_p = \left(\sum_{i=1}^{D} |v_i|^p\right)^{1/p}$. Often, we will just write $\|\cdot\|$ to mean $\|\cdot\|_2$. $\|\mathbf{v}\|_\infty = \max_i |v_i|$.
- D-sphere centred in the origin with radius R: $\mathbb{S}_R^D = \{\mathbf{x} \in \mathbb{R}^D : \|\mathbf{x}\|_2 \leq R\}$ (for the hollow sphere replace $\leq$ by $=$).
- D-hypercube centred in the origin with side $2R$: $\mathbb{C}_R^D = \{\mathbf{x} \in \mathbb{R}^D : \|\mathbf{x}\|_\infty \leq R\} = [-R, R]^D$ (for the hollow hypercube replace $\leq$ by $=$).
- Half-space of $\mathbb{R}^D$: $\mathbb{H}^D = \{\mathbf{x} \in \mathbb{R}^D : x_D \geq 0\}$.
- Coordinate unit vectors in $\mathbb{R}^D$: $\{\mathbf{e}_i\}_{i=1}^D$, $e_{ij} = \delta_{ij}$, $i, j = 1, \dots, D$.
- Vector of ones (of whichever dimension): $\mathbf{1} = (1, \dots, 1)^T$.
- Identity matrix (of whichever dimension): $\mathbf{I}$.
- $\mathbf{J} = \frac{1}{n}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$ for $\mathbf{I}_{n\times n}$, $\mathbf{1}_{n\times 1}$ and $n \in \mathbb{N}$. $\mathbf{J}$ verifies $\mathbf{J}^2 = \frac{1}{n^2}\mathbf{J}$, $\mathbf{J} = \mathbf{J}^T$, $\mathbf{J}\mathbf{1} = \mathbf{0}$.
- Square orthogonal matrix: $\mathbf{Q}^{-1} = \mathbf{Q}^T$.
- A projection $\Pi : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ of *base* $\mathcal{B} = \operatorname{im}\Pi$ and *direction* $\mathcal{D} = \ker\Pi$ is represented by a square matrix $\Pi_{D\times D}$, which is symmetrical ($\Pi = \Pi^T$) and idempotent ($\Pi^2 = \Pi$). The following properties hold:
  - $\mathbb{R}^D = \mathcal{B} \oplus \mathcal{D}$; $\dim\mathcal{B} = j$, $\dim\mathcal{D} = D - j$. Therefore Π is of rank j and can be represented in coordinates of $\mathcal{B}$ by a matrix $\mathbf{A}_{D\times j}$ ($\mathbf{A}^T : \mathbb{R}^D \longrightarrow \mathbb{R}^j$). The projection will be orthogonal if the column vectors of $\mathbf{A}$ (basis of $\mathcal{B}$) are orthonormal.
  - $\Pi\mathbf{z} = \Pi(\mathbf{x} + \mathbf{y}) = \mathbf{x}$ where $\mathbf{z} \in \mathbb{R}^D$, $\mathbf{x} \in \mathcal{B}$ and $\mathbf{y} \in \mathcal{D}$.
  - The matrix of the projection on a unit vector $\mathbf{v}$ is $\Pi = \mathbf{v}\mathbf{v}^T$.
- Matrix representation of a sample of n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ in $\mathbb{R}^D$: $\mathbf{X}_{D\times n} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$. The columns are coordinate vectors and the rows are univariate samples.
- Translation of a sample of n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ in $\mathbb{R}^D$ by vector $\mathbf{t} \in \mathbb{R}^D$:
  - Vector form: $\mathbf{x}'_i = \mathbf{x} + \mathbf{t}$
  - Matrix form: $\mathbf{X}_{D\times n} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, $\mathbf{X}' = \mathbf{X} + \mathbf{t}\mathbf{1}^T$.
- Mean $\overline{\mathbf{x}}$ and covariance matrix $\mathbf{\Sigma}_{D\times D} = (\sigma_{ij})$ of a sample of n vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ in $\mathbb{R}^D$:
  - Vector form: $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$, $\mathbf{\Sigma} = \mathrm{E}\left\{(\mathbf{x} - \mathrm{E}\{\mathbf{x}\})(\mathbf{x} - \mathrm{E}\{\mathbf{x}\})^T\right\} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T - \overline{\mathbf{x}}\overline{\mathbf{x}}^T$.
  - Matrix form: $\mathbf{X}_{D\times n} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$, $\overline{\mathbf{x}} = \frac{1}{n}\mathbf{X}\mathbf{1}$, $\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\mathbf{X}^T = \mathbf{X}\mathbf{J}\mathbf{X}^T$.
- Square root of a positive semidefinite symmetric square matrix $\mathbf{B}$: $\mathbf{B}^{1/2} = \mathbf{A} \Leftrightarrow \mathbf{A}\mathbf{A}^T = \mathbf{B}$. If $\mathbf{B} = \operatorname{diag}(b_i)$, then $\mathbf{B}^{1/2} = \operatorname{diag}\left(\sqrt{b_i}\right)$; otherwise, using a spectral decomposition of $\mathbf{B}$: $\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}\mathbf{U}^T \Rightarrow \mathbf{B}^{1/2} = \mathbf{U}\mathbf{\Lambda}^{1/2}$. Note that $\mathbf{B}^{1/2}$ is not unique: $\mathbf{B}^{1/2}\mathbf{Q}$ for $\mathbf{Q}$ orthogonal is also valid.

- Order statistics of a sample $\{X_i\}_{i=1}^n$: $\{X_{(i)}\}_{i=1}^n$ with $X_{(1)} \leq \cdots \leq X_{(n)}$. It is a handy notation to deal with ordered data.

- Average operator:

$$\mathrm{E}\{X\} = \begin{cases} \frac{1}{n}\sum_{i=1}^n X_i & \text{if } X = \{X_i\}_{i=1}^n \text{ is a sample} \\ \sum_i X_i f_i & \text{if } X \text{ is a discrete random variable with pmf } f \\ \int x f(x)\, dx & \text{if } X \text{ is a continuous random variable with pdf } f. \end{cases}$$

- Moment of order m of univariate density $f(u)$:

$$\mu_m(f) = \mathrm{E}\{u^m\} = \left(\frac{d^m}{dt^m}\right)_{t=0} \mathrm{E}\left\{e^{tu}\right\}.$$

  In particular, $\mu_1 = \mu = \mathrm{E}\{X\}$ is the mean of f and $\mathrm{var}\{X\} = \sigma^2(X) = \mu_2 - \mu_1^2$ its variance.

- Cumulant of order m of univariate density $f(u)$:

$$k_m(f) = \left(\frac{d^m}{i^m dt^m}\right)_{t=0} \ln \mathrm{E}\left\{e^{itu}\right\} \qquad i = \sqrt{-1}. \tag{B.2}$$

  In particular, $k_1 = \mu_1$ (mean), $k_2 = \mu_2 - \mu_1^2$ (variance), $k_3 = \mu_3 - 3\mu_1\mu_2 + 2\mu_1^3$ (skewness) and $k_4 = \mu_4 - 4\mu_3\mu_1 - 3\mu_2^2 + 12\mu_1^2\mu_2 - 6\mu_1^4$ (kurtosis). For a random variable with zero mean and unit variance $k_1 = 0$, $k_2 = 1$, $k_3 = \mu_3$ and $k_4 = \mu_4 - 3$ (e.g. for $\mathcal{N}(0,1)$, the standard normal, $k_1 = k_3 = k_4 = 0$, $k_2 = 1$).

- Pdf of the multivariate Gaussian or normal distribution in D dimensions $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and covariance matrix $\boldsymbol{\Sigma}_{D\times D} > 0$:

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{B.3}$$

  Pdf of the standard multivariate $\mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\phi(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}\|\mathbf{x}\|^2}. \tag{B.4}$$

  Pdf of the one-dimensional normal distribution $\mathcal{N}(\mu, \sigma)$ with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}^+$:

$$\phi(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}. \tag{B.5}$$

  Pdf of the standard normal $\mathcal{N}(0,1)$:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \tag{B.6}$$

  Cdf of the standard normal $\mathcal{N}(0,1)$:

$$\Phi(x) = \int_{-\infty}^{x} \phi(t)\, dt = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}\, dt. \tag{B.7}$$

- L_p-norm of a function f: $\|f\|_p = \left(\int f^p(\mathbf{x})\, d\mathbf{x}\right)^{1/p}$. Often, we will just write $\|\cdot\|$ to mean $\|\cdot\|_2$.

- $L_p(\mathbb{R}^D)$ space: space of random D-dimensional variables X such that $\mathrm{E}\{x^p\} < \infty$. Convergence $X_n \to X$ in L_p means $\mathrm{E}\{X_n - X\}^p \to 0$ when $n \to \infty$.

  It is also the space of functions $f : \mathbb{R}^D \longrightarrow \mathbb{R}$ such that $\|f\|_p^2 < \infty$. $p = 2$ is the space of square-integrable functions.

- Criteria to measure goodness of approximation of one function g to another one f (e.g. g can be a density estimate of the density f of a random variable X):

  - Sum of squares (L_2-norm): $\|f - g\|_2^2$.

– Relative entropy (or Kullback-Leibler distance):

$$D(f||g) = \mathrm{E}_f \left\{ \ln \frac{f(X)}{g(X)} \right\} \tag{B.8}$$

which is a distance but not a metric, because $D(f||g) \neq D(g||f)$ in general.

– Hellinger distance:

$$H(f,g) = \int \left(\sqrt{f(x)} - \sqrt{g(x)} \right)^2 \, dx. \tag{B.9}$$

– Mean Squared Error (pointwise error criterion), given a sample $\{X_i\}_{i=1}^n$:

$$\mathrm{MSE}_x(g) = \mathrm{E}\left\{ (g(x) - f(x))^2 \right\} = \mathrm{Var}\{g(x)\} + \mathrm{Bias}^2\{g(x)\} \tag{B.10}$$

where $\mathrm{Bias} = \mathrm{E}\{g(x)\} - f(x)$. This shows the **bias-variance decomposition**.

– Mean Integrated Squared Error (global error criterion), given a sample $\{X_i\}_{i=1}^n$:

$$\mathrm{MISE}(g) = \mathrm{E}\left\{ \|g - f\|_2^2 \right\} = \mathrm{E}\left\{ \int (g(x) - f(x))^2 \, dx \right\} = \int \mathrm{MSE}_x(g) \, dx.$$

- Entropy of a random variable X:

$$H(X) = \mathrm{E}\left\{ \ln \frac{1}{f(X)} \right\} \tag{B.11}$$

If X is a continuous random variable, its entropy is often called *differential entropy* $h(X)$.
Conditional entropy of random variable X given random variable Y:

$$H(Y|X) = -\,\mathrm{E}_{f(X,Y)}\{\ln f(Y|X)\} \tag{B.12}$$

Mutual information between random variables X and Y:

$$I(X;Y) = H(X) - H(X|Y) \tag{B.13}$$

C Properties of the Covariance Matrix

$\mathbf{\Sigma}_{D\times D}$ is symmetric semidefinite positive ($\mathbf{\Sigma} \geq 0$) and admits a spectral decomposition

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{C.1}$$

with $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_D)$ orthogonal, $\mathbf{\Lambda} = \operatorname{diag}(\lambda_i)$ and $\lambda_i \geq 0$, $i = 1, \dots, D$; $\mathbf{u}_i$ is the normalised eigenvector of $\mathbf{\Sigma}$ associated to eigenvalue λ_i.

C.1 Transformation of the covariance matrix under transformations of the sample

Using the spectral decomposition (C.1) of the covariance matrix and the matrix notation introduced in section B, it is very easy to construct table 3, in which $a \in \mathbb{R} - \{0\}$, $\mathbf{t} \in \mathbb{R}^D$, $\mathbf{D} = \operatorname{diag}(d_i)$ is diagonal, $\mathbf{Q}$ is orthogonal, the primes denote the new entity after the transformation f and the symbol $\neq$ is used to indicate that the subsequent transformation is complex (not obviously related to f).

