UCL
Université
catholique
de Louvain

LABORATOIRE DE MICROÉLECTRONIQUE

Louvain-la-Neuve

Machine learning of high-dimensional data:

Local artificial neural networks and the curse of dimensionality

Jury

M. Cottrell
E. de Bodt
J. Hérault
Y. Kamp
P. Sobieski
V. Wertz
C. Trullemans (Président)

Michel Verleysen

Thèse présentée en vue de l'obtention du grade d'agrégé de l'enseignement supérieur.

décembre 2000

Contents

Preface

Writing this thesis was both a difficult and an interesting experience.

It was difficult because it is an attempt to collect, in a single text, several contributions realized in the last few years, in various contexts. Some of the ideas presented here come from personal, independent work, some come from work realized with Ph.D. and/or M.Sc. students, some were developed in the frame of financially supported projects having their own constraints, some are the result of common work with research teams abroad. Trying to write a coherent text with these contributions was a way to assess the strengths and weaknesses of my own work, including how individual results contribute to a common goal. Obviously, the conclusion of this effort is that... there is still much to do, both to consolidate the ideas already developed and to extend them to more general frameworks.

But writing this text was also a very interesting and stimulating experience. At many places in the text, the reader will find comments like "This is a topic for further work". Rather than pointing out deadlocks (as it often happens in the literature...), these comments indicate in which directions the work has to be continued: many ideas on how to work in these directions are mentioned. Undoubtedly, this work will give new ideas for Ph.D. and M.Sc. thesis topics, and probably for research projects too.

It has to be clear that this thesis must not be seen as the completion of a research work. On the contrary, it is a report on the current status of a work that has to be continued in the coming years. Collaboration opportunities and research projects will most probably decide the precise goals of my work in the next few years, but keeping long-term objectives in mind is a must. One of my objectives is to carry on with the development of learning methods adapted to real engineering problems, and tentatively to implement them with dedicated hardware when this is justified.

The purpose of this text is to serve as a guide for further investigation. Some concepts necessary to understand the work are detailed, even if they do not result from personal work. However, this text is not written to serve as a comprehensive survey nor as a textbook. The reader should have a medium knowledge of the

field and of the relevant literature before reading this text.

Another voluntary limitation of this text is that it contains few examples. I deliberately chose to restrict the number of illustrative examples for the following reasons.

- Everybody knows that it is always possible to find examples where the performances of a specific algorithm are higher than any other method... It can be scientific dishonesty of course, but it can also be the fact that a researcher is faced to shortcomings of other methods and tries to overcome them with a new algorithm. Even without any ill intention, the developer will be prone to publish examples proving the usefulness of his/her development.

- When a specific example is chosen as a benchmark, and even if this benchmark is widely accepted by the scientific community, making a new algorithm work "better" on this benchmark is usually a question of effort devoted to the adjustment of learning parameters, etc. In the NN literature, one can find plenty of papers proving the superiority of MLP over RBFN and plenty of examples proving the opposite. It would not even be surprising to find the same examples used for both purposes... It is easy to show the superiority of RBFN over MLP if you spend two weeks to adjust the parameters of your RBFN and use a standard commercial software as MLP, and it is just as easy to do the opposite!

Anyway, despite these limitations, the reader may still refer to the original papers for examples about the methods presented in this text! Most illustrative examples however are restricted to 2- and 3-dimensional problems, for obvious representation reasons.

A preface is not complete without acknowledgements. In the case of this thesis, the following lines are much more than acknowledgements. Most of the work described in this text has been realized either in the frame of research projects (in collaboration with people outside my own university), and/or in the frame of Ph.D. theses realized at UCL. A great part of my research work these last years was devoted to the supervision of these works and Ph.D. theses. Clearly defining what is my own work and what comes from other people is difficult, and would make no sense. Good research is the result of collaboration between people. There is a Chinese proverb saying "There is more in two heads than in one". Actually, this is probably not a Chinese proverb, but anyway...

The following parts of this thesis have been developed with, in collaboration with, or by the researchers listed below. This list is probably not exhaustive; those not listed here must know that their omission is fully unintentional... This list was written in the order of presentation of this thesis; in no case it reflects a ranking in the quality or quantity of work, or whatever!

- I had the opportunity to meet Pierre Demartines both at the EPFL during my six-months visit and at INPG, where he realized his Ph.D. thesis under the supervision of Jeanny Hérault. I worked with him a little bit at EPFL, but not really on the topics of this text. Nevertheless, his Ph.D. thesis was a source of inspiration for this work, in particular for the chapter about "High-dimensional data" and the section about "Curvilinear Component Analysis". Hopefully (for me), his tendency to drive let's say quite fast and to enjoy aerobatics is *not* a source of inspiration for me...

- The chapter on "Local learning" mostly resulted from a European Esprit project Elena, coordinated by Christian Jutten. The work on vector quantization, on the IRVQ algorithm, and on RBFN networks, has been done with Philippe Thissen and Jean-Luc Voz, the two researchers working at UCL on this project, but also with Christian Jutten (himself) and Pierre Comon. Working with them, and with all other participants to the project, has been an enjoyable and very profitable experience for me. I would like to forget the long weeks writing reports day and night, but I will never forget the technical passionate discussions leading, late in the night, to philosophical considerations... after a few beers (in Louvain-la-Neuve) or "Chartreuses" (in Grenoble)!

- Katerina Hlavackova (from the Czech Academy of Sciences) also contributed to the developments about RBFN networks. I won't say I miss the meals in her institute's canteen, but working with her was fun, especially when an engineer (who wants to make things working) and a mathematician (who wants to prove that it *can* work) try to work together...

- The chapter on "Dimension reduction" is the theme of Amaury Lendasse's Ph.D. thesis; more exactly, its purpose is to take benefit from dimension reduction on input values to a prediction method in order to increase its performances. Amaury is obstinate, always late when he has to meet deadlines, in particular when writing a paper, and often forgets to indicate the units on graph axes, but he makes a good job and scientific discussions with him are always constructive. He also provided most simulations of this chapter.

- While this is not exactly his Ph.D. topic, John Lee works on the dimension reduction techniques, namely on CCA. John is sometimes strange. He finds flower structure in Santa Fe competition benchmarks, he doesn't drink beer, and he likes programming and LaTeX. But he is very efficient and I like to work with him. He kindly provided me several graphs and simulations used in this text.

- Nicolas Donckers is working towards the Ph.D. degree, on signal processing for sensors. At least he makes me believe that he works... between two scout activities. This text does not include directly his work, but he is always ready to help anybody who needs it, including me! He dreams that I will give him the

password of my PC and let him reinstall all the software, but that will remain a dream...

Comments, ideas and suggestions from other researchers were also hepful for this work. I think particularly to Jeanny Hérault and Anne Guérin-Dugué, who commented constructively a recent publication on CCA whose ideas are inserted in this text, and to the numerous M.Sc. students whom I had the opportunity to supervise since about 10 years.

Many other people contributed more indirectly to this work. But they contributed to make me enjoy research, to introduce me to the neural network field, and to make efficient work and fun compatible. Among them, my Ph.D. supervisor, Paul Jespers, was the first one to trust me and to make me work in a field that was more or less unknown at that time in my university. I would like to thank him for his confidence and his enthusiasm.

Since the beginning of my research experience, I had the opportunity to participate to international research projects. The first one was coordinated by Jeanny Hérault, and was a wonderful experience. I owe to Jeanny most of my knowledge of the field, and of my understanding of what is scientific research. Christian Jutten coordinated the second project, and excelled in combining efficient work and real fun. These two projects have been a wonderful experience for me, and I would probably not be working as researcher without them. I would like to thank Jeanny and Christian, but also all the people having worked on these projects. It would be too long to mention them all here, but my thanks go in particular to Anne Guérin-Dugué, François Blayo, Eric Vittoz, Joan Cabestany, Victor Tryba, ... and to many others!

François Blayo occupies a special place in this list. We worked together of course, but we also initiated a lot of other scientific activities. A few beers in a Brussels' café made that the ESANN conference was born; another few (?) beers made that the Neural Processing Letters journal was created, now taken in charge by Kluwer. He also made my six-month visit as invited professor at EPFL possible. All these activities with him were a lot of fun, and I hope he has as much fun in his new professional status.

Marie Cottrell and Eric de Bodt are also wonderful people. Despite their heavy schedule, they are always available for intense and fruitful discussions. Working with them is not only efficient, it is also a pleasure.

Working at UCL would not have been possible without all colleagues in my laboratory, some directly involved in my work, some not, but all of them (should I say most of them?) contributing to a (necessary) pleasant work atmosphere.

Last, but not least, I appreciated the comments made by the members of the jury of this thesis. Rather than criticisms, they made constructive comments that

greatly helped me to write the final version of this text. I really hope that I will have the opportunity to continue to work with all of them in the future.

Chapter 1

Introduction

The field of artificial neural networks is astonishing. On the one hand, ANN are powerful new tools that may be used in a wide variety of applications, when nonlinear modelling of data is needed. As most physical phenomena are inherently nonlinear, ANN are appropriate extensions to linear data analysis tools. They have proven to be both theoretically sound, and of practical interest in real applications.

On the other hand, the same field is often disparaged, sometimes with good reason, sometimes not. One (probably the main) reason for this is the abundance of literature about this field, including many books and articles of low (or null) scientific relevance. Reading this part of the literature can put the reader in a bad mood...

Despite these criticisms, I am convinced that ANN are really interesting and deserve to be studied more deeply. And, fortunately for me, many scientists share this opinion... Of course, all depends on what is meant by ANN. A large part of the non-specialized literature associates ANN with MLP (Multi-Layer Perceptron), which is nothing else than one specific ANN model, among many others. MLP are theoretically powerful methods, but very difficult to use in practical situations; many criticisms over what has been published in the ANN field also comes from the misuse of MLP.

It is not my intention to try to formulate yet another definition of what is an artificial neural network. Many known statistical concepts have been "reinvented" under the neural network label, so that it is a sensitive task to decide what is an ANN and what is not. I prefer to keep a vague and open to criticism definition: ANN are adaptive computation methods that learn a task from examples, without trying to build a physically sound model of the task. Many data analysis methods fall into this definition, some of them being far from the traditional ANN concepts. Rather than taking a stand in this controversy, I prefer to fuel it, insisting on the fact that ANN and "traditional" statistical methods are not much different...

While ANN are potentially powerful methods, it must be stressed that most (if not all) of them are difficult to use and usually need some expertise. This difficulty

comes from the fact that they are inherently nonlinear, and that maximizing or minimizing an objective function (a common goal in ANN) in a nonlinear context implies the use of complex optimisation algorithms. Problems related to the efficiency of these algorithms are thus often mixed with the limitations of the ANN methods themselves, making it difficult to appreciate the real power of ANN.

In that context, some questions about ANN are widely covered in the literature, but some other ones are often "forgotten" from an in-depth discussion. Among these last ones, I chose to discuss a few topics of particular importance, but usually not adequately covered by current works and literature:

- what happens to learning algorithms when they have to deal with high-dimensional data?

- are local learning methods equally or better suited to high-dimensional problems?

The details of these questions, and some elements of answer, are the main contribution of this work. Of course, this should not hide other important questions about artificial neural networks, as the following ones.

- Model selection. Once an algorithm has been chosen to achieve a specific task, the number of parameters in the model has to be determined (order of polynomials, number of layers and sigmoids in MLP, etc.).

- Number of degrees of freedom. Avoiding overfitting in neural methods is usually a question of the relative number of parameters in the methods (with respect to the learning samples). The evaluation of the *effective* number of degrees of freedom in a model is however a difficult question.

- Test. In most problems, it is easy to find models fitting a given dataset. Measuring a possible overfitting is however more complicated, and implies the use of advanced test methods, like cross-validation. Despite the fact that overfitting is certainly the most crucial problem with ANN, practitioners do not commonly use appropriate test methods.

- etc.

Two main arguments in favour of the use of local models will be developed. First, even if real-world data are represented in high dimensions, it does not mean that their true number of degrees of freedom (their *intrinsic dimensionality*) is as high: dimension reduction techniques can be successfully applied, as preprocessing to both local and global models. Secondly, while learning in high-dimensional spaces remains difficult and subject to numerical problems, instabilities, local minima, etc., the intuitive view that global models require less data than local ones to "fill" the space is not justified. Moreover, it is often easier to work with local models,

because of a reduced sensitivity to design parameters or a reduced risk to be trapped in local minima.

The following of this text is organized as follows.

Chapter 2, "High-dimensional data", lists and details common problems encountered when working with data, in space whose dimension is greater than 3. These problems illustrate why intuitive views of algorithms, obviously developed when thinking about examples in dimension 2 and 3, may sometimes not generalize to higher dimension. It is the justification of this work. This chapter does not contain any original contribution, but gathers concepts that are usually disseminated in various references.

Chapter 3, "Local learning", deals with the concept of learning in specific algorithms, where each parameter is adapted according to the information contents of a few learning examples in a limited area of the learning space, rather than according to all learning examples. Three types of algorithms are detailed: vector quantization, Bayesian classification based on vector quantization, and Radial-Basis Function Networks. The first is described because it sets the basics of local learning, and the two last ones are described because they contain original contributions. As a contribution to this thesis, the algorithms are examined in the light of high-dimensional data.

Chapter 4, "Dimension reduction", considers the problem from a different angle. When methods find their limitations or fail because of the dimension of the space, a solution is to reduce this dimension to bypass the problem. This chapter presents the concept of dimension reduction, linear and nonlinear techniques for that purpose, and an original application of dimension reduction to the analysis and prediction of time-series.

Finally, chapter 5 discusses the work and opens the way to further work.

Chapter 2

High-dimensional data

2.1 Introduction

Artificial neural networks, and more generally learning mechanisms whose goal is to capture information from data, have gained considerable interest in the last two decades. The increasing power and speed of modern computers is probably one of the reasons for this interest. On one hand, it is now possible to work with large databases, and consequently to study classes of problems that were hardly tractable before. On the other hand, computers now provide tools for data analysis that are able to work in real-time, or at least with computing times compatible with real-world constraints.

The goal of artificial neural networks and of other learning methods is to perform multivariate classification or regression. Lack of information on a problem forces the use of generic methods making no assumption on the shape of the function to be approximated, or the distribution of data. In our context, such generic methods are still *parametric* models: learning is precisely the process of fitting the parameters of the model according to the data available. However, in this case, the parameters are not related to specific and tangible information in the model (like a standard deviation in a mixture of Gaussians for instance): one speaks about information distributed in the model (each parameter contributes for a small part to the global behaviour of the model). Note that some literature erroneously uses the words *non-parametric* for such generic models, because they make no assumption on the shape of the function to be approximated, unlike linear or polynomial regression.

Multivariate is the keyword of this work. It means that we are dealing with data represented by several features, or components, i.e. with multi-dimensional vectors. One could argue that it is usual to deal with multivariate data. In most real-world situations indeed, information is complex, and many features are necessary to describe data. Data analysis and statistical tools make it possible to extract information from multivariate, or high-dimensional, data, in order to make this information usable in an intuitive way. Most traditional data analysis tools

were however developed having in mind small-dimensional problems, because small-dimensional problems are easy to imagine, to draw and to interpret. Problems in dimension 20 are seldom used as illustrative examples in textbooks!

Artificial neural networks are data analysis tools developed to deal with multi-dimensional data. Contrary to most "standard" data analysis techniques, artificial neural networks are nonlinear techniques. Let us take the example of the traditional Principal Component Analysis (PCA). PCA is a linear technique; one of its goals is to project data on smaller-dimensional spaces, for easier representation and interpretation. Obviously, our brains prefer to see 2- or 3- dimensional graphs than 4-dimensional ones... The PCA projection is linear. This means that any linear dependency between features can be easily detected, and adequately overcome. However, linear tools are not adapted to detect nonlinear dependencies between features, as it will be illustrated below. Moreover, obviously again, real-world data are governed by physical (or chemical, etc.) processes that are inherently nonlinear (at least in exact sciences), except maybe in some simple, low-dimensional cases! The paradox is then that a tool like PCA is used to help in the analysis of real-world data, but is really adapted to problems which are far from the reality!

Artificial neural networks (ANN) were developed to overcome this paradox. They are built to handle high-dimensional data, and no assumption is made on linear dependencies between features. One should be honest however: ANN suffer from limitations similar to those of conventional tools, concerning their ability to work with high-dimensional data. In the literature, one can find that ANN can "beat the curse of dimensionality". In fact, it is true that ANN perform better than many other conventional tools, when the dimension of the data increases. However, nobody is able to demonstrate today that no problem arises in high-dimensional spaces, even with ANN!

The controversy is further increased by the choice between local and global ANN. Global ANN are models where each parameter influences the function realized by the network (interpolation, classification, etc.) over a wide part of the input space. On the contrary, the influence of a parameter is restricted to a specific area of the input space in local models. Typically, multi-layer perceptrons (several layers of weighted sums of sigmoid-like functions) are known as global models (because sigmoids span the whole input space), while mixture of Gaussians are known as local models (because Gaussians vanishes rapidly with the distance to their centre). In the literature, one often finds that global models are better than local ones to beat the curse of dimensionality. In this work, we will probably not prove that this claim is wrong, but we will hopefully give enough elements to convince the reader that local models may be used in high-dimensional spaces, often with more success than global models. The first argument we will develop is the fact that the intrinsic dimensionality of data is usually (much) lower than the dimension of the data vectors; using dimension reduction techniques may thus help to bypass the problems of high dimensions. Secondly, we will show that it is usually easier to work with local models, and that the intuitive view that local models require

more learning data is not justified.

2.2 One, two, three, high!

Textbooks and scientific articles illustrate learning methods and other data analysis tools on one-, two- or three-dimensional examples, and measure their performances on higher-dimensional problems for which no representation is possible. It is also widely accepted that most real-world problems are high-dimensional ones. But where is the limit between small and high dimension?

We argue that the difficulties related to high-dimensional data are already found in the fourth dimension. Of course, this is related to the fact that our mental model is used to handle two- and three-dimensional images.

Let us take two examples showing the difficulties to construct mentally an object in dimension four. Our argument is not to show that it is easier to represent an object in dimension two or three than in dimension four; this is evidence. However, we would like to demonstrate that, while it is reasonably simple to extend our view of a simple problem or object to the dimension four or more, the same extension for a real-world problem is quite impossible.

The first example shown in Figure 1 represents the three standard views of a 3-D cone. Everyone is able to mentally reconstruct the cone and imagine what it looks like in 3-D. Note that the additional information about the density of points in the three views makes it possible to see that the cone was built with points on its lateral surface only (and not in its volume nor on its base).

Figure 2 shows the same example of cone, but now in dimension four. This 4-D cone was built using random points in the volume of a 3-D sphere for the three first coordinates, and the fourth coordinate as a monotonically decreasing function of the radial coordinate of these points. With some imagination, one could guess that these figures are projections of a 4-D cone, but obviously the exercise is not so easy as in Figure 1. The difficulty arises from the fact that one sees easily that the views resemble those of Figure 1, and thus that the underlying 4-D object could be a 4-D cone. However, it would be extremely difficult to guess the shape of the 4-D object from Figure 2 without relying on the 3-D example of Figure 1.

Mental representations of high-dimensional objects are thus constructed by generalization of 2-D or 3-D examples. But what about more complex set of points or objects, even in low dimensions (higher than three)? To illustrate the difficulty of high-dimensional data representation, let us take a slightly more complex (but still far from real-world data) example.

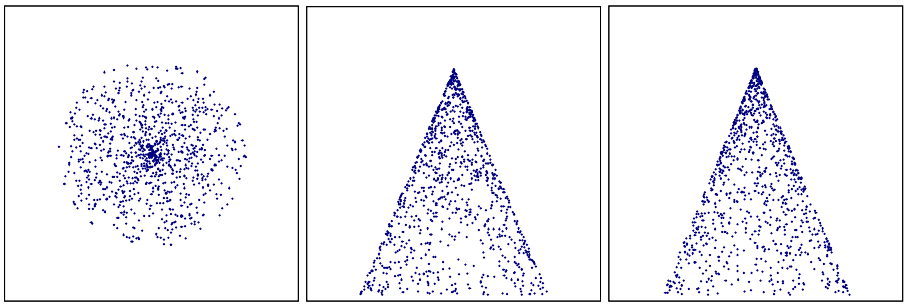

Figure 1: Three views (linear projections) of a 3-D cone. Left: [x_1-x_2] view. Centre: [x_1-x_3] view. Right [x_2-x_3] view.

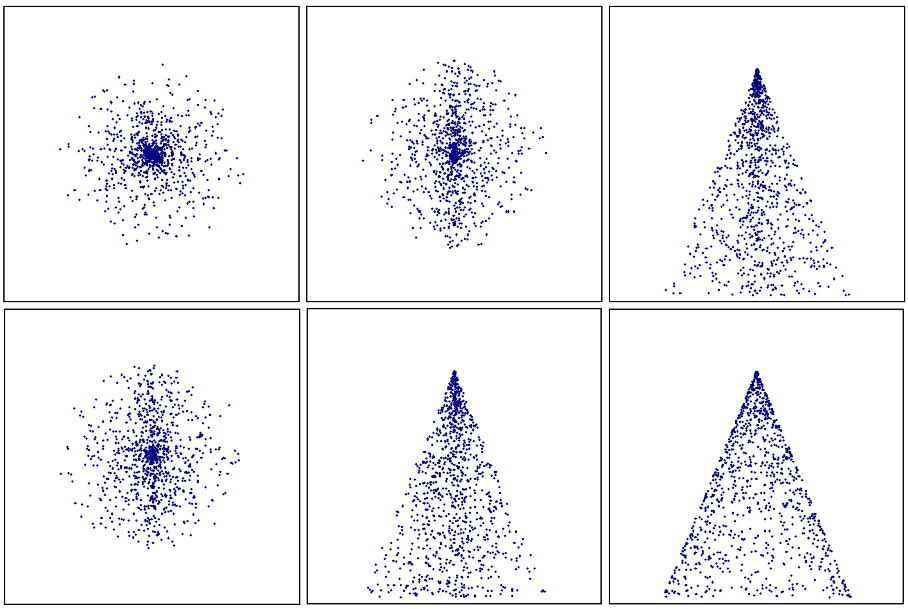

Figure 2: Six views (linear projections) of a 4-D cone. Upper left: [x_1-x_2] view. Upper centre: [x_1-x_3] view. Upper right [x_1-x_4] view. Lower left: [x_2-x_3] view. Lower centre: [x_2-x_4] view. Lower right [x_3-x_4] view.

We consider the problem of time-series prediction through a simple auto-regressive model. A sine wave of period $2\pi/10$ is built and sampled at unit time intervals:

$$f(t) = sin(10t) \tag{1}$$

The problem of time-series prediction (with an auto-regressive model) is to predict the value of the function f at time t, given the value of f at time t-1, t-2, …, t-p (p is the *auto-regressive order* of the problem). Obviously, $p = 2$ is sufficient in our sine wave example (the knowledge of the two last samples is sufficient to know the phase at time t on the sine wave, and thus the value of $f(t)$, if the amplitude and period are known). However, since the function f itself is not known in a prediction problem, one cannot guess the necessary auto-regressive order p; a sufficiently large order is then chosen, usually larger than necessary. Consider for example that the user chooses $p = 4$. At each time t, the *regressor*, formed here by the last p values of the series, is thus a 4-dimensional vector. The knowledge of the series until time t-1 means that function g defined as

$$f(t) = g(f(t-1), f(t-2), f(t-3), f(t-4)) \tag{2}$$

will be learned on t-4 vectors (regressors). These t-4 occurrences of the input vector to g can be plotted in a 4-dimensional space, leading to six representations plotted in the same way as Figure 2. Figure 3 shows these six representations, in a more realistic example where noise has been added to the true values of the past instances of function f. Again, as for the cone case in Figure 2, these graphs are not surprising once the true function f is known. Let us imagine now that we have no idea about the function f, and that the only information at hand is the set of graphs in Figure 3. One can see from the six projections that four degrees of freedom are probably not necessary to describe the data (in other words to identify uniquely a point in the regressor space); two are probably sufficient, since one can guess from Figure 3 that the intrinsic dimension of the data will probably be 2.

In this text, we will see how to estimate the size of an auto-regressive vector in time-series prediction: the estimation of a lower but still sufficient order will lead to easier interpretation, but also to easier prediction (model fitting). We will also show that linear projections as used for the representations in Figure 3 are not well suited to nonlinear problems (predicting a sine-type function is a nonlinear problem of course), and how nonlinear projections may help in this situation.

2.3 Empty space phenomenon

Problems related to high dimensions, illustrated in the previous section from the angle of data representation, may also be considered on the learning level. Artificial neural networks are learning machines: this means that whatever their goal is (function approximation, classification, etc.), ANN act as interpolators. The function to learn is known through a set of examples (the *learning set*). Obviously, the function will be easier to learn in a region of the input space including many points from the learning set than in a region where there are less points. Note that since the function to learn is only known through the learning set, any learning algorithm implicitly assumes that the learning set is representative of this function.

The meaning of "representative" here is another debate!

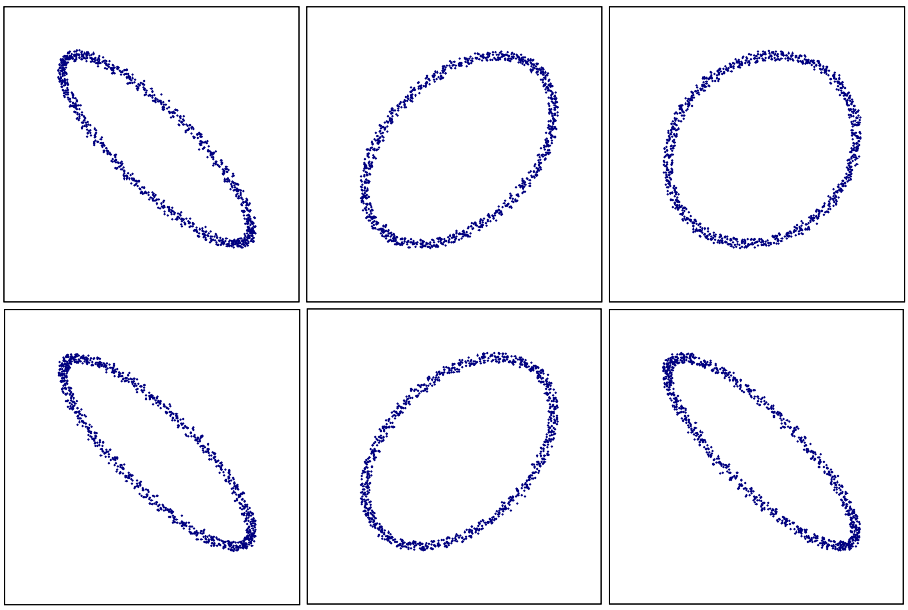

Figure 3: Six views (linear projections) of a 4-D autoregressive vector built on a sine wave (see text). Upper left: [x_1-x_2] view. Upper centre: [x_1-x_3] view. Upper right [x_1-x_4] view. Lower left: [x_2-x_3] view. Lower centre: [x_2-x_4] view. Lower right [x_3-x_4] view.

The more points you have in a region of the space, the more effective will be the approximation performed by the neural network. Note that this comment must be taken with caution: the question is not to consider a density of points per unit of volume in the input space: the same 3-D problem defined in a cube where the length of an edge is 100 or 1 after normalization of input features must lead to identical approximation quality! The question however is to see if the information contained in learning examples, in a specific region of the input space, is sufficient with respect to the smoothness of the function to approximate. Since this smoothness is itself unknown, this makes it impossible to appreciate in a quantitative way if the learning set contains enough points in this region.

