State Reconstruction for Determining Predictability in Driven Nonlinear Acoustical Systems

Diploma Thesis

MEDIA LABORATORY
Massachusetts Institute of Technology
Professor Neil Gershenfeld

Institut für Elektrische Nachrichtentechnik
Rheinisch-Westphälische Technische Hochschule Aachen
Univ. Professor Dr.-Ing. H.D. Lüke

by

Bernd Schoner

May 1996

State Reconstruction for Determining Predictability in Driven Nonlinear Acoustical Systems

by
Bernd Schoner

Diploma Thesis

MEDIA LABORATORY
Massachusetts Institute of Technology
Professor Neil Gershenfeld

Institut für Elektrische Nachrichtentechnik
Rheinisch-Westphälische Technische Hochschule Aachen
Univ. Professor Dr.-Ing. H.D. Lüke

March 1996

Abstract

This thesis addresses the problem of predicting the output time series of a driven nonlinear system given access to its input. Based on phenomenological observance of the physical system the internal state vector is reconstructed using time-lagged versions of the input and the output signals. The time-lagged vector represents the effective degrees of freedom of the system and is related to the unknown physical state vector by a diffeomorphic mapping. The reconstructed state space is used for predicting output signals given future input to the simulated system.

Although the results of this thesis can be generalized to any nonlinear system –particularly nonlinear oscillating systems– we focus on the musical application of predicting (synthesizing) the audio signal of a violin, given the sensed bowing input of a violin player. With the ultimate goal of on-line synthesis in mind, special prediction experts are defined, such as a steady state predictor and a predictor of the energy level. For any one specific task the guiding issues are: the minimum space dimension that allows reliable prediction, the most appropriate approximation technique for a specific space, and the most efficient parameter search given a model framework. Through experimental results from real-world violin audio and sensor data further measures in terms of model architecture and search algorithms are derived that eventually would enable the creation of an overall prediction system.

This thesis describes the theory of state-space reconstruction for nonlinear systems, the extension to input-output systems and its applicability for musical synthesis. It presents a new formalism to describe probabilistic cluster-based function approximation and forecasting as well as the specific training algorithms. Moreover it proposes hierarchical model architectures in time and space for efficient allocation of computational resources and defines a model and training framework for the specific musical multi-scale application.

Acknowledgments

I thank my two advisers: Professor Neil Gershenfeld, for inviting me to the Media Lab, sharing his great ideas with me and for keeping me going with his never ending enthusiasm; Professor Lüke, for generously supervising my thesis for my home university at Aachen. I am grateful to Joe Paradiso for designing and tuning the violin bow, to Julia Ogrydziak for playing the violin and for her programming help, to Josh Smith for programming the HC11 and for many other tips, to Eric Metois for interesting discussions, to Vadim Gerasimov for his help with tricky Windows problems and to the entire Physics and Media Group.

Also I thank my family, my friends and all the people who helped and supported me during the last six years.

This work was supported by the TTT-consortium and by a grant from the Otto-Junker-Foundation Aachen.

Contents

1 Introduction **7**

2 Embedding and Musical Synthesis **10**
- 2.1 The Overall Concept . . . 10
- 2.2 Time-Series-Prediction . . . 12
  - 2.2.1 Linear Systems . . . 12
- 2.3 State-Space-Reconstruction of Non-Linear Systems . . . 14
- 2.4 Input/Output Embedding . . . 16
- 2.5 Characterization of Dynamic Systems . . . 18
- 2.6 Preliminary Conclusions . . . 20

3 Function Approximation in the State-Space **23**
- 3.1 Bayesian Priors and Regularization . . . 23
- 3.2 A Polynomial Approach . . . 26
- 3.3 Probabilistic Forecasting with Cluster Weighted Local Models . . . 28
  - 3.3.1 Density Approximation of the State Space . . . 29
  - 3.3.2 The prediction step . . . 32
  - 3.3.3 Learning and Search . . . 36
- 3.4 Clustering Data by Melting . . . 36
- 3.5 The EM Algorithm . . . 40
- 3.6 Empirical Tests . . . 47

4 Working with the Violin **52**
- 4.1 The Input Space . . . 52
- 4.2 Prediction in the Steady State . . . 54
- 4.3 Polynomial Point Prediction . . . 55
- 4.4 Prediction of the Envelope . . . 56
- 4.5 Results, Problems and Potential Solutions . . . 57
  - 4.5.1 Multi-Scale Space Splitting . . . 58
  - 4.5.2 Hierarchical Cluster Structures . . . 59
  - 4.5.3 On-line Learning . . . 61

5 Toward a Final Model **69**
- 5.1 The Architecture . . . 69

5.2 Learning 71

6 Conclusion **74**

A Violin and Bow Hardware **77**

Nomenclature **83**

Bibliography **83**

Chapter 1

Introduction

1.1 Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.

1.12 Denn, die Gesamtheit der Tatsachen bestimmt, was der Fall ist und auch, was alles nicht der Fall ist.

L. Wittgenstein, TRACTATUS[1]

The two resonance holes of the violin remind one of a question mark. Asked why they have that shape, a violin maker once responded: "This is an expression of the fact that the violin stands for one big question mark."[Buc73, P.9].

This anecdote contains a lot of truth, both from a violin makers point of view and that of a scientist. Violin makers tried to understand the 'secrets' of masterpieces made by Stradivarius or Guaneri del Gesu for centuries, in order to build instruments of equal sound and beauty. None of the copies they made reached the quality and the aura of the original. More recently scientists started to analyze string instruments in order to understand the governing physical equations. Yet, they could never hope to describe the violin as a complex system with its many degrees of freedom. So the intuitive approach of the violin maker, as well as the analytical approach of the physicist, failed to explain the functionality of string instruments in general, and of masterpieces of the art of violin-making in particular.

Given these difficulties, it isn't surprising that most attempts to synthesize violin sound artificially haven't been very successful. While electronic pianos have come close to the acoustical original, the synthesis of string instruments still has a long way to go.

We believe that there are two major reasons for the failure of electronic and digital violin synthesis:

i) Conventional synthesis methods use linear models to represent the audio signals in time and frequency domains or to simulate the physical behavior of the instrument. However, these linear models are unsuitable for dealing with the highly non-linear behavior of string instruments.

[1] An English translation of the Wittgenstein quotes is given in chapter 6.

ii) Most conventional synthesis techniques use interfaces different from the original musical device to deliver input to the sound engine. These artificial devices are not suitable replacements for the original. A keyboard does not permit the same subtle control as a violin bow does. The physical input of a musical instrument is intrinsically linked to its physical output, so that any change in the input control space necessarily results in a different sound.

Given these assumptions we propose a new concept of musical synthesis, based on time series analysis, state space reconstruction and machine-learning techniques. We consider the violin as a physical system, driven by a set of control inputs that produces an audio output. The system can be seen as a black box that hides its internal physical states, but lets us access the input time series (e.g. bow position, bow pressure, finger position) and the output time series (audio signal). First, the relevant input data and the audio signal of a played violin are recorded simultaneously. Then a non-linear functional match between the input time series and the audio output is inferred. This function is exclusively reconstructed from the collected data. It phenomenologically describes the global physical behavior of the violin in a reconstructed state space, without consideration of interior physical mechanisms. Once the phenomenological training is done, the computer has literally learned to sound like a violin when given the appropriate control inputs.

Although the main motivation of this work is the musical application, the approximation and modeling techniques which have been developed are general. They are applicable to any physical system with a defined input and output. In fact, state-space-reconstruction as the principal foundation of our prediction technique is meaningful for any imaginable physical object. Moreover, non-physical real world systems can be represented by the machine learning techniques we are going to present, provided they share some very general features such as predominant non-linear behavior.

Within the imaginable physical non-linear systems, the violin represents an ambiguous case, as it seems at the same time both difficult and easy to be modeled. It is difficult to convince human perception that the synthesis model behaves as well as the original instrument. Any 'sloppiness' in modeling will certainly be detected by a musically trained and conditioned audience. For similar reasons, we might find it easier to model a violin than to model abstract physical phenomena. Our clearly defined goal is to synthesize high quality sound. In addition to abstract error measures, the perceptional error measure should be most helpful. So the violin seems to be an ideal test application for state-space-reconstruction, for model architectures and for parameter search techniques.

The first part of this paper deals with the theoretical framework of embedding synthesis and state space reconstruction. The embedding theorem is reviewed and its particular applicability for musical synthesis is pointed out. We then describe and justify the function approximation and pattern recognition techniques that were used for predictor models. Particular attention is paid to clustering algorithms, such as melting algorithms and the Expectation-Maximization algorithm. These techniques are developed towards cluster weighted modeling in the sense of simple local experts, e.g. local linear models, being softly weighted by clusters in the input space. We also point out the analogies between

the clustering algorithms and thermodynamic terminology. The efficiency of the proposed algorithms is then tested with some simple data sets.

In the second part of this thesis the function approximation techniques are applied to actual violin data. We analyze the data with regard to various prediction goals, build the specific model and test it in prediction and synthesis. We also discuss further improvements in model architecture and in computational efficiency which represent important steps towards the final model. In particular, hierarchical mixture architectures are considered that partition the prediction space in time and in the input domain.

Finally, a model is proposed that integrates the theoretical and experimental results described in this paper. The proposed predictor is subject to change until it is used for a stage performance, but it reflects insight and ideas which could be the basic structure of a model that one day will deserve the name 'Digital Stradivarius'. Although the final model is proposed in terms of musical synthesis, it can be interpreted as a general framework for the prediction of non-linear multi-time-scale systems.

Chapter 2

Embedding and Musical Synthesis

2.01231 Um einen Gegenstand zu kennen, muß ich zwar nicht seine externen-aber ich muß alle seine internen Eigenschaften kennen.
L. Wittgenstein, TRACTATUS

2.1 The Overall Concept

We are trying to built a model which matches the output of a physical system to its physical input. For the violin application, this functional match relates input time series such as bow velocity and bow pressure to the audio signal of the violin.[1]

The approximation of the input-output function is completely phenomenological and data-driven. Deciding on the relevant input series for our model requires some insight into the local physical behavior, such as the bow-string interaction, Helmholtz modes and acoustic properties. However, once the relevant input series are known, the details of this physical behavior are no longer a direct concern and we can proceed to build the phenomenological model, based on collected data. Unlike the case of a finite element model, insight into the governing equations of violin strings and corpus is helpful, but not necessary.

One might think that analysis in the frequency domain should be an important part of this paper. Indeed, many synthesis approaches are based on time/frequency representations of musical sound. We will point out below why the conscious renunciation of Fourier spectra is considered to be the strength of our approach.

As opposed to conventional techniques this synthesis approach is based on the reconstruction of the instrument's state-space. The concept of state-space-reconstruction has become quite popular as a result of the recent flurry of activity in the field of non-linear dynamic systems, particularly chaotic systems. The notion of state-space derives from the Hamiltonian representation of dynamic systems. The system is represented in a n-

[1] Once again we point out that the results from this paper apply to physical systems in general. Whenever phenomena or results are described in terms of the particular device violin, it is done for clarity, and not meant to be restrictive to this case.

dimensional space where the axes are labeled with the Hamiltonian coordinates of position and impulse [MS94, P.16].

The violin spans a high dimensional mechanical state-space. Yet, we do not have access to the state variables, nor could we hope to represent the many mechanical degrees of freedom of the violin. However, it is possible to reconstruct a space that is topologically equivalent to the real state space. Known as a time lag space, this space represents the effective degrees of freedom of the system.

Given that the audio signal of the violin is known, the state-space is reconstructed from this one observable and its time-lagged versions. Given the time series $s(t)$ the n-dimensional time-lag space of the system consists of n axes, representing $s(t), s(t-\tau), s(t-2\tau), ..., s(t-(n-1)\tau)$.

Since we are dealing with a driven system, information about the physical input must be included in the model. Therefore, the number of space dimensions is further increased by the input dimensions. Possible input time series are: the sensored bow position, bow pressure and finger position. Denoting the input space by the time series $i_1(t), i_2(t), ..., i_m(t)$, the complete space is defined by the axis

$$s(t), s(t-\tau_s), s(t-2\tau_s), ..., s(t-(n_s-1)\tau_s), i_1(t-\tau_i), i_1(t-2\tau_i), ...,$$
$$i_1(t-(n_i-1)\tau_i), ..., i_m(t-\tau_i), i_m(t-2\tau_i), ..., i_m(t-(n_i-1)\tau_i)$$

where τ_s, τ_i denote the time delay between two samples of the audio signal and the input time series and n_s, n_i denote the total number of lagged series of the audio signal and the input series. The total space dimension is therefore $D = n_s + m \cdot n_i$.

If D is sufficiently large[2], $s(t)$ can be described by a singular valued function of the other space dimensions. Our goal is then to approximate the hyper plane

$$\begin{aligned} s(t) \quad = \quad & f(s(t-\tau_s), s(t-2\tau_s), ..., s(t-(n_s-1)\tau_s), i_1(t-\tau_i), i_1(t-2\tau_i), \\ & ..., i_1(t-(n_i-1)\tau_i), ..., i_m(t-\tau_i), i_m(t-2\tau_i), ..., i_m(t-(n_i-1)\tau_i). \end{aligned} \tag{2.1}$$

Once f is found the relation between $s(t)$, delayed versions of $s(t)$ and the input vector is used in an iterated prediction process that now becomes discrete. Any audio sample $s(n)$ is predicted by f in which t is replaced by $n \cdot T$ and τ becomes an integer number of samples. The predicted $s(n)$ serves as new input for future prediction.

In the continuum of methods employed in musical synthesis, our approach can thus be characterized as lying between physical modeling and pure signal processing and sampling. Although the mechanics of the violin are not described, its physical behavior is represented in state space. This image is exclusively built on the analysis of collected data, yet, it does not imitate the signal itself but rather the physics behind it.

In order to built a finite element model of the violin, roughly 10^9 floating point operations per second are needed[3]. However, this amount of computational effort is still insufficient to

[2]The exact conditions will be discussed in section 2.4

[3]This approximation assumes a certain spatial model resolution of the violin corpus, which seems to be

produce finite element violin models which could reproduce humanly perceptible differences for different violins. In our case, we hope to use less computational resources, while at the same time preserving the distinguishability between different instruments.

2.2 Time-Series-Prediction

There have been many attempts to predict the future behavior of a time series, given some sample values of its past. In almost all disciplines people try to extract information from a system's observable, so that this one observable becomes predictable, and that the system can be classified with regard to typical invariants of dynamic systems such as degree of non-linearity, internal degrees of freedom or noise level. The range of examples for time series prediction reaches from physical systems such as laser fluctuations [WG93, P.4], over biological phenomena such as epidemic cycles [DES94] to sociological or economic issues such as stock prices. The analysis of the capital markets has perhaps become the paradigmatic application of time series analysis [WMS95, WZ96].

Time series analysis of acoustical systems is also not new. Yet, analysis in the time-domain has rarely been used for predicting musical sound – that is, for synthesis of musical signals at a subtle level. In the context of synthesis, frequency-domain approaches have played the predominant role. Only few attempts have been made in recent years to extract information from acoustical time series with non-linear techniques [LP88, Gib88, PCG91, MS94]. None of the work cited used the extracted information for re-synthesis.

Our attention is focused on the acoustical application. Yet, we remind the reader once more that all results of this paper are applicable to non-linear systems in general. In fact, the embedding theorem, which will be introduced in the next section, originally was formulated by Floris Takens [Tak81] in the completely different context of fluid dynamics. The common feature of Takens description of turbulence and our re-synthesis of the violin is, the non-linear behavior of the two systems.

The history of time series analysis can be seen as a development from linear approximation methods towards more and more non-linear techniques. In order to understand our own approach to time series prediction, we propose a short flash-back to the linear roots of time series theory and to linear systems theory.

2.2.1 Linear Systems

Any dynamical time-variant system, whether linear or not, can be written in the form

$$\begin{aligned} \dot{\vec{x}} &= f(t, \vec{x}, \vec{u}) \\ y &= \eta(t, \vec{x}, \vec{u}) \end{aligned} \tag{2.2}$$

adequate to represent the basic mechanical behavior of the body. The number of nodes ($\simeq 10^5$)is then multiplied by the number of operations per node ($\simeq 10$)and the audio sampling rate ($\simeq 10^3$).

where $\vec{x}$ is the systems state vector, $\vec{u}$ describes the driving forces of the system and y describes any of the systems output observable[4]. For linear systems, the functions f and η can be described by matrices and the system of differential equations adopts the simple form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{F}(t) \cdot \mathbf{x}(t) + \mathbf{G}(t) \cdot \mathbf{u}(t) \\ y(t) &= \mathbf{H}(t) \cdot \mathbf{x}(t) + \mathbf{D}(t) \cdot \mathbf{u}(t) \end{aligned} \tag{2.3}$$

where $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{D}$ are time-variant square matrices. The system is represented as a system of first-order differential equations, as all higher order differential equations can be transformed into a first-order system. For time-discrete systems, such as any digital representation, the equivalent equations[5] are

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}(k) \cdot \mathbf{x}(k) + \mathbf{B}(k) \cdot \mathbf{u}(k) \\ y(k) &= \mathbf{C}(k) \cdot \mathbf{x}(k) + \mathbf{D}(k) \cdot \mathbf{u}(t) \end{aligned} \tag{2.4}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are again time-variant matrices. We refer to the time-discrete system whenever possible, as it is closer to our final digital implementation of a physical system.