Transformation	$\mathbf{X}' = f(\mathbf{X})$	$\overline{\mathbf{x}}'$	$\mathbf{\Sigma}'$	$\mathbf{U}'$	$\mathbf{u}'$	$\mathbf{\Lambda}'$	λ'
Translation	$\mathbf{X} + \mathbf{t}\mathbf{1}^T$	$\overline{\mathbf{x}} + \mathbf{t}$	$\mathbf{\Sigma}$	$\mathbf{U}$	same	$\mathbf{\Lambda}$	same
Rotation	$\mathbf{QX}$	$\mathbf{Q}\overline{\mathbf{x}}$	$\mathbf{Q\Sigma Q}^T$	$\mathbf{QU}$	rotated	$\mathbf{\Lambda}$	same
	$\mathbf{XQ}$	$\overline{\mathbf{x}}\mathbf{Q}$	$\mathbf{\Sigma} - \overline{\mathbf{x}}'(\overline{\mathbf{x}}')^T$	$\neq$	$\neq$	$\neq$	$\neq$
Axis scaling	$\mathbf{DX}$	$\mathbf{D}\overline{\mathbf{x}}$	$\mathbf{D\Sigma D}$	$\neq$	$\neq$	$\neq$	$\neq$
Uniform axis scaling	$a\mathbf{X}$	$a\overline{\mathbf{x}}$	$a^2\mathbf{\Sigma}$	$\mathbf{U}$	same	$a^2\mathbf{\Lambda}$	scaled
Affine	$a\mathbf{X} + \mathbf{t}\mathbf{1}^T$	$a\overline{\mathbf{x}} + \mathbf{t}$	$a^2\mathbf{\Sigma}$	$\mathbf{U}$	same	$a^2\mathbf{\Lambda}$	scaled
Centring	$\mathbf{X} - \overline{\mathbf{x}}\mathbf{1}^T$	$\mathbf{0}$	$\mathbf{\Sigma}$	$\mathbf{U}$	same	$\mathbf{\Lambda}$	same
PCA	$\mathbf{U}^T(\mathbf{X} - \overline{\mathbf{x}}\mathbf{1}^T)$	$\mathbf{0}$	$\mathbf{\Lambda}$	$\mathbf{I}$	$\{\mathbf{e}_i\}_{i=1}^D$	$\mathbf{\Lambda}$	same
Sphering	$\mathbf{\Sigma}^{-1/2}(\mathbf{X} - \overline{\mathbf{x}}\mathbf{1}^T)$	$\mathbf{0}$	$\mathbf{I}$	$\mathbf{I}$	$\{\mathbf{e}_i\}_{i=1}^D$	$\mathbf{I}$	1

Table 3: Transformation of the covariance matrix under transformations on the sample.

C.2 Centring, scaling, sphering and PCA

Assume $\mathbf{X}_{D\times n}$ is a sample of n vectors in $\mathbb{R}^D$ with mean $\mathrm{E}\{\mathbf{X}\}$ and covariance matrix $\mathbf{\Sigma} = (\sigma_{ij})$, and that the spectral decomposition of $\mathbf{\Sigma}$ is $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ with $\mathbf{U}$ orthogonal and $\mathbf{\Lambda}$ diagonal. Let $\mathbf{A}_{D\times j} = (\mathbf{a}_1, \dots, \mathbf{a}_j)$, $\mathbf{a}_k \in \mathbb{R}^D$ a set of j projection directions.

- *Centring* is a procedure by which we translate the sample $\mathbf{X}$ so that its mean is at the origin: $\mathbf{X}' = \mathbf{X} - \mathrm{E}\{\mathbf{X}\} = \mathbf{X}(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T) \Rightarrow \mathrm{E}\{\mathbf{X}'\} = \mathbf{0}$.

  Centring is inherited by any set of projections: $\mathrm{E}\{\mathbf{X}\} = \mathbf{0} \Rightarrow \mathrm{E}\{\mathbf{A}^T\mathbf{X}\} = \mathbf{A}^T\,\mathrm{E}\{\mathbf{X}\} = \mathbf{0}$.

- *Scaling* achieves unit variance in each axis by dividing componentwise by its standard deviation σ_i: $\mathbf{X}' = \operatorname{diag}(\sigma_i^{-1})\,\mathbf{X} \Rightarrow \sigma'_{ii} = 1 \quad \forall i$.

- *Sphering* is an affine transformation that converts the covariance matrix (of the centred sample) into a unit variance matrix, thus destroying all the first- and second-order information of the sample: $\mathbf{X}' = \mathbf{\Sigma}^{-1/2}(\mathbf{X} - \mathrm{E}\{\mathbf{X}\}) \Rightarrow \mathrm{E}\{\mathbf{X}'\} = \mathbf{0}$ and $\mathbf{\Sigma}' = \mathbf{I}$, where[32] $\mathbf{\Sigma}^{-1/2} = \mathbf{U}\mathbf{\Lambda}^{-1/2}\mathbf{U}^T$.

  Sphering is inherited by any *orthogonal* set of projections: $\mathrm{E}\{\mathbf{X}\} = \mathbf{0}$ and $\operatorname{cov}\{\mathbf{X}\} = \mathbf{I} \Rightarrow \operatorname{cov}\{\mathbf{A}^T\mathbf{X}\} = \mathrm{E}\{(\mathbf{A}^T\mathbf{X})(\mathbf{A}^T\mathbf{X})^T\} = \mathbf{A}^T\,\mathrm{E}\{\mathbf{X}\mathbf{X}^T\}\,\mathbf{A} = \mathbf{A}^T\mathbf{A} = \mathbf{I}$.

- *PCA* is another affine transformation that converts the covariance matrix (of the centred sample) into a diagonal matrix, thus decorrelating the variables but preserving the variance information: $\mathbf{X}' = \mathbf{U}^T(\mathbf{X} - \mathrm{E}\{\mathbf{X}\}) \Rightarrow \mathrm{E}\{\mathbf{X}'\} = \mathbf{0}$, $\mathbf{\Sigma}' = \mathbf{\Lambda}$.

PCA and sphering are both translation and rotation invariant, i.e. applying a translation and a rotation to the data and then performing PCA or sphering produces the same results as performing them on the original data.

[32] Actually, $\mathbf{X}' = \mathbf{Q}\mathbf{\Sigma}^{-1/2}(\mathbf{X} - \mathrm{E}\{\mathbf{X}\})$ with $\mathbf{Q}$ orthogonal will produce the same result. The case $\mathbf{Q} = \mathbf{I}$ is the canonical transformation $\mathbf{X}' = \mathbf{\Sigma}^{-1/2}(\mathbf{X} - \mathrm{E}\{\mathbf{X}\})$; the case $\mathbf{Q} = \mathbf{U}^T$ gives $\mathbf{X}' = \mathbf{\Lambda}^{-1/2}\mathbf{U}^T(\mathbf{X} - \mathrm{E}\{\mathbf{X}\})$.

D Probability Density Function of a Projected Distribution

Let $f(\mathbf{x})$ a pdf in $\mathbb{R}^D$ and Π a projection of base $\mathcal{B}$ and direction $\mathcal{D}$, with $\dim \mathcal{B} = j$. The orthogonal projection of the distribution given by f on the subspace $\mathcal{B}$ has a new pdf in $\mathbb{R}^j$ given by:

$$f_\Pi(\mathbf{x}) = \int_{\mathcal{D}} f(\mathbf{x} + \mathbf{y})\, d\mathbf{y} \qquad \mathbf{x} \in \mathcal{B}$$

i.e. the integral for a given point $\mathbf{x}$ in the projection subspace $\mathcal{B}$ is extended to all points $\mathbf{x} + \mathbf{y}$ which project on the same point $\mathbf{x}$. If $\{\mathbf{a}_k\}_{k=1}^{j}$ and $\{\mathbf{b}_k\}_{k=1}^{D-j}$ are basis of $\mathcal{B}$ and $\mathcal{D}$, respectively, the previous integral is:

$$f_\Pi(\mathbf{t}) = \int_{\infty}^{\infty} \cdots \int_{\infty}^{\infty} f\left(\sum_{k=1}^{j} t_k \mathbf{a}_k + \sum_{k=1}^{D-j} \lambda_k \mathbf{b}_k\right) d\lambda_1 \ldots d\lambda_{D-j}$$

with $\mathbf{t} = (t_1, \ldots, t_j) \in \mathbb{R}^j$.

For the particular case $j = 1$ and $\mathbf{a}_k = \mathbf{e}_k$ we have the marginal densities; e.g. for $k = 1$:

$$f(x_1) = \int_{\infty}^{\infty} \cdots \int_{\infty}^{\infty} f(x_1, \ldots, x_D)\, dx_2 \ldots dx_D$$

For a cloud of points, projected clouds are obtained projecting stepwise on subspace $\mathcal{B}$.

E Density Estimation (Density Smoothing)[33]

In density estimation we have a set of unlabelled data $\{X_i\}_{i=1}^n$, drawn from an unknown distribution, which we want to model. The learning of this model is **unsupervised** in nature (cf. regression smoothing, appendix F).

A distribution can be characterised by its pdf $f(x)$, which predicts where observations cluster and occur more frequently and clearly shows skewness and multimodality. Therefore, the problem of density estimation can be stated as "estimate an unknown density[34] f given a sample $\{X_i\}_{i=1}^n$ of it" (i.e. X_i are realisations of iid random variables with density f).

E.1 Parametric and nonparametric density estimation

There are two approaches to estimate f:

- Parametric: assume f follows a certain model, equip it with a finite set of parameters and estimate them. This presents the problem of selecting the model. We will not deal further with it here.
- Nonparametric: no assumptions are made about f; its shape is dictated by the data $\{X_i\}_{i=1}^n$.

The main nonparametric density estimation approaches are shortly described next. First (section E.3) we consider only univariate data $\{X_i\}_{i=1}^n \subset \mathbb{R}$ and then (section E.4) we generalise the results to the multivariate case. For each method, the "local smoothness" and "pdf" properties of the resulting estimate are given (i.e. whether it is continuous and differentiable, and whether it is a pdf), as well as any problems they present.

All estimators given are consistent for f, i.e. $\text{MSE}(\hat{f}_h(x)) \to 0$ when $h \to 0$ and $nh \to \infty$.

E.2 The smoothing parameter

All estimators can be tuned by a smoothing parameter h, such that:

- For small h, the spurious fine structure of the data appears and the estimate is very noisy. In the limit case $h = 0$ one obtains a needleplot or a sum of Dirac deltas.
- For large h, all structure is obscured and the estimate is very smooth. In the limit case $h = \infty$ one obtains a constant function.

A common problem of all density estimation approaches is the optimum selection of the smoothing parameter. Several methods exist (e.g. cross-validation, bootstrap) to automatically select it from the data.

E.3 Nonparametric density estimation techniques: Univariate case

E.3.1 Histogram estimation

Divide the real line into bins $B_j = [x_0 + (j-1)h, x_0 + jh]$, $j \in \mathbb{Z}$, with bin width $h > 0$ and origin x_0. The estimator is the count of data that fall into each bin:

$$\hat{f}_h(x) = \frac{1}{nh}(\text{No. of } X_i \text{ in same bin as } x) = \frac{1}{nh}\sum_{i=1}^n \sum_j I_{B_j}(X_i) I_{B_j}(x)$$

or, for bins of varying size:

$$\hat{f}_h(x) = \frac{1}{n}\frac{(\text{No. of } X_i \text{ in same bin as } x)}{(\text{width of bin containing } x)}.$$

The histogram is a kernel estimator with uniform kernel.

Local smoothness: Discontinuous in bin boundaries; zero derivative everywhere else.

Problems: The choice of origin can affect the estimate. The next estimators are independent of origin choice.

[33]This appendix is mainly based on the books by Silverman [93], Scott [89] and Haerdle [45].

[34]It is also possible to estimate the cdf $F(x)$ and then obtain from it the pdf $f = \frac{dF}{dx}$ using the empirical cdf $\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^n I_{(-\infty,x]}(X_i)$. It can be shown that this is an unbiased estimator and has the smallest variance of all unbiased estimators of the cdf; however, the fact that the cdf is not continuous and that it is difficult to observe skewness or multimodality in it —particularly with multivariate data— makes it preferable to directly try to estimate the pdf.

E.3.2 Naive estimator

The naive estimator is an approximation to the formula $f(x) = \lim_{h \to 0} P(x - h < X < x + h)$, which holds by definition of cdf. For small h:

$$\hat{f}_h(x) = \frac{1}{2nh} \text{(No. of } X_i \text{ in (x-h,x+h))}$$

which can be interpreted as "a histogram where every point x is the centre of a bin" (therefore independent of origin choice).