The following 1-D example will illustrate this concept. Suppose that we want to generalize a 1-D function known through 10 samples (measured with noise). In the example on the left of Figure 4, it is intuitively obvious that a linear interpolation is appropriate. The figure in the centre shows the same example with different x-scale: the interpolation-generalization problem is of course identical. The figure on the right shows a 1-D problem with the same x-scale as the left figure, the same number of points, and the same level of noise. It is easy to see that the interpolating function will be more complex in this case, and thus will

require more parameters.

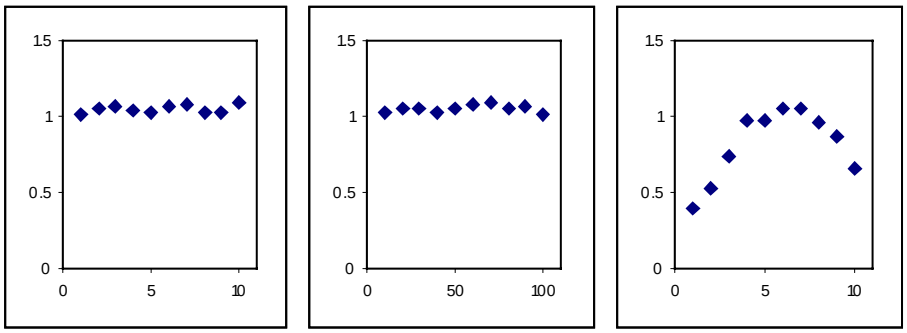

Figure 4: 1-D interpolation example. Left: 10 samples over [0-10] *x*-range, linear interpolator. Centre: same example over [0-100] *x*-range. Right: 10 samples over [0-10] *x*-range, nonlinear interpolator.

Let us imagine now the left and right examples of Figure 4, but with various numbers of points. Increasing the number of points in the first example will not influence the interpolating function. However, increasing the number of points in the second example could influence the interpolating function, depending on the position of the new points (and thus on the underlying unknown distribution). Figure 5 lower centre and right figures show two examples of different underlying distributions leading to two different interpolating functions; these two examples could however lead to the same interpolating function if the number of points is too low (undersampling – see Figure 5 lower left). On the contrary, the number of samples has no influence on a simpler interpolating function resulting from the two upper examples in Figure 5.

Intuitively, this leads to the conclusion that the more complex the underlying function is, the more points are necessary to build the interpolator. It is however important to consider the following comments. First, it is reminded that the underlying distribution is unknown, and thus that it is impossible to estimate how many samples are necessary for a "correct" interpolation. Secondly, we did not consider here the influence of the dimension of the input space: the number of samples necessary to reach a defined level of interpolation will increase with the dimension. Third, even the above discussion could be refuted on other simple or pathological examples as illustrated here. What we tried to point out here is that the problem of the number of necessary points for a correct interpolation is more prominent when the underlying function is more complex.

Extended to high-dimensional spaces, the conclusion is that the number of points available for interpolation is always too small in applications; how small is another question that is difficult to answer.

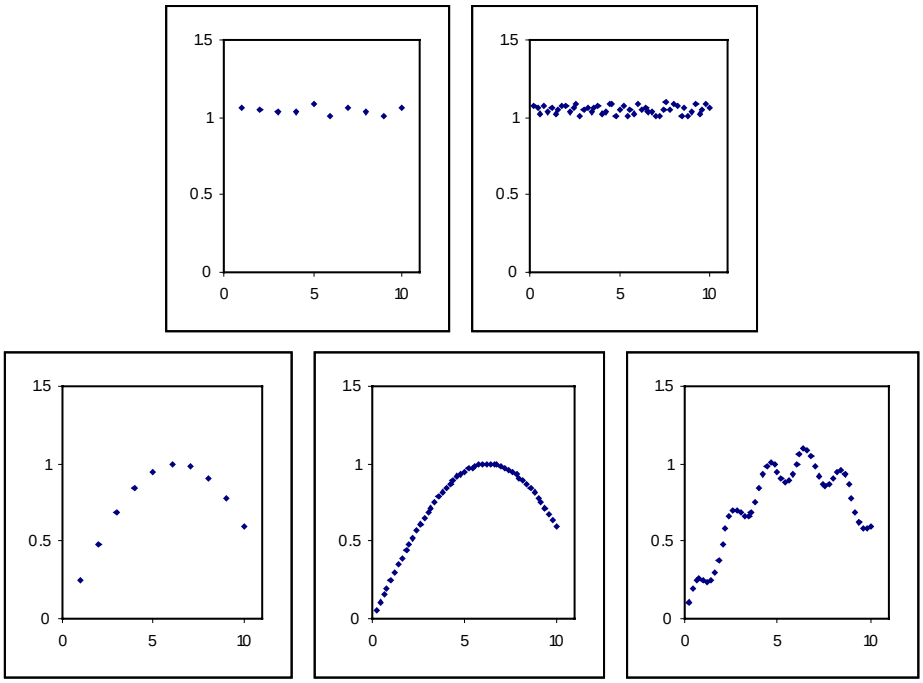

Figure 5: 1-D interpolation example with varying number of samples. Upper left and right: 10 and 100 samples respectively, linear interpolator. Lower left: 10 samples, linear nonlinear interpolator. Lower centre and right: two different distributions with 100 samples, leading to the left figure if undersampled to 10 points. Noise has been ommitted from lower figures for clarity.

Let us examine the comments about high dimensions into more details. Scott and Thompson [1] first noticed the problems related to data in high dimensions, and called them "empty space phenomenon".

Consider the volume of a sphere in dimension d. This volume is given by

$$V(d) = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d \tag{3}$$

where r is the radius, or equivalently by the recurrence equation

$$V(d) = V(d-2)\frac{\pi}{d-2} r^2 \tag{4}$$

with $V(1) = 2$ and $V(2) = \pi$. Looking at the graph of $V(d)$ when $r = 1$ (Figure 6)

leads to the surprising observation that the volume rapidly decreases towards 0 when *d* increases!

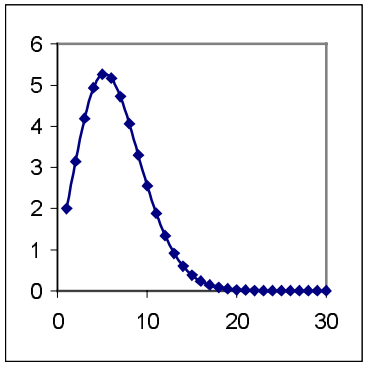

Figure 6: volume of the sphere (radius = 1) versus the dimension of the space.

Figure 6 is the standard graph found in the literature to illustrate the volume of the sphere. Nevertheless, it must be reminded that our intention here is to show that our intuitive view of the volume of a sphere is misleading in high dimension. We should thus compare this volume to a value that seems "natural" to us. One way to do this is to plot the ratio between the volume of a sphere and the volume of a cube (with edge length equal to the diameter of the sphere). Figure 7 shows this ratio.

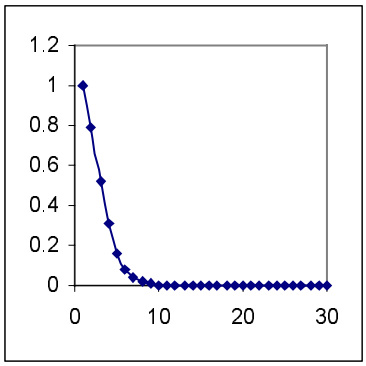

Figure 7: ratio between the volume of a sphere and the volume of a cube (length of an edge equal to the diameter of the sphere) versus the dimension of the space.

Having in mind a segment, a circle and a sphere respectively in dimension one, two and three, we understand that the ratio illustrated in Figure 7 will decrease with

the dimension of the sphere. What is more surprising is that this ratio is below 10% when the dimension is as low as 6!

Another way to consider this problem is to plot (Figure 8) the ratio between the volume of a sphere with radius 0.9 and a sphere with radius 1, versus the dimension. Obviously, this ratio is equal to 0.9 raised to the power *d*. The values plotted in Figure 8 mean that 90% of the volume of a sphere in dimension greater than 20 is contained in the spherical shell whose thickness is 10% of the initial radius!

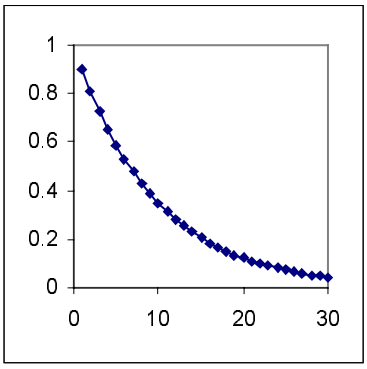

Figure 8: ratio between the volume of a sphere with radius 0.9 and the volume of a sphere with radius 1, versus the dimension of the space.

Another nice result about surprising behaviours in high dimension may be found in [Demartines 94]. It may be proved that, under soft conditions on the distribution of samples (uniform distribution is thus not required), the standard deviation of the norm of a set of samples remains almost constant when the dimension *d* of the space increases (for large dimensions). Naturally, the average Euclidean norm of samples increases with the square root of the dimension of the space. Through Chebychev's inequality, the consequence of this is that, in large dimensions, the samples seem to be normalized (see [Demartines 94] for details. This last comment will be of importance when we will deal with the problem of dimension reduction as a preprocessing to classification tasks (see chapter 4).

A last remark concerns Gaussian functions in high dimensions. We will see in the following of this text that Gaussian functions may be efficiently used as kernels in function approximators (the approximator is a weighted sum of Gaussian functions with adequate parameters). But what is a Gaussian function in high dimension? Intuitively Gaussian functions are used for their local properties: most of the "volume" (integral) of the function is contained in a limited volume around its centre. It is well known that 90% of the samples of a normalized scalar Gaussian distribution fall statistically in the interval [-1.65, 1.65]. What is less obvious is that

this percentage rapidly decreases to 0 with the dimension of the space! Figure 9 shows the percentage of samples of a Gaussian distribution falling in the sphere of radius 1.65, versus the dimension of the space. In dimension 10 already this percentage is below 1%!

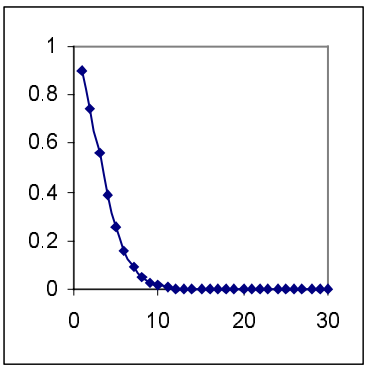

Figure 9: percentage of samples from a Gaussian distribution falling in the sphere of radius 1.65, versus the dimension of the space.

In other words, when the dimension increases, most of the volume of a Gaussian function is contained in the tails instead of near the centre! This suggests that a Gaussian function could not be appropriate in high dimensions, at least regarding the local character mentioned above. A solution could be to use super-Gaussian functions, as for example the kernels considered in [Comon 94].

Our discussion about the problems related to high dimensions is also intended to point out to the necessity for many samples (and even more...) when the dimension increases. It is extremely difficult to tackle the problem of finding the number of samples required to reach a predefined level of precision in approximation tasks. Intuitively, we understand that this number could increase exponentially with the dimension of the space. Looking back to Figure 5, and imagining a function of the same complexity in increasing dimensions, it seems natural that N^2 and N^3 samples are needed respectively in dimension 2 and 3, if N samples are necessary in dimension 1, for a defined quality of approximation.

Silverman [1986] has addressed the problem of finding the required number of samples for a specific problem in the context of the approximation of a Gaussian distribution with fixed Gaussian kernels. Silverman's results are summarized in Figure 10.

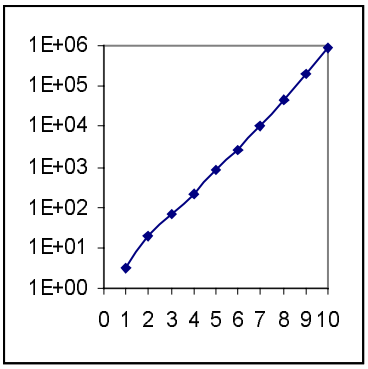

Figure 10: Number of samples required to approximate a Gaussian distribution with fixed Gaussian kernels, with an approximate error of about 10% (according to [Silverman 86], versus the dimension of the space.

Silverman's results can be approximated by [Comon 94]

$$\log_{10} N(d) \cong 0.6(d - 0.25). \tag{5}$$

In practice, any data set that does not grow exponentially with the dimension of the space will be referred to as *small*, or conversely, the dimension will be said to be *large*.

This section will be concluded by an important comment. The above discussion could make the reader think that one never has enough samples in high dimension. Consider indeed a problem where 10 samples would be required in dimension 1 and 100 in dimension 2 (imagine a segment and a square "filled" by respectively 10 and 100 samples). This would mean that 10^{10} and 10^{20} samples would be required in dimensions 10 and 20 respectively! This is obviously impossible in real-world problems (while dimensions much greater than 20 are common). However, it appears that real-world problem do not suffer so severely from the "curse of dimensionality" problem (while its importance should not be neglected tough). This simply means that the data are located near a manifold of dimension smaller than d. Reducing the dimension of the data to a smaller value than d, in order to decrease the problems related to high dimensions, is a key topic in this work.

2.4 Redundancy

A key issue in nonlinear data processing will be to reduce the dimension of the input space, under the assumption that data are located near a submanifold of the space. But why wouldn't we work on the problem of data representation itself

instead of reducing the dimension of the data after collecting them? In other words, would it be possible to build the problem or the application in such a way that the dimension of the working space is adjusted to the data?

Two major reasons prevent us to do this. First, obviously, the ideal representation of data is problem-dependent, and it is even not obvious that this ideal representation could be found in all situations, even with a large effort. But another reason must also be mentioned. In many problems, data are issued in one way or another from sensors, possibly after some transformation. Sensors are physical or chemical devices that are not "perfect": they are sensitive to several quantities, they are inherently nonlinear, they are noisy, they can break down, etc. The redundancy introduced by several sensors is thus necessary if we want to get rid of (some of) these imperfections! Using several sensors measuring different quantities is a common idea in many implementations of "smart sensors".

Figure 11 shows how redundancy in measurements can reduce the uncertainty on the measured values due to noise. The quantities to measure by the system are x_1 and x_2. Three sensors *A*, *B* and *C* each measure combinations of quantities x_1 and x_2. Measured values are indicated on the axes, and it is assumed that noise could affect the measurements in such a way that the true values are contained in an interval centred on the measured ones, as indicated by the dashed lines. The use of only two sensors *A* and *B* would lead to an uncertainty on the true values of x_1 and x_2 indicated by the area hatched in blue. The use of a third sensor *C* strongly reduces this area as indicated in red. Note that we assumed here that *A*, *B* and *C* are linear combinations of the quantities x_1 and x_2 (rotation of the axes), and furthermore that these combinations are known (we know how to place the *A*, *B* and *C* axes in the x_1 and x_2 space). The same conclusion could be drawn with nonlinear combinations of quantities (while it is more difficult to illustrate the phenomenon...). Dealing with unknown combinations is the subject of PCA (Principal Component Analysis) and ICA (Independent Component Analysis).

Of course, redundancy in measurements can also help when some sensors give no or meaningless information, when they break down, or simply when the combinations are unknown (in other words when the submanifold close to the data is unknown).

As already mentioned, our goal will be to reduce the dimension of data as much as possible before further processing. But a "blind" reduction must only be considered after all information about the problem has been taken into account! It must be clear that in the context of nonlinear data processing, no single step in the processing will be easy or straightforward. Each step will involve difficult parameter adjustments, questionable choices of a quality criterion, and numerical pitfalls. Any "easy" way of transforming the information into a more tractable form is thus welcome!

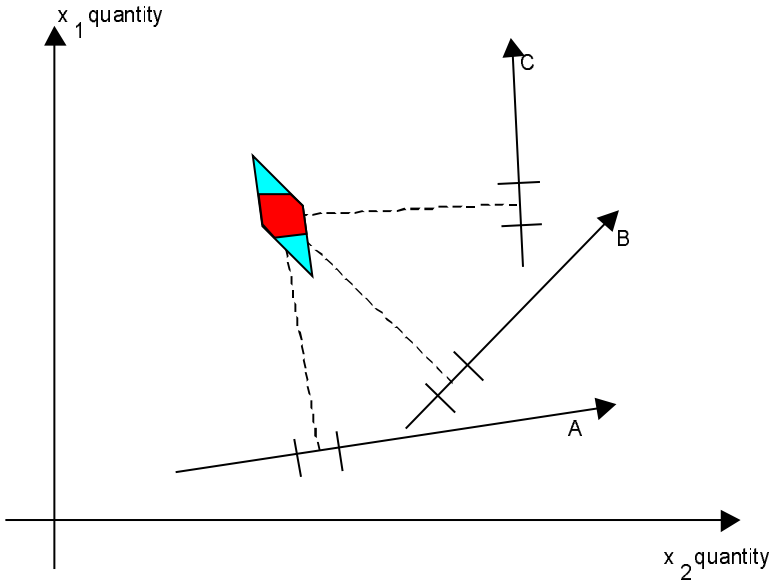

Figure 11: Three sensors *A*, *B* and *C* measuring two unknown quantities x_1 and x_2. The coloured area in blue shows the uncertainty on the measured quantities with two sensors, and the coloured area in red with three sensors (see text).

The next step after having considered all known structural information on the data is a "blind" reduction of the dimension of the space. Blind means that we make the hypothesis that data are situated near a submanifold, but that we have no idea about the dimension of this manifold nor a fortiori about its location and shape. For most dimension reduction techniques, we will need to estimate first the dimension of the manifold that will be used to project the data. This dimension is usually referred to as the *intrinsic* dimension of the data.

2.5 Intrinsic dimension

Without any idea about the shape of a submanifold, the only way to estimate its dimension is to use the data themselves rather than any hardly measurable property of the dataset (for example the number of significant eigenvalues of the data covariance matrix as used in PCA). When looking at the 3-D distribution in Figure 12, we are convinced that the intrinsic dimensionality of the data is two. An intuitive view of the dimensionality could be the number of degrees of freedom necessary to describe the data if curvilinear axes could be found.

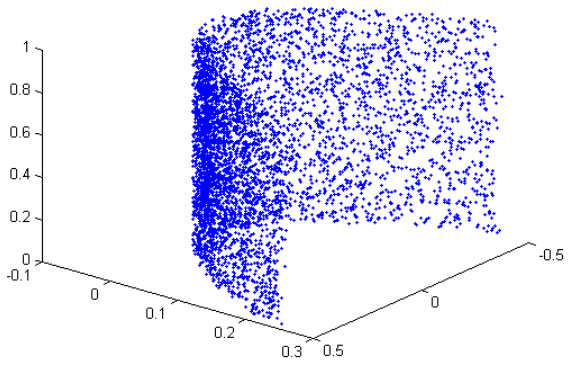

Figure 12: Horseshoe 3-D distribution with intrinsic dimension equal to 2.

The following paragraphs present four ways to estimate the intrinsic dimension of a dataset. It is important to notice that the intrinsic dimensionality is and must remain a local concept: in most distributions, it happens that different clusters of points have strongly different intrinsic dimensions (imagine for example two 3-D spheres connected by a rope in a 3-D space).

2.5.1 Local PCA

Principal Component Analysis is a linear projection method (see section 4.2). Its purpose is to project a dataset on successive axes chosen in such a way that the variance of the projected points is maximized (or equivalently in such a way that the projection error is minimized). A useful feature of PCA is that the percentage of the initial variance of points kept by the projection is easily measured by the normalized sum of the eigenvalues of the covariance matrix associated to the projection axes (these axes are the eigenvectors of the same matrix).

For example, when a set of 3-D points situated near or on a plane is projected on the two first axes found by the PCA method, nearly 100% of the variance will be kept. This will be proven by the fact that the third eigenvalue of the covariance matrix is negligible compared to the two first eigenvalues; the conclusion will be that two coordinates are sufficient to describe the data, therefore that the intrinsic dimension of the dataset is two.

The drawback of PCA is however that it is a linear method. In other words, the above conclusion can easily be drawn if the dataset is situated near a linear submanifold of the data space, but not if the submanifold is nonlinear as illustrated in Figure 12. However, since the intrinsic dimension must be considered locally (see previous section), it is possible to perform local PCA [Karhunen 99] in small regions of the space, and to deduce the local dimension according to the ratio of

eigenvalues as described above. The main limitation of this method is its computational load: each local PCA requires the inversion (or the diagonalisation) of a local covariance matrix!

2.5.2 Box counting

Another way to estimate the local dimension of a dataset is to recall that the volume of a d-dimensional cube is proportional to r^d, where r is edge of the cube. If we can make the hypothesis that the local density of points is constant over regions of "sufficient" size, the number of points contained in such a cube will be proportional to r^p where p is the intrinsic dimensionality. Imagine for example a 2-D surface in a 3-D space: any 3-D cube of increasing edge will contain a number of points growing with the square of the edge length, provided that the intersection of the cube with the distribution is approximately plane. A graph of the logarithm of the number of points contained in such a cube versus the logarithm of its edge length will thus have a slope equal to the local intrinsic dimensionality.

The estimation of the local dimensionality at each sample of the input space obviously requires the computation of the mutual distance between any pair of points, which is computationally cumbersome when the dataset is large. Conversely, when the dataset is small, the condition of constant density over "sufficiently large" regions of the space is seldom achieved.

An easier method to implement is the so-called box-counting method [Demartines 94], based on the following principle. The d-dimensional input space is divided into d-boxes of decreasing size. When the size of the box will be small compared to distance between neighbouring points in the dataset, the number of non-empty boxes will be proportional to r^{-p} where r is the length of an edge. For example, the number of boxes intersecting a 1-D string in a 3-D space is roughly equal to the length of the string divided by r, while the number of boxes intersecting a 2-D surface in a 3-D space is roughly equal to the area of the surface divided by r^2. A graph of the logarithm of the number of non-empty boxes versus the logarithm of the inverse of r will thus have a slope equal to the intrinsic dimensionality.

The Grassberger-Procaccia method [Grassberger 83] is similar to the box-counting procedure while different in its implementation. In the Grassberger-Procaccia method, all distances between any pair of points are computed. The method then leads to a global intrinsic dimensionality (instead of a local one). However, the advantage is that N points give $N(N-1)/2$ mutual distances, so that the number of points necessary to have an acceptable estimation of the intrinsic dimension is lower than in the box-counting method.

While the box-counting method is probably more convenient to implement than the Grassberger-Procaccia one, it suffers from two drawbacks in practice. First, when the intrinsic dimension is smaller than the dimension of the space (which is

usually the case when the goal is to estimate the intrinsic dimension!), the number of empty boxes dramatically increases, leading to an inefficient computational load. Secondly, the local character of the intrinsic dimension is only preserved when the largest of the boxes considered in the method is small enough to keep the constant density hypothesis valid (this makes the first drawback even worse).

Figure 13 shows the box-counting method applied to the intrinsic dimension estimation of a 1-D string in a 2-D space (from [Demartines 94]).

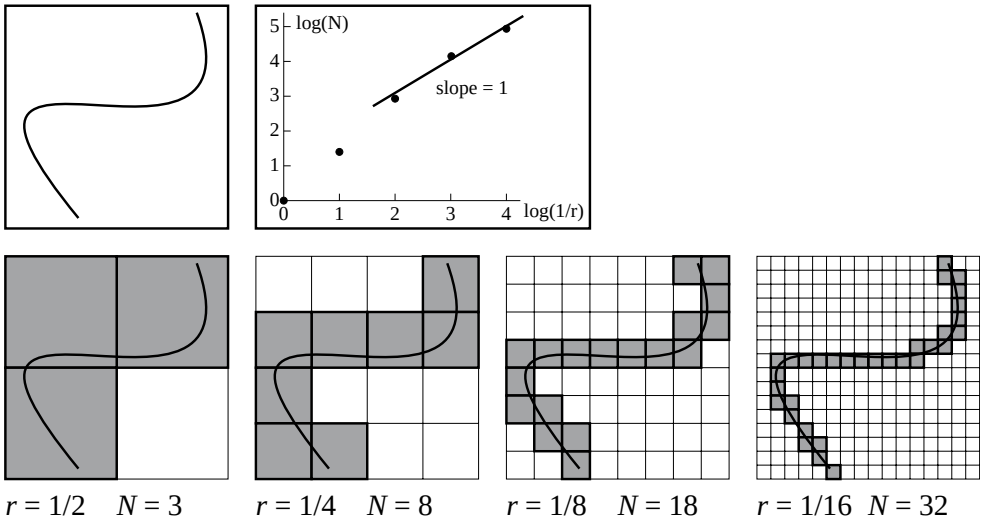

Figure 13: intrinsic dimension estimation of a 1-D string in a 2-D space by the box-counting method (from [Demartines 94]).

2.5.3 A posteriori dimension estimation

We consider a different way to estimate the intrinsic dimension of a dataset. In most situations, the information about the intrinsic dimension is required for a further projection of the data space onto a smaller dimensional one. Contrary to PCA, nonlinear projection methods require determining the dimension of the projection space before computing the projection itself.

A simple idea is then to try the projection for several dimensions of the projection space, and to evaluate the results. Projections onto spaces of higher dimension than the intrinsic one will be "good", in the sense that the projection will be nearly bijective. Projections onto space of smaller dimension than the intrinsic one will be "bad", in the sense that points far from each other in the input space will be eventually projected to the same or similar locations in the projection space.

This method presents two major difficulties. First, it is clear that it will be computationally intensive, since the projection method itself must be considered for several dimensions of the projection space, rather than once. Secondly, an appropriate criterion must be defined to evaluate the quality of the projection; furthermore, even with an adequate criterion, the limit between "good" and "bad" projections may be difficult to set in practical situations.

Nevertheless, it has a strong advantage. As written above, the knowledge of the intrinsic dimension is often a prerequisite for a further projection of the data. Linking the measure of the intrinsic dimension to the projection makes sure that the right dimension measure is evaluated (we remind that the concept of intrinsic dimension is related to its definition and the way to evaluate it).

This method to evaluate the intrinsic dimension of a dataset can be based on any projection method. The following examples are further detailed in Chapter 4, dealing with dimension reduction through non-linear projections. Indeed the same methods are used; in Chapter 4, the result of the projection is looked for, while the interest here is to determine which is the lower dimension of the projection space leading to a "good" projection.

A first example is Kohonens' self-organizing maps [Kohonen 95]. A Kohonen map is a vector quantization algorithm with a supplementary feature: each centroid (codeword) of the quantization is labeled on a 2-D (usually) grid. The purpose of this feature is a topological property: after convergence of the so-called Kohonen algorithm, two close points from the initial distribution will be projected on either the same centroid (as in classical vector quantization), either on two different centroids which are close on the 2-D grid. Besides advantages concerning the speed of convergence compared to classical adaptive VQ [de Bodt 99], the Kohonen algorithm can be used efficiently in various applications as for instance in image compression [Amerijckx 98].

Kohonen maps may be seen as a projection method, if the projection space is taken as the space of indexes on the grid. The difficulty in using Kohonen maps for nonlinear projection comes from the fact that there is no good criterion that measures both the quality of quantization and the topological property described above. Any such criterion will always result in a compromise between both objectives that must be considered together in the case of nonlinear projection. Furthermore, Kohonen maps are widely used with 2-D grids and sometimes with 1-D or 3-D meshes, but rarely with grids of dimension greater than three. This severely restricts the use of Kohonen maps in the context of nonlinear projection.