Given system 2.4, many important features and invariants are defined by the systems matrices, such as stability, observability and controllability. Linear systems can easily be characterized as well as designed, by analytically 'playing' with the system matrices.

From a more time-series-oriented perspective, equation 2.4 can also be written as an Autoregressive Moving Average Model (ARMA)[6] of the following form:

$$x(k) = a(k) + \sum_{m=1}^{M} b(m)x(k-m) + \sum_{n=1}^{N} c(n)u(k-n), \tag{2.6}$$

where we assumed a scalar input u(k)[7]. Since the map 2.6 is time-discrete and linear, it can be most compactly written as a z-transform, which transforms the convolution in the time domain in a simple product:

$$\begin{aligned} X(z) &= A(z) + B(z)X(z) + C(z)U(z) \\ &= \frac{A(z)}{1 - B(z)} + \frac{C(z)}{1 - B(z)}U(z) \end{aligned} \tag{2.7}$$

where $A(z)$, $B(z)$, $C(z)$ and $B(z)$ are polynomials in z, and $X(z)$ and $U(z)$ denote the z-transform of state and input vector. $Y(z)$ is usually identical with one of the systems states or can be described as a linear combination of all of them. The coefficient vectors can be obtained from the correlation functions as described in [Ger97, P.184].

[4] in the following vectors will be denoted $\mathbf{x}$ or $\vec{x}$.

[5] see [Mey93, II P.162] for the transformation from time-continuous to time-discrete state vectors.

[6] The AR part describes the part of the output that is a linear regression of its previous values, the MA part describes the part that is defined as a moving average of the input series.

[7] a multidimensional input vector $\mathbf{u}(k)$ simply causes multiple sums over the $u_i(k-n)$ in 2.6.

Thus, three formalisms have been presented to describe a linear system. They all contain the same information and they can be transfered into one another. Each of the representations is useful for different tasks such as the designing or characterization of filters. They work well for linear deterministic systems and for fully stochastic systems. However linear description techniques break down in the region between these two unlikely limits [Ger97], and unfortunately all three descriptions break down under the same conditions [Met96].

In fact, none of the above description techniques is any useful for non-linear systems. Although the concept of state-space is still meaningful, there is no compact description in the form of matrices that relates these states. Neither is any linear transform such as the Fourier-transform for time-continuous systems or the z-transform for time-discrete systems applicable.

Musical instruments 'look' very linear. Most of them have a clear harmonic spectrum, which certainly contains useful information about specifics of an instruments. However, the time-frequency representations of musical signals tell only half the truth. The important musical and instrumental characteristics are defined by global non-stationary and non-linear phenomena such as the attack noise or waveform shaping produced by a change in bowing. These phenomena can never be described adequately by linear methods as defined above.

Once the decision for a nonlinear model is taken, we are left with a small set of classification instruments, which are outlined in section 2.5. First, the notion of state-space-reconstruction has to be introduced, since nonlinear characterization is done almost exclusively based on this concept.

2.3 State-Space-Reconstruction of Non-Linear Systems

So far we have seen that it is fairly easy to describe and control linear systems. We suspect that it is difficult – if not impossible – to describe non-linear systems. In order to improve our understanding we have chosen a representative system known to behave non-linearly, the Lorentz-Attractor – one of the most striking examples for non-linear, chaotic behavior. Edward Lorenz tried to approximate Navier-Stokes equations for a convectional system [Ger88, P.8] and found the following set of first-order differential equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy \end{aligned} \tag{2.8}$$

Figure 2-1(a) shows 5000 sample points of the x variable of the Lorenz set in the time domain for the parameter set $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$ [8] and figure 2-1(b) shows the Fourier-transform of the same time series. Although the time-domain representation seems to have some sort of periodic behavior, the frequency-domain plot gives no insight at all into

[8] The time continuous differential equations were approximated by a fourth-order Runge-Kutta approximation (h=0.01), see [Ger97, P.68].

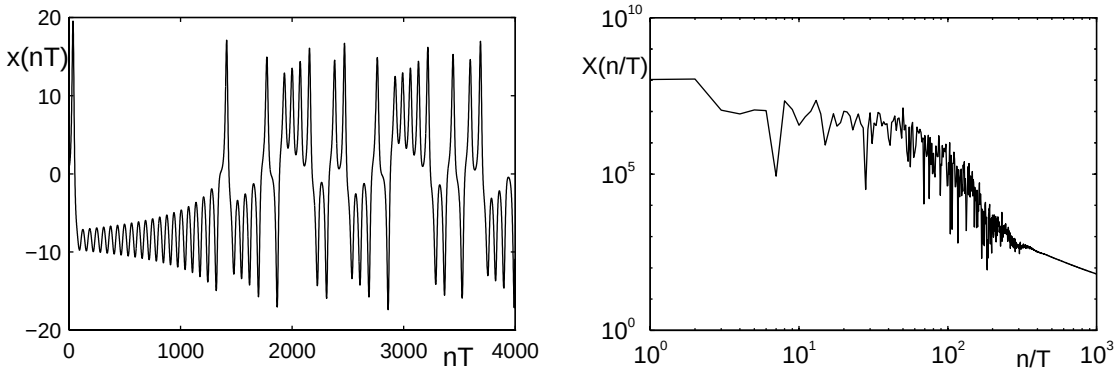

Figure 2-1: x-coordinate of the Lorenz set in (a) time and (b) frequency domain.

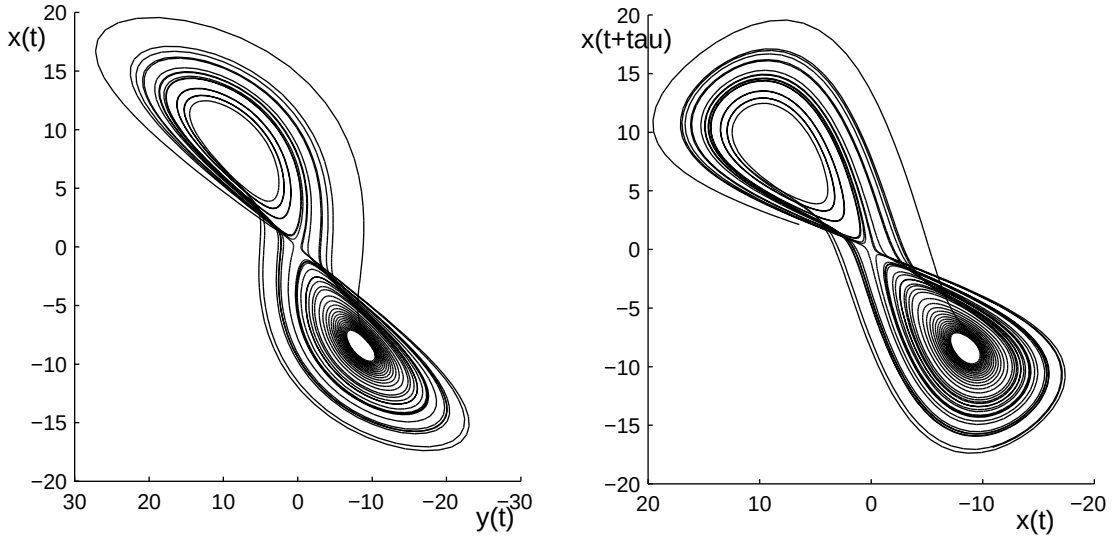

Figure 2-2: Lorenz Attractor in (a) the x-y-z state space projected in 2D. (b) 3D time-lag space of $x(t)$ projected in 2D ($\tau = 10$ samples).

the nature of the system. We tend to assume random behavior rather than deterministic equations.

Yet, if we look at the state space representation of figure 2-2(a), suddenly a very regular behavior occurs. We discover a chaotic attractor, that keeps 'drawing the same picture' in state space, no matter which initial conditions we choose for our differential equations. However, how should such state variables be found for an unknown system of arbitrary form and dimension? This is when the embedding theorem comes in.

In 1981 Floris Takens discovered, that there is a diffeomorphic mapping between the manifold of states in the state space $X \in \mathcal{R}^{d_1}$ of an arbitrary system and a second manifold $Y \in \mathcal{R}^{d_2}$ that can be reconstructed from one observable of the system. Given the output measurement $y(t)$ of a system, Y is reconstructed by assigning time delayed versions of $y(t)$ to the axis of Y, such that the axis are labeled $y(t), y(t-\tau), y(t-2\tau), ..., y(t-(d_2-1)\tau)$

where τ describes a time constant that will be discussed later. The new manifold in Y is now supposed to be topologically invariant to the original one in X. In figure 2-2(b) the Lorenz-Attractor is reconstructed in the described manner. The similarity to figure 2-2(a) is obvious. Not only is the overall topological shape preserved, but also local details of the orbit are easily identified in the two plots. We will sketch a proof of the embedding theorem in the next section, once its final form is known. Here we state phenomenologically the most important results:

1. The manifold formed by a dynamic system's Hamiltonian states is topologically identical to a second manifold formed by one single observable of the system and its $d_2 - 1$ time lags. The two vector manifolds differ by no more than a smooth and invertible local change of coordinates. They therefore define a local diffeomorphism.

2. The above is true only, if the dimension of the lag space d_2, referred to as the embedding dimension is large enough. In general, the embedding dimension should be chosen as $2{\cdot}d{+}1$, where d is the number of degrees of freedom of the system. However, this choice may be 'too good', meaning that a smaller d_2 works as well. The choice of $d_2 = 2 \cdot d + 1$ assures that there is no better way of preventing our solution from lying on a bad (ambiguous) subset in the embedding space. Yet, we are interested in the number of effective degrees of freedom, which, due to dissipation, can be much smaller than the number of mechanical degrees of freedom.

3. The embedding theorem is true generically, meaning that there are only very few unlikely cases for which it does not apply. In those pathological cases, however, only a small change of variables is needed to make the theorem work again.

4. Due to the robustness of the theorem, the choice of τ does not matter in principle. Some choices of τ make the manifold 'easier to read', but no choice would change the manifold's topology. However, a small τ lets the manifold degenerate into the space diagonal. A lag which is too big relates points of the time series which do not correlate dynamically.

2.4 Input/Output Embedding

So far we dealt only with autonomous systems and their embedding in time-lag-space. These systems keep producing output, based on a set of initial conditions and internal governing equations. They can be formally described as $\frac{\partial \mathbf{x}}{\partial t} = f(\mathbf{x}, \mathbf{x}_0)$. However, most systems depend on a time variant input as well. If the systems governing equations are unknown, we obtain a relationship as shown in figure 2-3.

Obviously, musical instruments belong to the class of input/output systems. The driving input reflects the intentions of the player. Therefore, we need to include the information from the driving forces in the embedding framework and we hope to end up with a description of the system, that relates the output uniquely to the input.

Given are the time dependent scalar input $u(t)$ and the output signal $y(t)$ of a system. In extension to the embedding theorem for autonomous systems M. Casdagli [Cas92, P.266] and others [Hun92] proposed the following solution for driven systems:

$$y(t) = f(y(t-\tau), y(t-2\tau), ..., y(t-(d-1)\tau), u(t), u(t-\tau), u(t-2\tau), ..., u(t-(d-1)\tau) \quad (2.9)$$

Casdagli shows that there is a diffeomorphic mapping between the reconstructed manifold defined by 2.9 and the state space of the driven system. Given a d-dimensional state vector, Casdagli claims that $2d+1$ dimensional time-lag-vectors for both, the input time series $u(t)$ and the output time series $y(t)$ guarantee the function $f()$ to be unique. The remarks concerning embedding in section 2 still apply. Yet, our model has become much more general and seems to cover the whole world of possible physical behavior.

The time-discrete version of 2.9 can be written as

$$\begin{aligned} y(n) &= f(y(n-k), y(n-2k), ..., y(n-(m+1)k), \\ &\quad u(n), u(n-k), u(n-2k), ..., u(n-(l+1)k) \end{aligned} \quad (2.10)$$

We sketch a proof for the time-discrete case as done in [Cas92]: Let's assume a finite dimensional unknown system determined by its d-dimensional state vector s_n and the following set of equations:

$$\begin{aligned} \vec{s}(n+1) &= f(\vec{s}(n), \vec{u}(n)) \\ y(n+1) &= h(\vec{s}(n+1)) \end{aligned} \quad (2.11)$$

If equation 2.11 is true the delay vector

$$\vec{v}_n = y_n, y_{n-k}, y_{n-2k}, ..., y_{n-(m-1)k}, u_n, u_{n-k}, u_{n-2k}, ..., u_{n-(l-1)k}$$

is needed to describe uniquely and smoothly the unknown state $s(n)$. Therefore there has to be a function $P : \mathcal{R}^{m+l} \Rightarrow \mathcal{R}$ for all y_n such that

$$y(n+1) = P(v(n)) \quad (2.12)$$

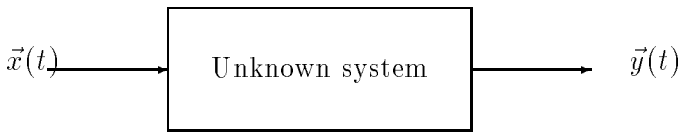

Figure 2-3:

We assume $m = l$ and define the smooth map $\Phi : \mathcal{R}^d \times \mathcal{R}^{m-1} \Rightarrow \mathcal{R}^m$ by

$$\begin{aligned} \Phi(\vec{s}, u_{n-1}, ..., u_{n-(l-1)k}) &= ..., h(f(f(\vec{s}, u_{n-(l-2)k}, u_{n-(l-1)k}))), \\ & \quad h(f(\vec{s}, u_{n-(l-1)k})), h(\vec{s}) \\ &= y_n, y_{n-k} ... y_{n-(m-1)k} \end{aligned} \tag{2.13}$$

For equation 2.13 to have a unique solution $\vec{s}$, m must be chosen such that $m > d$. For $m = d$ there would be d simultaneous non-linear equations for d unknown components of $\vec{s}$, which normally have more than one solution. If m is increased by one, one more equation is obtained which should reduce the set of solutions to a unique solution for almost the whole solution space. Yet, if we are unlucky, our solution lies on a bad subset Σ^{d-1} of $\mathcal{R}^d$ of dimension $d-1$. By increasing m by 1, we are reducing the subset dimension by 1 until finally for $m > 2d$, the bad subsets disappear. We ignore further bad subsets which can be induced by the input vector but which generically have measure 0 and which do not disappear with a further increase of m. Therefore, we cannot do better than choosing $m = 2d+1$.

Having showed that there is a solution for $\vec{s}$ for all n, we also proofed that there is a smooth function P as defined in equation 2.12. P can be obtained by plugging $\vec{s}$ into 2.11 and iterating. We summarize: There is a smooth and unique function P for generically all input sequences for $m, l > 2d$. For $m, l = d+1$ such a function P exists almost everywhere in the solution space [Cas92].

2.5 Characterization of Dynamic Systems

There are classification techniques for dynamical systems that apply to any system, no matter whether linear, nonlinear or chaotic. They all determine invariants of the system, preferably the embedding dimension of an unknown system. We state once more that the embedding dimension denotes the minimal state dimension that allows to uniquely represent the system. The embedding dimension is directly related to the number of internal degrees of freedom.

The most popular method for dimension estimation is the **correlation dimension** method. Given a reconstructed state space (equation 2.1) the correlation integral $C(D, N, r)$ is defined as

$$C(D, N, r) = \frac{1}{N} \sum_{i=1}^{N} B_i(\vec{x}_i, D, r) \tag{2.14}$$

where D is the space dimension, N is the number of data points, r denotes a radius around point $\vec{x}_i$, and $B_i()$ denotes the proportional number of points that are found within the hyper-sphere around point $\vec{x}_i$. The correlation dimension ν is defined by the asymptotic scaling of

$$C(D, N.r) \propto r^{\nu}.$$

ν can never be bigger than D, yet if D is increased, ν will not grow any further as soon as

its correct value is found. For chaotic systems ν can become non-integer and then defines a fractal dimension.

Lyapunov Exponents quantify the rate of divergence of nearby trajectories. They measure the deformation of an infinitesimal small sphere in time. Let's assume a hyper sphere of the original radius $r_i(0)$ in a reconstructed state space of Dimension D. The sphere is going to be deformed with time into an ellipsoid with the principal axis $r_i(t), i = 1, 2, ..., D$. The Lyapunov coefficient λ_i are then defined by

$$\lambda_i = \lim_{t \to \infty} \lim_{r_i(0) \to 0} \frac{1}{t} \log \frac{r_i(t)}{r_i(0)} \tag{2.15}$$

where the λ_i are usually ordered as $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_D$. The set $\{\lambda_i\}_{i=1}^{D}$ is called the Lyapunov Spectrum, $\sum_{i=1}^{D} \lambda_i$ describes the volume expansion rate. Whenever one or more Lyapunov coefficients become positive, chaotic behavior is observed. The Spectrum is related to the system dimension by the Kaplan-Yorke conjecture [Ger96].

Entropy is a further important characterization technique and is also reviewed very briefly. We do not discuss estimation problems and computational issues related to a practical entropy estimator but outline the principle idea. In particular, we do not discuss the accuracy of the entropy estimator with respect to the number of discrete bins N.