The naive estimator is a kernel estimator with uniform kernel.

Local smoothness: Discontinuous in $X_i \pm h$; zero derivative everywhere else.

E.3.3 Kernel density estimation

Kernel estimators can be motivated from several points of view, including numerical analysis (as finite difference approximations to the derivative of the cdf, i.e. the pdf) and signal processing (as convolution of the pdf with a window function[35]).

If we define a **kernel function** K as a nonnegative[36], with unit integral (and also typically even), i.e. a pdf:

$$K(x) \geq 0\, \forall x \qquad \int K(x)\, dx = 1 \qquad \text{and typically } K(x) = K(-x)\, \forall x$$

then the kernel estimator with bandwidth (or window width) h is given by:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

i.e. it is a sum of bumps (of shape K and width h) placed at the observations. Table 4 lists some of the more popular kernels and figure 16 shows their graphical aspect.

Local smoothness, pdf: Inherited from K.

Problems: When applied to data from long-tailed distributions, spurious noise appears in the tails of the estimate because h is fixed. Smoothing the estimate to avoid this masks the essential detail in the main body of the distribution (see fig. 17).

E.3.4 Nearest-neighbours estimation

The previous methods count the number of observations in a "box" of width h. In nearest-neighbours estimation, the amount of smoothing adapts to the local density: the box is wide enough to contain k samples. k is considerably smaller then the sample size n, typically $k \approx \sqrt{n}$.

Given x, arrange distances of X_i to x in descending order: $d_1(x) \leq \cdots \leq d_n(x)$. The nearest-neighbours estimate is:

$$\hat{f}_h(x) = \frac{k}{2nd_k(x)}$$

$d_k(x)$ is bigger in the tails than in the main body, which reduces the problem of undersmoothing in the tails. However: for $x < X_{(1)}$, $d_k(x) = X_{(k)} - x$, and for $x > X_{(n)}$, $d_k(x) = x - X_{(n-k+1)}$; therefore, the tails of $\hat{f}_h$die away at rate x^{-1} and $\hat{f}_h$ is not integrable and is not a pdf.

Local smoothness: Inherited from $d_k(x)$: continuous but discontinuous derivative at $\frac{1}{2}(X_{(j)} + X_{(j+k)})$.

Problems: Heavy tails.

[35] Actually, the kernel estimator is implemented using the Fast Fourier Transform.

[36] This is not strictly necessary (e.g. the Dirichlet kernel of table 4, or the kernel by Zhao and Atkeson [101] of fig. 12).

Kernel	$K(u)$	eff(K), exact to 4 d.p.
Epanechnikov	$\frac{3}{4}(1-u^2)I_{[-1,1]}(u)$	1
Biweight (quartic)	$\frac{15}{16}(1-u^2)^2 I_{[-1,1]}(u)$	$\sqrt{3087/3125} \approx 0.9939$
Triangular	$(1-\lvert u \rvert)I_{[-1,1]}(u)$	$\sqrt{243/250} \approx 0.9859$
Gaussian	$\frac{1}{\sqrt{2\pi}}e^{-u^2/2}$	$\sqrt{36\pi/125} \approx 0.9512$
Uniform (rectangular)	$\frac{1}{2}I_{[-1,1]}(u)$	$\sqrt{198/125} \approx 0.9295$
Triweight	$\frac{35}{32}(1-u^2)^3 I_{[-1,1]}(u)$	
Cosine	$\frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right) I_{[-1,1]}(u)$	
Dirichlet	$\frac{\sin(2N+1)\pi u}{\sin \pi u} I_{[-1,1]}(u)$	
Fejér	$\frac{1}{N+1}\left(\frac{\sin(2N+1)\pi u}{\sin \pi u}\right)^2 I_{[-1,1]}(u)$	

Table 4: Some typical kernel functions and their efficiencies.

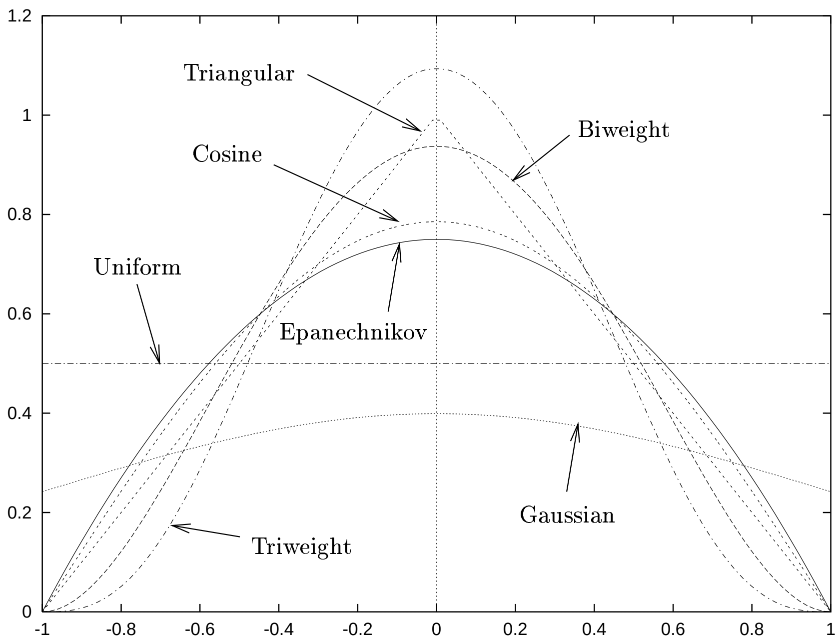

Figure 16: Plot of some typical kernel functions.

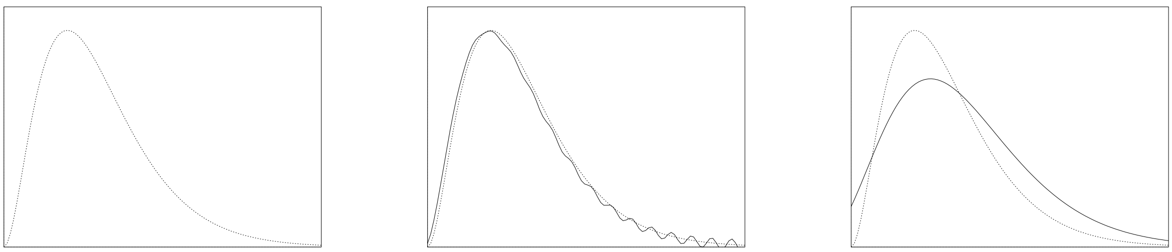

Figure 17: The drawback of kernel smoothing. The left picture shows the original distribution (always in dotted line); in the center picture, kernel smoothing with optimum h is applied and spurious noise appears in the tail; in the right one, kernel smoothing with a larger h is applied and the estimate is oversmoothed, producing a large bias in the peak.

E.3.5 Generalised k-th nearest-neighbour estimation

It is a kernel estimator evaluated at x with $h = d_k(x)$:

$$\hat{f}_h(x) = \frac{1}{n d_k(x)} \sum_{i=1}^{n} K\left(\frac{x - X_i}{d_k(x)}\right)$$

The uniform kernel gives ordinary nearest-neighbours estimation.

Local smoothness: Still not differentiable where $d_k(x)$ is not.

pdf: Precise integration and tail properties depend on the particular kernel.

E.3.6 Variable (or adaptive) kernel estimator

Define d_{ik} as the distance from X_i to the k-th nearest point in the set comprising the other $n-1$ data points. Then the variable kernel estimator is given by:

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h d_{ik}} K\left(\frac{x - X_i}{h d_{ik}}\right).$$

h controls the overall degree of smoothing and k the sensitivity to local detail. Data points in regions where the data are sparse will have flatter kernels associated to them. In other words, local bandwidths are used to reduce the bias in peaks or valleys of f and to avoid undersmoothing near the boundary of the interval to which f is restricted.

Local smoothness, pdf: Inherited from K.

E.3.7 Orthogonal series estimation

Assume that f and the weight function w are restricted to the $[a, b]$ interval and let $\{\varphi_j\}_{j=0}^{\infty}$ a family of orthonormal functions in $[a, b]$ with respect to the scalar product:

$$\langle f, g \rangle = \int_a^b w(x) f(x) g(x)\, dx$$

Then $\langle \varphi_j \varphi_k \rangle = \delta_{jk}$ and the sum $\sum_{j=0}^{\infty} c_j \varphi_j(x)$ with $c_j = \langle f, \varphi_j \rangle = \mathrm{E}\{w(x)\varphi_j(x)\}$ converges pointwise to f if $f \in \mathcal{C}([a, b])$. The "sample" coefficients $\hat{c}_j = \frac{1}{n}\sum_{i=1}^{n} w(X_i)\varphi_j(X_i)$ for $j = 0, \dots$ give a natural, unbiased estimate of f: $\hat{f}(x) = \sum_{j=0}^{\infty} \hat{c}_j \varphi_j(x)$. However, $\sum_{j=0}^{\infty} \hat{c}_j \varphi_j(x)$ converges to $\frac{1}{n}\sum_{i=1}^{n} \delta(x - X_i)$ because $\hat{c}_j = \left\langle \frac{1}{n}\sum_{i=1}^{n} \delta(x - X_i), \varphi_j \right\rangle$. In order to smooth the estimate, we apply a low-pass filter to it, typically by truncating the series expansion to k terms:

$$\hat{f}_k(x) = \sum_{j=0}^{k} \hat{c}_j \varphi_j(x)$$

where k is the smoothing parameter ($k \to 0$: very smooth, $k \to \infty$: very noisy). In this case the estimation becomes parametric in the $\{\hat{c}_j\}$.

More generally, one can define weights $\{\lambda_j\}_{j=0}^{\infty}$ such that $\lambda_j \to 0$ when $j \to \infty$:

$$\hat{f}_(x) = \sum_{j=0}^{\infty} \lambda_j \hat{c}_j \varphi_j(x)$$

where the rate at which the $\{\lambda_j\}$ converge to 0 is the smoothing parameter.

Local smoothness, pdf: Depend on the particular series and on the system of weights. In general $\hat{f}$ will not be nonnegative.

Usual families of orthogonal functions are:

- Fourier series: $\varphi_j(x) = e^{2\pi i j x}$ in $[0, 1]$ with respect to $w(x) = 1$, which can be considered as a Dirichlet kernel (see table 4).

- Legendre polynomials: given by *Rodrigues' formula*

$$\varphi_j(x) = p_j(x) = \frac{1}{j!2^j}\frac{d^j}{dx^j}(x^2-1)^j$$

in $[-1,1]$ with respect to $w(x)=1$, satisfying the recurrence relation

$$p_0 = 1 \qquad p_1 = x \qquad p_j = \frac{1}{j}\left((2j-1)xp_{j-1} - (j-1)p_{j-2}\right) \qquad p_j' = (2j-1)p_{j-1} + p_{j-2}' \qquad j \geq 2 \tag{E.1}$$

The normalised Legendre polynomials are $\sqrt{j+\frac{1}{2}}p_j$.

- Hermite polynomials: given by *Rodrigues' formula*

$$\varphi_j(x) = H_j(x) = (-1)^j e^{x^2}\frac{d^j}{dx^j}e^{-x^2}$$

in $[-\infty,\infty]$ with respect to $w(x)=e^{-x^2}$, satisfying the recurrence relation

$$H_0 = 1 \qquad H_1 = 2x \qquad H_j = 2xH_{j-1} - 2(j-1)H_{j-2} \qquad H_j' = 2jH_{j-1} \qquad j \geq 2 \tag{E.2}$$

The normalised Hermite polynomials are $H_j/\sqrt{2^j j!\sqrt{\pi}}$.

Series estimators give a compact representation of the estimates of class densities with a low number of coefficients and are particularly useful for low-dimensional projections.

E.3.8 Maximum penalised likelihood estimator

The likelihood of a curve g as a density underlying the set $\{X_i\}_{i=1}^n$ of iid observations is

$$L(g|X_1 \dots X_n) = \prod_{i=1}^n g(X_i)$$

which has no finite maximum over the class of all densities; this can be shown by taking g as close as desired to $\frac{1}{n}\sum_{i=1}^n \delta(x-X_i)$. We can define a penalised log-likelihood[37] as

$$l_\alpha(g) = \log L(g) - \alpha R(g)$$

where:

- L quantifies the goodness of fit.
- R quantifies the roughness of g, e.g. $R(g) = \int (g''(x))^2\,dx$.
- $\alpha > 0$ is the smoothing parameter.