Another class of algorithms for nonlinear projection is based on Sammon's mapping [Sammon 69] and MDS (Multi-Dimensional Scaling). Unlike Kohonen maps where the inherent vector quantization also reduces the number of vectors in the dataset after projection, Sammon's mapping projects the N d-dimensional samples of a database on N p-dimensional points (with $p < d$), using a pairwise distance criterion: the distance between two points after projection must be made

equal (as much as possible) to the distance between the two corresponding samples in the original database. The CCA (Curvilinear Component Analysis) [Demartines 97] that will be detailed in a following chapter is a powerful extension of Sammon's mapping. In our context of intrinsic dimension estimation, CCA has the advantage that the quality of projection is directly measured by the criterion minimized by the algorithm, i.e. a (possibly weighted) sum of errors defined as the difference between a distance in the original space and the corresponding distance after projection.

Whether we use Kohonen maps, Sammon's mapping, CCA or any other algorithm for nonlinear projection, determining the intrinsic dimension of a set of points by trial and error is of course computationally heavy. Nevertheless, it must be reminded that the intrinsic dimension estimation is usually required for a further processing; there is nothing more natural than to use this further processing as a measure of the quality of estimation, rather than any other less adequate measure! This method has been applied to the detection of the number of independent signals in mixtures before blind source separation [Donckers 99]; Figure 14 shows the projection error (in logarithmic scale) versus the dimension of the projection space, for a mixture of two signals as detailed in [Donckers 99]. The lower dimension leading to a minimum of the projection error is taken as the intrinsic dimension; this dimension is equal to the number (two) of independent signals.

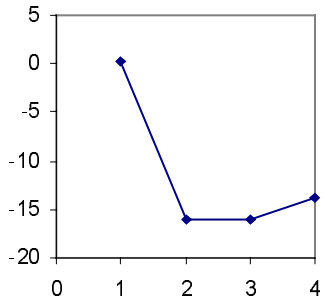

Figure 14: logarithm of the projection error versus the intrinsic dimension of data, in a source separation application [Donckers 99].

2.5.4 Limitations to the concept of intrinsic dimension

While the concept of intrinsic dimension is of great interest and practical use in adaptive and learning methods, all methods to estimate this dimension suffer from the same drawback which can be summarized as follows: one never has enough samples to be confident in the intrinsic dimension estimation...

The definition itself of intrinsic dimension has been purposely kept vague: while all methods presented above are of course linked [Hentschel 83], there is no guarantee that two methods will give the same result on the same datasets. Taking the "right" slope in Figure 13 can already be difficult and subject to subjectivity.

But even if the definition of the intrinsic dimension and the way to estimate it are clear enough, the limited size of the dataset can be a problem. It is clear from the dataset in Figure 12 that its intrinsic dimension is equal to two. This intuitive view comes from the fact that we are looking at the distribution under a correct "scale" (looking to the same distribution from too far would make us see a point, while looking from too close would make us see a 3-D set of points, the third dimension being the thickness of the surface). The estimation of the intrinsic dimension of this dataset, through the box-counting method for example, requires the use of boxes whose size is adapted to the dataset, i.e. simultaneously much smaller than the size of the cube embedding the dataset, and much larger than the thickness of the surface. While these two inequalities can easily be verified in the example of Figure 12, this could become less obvious in real-world examples. The problem is made worse in higher dimensions (the number of boxes necessary for a good estimation grows exponentially with the dimension of the space). We insist on the fact that these problems are related to the data themselves, much more than to the estimation method. They make thus the a posteriori estimation of the intrinsic dimension a probably better candidate than other methods, since it is more adapted to what will be subsequently done with the data.

2.6 Conclusion

This chapter presents a set of known issues related to data in high-dimensional spaces. It shows through examples why the intuition we may have on examples in dimension 2 and 3 cannot be transposed to higher dimensional spaces. This chapter also defines the "empty space phenomenon", i.e. the fact that the number of data available in real applications is never sufficient for learning, at least in theory.

For these reasons, many data analysis tools based on learning mechanisms may fail with high-dimensional data. However, data represented in high-dimensional spaces do not necessarily fill the whole space. If they are located near a submanifold, a convenient way to bypass some of the problems is to identify the dimension of the submanifold (the *intrinsic dimension* of the dataset, introduced in this chapter), and to project the data in a lower dimensional space. If such projection is not possible, one has to develop learning methods dealing with high-dimensional data. These two possibilities are discussed in chapters 4 and 3, respectively.

Chapter 3

Local learning

3.1 Introduction

Learning is the way to adapt parameters in a model, in function of known examples. *Learning* is the term used in the neural network community, while researchers in identification speak about *estimation*.

The literature often uses the term "local learning" for methods using combinations of local functions, as opposed to combinations of functions which span the whole input space. For example, RBFN (Radial-Basis Functions) networks use Gaussian functions which are considered as local since they rapidly vanish when the distance from their centres increases; RBFN networks are referred to as local. On the contrary, MLP (Multi-Layer Perceptrons) networks use sigmoid functions or hyperbolic tangents, which never vanish; MLP are referred to as global.

There is a tradition in the literature to consider that local models are less adapted to large-dimensional spaces than global ones. The reason for this is the empty space phenomenon. It has been shown intuitively in the previous chapter that the number of samples required for learning grows exponentially with the dimension of the space. This corresponds to our intuitive view of "filling" a space with local functions like Gaussian ones: if it is assumed that a Gaussian function corresponds to a fixed volume in the space, "filling" a distribution means to juxtapose a number of Gaussian functions proportional to the volume spanned by the distribution.

On the contrary, there is also a tradition in the literature to consider that global models do not have this limitation. Sigmoids (for example) span the whole input space, so it is assumed that a lower number of sigmoids is needed to fill the volume spanned by a distribution. We are convinced that this view of the problem is not correct; the following paragraphs explain three arguments in this direction.

First, it must be reminded that neural networks are interpolation tools, aimed to generalize information gathered on known data to other locations in the space.

Interpolation means that we must have sufficient information in the surroundings of a point in order to interpolate at that point. Look for example at Figure 15. Both the plain and dashed lines are good approximators (interpolators) of the learning data (8 markers on the figure), while they are based on different assumptions (models) on the data. However, despite the fact that learning data range from $x = 0.5$ to $x = 1.7$, the plain and dashed lines give very different approximations around the value $x = 1$; in this case, we speak about extrapolation instead of interpolation. Of course, the example in Figure 15 is obvious, and should not even be commented. Nevertheless, we remind that our point is to show the difficulties of learning in high-dimensional spaces. To imagine how a distribution looks in high dimension, and in particular if the space is "filled" with data or not, is not obvious at all. Filling the space is related both to the density of points in the distribution, and to its convexity. In most situations, it is quite impossible to appreciate if we have to speak about interpolation or extrapolation.

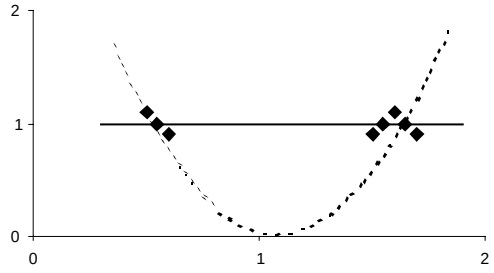

Figure 15: interpolation when the distribution of data is not convex. Plain and dashed lines are good interpolators of learning points, but extrapolate differently around $x = 1$.

Having this in mind, it is now clear that interpolation can be achieved only in regions of the space where there are "enough" data to interpolate; the words "richness of data" are sometimes used to define this concept. Even if the approximation function itself spans the whole space (for example in the case of sigmoids), interpolation has no sense in empty regions. The argument that sigmoids span a larger region than Gaussian functions is thus meaningless. Furthermore, the "lack of response" (more precisely outputs near to zero) of combinations of local functions in empty regions of the space could be used to appreciate the fact that there is not enough data to interpolate correctly.

The second argument is better explained by simple graphs. It must be reminded that neural network approximators, like MLP and RBFN, work by fitting a combination of sigmoid-like or Gaussian functions to the data to be approximated. Let us consider for simplicity that these combinations are linear. Figure 16 shows that the simple sum of two sigmoids looks like a Gaussian (weights, i.e. multiplying coefficients of the sigmoid in the sum, and thresholds have been chosen

appropriately). In most situations when using a MLP, a phenomenon similar to this one will happen. Multiplying coefficients will adjust so that each region of the space is approximated by a weighted sum of a few basis functions (sigmoids). Naturally, since a sigmoid spans the whole space, the sigmoids used in the approximation in a region of the space will influence the approximation function in other regions. Nevertheless, another sigmoid easily cancels this influence, and so on.

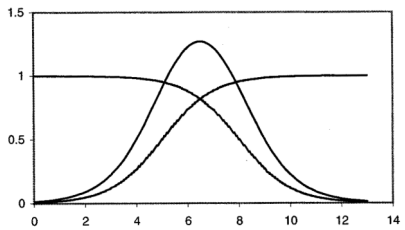

Figure 16: a weighted sum of sigmoids looks like a Gaussian function.

This phenomenon is common in MLPs, and is easily observed in small dimension with a limited number of sigmoids. The verification of this argument in large dimension and in a wide range of applications is a topic for further work.

The third argument is in the same range as the second one. It seems "natural" to think that approximation by a sum of local functions will lead to local functions having approximately the same width (standard deviation in case of Gaussian functions), and positive weights (multiplying factors); let us just imagine how a set of Gaussian functions could be combined to fill a uniform distribution in a compact region of the space. However, except in some specific situations, it can be shown that this is not the case in practical applications. In particular, we can see that weights are often set to negative values, but also that widths are sometimes set to very large values (compared with the pairwise distance between adjacent kernels), making the contribution of this kernel to the approximation function very flat. Again, this argument reinforces the idea that local and global basis functions are not much different when they are combined to contribute to an approximation function.

In the following, we will use the terms "local learning" for learning methods, or algorithms, using local functions, contrary to global ones like sigmoids. The spirit of these methods is to learn locally, i.e. to restrain the local function to a small part of the space. However, it must be kept in mind that even local methods may contribute to global approximations, as shown above: our intuitive view of the approximation of a global function by a smooth sum of positively weighted kernels with similar widths may not be verified in most situations of standard complexity.

There exist many types of local learning methods. It is impossible (and out of the

scope of this text) to give an overview of local learning methods: the diversity of goals (classification, approximation, dimension reduction, etc.) opens the way to a vast number of methods, algorithms... and publications on the topic! In the following, we will restrict ourselves to our own contributions to the vector quantization (for classification), RBFN networks (for approximation) and probability density estimations topics. Making the methods sound and effective in high-dimensional spaces is the key common theme of this chapter.

The following of this chapter is organized as follows. Section 3.2 presents conventional methods of vector quantization (VQ), used for classification tasks, and some comments about their use in high-dimensional spaces; VQ methods form the background of the two next sections. Section 3.3 presents original developments concerning the use of local probability density estimation, in order to build Bayesian classifiers. The probability density estimators are based on vector quantization algorithms, making their use realistic in practical applications. Section 3.4 presents original developments about Radial-Basis Function Networks (RBFN), which are local learning methods for function approximation.

3.2 Classification by vector quantization

3.2.1 Principle of vector quantization

3.2.1.1 Definition of the problem

Vector quantization is a technique aimed at representing a multi-dimensional distribution by a finite number of vectors. Shannon's theory shows that vector quantization always outperforms a scalar quantization on each feature [Gersho 92].

Let us define the vector quantization problems as follows. The continuous distribution X of points in a d-dimensional space is known through a finite set of random samples $\{x_i, i = 1...N\}$. We would like to quantize the distribution by selecting another set of vectors $\{y_j, j = 1...P\}$ which are *representative* of the original distribution X. Representative means here that when a sample x_i from the distribution is quantized (projected) to one of the y_j vectors, a predefined error function (for example the mean distance between x_i and y_j) is minimized. A vector quantizer will thus be defined by:

- a set of d-dimensional so-called *centroids* $\{y_j, j = 1...P\}$
- a quantization function $q(x_i)$ applying each sample x_i of the distribution to one of the centroids y_j.

Building a vector quantizer means to choose the set of centroids (the *codebook*) and the quantization function $q(x_i)$. The latter is often chosen as the minimum distance function defined as:

$$q(x_i) = y_j \quad \text{if } \text{dist}(x_i, y_j) < \text{dist}(x_i, y_k), \quad 1 \le k \le P \setminus \{j\}, \tag{6}$$

where dist() is some predefined distance measure (Euclidean, Mahalanobis, etc.).

3.2.1.2 Batch method

The conventional method of choosing a codebook is to iterate the following computations until convergence (after an initial codebook has been chosen):

- once a codebook is known, cut the data space into the Voronoi regions associated to each of the centroids y_j (the Voronoi regions are the regions of the space that are closer, according to the selected distance measure, to y_i than to any other centroid);
- choose the new location of the centroid y_j as the centre of gravity of the subset of points x_i belonging to the associated Voronoi region as computed in the first step (choosing the centre of gravity minimizes the mean quadratic error due to the quantization).

Iteration of these two steps converges to a solution. Nevertheless, the quality of the quantizer, defined as the mean value of the projection error computed on all samples x_i, will strongly depend on the codebook chosen initially. Many methods to initialize the codebook in the best possible way have been published in the literature. Several names are given to this batch method, the most known ones being the LBG (Linde, Buzo and Gray) algorithm and the Forgy algorithm.

3.2.1.3 On-line (stochastic) method

An alternative method to perform vector quantization is to use an adaptive (on-line, or stochastic) method, known as *(simple) competitive learning* (SCL) (see for example [Hertz 91]). SCL consists in adjusting the position of one of the centroids after each presentation of a sample x_i from the distribution. The cendroid y_j closest to the sample is moved according to

$$y_j \leftarrow y_j + \alpha(x_i - y_j) \tag{7}$$

where α is the adaptation constant (between 0 and 1) which must satisfy the Robbins-Monro [Robbins 51] conditions in order to guarantee convergence. Note

that mean convergence only is guaranteed, and that the number of points x_i must be finite (which is the case in data analysis).

There is no guarantee that SCL will converge to the same solution as the LBG algorithm: it will depend on initial conditions because of local minima in the error function. For the same reason, none of the two methods can be proven to perform better than the other one in all situations. Most experiments however show that the on-line version is less sensitive to initial conditions and escapes more easily from local minima.

SCL is an advantageous substitute to LBG when an adaptive algorithm is preferable, i.e. when the distribution of data is moving, or when the number of samples in the distribution is too high to consider a global method as LBG (where all samples must be considered at each iteration).

It must be mentioned that an intermediate version of the algorithm is also possible, where one centroid is moved at each presentation of a new sample: the centroid location is computed as the new centre of gravity of the Voronoi region including the new sample.

3.2.2 Vector quantization as supervised classification tool

Our concern here is to use vector quantization for classification problems. Classification means to attribute a label to each sample, after having learned which labels are attributed to known samples. Two methods, LVQ1 and LVQ2 (see for example [Kohonen 95]), are commonly used to solve classification tasks with vector quantization algorithms (LVQ holds for Learning Vector Quantization).

The principle of these algorithms is, as in any classification method, to associate a scalar class label to each sample x_i and to each cendroid y_j; this class label will be noted c_k ($1 \leq k \leq K$), so that each sample now consists in a so-called *input-output pair* $\{x_i, c_k\}$.

Classification through vector quantization means to build a codebook in each class k from the samples in the corresponding class. The number of centroids in each of these codebooks is usually chosen according to the proportion of the number of samples in the corresponding class. All centroids are then grouped in a single set, and the space is divided into Voronoi regions according to this set. Finally, a class is attributed to each Voronoi region according to the class of the associated centroid.

Classification by means of vector quantization is attractive because of its simplicity. The principle of LVQ1 [Kohonen 95] is to select the nearest centroid at each presentation of a sample from the distribution. If the sample and the centroid

have the same class, the centroid is moved in the direction of the sample according to equation (7). If they are from different classes, the centroid is moved in the opposite direction of the sample, using equation (7) where the plus sign is replaced by a minus one.

Intuitively, LVQ1 will try to quantize each region of the space where samples of one specific class are found by centroids of the same class. During the course of the algorithm iterations, if a centroid is located in a region where most samples belong to other classes, it will be moved away from this region. LVQ1 is thus a vector quantization algorithm (with separate codebooks for each class), one of its by-products being a classification of the space.

However, let us have a look to the simple classification problem in Figure 17. In this example, the distribution from class 1 is uniform above the horizontal axis and null beyond, while the distribution from class 2 is uniform (same density as class 1) below the horizontal axis and null above. After a sufficient number of iterations of the LVQ1 algorithm, centroids of the two classes (triangles for class 1 and bullets for class 2) are placed more or less uniformly above and below the horizontal axis respectively. Quantization of the two classes is thus successful. Nevertheless, the broken solid line shown in the figure represents the set of boundaries between Voronoi regions associated to centroids from different classes. This line is thus the boundary between the two classes, i.e. the solution of the classification problem. It is easily seen that only a few centroids participate to the definition of this boundary.

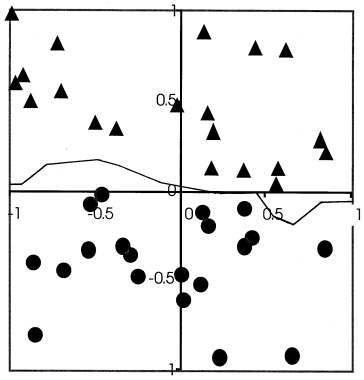

Figure 17: LVQ1 learning performed on two non-overlapping uniform distributions (respectively above and below the horizontal axis). Only a few centroids participate to the classification boundary.

Other algorithms were designed in order to make this boundary better approximate the Bayes boundary between classes (the Bayes boundary is the optimal separation when the number of misclassifications is taken as the error to be minimized). LVQ2 [Kohonen 95] is one of these algorithms. With respect to the

more classical LVQ1, LVQ2 has the drawback that the convergence is extremely slow. Furthermore, it has the same drawbacks as LVQ1 what concerns the following.

3.2.3 Improvements on vector quantization and related issues

3.2.3.1 Vector quantization for high-dimensional data

While the fact that the proportion of centroids participating to the boundary between classes is low, is easy to understand, its importance is usually underestimated, especially in high-dimensional spaces. Figure 17 illustrates that the proportion of centroids participating to the boundary in a generic classification problem is low. Moreover, recalling our previous results about the exponentially increasing number of points necessary for learning in high-dimensional spaces, we may intuitively expect that the number of centroids necessary for the definition of a boundary will also increase exponentially with the space dimension. These two contradictory arguments prove sufficiently that vector quantization is not a powerful way to solve classification problems in high-dimensional spaces. Another view of the same problem is that one easily understands that the dimension of the border is d-1 for a d-dimensional task; building a border thus means to "fill" a space of dimension d-1, confirming the exponential increase in the number of necessary centroids.

Despite this fact, vector quantization is often used, even in high dimensional spaces, to perform classification (see for example [Gonzales 97]). For this reason, we tried to improve the performances of vector quantization algorithms used in classification first by concentrating the centroids near the boundaries [Verleysen 93-2], and secondly by improving the definition of the boundary between regions associated to centroids from different classes [Verleysen 93]; the latter consists in choosing a better separation than the Voronoi boundary between centroids, making some hypotheses on the unknown distribution of samples. Because of the above-described limitations of the whole approach, these improvements are not detailed here; the reader is invited to consult references [Verleysen 93] and [Verleysen 93-2] for more details.

A better estimation of the unknown probability densities is the key idea behind this work, the reason being that Bayes boudaries can be estimated once the probability densities of each class are known. The same idea will be used in RBFN networks in a further section.

The vector quantization principles have also been applied to the RCE ("Restricted Coulomb Energy") algorithm. RCE [Reilly 82] is a simplistic learning algorithm for classification problems, which uses the following scheme. RCE uses vectors

assigned to each of the classes. Each of these vectors (we will call them centroids by analogy to vector quantization) is the centre of an hypersphere, assumed to contain samples from the corresponding class only. When a new sample is presented to the method, it is verified if it belongs to one or several of the already existing hyperspheres. If it belongs only to hypersphere(s) of the correct class, the sample is correctly classified. If it belongs to hypersphere(s) of a wrong class, their radius is decreased sufficiently so that they do not contain the sample anymore. If it does not belong to at least one hypersphere of the correct class, a new hypershpere is created centred on the sample, with a predefined radius. Iteration of the RCE algorithm on several presentations of the whole set of samples leads to convergence, i.e. to 100% correct classification of the dataset. Unfortunately, the RCE algorithm has several drawbacks, which prevent it from a successful use in high-dimensional spaces. First, the number of centroids that must be created will again increase exponentially with the dimension of the space. Secondly, the final classifier is highly dependent on the order of presentation of the samples. Last, but not least, the RCE algorithm will lead to very small hyperspheres in the regions where the distributions from several classes overlap, leading to poor generalization on samples not contained in the training set. The problem is that these regions are precisely those where it is difficult (and thus interesting) to know the optimal boundary! Despite its strong shortcomings, the RCE algorithm is used in commercial packages implementing "neural networks". Some scepticism about the whole ANN field probably comes from the blind use of such packages, without information about their limitations!

The RCE algorithm has been improved [Blayo 92, Verleysen 92] by moving the centroids at each iteration according to the SCL and LVQ principles. Again, a better approximation of the probability densities is the underlying objective. This improvement diminishes the shortcomings of the RCE method. But as in the previous situation, it must be reminded that average-performance algorithms lead to poor solutions when used in high-dimensional spaces, where the problems related to the dimension itself are added to the limitations of the methods.

A more interesting use of the density estimation principle for classification will be illustrated in the following section (IRVQ method). The principle of this method is to go straight to the estimation of the densities in each class, and then to apply the Bayes rule in order to find the most probable class at each location in the space. IRVQ combines the theoretical soundness of kernel estimators with the advantages of vector quantization in terms of number of computations and smoothness of the classification boundary.

3.2.3.2 Density of centroids

A last comment concerning vector quantization concerns the underlying probability density of centroids. When using a vector quantization method, samples x_i are assumed to be randomly drawn from an unknown distribution f. The real objective of vector quantization is to quantize the distribution f rather than the samples.

Nevertheless, since the distribution is unknown, the available information, i.e. the finite set of samples, is used.

This question is of particular importance when using vector quantization as a way to locate local functions in the space, as for example in the probability density estimation described in the next section. Indeed the real objective is to locate local kernels (as Gaussian ones) in the space, according to the probability desnity of the samples (more local kernels in regions where there are more samples, in order to reach better approximation levels). As vector quantization is used, one may expect that the VQ process will lead to a distribution of centroids in accordance (i.e. equal) to the distribution of samples. The following paragraphs are intended to comment this question.

Centroids y_j $(1 \leq j \leq P\}$ after quantization are also vectors. While they are not random (they are the result of a deterministic process), one could be interested in knowing from which distribution we could have randomly selected P samples and obtained a similar set of vectors. We will call this distribution the underlying distribution of centroids g.

When using vector quantization in applications, as in image compression for example, it is implicitly assumed that distributions f and g are equal. This also corresponds to our intuitive view of the fact that the set of centroids is the best set (of size P) representing the set of samples.

However many authors ([Gersho 79], [Ritter 86], [Kohonen 98], [Fort 00], [Graf 00]) give arguments that show that the vector quantization which leads to a minimization of the quantization error corresponds to a discrete distribution which converges asymptotically (when the number of centroids goes to infinity) to a distribution with density :

$$g(x) = kf(x)^{\alpha} \tag{8}$$

where k is a constant and $\alpha = d/(d+2)$. α is called the *magnification factor* and is equal to 1/3 in the one-dimensional case.

Many of these arguments find their origin in a publication by Zador [82] which has been badly understood by many authors, but recent work seems to prove that equation (8) is valid anyway. In [de Bodt 99] – annex A, we developed experimental arguments in dimension 1 in favour of this thesis.

It must be mentioned however that when the goal is to find a set of centroids verifying $f(x) = g(x)$, weighting the centroids by the probabilities of finding a sample in their associated Voronoi regions solves the problem [Pagès 97, de Bodt 99]. Moreover, in our context of data analysis in high-dimensional spaces, the magnification factor may be rapidly assimilated to 1 when the dimension of the

space is sufficiently large, proving that, unlike many other difficulties, this one disappears with high-dimensional problems.

3.2.4 Further topics on vector quantization

Vector quantization is a research field in itself. It concerns so many methods and so many applications that it would be impossible to summarize here what has to be done as further work. Nevertheless, we concentrate here on vector quantization to be used in further algorithms (see next sections), as a convenient way to locate the local kernels used for approximation. In that spirit, relying on very accurate positions of centroids, or on strict equality between distributions of samples and of centroids, is not so important. This is why we only sketched the concepts here, without going into further details that would be irrelevant in our context.

Nevertheless, relying on stable VQ methods that may be used in high-dimensional context is important. Our feeling is that much has still to be studied in that context. Even if ultimate performances are not looked for, current vector quantization methods are extremely sensitive to initial conditions. While this problem can be compensated by numerous iterations in small-dimensional problems, this becomes impossible in higher dimensions. As a result, the reliability of vector quantization in high-dimensional spaces is questionable. Solving this shortcoming is not an easy thing; rather, we believe that solutions should be looked for in the direction of algorithms using VQ as part of them, but for which the performances are not determining. The algorithms developed in the following of this chapter are attempts in this direction. Nevertheless, we believe that it would be worth to study more quantitatively the performances of VQ methods in high-dimensional spaces; surprisingly, a serious study about this topic does not seem to exist in the literature, at least to our knowledge.

3.3 Bayesian classification by probability density estimation

3.3.1 Principle of Bayesian classifiers

In multi-dimensional classification tasks, the challenge is to attribute a class label to a vector presented to the system, which previously "learned" the spatial distribution of each class, on a set of training vectors. The Bayesian classification theory provides an ideal method for classification of data, once the a priori probabilities of the classes and their probability density functions are known. The principle of Parzen windows [Cacoullos 66] or kernel estimators is to estimate the probability density functions with the learning vectors, and then to use these

estimates in the Bayes law.

Parzen windows however require a computational load that is unrealistic in practical situations (it requires among others the evaluation of a number of Gaussian functions equal to the number of vectors in the learning set); we present here a method to drastically reduce the number of operations involved in Bayesian classification, by using a vector quantization technique to replace the initial learning set by another one with a strongly reduced number of samples, while minimizing the approximation error on the probability density functions. This method allows considering favourably the use of kernel estimators in realistic classification tasks.

Assume the problem consists in classifying an observed vector x of R^d among K classes. Assume that x is random and that its d components admit a joint probability density function $p_x(x|c_k)$ in class c_k. If all wrong decisions are given the same penalty, the Bayes law may be expressed as:

$$P(c_k|x) = \frac{p_x(x|c_k)P(c_k)}{\sum_{l=1}^{K} p_x(x|c_l)P(c_l)} \tag{9}$$

where $P(c_k)$ is the a priori probability of class c_k and $P(c_k|x)$ the probability that vector x belongs to class c_k. The Bayesian decision to select the most probable class is then:

$$\text{decide } x \in c_k \Leftrightarrow k = \text{Arg}\max_{i \le l \le K}\{p_x(x|c_l)P(c_l)\}. \tag{10}$$

Using equation (9) necessitates the knowledge of the a priori probabilities $P(c_k)$ of the classes and of the class-dependent probability densities $p_x(x|c_k)$. These values and functions are never known; the only information we have at our disposal is the set of samples $\{x_i, i = 1...N\}$ and their associated classes $\{c_i, i = 1...N\}$.