Lets assume a number N of discrete bins that partition the range of the variable x_t. A probability is assigned to every bin which is proportional to the number of data points that lie within the range of the bin.

The scalar entropy of a time series can then be written as

$$H_1(x_t) = -\sum_{N} p(x_t) \cdot \log_2 p(x_t) \tag{2.16}$$

The entropy of a joint distribution of the signal in two dimensional time lag space is defined as

$$H_2(\tau) = -\sum_{N_t} \sum_{N_{t-\tau}} p(x_t, x_{t-\tau}) \cdot \log_2 p(x_t, x_{t-\tau}) \tag{2.17}$$

Consequently, the block entropy for a D-dimensional lag space vector becomes

$$H_D(\tau) = -\sum_{N_t} \sum_{N_{t-\tau}} ... \sum_{N_{t-(D-1)\tau}} p_D \cdot \log_2 p_D \tag{2.18}$$

with

$$p_D = p(x_t, x_{t-\tau}, ..., x_{t-(D-1)\tau}).$$

Now the redundancy between two block entropies of order D and D+1 can be expressed as

$$R_D(\tau) = H_1(\tau) + H_{D-1}(\tau) - H_D(\tau) \tag{2.19}$$

R_D is equal to the scalar entropy of signal $x_{t-(D-1)\tau}$, plus the joint entropy H_{D-1}, minus the joint entropy H_D. R_D is zero, if the new dimension is completely uncorrelated from the

dimensions which defined H_{D-1}. R_D becomes equal to the scalar entropy H_1 if the D-th dimension is completely determined by the other dimensions.

The system dimension is then equal to the smallest D that makes $R_D = H_1$. In order to be more precise we need to define the source entropy $h(\tau)$ as

$$h(\tau) = \lim_{d \to \infty} H_D(\tau) - H_{D-1}(\tau) \tag{2.20}$$

The dimension now can be calculated as the minimal D for which $h_D(\tau) = h_\infty(\tau)$.

None of the described techniques has been used for the experimental work of this thesis. In fact we concentrated on an alternative class of characterization methods which is characterization by learning. The embedding dimension we are looking for is the minimum dimension D which allows the most efficient prediction of the system. The main quality of the reconstructed space should therefore be its predictability which can be measured by the expected variance of its output. D has to be determined by cross validation. One more description level up, the embedding dimension D could be defined as the smallest dimension D which allows perceptionally optimal sound re-synthesis.

2.6 Preliminary Conclusions

It was been shown in the previous sections that the state space of any driven physical system can be reconstructed, given access to one observable of the system and to the driving forces. What does this result imply in the context of musical synthesis? Although there have been some attempts to characterize the non-linear behavior of acoustical systems in time-lag spaces [LP88, MLS93], state-space-reconstruction has never been used for musical synthesis. However, it seems to be most natural to think of a musical instrument as a black box which expects control information and in exchange puts out an audio signal. Making music can be interpreted as driving a complex physical device that offers a certain range of signal response.

We decided to follow the concept of state-space-reconstruction for our synthesis goal and therefore need to find answers on the following issues: How many effective degrees of freedom does the system violin have? Which is the minimal embedding dimension? Which are the relevant inputs and which is the minimal dimension of the input series? Might there be different time lags within the same time series? More practically we need an efficient approximation of the reconstructed state-space. The fact that the existence of such a space has been proven, is only half way through to its actual reconstruction. The major part of this thesis, therefore, deals with approximation techniques which extract an explicit functional form for f (equation 2.11).

To finish this section we present a simple reconstruction example in order to get a sense of the possible solution space. The system that comes closest to real world vibrating systems such as the violin is the quasi periodic system [BPV84, P.71]: Imagine a bi-periodic system

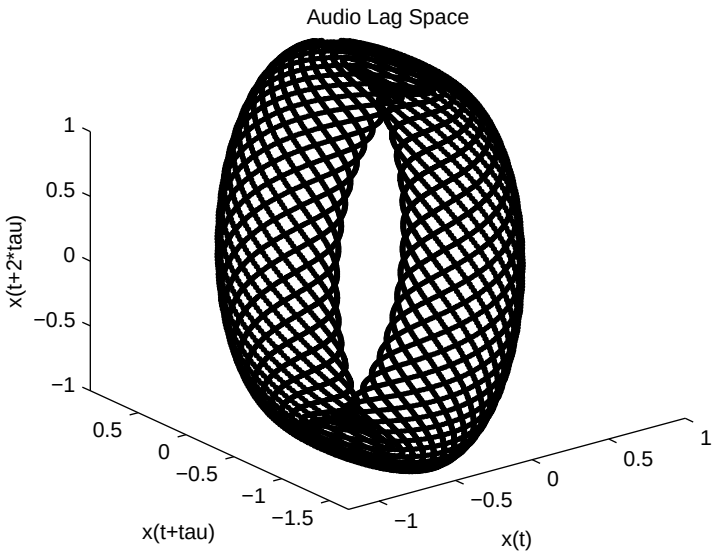

Figure 2-4: Quasi periodic attractor in three dimensional lag space. $x(t) = sin(2\pi \cdot 400 \cdot t) + sin(2\pi \cdot 1185 \cdot t)$.

characterized by two frequencies f_1 and f_2.

$$y(t) = sin(2\pi f_1 t) + sin(2\pi f_2 t) \tag{2.21}$$

Such a system creates a torus T^2 which can be visualized in $\mathcal{R}^3$ (figure 2-4). The trajectory on the torus may be interpreted as a superposition of two movements: an ellipsoid, described by the lower frequency and an tournament around that ellipsoidal trajectory caused by the higher frequency.

If f_2/f_1 is irrational, the trajectory does never close itself but keeps describing the surface of the torus. This case corresponds to the undamped steady state of a simplified musical instrument. Any string instrument tone (piano) can be momentarily described as a slightly in-harmonic superposition of multiples of the basic frequency, where the inharmonicity is function of the material properties of the strings.

If f_2/f_1 is rational, the trajectory is not dense on the torus, but describes a closed and periodic trajectory in space.

Finally, in order to get to know the space we are working in, we present a reconstructed piece of violin sound in the time-lag-space (figure 2-5).

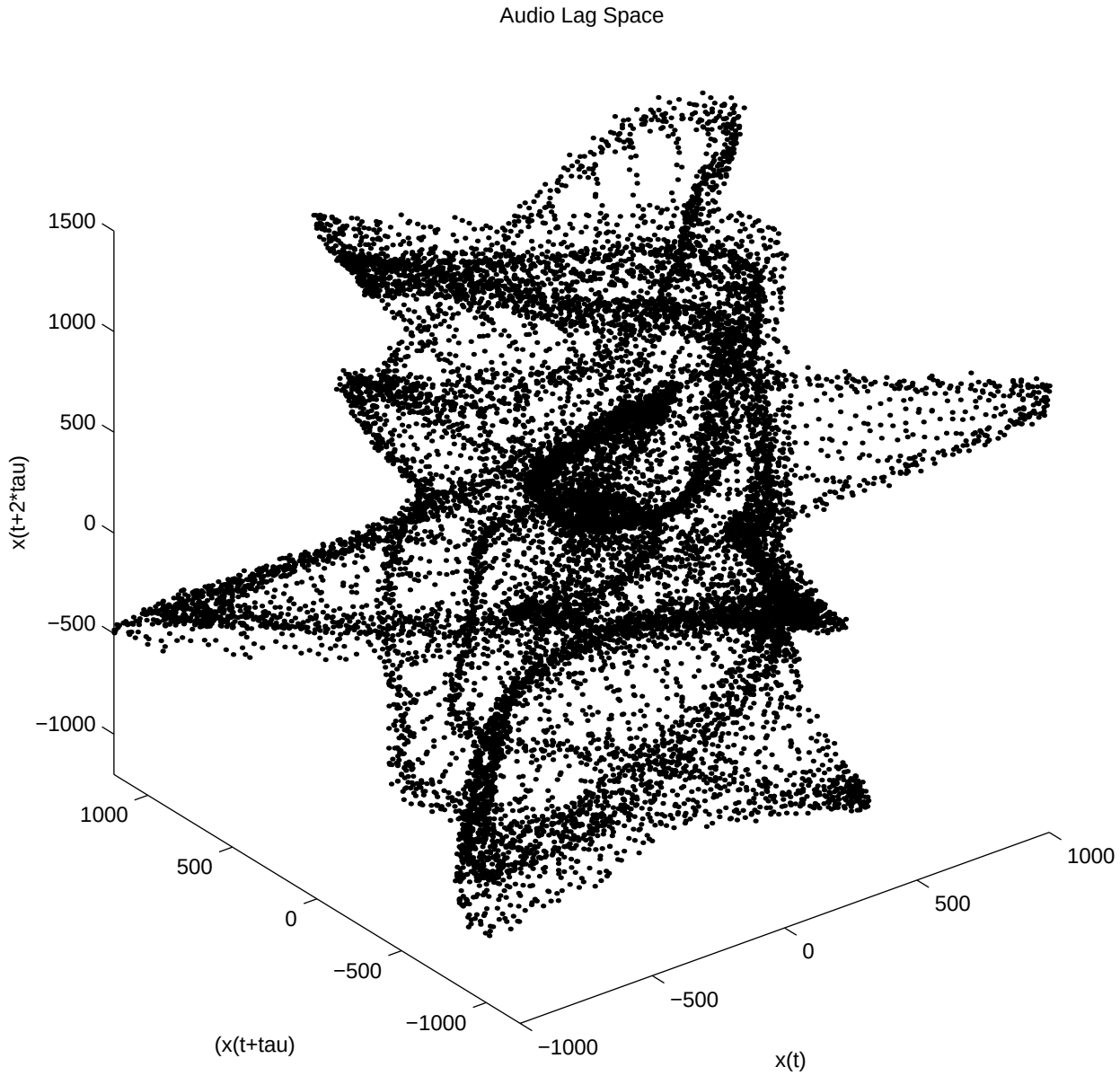

Figure 2-5: violin audio signal in the three dimensional lag space (4000 data points)

Chapter 3

Function Approximation in the State-Space

> 3.032 Etwas "der Logik widersprechendes" in der Sprache darstellen, kann man ebensowenig, wie in der Geometrie eine den Gesetzen des Raumes widersprechende Figur durch ihre Koordinaten darstellen; oder die Koordinaten eines Punktes angeben, welcher nicht existiert.
>
> 3.0321 Wohl können wir einen Gegenstand räumlich darstellen, welcher den Gesetzen der Physik, aber keinen, der den Gesetzen der Geometrie zuwiderliefe.
>
> L. Wittgenstein, TRACTATUS

3.1 Bayesian Priors and Regularization

The modeling objective and modeling techniques are most conveniently described in a Bayesian formalism. 'Bayesian language' has become popular in many disciplines, sometimes even overs tressed. In the context of function approximation the Bayesian framework applies most naturally and directly.

We are looking for the most likely model $\mathcal{M}$ given a set of data $\mathcal{D}$. Formally this task can be described as the maximization problem of a probability density function. The model $\mathcal{M}$ is to be found that maximizes

$$p(\mathcal{M} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mathcal{M}) \cdot p(\mathcal{M})}{p(\mathcal{D})} \tag{3.1}$$

The argument $\mathcal{M}$ which maximizes $p(\mathcal{M} \mid \mathcal{D})$, also maximizes $\log[p(\mathcal{M} \mid \mathcal{D})]$. Therefore the

maximization problem becomes:

$$\begin{aligned} max_{\mathcal{M}}\{p(\mathcal{M} \mid \mathcal{D})\} &= max_{\mathcal{M}}\{log\frac{p(\mathcal{D} \mid \mathcal{M}) \cdot p(\mathcal{M})}{p(\mathcal{D})}\} \\ &= max_{\mathcal{M}}\{log(p(\mathcal{D} \mid \mathcal{M})) + log(p(\mathcal{M})) - log(p(\mathcal{D}))\} \end{aligned} \tag{3.2}$$

The problem is characterized by a trade off between three terms. The first term evaluates how well a model describes the data. It can be read the opposite way as the probability that the data has been generated by the proposed model. The probability is expressed as a function of an arbitrary error measure. If Gaussian noise is assumed the first term becomes a least square estimate.

The second term evaluates the model. It expresses prior beliefs about what we think is a good model. These beliefs could be smoothness assumptions or exogenous model parameters such as the number of basis functionals. In a Minimum Description Length approach this term penalizes the complexity of the model. In fact the complexity aspect is a concern with any approximation technique. The model has to be kept small for computational as well as for 'philosophical' reasons.

The third term describes how important specific data is evaluated within the total set of data. This term comes in when on-line search algorithms such as the Kalman Filter are used. Usually old data is wanted to influence the model less than new data, so that non-stationary behavior can be recognized. This assumption can be translated into a decay parameter λ which reduces the weight of the old data, whenever new data is considered (see section 4.5.3).

A second related general estimation framework is Regularization theory. It has been shown in [GJP95] that function approximation can be interpreted and implemented as a variational problem, which takes into account the data as well as prior assumptions about the model, in particular smoothness assumptions. Given a set of data $g = \{(y_i, \mathbf{x}_i) \in \mathcal{R} \times \mathcal{R}^D\}_i^N$, we assume that y_i has been generated by a function $f() : \mathcal{R}^d \rightarrow \mathcal{R}$ in the presence of noise. The unknown $f()$ is to be reconstructed, but, as the number on sample points N is finite the problem has an infinite number of solutions; it is ill posed. In order to find the best solution out of the set of possible solutions further constraints have to be introduced: the Regularization priors. Those additional constraints may be of very different form. They reach from explicit smoothness priors to more implicit constraints expressed by the choice of basis functions. In fact the choice of local models represents a very strong a priori assumption about how the data looks like and what the model should be able to describe.

The fitting problem is translated into a functional H(f), which is to be minimized with respect to $f()$:

$$H(f) = \sum_{i=1}^{N}[f(\vec{x}_i) - y_i]^2 + \lambda \cdot \Phi(f()) \tag{3.3}$$

The first term measures the estimation error of the model. The second term measures how well the prior has been realized by the model. It is weighted by the regularization parameter

λ. A small λ indicates that after all we believe strongly in the training data and want any single data point to be well represented. A high λ indicates that we need to avoid over fitting of noisy data. λ is to be determined by cross validation with in-sample or out of sample data.

[GJP95] proposes a general form for the regularizer $\Phi(f())$:

$$\Phi(f) = \int_{\mathcal{R}^D} \frac{|\tilde{F}(\vec{s})|^2}{\tilde{G}(\vec{s})} d\vec{s} \tag{3.4}$$

where $\tilde{F}(\vec{s})$ describes the Fourier transform of $f(\vec{x})$ and $1/\tilde{G}(\vec{s})$ describes a high pass filter. $\lambda\Phi$ penalizes high frequency partials in the approximation functional. This particular choice of Φ interprets smoothness as a slow change of $f()$ with small changes of $\vec{x}$. The regularizer we are going to use in the next section is formally slightly different from expression 3.4, but expresses the same idea.

As has been mentioned before, besides the formal priors we dispose of a very natural quality criteria which is perception. No matter how the model architecture and its parameter look like from a mathematical point of view, the best model is the model that sounds best. Of course, we hope that perception, physics and mathematical description correlate. We dream of a model that is described very compactly, that represents the entire physical behavior of the instrument and at the same time produces the most beautiful sound.

A further general classification of modeling and approximation techniques comes from the degree of *nonlinear descriptive power*, built into the basis functions and the parameter set to be varied. Linear ARMA models do not have any such power. They consist of linear terms weighted by linear coefficients. The first step toward nonlinear behavior consists in adding higher order terms to the ARMA model. The output becomes a polynomial function of the input dimensions. Yet, the set of parameters to be adjusted to the data is a set of linear coefficients. They can be found by a least square fit in one single operation (see next section). In general this class of approximation functions is described by

$$\hat{y} = \sum_{j=1}^{M} a_j \; f_j(\vec{x}) \tag{3.5}$$

Given this class of functions the number of basic terms (cross terms) increases exponentially with space dimension. Therefore these techniques do well for 'small' problems, but they are inefficient for high dimensional spaces. The next step is to assign a nonlinear behavior to coefficients by making them part of the basis terms. The functional form now becomes

$$\hat{y} = \sum_{j=1}^{M} f_j(\vec{x}, \vec{a}_j)_i \tag{3.6}$$

Equation 3.6 expresses the most general form of nonlinear estimation. The coefficients $\vec{a}_j$ cannot be found by simple matrix inversion anymore. Fancier search techniques have to be applied, but in exchange the model has become much more powerful.

We propose two nonlinear fits, each of which belongs two one of the classes of nonlinear approximation techniques.

3.2 A Polynomial Approach

A classical realization of the regularization principle are polynomial approximations. Based on the idea that orthogonal polynomials span an orthogonal and complete space for the class of bounded functions, a polynomial approximation scheme can be interpreted as a non-linear extension of linear state and time series models (see section 2.2.1). We propose a polynomial model, where the prior assumption is to penalize strong second order derivatives.