The maximum penalised likelihood estimator is then:

$$\hat{f}_\alpha = \arg\max_{\mathcal{G}} l_\alpha(g)$$

where $\mathcal{G}$ is the set of functions such that g is a pdf and $R(g) < \infty$.

Local smoothness: As imposed by the search space $\mathcal{G}$.

pdf: Yes, by definition of $\mathcal{G}$.

Problems: Difficult calculation (implicit definition of $\hat{f}$).

[37]Something similar is done in the regularised GTM algorithm of section 5.2.3.

E.3.9 General weight function estimator

This is a generalisation of all the previous estimators, useful from the theoretical point of view. Consider a weight function $w(x, y)$ such that $w(x, y) \geq 0 \; \forall x, y$ and $\int w(x, y)\, dy = 1$. The general weight function estimator is:

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} w(X_i, x)$$

The previous estimators can be obtained with a particular election of the weight function w:

- Histogram: $w(x, y) = \begin{cases} 1/h(x) & \text{if } x,\, y \text{ fall in the same bin} \\ 0 & \text{otherwise} \end{cases}$, where $h(x)$ is the width of the bin containing x.
- Kernel: $w(x, y) = \frac{1}{h} K\left(\frac{y-x}{h}\right)$.
- Orthogonal series: $w(x, y) = \sum_{j=0}^{\infty} \lambda_j \varphi_j(x) \varphi_j(y)$.

Local smoothness: Inherited from w.

pdf: Yes.

E.4 Nonparametric density estimation techniques: Multivariate case

Several of the methods of section E.3 don't work well in higher dimensions for different reasons. For example, in the case of the multivariate histogram:

- Due to its discontinuous nature, it is difficult to visualise even in two dimensions.
- One has to specify not only the origin, but also the bin orientation and the bin length in each direction. For D dimensions, the bins are hypercubes with volume $\prod_{i=1}^{D} h_i$ and marginal bin widths h_i.
- It is not possible to draw meaningful contour diagrams of a bivariate histogram.
- If h is too small, the total number of bins becomes very large compared to the sample size, due to the curse of the dimensionality.

But *all* density estimation methods suffer in more or less degree from the curse of the dimensionality and the empty space phenomenon (see section 1.4). This means that:

- It is difficult to estimate the density except for enormous samples.
- The density itself can give a false impression of the likely behaviour of sample data sets.

The methods outlined below are easily extended to D dimensions.

E.4.1 Nearest-neighbour estimation

$$\hat{\mathbf{f}}_h(x) = \frac{k/n}{V(\mathbb{S}^D_{d_k(\mathbf{x})})}$$

$V(\mathbb{S}^D_R) = V(\mathbb{S}^D_1) R^D$ is the volume of the D-dimensional sphere of radius R (see section I.1).

E.4.2 Generalised nearest-neighbour estimation

$$\hat{\mathbf{f}}_h(x) = \frac{1}{n d_k^D(\mathbf{x})} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{d_k(\mathbf{x})}\right)$$

E.4.3 Kernel estimation

The multivariate kernel is usually a radially symmetric unimodal pdf, e.g. $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The estimator is:

$$\hat{\mathbf{f}}_h(x) = \frac{1}{n h^D} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right)$$

If the dispersion of the data is no the same in all directions, it is better to spherise the data than to complicate the kernel introducing a vector of smoothing parameters h_i.

E.5 Approximation (sampling) properties

Using the general weight function estimator of section E.3.9 and expanding the unknown density f in Taylor series, one can find approximated expressions for the mean squared error MSE and mean integrated squared error MISE (as defined in appendix B) of a given estimator. Table 5 lists the statistics of several of the smoothers presented; only the order, and not the particular full expression, is given. The decomposition (B.10) shows the **bias-variance trade-off**: to have a small bias we need a small bandwidth h (penalising oversmoothing), but to have a small variance nh has to be large (penalising undersmoothing). In other words, to have a good estimate we need a small bandwidth with enough observations. There is therefore an optimal bandwidth h_0 that minimises the MSE (and another one for the MISE).

The table also shows the effect of the curse of the dimensionality on multivariate kernels of order p and dimension D: for fixed h (in each component) we need a sample size that grows exponentially with the dimension. This makes difficult the use of kernel smoothers of even small dimension.

Smoother	Bias	Variance	Optimal bandwidth h_{opt}	MSE_{opt}, MISE_{opt}
Histogram	h	$1/nh$	$n^{-1/3}$	$n^{-2/3}$
Kernel	h^2	$1/nh$	$n^{-1/5}$	$n^{-4/5}$
Multivariate kernel	h^{2p}	$1/nh^D$	$n^{-1/(2p+D)}$	$n^{-2p/(2p+D)}$
k nearest neighbours	$(k/n)^2$	$1/k$	$n^{4/5}$	$n^{-4/5}$

Table 5: Statistics for several smoothers (only the order $\mathcal{O}(\cdot)$ for n sufficiently large is shown, not the actual value of the statistic).

Another nasty effect of the curse of the dimensionality occurs when the data are rank-deficient, due to linear correlations (a very common situation in high dimensions): it can be proven that in this case the optimal bandwidth goes to zero and the corresponding optimal MISE goes to infinity, however large the sample size n is [89].

E.5.1 Approximation properties for kernel estimators

It can be proven that, for symmetric kernels, the minimum MISE for the kernel estimator is reached at $h^5_{\text{opt}} = \frac{\beta}{\alpha^2\gamma} n^{-1}$, where

$$\alpha = \int t^2 K(t)\,dt \qquad \beta = \int K^2(t)\,dt \qquad \gamma = \int \left(f''(t)\right)^2\,dt$$

and the minimum is $MISE_{\text{opt}} = \frac{5}{4} C(K)\gamma^{1/5} n^{-4/5}$, where $C(K) = \alpha^{2/5}\beta^{4/5}$. Therefore both h_{opt} and $MISE_{\text{opt}}$ depend on the unknown density f.

Among nonnegative kernels, β and $C(K)$ are minimised by the Epanechnikov kernel $K_e(x)$ of table 4, which suggests defining the efficiency of a kernel as

$$\text{eff}(K) = \left(\frac{C(K_e)}{C(K)}\right)^{5/4}.$$

It is remarkable that the efficiency of most kernels is close to 1, as table 4 shows, so there is not much difference between kernels on the basis of MISE, and other considerations may be taken into account instead when choosing the kernel (e.g. degree of differentiability or computational effort required). This fact finds a parallel with radial basis functions in neural networks: the actual shape of the RBFs is relatively unimportant [5].

For h depending in some way on the sample size n, pointwise as well as global convergence (in probability for both the uniform and the integration sense) can be proven for kernels satisfying a few very general conditions [93].

E.6 What method to use?

Silverman [93] proposes the following rule of thumb for choosing the density estimation method:

1. First choice: the kernel estimator. It has the only practical drawback being unable to deal satisfactorily with the tails of the distribution without oversmoothing the main part of it.

2. Second choice: the variable kernel estimator, if greater accuracy in the tails is required.

F Regression Smoothing[38]

Contrarily to the case of density estimation, in regression smoothing the data are paired: $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, and it is not the distribution of the $\mathbf{x}$ or the $\mathbf{y}$ that we want to model, but the relationship between both. The learning is **supervised**. Table 6 compares density estimation and regression smoothing.

Smoothing	Data	Objective	Learning
Density	Unlabelled $\{\mathbf{x}_i\} \subset \mathbb{R}^D$	To model the distribution of X	Unsupervised
Regression	Paired $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n \subset \mathbb{R}^D \times \mathbb{R}^q$	To model the relationship $\mathbf{y} = \mathbf{f}(\mathbf{x})$	Supervised

Table 6: Density estimation vs. regression smoothing.

F.1 The multivariate regression problem

We consider the following multivariate regression problem [54]:

- **Given** n pairs of vectors $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ that have been generated from unknown models $\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i) + \boldsymbol{\epsilon}_i$, $i = 1, \dots, n$, with:
  - $\mathbf{y}_i \in \mathbb{R}^q$ multivariate **response vectors**.
  - $\mathbf{x}_i \in \mathbb{R}^D$ **independent** or **explanatory variables**, or **predictors**, or **carriers**.
  - $\mathbf{f} : \mathbb{R}^D \longrightarrow \mathbb{R}^q$ with $\mathbf{f} = (f_1, \dots, f_q)^T$ unknown smooth nonparametric functions.
  - $\boldsymbol{\epsilon}_i$ multivariate random variables with zero mean, independent of $\mathbf{x}_i$ and often assumed iid.
- **Construct** estimator $\hat{\mathbf{f}} = (\hat{f}_1, \dots, \hat{f}_q)^T$ which is a function of the data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ to best approximate the mean regression curve $\mathbf{f}(\mathbf{x}) = \mathrm{E}\{Y|X = \mathbf{x}\} = \int \mathbf{y} g(\mathbf{x}, \mathbf{y})\, d\mathbf{y} / g(\mathbf{x})$, where $g(\mathbf{x}, \mathbf{y})$ is the joint density and $g(\mathbf{x})$ is the marginal density of $\mathbf{x}$, and use it to predict a new $\hat{\mathbf{y}}$ given a new independent vector $\hat{\mathbf{x}}$: $\hat{\mathbf{y}} = \hat{\mathbf{f}}(\hat{\mathbf{x}})$.

F.2 Parametric and nonparametric regression

As in density estimation, we have two types of regression:

- In **parametric multiple regression**, the functional form of the regression curve is fixed through several parameters on whose space an optimum is searched for; for example, linear regression. This approach is valid if the model (i.e. the chosen curve) is correct —but this can be difficult to verify.
- **Nonparametric** or **model-free methods** make a few very general assumptions about the regression surface. Most nonparametric methods are based on the following two approaches:
  - *Local averaging* (e.g. smoothing by kernel, nearest-neighbours or splines): the estimate of the regression surface at point $\mathbf{x}_0$ is the average of the responses $\mathbf{y}_k$ of those observations with predictors $\mathbf{x}_k$ in a neighbourhood h of $\mathbf{x}_0$. Formally, $\hat{\mathbf{f}}_h(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n W_h(\mathbf{x}; \mathbf{x}_1, \dots, \mathbf{x}_n)\mathbf{y}_i$, with W_h a weight function depending on the smoothing parameter h and the sample $\{\mathbf{x}_i\}_{i=1}^n$ of the explanatory variables.

    This has desirable asymptotic properties, but requires many samples in a high-dimensional space.
  - *Successive refinement* (e.g. polynomial regression, recursive partitioning): a hierarchy of models of increasing complexity (i.e. degrees of freedom) is formulated. At each step, the model of the subsequent level of hierarchy that best fits the data is selected.

F.3 Nonparametric regression techniques

Many of the techniques presented here are analogous to those of appendix E.

[38]This appendix is mainly based in [45, 89].

F.3.1 Kernel regression smoothing (the Nadaraya-Watson estimator)

Estimating the joint density $g(\mathbf{x}, \mathbf{y})$ with a multiplicative kernel smoother, we obtain the following weight function:

$$W_h(\mathbf{x}; \mathbf{x}_1, \dots, \mathbf{x}_n) = \frac{1}{h} \frac{K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\hat{g}_h(\mathbf{x})}$$

with the following properties:

- W_h depends on the whole sample $\{\mathbf{x}_i\}_{i=1}^n$ through the marginal density $\hat{g}_h(\mathbf{x})$; the neighbourhood of $\mathbf{x}$ is fixed.
- The observations $\mathbf{y}_i$ receive more weight in those areas where the corresponding $\mathbf{x}_i$ are sparse.
- If $\hat{g}_h(\mathbf{x}) = 0$, we set $\hat{\mathbf{f}}_h(\mathbf{x}) = 0$.
- h controls the smoothness of the regression curve:
  - If $h \to 0$ then $W_h(\mathbf{x}) \to n$ if $\mathbf{x} = \mathbf{x}_i$ and is not defined anywhere else, i.e. $\hat{f}_h(\mathbf{x}_i) \to \mathbf{y}_i$ (interpolation).
  - If $h \to \infty$ then $\hat{f}_h(\mathbf{x}) \to \mathrm{E}\{\mathbf{y}\}$.