An unbiased estimation of the a priori probabilities $P(c_k)$ of the classes is the ratio between the number of samples belonging to class i and the total number N of samples.

A kernel density estimator is a consistent estimate of a multivariate probability density function [Cacoullos 66, Comon 95]. Using such estimator, the probability density in each class c_k can be estimated by

$$\hat{p}(x|c_k) = \frac{1}{N_k} \sum_{c(x_i)=c_k} K\left(\frac{x - x_i}{h_i}\right) \tag{11}$$

where the sum over i is taken for each sample belonging to class c_k, and N_k is the number of terms in this sum i.e. the number of samples in class k.. The parameter h_i is the *width factor* of the *kernel* K. Note that, according to the usage in the literature, the notation K is used here to define the kernel functions, even if the same letter K is used to define the number of classes in a classification problem.

The estimator is said to be *variable* if h depends of i and *fixed* otherwise. Variable estimators provide better estimates, but it is very difficult to compute the optimal value of h_i.

Because of their nice properties (smoothness, continuity, etc.) Gaussian kernels are often used:

$$K\left(\frac{x - x_i}{h_i}\right) = \frac{1}{\left(h_i\sqrt{2\pi}\right)^d} \exp\left(-\frac{1}{2}\left(\frac{\|x - x_i\|}{h_i}\right)^2\right). \tag{12}$$

Using equation (10) with the estimate provided by (11) leads to a classifier which requires an extremely light computational cost during learning (a simple storage of the training patterns) and very good performances. Unfortunately, if large training sets are available, the required memory size and the computational cost of the classification become incompatible with real time classification tasks. Another drawback seldom mentioned in the literature is the fact that the density estimator (11) cannot be considered as smooth in case of a restricted number of samples in the training set: the estimate will be accurate near the samples, but less and less accurate at locations of the space further from the samples. Since equation (10) precisely requires the accurate estimation of the probability densities at other locations than those of the learning samples, performances can drastically decrease when the size of the learning set is not "sufficient".

We do not intend to go here into details on the choice of the width factors $h(i)$. We must mention that theoretical results on those factors are only available in asymptotic (unrealistic) cases, while experimentation and ad-hoc a priori choices (using knowledge on the data) is the rule in practical situations. However, it can be shown experimentally that accurate values are not mandatory when the number of samples x_i is high enough; nevertheless, the necessity for adequate values grows as this number descreases.

The purpose of the suboptimal Bayesian classifier presented below is to drastically reduce the number N_k of kernels in each class, in order to use equation (11) in realistic situations, avoiding to reduce the quality of the density estimation approximation.

3.3.2 Improved practicability by vector quantization: IRVQ algorithm

Kernel density estimators using equation (11) are difficult to use for two reasons. First, they imply the use of as many kernels as there are samples, which can be computationally too intensive. Secondly, non-asymptotic laws to adjust the size of the local kernels are difficult to find. In the following, we develop an original method to improve the practicability of kernel density estimation, through the use of vector quantization on the samples.

The principle of the proposed method is to split the portion of the space where vectors can be found in clusters. A vector quantization technique will be used to find the clusters and their centres of gravity, and it will be assumed that the error generated by the vector quantization will be sufficiently small so that the true probability density inside each cluster can be approximated by a constant. In the portions of the space where the vector quantization will lead to small clusters, this last assumption will be verified. On the other side, in the portions of the space where the clusters are large, this means that the number of learning vectors which lead to these clusters is small, and hence that an error in the approximation of the density function is of less importance.

Other algorithms exist to reduce the size of the learning set before using equation (11). The first one [Fukunaga 89] extracts a reduced set from the original one in an optimal way to reduce the differences between the probability density estimate before and after this reduction; this method is however heavy from a computational point-of-view, and leads to unsatisfactory results for high reduction rates [Xie93]. Another algorithm [Comon92] uses a vector quantization technique to reduce the size of the learning set, as does our algorithm, but is based on a Gaussian hypothesis of distribution inside each cluster, instead of a constant one for ours; the Gaussian hypothesis is more appropriate when the vector quantization leads to clusters which represent the modes of the distribution, which is the case when the number of clusters is much smaller than in our hypotheses.

3.3.2.1 Vector quantization

Reducing the size of the learning set means substituting the training set by a reduced set, keeping as much information as possible. This is precisely the aim of vector quantization techniques, as for example the SCL method described above. In this context however, we have to keep in mind that a class attribute is associated to each sample in the learning set. Instead of performing vector quantization on the whole learning set, we will perform it separately on each subset of the initial learning set corresponding to samples belonging to a specific class. In other terms, suppose that the initial learning set $A = \{x_i, i = 1...N\}$ can be divided into subsets $A_k = \{x_i \mid c(x_i) = c_k, i \in \{1...N\}\}$. We will perform vector quantization separately on each subset A_k to obtain sets of centroids denoted by

B_k. In order to keep the information on the a priori probabilities of the classes, the number of centroids in B_k is chosen to be proportional to the number of samples in A_k, the total number of centroids being a parameter of the model. SCL (equation 7) is used here as vector quantizer, but any other vector quantization method could be used.

The purpose of this vector quantization in our problem of estimation of probability densities for Bayesian classification is to use the reduced sets B_k instead of the original sets A_k for the estimations of the probability densities (11). Sets B_k are representative of the original sets A_k when the dimension of the space is high, as detailed in the section about the density of centroids. As mentioned above, the purpose of this is not only to decrease the computational load of the method, but also to increase the smoothness of the approximator. The smoothness is controlled by a parameter: the total number of centroids. There is however no direct way to choose this number of centroids according to a desired smoothness; the only way is to measure this smoothness a posteriori.

Once the sets of centroids have been designed, the question on how to choose the width factors remains. It was mentioned above that accurate values for the width factors were not mandatory if the number of samples x_l was large enough. In the current situation however, the number of samples is replaced by the number of centroids, of course significantly lower. An adequate method to estimate the width of the Gaussian functions centred on the centroids is thus required. Figure 18 [Specht 90] shows the approximation of a probability density through Gaussian kernels, with various widths. It is clear that choosing a too small width leads to overfitting (left figure), while choosing a too large width leads to oversmoothing.

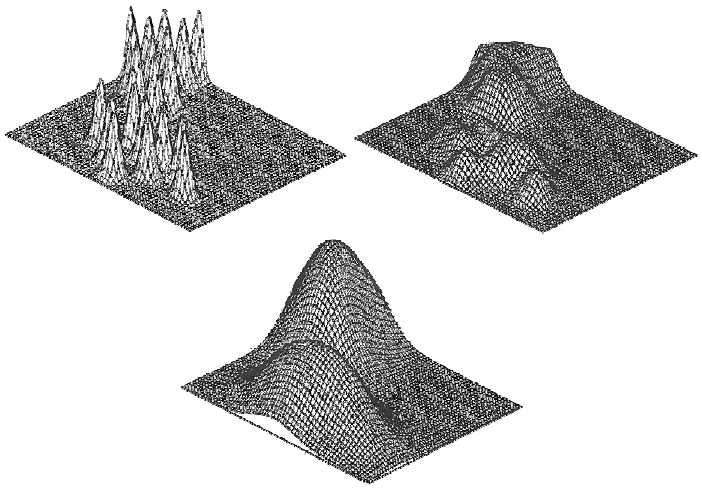

Figure 18: approximation of probability densities through kernels with different widths. From [Specht 90].

3.3.2.2 Width factors

During the adaptation process (7), it is possible to keep track of the mean distance between a pattern x_i and its closest centroid y_j, by affixing an *inertia* coefficient i_j to each centroid. This inertia coefficient is randomly initialized to a small value and then adapted at the same time as the centroid locations (7) according to

$$i_j \leftarrow i_j + \alpha\left(\left\|x_i - y_j\right\|^2 - i_j\right). \tag{13}$$

This equation means that the inertia coefficient i_j is adapted at each presentation of a pattern x_i to a convex combination between the actual value of i_j and the norm of the distance between x_i and the closest centroid y_j. After learning, parameters i_j will converge to the inertia of points in the clusters associated to y_j. Note that convergence can be easily proven if α remains constant. We will use this inertia coefficient i_j for the computation of the optimal width factor associated to the kernel centreed on y_j in the reduced classifier.

Let us make now the main hypothesis of this estimation of width factors: we consider that the size of the clusters is small enough to approximate the true density $p_x(x|c_k)$ in any class k by a constant over two consecutive clusters. Clusters are here defined as the sets of points nearest to their associated centroid y_j than from any other centroid y_l ($1 \le l \le K$). Consecutive clusters are clusters sharing a common border. This assumption is of course only an approximation, but is not too restrictive when the true densities are smooth.

The purpose will then be to fix the width factors in order to have a constant estimate of the probability densities over two consecutive clusters too. Let us first examine the problem in dimension 1, with an estimate computed by the sum of two kernels A and B (see Figure 19). In the following, we will omit indices over classes and class-dependent notations (on probability densities). Indeed vector quantization to obtain the sets of centroids B_k instead of the original sets A_k is performed separately for each class. The probability density is thus also estimated separately for each class.

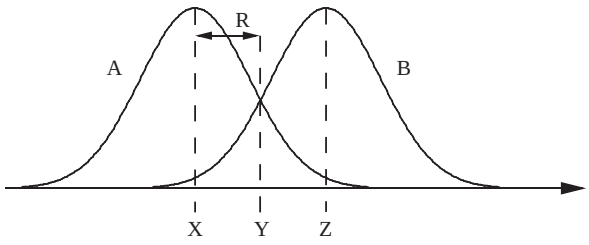

Figure 19: sum of two Gaussian functions in dimension 1, and related notations.

Using equation (12) for both kernels, the estimate of the probability density at location X is given by

$$\hat{p}(X) = \frac{1}{h_j \sqrt{2\pi}} \tag{14}$$

if the contribution of kernel B is neglected at X (this is a second hypothesis aimed to reduce the complexity of the computations). In the same way, the estimate of the probability density at location Y is given by

$$\hat{p}(Y) = \frac{2}{h_j \sqrt{2\pi}} \exp\left(\frac{-R^2}{2{h_j}^2}\right) \tag{15}$$

where R is the distance between X and Y ($2R$ is the distance between two consecutive centroids). Width factors h_j are assumed to be equal for this computation. Making the estimates (14) and (15) of probability density at points X, Y and Z to be equal leads then to

$$R = \sqrt{2\ln 2} h_j \,. \tag{16}$$

A similar development may be done in dimension 2. In this case, we can consider the approximation of probability density due to four Gaussian functions A, B, C and D centreed on the four vertices of a square:

- at the location X of a vertex of the square,
- at the centre Y of the square, and
- at the midpoint Z of an edge of the square.

Keeping the distance between two centroids on an edge of the square being equal to $2R$, and neglecting the influence of kernels at a distance greater or equal to $2R$, we have respectively

$$\hat{p}(X) = \frac{1}{\left(h_j \sqrt{2\pi}\right)^2} \,, \tag{17}$$

$$\hat{p}(Y) = \frac{4}{\left(h_j \sqrt{2\pi}\right)^2} \exp\left(\frac{-R^2}{{h_j}^2}\right), \text{ and} \tag{18}$$

$$\hat{p}(Z) = \frac{2}{\left(h_j \sqrt{2\pi}\right)^2} \exp\left(\frac{-R^2}{2{h_j}^2}\right). \tag{19}$$

One can make the estimations (17), (18) and (19) to be equal, by setting the width factor h_j according to equation (16), which gives thus the same result as in dimension 1. An identical development can be made in dimension 3, by considering the influence of 8 Gaussian kernels located on the vertices of a cube, respectively at the locations of these vertices, of the midpoint of any edge, of the centre of a face, and of the centre of the cube. Again, the estimations of probability densities will be equal if equation (16) is satisfied; it will also be the case in dimension d greater than 3.

The last step is now to set the relation between the estimated inertia in each cluster i_j and the width of the cluster h_j. For this purpose, we make the supplementary hypothesis that, within a short volume in the d-dimensional space, all centroids are equally spaced on a regular isotropic grid; this hypothesis is similar to the one of a constant true density over consecutive clusters, since a vector quantization applied to a constant distribution will lead to centroids on such a regular isotropic grid. We can thus consider that a cluster associated to a particular centroid will be a d-dimensional hypercube with edges of length $2R$, denoted by V. The inertia in such a cluster is

$$i_j = \frac{1}{(2R)^d} \int_V \left\| x_i - y_j \right\|^2 dV = \frac{dR^2}{3}. \tag{20}$$

Note that the hypercube assumption is not exactly true. Vector quantization in a 2-dimensional space with a constant density will lead to hexagonal clusters and not to square ones. The exact solution of this problem in dimension greater than 2 is not known, but it has been verified experimentally that the error introduced by this assumption is moderate in high dimension.

Combining equations (16) and (20) leads to a width factor h_j given, in dimension d, by

$$h_j = \sqrt{\frac{3i_j}{2d\ln 2}}. \tag{21}$$

Finally, the estimation of probability density in each class will be calculated through equation (11), applied on a set of centroids fixed by (7), and the width of the kernels being fixed by (21). Bayesian classification is then carried out based on relation (10), where the probability densities are replaced by the above estimates, and the a priori probabilities by percentage of occurrence of prototypes x_i in each class. This constitutes the IRVQ (Inertia-Rated Vector Quantization) method.

3.3.2.3 Results

For illustration purposes, the IRVQ algorithm has been applied on two artificial databases. Figure 20 shows the results of the IRVQ algorithm on these two different two-class problems, the first one with normal distributions, and the second one for uniform concentric circular distributions; centroids are represented with a circle of radius h_j.

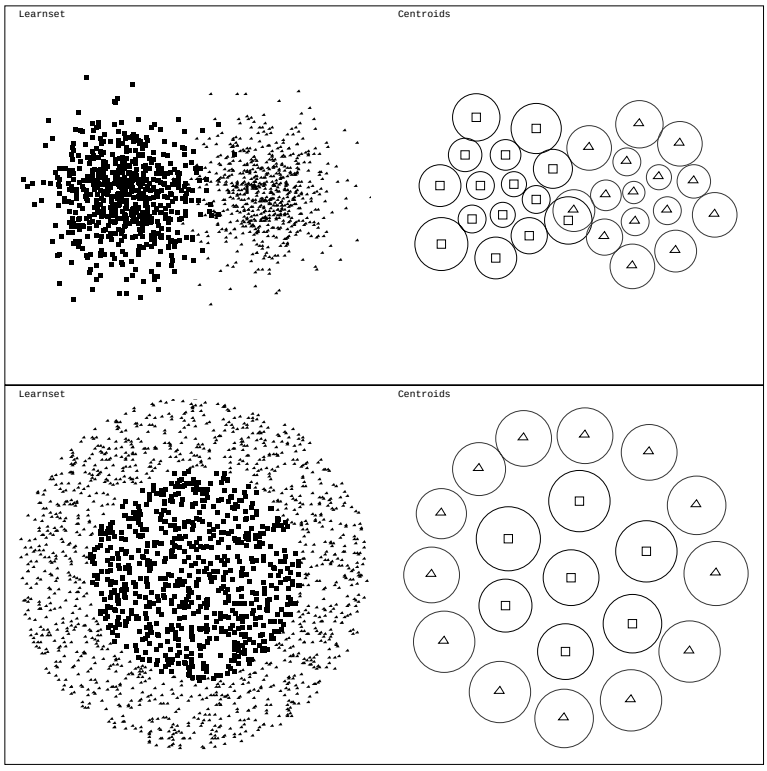

Figure 20: centroids and their associated optimal width factors obtained by the IRVQ algorithm for two different two-class 2-dimensional databases.

Results on a real database of preprocessed handwritten digits provided by ATT Bell Laboratories are presented in [Voz 94] where it can be seen that the IRVQ algorithm gives higher recognition rates than the best classifiers actually reported on this database, while the computation cost is much lower.

3.3.3 Further research topics

The IRVQ method presented here seems to give excellent results, and to be a good compromise between efficiency (percentage of correct classification) and computational cost. Nevertheless, two aspects of this algorithm have still to be examined.

First, several assumptions were made in the development of the algorithm. While these assumptions seem natural and have been verified independently, it should be checked if their joint influence has still a negligible impact on (21). Theoretical proofs would be preferred, but seem difficult to develop in dimensions greater than 2, for example because even the exact solution of the quantization of a uniform density is not known. Secondly, one could try to make the same development without the hypothesis that the density over two consecutive clusters is constant. Replacing it by a linear hypothesis seems feasible. These two aspects, together with in-depth experimentation of the method in high dimension, are a topic for further work.

Another question is related to our earlier discussion about non-intuitive extrapolation of what is obvious in dimensions 1 and 2. For example, let us recall Figure 9, showing the percentage of samples from a Gaussian distribution falling closer to the centre than a specific distance. A Gaussian function in high-dimension is thus *not* a function for which most of its integral is concentrated near its centre. This may be seen as contradictory to our arguments in the development of the IRVQ algorithm.

Part of the solution to this problem may be found in our work [Comon 94] about the "Rough-Refined Estimator" (RRE). Without going into details, the principle of this method is to consider shapes different from a Gaussian function, in high dimensions. The general shape considered in the RRE method is given by

$$K\left(\frac{x - x_i}{h(i)}\right) = B\exp\left(-\left(A\|x - x_i\|\right)^g\right). \qquad (22)$$

where g is a real number greater than 0.5 setting the rate at which the kernel function drops off, and A and B are determined so as to have a unit sum of the density. Using super-Gaussian kernels defined by equation (22) instead of conventional Gaussian kernels is advantageous in high-dimensional settings. Indeed, Figure 9 shows that Gaussian kernels are no more local functions in high-

dimensional spaces (most of the points are not concentrated anymore around the centre). Using kernels defined by equation (22) may remedy to this problem: a proper choice of coefficient *g* may compensate for the distortion due to the dimension, and thus make problems similar in small and high dimensional spaces.

The RRE method as it stands is difficult to use in high dimension, mainly for convergence reasons. It assumes a first estimation of the density, from which the width factors are derived analytically, under the assumption that the first estimate is close from the final one. When this condition is not verified, the algorithm may be hardly usable. Nevertheless, a way to explore is to combine the idea of a different kernel as given in (22) with the IRVQ algorithm. Rather than estimating a posteriori the parameters in equation (22) (using an estimate of the probability density, which is the goal of the method), it could be possible to use an a priori method, based on considerations on the shape of a (Gaussian or not) kernel in high dimension. This is also a topic for further work.

3.4 Radial Basis Function Networks for approximation

3.4.1 Principle of Radial Basis Function Networks

RBFN (Radial-Basis Function Networks) are known as local models, as opposed to global models as MLP (Multi-Layer Perceptrons). RBFN use combinations of radial (usually monotonically decreasing) functions, vanishing when the distance from their centre increases; this justifies their name.

RBFN approximate functions. They differ from classification problems where the class label is a discrete value. They also differ from the probability density approximation problem. In function approximation, the problem is to approximate in the best possible way targets, or outputs values, that are known for each input data. Learning in function approximation is known as *supervised*, since a supervisor measures in some way the quality of the approximation (usually by means of the difference between the targets and the corresponding outputs of the approximator). Let us also mention that classification problems were best solved by focusing the attention (the performances) of the model to the borders between classes, while approximation concerns the whole space, a priori without focusing on a specific location.

However RBFN resemble density approximators in their structure: the standard RBFN approximator uses linear combinations of Gaussian functions, as in the kernel density approximators. Gaussian functions are however weighted in RBFN, which is natural in case of supervised learning, while they are not weighted in density approximator, since there is no target that can be used to optimise

weights. Let us mention however that learning classification tasks is also a supervised process. For example in Bayesian classification, supervision takes place when the data are segregated according to classes before being used for the estimation of the probability density of each class; each of the density estimation is unsupervised though.

3.4.2 Linear Radial Basis Function Networks

RBFN are often mentioned in the literature as *linear* models. Let us share this view, at least for a few paragraphs... We will see in the following that RBFN can become powerful models, if they are considered as nonlinear. In fact, RBFN are linear with respect to some of their parameters, and nonlinear with respect to others. Viewing RBFN as linear or nonlinear models depends on which parameters are optimised during learning.

For the following discussion about "linear" RBFN, we will mainly follow Orr's presentation [Orr 96].

3.4.2.1 Model

A linear model for a function $y(x)$ takes the form

$$f(x) = \sum_{j=1}^{P} w_j K_j(x). \tag{23}$$

The model f is expressed as a linear combination of a set of P fixed functions (often called *basis functions* by analogy with the concept of a vector being expressed as a linear combination of basis vectors). The choice of the character w (*weights*) for the coefficients of the linear combinations reflects the commonly admitted terminology in the neural network community.

The flexibility of f, its ability to fit many different functions, derives only from the freedom to choose different values for the weights. The basis functions and any parameters that they might contain are fixed. If this is not the case, if the basis functions can change during the learning process, then the model is nonlinear.

Any set of functions can be used as the basis set although it helps, of course, if they are well behaved (differentiable). However, models containing only basis functions drawn from one particular class have a special interest. Classical statistics abounds with linear models whose basis functions are polynomials ($K_j(x) = x^j$ [Hlavackova 97], Taylor's expansions or variations on the theme). Combinations of sinusoidal waves (Fourier series),

$$K_j(x) = sin\left(\frac{2\pi j(x-\theta_j)}{m}\right) \tag{24}$$

are often used in signal processing applications. Logistic functions, of the form

$$K_j(x) = \frac{1}{1+exp(w_j x - w_{j0})} \tag{25}$$

are popular in artificial neural networks, particularly in multi-layer perceptrons (MLPs).

Linear models are simpler to analyse mathematically. In particular, if supervised learning problems are solved by least squares then it is possible to derive and solve a set of linear equations for the optimal weight values implied by the training set. The same does not apply for nonlinear models, such as MLPs, which require iterative numerical procedures for their optimisation (the parameters w_j are inside the nonlinear function, hence prohibiting a direct computation).

3.4.2.2 Radial functions

Radial functions are a special class of function. Their characteristic feature is that their response is a function of the distance from a central point (usually, a monotonically decreasing function). The centre, the distance scale, and the precise shape of the radial function are parameters of the model, all fixed if it is linear.

A typical radial function is the Gaussian function

$$K_j(x) = exp\left(-\frac{\|x-c_j\|^2}{h_j^2}\right). \tag{26}$$

Its parameters are its centre c_j and its radius h_j. Gaussians are not the only possibility to radial functions. For example, any function monotonically decreasing with the distance from a centre, as in equation (26), but with a different exponent in the exponential, could be considered. Such functions will be discussed later.

Radial functions are simply a class of functions. In principle, they could be employed in any kind of model (linear or nonlinear) and any kind of network (single-layer or multi-layer). However, since Broomhead and Lowe's seminal paper [Broomhead 88], radial basis function networks (RBFN) have traditionally been associated with radial functions in a single-layer linear network as in equation

(23), and almost exclusively with Gaussian functions.

The reason for this is that linear single-layer networks such as those described by (23) have the nice *universal approximation property* [Moody 89]. Shortly, this means that, under soft mathematical conditions on the function to approximate, RBFN networks so defined can approximate any function of any number of variables on a compact, to any desired accuracy, provided that a sufficient number of basis functions are incorporated in the model. Why thus considering networks more complicated than (23), if this model is sufficient to have this property? The answer is that the universal approximation property is of no interest in practical situations: the number of basis functions has to be limited in order to ensure good generalization, which is in contradiction to the hypothesis of the property. Once again, we see that approximation and interpolation lead to the necessity of compromises, so that practical models are not necessarily those studied at limit conditions...

A last comment about the universal approximation property is that it holds even if the width h_j of the Gaussian kernels is fixed. Nevertheless, again, it is sometimes preferable to give more degrees of freedom to the network (by making more parameters adjustable) rather than increasing its size, as we will detail below.

3.4.2.3 Optimal weights

The general problem of function approximation is defined by a learning set containing so-called *input-output pairs* that have to be approximated. Let us define the learning set as

$$L = \{(x_i, y_i), i = 1 \ldots N\}. \tag{27}$$

In the following, we will consider x_i as p-dimensional vectors, and y_i as scalars. Generalization to vector output is straightforward, but will be omitted here for simplicity. We will also restrict ourselves to linear single-layer RBFN networks as given by (23), with Gaussian kernels (26).

When interpolation is looked for, direct computation of the weights in equation (23) can be achieved in the following way. A RBFN model with P Gaussian functions is built. Centres and widths of Gaussian kernels are fixed a priori. Then, a matrix H is defined as

$$\begin{aligned} H &= \lfloor h_{ij} \rfloor \\ h_{ij} &= K_j(x_i) \end{aligned} \tag{28}$$

where $K_j(x_i)$ is defined by (26). Interpolating the input-output pairs of the training

set is then equivalent to solving the linear system

$$Hw = y \tag{29}$$

where w is the column vector whose components are the weights w_i, and y is the column vector whose components are the targets y_i. However, since the number P of Gaussian kernels is lower than the number N of samples, equation (29) cannot be solved exactly. One thus has to define a criterion in order to chose the "optimal" set of weights. Optimal means that some error will be minimized. The LMS criterion is usually chosen in this context:

$$E = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2 . \tag{30}$$

An exact solution to equation (29) cannot be found, but the solution minimizing (30) will minimize the difference between the left and right sides of formula (29). Note that the dimension of matrix H is $N \times K$, and the respective dimensions of vectors w and y are $P \times 1$ and $N \times 1$.

The solution of this problem is known as the pseudo-inverse matrix of H given by

$$H^+ = \left(H^T H\right)^{-1} H^T . \tag{31}$$

The optimal set of weights according to criterion (30) will then be

$$w = H^+ y = \left(H^T H\right)^{-1} H^T y . \tag{32}$$

Note that the universal approximation property of RBFN is valid when $K < N$, but of course still in limit conditions, i.e. when both K and N grow to infinity! Again, this property is thus not of practical interest in applications.

Literature abounds with papers dealing with linear RBFN. While the problem of finding the optimal weights once the centres and widths of Gaussian functions have been fixed seems simple (see equation 32), matrix H can be huge, making $H^T H$ tedious to invert, and the product $H^T H$ is often ill-conditioned, making its inversion even more difficult; Singular Value Decomposition (SVD) is then used to compute the pseudo-inverse of matrix H. Many papers also deal with implicit regularization, i.e. by adding constraints to the error criterion or to the derivation of weights in order to ensure some smoothness in the approximation function f. While this problem is interesting, we chose not to discuss it here. We will focus our attention to another question not often dealt with in the literature: the choice of

kernel locations and widths. Indeed it is easy to understand that a wrong choice of these parameters will lead to weak results. Figure 21 shows a simple 1-dimensional example making this comment obvious. The function to learn is known through 200 samples. The first 100 ones form a sine wave, while the last 100 ones form a horizontal line. The density of points along the *x*-axis is constant over the first 100 samples and constant over the last 100 ones, but the first density is twice the last.

Without a procedure to choose adequate locations for the basis functions, one could distribute them equally along the *x*-axis. As an example, the distribution of seven Gaussian basis functions leads to the centre locations marked by bullets on the *x*-axis in Figure 21. Obviously, the approximation of the sine function will be hard with only three to four basis functions in the corresponding regions. The remaining basis functions could contribute if their width is chosen sufficiently large, but the advantages of local approximation are then lost. A better choice of locations for the basis function centres would have been to distribute more centroids in the left region (sine wave), and less in the right region, easier to learn.