Our starting point is a set of data $g = \{(\mathbf{x}_i, y_i) \in \mathcal{R}^d \times \mathcal{R}\}_{i=1}^N$ which has been obtained by sampling data from an input/output system. In our specific application y_i describes the audio signal of the violin, and $\mathbf{x}_i$ describes the input vector consisting of the actual input time series, their lagged versions and the lagged vector of the output. We assume that this set of data is generated by a function $f : \mathcal{R}^d \rightarrow \mathcal{R}$ and has been corrupted by noise. A least square error function is considered in order to evaluate the fit of the data and the following functional form is proposed:

$$f(x) = \sum_{k}^{K} a_k \Psi_k(\mathbf{x}) \tag{3.7}$$

with

$$\Psi_k(\mathbf{x}) = {x_1}^{e_{k1}} \cdot x_2^{e_{k2}} \cdot ... x_D^{e_{kD}} = \prod_{d=1}^{D} x_d^{e_{kd}}$$

We denote O the order of our approximation and allow any combination of integer values for e_{kd} such that $e_{k1} + e_{k2} + ...e_{KD} \leq O$. All these terms are present in the model, including the constant term $e_{0,d}$, for which all the exponents $e_{ki} = 0$.

The number of polynomial terms K can be expressed in terms of the space dimension D and the polynomial order O as [Met96, P.62]

$$K(D, O) = \left(\begin{array}{c} (D+O) \\ D \end{array} \right) = \frac{(D+O)!}{D! \cdot O!}. \tag{3.8}$$

As regularizer the integral over the second order derivative (Laplace Operator)[1] of $f()$ is chosen. This term turns out to be easy to evaluate given that our polynomial terms are simply superposed, and that we can easily normalize our data on the finite support $\mathcal{R}^D \cap [0, 1]$. Note, that such a derivative term in the frequency domain becomes a low-pass filter and that therefore our choice is very close to the general form of Regularization terms

[1] the first order derivative is even easier to determine and a mixed regularizer consisting of the two terms has also been tested. However, as two different weighting terms have to be optimized, the cross-validation search becomes much more complex.

as described in section 3.1 (equation 3.10). The regularizing term becomes

$$\begin{aligned} H_1 &= \int_{x_1=0}^{1} ... \int_{x_D=0}^{1} (\Delta f)^2 dx_1 ... dx_D \qquad (3.9) \\ &= \int_{x_1=0}^{1} ... \int_{x_D=0}^{1} \sum_{i=1}^{D} (\frac{\partial^2 f}{\partial x_i^2})^2 + 2 \cdot \sum_{i=1}^{D} \sum_{j=1}^{i-1} (\frac{\partial^2 f}{\partial x_i \partial x_j})^2 dx_1 ... dx_D \end{aligned}$$

and the complete variational term is evaluated as

$$\begin{aligned} H(f) &= H_1 + \lambda \cdot H_2 \qquad (3.10) \\ &= \int_{x_1=0}^{1} ... \int_{x_D=0}^{1} (\Delta f)^2 dx_1 ... dx_D + \lambda \cdot \frac{1}{N} \sum_{i=1}^{N} [f(\mathbf{x}_i) - y_i]^2 \end{aligned}$$

We need to find the minimum of $H(f)$ with respect to the parameters of $f()$. Therefore $H(f)$ is derived in direction of every single parameter a_k:

$$\begin{aligned} \nabla_{\vec{a}} H &= \nabla_{\vec{a}} H_1 + \lambda \cdot \nabla_{\vec{a}} H_2 \qquad (3.11) \\ &= \mathbf{A} \cdot \vec{a} + \lambda \cdot (\mathbf{B} \cdot \vec{a} - \vec{c}) \qquad (3.12) \\ &= \vec{0} \end{aligned}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ denote matrices which describe equation 3.11. $\vec{a}$ denotes the vector of coefficients a_k. We finally obtain for $\vec{a}$

$$\vec{a} = \lambda(\mathbf{A} + \lambda \mathbf{B})^{-1} \cdot \vec{c} \qquad (3.13)$$

Given this general solution we try to find the appropriate value for λ by cross-validation with in-sample and out-of-sample data. This search may take quite a bit of time for high order approximations of high dimensional spaces, but there is no other way to determine the right λ than by empirical search.

What are the good features and what are the weaknesses of this approach? It is a well known fact that polynomial approximations are unstable on an infinite support. For $\|\mathbf{x}\| \to \infty$, $f(\mathbf{x})$ goes to infinity also. As we need the approximation to be stable only on the finite support which contains the training points, and as this is also the support, where we apply our Regularization term, we expect our function to behave well for the possible input data.

Yet, we are confronted with another stability problem which is due to the iterative structure of our model. As any predicted value is fed back as part of the input for future predictions, our prediction errors ad up. Input vectors may occur that are not in the range of $\mathbf{x}$ where $f()$ has been trained. This phenomena of non-locality of errors either forces the system to its fixed point or to infinity, whereas it is supposed to describe a periodic wave form. The case where there are no errors or where the errors add up to zero is unfortunately very unlikely.

Also, polynomials are computationally restrictive, for the number of terms grows expo-

nentially with dimension. As we need to calculate and store a $K \times K$ matrix[2], the number of terms quickly becomes prohibitive[3]. This phenomenon is known as the curse of dimensionality. Although we could consider using more computational resources, the principal problem of high computational cost remains.

The regularizer polynomial model in- and extrapolates pretty well, as will be shown in section 3.6. It allows to predict points that are placed 'far away' from points that have been part of the learning data, in the input as well as in the output space. Although the physical meaning of such true out-of-sample points has to be examined, in principle we like this feature. Due to its high dimensionality we cannot hope to fill the violin space entirely with data. Even if the musical solution-space were covered with training points, we need the system to respond to strange input constellations in the actual synthesis situation as well.

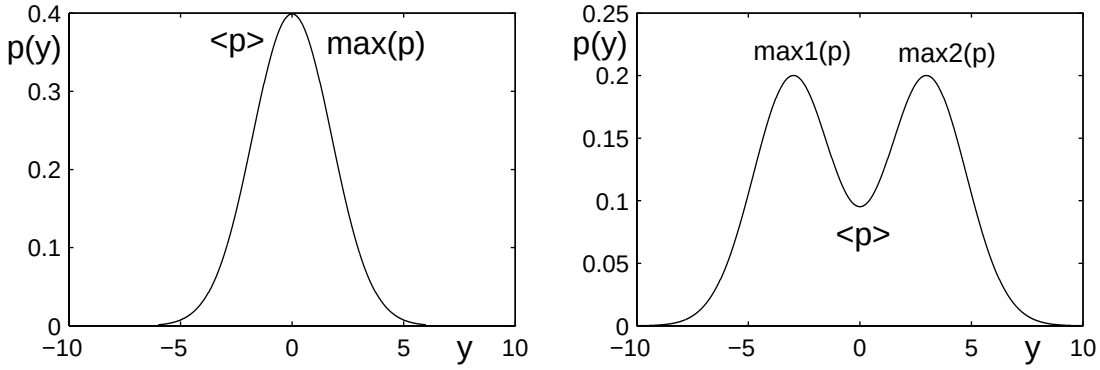

Figure 3-1: (a) Deterministic and (b) nondeterministic probability distribution

3.3 Probabilistic Forecasting with Cluster Weighted Local Models

Instead of fitting a deterministic function as in the last section, we consider a probabilistic approach. We do not predict a deterministic value, but let the system find the most likely value. The degree of randomness (the variance) will then be a measure for the quality of our model. A probabilistic approach to function fitting as opposed to deterministic hard decisions intuitively seems to be more appropriate in the context of prediction problems.

The probabilistic approach literally reconstructs the solution manifold of the state space by reconstructing its probability density function (PDF). The estimate of the PDF is done with Gaussian basis functions that weight local models over a limited domain of influence. Although cluster weighted modeling has been used before [Hun92], it has never been described in a formalism as compact as the one recently developed by Neil Gershenfeld and

[2]K denotes the number of polynomial terms (see 3.8)

[3]The upper most combination of D and O we could afford to calculate a parameter set within one day, was D=6 and O=8.

used for this paper[4]. The novelty of the proposed framework lies in its natural descriptive combination of density estimation and function approximation. We first present the principles and the mechanism of probabilistic estimation and prediction. Then two different learning algorithms are presented: Clustering by melting and the Expectation Maximization (EM) algorithm.

3.3.1 Density Approximation of the State Space

Here we present each of the steps of the iterated probabilistic forecasting procedure from data collection to the actual prediction step. The concept becomes most clear by following the model to be built up.

Again our starting point is a set of vectors

$$g = \{(Y_i, \mathbf{X}_i^D)\}_{i=1}^N = \{\mathbf{Z}_i^{D+1}\}_{i=1}^N.$$

$\mathbf{Z}^{D+1}$, Y and $\mathbf{X}^D$ are particular realizations of the random vector

$$\mathbf{z} = \{y, x_1, x_2, ..., x_D\}$$

where $\mathbf{x}$ denotes the reconstructed state vector which contains time lags of the audio signal, the input signals and their lags. y denotes the non-delayed output signal which we intend to predict. We assume y to be a scalar, but could easily extend the framework to a vectorial output. The time-discrete, state-continuous case is considered, where the $D+1$ elements of $\mathbf{z}$ are elements of $\mathcal{R} \times \mathcal{R}^D$. As there is only a limited set of possible values for $\mathbf{z}$ we end up with a subset

$$\mathcal{Y} \times \mathcal{X}^D \in \mathcal{R} \times \mathcal{R}^D.$$

In order to assure homogeneity in the probability space, we assume the elements x_d to share the same range of $\mathcal{R}$ which can easily achieved by normalizing the data set. We also assume the elements of g to be independent[5] and to be drawn from the same continuous probability density function $p_z(\mathbf{z})$. $p_z(\mathbf{z})$ is defined as

$$\int_C p_z(\mathbf{z}) d\mathbf{z} = P_z(\mathbf{z} \in C)$$

where $P_z(\mathbf{z} \in C)$ denotes the probability of $\mathbf{z}$ being part of $C \in \mathcal{Y} \times \mathcal{X}^D$. $p(\mathbf{z})$ fulfills the usual requirements to a PDF.

Furthermore we define a second PDF $p_x(\mathbf{x})$ as

$$\int_C p_x(\mathbf{x}) d\mathbf{x} = P_x(\mathbf{x} \in C^*)$$

[4] I explicitly thank my advisor for sharing all his great ideas with me and for letting them be part of this thesis.

[5] Although there is obviously a strong dependence between our sample points of a continuous audio signal, this dependence doesn't affect the model building.

where C^* is a subset of $\mathcal{R}^D$. p_z and p_x are not independent but connected by Bayes's-Law for joint probabilities:

$$p_z(\mathbf{z}) = p_z(y, \mathbf{x}) = p_{y|x}(y \mid \mathbf{x}) \cdot p_x(\mathbf{x}). \tag{3.14}$$

In terms of prediction we are most interested in the conditional probability $p_{y|\mathbf{x}}$ which denotes the probability of y given a particular realization $\mathbf{X}$ of $\mathbf{x}$:

$$\begin{aligned} p_{y|x}(y \mid \mathbf{x}) &= p(y \mid X_D, ..., X_1) \\ &= \frac{p(y, X_D, X_{D-1} ..., X_1)}{p(X_D, X_{D-1} ..., X_1)} \end{aligned} \tag{3.15}$$

As estimate for $p_x(\mathbf{x})$ we use a weighted sum of local Gaussian distributions, such that

$$\hat{p}_x(\mathbf{x}) = \sum_{j=1}^{M} \omega_j \cdot \chi_j^x(\mathbf{x}) \quad {}^{6} \tag{3.16}$$

with

$$\chi_j^x(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^D \mid \mathbf{P}_j|}} \cdot e^{-\frac{1}{2}(\mathbf{x}-\vec{\mu}_j)^T P_j^{-1}(\mathbf{x}-\vec{\mu}_j)}. \tag{3.17}$$

$p_x(\mathbf{x})$ describes a D-dimensional Gaussian distribution with a mean vector $\vec{\mu}_j$ and a covariance matrix $\mathbf{P}_j$ for each j. M denotes the total number of Gaussian basis functions. Qua definition $\int_{\mathcal{R}^D} \chi_i^x(\mathbf{x}) d\mathbf{x}$ is normalized to one for all j. ω_j denotes the weight of the local model j. We assure $p_x(\mathbf{x})$ to be normalized by choosing the ω_j such that $\sum_{j=1}^{M} \omega_j = 1$.

The probability estimate $p_z(\mathbf{z})$ has the same overall form as $p_x(\mathbf{x})$:

$$\hat{p}_z(\mathbf{z}) = \sum_{j=1}^{M} \omega_j \cdot \chi_j^z(\mathbf{z}) \tag{3.18}$$

The ω_j as well as the total number of local models M are identical to ω_j and M in expression 3.16. $\chi_j^z(\mathbf{z})$ represents a local estimate of p_z. It is composed of two terms:

$$\chi_j^z(\mathbf{z}) = \chi_j^y(y|\mathbf{x}) \cdot \chi_j^x(\mathbf{x}) \tag{3.19}$$

where $\chi_j^x(\mathbf{x})$ has already been defined as a D-dimensional Gaussian (equation 3.16), whereas $\chi_j^y(y|\mathbf{x})$ is of the form

$$\chi_j^y(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_j^{2y}}} \cdot e^{-\frac{(\Psi_j^y(\mathbf{X})-y)^2}{2\sigma_j^{2y}}} \tag{3.20}$$

$\Psi_j^y(\mathbf{x})$ describes a local function that approximates y given $\mathbf{X}$. We need $\Psi_j^y(\mathbf{x})$ to be bounded and smooth in the domain of influence which is characterized by $\vec{\mu}_j$, σ_j^y, $\mathbf{P}_j$ and ω_j. This condition will be discussed further down once we clearly know the task of $\Psi_j^y(\mathbf{x})$.

In the simplest case $\Psi_j^y(\mathbf{x})$ is a constant function $\Psi_j^y(\mathbf{x}) = \mu_j^y$. For that particular choice

[6] $\hat{p}$ is used for estimated or predicted values whenever the distinction is contextually unclear.

of $\Psi_j^y(\mathbf{x})\ \chi_j^y(y|\mathbf{x})$ reduces to an ordinary one-dimensional Gaussian distribution. However, $\Psi_j^y(\mathbf{x})$ may be chosen arbitrarily complex as a non-linear function, e.g. as a superposition of polynomials or it may become a probabilistic local model by itself. We need to show that $\chi_j^z(\mathbf{z})$ is normalized to one in order to fulfill the requirements of a PDF:

$$\begin{aligned} & \int_{\mathcal{R}\times\mathcal{R}} \chi_j^z(y,\mathbf{x})\ d\mathbf{x}\, dy \\ = & \int_{\mathcal{R}} \int_{\mathcal{R}^D} \chi_j^y(y|\mathbf{x})\ \chi_j^x(\mathbf{x})\ d\mathbf{x}\, dy \\ = & \int_{\mathcal{R}^D} \underbrace{\int_{\mathcal{R}} \chi_j^y(y|\mathbf{x}) dy}_{I(\mathbf{x})}\ \chi_j^x(\mathbf{x})\ d\mathbf{x} \end{aligned} \tag{3.21}$$

$I(\mathbf{x})$ is normalized to 1, because any possible (bounded) value for $\Psi_j^y(\mathbf{x})$ and therefore any possible value for $\mathbf{x} \in \mathcal{R}^D$ converts χ_j^y into a one dimensional Gaussian distribution over $\mathcal{R}$. As this Gaussian is normalized by σ_j^y the integral over $\mathcal{R}$ results in unity. We are left with the outer integral, a multidimensional Gaussian. However, it has already been shown that $\int_{\mathcal{R}^D} 1 \cdot \chi_j^x(\mathbf{x})\ d\mathbf{x} = 1$. Therefore our PDF is perfectly normalized, although that was not obvious at first glance. The above demonstration also clarifies that $\chi_j^y(y|\mathbf{x})$ must indeed be interpreted as a conditional probability $p(y|\mathbf{x})$, as for any given realization $\mathbf{X}$ of $\mathbf{x}$ a PDF over y only is obtained.

We shortly discuss particular choices for $\Psi_j^y(\mathbf{x})$. These choices become really meaningful once we consider the prediction step.

a)

$$\Psi_j^y = \mu_j^y \tag{3.22}$$

If Ψ_j^y is chosen constant for all $\mathbf{x}$, $\chi_j^z(\mathbf{z})$ becomes a $D+1$ dimensional Gaussian PDF, defined by $\vec{\mu}_j^z = (\mu_j^y, \vec{\mu}_j^x)$ and $\mathbf{P}_z$. $\mathbf{P}_z$ is directional only in the $\mathbf{x}$-dimensions, not in y-direction. A further simplification is done by choosing all the Gaussian directional with the space axes. Then $\mathbf{P}_z$ reduces to a diagonal matrix and all the Gaussians become separable. Although this is the actual form of $\mathbf{P}_z$ we are using, we keep describing the model architecture in the general form.

b)

$$\Psi_j^y(\mathbf{x}) = a_j + \mathbf{b}_j \cdot (\mathbf{x} - \vec{\mu}_j) \tag{3.23}$$

$\Psi_j^y(\mathbf{x})$ describes a linear function of $\mathbf{x}$. a_j represents the mean value of χ_j^y which is predicted for $\mathbf{x} = \vec{\mu}_j$. Any deviation of $\mathbf{x}$ from $\vec{\mu}_j$ results in a linear correction of the predicted values.

c)

$$\Psi_j^y(\mathbf{x}) = a_0 + \sum_{k=1}^{K} a_k \cdot \prod_{d=1}^{D} (x^d - \mu_j^d)^{\epsilon_k^d} \tag{3.24}$$

$\Psi_j^y(\mathbf{x})$ is a polynomial function of $\mathbf{x}$ anchored at $\vec{\mu}_j$. This model has already been

subject of discussion as a global estimator with anchor point 0. The parameter search algorithm is naturally derived from the global search as discussed in section 3.2.