F.3.2 Nearest-neighbour regression smoothing

Here we have varying neighbourhoods in the $\mathbf{x}$ variable:

$$W_{ki}(\mathbf{x}) = \begin{cases} n/k & \text{if } \mathbf{x}_i \text{ is one of the } k \text{ nearest observations to } \mathbf{x} \\ 0 & \text{otherwise} \end{cases}$$

The number of neighbours k regulates smoothing, so that the larger k is the smoother the estimate is: for $k = n$, $\hat{f}_h(\mathbf{x}) = \mathrm{E}\{\mathbf{y}\}$; for $k = 1$, $\hat{f}_h(\mathbf{x})$ is a step function matching $\mathbf{y}_i$ for $\mathbf{x}_i$ and jumping in the middle between adjacent $\mathbf{x}_i$.

F.3.3 Spline regression smoothing

Assume $D = 1$: $\mathbf{x} = x \in \mathbb{R}$ and $x_1 \leq \dots \leq x_n$. Reinsch [83] proved that the variational problem (where $\lambda > 0$ is a smoothing parameter)

$$\min_{\hat{f}} S_\lambda(\hat{f}) = \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 + \lambda \|\hat{f}''\|_2^2 \tag{F.1}$$

has a unique solution $\hat{f} = \hat{f}_h(x) \in \mathcal{C}^2([x_1, x_n])$: the *cubic spline*, i.e. a set of cubic polynomials $p_i(x)$ between adjacent points x_i, x_{i+1} satisfying boundary conditions

$$\begin{aligned} p_i(x_i) &= p_{i-1}(x_i) \\ p'_i(x_i) &= p'_{i-1}(x_i) \\ p''_i(x_i) &= p''_{i-1}(x_i) \\ p''_1(x_1) &= p''_n(x_n) = 0 \end{aligned}$$

Smoothing grows with λ, from 0 (pure interpolation) to ∞ (least squares linear fit). The shape of the smoothing spline converges to the kernel [92]:

$$K(u) = \frac{1}{2} e^{-\frac{|u|}{\sqrt{2}}} \sin\left(\frac{|u|}{\sqrt{2}} + \frac{\pi}{4}\right). \tag{F.2}$$

Unfortunately, there are theoretical difficulties to extend analytically the variational problem (F.1) to several dimensions, which prevents using spline regression smoothing for $D > 2$.

F.3.4 Supersmoother

The supersmoother (Friedman [33]) is a fast, nonparametric, variable bandwidth smoother that provides piecewise interpolation. However:

- It requires storage of huge regression tables of estimated values $\{\hat{g}_k(\mathbf{x}_i^T \mathbf{a}_k)\}_{i=1}^n$ for each ridge function $\hat{g}_k$, $k = 1, \dots, j$.
- The derivative $\hat{g}'_k$ must be estimated by first-order finite differences of the estimates $\{\hat{g}_k(\mathbf{x}_i^T \mathbf{a}_k)\}_{i=1}^n$ for constant $\mathbf{a}_k$, which can become unstable.

F.3.5 Orthogonal series regression

Orthogonal series were introduced in section E.3.7. From the point of view of regression, orthogonal polynomials have the following advantages:

- They provide a smooth interpolation instead of piecewise one.
- Fast and accurate derivatives can be computed using the recursive relations for the polynomials and their derivatives.
- Contrarily to the supersmoother (section F.3.4), no huge regression tables are needed.

F.3.6 Projection pursuit regression

See section 3.7.

F.3.7 Partitioning methods (regression trees)

Partitioning methods (e.g. CART [10], ID3 [82]) operate as follows:

- Partition the input space into regions according to the data during training (typically with hyperplanes parallel to the coordinate axes). Binary partitions are common; each region is represented by a leaf of the binary tree.
- Fit a different mapping in each region (in the simplest case fit a constant).
- Optionally, prune the tree to control the complexity of the model.

G Generalised Linear Models

Given a multivariate regression problem (see appendix F) with data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^L$ and $\mathbf{t}_i \in \mathbb{R}^D$, a *generalised linear model* learns a mapping of the form $\mathbf{y}(\mathbf{x}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}$ is nonlinear and $\mathbf{W}$ is linear:

$$\begin{array}{ccccc} \mathbb{R}^L & \xrightarrow{\boldsymbol{\phi}} & \mathbb{R}^M & \xrightarrow{\mathbf{W}} & \mathbb{R}^D \\ \mathbf{x} & \longmapsto & \boldsymbol{\phi}(\mathbf{x}) & \longmapsto & \mathbf{y}(\mathbf{x}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{x}). \end{array}$$

The linear mapping $\mathbf{W}$ can be implemented as a linear, single-layer neural network with biases (see fig. 18), often called a *generalised linear network* [5].

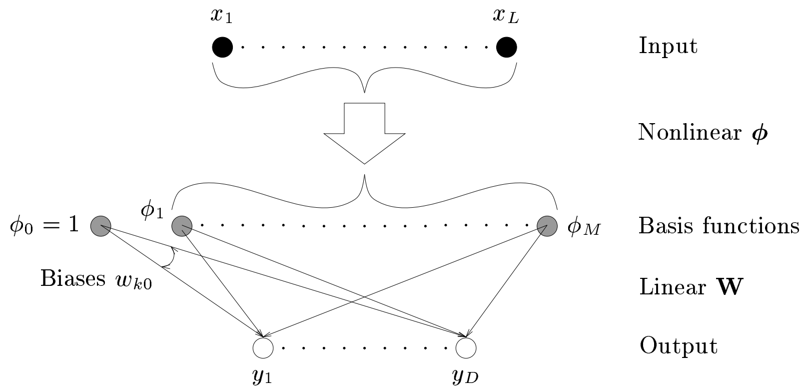

Figure 18: Generalised linear network.

Learning is as follows:

- Supervised learning for $\boldsymbol{\phi}$ using **only** the independent variables $\{\mathbf{x}_i\}_{i=1}^n$. Typically $\boldsymbol{\phi}$ will be a vector of localised radial basis functions, such as Gaussians:

$$\phi_j(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}} \qquad j = 1, \dots, M$$

  with parameters $\{\boldsymbol{\mu}_j, \sigma_j\}_{j=1}^M$ to be determined.

- Supervised learning for $\mathbf{W}$ using the full data $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^n$, according to some criterion. For the sum of squared errors (L_2-norm criterion)

$$\min_{\mathbf{W}} E(\mathbf{W}) = \sum_{i=1}^n \|\mathbf{t}_i - \mathbf{W}\boldsymbol{\phi}(\mathbf{x}_i)\|^2$$

  and assuming the ϕ_j fixed, E reaches a unique minimum value, given by the solution of the matrix equation [5]

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{W}^T = \boldsymbol{\Phi}^T\mathbf{T} \qquad \Longrightarrow \qquad \mathbf{W} = [(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^+\boldsymbol{\Phi}^T\mathbf{T}]^T$$

  where $\mathbf{T} = (t_{ik})$ is $N \times D$, $\boldsymbol{\Phi} = (\phi_{ij}) = (\phi_j(\mathbf{x}_i))$ is $N \times (M+1)$ and $\mathbf{W} = (w_{kj})$ is $D \times (M+1)$. The computation of $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^+$, the pseudoinverse matrix of $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$, can be performed in time $\mathcal{O}(M^3)$ by various standard numerical algorithms[39] (see [81] for some of them), and is usually much faster than gradient descent (backpropagation).

Generalised linear networks are, like the MLP, universal approximators, provided the basis functions are chosen appropriately. However, the number of basis functions required to obtained a given accuracy grows exponentially with the dimension L of the space of predictor variables.

[39] If $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is not singular then it is positive definite and Cholesky decomposition can be used; otherwise, one can use the somewhat slower singular value decomposition.

H Manifolds in $\mathbb{R}^n$

This appendix[40] briefly formalises the concept of k-dimensional manifold in $\mathbb{R}^n$. The main idea to keep in mind is that, while a manifold of $\mathbb{R}^n$ itself is just a subset of $\mathbb{R}^n$, it has a dimension from a geometrical point of view. Indeed, a k-dimensional manifold in $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that has only k degrees of freedom, i.e. that can be described with only k coordinates. For example, if we restrict ourselves to the case of vector subspaces, a vector subspace of dimension k can be described by a system of k linearly independent vectors (a basis); the projection of a vector $\mathbf{x}$ of the subspace onto that basis gives k real numbers, which are the coordinates of $\mathbf{x}$ in the coordinate system of that basis. Needless to say, the election of the coordinate system is not unique.

Let us now define more formally the previous ideas. First, we introduce the following naming convention:

- We will call k-manifold a k-dimensional manifold in $\mathbb{R}^n$.
- We will consider that a mapping is *differentiable* iff it is continuous and has continuous derivatives of all orders.
- A *diffeomorphism* h is a differentiable mapping $h : U \longrightarrow V$ between two open sets $U, V \subset \mathbb{R}^n$ with differentiable inverse h^{-1}.

Definition H.1. $\mathcal{M} \subset \mathbb{R}^n$ is a k-manifold iff for all $\mathbf{x} \in \mathcal{M}$ the following condition holds:

(M) There exist two open sets $U, V \subset \mathbb{R}^n$ with $\mathbf{x} \in U$ and a diffeomorphism $h : U \longrightarrow V$ such that:

$$h(U \cap \mathcal{M}) = V \cap (\mathbb{R}^k \times \{\mathbf{0}\}) = \{\mathbf{y} \in V : y_{k+1} = \cdots = y_n = 0\}.$$

For example, in $\mathbb{R}^n$:

- A point is a 0-manifold.
- A k-dimensional vector subspace is a k-manifold.
- The hollow n-sphere is an $(n-1)$-manifold.
- Any open subset is an n-manifold.

Figure 19 shows two examples of manifolds.

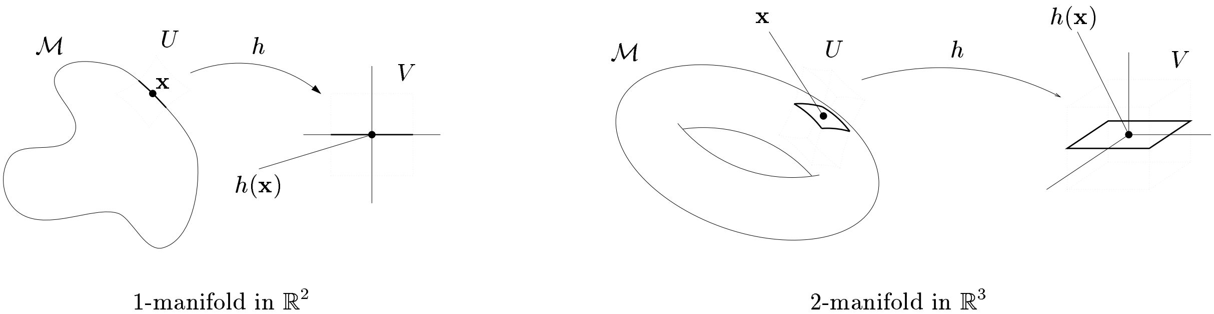

Figure 19: Examples of manifolds.

Most manifolds can be expressed by a functional formula, like the hollow unit n-sphere: $\sum_{i=1}^n x_i^2 = 1$. The following theorem helps to find the dimension in those cases.

Theorem H.1. *Let A be an open subset in $\mathbb{R}^n$ and $g : A \longrightarrow \mathbb{R}^p$ differentiable. If the Jacobian $g'(\mathbf{x})$ of g has rank p for $g(\mathbf{x}) = \mathbf{0}$, then $g^{-1}(\mathbf{0})$ is an $(n-p)$-manifold of $\mathbb{R}^n$.*

For example, $S_1^n = g^{-1}(\mathbf{0})$ for $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined as $g(\mathbf{x}) = \|\mathbf{x}\|^2 - 1 = \sum_{i=1}^n x_i^2 - 1$.

The following theorem introduces the concept of coordinate system of a manifold.

Theorem H.2. *$\mathcal{M} \subset \mathbb{R}^n$ is a k-manifold of $\mathbb{R}^n$ iff for all $\mathbf{x} \in \mathcal{M}$ the following condition holds:*

[40] Which is mainly based in Spivak's book [94].