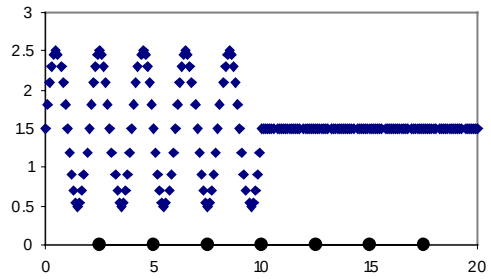

Figure 21: approximation of a 1-dimensional function by a RBFN network with seven Gaussian functions equally spaced on the *x*-axis. Dots on the *x*-axis show possible centroids.

Two ideas have been purposely mixed in this example. Locations of basis functions could follow the density of samples (along the *x*-axis), and could also be influenced by the complexity of the function to approximate. Vector quantization is the solution to the first idea, while the last remains a topics for further study.

In the following, we will address the (standard) problem of choosing the basis function centres, but also the problem of choosing correct width factors for the basis functions. This last question is certainly as important as the first one, although it is much less studied in the literature.

3.4.3 Optimization of centres and widths

Nonlinear RBFN are models where the dependency of the error term on *some of* the parameters that are adjusted during learning is nonlinear. In the context of the model (23) with kernels (26), this means that the derivative of the error term (30) with respect to these parameters still depend on the parameters themselves; these parameters are the centres of the Gaussian functions, and their widths.

Several algorithms and heuristics have been proposed to evaluate these parameters. Most concern the centre locations: the choice of widths is seldom addressed. In the widely used algorithm by Moody and Darken [Moody 89], locations of centroids, widths and weights are determined sequentially, without feedback between the computation of these three groups of parameters. Locations of centroids are first fixed by an estimation of the probability density of the input samples (vector quantization); the widths are then calculated to ensure a predefined overlap between Radial Basis Function Networks centred on these centroids, and finally weights are computed according to criterion (30), leading to (32). Unfortunately, once the locations of the centroids are fixed, they are not allowed to be ajusted anymore, in order to globally minimize the error function (30). The same comment applies to widths. Only weights are thus optimal in the sense of criterion (30).

The following describes an original method to compute efficiently the centres and widths parameters in a RBFN network similar to Moody's and Darken's one. In this method however, width factors are determined with respect to the standard deviation inside the "region of influence" (Voronoi region) of a centroid, rather than fixing an a priori overlap between functions. Furthermore, values found for the centroid locations, width factors and weights are adjusted in a subsequent phase: all parameters are optimized according to gradient descent on criterion (30). Values of parameters found in the first phase of the method may thus be seen as (good) initial conditions for a standard global gradient descent-based optimisation. This second phase looses the advantages of splitted learning, but usually requires only a few iterations of global optimization, which reinforces the computational advantages of RBFN over MLP-like networks. Note that a direct gradient descent on all parameters after random initialization is usually not used, because it suffers from the same drawbacks as MLP: greater risk of local minima, flat regions in the error function leading to unefficient gradient descent, need for complex optimization algorithms, etc.; RBFN would loose all their advantages with direct gradient descent.

3.4.3.1 Location of centroids

Locations of centroids are usually chosen according to the density of the input set x_i; such a choice leads to more centroids, and so naturally to a better approximation of targets y_i, in regions of the input space covered by more input

vectors, which seems a good heuristic in many applications.

Moody and Darken [Moody 89] proposed to use a k-means clustering algorithm to find the locations of the centroids c_j. As detailed in [Verleysen 96] and [Verleysen 94], we suggest to use a Simple Competitive Learning (SCL) method which leads to similar results, with the advantages first of being adaptive (continuous learning, even with an evolving input database), and secondly of helping the choice of the width factors as explained in the next section. The principle is thus to initialize the P centroids c_j to the first (or to random) P input patterns x_i, $1 \le i \le P$. Then, input vectors x_i, $1 \le i \le N$, are sequentially or randomly presented. The centroid c_j closest to x_i is selected, and moved according to the SCL principle

$$c_j \leftarrow c_j + \alpha(x_i - c_j) . \tag{33}$$

After convergence of this SCL procedure, the density of the centroids will approximate the density of the input data (at least if the magnification factor as described in the section about vector quantization is taken into account). We remind that mean convergence is guaranteed in the case of a finite databse, and if α satisfies the Robbins-Monro conditions.

It is important to note that the SCL procedure (or any other vector quantization on the input data) will lead to position of centroids selected according to the probability density of input data x_i, but that targets y_i are not taken into account here. Therefore, some of the limitations mentioned in the context of the example (Figure 21) still apply.

3.4.3.2 Width factors

In order to evaluate the width factors h_j of the Gaussian functions, Moody and Darken [89] proposed to minimize a cost function measuring the overlap between adjacent units:

$$E_{ov}(h_1,\ldots h_K) = \sum_{j=1}^{K}\left[\sum_{k=1}^{K} exp\left(-\left(\frac{\|c_j - c_k\|}{h_j}\right)^2\right)\left(\frac{\|c_j - c_k\|}{h_j}\right)^2 - Q\right]^2 . \tag{34}$$

This function however includes a parameter Q difficult to choose, what effectively transfers the problem of choosing the h_j to the choice of this parameter. The problem is simplified, since there is only one remaining parameter, but not solved.

In [Verleysen 94] we proposed to use a measure of the dispersion of points in the clusters associated to the centroids in order to fix the width factors h_j. A slightly

more general method is described in [Verleysen 96]. The principle is to estimate iteratively the width factors through a convex combination between the previous estimation and a new value according to

$$h_j \leftarrow (1-\beta)h_j + \beta q \left\| x_i - c_j \right\| . \tag{35}$$

Using this formula repetitively on all samples x_i leads to h_j converging to the mean value of distance between any sample x_i in a Voronoi region and its centre c_j, multiplied by a constant q (provided that β remains constant).

One could argue that equation (35) still contains a parameter β that has to be adjusted. Nevertheless, a non-optimal choice of β would only influence the speed of the convergence, while a bad choice for Q in Moody and Darken's method directly influences the h_j values.

If q is set to 1, h_j will converge to the standard deviation of the cluster j. The role of $q \neq 1$ is explained below, under hypotheses of increasing complexity.

First hypothesis: constant density, constant function

We first consider that the true probability density of input vectors x_i can be assimilated to a constant over two consecutive clusters, and also that the function f to be approximated can itself be assimilated to a constant in first approximation on the same range; this hypothesis leads to the constraint that we will choose the width factor of the Gaussian function associated to each cluster in order to keep the estimate (23) of the function as constant as possible over two consecutive clusters. Let us use the same notations as in Figure 19, repeated here for convenience. The discussion below is similar to the discussion about the widths of kernels in the IRVQ procedure for probability density estimation, although the knowledge of targets y_i gives additional information that will be taken into account.

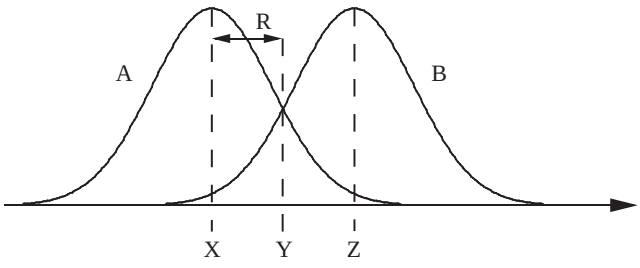

Figure 22: sum of two Gaussian functions in dimension 1, and related notations.

X and Z represent the centres of two consecutive clusters A and B, $2R$ the

distance and Y the midpoint between them. The purpose of the method is to set the relation between R and the width factor h of the Gaussian functions A and B, in order to have a constant approximate of the probability density over the segment $[X, Z]$. We will simplify the computation of h_j by setting its value in order to have the same estimate of probability density at points X, Y and Z; we assume that the fluctuations inside the segments $[X, Y]$ and $[Y, Z]$ may be neglected. We will also neglect the influence of a kernel at a distance $2R$ of its centre (which, when verified a posteriori, will cause a maximum error of about 6 % in the local value of the estimate).

With these hypotheses, we can evaluate the contribution of the Gaussian functions A and B respectively at locations X (or Z) and Y:

$$f(X) = w\,, \tag{36}$$

$$f(Y) = 2w \exp\left(-\frac{R^2}{{h_j}^2}\right). \tag{37}$$

The above two equations first suppose that we attributed the same width factor h_j to the two clusters centreed on X and Z, and secondly that we suppose that the weight factors w which are respectively associated to these clusters are equal too (which is natural since we consider the function as constant over that range).

Making the estimates (36) and (37) to provide the same approximation for the function at points X, Y and Z then leads to

$$R = \sqrt{\ln 2}\, h_j\,. \tag{38}$$

As explained in the section about the IRVQ estimate of probability densities, a similar development which leads to the same result (independent of the dimension of the input space) can be made in dimensions greater than 1.

Now that we have the relation between h_j and R, we need a method to evaluate R. First, we will evaluate the inertia (variance) of each cluster, by using an adaptive method exactly as the competitive learning does for the locations of the centres. The inertia coefficient i_j for each cluster is computed according to

$$i_j \leftarrow i_j + \alpha\left(\left\|x_i - c_j\right\|^2 - i_j\right). \tag{39}$$

where j is the index of the closest centroid to a learning vector x_i. Equation (39) is

a convex combination at each iteration between the previously estimated value of i_j and a new contribution $||x_i - c_j||^2$ due to the input vector x_i. After learning, parameters i_j, $1 \leq j \leq K$, will converge to the average inertia of points in the clusters associated to c_j.

The last point to solve is the relation between the estimated inertia i_j and the distance R. If we consider that, under the locally uniform density approximation as above, the local arrangement of the centres of consecutive clusters will be as the vertices of an hypercube with edges of length 2 R, the relation between the inertia of each cluster and R is

$$i_j = \frac{1}{(2R)^d} \int_V \left\| x_i - c_j \right\|^2 dV = \frac{dR^2}{3} \tag{40}$$

where d is the dimension of the space. Combining equations (40) and (38) then leads to a width factor h_j given for each cluster j by

$$h_j = \sqrt{\frac{3i_j}{d\ln 2}} . \tag{41}$$

Reminding that h_j is adapted according to equation (35), leading to

$$h_j = q\sqrt{i_j} \tag{42}$$

after convergence, the optimal value of q is given by

$$q = \sqrt{\frac{3}{d\ln 2}} . \tag{43}$$

Second hypothesis: constant density, linear function

We know consider a less restrictive case, where the density of input vectors x_i can still be considered as constant over two consecutive clusters, but where the function y to be approximated is no longer constant, but can be linearly approximated over the range of two consecutive clusters.

In this case, it is not natural anymore to consider that the weight factors w associated to two consecutive clusters are equal; however, compared to the first case examined above, we can consider that the linear approximation on function y only influences the computation of the weight factors w proportionally to the slope

of function y, without any other influence on the computation of the centres and widths of the Gaussian kernels.

It can easily be verified that adding the linear hypothesis on y instead of the constant hypothesis in equations (36) and (37) does not influence the result of equation (38). The main result of equation (43) is thus still valid too.

Let us mention however that the above development is no longer correct if the function y to be approximated greatly differs from its linear approximation on two consecutive clusters; rather than being a limitation on the functions that can be approximated by this method, this fixes an upper bound (or at least an order of magnitude) on the distance between two consecutive clusters or, in other words, a lower bound on the number of clusters.

Third hypothesis: linear density, linear function

Let us finally consider an even less restrictive case. We make now the hypothesis that both the function y and the density of input vectors x_i can be linearly approximated on two consecutive clusters. This hypothesis is not far from what can be found in real cases: again, such a hypothesis determines a lower bound on the number of clusters (or Gaussian kernels) that must be used in the approximation, rather than being a limitation on the function y itself.

The only relation where the density of points appears is equation (40) that must be changed into

$$i_j = \frac{1}{(2R)^d} \int_V \left\| x_i - c_j \right\|^2 p(x) dV \tag{44}$$

where $p(x)$ is the density of points x_i in the cluster. However if this density is linear and if c_j is the centre of the cluster, the previous result

$$i_j = \frac{dR^2}{3} \tag{45}$$

holds (if the density is linear, summing the contributions of $p(x)$ at equal distances on both sides of the centre gives a constant; multiplying by $||x_i - c_j.||^2$ does not change anything to this comment because of the power 2). Moreover, as in the second hypothesis, the multiplication of the Gaussian kernel outputs by different w_j weight factors does not influence the computation of the centres and widths of the kernels. As a consequence, result (43) is still valid in the third least restrictive case.

3.4.3.3 Global optimisation

It has already been mentioned that the method of splitting the search for parameters c_j, h_j and w_j into three independent sets only leads to a minimization of error E (30) with respect to parameters w_j, but not with respect to the two other sets. To avoid this drawback and to find a (local) minimum of function E (30) defined with respect to the three sets of parameters, one can perform a gradient descent on function E simultaneously on the three sets of parameters. Using this method however suppresses the advantage of simplicity of learning, which makes RBFN so attractive in comparison with other neural network models. As a compromise between precision and simplicity, a gradient descent can be performed on the function E with respect to the three sets of parameters, but taking as initial conditions the results of the splitted evaluation of parameters in the previous sections. This will thus converge to a local minimum of function E defined with respect to the whole sets of parameters, but one can expect that the number of iterations needed in the gradient descent process will be small since the initial conditions will be close to the convergence point. For details about this optimization procedure, see [Verleysen 94]. Note that iterating gradient descent on nonlinear parameters and direct computations of weights is also possible.

3.4.4 Further research topics

The same reference [Verleysen 94] shows an example of approximation of a 1-dimensional function, using the RBFN procedure described above. This example shows improved performance of this method over the traditional Moody's and Darken's procedure. Nevertheless, we chose not to detail here simulations, for the following reasons:

- general scepticism about simulations, as detailed in the preface...

- while good results have been obtained on most simulations performed on standard benchmarks in small dimensions (1, 2 and 3), results are more controversial in higher dimensions. Some are convincing (clearly improved performances with respect to Moody and Darken's method), some are less convincing because the variance of the results is too high. This can be explained by the following comment. Most of the experiments have been carried out on artificial databases, which were built complex enough to lead to poor approximation results with conventional algorithms. It seems that vector quantization on the input samples on these examples could have been deficient, with consequences on the approximation results.

- the above development of the learning procedure for RBFN parameters is based on the Gaussian shape of kernels. As mentioned in the context of the IRVQ probability density estimation, the Gaussian shapes appear not to be optimal in high dimension.

- the derivation of factor q is based on assumptions about the small size of Voronoi regions, making the constant or linear hypotheses over the function to be approximated valid. Nevertheless, small sizes in high dimensions means an exponentially increasing (with the dimension) number of clusters. A method to estimate the factor q should be found without the need for this hypothesis (thus taking into account the targets y_i).

- finally, centres of Gaussian (or other radial) functions should not be fixed (only) according to the density of input values, but also according to the difficulty to approximate function y locally. The placement of the basis functions should also take into account the (estimate of the) first and second derivatives of function y.

While the above procedure for RBFN learning shows conclusive advantages over traditional methods, answering the above questions is a topic for further work.

3.5 Further developments on local learning

This chapter does not pretend to cover all aspects of learning with local models. On the contrary, we chose to focus on a few aspects, i.e. on original developments first about Bayesian classification through probability density estimation, and then on approximation with Radial Basis Function Networks. Some issues about vector quantization are also developed, for they constitute the background of these two original methods.

The developments presented in this chapter are improvements over traditionally used algorithms. In particular, they were designed to handle more efficiently high-dimensional data. Experiments proved the improvements. Nevertheless, our experience with these algorithms suggests that further work has to be achieved, still to improve the performances in high-dimensional settings. In particular, the method developed to estimate the probability density of a distribution (with Bayesian classification in view) uses Gaussian kernels, whose shape is not adapted to high-dimensional spaces (see chapter 2). We expect that using more appropriate kernels could lead to improvements. Hypotheses used in the derivation of the algorithm must also still be validated in high dimensions.

Concerning function approximation, we strongly believe in the advantages of Radial Basis Function Networks, compared to more conventional Multi-Layer Perceptrons for example. RBFN are easier to use, their performances are less sensitive to local minima, they do not require complex optimization algorithms, etc. Nevertheless, again, successful applications of RBFN are usually found in the literature for low-dimensional spaces only. The learning principles presented in this chapter, in particular the way to estimate adequate kernel locations and widths, are designed to improve the performances of RBFN, while keeping learning simple. These principles were developed under hypotheses that are not

very different from those considered in density approximation. For the same reasons, these hypotheses should be checked in high-dimensional settings, and the algorithm modified accordingly; the possibility to use non-Gaussian kernels should be considered too.

Chapter 4

Dimension reduction

4.1 Introduction

By dimension reduction, we mean a way to transform vectors (samples) defined in dimension d, into vectors defined in dimension q, with $q \leq d$. Of course, dimension reduction must obey certain rules, or criterions, in order to be useful.

Dimension reduction is made necessary because handling data in large dimensions is not easy. Chapter 2 dealt with the problems related to visualisation and intuitive perception of data in dimension greater than 3, while chapter 3 concerns learning in high-dimensional spaces. Despite the techniques described in these chapters, working with high-dimensional data remains difficult. All efforts should thus be done in order to work with data whose dimension is as small as possible.

When some knowledge about the data is available, one should of course take benefit from this knowledge in order to represent or decompose the data in small-dimensional vectors.

This chapter deals with the situation where no information is available about high-dimensional vectors, making it possible to decompose them (or when all information available has already been taken into account, but still leading to high-dimensional data). Our aim here will be to find *blind* methods to reduce the dimensionality of the space; blind means here that no a priori information about the data is used to perform this dimension reduction.

4.1.1 Dimension reduction and bias-variance trade-off

Dimension reduction does not only lead to easier representation and improved learning. It is also an important step towards better generalization (at fixed quality of learning), or, in more statistical terms, towards a better bias-variance trade-off. However, let us examine here intuitively this crucial question related to learning.

Figure 23 shows a dataset of 5 points in a 1-dimensional space. For illustration purposes, we used here polynomials of increasing order to approximate this dataset, rather than neural networks. The reason is that the number of parameters in a polynomial can be incremented easily and is directly related to the number of *degrees of freedom* of the approximation function. Such discussion is not so simple for example with RBFN networks, where weights obviously play the role of free parameters, but centres and kernel widths do not play exactly the same role. Nevertheless, the following comment is valid for polynomial interpolation as well as for RBFN (or other neural model) learning.

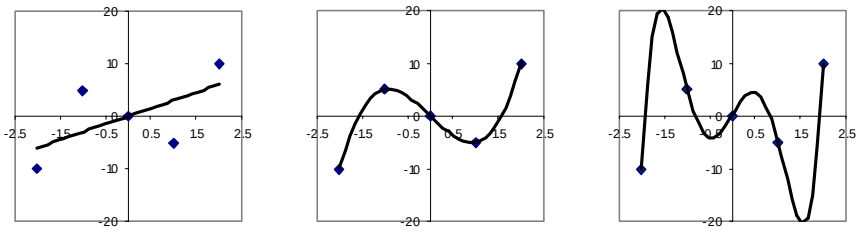

Figure 23: approximation of a dataset by polynomials of increasing order.

The dataset in Figure 23 is approximated by polynomials respectively of degree 1, 3 and 5. Degree 1 is obviously not enough in order to approximate data correctly. On the contrary, the centre and right graphs of the figure show that both degrees 3 and 5 are adequate to interpolate correctly the dataset. More precisely, the mean square error between the targets and the approximation, measured on all points of the dataset, is greater than 0 in the left graph, while it is equal to 0 in the middle and right graphs.

Nevertheless, if we have to choose between the approximations of the middle and right graphs, we will choose the centre one, without discussion. The reason is that we see intuitively on the graphs that intermediate points, not contained in the dataset, will be better approximated by the function in the middle figure than the function in the right one, if these intermediate points are drawn from the same "smooth" distribution. The approximation function in the left graph has a bias, while the one in the right graph expresses too much variance with respect to the dataset.

The conclusion of this discussion is that the number of degrees of freedom of an approximation function must be chosen high enough to interpolate the data correctly, but not too high in order to limit *overfitting*. Choosing a good compromise is an important and difficult question that will not be addressed here. However, our efforts towards dimension reduction can be seen as a contribution to this problem. Indeed, neural models such as MLP and RBFN, have a number of

parameters directly dependent on the number of their inputs. Single-output MLP with one hidden layer have a number of parameters equal to

$$P = Hd + 2H + 1 \tag{46}$$

where H is the number of neurons in the hidden unit and d the dimension of input vectors (assuming parametric thresholds in each neuron), while RBFN have

$$\text{between } P = H \text{ and } P = H(d + 2) \tag{47}$$

parameters, depending whether the centres and/or widths of radial kernels may be considered as free parameters or not. (Our point of view is that it is not correct to count them as parameters equivalent to weights, since the quality of approximation is not sensitive to small variations of these parameters, neither it is correct to count only weights, since an adequate placement of centres and widths improves approximation; the correct evaluation of the number of *degrees of freedom* is probably between these two extremes.)

In both cases, we see that reducing the number of inputs d leads to a corresponding reduction in the number of degrees of freedom of the problem, and thus to a better bias-variance trade-off. Of course, the dimension reduction should not lead to loss of information: if we compare two problems, one with d inputs and the other with q ($\leq d$) inputs, the transformation from the first to the latter must leave all other considerations unchanged. In particular, the information contents in the input data (and in the targets) must remain unchanged too. Reducing the dimension of vectors (input samples) without altering (too much) the information they contain is the key question in the following.

4.1.2 Dimension reduction and intrinsic dimension

The assumption justifying dimension reduction is that the dataset actually lies on a (nonlinear) manifold of smaller dimension than the data space. In this case, it is theoretically possible to perform dimension reduction without significant loss of useful information. It is then natural to consider that dimension reduction will be efficient down to the intrinsic dimension of the dataset, but inefficient for smaller dimensions. Efficient means here that the goal of dimension reduction is achieved, i.e. that there is no significant loss of information in the reduction.

Measuring the usefulness of information in this context, i.e. deciding if the (unavoidable) loss of information has an impact or not on further processing, is a difficult task. As we will see below, the loss of information is usually measured in terms of variance of the dataset in the linear dimension reduction case. A similar measure in nonlinear context inevitably depends on the reduction method itself.

Nevertheless, whatever the measure considered, the method used to perform dimension reduction will not do better than guessing the submanifold on which the data are supposed to be located. Any deviation of the initial data (the samples) from this submanifold will thus be erased in the reduction process, exactly as deviations from a planar surface in a three-dimensional space disappear when the data are projected on the surface. In some situations, these deviations may be considered as noise. Therefore, dimension reduction not only reduces the size of the vectors, but also reduces the noise in the data. These two aspects of dimension reduction should be considered and evaluated separately. Nevertheless, as they both are consequences of the same operation, it is extremely difficult to appreciate what are the respective amounts of noise reduction and of loss of useful information. This aspect of dimension reduction should however be kept in mind, advantages and drawbacks being closely mixed here.

Besides this (important) question about the relevance of the information lost in the dimension reduction process, two difficulties remain that will impinge on how the reduction is performed.

First, the intrinsic dimension of the data is usually unknown. Chapter 2 described methods to estimate the intrinsic dimension, but it was made clear that the concept itself of intrinsic dimension may be discussed. For example, different definitions and associated measures lead to different estimations, all of them being justifiable. Intrinsic dimension estimations must be taken as approximate measures of a lower bound to the dimension on which a dataset can be projected. This problem is related to the above discussion of bias-variance trade-off: when the dataset is reduced to a too low dimension, some information is lost, and any further approximation (or classification, etc.) will be less successful than if it had been performed on the original dataset; bias will be introduced. On the other hand, if the dimension of the dataset is not sufficiently reduced, some non-necessary information (like noise) or correlated information (see below) will remain, worsening the performances of the subsequent approximation too. A good choice of the intrinsic dimension (or better the dimension on which the data will be projected) is thus a compromise between these two drawbacks; only simulations (trials and errors) can usually lead to a good choice.

Secondly, and this is the justification of the following in this chapter, data are usually linked by nonlinear relationships. Remember the example of Figure 12, where two-dimensional data form a non-planar surface in a three-dimensional space. The importance of nonlinearities is usually underestimated, based on our traditional experience of conventional statistics. Indeed most standard tools as the Principal Component Analysis described below are aimed at removing linear dependencies (second-order correlations) between data or features. Second-order correlations have numerous advantages: they are easy to measure, the differentiation of error functions leads to linear formulas, etc.; this is the main reason for their widespread use. Nevertheless, real-world data exhibit nonlinear correlations; adapted methods should then be developed, as it will be seen below.

Note that in some literature, the term *correlation* is restricted to second-order dependencies. In the following, we will use this term for any (linear or not) dependency, following the conventional use of this term in the neural network community rather than in the statistical one.

4.1.3 Dimension reduction in classification

Classification methods and algorithms usually perform clustering of the space. They rely on distance measures between vectors; for example, regions defined by the set of points nearest from a centroid than from any other (Voronoi regions) are associated to class representatives (the centroids) and by this way to their class. The distance measure between samples is thus a key criterion for the classification.

Nevertheless, we mentioned in chapter 2 that the concept itself of distance between vectors becomes meaningless in high-dimensional spaces. In particular, we mentioned a result proving that samples on a random distribution seem to be normalized in high-dimensional spaces, whatever the distribution. But if samples are randomly drawn from a distribution, distances between samples are random vectors too. Distances between vectors seem thus also to be normalized (concentrated around a fixed value)! This is one of the reasons why classification algorithms fail in high-dimensional spaces; the reason is not related to the method, but to the concept of (Euclidean) distance behind. An efficient dimension reduction of the samples will thus improve the results of a subsequent classification in such situations.

4.1.4 Why and how to perform dimension reduction?

Dimension reduction is made necessary in a wide range of learning problems, because learning high-dimensional data remains difficult, despite all efforts made to overcome the problems. When this is possible, data should thus be made as low-dimensional as possible, of course without loosing relevant information. Dimension reduction could be compared to preprocessing of data as performed in most pattern analysis schemes. In the latter case however, features are extracted in a "non-blind" way, i.e. according to what can be expected from the data, while in the first one blind extraction of information is performed.

The following of this chapter describes method to perform blind dimension reduction. Classical linear Principal Component Analysis is first described, before non-linear methods more adapted to the preprocessing of data before non-linear data analysis tools. Finally, an original application of the concept of non-linear dimension reduction to time-series prediction is presented.

4.2 Linear dimension reduction

4.2.1 Principal component analysis

Principal Component Analysis (PCA), that finds its origin in [Pearson 01], is the standard, commonly used method to reduce the dimension of a dataset. The concepts of PCA have been described in chapter 2. While PCA is a very classical method widely used by some scientific communities, it remains unfortunately unknown or badly used by others. For this reason, and because it forms the background to nonlinear projection methods described in the next section, we chose to detail PCA here in more mathematical terms. For this description, we will follow [Choppin 98].

PCA is a method that projects d-dimensional samples on a submanifold of lower dimension. The principle is to find an "ideal" subspace to project the samples, i.e. a subspace where the projections of the samples will keep most of the information contained in the original set.

The term *projection* takes here its meaning of linear operator. The traditional concept of projection means that a sample x_i will be replaced by (projected to) another d-dimensional point y_i situated on the subspace. The projection is orthogonal, i.e. y_i is chosen to minimize the distance between y_i and the original sample x_i. In some literature the term *projection* is reserved for linear transforms of data. We will use it here both for linear and nonlinear transforms, linear projections being associated to the conventional PCA method.