We summarize the approximation elements: $p_x(\mathbf{x})$ describes the probability of observing the input-state-vector $\mathbf{x}$, $p_z(\mathbf{z})$ describes the joint probability $p_z(y, \mathbf{x})$ of observing the state vector $\mathbf{x}$ and the output scalar y at the same time. We denote Θ_j^z the set of parameters which describe $\chi_j^z(\mathbf{z})$, including $\mathbf{P}_j$, $\vec{\mu}_j$, σ_j^y and the parameters which describe Ψ_j^y. We denote Θ^z and Θ^x the total set of parameters that characterize p_z and p_x. Note that $\Theta^x \in \Theta^z$. Once p_z is found, p_x is determined as well.

3.3.2 The prediction step

Let's assume a parameter-set Θ^z has been found that describes the training data set reasonably well with respect to some error measure. In order to reconstruct a prediction surface $y(\mathbf{X})$, we need to calculate $p_{(y|x)}(y|\mathbf{X})$ by Bayes' rule:

$$p_{y|x}(y \mid \mathbf{X}) = \frac{p(y, \mathbf{X})}{p(\mathbf{X})} \tag{3.25}$$

$$= \frac{p(y, X_D, X_{D-1}, ..., X_1)}{p(X_D, X_{D-1}, ..., X_1)} = \frac{\sum_{j=1}^{M} \omega_j \cdot \chi_j^y(y, \mathbf{X}) \cdot \chi_j^x(\mathbf{X})}{\sum_{j=1}^{M} \omega_j \cdot \chi_j^x(\mathbf{X})} \tag{3.26}$$

$p_{y|x}(y \mid \mathbf{X})$ evaluates a likelihood for y given a specific input vector $\mathbf{X}$. In order to transform this likelihood into a prediction function we consider three possible solutions:

- $\hat{y} = y \mid \max_y\{p_{y|x}(y \mid \mathbf{X})\}$. It seems to be most natural to choose the particular y which maximizes the conditional probability $p_{y|x}$. In this case the most likely value for y is predicted. In the limit of the 'perfect model', the prediction surface becomes completely deterministic, with $p_{y|x} = 1$ for $y = \hat{y}$.

  However, though intuitive, this choice turns out to be the most difficult and expensive one in terms of computation. Even if we consider the easiest form for $p_z(\mathbf{z})$ with a diagonal covariance matrix $\mathbf{P}_j$ and $\Psi_j^y = \mu_j^y$, the maximization problem could not be solved analytically. There are in fact applications where the resolution is 'poor enough', so that $p_{y|x}$ can be computed for any possible value of y and then the y is chosen that maximizes $p_{y|x}$. This method was used, e.g., for 8 bit representations of image pixels [PP93]. Yet for 16 bit audio samples this method is prohibitive, especially as each $p_{y|x}$ is computationally expensive by itself.

- $\hat{y}$ could be obtained by a random draw from $p_{y|x}(y \mid \mathbf{x})$. On the one hand this approach reintroduces stochasticity into sound-prediction, after some characteristic noise has been eliminated in the learning process. On the other hand this system would assure deterministic behavior in the deterministic limit, as any draw from a deterministic

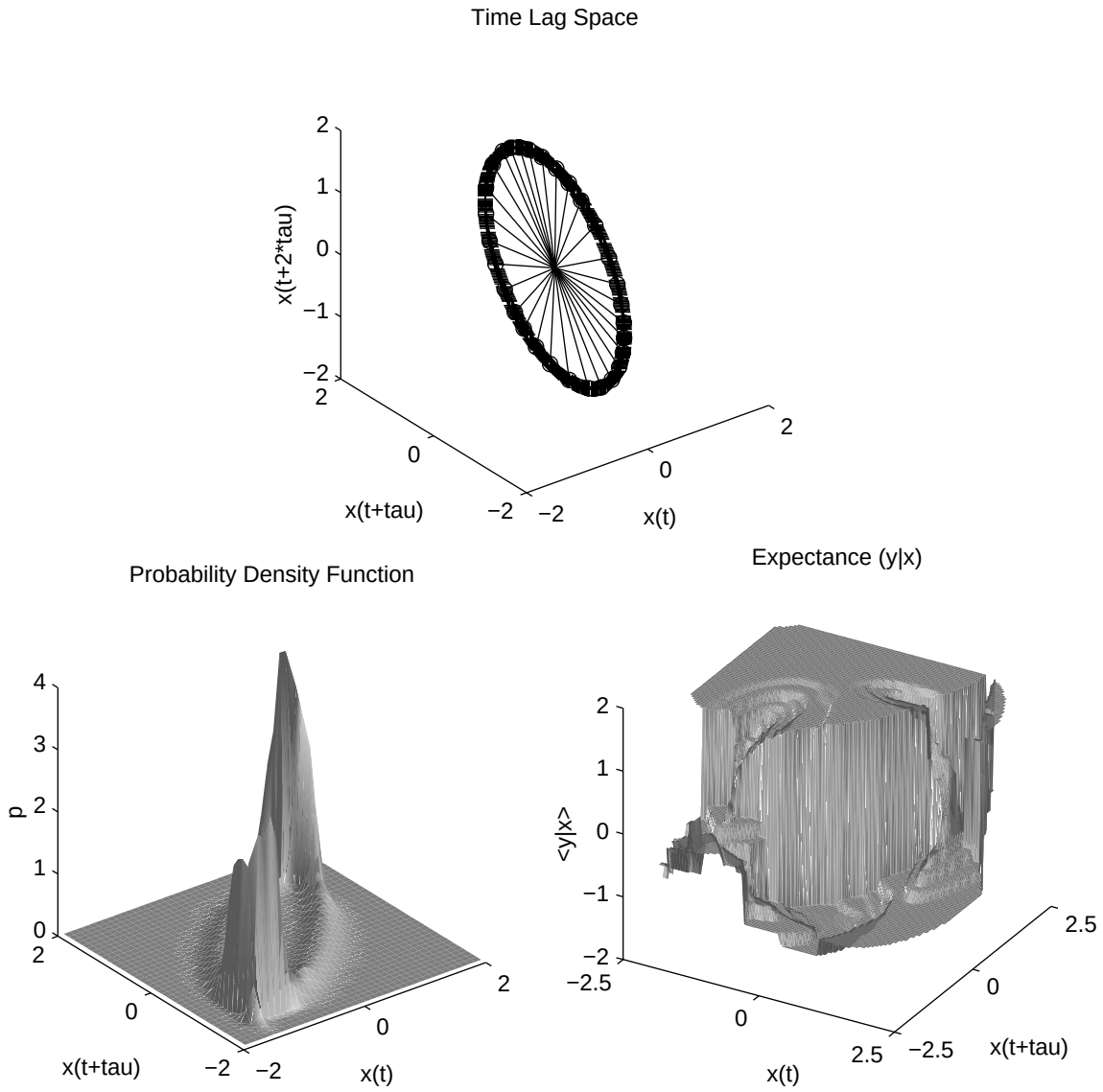

Figure 3-2: (a) Sinusoid in the reconstructed time-lag-space. The axis define cluster centers on the orbit. (b) Probability density function of the sinusoid. (c) Expectation $< p_{y|x} >$ of the sine wave

probability function results in the one value y for which $p_{y|x} = 1$. However, the random draw is computationally expensive as well, as a random number generator as well as the inverse function of $P_{y|x}$ is needed. We do not know whether the perception would justify this computational effort.

- $\hat{y} = E\{p_{y|x}(y \mid \mathbf{X})\} = \int_{\mathcal{R}} y \cdot p_{y|x}(y \mid \mathbf{X})\, dy$. It may look like the least obvious solution to use the expected value of y for $\hat{y}$. Yet, this approach has been exploited for this work for two major reasons:

  First, given our special form of $p_{y|x}$, the expectation of y is computationally most efficient to determine. For all considerable local models the expression for $< p_{y|x} >$ reduces to a simple weighted sum over local experts.

  Secondly, one could be concerned that $< p_{y|x} >$ may not coincide with the maximum value of p due to a bimodal form of the PDF. Given an arbitrary PDF such as shown in figure 3-1(b) $\hat{y} =< p >$ does not correlate at all with the two likely values for y. However, remember that we chose an embedding dimension which assures $y(\mathbf{x})$ to lay on a single valued surface. Therefore a situation as shown in figure 3-1(b) should never occur. Or, from the opposite point of view, we could state that, if such a situation occurs, the embedding dimension has been chosen badly.

  Briefly, given the Gaussian structure of our model, we expect $p_{y|x}$ to have one global maximum that coincides with $< p_{y|x} >$, as shown in figure 3-1(a). The experiments we did proved this assumption valid.

Which is the particular form of $< p_{y|x} >$, given our specific form of p_x and p_z?

$$\begin{aligned} < p_{y|x} > &= \int_{\mathcal{R}} y \cdot p_{y|x}\, dy \\ &= \int_{\mathcal{R}} y \frac{\sum_{j=1}^{M} \omega_j\ \chi_j^y(y, \mathbf{X})\ \chi_j^x(\mathbf{X}) dy}{\sum_{j=1}^{M} \omega_j\ \chi_j^x(\mathbf{X}R)} \\ &= \frac{\sum_{j=1}^{M} \int_{\mathcal{R}} y\ \chi_j^y(y, \mathbf{X}) dy\, \omega_j\ \chi_j^x(\mathbf{X})}{\sum_{j=1}^{M} \omega_j\ \chi_j^x(\mathbf{X})} \end{aligned} \tag{3.27}$$

We define a local predictor $\hat{y}_j$ as

$$\begin{aligned} \hat{y}_j &= \int_{\mathcal{R}} y \cdot \chi_j^y(y, \mathbf{X}) dy \\ &= \int_{\mathcal{R}} y \cdot \frac{1}{\sqrt{2\pi\sigma_j^{2y}}} \cdot e^{-\frac{(\Psi_j^y(\mathbf{X}) - y)^2}{2\sigma_j^{2y}}}\, dy \\ &= \Psi_j^y(\mathbf{X}), \end{aligned} \tag{3.28}$$

and obtain a weighted sum over all local predictors

$$\hat{y} =< p_{y|x} >= \frac{\sum_{j=1}^{M} \Psi_j^y(\mathbf{X}\, \omega_j\, \chi_j^x(\mathbf{X})}{\sum_{j=1}^{M} \omega_j\, \chi_j^x(\mathbf{X})}. \tag{3.29}$$

$\sum_{j=1}^{M} \omega_j \cdot \chi_j^x(\mathbf{X})$ represents the partition function of the local models. It is going to play a major part in the learning algorithms (EM).

For $\Psi_j^y = \mu_j^y$ and $\mathbf{P}_j = \vec{\sigma}_j^x$ (diagonalized $\mathbf{P}_j$) this expression becomes a sum over constant values, weighted by separable Gaussian distributions.

$$\hat{y} = \frac{\sum_{j=1}^{M} \mu_j^y \cdot \omega_j \cdot \prod_{d=1}^{D} \chi_j^d(X_d)}{\sum_{j=1}^{M} \omega_j \cdot \prod_{d=1}^{D} \chi_j^d(X_d)} \tag{3.30}$$

with

$$\chi_j^d(x_d) = \frac{1}{\sqrt{2\pi\sigma_j^{d2}}} \cdot e^{-\frac{(X_j^d - \mu_j^d)^2}{2\sigma_j^{d2}}}.$$

For a linear function Ψ_j^y we obtain

$$\hat{y} = \frac{\sum_{j=1}^{M} (a_j + \mathbf{b}_j \cdot \mathbf{X}) \cdot \omega_j \cdot \prod_{d=1}^{D} \chi_j^d(X_d)}{\sum_{j=1}^{M} \omega_j \cdot \prod_{d=1}^{D} \chi_j^d(X_d)} \tag{3.31}$$

At this point we reconsider the stability of our system for point prediction. We need $\hat{y}$ to be finite for any bounded vector $\mathbf{x}$. Obviously we can assure that by choosing $\Psi_j^y(\mathbf{x})$ to be bounded for all possible $\mathbf{x}$. Unfortunately this is about all we can say in general. As we are dealing with partition functions, it is clear, that even for a vector $\mathbf{X}$ which is 'far away' from $\vec{\mu}_j$ expert j may nevertheless be in charge of that $\mathbf{X}$. In fact for any $\mathbf{x}$ with $\|\mathbf{x}\| \to \infty$ there is a expert that becomes stronger than all the others and therefore is in charge of prediction. This expert should rather be bounded!

Also the stability problem for iterated prediction that occured with polynomial models in section 3.2, arises again with cluster weighted modeling. There may be a way to tune a model such that intrinsic instabilities of the different experts add up to a global stable model even for iterated prediction. However, that solution seems to be quite difficult to achieve. To solve the problem of iterated prediction, there are basically two possibilities:

1. Choose $\Psi_j^y(\mathbf{x}) = \mu_j^y$. For this special case we know, that clusters are stable. Yet, we give up the powerful behavior of local models.

2. Choose $\Psi_j^y(\mathbf{x})$ such that it decays to a fixed value $(\mathbf{x}_j - \vec{\mu}_j) \to \infty$. We did not test this class of functions. Yet, it looks like the class of functions that work with cluster weighted modeling are those that are bounded or periodic, e.g. sigmoidal functions. Further work should be done on this topic.

Another possible solution could be a linear feedback which compares the amplitude of the output to a desired value and works on the amplitude of the iterated space coordinates.

To illustrate the basic steps of cluster based prediction, figure 3-2 shows the prediction steps for a sine wave. A sine wave becomes a flat ellipsoid in the time lag space (3-2(a)). It' orbit is approximated by clusters which expand a PDF(3-2(b)). The PDF can be used for prediction or to construct a prediction surface on $\mathbf{x}$ (3-2(c)).

3.3.3 Learning and Search

We believe our data to be represented or generated by classes, the local models. Only we do not know how these classes look like. We do not know their principle structure, the structure that all local models have in common. Neither do we know their particular configuration characterized by the local parameter set Θ_j^z. Our priors are not only determined by belief in the most powerful local basis function but also by considerations of computational practicability, search algorithms and convergence behavior. Once again, the best prior we could possibly realize is the one, that helps us to reproduce the best violin sound, given our resources. Philosophical considerations may only be helpful to fulfill our perceptional task.

In the following sections we review two search methods of unsupervised learning and extend them for the use with cluster weighted local experts. Both of the proposed algorithms, clustering by melting and the EM algorithm, are least square search algorithms which minimize the systems entropy with respect to the data and the unknown model parameters. As there are plenty of analogies between the search techniques and a statistical mechanics language we use this pre-formulated terminology whenever possible and reasonable.

3.4 Clustering Data by Melting

We assume Θ_j^y constant ($\Theta_j^y = \mu_j^y$) for now. The extension for arbitrary Θ_j^y becomes easy with the EM algorithm in the next section. Whenever we are talking about clusters in the following, we implicitly mean classes as well. We assume a cluster/class to be an internal, hidden state of our system that is 'responsible' for some subset of the observed data. There is no physical meaning or justification behind this prior[7]. Hidden states are no more than a convenient way to describe the optimization problem of fitting basis functions to some data. They can be seen as a metaphor.

The basic principles that characterize clustering data by melting are the following:

- Clusters do not interact. They try to describe local data with respect to their proper likelihood maximum.
- A priori there are no specific assumptions concerning the data distribution. An arbitrary cost function (energy cost) has to be defined. We denote $E_{i,j}$ the energy associated with the assignment of data point $\mathbf{z}_i$ to cluster center $\vec{\mu}_j^z$.
- Clusters which happen to describe the same data and share the same center are melted. The starting number of clusters is therefore higher than the final number of model

[7] It is in fact remarkable how little common machine learning literature notices the fact that nature usually does not describe states at the scaling of clusters.

clusters. In fact we can only hope to obtain a good partitioning of our space if we lose enough clusters due to melting during the iteration steps.

- Clustering data by melting only finds $\vec{\mu}_j^x$ and μ_j^y. It does not serve as search for the ω_j and the $\mathbf{P}_j$. Those variables must be found afterwards. There is no distinction between cluster directions during the search.