(C) There exist two open sets $U \subset \mathbb{R}^n$, $W \subset \mathbb{R}^k$ with $\mathbf{x} \in U$, and a differentiable one-to-one mapping $f : W \longrightarrow \mathbb{R}^n$ such that:

1. *$f(W) = M \cap U$.*
2. *The Jacobian $f'(\mathbf{y})$ has rank k for all $\mathbf{y} \in W$.*
3. *f has a continuous inverse $f^{-1} : f(W) \longrightarrow W$.*

f is called a **coordinate system** around $\mathbf{x}$. f and $U \cap \mathcal{M}$ define the **coordinate neighbourhood** of $\mathcal{M}$. Figure 20 illustrates the point.

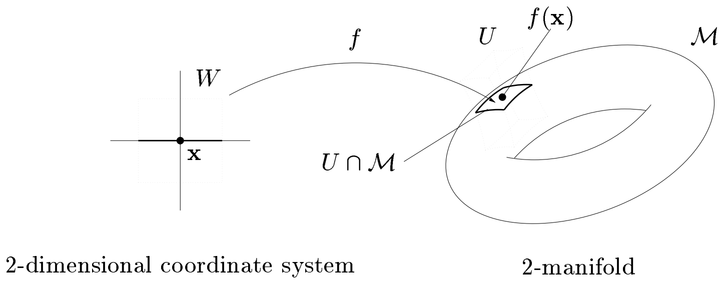

Figure 20: Coordinate system of a 2-manifold in $\mathbb{R}^3$.

Finally we introduce the **manifolds-with-boundaries**.

Definition H.2. $\mathcal{M} \subset \mathbb{R}^n$ is a k-manifold-with-boundary iff for all $\mathbf{x} \in \mathcal{M}$ either condition (M) or the following condition hold:

(M') There exist U, V open subsets of $\mathbb{R}^n$ with $\mathbf{x} \in U$ and a diffeomorphism $h : U \longrightarrow V$ such that:

$$h(U \cap \mathcal{M}) = V \cap (\mathbb{H}^k \times \{\mathbf{0}\}) = \{\mathbf{y} \in V : y_k \geq 0 \text{ and } y_{k+1} = \cdots = y_n = 0\}$$

and $h(\mathbf{x})$ has its k-th component equal to 0.

This definition separates a manifold $\mathcal{M}$ into two disjoint sets:

- The boundary of $\mathcal{M}$, $\partial\mathcal{M} = \{\mathbf{x} \in \mathcal{M} : \text{(M') holds}\}$, which is a $(k-1)$-manifold.
- The rest, $\mathcal{M} - \partial\mathcal{M} = \{\mathbf{x} \in \mathcal{M} : \text{(M) holds}\}$, which is a k-manifold.

Figure 21 gives two examples of manifolds-with-boundaries.

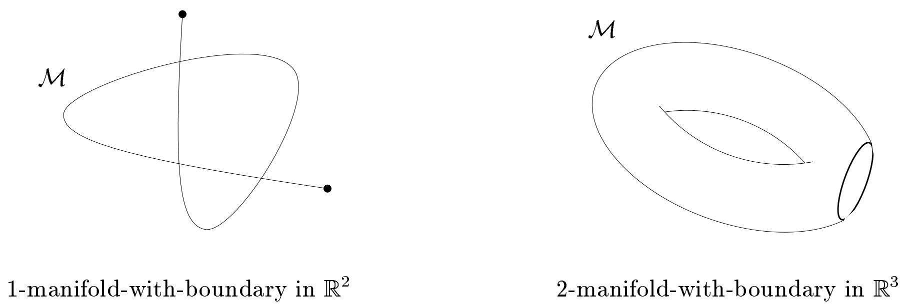

Figure 21: Examples of manifolds-with-boundary. The highlighted points belong to the boundary.

I The Geometry of High-Dimensional Spaces[41]

The geometry of high-dimensional spaces provides a few surprises. Although, in fact, one should say that the surprises are in the *usual*, *intuitive* low-dimensional cases of 1 to 3 dimensions, when compared to the general (asymptotic) case of higher dimensions.

In the following we will consider the Euclidean space $\mathbb{R}^D$.

I.1 Hypervolumes

- The volume of the D-hypersphere of radius R is $V(\mathbb{S}_R^D) = V(\mathbb{S}_1^D)R^D$ with dimension-dependent constant
$$V(\mathbb{S}_1^D) = \frac{\pi^{D/2}}{\Gamma(\frac{D}{2}+1)}$$
where $\Gamma(x)$ is the gamma function.

- The volume of the D-hypercube of side $2R$ is $V(\mathbb{C}_R^D) = V(\mathbb{C}_1^D)R^D$ with dimension-dependent constant $V(\mathbb{C}_1^D) = 2^D$.

Both volumes depend exponentially on the linear size of the object, but the constants are very different. This has as an interesting consequence a distortion of the space, as the next section shows.

I.2 Consequences in the limit of high dimensions

- *Sphere inscribed in a hypercube*: the ratio of the volume of the hypersphere to the volume of the hypercube is
$$\frac{V(\mathbb{S}_1^D)}{V(\mathbb{C}_1^D)} = \frac{\pi^{D/2}}{2^D\Gamma(\frac{D}{2}+1)} \xrightarrow[D\to\infty]{} 0.$$
That is, with increasing dimension the volume of the hypercube concentrates in its corners and the centre becomes less important. Table 7) and figure 22 show the volumes $V(\mathbb{S}_1^D)$, $V(\mathbb{C}_1^D)$ and the ratio between them for several dimensions.

- *Hypervolume of a thin shell* [98]: consider the volume between two concentric spheric shells of respective radii R and $R(1-\epsilon)$, with ϵ small. Then the ratio
$$\frac{V(\mathbb{S}_R^D) - V(\mathbb{S}_{R(1-\epsilon)}^D)}{V(\mathbb{S}_R^D)} = 1-(1-\epsilon)^D \xrightarrow[D\to\infty]{} 1.$$
Hence, virtually all the content of a hypersphere is concentrated close to its surface, which is only a $(D-1)$-dimensional manifold (see appendix H). Thus, for data distributed uniformly over both the hypersphere and the hypercube, most of the data fall near the boundary and edges of the volume. This example illustrates one important aspect of the curse of the dimensionality, introduced in section 1.4. Figure 22 illustrates this point for $\epsilon = 0.1$.

- *Tail probability of the multivariate normal*: the preceding examples make it clear that most (spherical) neighbourhoods of data distributed uniformly over a hypercube in high dimensions will be empty. In the case of the standard D-dimensional normal distribution of eq. B.4, the equiprobable contours are hyperspheres. The probability that a point is within a contour of density ϵ times the value at the mode, or, equivalently, inside a hypersphere of radius $\sqrt{-2\ln\epsilon}$, is:
$$\Pr\left[\|\mathbf{x}\|^2 \le -2\ln\epsilon\right] = \Pr\left[\chi_D^2 \le -2\ln\epsilon\right] \tag{I.1}$$
because if $\mathbf{x} = (x_1, \dots, x_D)$ is distributed as a standard normal, then x_i, $i = 1, \dots, D$ are univariate standard normal and $\|\mathbf{x}\|^2 = \sum_{i=1}^D x_i^2$ is distributed as a χ^2 distribution with D degrees of freedom. Equation I.1 gives the probability that a random point will not fall in the tails, i.e. that it will fall in the medium- to high-density region. Figure 22 shows this probability for $\epsilon = 0.01$ (a radius of 3 standard deviations) and several dimensions: notice how around $D = 5$ the probability mass of a multivariate normal begins a rapid migration into the extreme tails. In very high dimensions the entire sample will be in the tails!

[41] This appendix is greatly based in chapter 1 of Scott's book [89].

- *Diagonals in hyperspace*: consider the hypercube $[-1,1]^D$ and let any of the diagonal vectors from the centre to a corner be denoted by $\mathbf{v}$. Then $\mathbf{v}$ is one of the 2^D vectors of the form $(\pm 1, \pm 1, \dots, \pm 1)^T$. The angle between a diagonal vector $\mathbf{v}$ and a coordinate axis $\mathbf{e}_i$ is given by

$$\cos\theta_D = \frac{\mathbf{v}\mathbf{e}_i}{\|\mathbf{v}\|\|\mathbf{e}_i\|} = \frac{\pm 1}{\sqrt{D}} \xrightarrow[D\to\infty]{} 0.$$

  Thus, the diagonals are nearly orthogonal to all coordinate axes for large D.

  Pairwise scatter diagrams essentially project the multivariate data onto all the 2-dimensional coordinate planes. Hence, any data cluster lying near a diagonal in hyperspace will be mapped into the origin in every paired scatterplot, while a cluster along a coordinate axis will be visible in some plot. Thus the choice of coordinate systems is critical in data analysis: 1- or 2-dimensional intuition is valuable but not infallible when continuing on to higher dimensions.

D	1	2	3	4	$\cdots$	10
$V(\mathbb{S}_1^D)$	2	π	$\frac{4}{3}\pi$	$\frac{\pi^2}{2}$	$\cdots$	$\frac{\pi^5}{120} \approx 2.55$
$V(\mathbb{C}_1^D)$	2	4	8	16	$\cdots$	1024
$\frac{V(\mathbb{S}_1^D)}{V(\mathbb{C}_1^D)}$	1	$\frac{\pi}{4}$	$\frac{\pi}{6}$	$\frac{\pi^2}{32}$	$\cdots$	$\frac{\pi^5}{122880} \approx 0.0025$

Table 7: Volumes of unit D-hypersphere and D-hypercube.

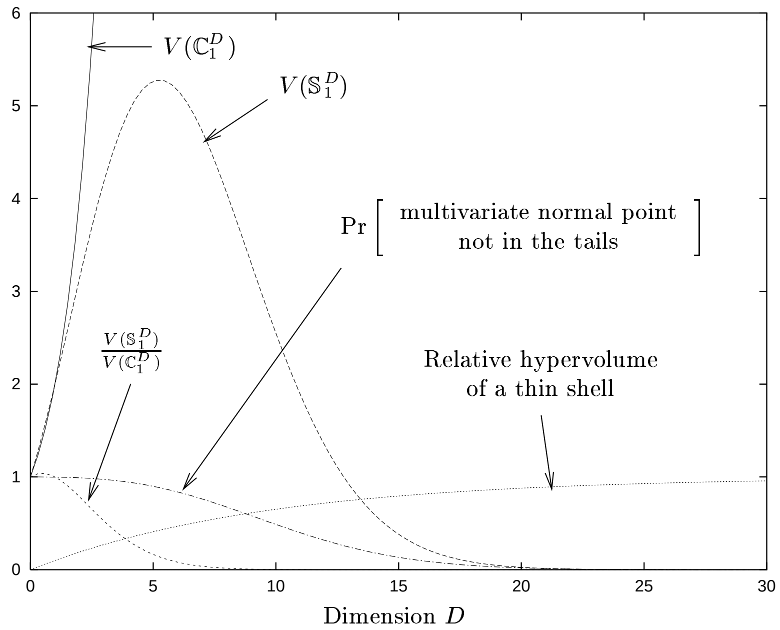

Figure 22: Dependence of several geometric quantities with the dimension (of course, only natural numbers $D = 1, 2, \dots$ are meaningful). See the main text for an explanation.

J Multidimensional Scaling[42]

Multidimensional Scaling (MDS) is a set of mathematical techniques that enable a researcher to uncover the hidden structure of data. It has applications in Psychology, Sociology, Anthropology, Economy, Educational Research, etc.

Suppose we have a set of I objects (e.g. several auditory stimuli) and that a measure of the similarity of these objects between themselves is known. This measure, called *proximity*, is a number that indicates how similar or how dissimilar two objects are or are perceived to be. It can be obtained in different ways, e.g. by asking people to judge the psychological closeness of the stimulus objects. What MDS does is to draw a spatial representation or *map* in which each object is represented by a point and the distances between points resemble as faithfully as possible the original similarity information; i.e. the larger the dissimilarity between two objects, the farther apart they should be in the spatial representation. This geometrical configuration of points reflects the hidden structure of the data and often makes it much easier to understand.

Let us consider an example. Confusions among 36 auditory Morse code signals were collected by Rothkopf [86]. Each signal consists of a sequence of dots and dashes, such as `-.-` for `K` and `..---` for `2`. Subjects who did not know Morse code listened to a pair of signals (produced at a fixed rapid rate by machine and separated by a quiet period of 1.4 seconds), and were required to state whether the two signals they heard were the same or different. Each number in table 8 is the percentage of roughly 150 observers who responded "same" to the row signal followed by the column signal. This matrix is roughly symmetric, with diagonal entries large and off-diagonal entries small, as expected, and contains the proximities data.