Linear projection means that PCA is defined by a *projection matrix*, whose lines (or columns) are the reference vectors of the subspace. The point here is to determine the subspace to project on, with the minimum loss of information. In other words, it means to define the subspace so that average distance between the initial points x_i and the projected ones y_i is minimal.

4.2.1.1 PCA reference vectors

We may see the task of determining the subspace to project on as a sequential process: find the direction corresponding to the best axis passing through the set of samples; then supposing that all the points were in a plane perpendicular to the first axis, find the second best direction; and so on. The "best" axis is defined in the least mean square sense, i.e. by minimizing the mean difference between a sample and its projection on the axis.

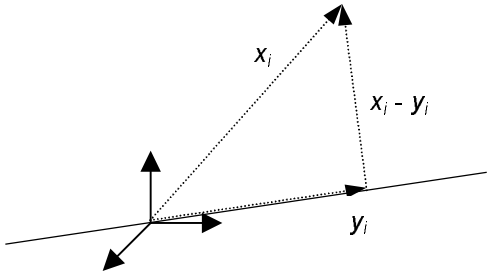

Figure 24: projection of samples on an axis, and associated notations.

Let us look more formally at the first best axis of projection. Looked for is the unit vector u_1 for which (see Figure 24)

$$E_1 = \sum_{i=1}^{N} \|x_i - y_i\|^2 \tag{48}$$

is minimized. Since $\|x_i - y_i\|^2 = \|x_i\|^2 - \|y_i\|^2$, this is equivalent to maximize

$$E_1' = \sum_{i=1}^{N} \|y_i\|^2 . \tag{49}$$

As

$$y_i = x_i^T \frac{u_1}{\|u_1\|} . \tag{50}$$

is the orthogonal projection of x_i on u_1, the projection of each sample can be rewritten in matrix notation:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} x_1^T u_1 \\ x_2^T u_1 \\ \vdots \\ x_N^T u_1 \end{pmatrix} = X u_1 . \tag{51}$$

The problem is finally to find the unit vector u_1 that maximizes equation (49), i.e.

$$u_1 = \arg \max_{u_1^T u_1 = 1} \left(u_1^T X^T X u_1 \right). \quad (52)$$

The solution of this problem is well known in linear algebra: u_1 is the eigenvector of $X^T X$ associated with its largest eigenvalue λ_1.

Looking at the quantity (49) to maximize, it appears that this quantity is nothing else than the variance of projected points. In other words, the axis for which the mean square distance (48) is minimal is the axis that maximizes the variance (49) of the projected points.

4.2.1.2 Geometrical properties

Figure 25 shows two examples of projections of dataset on the best and worst axes, respectively left and right figures. In the first case, the variance of the projected points is maximal, while it is minimal in the second case. The left figure also corresponds to the most appropriate representation of projected points: they are as apart as possible on the axis.

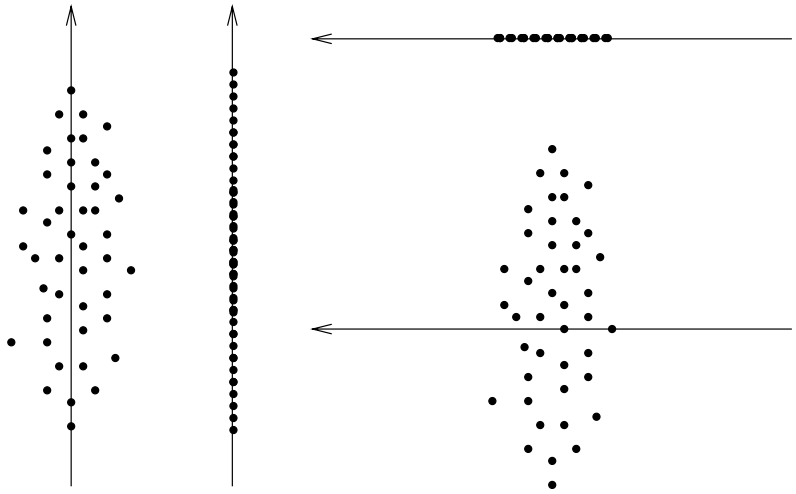

Figure 25: projection of samples on the best and worst axes.

The first axis u_1 found by maximizing the variance of the projected points is called the *first canonical axis* or the *first principal component*.

The process of finding the "best" axis can be repeated on the data after projection on a subspace orthogonal to the first canonical axis, to find the second one, and so on. It can be proved that the axes found by this process correspond to the

eigenvectors of matrix $X^T X$, in decreasing order of their associated eigenvalues.

Another nice result of PCA is that the percentage V_p of variance "kept" after projection on the first p canonical axes is equal to the normalized sum of the associated eigenvalues

$$V_p = \frac{\sum_{j=1}^{p} \lambda_j}{\sum_{j=1}^{N} \lambda_j}. \tag{53}$$

This result gives the possibility to choose the dimension of the projection space, once a required percentage of variance kept after projection has been fixed. If all dependencies between data could be linear, this would give us a way to estimate the intrinsic dimension of a dataset. Unfortunately, nonlinear dependencies are more common in engineering and physical problems than linear ones, and are not taken into account by a second-order method as PCA.

4.2.1.3 Advantages and drawbacks

PCA is a widely used method for dimension reduction. Its success partly comes from its applications in visualisation, and more precisely from the similarity between PCA and natural vision. Let us imagine indeed that we have to view (in two dimensions) a three-dimensional object. If the possibility exists, we will move (or move the object) so that we will see its "best" view. Best is defined in the sense that we will see the largest number of details in the object, or, in mathematical terms, we will maximize the variance of its view. PCA is nothing else than a generalization of this concept to higher-dimensional spaces.

Besides this fact, PCA has a number of advantages:

- PCA is a well-known technique. It is used and has proved to be useful in many applications fields, not only by specialists of data analysis. Its success makes that efficient implementations of the method exist, and can be found in commercial softwares.

- Unlike many adaptive or neural-based methods, PCA does not suffer from parameters that influence the convergence or the performances of the method. Except numerical problems that could occur in matrix inversion, the PCA method is straightforward.

- The PCA method leads to an objective measure of the information kept/loss in the process.

But it also has drawbacks:

- Despite the fact that there is no parameters tuning in PCA, inversion or diagonalization of matrix $X^T X$ may be (and is often) ill-conditioned. Numerical techniques must be used to properly find its eigenvalues and eigenvectors.

- PCA is a linear method, which is sensitive to (second-order) correlations only. PCA will thus not catch any nonlinear relationship between data. Looking back for example to Figure 12, it can be seen that it is not possible to find a two-dimensional plane adequate for a projection of this dataset.

This last comment is of highest importance for the following of our work. We showed that it is necessary to reduce the size of the space in many situations, for the reasons explained in the introduction of this chapter. Now, we showed that PCA is not an adequate method (for dimension reduction), at least if strong nonlinear relationships exist between data (which is the case in most real situations). There is thus a need for nonlinear dimension reduction tools. This is the topic of the next sections.

4.3 Nonlinear dimension reduction

A lot of tools or algorithms that can be used for nonlinear dimension reduction exist in the literature. It is not our intention here to make an exhaustive review of these techniques. Some are used in data analysis, some in statistics, some in the neural network community, or in other areas, and it is difficult to compare them objectively in general situations. We have to remind that, unlike linear methods such as PCA, nonlinear methods suffer from two difficulties concerning the evaluation of their performances:

1. nonlinear methods are usually adaptive and/or iterative; this means that parameters (strongly) influence the convergence of the algorithm. Making a method obtaining good results is thus often a question of efforts devoted to parameter tuning. Of course, a primary objective in designing the methods themselves is to make them as insensitive as possible to all parameters. Nevertheless, some sensitivity still remains in most cases.

2. Quality criterions are not so obvious as in linear cases. As described in chapter 2, Euclidean distance measures loose their meaning in high dimensions. Furthermore, most standard correlation criteria are based on second-order relationships; they are thus adapted to measure linear projections, but not nonlinear ones.

In the following, we will briefly describe two methods used for dimension reduction, that we feel necessary to understand the following: Kohonen maps in section 4.3.1, and Multi-Dimensional Scaling and Sammon's mapping in section 4.3.2.

Then, section 4.3.3 will detail a third method on which we focused our attention, for some reasons explained below. The first method is known in the neural network field, while the second one is mostly known is statistics. The third one uses interesting features from both.

4.3.1 Kohonen maps

Kohonen maps [Kohonen 95], or self-organizing feature maps, are now widely known methods in the field of artificial neural networks. Kohonen maps are basically a vector quantization, coupled to an interesting property of topology preservation. Shortly, the two aspects of Kohonen maps work as follows.

First, *d*-dimensional samples are quantized as in any other vector quantization methods. This means that the initial dataset containing *n* *d*-dimensional samples is replaced by a reduced set, containing *m* *d*-dimensional samples. We call *codebook* the reduced set. The codebook is designed in such a way that it contains as much information as possible from the original set. It is usually built by minimizing the mean Euclidean distance between the original samples and the centroids (samples from the codebook), but it can also be viewed as building a set of centroids with the same properties as if they were randomly sampled from the initial (unknown) distribution of vectors. The number *m* of centroids is a parameter of the method.

Besides this traditional vector quantization aspect, centroids are labeled in the Kohonen algorithm. These labels can be natural numbers, pairs of natural numbers, or *n*-tuples of natural numbers; usually couples of natural numbers are considered. The centroids can thus be seen as ordered on a grid, whose coordinates are these sets of natural numbers. More precisely, when the centroids are placed in the *d*-dimensional space (after vector quantization), a grid can be drawn, that goes through the centroids in the order of their labels.

The nice topological property of the Kohonen algorithm is that, after convergence, the grid will be smooth. In other terms, two centroids whose labels are close to one another (two close centroids on the grid) will have close locations in the *d*-dimensional space. How this property is achieved is the consequence of the algorithm itself, and goes beyond the scope of this discussion.

Our interest here is to see that centroids can be characterized in two ways. First, they are defined by their position in the *d*-dimensional space. Secondly, they are identified by their position on the grid; this position is a two-dimensional vector. Going from one system of coordinated to the other may thus be seen as a reduction from dimension *d* to dimension 2. Figure 26 shows a 2-dimensional Kohonen map after convergence in a horseshoe distribution. In theory, any dimension other then 2 could be taken too. (Nevertheless, Kohonen maps are mostly used for visualisation purposes, so dimension 3 is seldom exceeded.)

Since the transformation performed by the Kohonen map preserves the topology, these can be seen as a nonlinear dimension reduction method. This property has for example been exploited in an image compression scheme as described in [Amerijckx 98].

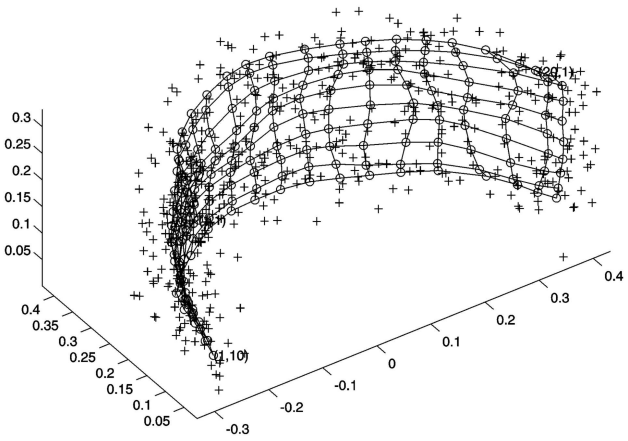

Figure 26: Horseshoe distribution and resulting Kohonen map. From [Choppin 98].

While Kohonen maps are easy, powerful methods in many data analysis problems, they suffer from two drawbacks concerning dimension reduction:

1. We already mentioned that the grid dimension of Kohonen maps is usually set to two. Higher dimensions are rarely used. There is thus not so much experience about the efficiency of Kohonen maps (including possible convergence problems, high-dimensional restrictions, etc.) in these dimensions.

2. The grid used in the Kohonen map is fixed in advance. This means that the topology of the grid is not necessarily in accordance with the dataset. Figure 27 shows a typical well-known example of Kohonen map which is not adapted to the distribution: the map used is a typical two-dimensional rectangular one, while the data are distributed in a cactus form (from [Kohonen 88]). It may be seen that even after convergence centroids are located in regions of the space where there are no samples from the distribution (as a result of the non-convexity of the distribution). More seriously than in the toy example illustrated here, this phenomenon easily occurs with high-dimensional distributions. Note that there are now attempts to develop variants of the Kohonen algorithm with evolvable grid; see [Dittenbach 00] for example.

3. Kohonen's algorithm includes vector quantization. Samples are thus not only projected, but also quantized. This may be undesirable in practical situations. To remedy this point, several possibilities exist to interpolate between centroids in the projection space. Among them, projection on the plane defined by the three nearest centroids, mapping similar to Sammon's one (as explained in the next section), or variations around these themes, are the most common [Lee 99].

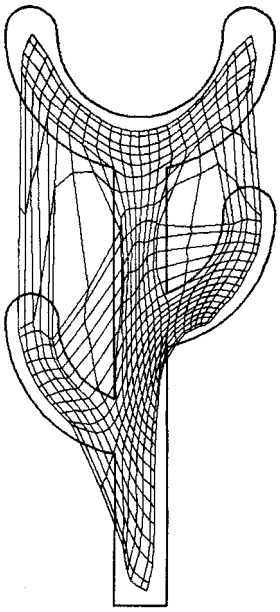

Figure 27: Two-dimensional rectangular Kohonen map in a "cactus" distribution. From [Kohonen 88].

4.3.2 Nonlinear multi-dimensional scaling and Sammon's mapping

The purpose of these two methods is again to project *d*-dimensional samples on *q*-dimensional ones, keeping most or all of the information contained in the initial samples. The principle is here to place points in the *q*-dimensional space, and to measure how the distances between these points in the output space are similar to the distances between the corresponding points in the input space.

In the case of nonlinear multi-dimensional scaling (MDS) [Shepard 62, Shepard 65], the objective function is simply the ratio of the input distances by the output distances, weighted in such a way that small output distances are more important

than large ones. Weighting aims at conserving a local topology (locally, sets of input points will resemble sets of output points). Conserving global topology is only possible in specific unrealistic situations; emphasizing local topology is thus a way to realize most of the objective in practical circumstances.

Sammon's mapping [Sammon 69] is similar to MDS. However, the objective function is now a mean square error between distances (between pairs of samples) in the input and output spaces. Contrary to MDS, weighting is done with respect to input distances in Sammon's mapping.

In the next section, we will present a new nonlinear projection tool, the *curvilinear component analysis* (CCA), proposed by Demartines and Hérault [Demartines 97], and our improvements to this method. CCA has common points with MDS and Sammon's mapping, but has several advantages too.

4.3.3 Curvilinear component analysis (CCA)

4.3.3.1 Description of the method

The basic idea behind CCA is similar to Sammon's mapping and MDS. The aim is to fix points in the q-dimensional space, with a topology similar to the topology of the initial points in the d-dimensional space. In CCA's spirit, the word *topology* means "the distances between all pairs of points in the database". So CCA tries to find coordinates in the projection space such that they reproduce the distances measured in the initial space. Formally, CCA works by minimizing an error function that is nothing more than a sum of squares of the differences between distances in the original and projection spaces:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(X_{ij} - Y_{ij} \right)^2 F\left(Y_{ij}, \lambda \right) \tag{54}$$

where N is the number of points in the database, X_{ij} is the distance between points x_i and x_j in the original d-dimensional space, and Y_{ij} is the distance between points y_i and y_j in the projection q-dimensional space. $F(Y_{ij}, \lambda)$ is a monotonically decreasing function of the distance Y_{ij} in the projection space, parameterised by λ. This function gives more importance to small distances, and therefore to the conservation of the local topology. Demartines suggests to use a step function of the form

$$F\left(Y_{ij}, \lambda \right) = \begin{cases} 1 & \text{if } Y_{ij} \leq \lambda \\ 0 & \text{if } Y_{ij} > \lambda \end{cases}. \tag{55}$$

The weighting function $F(Y_{ij}, \lambda)$ depends on the output distances (contrary to most dimension reduction methods). This implies a recursive procedure to find the locations of the point y_i in the projection space.

As conventional gradient descent on the error function E can be particularly tedious and computationally heavy, Demartines suggests a simplified procedure where each point y_i is in turn considered as fixed, and all other points around y_i are moved. Under some supplementary assumptions, this lead to the adaptation rule

$$\Delta y_j = \alpha(t) F(Y_{ij}, \lambda)(X_{ij} - Y_{ij}) \frac{y_j - y_i}{Y_{ij}}, \quad \forall j \neq i . \tag{56}$$

Working with equations (54 to 56) is however still computationally intensive: the number of distances both in the initial and projection spaces is proportional to N^2! CCA then uses vector quantization to reduce the size of the database in the initial space, in order to decrease drastically this computational load. Only centroids are then adjusted according to criterion (54) and adaptation rule (56). The drawback is that centroids only are projected, and an interpolation procedure must be designed to make the correspondence between a point x_k (different from a centroid) in the initial space and a point y_k in the projected space. This is done by minimizing the same cost function (54) but for point y_k only.

For in-depth comparison between CCA and other conventional nonlinear projection tools, we refer the reader to [Demartines 97]. The same reference also suggests a way to measure the quality of the projection, in the form of a graph.

It can be shown that CCA outperforms other nonlinear projection algorithms in many situations. Using a simple criterion based on distances in the projection space, and using a simplified procedure for gradient-like descent are the two major arguments.

4.3.3.2 Choosing the parameters for CCA

CCA requires three parameters: first the projection space dimension q, and secondly the two time decreasing parameters $\alpha(t)$ and $\lambda(t)$ used by the adaptation rule. Dimension q can be computed by fractal dimension estimation [Grassberger 83] or by LPCA (see below). The learning factor $\alpha(t)$ requires no particular attention (for example, exponential decrease between 0.95 to 0.01). However, the neighborhood factor $\lambda(t)$ is critical: if $\lambda(t)$ decreases too slowly, the nonlinear dependencies are not well unfolded, whereas a fast decrease compromises the convergence.

Although CCA outperforms many other nonlinear projection algorithm, this dependency on critical parameters (and on the density of samples) raises

difficulties to unfold "hard nonlinear" structures like a spiral. In such a case, CCA converges with difficulty: large distances X_{ij} in the initial structure are poorly correlated with the corresponding distances Y_{ij} in the unfolded and projected structure.

4.3.3.3 Curvilinear Distance Analysis

Looking at the example of a spiral, CCA globally remains a good idea, but perhaps the use of another distance than the Euclidian one could improve the convergence. The best distance function should produce the same result for both the initial spiral and its projection. To reach this goal, one needs a kind of "curvilinear distance" δ_{ij}, like in Figure 28 (c). Such a distance, in accordance with the idea of sparse distance matrix suggested in [Guérin-Dugué 99], is computed *inside* the spiral and not *through* the spiral, like the Euclidian distance [Lee 00].

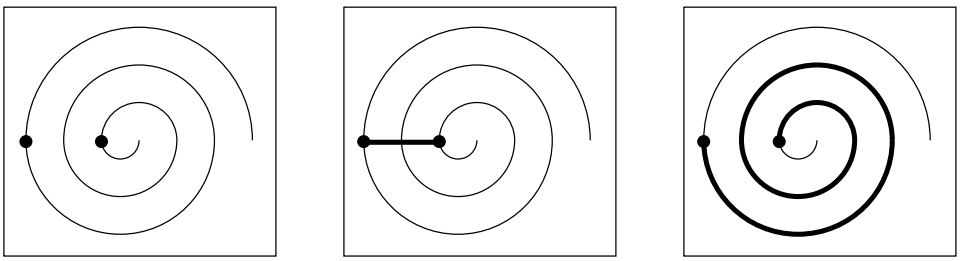

Figure 28: The curvilinear distance. (-a-) two points in a spiral. (-b-) the Euclidian distance between the two same points. (-c-) the curvilinear distance.

An approximation of the curvilinear distance can be computed in two steps (see Figure 29):

Step 1: Linking the centroids

After the vector quantization, the centroids can be linked (or connected) so that they become a graph. Two centroids get linked when they are the nearest ones from a database vector. This idea of linking centroids is not new (see for example the work of Bernd Fritzke [Fritzke 91]). In CCA, the first utility of links is visual: for example, crossing links often means projection faults.

Step 2: Computing a distance *via* the links

Links also have a second utility: they help to compute the above mentioned curvilinear distance. A good approximation of δ_{ij} is given by the sum of the

Euclidian lengths of all links in the shortest path from centroid *i* to centroid *j*, provided there are no "shortcut" links.

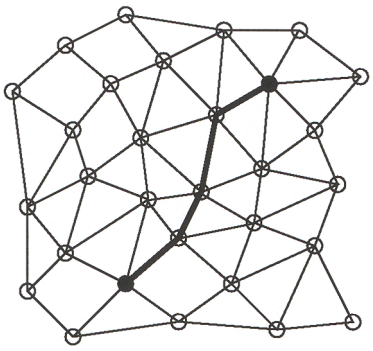

Figure 29: Approximation of the `curvilinear distance' by means of the shortest path the links between centroids (here the distance between both blackened centroids).

We propose an enhanced version of Demartines' CCA, called CDA (Curvilinear Distances Analysis). The objective function remains identical, but the Euclidian distance X_{ij} in (54) is replaced by the curvilinear distance δ_{ij}:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\delta_{ij} - Y_{ij}\right)^2 F\left(Y_{ij}, \lambda\right). \tag{57}$$

This objective function gives a new adaptation rule:

$$\Delta y_j = \alpha(t) F\left(Y_{ij}, \lambda\right)\left(y_j - y_i\right)\frac{\left(\delta_{ij} - Y_{ij}\right)}{Y_{ij}}, \quad \forall j \neq i. \tag{58}$$

According to (55), the weighting function $F(Y_{ij}, \lambda)$ should be a negative step function. In order to make the choice of λ less crucial, we suggest to relate the location of the step to the largest value of the argument Y_{ij}; moreover, λ is made decreasing with time, by analogy with adaptation parameters in gradient descent algorithms: starting with a too large value and then decreasing it makes it possible to have a rough approximation during the first iterations, and then to refine it gradually. Equation (55) then becomes

$$F\left(Y_{ij}, \lambda\right) = \operatorname{sign}\left(\lambda(t) \max_{k,l}(Y_{kl}) - Y_{ij}\right). \tag{59}$$

In order to obtain a compromise between the original CCA and the new objective function, one may define a *generalized distance* between centroids *i* and *j* in the original *d*-dimensional space:

$$\Delta_{ij} = (1-\omega(t))X_{ij} + \omega(t)\delta_{ij} . \quad (60)$$

The generalized distance Δ_{ij} helps to build a unique algorithm that combines the Euclidian distance and the curvilinear one; the result is an algorithm with a third parameter $\omega(t)$, varying between 0 and 1, and allowing to dynamically switch between classical CCA and CDA.

4.3.3.4 Automatic choice of the parameters

The second improvement brought to CCA is the complete automation for the choice of parameters. The goal is to get a method as simple as PCA, i.e. a method with a single parameter V_p (the percentage of variance kept after projection).

Parameters for vector quantization

Samples in the initial space are quantized in CCA, by a vector quantization (VQ) technique. Usually, adaptive VQ (as Simple Competitive Learning, Learning Vector Quantization, etc.) requires two parameters: the number of centroids and a learning factor $\alpha(t)$. In CDA, we use a dynamic VQ in which centroids are created when all availables ones lie further from a sample than a fixed threshold *r*. In other words, when a sample x_i is introduced in the VQ (SCL or other adaptive one) process, the closest centroid y_j is moved according to equation (7), only if

$$\mathrm{dist}(x_i, y_j) < r . \quad (61)$$

Otherwise, a new centroid y_k is created at the location of sample x_i. Of course, the smaller the threshold *r*, the better the VQ quality, so *r* can be made proportional to the tolerable loss *l*:

$$r = l \max_{i,j}(X_{ij}) . \quad (62)$$

Unlike PCA, the use of a "tolerable loss" *l* is qualitative only. There is obviously no proof that the use of equation (62) will ensure a quantization error smaller than *l*.

Parameters for CDA

CDA requires four parameters: first the projection space dimension q, and then the three parameters used by the adaptation rule $\alpha(t)$, $\lambda(t)$ and $\omega(t)$.

The optimal dimension q of the projection space is easily determined by a method called LPCA (local PCA [Kambhatla 97]). LPCA works by performing a vector quantization and then a PCA on each Voronoi region after vector quantization, assuming that the database is locally linear (i.e. at the scale of the Voronoi regions). Given the tolerable loss l, the required dimension p_{Vj} of the projection space is computed for each Voronoi region V_j, according to PCA standard procedure (the dimension is chosen so that the effective loss of variance is smaller than l). The global projection dimension p is simply the mean of all p_{Vj}. Again, no mathematical proof guarantees that p will lead to an effective loss smaller than l; however, we assume (and verify experimentally) that, in the range of a Voronoi region, CDA works at least as well as PCA. Note that nothing prevents to use the same vector quantization for CDA and for LPCA!

For the learning factor $\alpha(t)$ and the neighborhood factor $\lambda(t)$, one uses the classical exponential decrease (as in numerous adaptative algorithms), within the following arbitrary chosen bounds:

$$1 \geq \alpha(t) \geq 0.02 \,, \tag{63}$$

and

$$1 \geq \lambda(t) \geq \frac{\min_{i,j}(\delta_{ij})}{\max_{i,j}(\delta_{ij})} \,. \tag{64}$$

Note for completeness that an exponentially decreasing law does not satisfy the Robbins-Monro conditions; it is however widely and successfully used in practice.

In Demartines' CCA, the choice of the bounds for $\lambda(t)$ is quite difficult. The CDA method is less sensitive to the choice of $\lambda(t)$, due to the enhancement brought by the curvilinear distance: experiments show that the convergence is improved and accelerated.

Finally, how to determine the last parameter $\omega(t)$? Intuitively, if the dependencies in the database are almost linear, $\omega(t)$ should be set near zero since CCA works perfectly in this case. In the same way, if the dependencies are strongly nonlinear, a value for $\omega(t)$ near one should take profit of the curvilinear distances. With this reasoning in mind, the value of $\omega(t)$ is empirically set as follows. At time t, the current neighborhood D is defined as the couples (i, j) for which the curvilinear

distance between points x_i and x_j is smaller than the threshold defined in (59):

$$D(t) = \left\{ (i,j) \middle| \delta_{ij} \leq \lambda(t) \max_{i,j}(\delta_{ij}) \right\}. \tag{65}$$

We then measure the linearity of the database in this neighborhood. An indicative measure of the linearity is the mean of the ratios between the Euclidean distances and the corresponding curvilinear ones. The more "linear" the database is, the closer these ratios will be from 1. This results in $\omega(t)$ set as

$$\omega(t) = \min\left\{ \frac{\pi}{\pi - 2\sqrt{2}} \left(1 - \frac{1}{|D|} \sum_{(i,j)\in D(t)} \frac{X_{ij}}{\delta_{ij}} \right), 1 \right\}. \tag{66}$$

Note that the above described mean ratio will come close to 1 after the database is successfully unfolded (after convergence of the algorithm). Moreover, looking at equation (60), $\omega(t)$ must be close to 1 at the beginning of the convergence (to take advantage of the notion of curvilinear distance), but must approach 0 after convergence (the curvilinear distance δ_{ij} remains an approximation, so precision of the convergence will be increased by using the Euclidean distance X_{ij} instead δ_{ij} of as soon as the two distance measures are deemed to be representative of the same quantity). This explains the last parenthesis in equation (66): at the beginning of the convergence the neighborhood D includes pairs of points distant from one another (thus having low ratio X_{ij}/δ_{ij}), while only close points (with ratio X_{ij}/δ_{ij} close to 1) remain at the end. The multiplying factor of this parenthesis has been set heuristically (it has been calculated in order to make the product equal to 1 in case of a circle). Finally, $\omega(t)$ is bounded by 1 as required by its definition (60).