Theoretical Foundations and the Thermodynamic Analogy

We denote $E_{i,j}$ the energy associated with the assignment of data point z_i to cluster C_j. We denote $p_{i,j}$ the relative probability factor of point x_i being generated by cluster C_j, considering one single cluster C_j only. We then obtain an average total cost associated with our system:

$$< E >= \sum_{j=1}^{M} \sum_{i=1}^{N} p_{i,j} \cdot E_{i,j} \tag{3.32}$$

Expression 3.32 serves as the boundary condition for the data distribution. We assign a certain total energy or temperature to our system that is directly related to $< E >$. Furthermore we use the principle of maximum entropy to find a stable distribution at each iteration step. Given the constraint 3.32, the $p_{i,j}$ that maximize the total entropy

$$H = -\sum_{j=1}^{M} \sum_{i=1}^{N} p_{i,j} \log_2(p_{i,j}) \tag{3.33}$$

are Boltzmann distributions of the form

$$p_{i,j} = e^{-\beta E_{i,j}}/Z_j \tag{3.34}$$

where Z_j denotes the partition function

$$Z_j = \sum_{i=1}^{N} e^{-\beta E_{i,j}}, \tag{3.35}$$

and β denotes the Lagrange multiplier of the minimization problem. It is determined by the choice of $< E >$. In the thermodynamic analogy it is proportional to $1/T$. Therefore decreasing β implies increasing activity and disorder in the system, whereas a high β freezes the system and lets only the closest data points influence a cluster.

Pushing the thermodynamic analogy ahead we define the effective cost function F. As we assume independence between clusters and between the $p_{i,j}$ of different clusters, we define F_j as the free energy associated with one cluster C_j [8]:

$$F_j = -1/\beta \cdot \log Z_j \tag{3.36}$$

[8] Note that the thermodynamic free energy consists of two terms. We will find the second term in the next section in the context of a convergence proof.

The condition for a minimum of the effective cost naturally becomes

$$\Delta_{\vec{\mu}_j} F_j = 0, \forall j. \tag{3.37}$$

This gives us j conditions for our j clusters centers $\vec{\mu}_j$ which we could exploit for any cost function $E_{i,j}$. It is only now that we choose a particular form for $E_{i,j}$, the square distance:

$$\begin{aligned} E_{i,j} &= \mid \mathbf{z}_i - \vec{\mu}_i^z \mid^2 \\ &= \mid y_i - \mu_i^y \mid^2 + \mid \mathbf{x}_i - \vec{\mu}_i^x \mid^2 \end{aligned} \tag{3.38}$$

Plugging expression 3.39 into equation 3.37 we end up with an implicit equation for $\vec{\mu}_j^z$:

$$\vec{\mu}_j^z = \sum_{i=1}^{N} \frac{\mathbf{z} e^{-\beta(\mathbf{z}-\vec{\mu}_j^z)^2}}{\sum_{i=1}^{N} e^{-\beta(\mathbf{z}-\vec{\mu}_j^z)^2}}, \tag{3.39}$$

Solutions for $\vec{\mu}_j^z$ cannot be computed analytically, however, they can be obtained by fixed point iterations of map 3.40:

$$\vec{\mu}_j^z(n+1) = \frac{\sum_{i=1}^{N} \mathbf{z} e^{-\beta(\mathbf{z}-\vec{\mu}_j^z(n))^2}}{\sum_{i=1}^{N} e^{-\beta(\mathbf{z}-\mu_j^z(n))^2}} \tag{3.40}$$

3.40 is iterated until stability of $\vec{\mu}_j^z$. This iteration process assures convergence in a local minimum with respect to the initial conditions and to a certain β. β translates our notion of the scaling of the data. It also reflects the number of clusters which we want to represent the data.

In order to obtain a reasonable partition of the data set we start with a high β and a high number of clusters. We then iterate until convergence, remove one of those clusters that have become identical, decrease β and iterate again. Ideally the complete set of data points is used to initialize the clusters, in order to assure any data point being covered by the model. In praxis fewer clusters may be enough, if chosen carefully.

We propose the complete melting algorithm [Wo93]:

1. Initialize clusters by assigning a cluster to each of the data points or by assigning a reasonable number of clusters to a subset of data points.

2. Choose β_{max} such that the system is quasi frozen and set $\beta = \beta_{max}$.

3. Iterate according to map 3.40 n times or until convergence.

4. Keep only one cluster of subsets of clusters that have become identical.

5. If there is too many clusters left, decrease β and go to 3.

For any value of β the converged state represents an entropy minimum. However, neither model nor data alone tell us which of these states best describes the data. In fact one of the weak points of this search algorithm is the fact that there are many parameters left that

must be tuned 'by hand'. Important parameters such as β_{max}, the final number of clusters and the decay factor for β must be fixed from experience or by cross-validation. This is particularly critical, as the algorithm by itself is already computationally expensive. Also, these cross-validation iterations do not go very well with the idea of unsupervised learning.

Once the cluster centers are found we need to determine the remaining parameters of Θ^z. The first approach that we consider is based on the concept of hard-clustering and the K-Means Algorithm [DH73]. We do not review these algorithms, but remind the reader that in a hard-clustering framework every data point is assigned with probability $p_{i,j} = 1$ to exactly one cluster and with probability $p_{i,j} = 0$ to each of the other clusters. $p_{i,j}$ is one for the j that minimizes the distance $d_{i,j}$ where $d_{i,j}$ stands for the Euclidean distance. If we calculate $p_{i,j}$ for each i and j, we define j subsets of points C_j^z. A point $\mathbf{z}_i$ belongs to C_j^z if and only if $p_{i,j} = 1$. The sum over the elements in our subsets therefore is the same as the total number of points.

Given the C_j^z we calculate ω_j and $\mathbf{P}_j$ as follows. The $\mathbf{P}_j$ are assumed to be diagonal and therefore can be expressed by $\vec{\sigma}_j^{2z}$ [9]. This assumption seems to be reasonable, because the clustering algorithm does not take into account directional resolution of clusters at all.

$$\begin{aligned} \omega_j &= \frac{N_j}{N} \\ \vec{\sigma}_j^{2z} &= \frac{1}{N_j} \sum_{\mathbf{z} \in C_j^z} (\mathbf{z}_j - \vec{\mu}_j)^2 \end{aligned} \tag{3.41}$$

where N_j denotes the cardinality of the subset of points associated with cluster C_j.

With the spirit of soft clustering in mind a second method to calculate ω_j and σ_j^{2z} is considered. Once again a β is chosen which we belief represents the scaling of our training data set and the number of clusters we obtained. E.g., the β may be used that had been used for the last melting iteration. Then a probability $\alpha_{i,j}$ is calculated that expresses the total probability of a point i being generated by a cluster j.

$$\alpha_{i,j} = \frac{e^{-\beta(|\mathbf{z}_j - \vec{\mu}_j|^2}}{\sum_j e^{-\beta(|\mathbf{z}_j - \vec{\mu}_j|^2}} \tag{3.42}$$

is characterizing a soft assignment between each data-point and each cluster. The parameter-set is then calculated as follows

$$\begin{aligned} \omega_j &= \frac{\sum_{i=1}^N \alpha_{i,j}}{\sum_{j=1}^M \sum_{i=1}^N \alpha_{i,j}} \\ \vec{\sigma}_j^x &= \frac{\sum_{i=1}^N \mid \mathbf{x}_j - \vec{\mu}_j^x \mid^2 \cdot \alpha_{i,j}}{\sum_{i=1}^N \alpha_{i,j}} \\ \sigma_j^y &= \frac{\sum_{i=1}^N \mid y_j - \mu_j^y \mid^2) \cdot \alpha_{i,j}}{\sum_{i=1}^N \alpha_{i,j}} \end{aligned} \tag{3.43}$$

[9] Fully directional $\mathbf{P}_j$s would require an eigenvalue decomposition of the data-subset owned by a cluster.

Neither of the two proposed methods is better from a theoretical point of view. And in fact none of the two methods gives the right results for the parameters we were looking for. Fortunately we need not care too much about how well clusters overlap, as eventually they are only used for weighting in the prediction step. However smooth overlapping becomes important in border regions between two basis functions where clusters are supposed to interpolate.

In order to guarantee a smooth probability function, variances may be tested with uniform data distributions. Then an additional factor κ varies the variances such that the estimated distribution becomes as uniform as possible.

$$\begin{aligned} \tilde{\vec{\sigma}}_j^{2x} &= \kappa \cdot \vec{\sigma}_j^{2x} \\ \tilde{\sigma}_j^{2y} &= \kappa \cdot \sigma_j^{2y} \end{aligned} \tag{3.44}$$

This method is empirically motivated. In [PP93] $\kappa \approx 1.5$ was found as a reasonable value for κ. In our own experiments, we usually used $\kappa = 1$ for the above mentioned reason that our main goal consists in weighting local models, not in most properly approximating the state-density.

3.5 The EM Algorithm

The EM (Expectation-Maximization) algorithm is another iterative technique for maximum likelihood approximation of training data. It has been applied to simple unsupervised learning in pattern recognition and machine learning problems. It was generalized to more complex systems such as hierarchical expert models [JJ93] and to Hidden Markov Models. Each iteration of EM consists of an Estimation(E) and a Maximization(M) step. The E-step evaluates a probability distribution for the data given the model parameters form the previous iteration. The M step then finds the new parameter set that maximizes the probability distribution. In a generalized EM algorithm (GEM) the M step does improve the data likelihood but is not iterated until full convergence.

We review the EM algorithm in its simple non-hierarchical form for Gaussian basis functions and extend it to an application with cluster weighted local modeling. We show how the extended version of EM fits into the thermodynamic analogy and why convergence to local minima is assured by conventional proofs.

The EM algorithm is based on the idea that there is a set of 'hidden' variables that relates the data to the unknown states (clusters). If those 'hidden' or missing variables were known, the maximum likelihood distribution would be easy to calculate. As they are unknown we need a fancier technique to reconstruct the probability estimate. We try to be consistent with the notation used so far and denote furthermore:

$\mathcal{Z}$: the observed data.

$\mathcal{W}$: the unobserved data. Note that what the unobserved data actually represents is not made explicit. This lack of clearness may bother the reader in the beginning. However,

eventually it turns out to be very handy.

$\mathcal{V}$: the complete data ($\mathcal{Z} + \mathcal{W} = \mathcal{V}$).

Θ^z: The set of variables relating $\mathcal{Z}$ and $\mathcal{W}$.

We assume a joint probability $p(\mathcal{Z}, \mathcal{W} \mid \Theta^z)$ for $\mathcal{Z}$ and $\mathcal{W}$, conditioned by Θ^z. Summing over W we obtain a marginal probability for $\mathcal{Z}$ $p(\mathcal{Z} \mid \Theta^z) = \sum_{\mathcal{W}} p(\mathcal{Z}, \mathcal{W} \mid \Theta^z)$. Ultimately we are searching for the set Θ^z, that maximizes $p(\mathcal{Z} \mid \Theta^z)$ and therefore also maximizes the log-likelihood $\log p(\mathcal{Z} \mid \Theta^z)$ [NH93]. The E and the M step are repeated successively with increasing n:

E-step: Calculate the complete data-likelihood given the known data $\mathcal{Z}$ and the current set ${\Theta^z_{n-1}}$[10].

$$\tilde{p}_n(\mathcal{W} = P(\mathcal{W} \mid \mathcal{Z}, \Theta^{n-1}) \tag{3.46}$$

M-step: Calculate the Θ^n that maximizes the expectation of the joint probability $p(\mathcal{W}, \mathcal{Z} \mid \Theta)$.

$$\Theta^n = \max_{\Theta} E(p(\mathcal{W}, \mathcal{Z} \mid \Theta)) \tag{3.47}$$

The concrete update rules of E- and M-step will be discussed further down.

Back to the Thermodynamic Analogy

Let's recall some results from the last section. We defined a partition function Z_j and the corresponding free energy F_j of a single Cluster in equation 3.36. Minimizing (3.37) led to the implicit expression 3.39 for $\vec{\mu}_j^z$. In order to get interaction between clusters and in order to find an overall entropy minimum, instead of single Z_js now consider the total partition function

$$Z = \prod_{j=1}^{M} Z_j \tag{3.48}$$

and the corresponding effective cost function

$$\begin{aligned} F &= -\frac{1}{\beta} \cdot \log Z \\ &= -\frac{1}{\beta} \sum_{j=1}^{M} \cdot \log Z_j \end{aligned} \tag{3.49}$$

[10] Concretely we need to calculate the following conditional probabilities, that relate a specific data point and a specific cluster. We denote this probability $\alpha_{i,j}$. The meaning of $\alpha_{i,j}$ will become clear in the following.

$$\alpha_{i,j} = \frac{p_{i,j}}{\sum_{j=1}^{M} p_{i,j}} \tag{3.45}$$

where $p_{i,j}$ has been defined in equation 3.34.

Again we specify the cost function $E_{i,j}$ as the square distance between data point $\vec{x}_i$ and cluster C_j. However, as we do not assume a special form of Ψ_j^y we need to find a special arrangement for the y-direction of our space. We define

$$E_{i,j} = \mid \vec{x}_i - \vec{\mu}_i^x \mid^2 + \mid \Theta_j^y(\vec{x}_i) - y_i \mid^2 \tag{3.50}$$

Given this cost function we obtain

$$p_{i,j} = \frac{e^{-\beta[|\vec{x}_i - \vec{\mu}_i^x|^2 + |\Psi_j^y(\vec{x}_i) - y_i|^2]}}{\sum_{i=1}^N e^{-\beta[|\vec{x}_i - \vec{\mu}_i^x|^2 + |\Psi_j^y(\vec{x}_i) - y_i|^2]}} \tag{3.51}$$

and furthermore

$$\begin{aligned} F &= -\frac{1}{\beta}\sum_{j=1}^M \log Z_j \qquad (3.52)\\ &= -\frac{1}{\beta}\sum_{j=1}^M \log \prod_{i=1}^N e^{-\beta E_{i,j}} \\ &= -\frac{1}{\beta}\sum_{j=1}^M \log \prod_{i=1}^N e^{-\beta[|\vec{x}_i - \vec{\mu}_i^x|^2 + |\Psi_j^y(\vec{x}_i) - y_i|^2]} \end{aligned}$$

Deriving this expression with respect to the μ_j^d we obtain $d \times M$ equations for the $\vec{\mu}_j^x$:

$$\frac{\partial F}{\partial \mu_j^d} = \sum_{i=1}^N \frac{(x_i^d - \mu_j^d) \cdot e^{-\beta E_{i,j}}}{\sum_{j=1}^M e^{-\beta E_{i,j}}} = 0 \tag{3.53}$$

Unfortunately equation 3.53 cannot be solved analytically. However, we may obtain a solution for μ_j^d by fixed point iterations of the following map:

$$\mu_j^d(n+1) = \frac{\sum_{i=1}^N \frac{x_i^d \cdot e^{-\beta E_{i,j}}}{\sum_{j=1}^M e^{-\beta E_{i,j}}}}{\sum_{i=1}^N \frac{e^{-\beta E_{i,j}}}{\sum_{j=1}^M e^{-\beta E_{i,j}}}} \tag{3.54}$$

In 3.54 we find back the expression for $\alpha_{i,j}$ as used in equation 3.45 The update rule for $\vec{\mu}_j$ becomes a weighted sum over the locations of the data points. As opposed to the update rules in the melting algorithm, now there is interaction and repulsion between clusters. Clusters share points, but the assignment of points is such that the total weight of any one point is one. The update rule can be written as

$$\vec{\mu}_j^x(n+1) = \frac{\sum_{i=1}^N \mathbf{x}_i \cdot \alpha_{i,j}}{\sum_{i=1}^N \alpha_{i,j}} \tag{3.55}$$

with

$$\alpha_{i,j} = \frac{p_{i,j}}{\sum_{j=1}^M p_{i,j}}$$

and

$$p_{i,j} = e^{-\beta E_{i,j}}.$$

In order to find the parameters of Ψ_j^y we need to go back to the definition of F. Let's denote Θ_j^Ψ the set of parameters which characterize Ψ_j^y, with $\Theta_j^\Psi = \{\nu_j^1, \nu_j^2, ..., \nu_j^K\}$. We now derive F with respect to the parameters ν.

$$\frac{\partial F}{\partial \nu_j^k} = \sum_{i=1}^{N} \frac{(\Psi_j^y(\vec{x}_i, \nu_j^1, \nu_j^2, ..., \nu_j^K) - y_i) \cdot \frac{\partial \Psi_j^y}{\partial \nu_j^k} \cdot e^{-\beta E_{i,j}}}{\sum_{j=1}^{M} e^{-\beta E_{i,j}}} = 0 \tag{3.56}$$

For 3.56 to be solved a particular choice for Ψ_j^y has to be made. In section 3.3.1 three major possibilities to express Ψ_j^y have been defined. For $\Psi_j^y = \mu_j^y$ there is only one more parameter μ_j^y to be calculated, in perfect analogy to equation 3.55. For the local linear model, as defined in equation 3.23 we obtain a condition for a_j and d conditions for $\mathbf{b}_j$.

$$\frac{\partial F}{\partial a_j} = \sum_{i=1}^{N} \frac{(y_i - \mathbf{b}_j \mathbf{x}_i - a_j) \cdot e^{-\beta E_{i,j}}}{\sum_{j=1}^{M} e^{-\beta E_{i,j}}} = 0 \tag{3.57}$$

$$\frac{\partial F}{\partial b_j^d} = \sum_{i=1}^{N} \frac{(y_i - a_j - \mathbf{b}_j^* \mathbf{x}_i^*)\, x_i^d \cdot e^{-\beta E_{i,j}}}{\sum_{j=1}^{M} e^{-\beta E_{i,j}}} = 0 \tag{3.58}$$

with

$$\mathbf{b}_j^* = [b_1, b_2, ..., b_{d-1}, b_{d+1}, ...b_D]$$

$$\mathbf{x}_i^* = [x_1, x_2, ..., x_{d-1}, x_{d+1}, ...x_D]$$

We transform the implicit form of expression 3.57 and 3.58 into fix point iterations and obtain

$$a_j(n+1) = \frac{\sum_{i=1}^{N} (y_i - \mathbf{b}_j \mathbf{x}_i) \cdot \alpha_{i,j}}{\sum_{i=1}^{N} \alpha_{i,j}} \tag{3.59}$$

$$b_j^d(n+1) = \frac{\sum_{i=1}^{N} (y_i - a_j - \mathbf{b}_j^* \mathbf{x}_i^*) \cdot x_i^d \cdot \alpha_{i,j}}{\sum_{i=1}^{N} (x_i^d)^2 \cdot \alpha_{i,j}}$$