	A	B	C	...	0
A	82	04	08		03
B	06	84	37		04
C	04	38	87		12
...				...	
0	09	03	11		94

Table 8: Rothkopf's data on similarities among Morse code symbols.

Figure 23 (left) shows the result of applying MDS to the proximities of table 8 (from Shepard [91]) using a two-dimensional map. The 36 circles represent the points found and are labelled with the corresponding Morse code. In this case, MDS clearly shows that the data are governed by a sort of length parameter, the number of components of the signal, as well as by the individual numbers of dots and dashes.

J.1 Two-way MDS[43]

Formally, assume the input data is a (roughly) symmetric matrix of $I \times I$ containing the proximities: $\mathbf{\Delta} = (\delta_{ij})$. For an L-dimensional map, the output data will be a set of points $\{\mathbf{x}_i\}_{i=1}^{I} \in \mathbb{R}^L$, referred to some unimportant coordinate system, such that the distances $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)\ \forall i, j = 1, \dots, I$ (typically Euclidean) are as close as possible to a function f of the corresponding proximities, $f(\delta_{ij})$. MDS is called *metric* if f is linear, otherwise it is called *nonmetric*. f can be plotted together with the pairs (δ_{ij}, d_{ij}) in a scatter (or Shepard) diagram, that plots the distance in L-dimensional space versus the proximities. If the proximities are dissimilarities (i.e. dissimilar objects have a large proximity value), it will be a rising pattern; otherwise, it will be a falling one. For example, in the Morse code case the proximities are similarities; the corresponding falling scatter diagram is shown in fig. 23 (right).

The computational procedure is as follows: define an objective function *stress* to be minimised by f:

$$f\text{-stress}(\mathbf{\Delta}, \mathbf{X}, f) = \sqrt{\frac{\sum_{i,j} \left(f(\delta_{ij}) - d_{ij}\right)^2}{\text{scale factor}}}$$

where the scale factor will typically be $\sum_{i,j} d_{ij}^2$. Then, for $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_I)$, find the function that produces the minimum stress:

$$\text{stress}(\mathbf{\Delta}, \mathbf{X}) = \min_f f\text{-stress}(\mathbf{\Delta}, \mathbf{X}, f).$$

[42] This appendix draws largely from Kruskal's book [71].

[43] In *three-way MDS* we have K matrices $\mathbf{\Delta}_k = (\delta_{ij,k})$, which can correspond to different measures of the proximities (e.g. in different times or by different subjects).

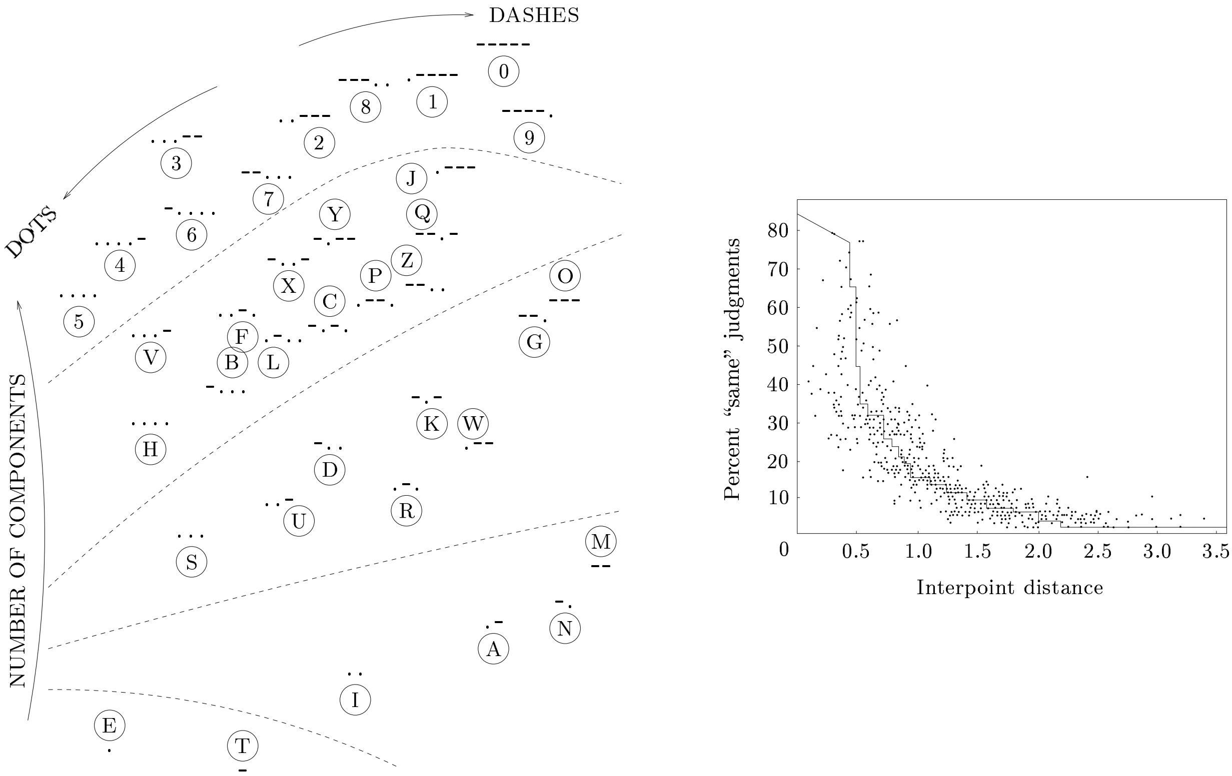

Figure 23: Left: 2D map resulting from the Morse code similarities and Shepard's interpretation of it. Right: Scatter diagram corresponding to the map.

Once f has been determined, the optimal map $\hat{\mathbf{X}}$ is found as

$$\operatorname{stress}(\boldsymbol{\Delta}, \hat{\mathbf{X}}) = \min_{\mathbf{X}} \operatorname{stress}(\boldsymbol{\Delta}, \mathbf{X}).$$

Any solution map $\hat{\mathbf{X}}$ can be freely translated and rotated (perhaps to appear in a more aesthetical way) without changing the value of the stress; therefore the actual coordinate system in the map is meaningless.

Degeneracy can happen if the objects have a natural clustering and the dissimilarities between objects in different clusters are (almost) all larger than the dissimilarities within each cluster. In this case, (almost) all points of a single cluster will converge to a single location, the stress will converge to 0 and the scatter diagram will be a staircase.

If each of the objects has an associated value v_i, (linear) regression can be performed on the generated map to further help to interpret the data: $\{(\mathbf{x}_i, v_i)\}_{i=1}^{I}$.

J.2 Selection of the map dimension

Obviously, the larger the dimension of the map is, the smaller the stress will be; however, one should keep L as small as possible so that (1) the map can be visualised and (2) no more than necessary dimensions are used (to avoid false interpretations).

Also, to assure an adequate degree of statistical stability, the dimension cannot be arbitrarily large for a given sample size. A convenient rule of thumb is that the number of (significant) pairs of objects should be at least twice the number of parameters to be estimated:

$$\frac{I(I-1)}{2} \geq 2IL \Longrightarrow I \geq 4L + 1$$

Too large or too small the dimension of the map can give a misleading view of the data. For example, points apparently clustered in a 2D map can actually lie far apart in a 3D one. A simple way to embed information from the original data in the map is to draw a line between every pair of objects whose proximity exceeds some threshold value: the presence of long, haphazardly crossing lines will indicate a discrepancy between closeness in the data and closeness in the space. Clusters will only be valid if they are consonant with the lines, i.e. points within a cluster should be well connected with each other and poorly connected with those outside the cluster.

Sometimes the lines can connect many points in some nonlinear shape, like in figure 24, which suggests that only one curvilinear dimension would be enough to give a reasonable description of the data. This is called the *horseshoe phenomenon*.

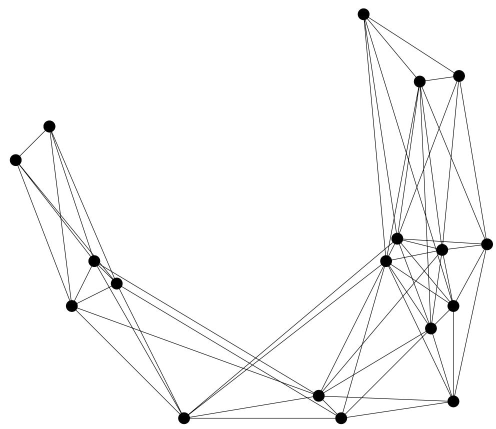

Figure 24: The *horseshoe phenomenon*.

J.3 Problems of MDS

- It is difficult to select the appropriate dimension of the map; one must try several.
- MDS does a much better job in representing large distances (the global structure) than small ones (the local structure).
- Contrarily to principal component analysis, in MDS one cannot obtain an $(L-1)$-dimensional map out of an L-dimensional one by dropping one coordinate (or, in general, by linearly projecting along some direction).

References

[1] D. Asimov, *The grand tour: A tool for viewing multidimensional data*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 128–143.

[2] P. Baldi and K. Hornik, *Neural networks and principal component analysis: Learning from examples without local minima*, Neural Networks, 2 (1989), pp. 53–58.

[3] R. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, 1961.

[4] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, *Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex*, J. Neurosci., 2 (1982), pp. 32–48.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, Oxford, 1995.

[6] C. M. Bishop, M. Svensén, and C. K. I. Williams, *EM optimization of latent-variable density models*, in Advances in Neural Information Processing Systems, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds., vol. 8, MIT Press, Cambridge, MA, 1996.

[7] ——, *GTM: A principled alternative to the self-organising map*, Tech. Rep. NCRG/96/015, Neural Computing Research Group, Aston University, 4 Apr. 1996. Accepted for oral presentation at NIPS*96 and submitted to Neural Computation.

[8] ——, *Magnification factors for the GTM algorithm*, tech. rep., Neural Computing Research Group, Aston University, 1996.

[9] H. Bourlard and Y. Kamp, *Autoassociation by the multilayer perceptrons and singular value decomposition*, Biological Cybernetics, 59 (1988), pp. 291–294.

[10] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, Calif., 1984.

[11] M. Á. Carreira Perpiñán, *Compression neural networks and feature extraction: Application to human recognition from ear images*, Master's thesis, Facultad de Informática, Technical University of Madrid, Sept. 1995.

[12] B. Cheng and D. M. Titterington, *Neural networks: A review from a statistical perspective*, Statistical Science, 9 (1994), pp. 2–30 (with comments, pp. 31–54).

[13] H. Chernoff, *The use of faces to represent points in k-dimensional space graphically*, J. Amer. Stat. Assoc., 68 (1973), pp. 361–368.

[14] D. Cook, A. Buja, and J. Cabrera, *Projection pursuit indexes based on orthonormal function expansions*, Journal of Computational and Graphical Statistics, 2 (1993), pp. 225–250.

[15] G. W. Cottrell, P. W. Munro, and D. Zipser, *Image compression by backpropagation: A demonstration of extensional programming*, in Advances in Cognitive Science, N. E. Sharkey, ed., vol. 2, Abbex, Norwood, NJ, 1988.

[16] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Series in Telecommunications, John Wiley & Sons, New York, London, Sydney, 1991.

[17] J. D. Cowan, G. Tesauro, and J. Alspector, eds., *Advances in Neural Information Processing Systems*, vol. 6, Morgan Kaufmann, San Mateo, 1994.

[18] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control, Signals and Sys., 2 (1989), pp. 304–314.

[19] P. Dayan and G. E. Hinton, *Varieties of Helmholtz machine*, Neural Networks, 9 (1996), pp. 1385–1403.

[20] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, *The Helmholtz machine*, Neural Computation, 7 (1995), pp. 889–904.

[21] P. Dayan and R. S. Zemel, *Competition and multiple cause models*, Neural Computation, 7 (1995), pp. 565–579.

[22] P. Demartines and J. Hérault, *Curvilinear Component Analysis: a self-organizing neural network for nonlinear mapping of data sets*, IEEE Trans. Neural Networks, (1996). To appear.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, B, 39 (1977), pp. 1–38.

[24] P. Diaconis and D. Freedman, *Asymptotics of graphical projection pursuit*, Annals of Statistics, 12 (1984), pp. 793–815.