4.3.3.5 Examples

This section shows some artificial databases projected with the CDA algorithm. Their purpose is only illustrative: much more complex structures can be successfully handled by CDA, but their visual aspect is less meaningful.

Although the implementation allows to tune the parameters, all examples below are projected by the automatic method. Figure 30, Figure 31 and Figure 32 below include some vectors of the database (shown as points), all centroids (circles) and links (lines); centroids and links in the projection space are however not shown for Figure 31 and Figure 32.

Horseshoe

The horseshoe is a two-dimensional rectangle embedded in a three-dimensional space, slightly curved to obtain three quarters of a cylinder. The horseshoe is a classical benchmark for nonlinear projection method.

Figure 30 shows (left) the horseshoe distribution in a 3-dimensional space, and (right) its projection by CDA on a 2-dimensional plane.

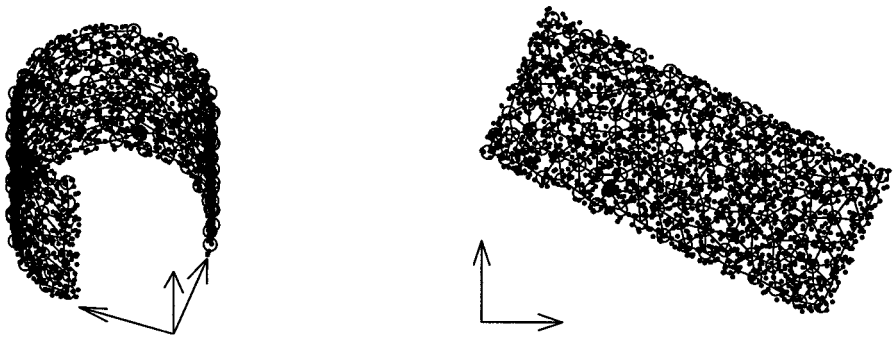

Figure 30: Left: horseshoe distribution in a 3-dimensional space. Right: its projection on a 2-dimensional plane.

Trefoil knot

The trefoil knot (Figure 31) is a mono-dimensional object embedded in a three-dimensional space. The CDA unties it in a mono-dimensional space rapidly and automatically.

Figure 31 shows (left) the trefoil knot in a 3-dimensional space and (right) its projection on a 1-dimensional line. Unit axes are represented to illustrate the scale of the diagrams.

Sphere

The projection of the sphere is a more complex. Indeed, a good projection on a plane requires that the algorithm cuts and stretches the sphere (if not, the projection would not be bijective).

Figure 32 shows (left) the sphere in a 3-dimensional space, and (right) its projection on a 2-dimensional plane. Links are not represented in the projected figure for clarity of the illustration.

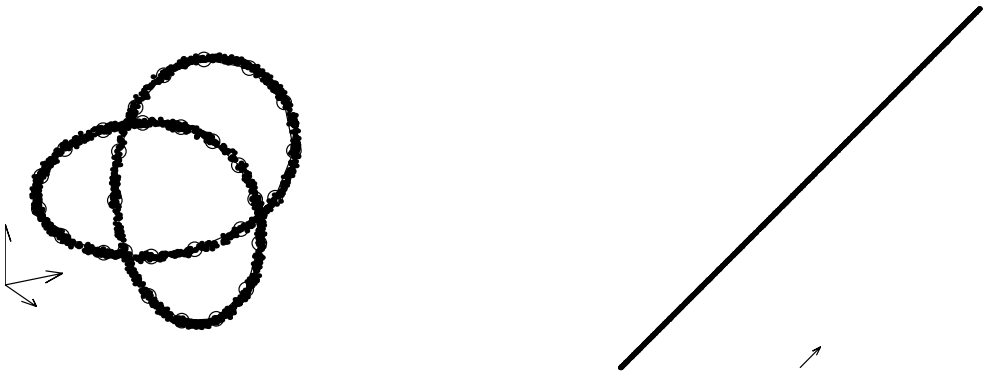

Figure 31: Left: trefoil knot in a 3-dimensional space. Right: its projection on a 1-dimensional line.

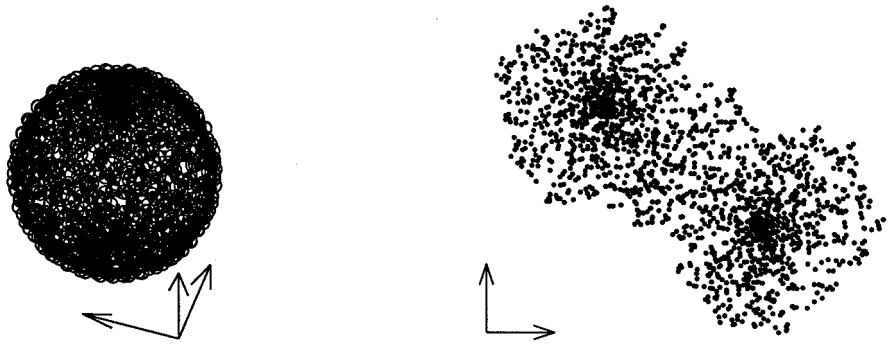

Figure 32: Left: sphere in a 3-dimensional space. Right: its projection on a 2-dimensional plane.

4.3.3.6 Discussion and further work

The CCA method nicely answers the problem of nonlinear projection. The whole process is as easy as PCA, but with the advantage of nonlinear capabilities. The method has similarities with better known projection methods like Sammons' mapping and MDS, but has proven to perform better on difficult artificial databases.

Two improvements have been suggested to enhance the robustness of CCA: the approximation of curvilinear distances, and more automation in the choice of the parameters. The enhanced CDA method has thus been developed to be as generic as possible. It must be clear however that, in such nonlinear context, there is a compromise to find between the generic character of a method, and its

performances on specific databases. Another concern, not addressed here, would be to develop CCA in order to make it as powerful as possible on a specific database, looking at the way to adjust the parameters according to the specific application, etc.

In our context of the development of a generic projection method, we believe that the interpolation algorithm could still be improved. Testing the performances of the method on various real-world datasets is also still a topic for further work.

Another point merits to be discussed. The concern of nonlinear projection with algorithms such as CDA is to unfold the datasets, better than what PCA for example would be able to do. Looking at Figure 31, this means that PCA applied on the dataset of the left figure would create confusions between points (non-bijective projection), while CDA generates a bijective projection. Unfortunately, this advantage is accompanied by a drawback: there is no other choice than to "break" the knot, at one point or another, in order to unfold it. Local neighborhoods are thus not maintained, at least for a small part of the dataset. The consequences of this drawback on the use of projected data instead of the original ones should be evaluated.

4.4 Application of dimension reduction to time-series forecasting

4.4.1 The time-series forecasting problem

Time series forecasting is a problem encountered in many industrial (electrical load, river flow...) and economic (exchange rates, stock exchange...) tasks. Often, prediction must be done without indication about the (unknown) underlying process; input values to the prediction method must thus be chosen by trial and error. In some situations, a priori information can be fed into the prediction method, but this remains an exception: as an example, weekly and monthly past values are obviously good candidates to predict the electrical load.

In most situations however information about the underlying process is hardly available. The selection of a *regressor*, i.e. a vector of past values that will be used as input to the forecasting algorithm, is thus a difficult task. On one hand, it is interesting to have a large vector, containing as much information as possible. The idea is that the forecasting algorithm will itself choose what the key features are to take into account from this vector. On the other hand, high-dimensional input vectors lead to a high-dimensional parameter space, with all problems and limitations detailed in the first chapter of this work. A compromise is thus necessary for optimal performances.

Forecasting with nonlinear methods is then usually achieved through one of the two following methods:

1. Linear prediction tools (for example ARX) are built on the same problem and data. In the linear case, methods exist to choose the optimal regressor. Optimal means here that the selection will lead to the best performances, under the assumption that the forecasting model is linear. Of course, better forecastings will hopefully be obtained with a nonlinear model. Moreover, there is no guarantee that the optimal regressor chosen in the linear case will lead to optimal performances when a nonlinear forecasting model will be used. On the contrary, it will be seen in the examples below that nonlinear forecasting algorithms will usually need smaller input vector than linear methods. Nevertheless, choosing the optimal (in the linear case) regressor to be used in the nonlinear algorithm is a good starting point.

2. Trials and errors (usually coupled to cross-validation) are used to evaluate the optimal regressor with a nonlinear forecasting algorithm. Nevertheless, it should be noted that this method is particularly computationally intensive. Not only a large number of simulations must be carried out with different input vectors, but it must be reminded that any simulation involving nonlinear optimisation must itself be repeated several times, to compensate the risks for falling in local minima, etc. Furthermore, choosing the regressor does not only mean to choose the auto-regressive order, but rather each of the components of the vector: even if a p-dimensional regressor is sufficient, maybe a better choice than the p last values of the series could be done.

In the following, we will concentrate our investigations on auto-regressive models: input values to the forecasting algorithm are restricted to past values of the series. In real problems, so-called *exogenous* variables are used in addition to past values. For example, weather variables are used in electrical load prediction, and tendencies from other Stock Exchanges are used to predict the daily fluctuation of a specific index. Our discussion below can be easily extended to situations where exogenous variables are used. Nevertheless, extension of the theorem used below, and experiments, remain a topic for further work.

The following paragraphs describe an original method used to choose the variables that will feed a nonlinear forecasting algorithm. It uses nonlinear tools, avoiding the first above drawback, and starts from the variables instead of checking the results a posteriori, decreasing the computational load.

We will not address the problem of forecasting itself. We will concentrate on the selection or the construction of the input vector to the forecasting method. We will make the assumption that these two operations do not depend on one another: the first step is to build the shortest vector containing the most information for *a* further forecasting, and the second step consists in using this vector in the best possible way. For our experiments, we will use standard MLP or RBFN (nonlinear)

networks for the prediction.

4.4.2 Input variable selection

Most prediction methods use variable selection. In the case of auto-regressive models, input variables (to the prediction method) are selected among all past values of the series, according to some criteria. For example, forecasting of the hourly electrical load in a region or country (at time $t + 1$) usually requires the values of the load at times t, $t - 1$, ... , $t - p$, but also the values at times $t - 7$, $t - 365$, etc. since these last values have a probably stronger influence on the prediction than other randomly selected ones.

Variable selection is the process of finding the best past values in the series. Let us now imagine a hypothetic series where some past values are linearly correlated, for example that

$$x(t - p_1) = kx(t - p_2). \tag{67}$$

where k is some constant. In this case, either $x(t - p_1)$ or $x(t - p_2)$ can be used in a linear auto-regressive prediction model; the form

$$x(t+1) = w_0 x(t) + w_1 w(t-1) + \cdots + w_{p_1} x(t - p_1) + \cdots \tag{68}$$

is strictly equivalent to the form

$$x(t+1) = w_0 x(t) + w_1 x(t-1) + \cdots + w_{p_2} x(t - p_2) + \cdots; \tag{69}$$

the only difference is that the parameters multiplying $x(t - p_1)$ or $x(t - p_2)$ in both expressions will be adjusted in such a way that

$$w_{p_2} = k w_{p_1}. \tag{70}$$

This adjustment will occur automatically in the fitting process of parameters w. Note that we made the assumption that the term corresponding to time $t - p_1$ is null in equation (69) and the same for the term corresponding to time $t - p_2$ in equation (68).

In linear context, one could think to use PCA to select variables. PCA performed on a set of past values including $x(t - p_1)$ and $x(t - p_2)$ will lead to a reduced set containing *one* multiple of either $x(t - p_1)$ or $x(t - p_2)$. Regardless of the multiplying coefficient, which in any case will be learned, or fitted, in the same way

as in equation (70), this means that PCA will perform the selection of variables between $x(t-p_1)$ and $x(t-p_2)$. Of course, the same argument could be developed when variables (past values) are linear combination of several other past values, since PCA will capture the linear relationship between any set of variables.

To conclude the discussion about the linear case, we have to mention that using PCA to reduce the size of the auto-regressive vector will have little effect. Indeed, if two past values are correlated as in the example above, keeping the two terms in the prediction model will *not* increase the effective number of degrees of freedom of the model (since the two terms are multiple one from the other). Using fewer variables after PCA is thus only a formal change in the model, but without effect on the bias-variance trade-off. Nevertheless, even if the dimension reduction performed by the PCA will have no effect in this context, PCA will have a beneficial effect in the reduction of the noise of the series, as detailed in the introductory part of this chapter.

Note that the use of the terms "variable selection" is not strictly correct when PCA is used. Indeed PCA will project the initial values, so that the resulting variables are no longer *selected from* the initial set, but are *linear combinations of* the initial values. However, a linear prediction tool used either on the initial values or on the result of the PCA will compensate for PCA's linear combinations.

In real applications however, relations between past values, if any, will be nonlinear in general. PCA will not capture these relations, and thus will be unable to reduce the size of the auto-regressive vector. We remind once again that reducing this size is of utmost importance for the quality of the prediction, as soon as a nonlinear prediction model is used. There is thus a need for nonlinear dimension reduction as explained in this chapter. In the following, we will mainly use CCA as a way to project an initial regressor on a reduced one. The initial regressor will be chosen so that it is deemed to contain all the information necessary for an optimal forecasting. Then this vector will be projected on a smaller one, whose (minimal) dimension must be determined. Choosing adequate sizes for the initial and for the projected regressors will be discussed from the viewpoint of Taken's theorem as detailed below.

In the following, we present two examples of time-series forecasting, using nonlinear dimension reduction as preprocessing tool applied to the auto-regressive vector. These two examples are variations around the same theme, but differ in the implementation details (choice of the nonlinear dimension reduction method, evaluation of the intrinsic dimension of the auto-regressive vector, choice of the forecasting model used after dimension reduction, etc.) [Verleysen 99], [Lendasse 00], [Lendasse 00-2], [Lendasse 00-3]. Setting up a robust (insensitive to parameters) method valid in a large spectrum of situations is still a research topic. For this reason, we sacrifice our principle to refrain from examples. The validation of the proposed method will not be possible in a theoretical way, as all possible improvements in the prediction will result from a (positive) balance between advantages (in the prediction, because of a smaller input vector) and the

drawbacks (the loss of information in the dimension reduction process). Our point here is to show that preliminary studies will at least not decrease the performances. Furthermore, even at equal performances, the fact that the method is quite straightforward (no parameter tuning or sensitive choice) is viewed as an advantage. For the same reasons, we present the two examples below without attempt to compare the respective methodologies and implementation details. Future work will include the design of a unique methodology to perform time-series forecasting with nonlinear dimension reduction of the auto-regressive vector.

Before presenting the examples, we have to mention a central theorem related to the choice of the autoregressive vector.

4.4.3 Taken's theorem

An important question when nonlinear dimension reduction is considered, is to know the dimension of the space on which an initial regressor may be projected without loss of information.

A first idea would be to consider the regressors as high-dimensional vectors, and to extract the intrinsic dimension of the set of regressors, as explained in the first chapter. As the intrinsic dimension is the "true" dimension of the set of vectors (or, more precisely, of the submanifold containing the vectors), one could think that it will be possible to project on a subspace of the same dimension.

Indeed this is possible, for example using the CCA method. Unfortunately, the projection can be non-bijective, as illustrated in the example below. This is a strong limitation in our case. Indeed, our idea to project the initial regressors on smaller ones implicitly assumes that the same information is contained in both vectors (or both sets of vectors). Not loosing information in the projection means that the initial vectors could be retrieved from the projected ones (i.e. that the projection may be inverted). Unfortunately, projecting on a space of dimension p, where p is the intrinsic dimension of the set of initial regressors, will not guarantee this condition. Let us examine for example the artificially generated series illustrated in Figure 33. Figure 34 shows (left) the set of points in a two-dimensional space, formed by the value of any sample in the series as a function of the preceding sample, and (right) the set of points in a three-dimensional space, formed by the value of any sample in the series as a function of the two preceding samples. In both figures, the set of points forms a line, leading to the fact that the intrinsic dimension of the series is one. Note that the thickness of the lines is due to noise added when the series has been generated.

Nevertheless, knowing the value of x_{t-1} is not sufficient to predict the next value x_t. The set of points in Figure 34 (left) is clearly not a function: between two to four x_t values may correspond to the same x_{t-1}. The situation is improved in Figure 34 (right), but it is difficult to see if the illustrated relation $x_t = f(x_{t-1}, x_{t-2})$ is now a

function or not.

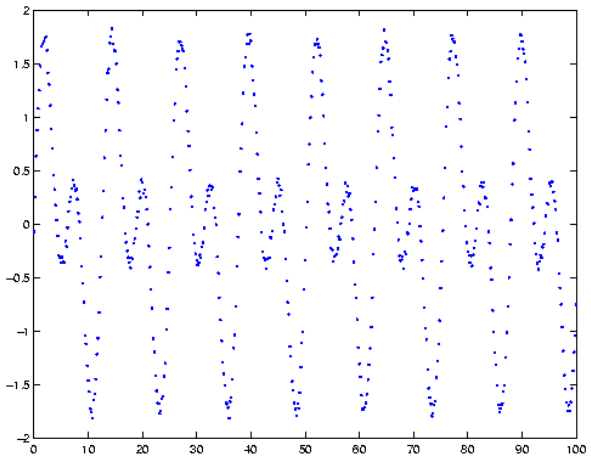

Figure 33: Artificial series to illustrate Taken's theorem

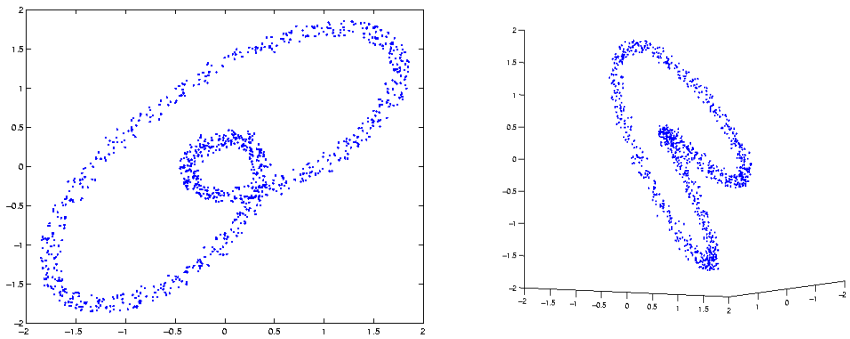

Figure 34: State diagram of the series illustrated in Figure 33. Left: y_t as a function of y_{t-1}. Right: y_t as a function of y_{t-1} and y_{t-2}.

Intuitively, it seems thus that, in some specific cases, one should project on a subspace of dimension greater than p (the intrinsic dimension) in order to maintain the projection bijective, i.e. to keep all the information in a set of regressors. Taken's theorem [Takens 85, Camastra 99] makes this intuition more explicit. Without going into the mathematical details that can be found in the references, Taken's theorem states the following.

Let us imagine a time series $x(t)$. The series must be sufficiently long and stationary to make some kind of prediction possible. By stationary, we mean here constant mean, variance and auto-correlation coefficients. Regressors are built based on this series. If the series contains N values, and if the size chosen for the regressors is d (d last values of the series), it will be possible to construct $N-d$ regressors associated to the next value in the series. It is assumed that size d is chosen sufficiently large to contain all the information necessary for a "good" prediction. If the series is known from value $x(1)$ to value $x(N)$, the set of regressors will be

$$R = \begin{pmatrix} x(N-d) & x(N-d+1) & \cdots & x(N-2) & x(N-1) \\ x(N-d-1) & x(N-d) & \cdots & x(N-3) & x(N-2) \\ \vdots & \vdots & & \vdots & \vdots \\ x(1) & x(2) & \cdots & x(d-1) & x(d) \end{pmatrix}. \quad (71)$$

Note that value $x(N)$ is not used here in the set of regressors, since it will be used as output value (target) during learning of the forecasting function, for the first regressor in the matrix (first line).

Then, the intrinsic dimension of the set of $N-d$ d-dimensional regressors is computed. Let q be this intrinsic dimension, with $q < d$.

Taken's theorem states that it is sufficient to use a size of regressor between q and $2q+1$ for the prediction, without loss of information (regardless of the way how the prediction is performed).

Two conditions must be checked to validate the use of Taken's theorem:

1. If the value found for $2q+1$ is greater than d, this means that the initial value d has not been taken sufficiently large. In this case, a greater value must be chosen and the whole procedure must be repeated.

2. In any case, the procedure should be repeated with at least a second value for d, greater than the initial one. The intrinsic dimension q found in both cases must be equal; otherwise, the procedure should again be repeated with a greater value for d.

Taken's theorem will be used in our procedure for time-series forecasting: we will start from regressors of size $2q+1$ instead of the intuitive size q. The reason for this has been made clear with the example in Figure 34.

The geometrical interpretation of Taken's theorem is that there exists a space of dimension between q and $2q+1$, where the parametric representation of a q-

surface is bijective with respect to the coordinates of the space. In other words, the process underlying the time series has an intrinsic dimension equal to q, but the q-surface could have intersection points if drawn in a space whose dimension is too small. The smallest dimension of the space in which it is guaranteed that the q-surface will not intersect itself is $2q + 1$. It is thus necessary to use regressors of dimension $2q + 1$ to know the current position of the process in the state space.

Despite these comments, and since the intrinsic dimension of the set of regressors is q, it is possible to project the regressors from dimension $2q + 1$ to q-dimensional vectors, the projection being bijective. This is a direct consequence of the intrinsic dimension concept. We will use CCA (or other projection methods) for this task. It should be clear that the vectors before projection are regressors (they contain past values of the initial series), while the vectors after projection aren't regressors anymore: they do not contain past values of the series, but a (lower dimensional) parametric representation of these values. This is in the spirit of a *state vector*. Nevertheless, as CCA is used as preprocessing to the forecasting method, we will continue to use the term *regressor* whether the initial regressor has been reduced or not.

4.4.4 Example 1

The first example used to illustrate the method is an artificial time series built from the nonlinear equation

$$x(t+1) = ax(t)^2 + bx(t-2) + \varepsilon . \tag{72}$$

where a and b are constants and ε stands for noise.

Obviously, the nonlinear regressor order of this time series is 2 (it is generated from 2 past values). Let us note the absence of a $x(t-1)$ term, as well as the presence of a noise ε (about 10% of the maximum value of the series). The series does not contain any exogenous variable, and is shown in Figure 35.

We begin by looking at the results of a forecasting by a linear (auto-regressive) model on this series. Figure 36 shows the sum (on 1000 points) of the quadratic errors obtained with a linear AR model of increasing order. Obviously, the error decreases with the order. However, it is also evident from Figure 36 that a linear model cannot capture the nonlinear dynamics of the series with a low regressive order (although only two past values are used to build the series).

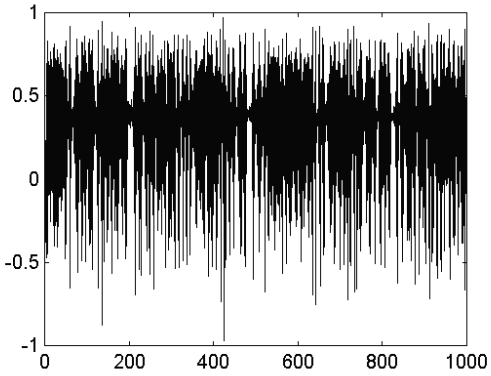

Figure 35: artificial time series generated by equation (72).

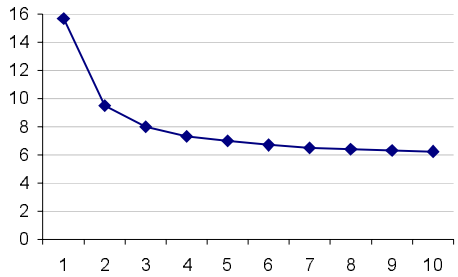

Figure 36: Sum of quadratic errors (on 1000 test points) obtained with an AR model for different values of the regressor order.

We then proceed by using the nonlinear methodology described above. To ensure that the whole dynamics of the series is collected, we build an initial regressor matrix (equation 71) of order 6. The Grassberger-Procaccia algorithm to estimate the intrinsic dimension of the series (i.e. of the set of regressors) gives 2.12, which is close to the exact value 2. Note that the noise ε added to the series inevitably increases the intrinsic dimension.

The following step of the method is the projection of the set of the points (rows of the regressor matrix) from R^6 to R^2 by CCA. The dimension of the final regressor vector is thus 2. Note that Taken's theorem informs us that choosing 5 as initial dimension would have been sufficient.

In a next step we use this 2-dimensional regressor as input to a nonlinear prediction model. As an example, we use a Multi-Layer Perceptron with one hidden layer and five hidden units. The sum of quadratic errors obtained with this

MLP is around 5 (on 1000 points), which is significantly lower than the errors illustrated in Figure 34 (linear model).

We also compare this result to the error obtained with a similar Multi-Layer Perceptron, where the input vector is the set of *d* last values from the raw series. Figure 37 shows this error as a function of *d*. The horizontal dotted line corresponds to the error obtained with our method, for comparison; we conclude that we obtain (for this example) an error similar to a result obtained by trial and error on several nonlinear (MLP) models, which was the goal of our investigation. This ease of implementation will be valuable when dealing with a "real-size" data set for which the nonlinear regressor order is unknown.

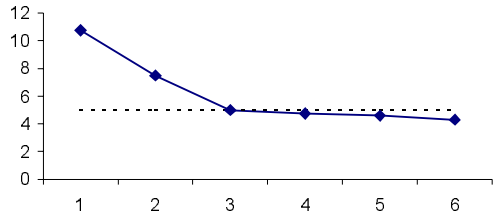

Figure 37: Sum of quadratic errors (on 1000 points) obtained with a MLP network for different values of the regressor order. The dotted horizontal line corresponds to the result of the proposed method for comparison.

4.4.5 Example 2

An interesting example of time series in the field of finance is the Belgian Bel 20 index. The application of time series forecasting to financial market data is a real challenge. The efficient market hypothesis (EMH) remains up to now the most generally admitted one in the academic community, while essentially challenged by the practitioners. Under EMH, one of the classical econometric tools used to model the behaviour of stock market prices is the geometric Brownian motion. If it does represent the true generating process of stock returns, the best prediction that we can obtain of the future value is the actual one (they follow a random walk). Results presented in this section must therefore be analysed with a lot of caution.

To succeed in determining the variations of the Bel 20 index, other variables that could have influence on the index are included as inputs (exogenous variables). We selected international indices of security prices (SBF 250, S&P500, Topix, FTSE100, etc), exchange rates (Dollar/Mark, Dollar/Yen, etc), and interest rates (T-Bills 3 months, US Treasury Constant Maturity 10 years, etc).

We used 2600 daily data of the BEL 20 index over 10 years to have a significant data set. The problem considered here is to forecast the sign of the variation of the Bel 20 index at time $t+5$, from available data at time t.