Finally we also give the final fix-point map for the general polynomial model as defined in 3.24.

$$a_j^{k'}(n+1) = \frac{\sum_{i=1}^{N} (y_i x_i^d - a_j - \sum_{k \neq k'} a_j^k)(\prod_{d=1}^{D} x_{id}^{n_{k'}^d}) \cdot \alpha_{i,j}}{\sum_{i=1}^{N} (\prod_{d=1}^{D} x_{id}^{2 n_{k'}^d}) \cdot \alpha_{i,j}} \tag{3.60}$$

The Final Update Rules

The equations and update rules we developed so far depend on the scaling factor β. Yet, we want the clusters to discover the scaling of the system autonomously. Also clusters should be flexible enough to dominate regions of different sizes. Therefore the original definition

of $p_{i,j}$ (3.34) is replaced by

$$p_{i,j} = \chi_j^y(y_i|\mathbf{x}_i) \cdot \chi_j^x(\mathbf{x}_i) \tag{3.61}$$

with

$$\chi_j^x(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^D|\mathbf{P}_j|}} \cdot e^{-\frac{1}{2}(\mathbf{x}-\mu_j)^T P_j^{-1}(\mathbf{x}-\mu_j)}$$

and

$$\chi_j^y(y \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma_j^{y2}}} \cdot e^{-\frac{(\Psi_j^y(\mathbf{x})-y)^2}{2\sigma_j^{y2}}}.$$

We also need to reformulate the expression for $\alpha_{i,j}$, the probability of data point $\mathbf{x}_i$ being generated by cluster C_j.

$$\alpha_{i,j} = \frac{\chi_j^y(y_i \mid x_i) \cdot \chi_j^x(x_i)}{\sum_{j=1}^M \chi_j^y(y_i \mid x_i) \cdot \chi_j^x(x_i)} \tag{3.62}$$

Now every cluster has its own directional scaling properties. These properties may change during the EM-iterations as the scaling parameters are updated themselves.

In the practical application we use a diagonal covariance matrix, such that $\mathbf{P}_j = \mathbf{I} \cdot \vec{\sigma}_j^{2x}$. We are now ready to propose the remaining M-step update rules. $\vec{\sigma}_j^{2z}$ follows directly from 3.55:

$$\begin{aligned} \vec{\sigma}_j^{2x}(n+1) &= \frac{\sum_{i=1}^N (\mathbf{x}_i^x - \vec{\mu}_j^x)^2 \cdot \alpha_{i,j}}{\sum_{i=1}^N \alpha_{i,j}} \\ \sigma_j^{2y}(n+1) &= \frac{\sum_{i=1}^N (y_i - \mu_j^y)^2 \cdot \alpha_{i,j}}{\sum_{i=1}^N \alpha_{i,j}} \end{aligned} \tag{3.63}$$

The cluster weight is supposed to reflect the number of points the cluster is owning, which can be expressed as follows:

$$\omega_j(n+1) = \frac{\sum_{i=1}^N \alpha_{i,j}}{\sum_{j=1}^M \sum_{i=1}^N \alpha_{i,j}} \tag{3.64}$$

We introduce a further factor γ, that turned out very useful in the practical implementation.

$$\tilde{\alpha}_{i,j} = \alpha_{i,j}^\gamma, \gamma \in \mathcal{R}^+ \tag{3.65}$$

The convergence rate of the model parameters in some cases turned out to be fairly bad. However, particular choices of γ for particular data increased convergence speed and accuracy of the model significantly. The values for γ are usually between 1 and 2, but the optimal γ may even be smaller than 1 or $\gg 10$. Any $\gamma > 1$ makes decisions clearer, because an increased γ lets $\tilde{\alpha}_{i,j}$ behave like a sharper and sharper step function. This shape seems to enforce faster convergence, with the trade off that clusters might get stuck to early. γ does not really fit in the theoretical framework we proposed. However empirically it turned out to be very important.

An Alternative Update

We proposed a method to calculate the parameter set of the local functionals Θ_j by fixed point iterations, which can be derived within the EM formalism. However, there is a non-iterative maximum likelihood solution which is based on the same idea as the global polynomial approximation in section 3.2.

Our starting point is once again the derivative of the free energy:

$$\frac{\partial F}{\partial a_j^{k'}} = \sum_{i=1}^{N} \frac{p_{i,j} \cdot (y_i - \sum_{k=1}^{K} a_j^k \prod_{d=1}^{D} x_{id}^{n_{jd}^k}) \cdot \prod_{d=1}^{D} (x_{id}^{n_{jd}^{k'}})}{p_{i,j}} = 0 \tag{3.66}$$

We use the convention

$$\sum_{i=1}^{N} \frac{p_{i,j} \cdot \Phi(i,j)}{p_{i,j}} = < \Phi(i,j) >_i$$

and denote furthermore

$$\Pi^k = \prod_{d=1}^{D} x_{i,d}^{n_{j,d}^k}$$

$$\Pi^0 = 1.$$

so that a_j^0 denotes the coefficient of the constant term. Equation 3.66 becomes

$$0 = < y_i \cdot \Pi^{k'} > - (\sum_k \Pi^k) \cdot \Pi^{k'} > \tag{3.67}$$

$$= < y_i \cdot \Pi^{k'} > - < a_0 \Pi^0 \Pi^{k'} > - < a_1 \Pi^1 \Pi^{k'} > - ... - < a_k \Pi^k \Pi^{k'} > \tag{3.68}$$

and we end up with a system of K equations for K coefficients:

$$\vec{0} = \begin{bmatrix} < y_i \cdot \Pi^0 >_i \\ < y_i \cdot \Pi^1 >_i \\ . \\ . \\ < y_i \cdot \Pi^K >_i \end{bmatrix} - \vec{a} \cdot \begin{bmatrix} < \Pi^0 \Pi^0 >_i & < \Pi^1 \Pi^0 >_i & ... & < \Pi^K \Pi^0 >_i \\ < \Pi^0 \Pi^1 >_i & < \Pi^1 \Pi^1 >_i & ... & < \Pi^K \Pi^1 >_i \\ ... & ... & ... & ... \\ < \Pi^0 \Pi^K >_i & < \Pi^1 \Pi^K >_i & ... & < \Pi^K \Pi^K >_i \end{bmatrix}$$

$$= < y_i \cdot \vec{\Pi}_j > - \vec{a} \cdot \underline{\eta}_j \tag{3.69}$$

$\underline{\eta}_j$ denotes the weighted covariance matrix with respect to cluster j. It has the same form as **A** in section 3.2 for the global polynomial model, except that this time the influence of any single point on a particular local model is weighted. In the limit of a one cluster model $\underline{\eta}_j$ becomes identical to **A** in equation 3.13. We invert $\underline{\eta}$ and obtain

$$\vec{a}_j = \underline{\eta}_j^{-1} \cdot < y_i \cdot \vec{\Pi} > \tag{3.70}$$

Using this method optimal values for $\vec{a}$ are evaluated at each iteration. It therefore may be considered to determine Θ_j^y only occasionally after multiple updates of the ordinary cluster

parameters. Alternatively, the parameter search for the clusters could be done in the input space only first. The local functions would then be evaluated with the cluster centers kept fixed.

Convergence

The system is supposed to converge towards a likelihood maximum of the model given the data, which corresponds to an entropy maximum of the clustered space. As the iterative EM algorithm consists of two different steps, total convergence is proved once convergence of either one step is proved.

It has been shown by Neal and Hinton [NH93] that there is a functional form $F(\tilde{P}, \Theta)$ that relates the E and the M step with respect to a general error term

$$\begin{aligned} F(\tilde{p}, \Theta) &= E_{\tilde{p}}[\log p(\mathcal{W}, \mathcal{Z} \mid \Theta)] - E_{\tilde{p}}[\log \tilde{p}(\mathcal{W})] \\ &= E_{\tilde{p}}[\log p(\mathcal{W}, \mathcal{Z} \mid \Theta)] + H(\tilde{p}) \end{aligned} \tag{3.71}$$

where $\mathcal{Z}$ is the observed data and $\mathcal{W}$ describes the hidden parameters. The first term of F corresponds to the E-step and to the average energy in the thermodynamic analogy. The second term corresponds to the M step and the system's thermodynamic Entropy. Ignoring a change of sign the functional $F(\tilde{p}, \Theta)$ is equivalent to the physical free energy F.[11] Finally an expression has been found that describes both terms of F and the analogy seems to be completed, which is nice from a systems theoretical point of view and which is very helpful in terms of understanding EM. Due to the change of sign we need to maximize F.

Neal and Hinton showed that any partially or completely performed E or M step results in an independent increase (stagnation for converged systems) of the corresponding term in $F(\tilde{p}, \Theta)$. They also proved that any such iteration also increases the log−likelihood $\log p(\mathcal{Z} \mid \Theta)$, which forces the system towards a local maximum of $\log p(\mathcal{Z} \mid \Theta)$.

We do not review the proof in this paper, but we would like to know how it relates to our extension of cluster based density estimation. In fact the proof remains valid because we expressed any extension of Cluster modeling as a change in the energy cost function. We recall that we defined our proper $E_{i,j}(\Theta_j)$ as opposed to an ordinary square distance energy. This simple change in the cost function lets the principle proof unchanged, although the final model looks different from an ordinary cluster model.

Unfortunately we did not succeed in expressing γ in an equivalently compact way. We do not know its theoretical effects on the stability, but we got to know it as an empirically very useful additional parameter.

[11] The definition of the free energy we have in mind is

$$F = W - TS \tag{3.72}$$

where W denotes the energy, S the entropy and T the temperature of the system.

3.6 Empirical Tests

It has been shown that the proposed algorithms converge towards some sort of 'best solution'. Unfortunately, the theoretical proofs have little to do with practical implementation issues. Nothing can be said a priori about convergence speed, about the necessary number of iterations until convergence, nor about the approximation error. Therefore, a set of test functions was used, in order to compare our algorithms quantitatively.

We define the following test functions of the form $\mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$. All the data is either normalized to the support $[-1,1] \times [-1,1]$ or to zero mean and unit variance:

Uniform Probability Density Function: $p(x,y) = constant$. $p(x,y) = 1/4$ for $\mid x \mid, \mid y \mid \leq 1$, $p(x,y) = 0$ every where else. The data is seeded on a lattice for $x, y \in [-1,1]$. This function mainly serves to calibrate the Gaussian Distributions. It represents the 'difficult case' of perfectly symmetric data.

Gaussian Distributions: Data drawn from a two dimensional Gaussian distribution is seeded around five anchor points (x_g, y_g), $g = 1,..,5$, ($\sigma^{2g}_{x,y} = 1$), 100 points per Gaussian). z is calculated as a linear function $f_g(x,y) = a^g + b^g_x \cdot x + b^g_y \cdot y$. The following parameters have been used:

	x^g	y^g	a^g	b^g_x	b^g_y
1	0.5	0.5	0.5	1	0
2	-0.5	-0.5	0.5	0	-1
3	0.5	-0.5	-0.1	0.5	0
4	- 0.5	0.5	0.1	0	-0.5
5	0	0	0	1	1

The whole data set is noted, in order to test whether the polynomial model recovers the original parameters exactly.

Step Function: $z = f(x,y) = 1$ for $\mid x \mid, \mid y \mid \leq 1$, $z = f(x,y) = 0$ everywhere else. The data is seeded randomly on $x, y \in [-1,1]$.

Sinc Function: $z = \frac{\sin 3\pi R}{3\pi R}$, with $R = \sqrt{x^2 + y^2}$. 4000 data points are seeded randomly on $x, y \in [-1,1] \times [-1,1]$. See figure 3-5.

Polynomials $z = f(x,y) = 0.5 \cdot x + y - 0.5 \cdot xy + 0.25 \cdot x^3 - 0.25 \cdot y^3$. The data is randomly seeded on $x, y \in [0,1] \times [0,1]$.

For the polynomial-based approximation as well as for the cluster based EM approximation the following criteria have been used:

- Number of basis terms M. For the EM approximation this number corresponds to the number of clusters, for the polynomial approximation it corresponds to the number of polynomial terms.

Data	Number of Points	Number of Clusters	γ	Iterations until Convergence	Maximal Error	Average Error	Remarks
Uniform PDF	441	33	1.2	53	0	0	
Gaussians	500	5	0.9	16	0.1	$8\,10^{-3}$	Centers are perfectly found by the Clusters. Tuning gets easier for more than 5 clusters (see figure 3-3 and 3-4
step function	2000	100	1	49	0,4	$3\,10^{-3}$	
sinc	4000	200	0.9	30	$3\,10^{-2}$	$6\,10^{-4}$	see figure 3-5)
polynomials	1000	200	1.2	21	$2\,10^{-3}$	$3\,10^{-2}$	

Table 3.1: Experimental results of the cluster based approximation. The local model is reduced to a constant mean ($\Psi_j^y = \mu_j^y$).

- Square error. The predicted value is compared to the real value and then averaged over the whole data set. $E^2 = 1/N \cdot \sum_{i=1}^{N} (\hat{z} - z)^2$. As the data sets are normalized on the same range for x, y and z, the error itself is not normalized.

With the values obtained by the cluster based approximation trained by EM, we also give the number of iterations until convergence and the specific value of γ (see equation 3.65). Convergence is assumed, once the average change of any of the model parameters is smaller than 10^{-5}. See table 3.1 for cluster based approximation without local model and table 3.2 for cluster based approximation with a local linear model.

For the Polynomial Approximation the value of the regularizing parameter λ (see section 3.2) is given with the average error (see table 3.3).

Clearly cluster based modeling is much more flexible than polynomials. Polynomials behaved well with data drawn from polynomials. In fact the parameters of the Gaussian distribution were all recovered. Yet, polynomials are unable to explain highly nonlinear behavior, such as the sinc or the step function.

Cluster models easily discovered and approximated strange behavior in the solution space. On the other hand many basis terms are needed, to approximate such extreme behavior. 200 clusters for a sinc function on the range of 2π means quite a bit of resources. Also, the EM algorithm needs a lot of tuning, until a stable and reasonable solution can be found. It is sensitive to the initial value of the Cluster Variances, to the number of Clusters and also to γ. Using linear local functions the algorithm did not converge at all for some of the test functions. Clusters do not find stable space positions all over the space, but tend to converge towards few identical space domains.

It has been mentioned in the beginning of this chapter that there is the alternative of explicit parameter search as opposed to the EM algorithm. Stability would probably be less of a problem with such explicit search algorithms. We are considering using such algorithms for the more complex models we will have to built.

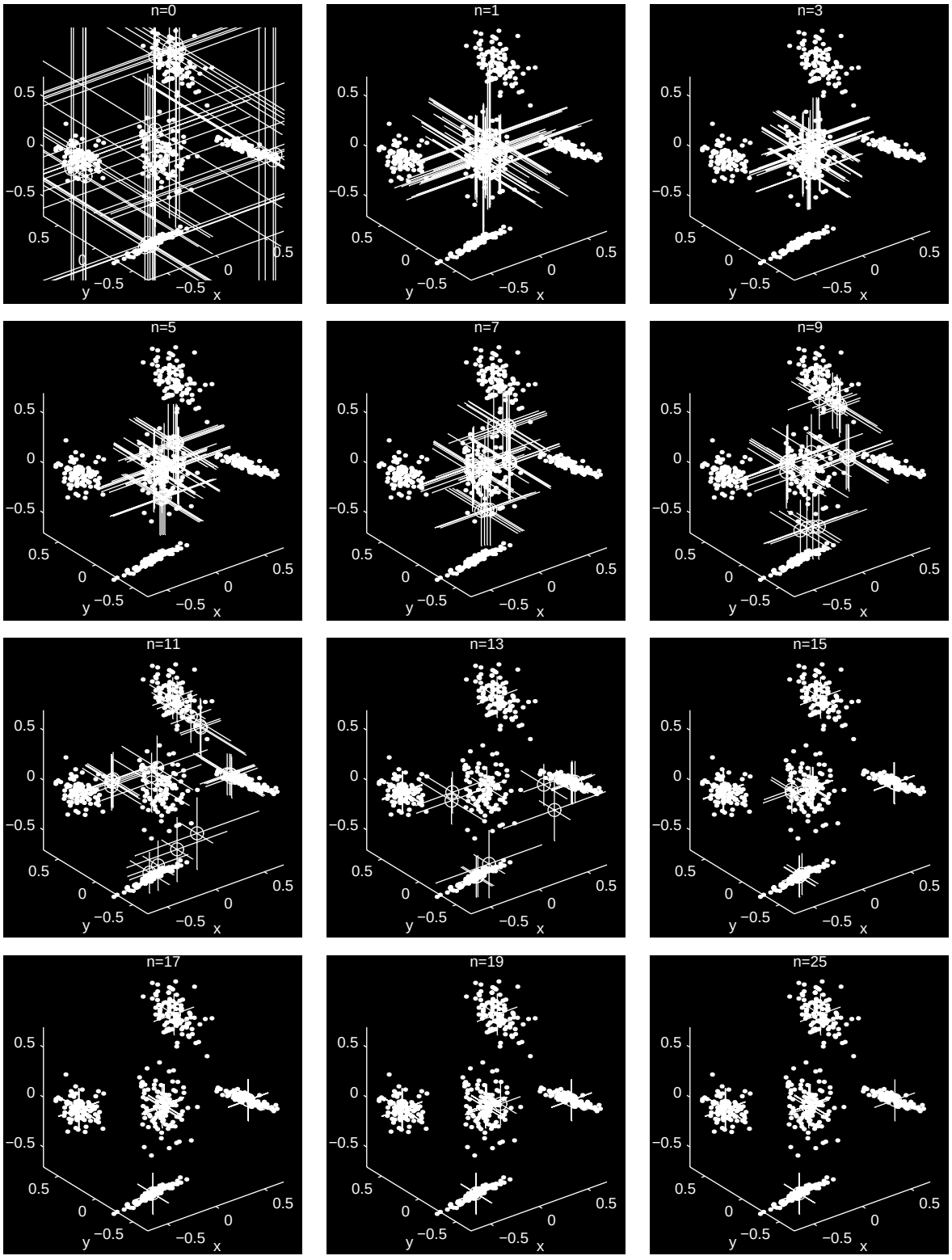

Figure 3-3: EM iterations for data distributed on five different means. Clusters are represented by small circles. The corresponding variances are represented by lines in the space direction.