[25] P. Diaconis and M. Shahshahani, *On nonlinear functions of linear combinations*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 175–191.

[26] K. I. Diamantaras and S.-Y. Kung, *Principal Component Neural Networks. Theory and Applications*, Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications, and Control, John Wiley & Sons, New York, London, Sydney, 1996.

[27] G. H. Dunteman, *Principal Component Analysis*, no. 07–069 in Sage University Paper Series on Quantitative Applications in the Social Sciences, Sage Publications, Beverly Hills, 1989.

[28] G. Eslava and F. H. C. Marriott, *Some criteria for projection pursuit*, Statistics and Computing, 4 (1994), pp. 13–20.

[29] B. S. Everitt, *An Introduction to Latent Variable Models*, Monographs on Statistics and Applied Probability, Chapman & Hall, London, New York, 1984.

[30] S. E. Fahlman and C. Lebiere, *The cascade-correlation learning architecture*, in Advances in Neural Information Processing Systems, D. S. Touretzky, ed., vol. 2, Morgan Kaufmann, San Mateo, 1990, pp. 524–532.

[31] M. K. Fleming and G. W. Cottrell, *Categorization of faces using unsupervised feature extraction*, in Proc. Int. J. Conf. on Neural Networks, vol. II, 1990, pp. 65–70.

[32] P. Földiák, *Adaptive network for optimal linear feature extraction*, in Proc. Int. J. Conf. on Neural Networks, vol. I, 1989, pp. 401–405.

[33] J. H. Friedman, *A variable span smoother*, Tech. Rep. 5, Stanford University, 1984.

[34] ——, *Exploratory projection pursuit*, J. Amer. Stat. Assoc., 82 (1987), pp. 249–266.

[35] ——, *Multivariate adaptive regression splines*, Annals of Statistics, 19 (1991), pp. 1–67 (with comments, pp. 67–141).

[36] J. H. Friedman and W. Stuetzle, *Projection pursuit regression*, J. Amer. Stat. Assoc., 76 (1981), pp. 817–823.

[37] J. H. Friedman, W. Stuetzle, and A. Schroeder, *Projection pursuit density estimation*, J. Amer. Stat. Assoc., 79 (1984), pp. 599–608.

[38] J. H. Friedman and J. W. Tukey, *A projection pursuit algorithm for exploratory data analysis*, IEEE Trans. Computers, C–23 (1974), pp. 881–889.

[39] B. Fritzke, *Growing cell structures —a self-organizing network for unsupervised and supervised learning*, Neural Networks, 7 (1994), pp. 1441–1460.

[40] ——, *Some competitive learning methods*, draft, Institute for Neural Computation, Ruhr-Universität Bochum, 24 June 1996.

[41] R. M. Gray, *Vector quantization*, IEEE ASSP Magazine, (1984), pp. 4–29.

[42] P. Hall, *On polynomial-based projection indices for exploratory projection pursuit*, Annals of Statistics, 17 (1989), pp. 589–605.

[43] W. J. Hardcastle, F. E. Gibbon, and K. Nicolaidis, *EPG data reduction methods and their implications for studies of lingual coarticulation*, J. of Phonetics, 19 (1991), pp. 251–266.

[44] W. J. HARDCASTLE, W. JONES, C. KNIGHT, A. TRUDGEON, AND G. CALDER, *New developments in electropalatography: A state-of-the-art report*, J. Clinical Linguistics and Phonetics, 3 (1989), pp. 1–38.

[45] W. HÄRDLE, *Smoothing Techniques with Implementations in S*, Springer Series in Statistics, Springer-Verlag, Berlin, 1991.

[46] T. J. HASTIE AND W. STUETZLE, *Principal curves*, J. Amer. Stat. Assoc., 84 (1989), pp. 502–516.

[47] T. J. HASTIE AND R. J. TIBSHIRANI, *Generalized additive models*, Statistical Science, 1 (1986), pp. 297–318 (with comments).

[48] ——, *Generalized additive models: Some applications*, J. Amer. Stat. Assoc., 82 (1987), pp. 371–386.

[49] ——, *Generalized Additive Models*, no. 43 in Monographs on Statistics and Applied Probability, Chapman & Hall, London, New York, 1990.

[50] K. HORNIK, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, 4 (1991), pp. 251–257.

[51] K. HORNIK, M. STINCHCOMBE, AND H. WHITE, *Multilayer feedforward networks are universal approximators*, Neural Networks, 2 (1989), pp. 359–366.

[52] ——, *Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks*, Neural Networks, 3 (1990), pp. 551–560.

[53] P. J. HUBER, *Projection pursuit*, Annals of Statistics, 13 (1985), pp. 435–475 (with comments, pp. 475–525).

[54] J.-N. HWANG, S.-R. LAY, M. MAECHLER, R. D. MARTIN, AND J. SCHIMERT, *Regression modeling in back-propagation and projection pursuit learning*, IEEE Trans. Neural Networks, 5 (1994), pp. 342–353.

[55] J.-N. HWANG, S.-S. YOU, S.-R. LAY, AND I.-C. JOU, *The cascade-correlation learning: A projection pursuit learning perspective*, IEEE Trans. Neural Networks, 7 (1996), pp. 278–289.

[56] N. INTRATOR, *Localized exploratory projection pursuit*, in Proceedings of the 23rd Conference on the Interface between Computer Science and Statistics, Seattle, 1991.

[57] ——, *Feature extraction using an unsupervised neural network*, Neural Computation, 4 (1992), pp. 98–107.

[58] N. INTRATOR, *Combining exploratory projection pursuit and projection pursuit regression with application to neural networks*, Neural Computation, 5 (1993), pp. 443–455.

[59] N. INTRATOR AND L. N. COOPER, *Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions*, Neural Networks, 5 (1992), pp. 3–17.

[60] N. INTRATOR, D. REISFELD, AND Y. YESHURUN, *Face recognition using a hybrid supervised/unsupervised neural network*, tech. rep., Dept. of Computer Science, Tel-Aviv University, 22 June 1995.

[61] J. E. JACKSON, *A User's Guide to Principal Components*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, London, Sydney, 1991.

[62] I. T. JOLLIFFE, *Principal Component Analysis*, Springer Series in Statistics, Springer-Verlag, Berlin, 1986.

[63] L. K. JONES, *On a conjecture of Huber concerning the convergence of projection pursuit regression*, Annals of Statistics, 15 (1987), pp. 880–882.

[64] M. C. JONES, *The Projection Pursuit Algorithm for Exploratory Data Analysis*, PhD thesis, University of Bath, 1983.

[65] M. C. JONES AND R. SIBSON, *What is projection pursuit?*, Journal of the Royal Statistical Society, 150 (1987), pp. 1–18 (with comments, pp. 19–36).

[66] H. F. KAISER, *The varimax criterion for analytic rotation in factor analysis*, Psychometrika, 23 (1958), pp. 187–200.

[67] N. KAMBHATLA AND T. K. LEEN, *Fast non-linear dimension reduction*, in Cowan et al. [17], pp. 152–159.

[68] S. KLINKE AND D. COOK, *Kernel-based projection pursuit indices in XGobi*, tech. rep., Humboldt-University of Berlin, error.

[69] T. K. KOHONEN, *The self-organizing map*, Proc. IEEE, 78 (1990), pp. 1464–1480.

[70] A. KROGH, M. BROWN, I. S. MIAN, K. SJÖLANDER, AND D. HAUSSLER, *Hidden Markov models in computational biology*, J. of Molecular Biology, 235 (1994), pp. 1501–1531.

[71] J. B. KRUSKAL AND M. WISH, *Multidimensional Scaling*, no. 07–011 in Sage University Paper Series on Quantitative Applications in the Social Sciences, Sage Publications, Beverly Hills, 1978.

[72] S. Y. KUNG, K. I. DIAMANTARAS, AND J. S. TAUR, *Adaptive principal component extraction (APEX) and applications*, IEEE Trans. Signal Processing, 42 (1994), pp. 1202–1217.

[73] A. A. LUBISCHEW, *On the use of discriminant functions in taxonomy*, Biometrics, 18 (1962), pp. 455–477.

[74] D. J. C. MACKAY, *Bayesian neural networks and density networks*, Nuclear Instruments and Methods in Physics Research A, 354 (1995), pp. 73–80.

[75] K. V. MARDIA, J. T. KENT, AND J. M. BIBBY, *Multivariate Analysis*, Probability and Mathematical Statistics Series, Academic Press, New York, 1979.

[76] R. M. NEAL, *Bayesian Learning for Neural Networks*, Springer Series in Statistics, Springer-Verlag, Berlin, 1996.

[77] E. OJA, *Principal components, minor components, and linear neural networks*, Neural Networks, 5 (1992), pp. 927–935.

[78] C. POSSE, *An effective two-dimensional projection pursuit algorithm*, Communications in Statistics —Simulation and Computation, 19 (1990), pp. 1143–1164.

[79] ——, *Projection pursuit exploratory data analysis*, Computational Statistics and Data Analysis, 20 (1995), pp. 669–687.

[80] ——, *Tools for two-dimensional exploratory projection pursuit*, Journal of Computational and Graphical Statistics, 4 (1995), pp. 83–100.

[81] W. H. PRESS, S. A. TEUKOLSKY, W. T. VETTERLING, AND B. P. FLANNERY, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, U.K., second ed., 1992.

[82] J. R. QUINLAN, *Induction of decision trees*, Machine Learning, 1 (1986), pp. 81–106.

[83] C. H. REINSCH, *Smoothing by spline functions*, Numerische Mathematik, 10 (1967), pp. 177–183.

[84] B. D. RIPLEY, *Neural networks and related methods for classification*, Journal of the Royal Statistical Society, B, 56 (1994), pp. 409–437 (with comments, pp. 437–456).

[85] H. ROBBINS AND S. MONRO, *A stochastic approximation method*, Annals of Mathematical Statistics, 22 (1951), pp. 400–407.

[86] E. Z. ROTHKOPF, *A measure of stimulus similarity and errors in some paired-associate learning tasks*, J. of Experimental Psychology, 53 (1957), pp. 94–101.

[87] J. W. SAMMON, JR., *A nonlinear mapping for data structure analysis*, IEEE Trans. Computers, C–18 (1969), pp. 401–409.

[88] T. D. SANGER, *Optimal unsupervised learning in a single-layer linear feedforward neural network*, Neural Networks, 2 (1989), pp. 459–473.

[89] D. W. Scott, *Multivariate Density Estimation. Theory, Practice, and Visualization*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York, London, Sydney, 1992.

[90] D. W. Scott and J. R. Thompson, *Probability density estimation in higher dimensions*, in Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface, J. E. Gentle, ed., Amsterdam, New York, Oxford, 1983, North Holland-Elsevier Science Publishers, pp. 173–179.

[91] R. N. Shepard, *Analysis of proximities as a technique for the study of information processing in man*, Human Factors, 5 (1963), pp. 33–48.

[92] B. W. Silverman, *Some aspects of the spline smoothing approach to non-parametric regression curve fitting*, Journal of the Royal Statistical Society, B, 47 (1985), pp. 1–52.

[93] ———, *Density Estimation for Statistics and Data Analysis*, no. 26 in Monographs on Statistics and Applied Probability, Chapman & Hall, London, New York, 1986.

[94] M. Spivak, *Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus*, Addison-Wesley, 1965.

[95] D. F. Swayne, D. Cook, and A. Buja, *User's Manual for XGobi, a Dynamic Graphics Program for Data Analysis Implemented in the X Window System (Release 2)*, Bellcore, Nov. 1991.

[96] A. Utsugi, *Hyperparameter selection for self-organizing maps*, Neural Computation, 9 (1997). To appear.

[97] ———, *Topology selection for self-organizing maps*, Network: Computation in Neural Systems, 7 (1997), pp. 727–740.

[98] E. J. Wegman, *Hyperdimensional data analysis using parallel coordinates*, J. Amer. Stat. Assoc., 85 (1990), pp. 664–675.

[99] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, Oxford, 1965.

[100] R. S. Zemel and G. E. Hinton, *Developing population codes by minimizing description length*, in Cowan et al. [17], pp. 11–18.

[101] Y. Zhao and C. G. Atkeson, *Implementing projection pursuit learning*, IEEE Trans. Neural Networks, 7 (1996), pp. 362–373.