According to Refenes et al. [Refenes 97] and Burgess [Burgess 95], we use 42 technical indicators directly resulting from the inputs and the exogenous variables, for example:

- x_t , x_{t-10}, $x_{t-20,}$ x_{t-40}, ..., y_t , y_{t-10}, ...: returns ;
- $x_t - x_{t-5}$, $x_{t-5} - x_{t-10}$, ..., $y_t - y_{t-5}$, ...: differences of returns ;
- K(20), K(40), ... : oscillators ;
- MM(10), MM(50), ... : moving averages ;
- MME(10), MME(50), ... : exponential moving averages ;
- etc.

If we carry out a Principal Component Analysis (PCA) on these 42 variables, we note that 95% of the original variance is kept with the first 25 principal components: 17 variables can be removed without significant loss of information. The PCA is used to facilitate the subsequent processing by the CCA algorithm (lower computational load and better convergence properties).

The time series of the target variable, x_{t+5}, whose sign has to be predicted, is illustrated in Figure 38.

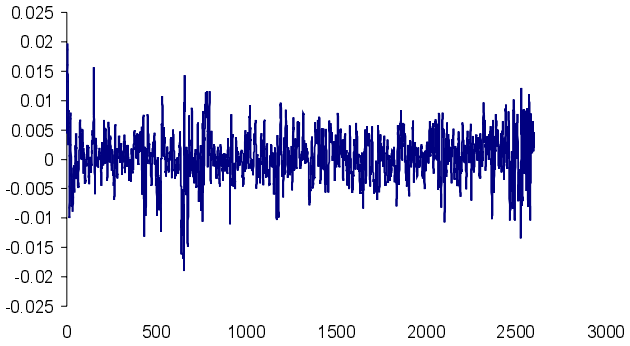

Figure 38: Daily values of the BEL20 index.

This variable has to be predicted using the resulting 25 variables selected after PCA. The interpolator we use is a Radial-Basis Function (RBFN) network as described in chapter 3. Our interest goes to the sign of the prediction only, which will be compared to the real sign of the target variable.

The Grassberger-Proccacia method is used to estimate the intrinsic dimension q of the data set; we obtain an approximate value of 9. We then use the CCA algorithm to project the 25-dimensional data (after PCA) on a 9-dimensional space. The RBFN interpolator is used on the resulting 9-dimensional input vectors.

Note that Takens' theorem does not apply here, since we have exogeneous variables. Nevertheless, we use the same idea as behind this theorem. We start from a set of sufficiently large vectors containing the information, and we project them on vectors whose dimension is chosen according to the intrinsic dimension of the initial set.

The network is trained with a moving window of 500 data. Each of these data consists in a 9-dimensional input vector (see above) and a scalar target (variation of the BEL 20 index). We use 500 data as a compromise between

- a small stationary set but insufficient for a successful training, and
- a large but less stationary training set.

For each window, the 500 input-target pairs form the training set, while the test set consists in the input-target pair right after the training set. This procedure is repeated for 2100 moving windows. On average, we obtain 60,3% correct approximations of the sign of the series on the training sets, and 57.2% on the test sets.

These results are encouraging. Moreover, it can be seen that better results are obtained during some periods and worse results during others. The first ones correspond to time periods where the series is more stationary than the last ones. Figure 39 represents a moving average on 90 days on the results of the prediction. It clearly shows that that the prediction results themselves do not form a random series: when the forecasting is correct over several consecutive days, the probability that it will be correct at the next time step is high.

To quantify this idea, we filter the results with the following heuristics. We look at the average of sign predictions (correct – not correct) over the last 5 days. If this average increases or remains constant at time t, then we take the forecasting at time $t+1$ into consideration. If it decreases, then we disregard the forecasting at time $t+1$. With this method, we keep 75.4% of the forecasts; the average score of correct prediction rises to 65.3% (about 70% of increases and 60% of decreases). This way of working is a first attempt to use our mathematical procedure in a real-

world financial context.

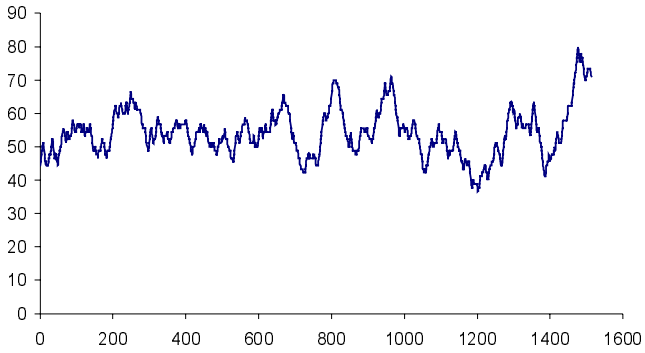

Figure 39: Percentage of correct approximations of the sign on a 90-days moving window.

4.5 Conclusion and further work

The objective of this chapter is to bypass the difficult problem of learning high-dimensional data, by projecting the data on lower dimensional spaces. Traditional PCA may be used, either on the whole dataset either locally, or more advanced nonlinear projection methods. We chose to detail the promising Curvilinear Component Analysis algorithm, and we added a few improvements to this method in order to make it more robust for a wide range of applications.

The main original contribution of this chapter is the application of the concept of nonlinear projection to the problem of forecasting time series. Taken's theorem gives us a theoretical justification of the possibility to predict time series with a limited number of past values. Although Taken's theorem does not apply stricto sensu to problems with exogeneous variables, we use the idea behind the theorem to project large-dimensional input vectors on smaller ones, in order to improve the prediction.

The results presented on two examples are preliminary. We are convinced that the idea could be successfully used in complex prediction problems, like those encountered in finance. Nevertheless, some progress has still to be made in the robustness of the projection algorithm. Moreoever, the applicability of Taken's theorem to problems with exogeneous variables should still be invectigated too. Finally, the examples presented in this chapter use standard prediction methods after the projection procedure; a better coupling between the projection and the prediction should still be investigated.

Chapter 5

Discussion

In this work, we tried to describe problems and limitations related to the concept of "learning from data", when high-dimensional data are involved. Working with high-dimensional data is not a mathematical or theoretical question without consequence in practical situations. On the contrary, most data analysis problems encountered in real world applications explicitly deal with high-dimensional data. Indeed high-dimensional related problems already arise in dimensions as low as 4 or 5!

Artificial neural networks have been "invented" to solve problems where other more traditional data analysis tools fail. Since artificial neural networks can effectively outperform other methods in specific situations, it has been argued that they solve all problems, including those related to high dimensions! This is obviously not true, even if this work shows methods to deal with such problems, probably more effectively than conventional data analysis tools.

All data analysis tools have difficulties to work with high-dimensional data. Generally speaking, this is not due to limitations of the methods themselves, but rather on the intrinsic nature of high-dimensional data. Some examples were given in Chapter 2 to show that basic concepts, such as the (Euclidean) distance between vectors, lose their meaning when the dimension of the space increases. It is thus not surprising that standard analysis tools based on these concepts will fail.

In particular, two models for approximation of functions widely known in the neural network community, the multi-layer perceptron (MLP) and radial basis function networks (RBFN), do not perform identically with low- or high-dimensional data. But the same problem exists with other methods, such as polynomials. Actually, things are much worse with polynomials than with MLP or RBFN! Contrary to some generally accepted ideas, we are convinced that RBFN networks, and more generally models based on *local* approximation of data, are at least equally (and probably better) suited to high-dimensional problems than global methods such as MLP. The advantages of learning with local models are thus critical for the choice of an approximation method.

It remains that learning high-dimensional data is not an easy task and usually results in compromises. Statisticians and mathematicians call these compromises the *bias-variance dilemma*, or *bias-variance tradeoff* (which is probably more appropriate). The bias-variance tradeoff is detailed in Chapter 2. Shortly, it can be viewed as the fact that the better a finite set of data (with noise) is approximated by a function, the less useful this function is, because it fits the noise rather than an underlying distribution of data. Of course, intentionally degrading the approximation of the dataset to get rid of noise fitting is not a solution (at least not an easy one)!

Most criticism over "neural network" methods over the last ten years is a consequence of the underestimation of the bias-variance tradeoff by many authors. It is not unusual to find articles in the literature, where authors use more parameters in their model than the number of samples available for learning; obtaining "good" learning performances is not difficult in these conditions! Realistic performances, i.e. performances estimated in generalization, are however much worse...

In this work, we tried to build methods that are *a priori* better suited than other ones, for high-dimensional problems. This is the topic of Chapter 3. In most cases, there is no *proof* that any of the proposed method will perform better than any other one, in real high-dimensional problems. Everyone who ever worked with high-dimensional nonlinear optimization methods knows that it is always possible to find a problem that will *prove* the superiority of a method over the others, whatever the method... For this reason, we avoided as much as possible to base our arguments on examples. Our view is thus more guided by intuition, theoretical proofs being hardly feasible in this context.

The second main direction of this work is a direct consequence of the bias-variance tradeoff. Since a large part of the problem is based on the *relative* number of parameters in a model compared to the number of samples available for learning, the idea is to work in a space whose dimension is as low as possible.

Most real-world data are high-dimensional because they are *represented* in a high-dimensional space. Indeed each dimension corresponds to a *feature* of the data, which is often a physical quantity measured by a sensor in an engineering context, or an indicator on the data in more mathematical framework. This does not mean that each feature is necessary (for further approximation or classification), nor that the features are independent one from another! One could take profit from this redundancy or from the lack of usefulness of some information to reduce the dimension of the feature space (keeping the same number of learning data), in order to make the bias-variance tradeoff moving towards to right side. The justification of this approach resides in the concept of intrinsic dimensionality of data, which is usually lower than the representation dimension. This is the topic of Chapter 4, where nonlinear dimension reduction tools are considered.

Using tools that are better suited to high-dimensional data, and reducing the dimension of data in real applications, does not mean that the curse of dimensionality problem is solved. It must be accepted that this problem remains effective, but that the two axes of this work are attempts to reduce it.

If the tradeoff cannot be entirely eliminated, a framework must be set up, where *a posteriori* tests must be able to measure the results of learning, in an objective way. Again, learning results are of no interest if they do not reflect the *generalization* ability of a method, i.e. a measure of its performance when it is used with *new* data not used during learning.

Regarding the fact that the number of data available in a real application is always too low (with respect to the dimension of the space), it is inefficient to partition the dataset into a learning and a test set, the first one being used to fit the model, and the second one to test it. Using only a part of the dataset for learning will further decrease the performances of the method. On the other hand, using the same data for learning and test may lead to erroneous conclusions concerning the generalization ability of the method.

Cross-validation [Stone 74] must be used to remedy to this problem. Shortly, the dataset is again split into two parts, one used for learning and the other for test, but this procedure is repeated for several (many) different partitions. Averaging the results leads to a reasonable estimate of the performances of the method applied to the whole dataset.

The ultimate version of cross-validation is the leave-one-out procedure, where only one data is used for test at each iteration. Using almost all data for learning gives an average performance that is close to the true performance of the method. Nevertheless, it must be mentioned that, unlike most linear data analysis tools, the learning must be entirely repeated for each partition of the dataset. (With linear methods, it is usually possible to set up an incremental learning procedure where the result of the learning with one training set is updated to reflect the change in one data of the learning set; this is not possible in general with nonlinear methods). It must be mentioned that leave-one-out applied to nonlinear optimization algorithms often implies a huge computing load.

Other ways not considered here, to improve the bias-variance dilemma, are related to pruning and/or regularization.

Pruning consists in the design of large networks (or methods with many parameters), thus overfitting the data, and then reducing the complexity of the network by removing connections (or parameters) based on a statistical test on the learning data. Shortly, if a parameter has no or little influence on the interpolation performance, it can be removed to improve the bias-variance tradeoff. Pruning is an efficient method implemented in many commercial and freeware softwares, but sometimes suffers from the lack of an objective stopping criterion. [Cottrell 95] is

to our knowledge the only existing attempt to relate the pruning stopping condition to an objective statistical criterion.

Another possible direction to improve the bias-variance tradeoff is to include a penalty term in the function to optimize. This term is used for regularization, i.e. to avoid large parameters, or parameters having (alone) too much influence on the interpolation results. The purpose is to make the approximation function smoother, without changing the number of parameters. The recent Support Vector Machines (SVM) theory is based on these concepts (see [Campbell 00] for a review on SVM learning).

All these ways to improve the bias-variance tradeoff (and to improve learning with neural networks in realistic problems) should not make forget that learning complex, highly nonlinear data remains difficult. While tremendous progress has been made in data analysis tools in the last decade, the design of robust methods, aimed to be used in a large class of applications, without extensive a priori knowledge about the data, remains an exciting challenge for further research.

References

[Amerijckx 98] C. Amerijckx, M. Verleysen, P. Thissen, J.D. Legat, "Image compression by self-organized Kohonen map", *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 503-507, May 1998.

[Blayo 92] F. Blayo, M. Verleysen, "Setting initial conditions for the RCE model", in *Proceedings of the 1st IFIP Working Group 10.6 Workshop*, Grenoble (France), March 1992, pp.31-35.

[Broomhead 88] D.S. Broomhead, D. Lowe, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, no. 2, pp. 321-355, 1988.

[Burgess 95] A.N. Burgess, "Nonlinear model identification and statistical significance tests and their application in financial modelling", in *Artificial Neural Networks*, Proceedings of the Inst. Elect. Eng. Conf., 1995.

[Cacoullos 66] T. Cacoullos, "Estimation of a multivariate density", *Annals of Inst. Stat. Math.*, vol. 18, pp. 178-189, 1966.

[Choppin 98] A. Choppin, *Unsupervised classification of high dimensional data by means of self-organizing neural networks*. M.Sc. thesis, Université catholique de Louvain, Computer Science Dept., June 1998.

[Camastra 99] F. Camastra, A.M. Colla, "Neural short-term prediction based on dynamics reconstruction", *Neural Processing Letters*, vol. 9, no. 1, pp. 45-52, February 1999.

[Campbell 00] C. Campbell, "Algorithmic approaches to training Support Vector Machines: a survey", in *ESANN'2000 Proceedings, European Symposium on Artificial Neural Networks*, Bruges (Belgium), April 2000, to be published, D-Facto publications (Brussels).

[Comon 92] P. Como, G. Bienvenu, T. Lefebvre, "Supervised design of optimal receivers", in *NATO Advanced Study Institute on Acoustic Signal Processing and Ocean Exploration*, Madeira (Portugal), July-August 1992.

[Comon 94] P. Comon, J.-L. Voz, M. Verleysen, "Estimation of performance bounds in supervised classification", in *ESANN'94 Proceedings, European Symposium on Artificial Neural Networks*, Brussels (Belgium), April 1994, pp. 37-42, D-Facto publications (Brussels).

[Comon 95] P. Comon, "Supervised classification: a probabilistic approach", in *ESANN'95 Proceedings, European Symposium on Artificial Neural Networks*, Brussels (Belgium), April 1995, pp. 111-128, D-Facto publications (Brussels).

[Cottrell 95] M. Cottrell, B. Girard, Y. Girard, M. Mangeas, C. Muller, "Neural modeling for time series: a statistical stepwise method for weight elimination", *IEEE Trans. on Neural Networks*, vol. 6, no. 6, pp. 1355-1364, 1995.

[de Bodt 99] E. de Bodt, M. Cottrell, M. Verleysen, "Using the Kohonen algorithm for quick initialization of simple competitive learning algorithm", in *ESANN'99 Proceedings, European Symposium on Artificial Neural Networks*, Bruges (Belgium), April 1999, pp. 19-26, D-Facto publications (Brussels).

[Demartines 94] P. Demartines, "Analyse de données par réseaux de neurones auto-organisés", Ph.D. dissertation, Institut National Polytechnique de Grenoble, France, 1994.

[Demartines 97] P. Demartines, J. Hérault, "Curvilinear Component Analysis: a self-organizing neural network for nonlinear mapping of data sets", *IEEE Trans. on Neural Networks*, vol. 8, no.1, pp. 148-154, January 1997.

[Dittenbach 00] M. Dittenbach, D. Merkl, A. Rauber, "Using growing hierarchical Self-Organizing Maps for document classification", in *ESANN'2000 Proceedings, European Symposium on Artificial Neural Networks*, Bruges (Belgium), April 2000, pp. 7-12, D-Facto publications (Brussels).

[Donckers 99] N. Donckers, A. Lendasse, V. Wertz, M. Verleysen, "Extraction of intrinsic dimension using CCA – Application to blind sources separation", in *ESANN'99 Proceedings, European Symposium on Artificial Neural Networks*, Bruges (Belgium), April 1999, pp. 339-344, D-Facto publications (Brussels).

[Fort 00] J.C. Fort, G. Pagès, "Asymptotics of optimal quantizers for some scalar distributions", submitted to *Journal of Applied Probability*, 2000.

[Fritzke 91] B. Fritzke, "Let it grow – self organizing feature maps with problem dependent cell structure", in *Artificial neural networks*, vol. 1, T. Kohonen, K. Mäkisara, O. Simula, J. Kangas eds., *ICANN'91 Proceedings, International Conference on Artificial Neural Networks*, Helsinki (Finland), June 1991, pp. 403-408, North-Holland (Amsterdam), 1991.

[Fukunaga 89] K. Fukunaga, R.R. Hayes, "The reduced Parzen classifier", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 4, pp. 423-425, April 1989.

[Gersho 79] A. Gersho, "Asymptotically optimal block quantization", *IEEE Trans. Information Theory*, vol. 25, pp. 373-380, 1979.

[Gersho 92] A. Gersho, R.M. Gray, *Vector quantization and signal processing*. Kluwer Academic Publishers (Boston – Dordrecht – London), 1992.

[Gonzales 77] R.C. Gonzales, P. Wintz, *Digital image processing*. Addison-Wesley Publishing Company (Reading, MA), 1977 (2nd edition 1979).

[Gonzales 97] A.I. Gonzales, M. Graña, A. D'Anjou, F.X. Albizuri, M. Cottrell, "A sensitivity analysis of the self-organizing maps as an adaptice one-pass non-stationary clustering algorithm: the case of color quantization of image sequences", *Neural Processing Letters*, vol. 6, no. 3, pp. 77-89, December 1997.

[Graf 00] S. Graf, H. Luschgy, *Foundations of quantization of probability distributions*. Spirnger-Verlag, Lecture Notes in Mathematics 1730, 2000.

[Grassberger 83] P. Grassberger, I. Procaccia, "Measuring the strangeness of strange attractors", *Physica D*, vol. 56, pp. 189-208, 1983.

[Guérin-Dugué 99] Guérin-Dugué A., Teissier P., Delso Gafaro G., Hérault J., "Curvilinear Component Analysis for high dimensional data representation: II. Examples of additional mapping constraints in specific applications", in *Engineering Applications of Bio-Inspired Artificial Neural Networks*, J. Mira, J.V. Sanchez-Andres eds., *IWANN'99 Proceedings, International Work-Conference on Artificial and Natural Neural Networks*, Alicante (Spain), June 1999, pp. 635-644, Springer-Verlag, Lecture Notes in Computer Science 1607.

[Hentschel 83] H. G. E. Hentschel, I. Procaccia, "The infinite number of generalized dimensions of fractals and strange attractors", *Physica 8D*, pp. 435-444, 1983.

[Hertz 91] J. Hertz, A. Krogh, R. Palmer, *Introduction to the theory of neural computation*. Santa Fe Institute, Addison-Wesley (Redwood City, CA), 1991.

[Hlavackova 97] K. Hlavackova, M. Verleysen, "Synthesis of neural networks using splines for approximation of functions", *Neurocomputing*, vol. 17, nos. 3-4, pp. 159-167, 1997.

[Kambhatla 97] N. Kambhatla, T.K. Leen, "Dimension reduction by local principal component analysis", *Neural Computation*, vol. 9, no. 7, pp. 1493-1516, October 1997.

[Karhunen 99] J. Karhunen, S. Malaroiu, "Local independent component analysis using clustering", in *ICA'99 Proceedings, First International Workshop on Independent Component Analysis and Signal Separation*, Aussois (France), January 1999, pp. 43-48.

[Kohonen 88] T. Kohonen, *Self-organization and Associative Memory*. Springer Series in Information Sciences, Vol. 8, 2nd edition, Springer (Berlin), 1988.

[Kohonen 95] T. Kohonen, *Self-organising Maps*. Springer Series in Information Sciences, Vol. 30, Springer (Berlin), 1995.

[Kohonen 98] T. Kohonen, "Comparison of SOM point densities based on different criteria", *Neural Computation*, vol. 11, no. 8, pp. 2081-2095, November 1999.

[Lee 99] J. Lee, B. van Hout, *Analyse de données non linéaires par réseaux de neurones artificiels (cartes auto-organisatrices)*. M.Sc. thesis, Université catholique de Louvain, Computer Science Dept., June 1999.

[Lee 00] J. Lee, A. Lendasse, N. Donckers, M. Verleysen, "A robust nonlinear projection method", in *ESANN'2000 Proceedings, European Symposium on Artificial Neural Networks*, Bruges (Belgium), April 2000, pp. 13-20, D-Facto publications (Brussels).

[Lendasse 00] A. Lendasse, E. de Bodt, V. Wertz, M. Verleysen, "Non-linear time series forecasting – Application to the Bel 20 stock market index", *European Journal of Economic and Social Systems*, vol. 14, no. 1, pp. 81-92, 2000.

[Lendasse 00-2] A. Lendasse, J. Lee, V. Wertz, M. Verleysen, "Time series forecasting using CCA and Kohonen maps – Application to electricity consumption", in *ESANN'2000 Proceedings, European Symposium on Artificial Neural Networks*, Bruges (Belgium), April 2000, pp. 329-334, D-Facto publications (Brussels).

[Lendasse 00-3] A. Lendasse, J. Lee, E. de Bodt, V. Wertz, M. Verleysen, "Dimension reduction of technical indicators for the prediction of financial time series – Application to the Bel 20 stock market index", submitted to *IEEE Transactions on Neural Networks, special issue on Financial Analysis*.

[Moody 89] J. Moody, C. Darken, "Learning with localized receptive fields", in *Proceedings of the 1988 Connectionist Models Summer School*, G. Hinton, T. Sejnowski eds., San Mateo (CA), Morgan Kaufmann, 1989.

[Orr 96] M.J.L. Orr, *Introduction to Radial Basis Function networks*. Technical report, University of Edinburgh, Centre for Cognitive Science. Available from http://www.anc.ed.ac.uk/~mjo/papers/intro.ps.

[Pagès 97] G. Pagès, "A space quantization method for numerical integration", *Journal of Computational and Applied Mathematics*, vol. 89, pp. 1-38, 1997.

[Pearson 01] K. Pearson, "On lines and planes of closest fit to systems of points in space", *Phil. Mag.*, vol. 6, pp. 559-572.

[Poggio 90] T. Poggio, F. Girosi, "Networks for approximation and learning", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481-1497, September 1990.

[Refenes 97] A.N. Refenes, A.N. Burgess, Y. Bentz, "Neural networks in financial engineering: a study in methodology", *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1222-1267, 1997.

[Reilly 82] D. Reilly, L. Cooper, C. Elbaum, "A neural model for category learning", *Biological Cybernetics*, vol. 45, no. 1, pp. 35-41, 1982.

[Ritter 86] H. Ritter, "Asymptotic level density for a class of vector quantization processes", *IEEE Transactions on Neural Networks*, vol.2, pp. 173-175, 1991.

[Robbins 51] H. Robbins, S. Monro, "A stochastic approximation method", *Ann. Math. Stat.*, vol. 22, pp. 400-407, 1951.

[Sammon 69] J.W. Sammon, "A nonlinear mapping algorithm for data structure analysis", *IEEE Trans. on Computers*, vol. C-18, pp. 401-409, 1969.

[Shepard 62] R. N. Shepard, "The analysis of proximities: Multidimensional scaling with an unknown distance function, parts I and II", *Psychometrika*, vol. 27, pp. 125-140 and 219-246, 1962.

[Shepart 65] R.N. Shepard, J.D. Carroll, "Parametric representation of nonlinear data structures", in, *Proceedings of the International Symposium on Multivariate Analysis*, P. R. Krishnaiah, ed. pp. 561-592, Academic Press, 1965.

[Silverman 86] B.W. Silverman, *Density estimation for statistics and data analysis*. Chapman and Hall, 1986.

[Specht 90] D.F. Specht, "Probabilistic Neural Networks", *Neural Networks*, vol. 3, no. 1, pp. 109-118, 1990.

[Stone 74] M. Stone, "Cross-validatory choice and assessment of statistical predictions", *J. R. Statistical Soc. B*, vol. 36, pp. 111-147.

[Takens 85] F. Takens, "On the numerical determination of the dimension of an attractor", in *Lecture Notes in Mathematics vol. 1125*, pp. 99-106, Springer-Verlag, 1985.

[Verleysen 92] M. Verleysen, F. Blayo, J.D. Legat, “LVQ-like procedure for the initialization of the RCE model”, *in Proceedings of the Congrès Satellite du Congrès Européen de Mathématiques: Aspects Théoriques des Réseaux de Neurones*, Paris (France), July 1992, pp.35-45.

[Verleysen 93] M. Verleysen, P. Thissen, J.D. Legat, "Optimal decision surfaces in LVQ1 classification of patterns", in *ESANN'93 Proceedings, European Symposium on Artificial Neural Networks*, Brussels (Belgium), April 1993, pp. 209-214, D-Facto publications (Brussels).

[Verleysen 93-2] M. Verleysen, P. Thissen, J.D. Legat, "Learning vector classification: an improvement on LVQ algorithms to create classes of patterns", in *New Trends in Neural Computation*, J. Mira, J. Cabestany, A. Prieto eds., *IWANN'93 Proceedings, International Workshop on Artificial Neural Networks*, Sitges (Spain), June 1993, pp. 340-345, Springer-Verlag, Lecture Notes in Computer Science 686.

[Verleysen 94] M. Verleysen, K. Hlavackova, "An optimized RBFN network for approximation of functions", in *ESANN'94 Proceedings, European Symposium on Artificial Neural Networks*, Brussels (Belgium), April 1994, pp. 175-180, D-Facto publications (Brussels).

[Verleysen 96] M. Verleysen, K. Hlavackova, "Learning in RBFN networks", in *ICNN'96 Proceedings, International Conference on Neural Networks*, Washington DC (USA), June 1996, special sessions volume pp. 199-204.

[Verleysen 99] M. Verleysen, E. de Bodt, A. Lendasse, "Forecasting financial time series through intrinsic dimension estimation and non-linear data projection", in *Engineering Applications of Bio-Inspired Artificial and Natural Neural Networks*, J. Mira, J. Sanchez-Andres eds., *IWANN'99 Proceedings, International Workshop on Artificial Neural Networks*, Alicante (Spain), June 1999, pp. II596-II605, Springer-Verlag, Lecture Notes in Computer Science 1607.

[Voz 94] J.-L. Voz, P. Thissen, M. Verleysen, J.-D. Legat, " Application of suboptimal Bayesian classification to handwritten numerals recognition", in *Proceedings of the IEE European Workshop on Handwriting Analysis and Recognition: A European Perspective*, Brussels (Belgium), July 1994, pp. 9-1 – 9-8, IEE publications (London).

[Xie 93] Q. Xie, C.A. Laszlo, R.K. Ward, "Vector quantization technique for nonparametric classifier design", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 12, pp. 1326-1330, December 1993.

[Zador 82] P.L. Zador, "Asymptotic quantization error of continuous signals and the quantization dimension", *IEEE Trans. on Information Theory*, vol. IT-28, no. 2, pp. 139-149, March 1982.