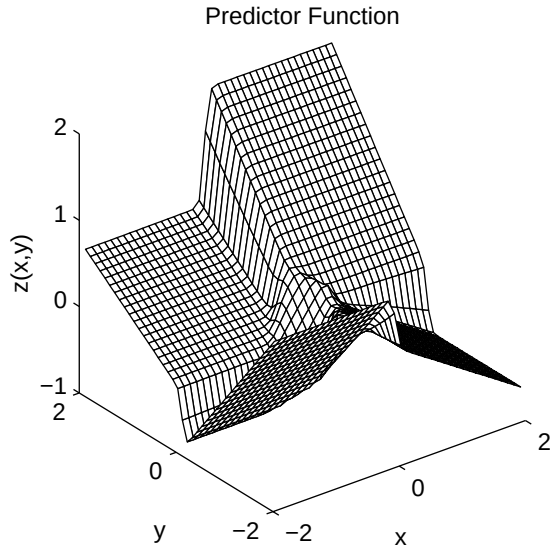

Figure 3-4: Prediction Surface of the Gaussian data approximated by locally linear models. With EM all the data centers and the linear slopes in z-direction are recovered.

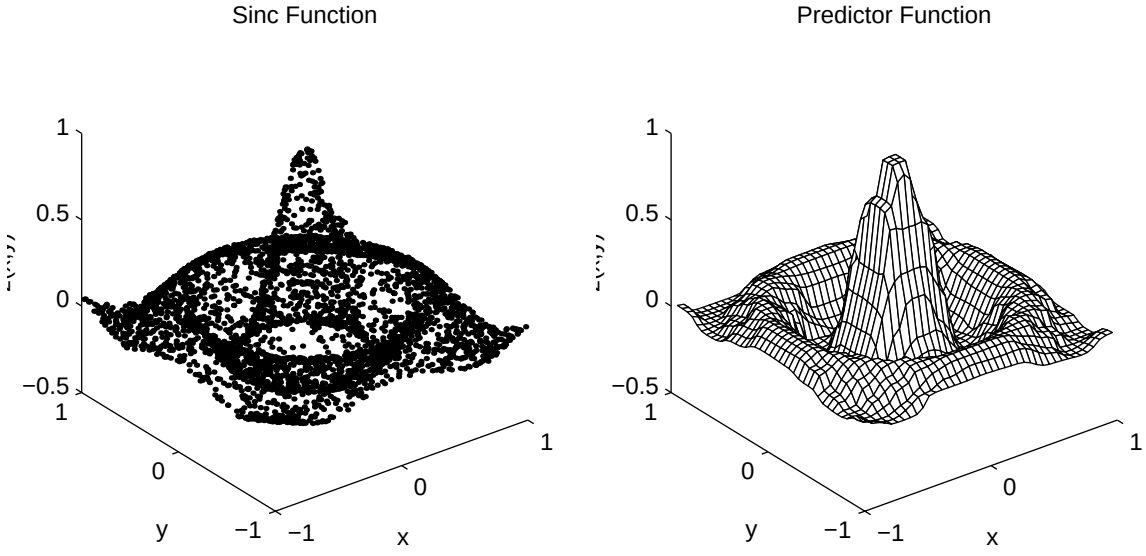

Figure 3-5: (a) data drawn from a sinc function (4000 points) and (b) a cluster based approximation (400 clusters) of the data.

Data	Number of Points	Number of Clusters	γ	Iterations until Convergence	Maximal Error	Average Error	Remarks
Uniform PDF	441	17	1.2	49	0	0	
Gaussians	500	10	1.2	22	$1.5\ 10^{-3}$	$2.6\ 10^{-5}$	The clusters find back the exact parameters of the Gaussians. There is 5 different type of clusters. For doubled clusters, usually one takes over and the others decay to 0.
step function							no need for local models
sinc							algorithm is not stable
polynomials							good performance

Table 3.2: Experimental results with a cluster based approximation. A local linear model is used ($\Psi_j^y = a_j + \mathbf{b}_j \cdot \mathbf{x}$).

Data	Number of Points	Order	Number of Polynomials	λ	Average Error	Remarks
Uniform PDF						perfectly linear (no challenge)
Gaussians	500	4	15	1	$4\ 10^{-3}$	good interpolation
step function						bad performance
sinc function	4000	10	66	100000	0.013	very bad performance. Approximation does not 'detect' the pic at $x, y = 0$.
polynomials	1000	3	10	1	0	All the parameters of the original polynomials are perfectly recovered (see table above).

Table 3.3: Experimental results with the polynomial approximation

Chapter 4

Working with the Violin

> 4.014 Die Grammophonplatte, der musikalische Gedanke, die Notenschrift, die Schallwellen, stehen alle in jener abbildenden internen Beziehung zu einander, die zwischen Sprache und Welt besteht. Ihnen allen ist der logische Bau gemeinsam.
> (Wie im Märchen die zwei Jünglinge, ihre zwei Pferde und ihre Lilien. Sie sind alle in gewissem Sinne Eins.)
> L. Wittgenstein, TRACTATUS

In this chapter the experimental work is described which had been done with real world data. We define various prediction tasks, and use them to test the algorithms proposed in the last chapter. As the global approach turned out to be computationally too expensive, given the hardware resources we used so far, we restricted our test to data sets of 1/10 to 30 seconds, depending on the specific problem. In the last section of this chapter we propose some measures that should make an overall model possible, in terms of computation as well as stability.

See Appendix A for a description of the hardware that has been used to sample violin data. These devices include the instruments themselves and the special violin bow, designed to measure bowing data.

4.1 The Input Space

Our choice of input variables has to some extent been conditioned by the available hardware equipment. However, the sensor data we are going to use for the final model should be very similar to the data we used so far. We used the the bow-position relative to the strings (x-position), the bow-placement relative to the bridge (y-position) and the bow-pressure [Appendix A]. From a performer's point of view it is clear that these inputs are not independent. Any combination does not result in a reasonable reaction of the instrument.

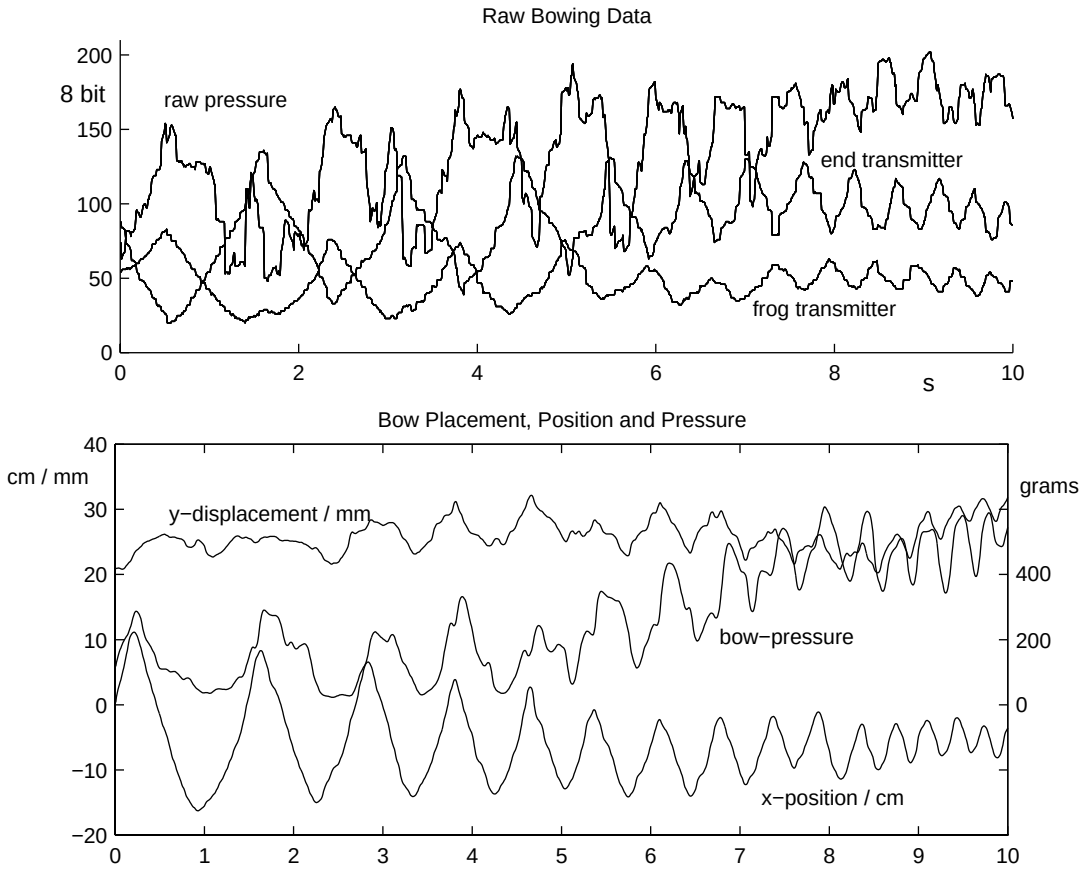

Figure 4-1: (a) Raw input data and (b)inferred and calibrated input signals for a 5 seconds sample going from slow to fast détaché

The basic dependency is the following: An increase of bow velocity $\frac{\partial x}{\partial t}$ implies a decrease of bow pressure, an increase of the distance bow/bridge y and a decrease of volume. A decrease of y implies a decrease of $\frac{\partial x}{\partial t}$, a higher bow pressure and an increasing volume. It follows that some information given by the three time series overlaps. However, as we know from chapter 2 state space and input lag-dimensions can be replaced by other observables of the system. Using inputs which are not completely independent therefore helps save time lag dimensions.

Yet, there are other degrees of freedom associated with the violin bow that are to be considered: e.g., the vertical angle between the plane defined by bow bar and bow hair and the strings represents an important control parameter for the player. Although this angle is not independent from bow pressure and y-position it most probably contains some information that we have not access to yet. Also the musician controls the angle between bow and finger board, in order to change the y-position of the bow. Again, though not independent from the input series we do measure, we might need additional information about this particular angle for a more accurate model.

We have not taken into account a change of pitch in our model so far, which might look

like the biggest restriction of our model at that point. However, the behavior of the violin at different pitches could be interpreted as independent. Given that assumption the complete model becomes a sum of models for each possible pitch or each possible finger position. Although we would prefer an integral solution that includes pitch as a space dimension at the same description level as the other inputs, there are compromise solutions such as a one pitch/one model solution or a pitch warping approach.

Figure 4-1 shows the three raw input signals of a 10 second sound example and the extracted time series $x(t)$, $y(t)$ and pressure. The signals are normalized, calibrated and filtered by a causal moving-average-filter. See Appendix A for the calibration relations.

4.2 Prediction in the Steady State

The cluster based prediction model was applied to simple sinusoidal functions and to superpositions of harmonic sine waves (section 3.3). In either of these cases discrete and closed trajectories in the state-space were obtained. We are now considering an interval of a violin signal that is short enough such that decay effects due to dissipation as well as energy increase due to the driving forces are neglectable. We embed such a sound piece in a 5-dimensional lag-space. In one space-dimension the signal itself is presented, in the four remaining dimensions time-lagged audio signals are represented. The model is built on 1000 to 3000 data points, 200 to 400 clusters and locally constant predictor functions (see section 3.3). $\Theta_j^y = \mu_j^y$ was in fact the only choice that did not cause stability problems. The model was built with the EM algorithm. It worked fine for the electrical as well as for the acoustical violin.

Figure 4-2 shows the embedded signal in a 3-dimensional state-space. Figure 4-3 shows some periods of the training signal and the corresponding re-synthesized signal in the same time scale. The re-synthesis is done with the state vector initialized to zero. The signal takes some time to settle down on the attractor, but then the signal shape as well as pitch are preserved perfectly.

The number of clusters necessary depends on the complexity of the particular piece of sound serving as training data, but it has to be bigger than the number of samples per period. In our example we have ~ 50 samples per period (440 Hz, 22050 samples/s) and 200 clusters were used for the density approximation.

As mentioned before there is a trade off between stability and extrapolation. The cluster model performs well and stable as only points are predicted which are part of the orbit. Any 'strange' initial condition is attracted by the orbit immediately. In this sense the approximated trajectory is working literally like a 'state-space-attractor'. On the other hand the model is not able to extrapolate for input constellations which have not been part of the training set. For that reason it turned out to be impossible to use the cluster based model for transitory prediction with input dimensions. The input clears the orbit as shown in figure 4-4(b). As opposed to this image see figure 4-4(a), where no input were used to embed 4000 sample points. Clearly the latter state space representation cannot be used for prediction whereas the orbit in figure 4-4(b) seems to be clear and usable. Unfortunately an

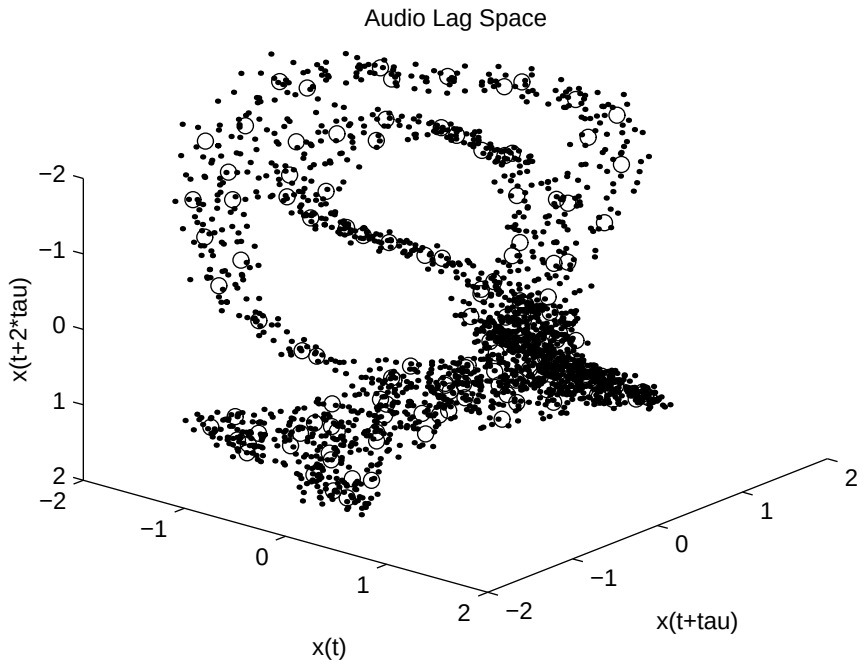

Figure 4-2: Steady state violin signal embedded in the three dimensional lag space ($\tau = 2$)

enormous amount of data and clusters is needed to do a reasonable cluster representation of the input orbit.

4.3 Polynomial Point Prediction

How well does the polynomial approximation technique (section 3.2) with real world data? We worked with sets of 1000 to 4000 sample points, 6 and 7 dimensional state-spaces and 6th to 8th order polynomials. The regularizer λ was varied between 0.01 and 1000. In general, 5 space dimensions for the audio signal and its time lags were used and one or two space dimensions for the input signals. The input $v(n)$ is proportional to the bow velocity. $v(n) = x(n) - x(n-k)$ represents a differential expression of the smoothed bow position.

Our concerns about polynomial models are confirmed. The iterated re-synthesis of the audio signal decays to zero after few iteration steps during which the system oscillates. apparently the energy supply provided by one or two inputs is not sufficient to keep the system on an oscillatory track. On the other hand more than seven space dimensions are computationally not affordable.

However, the function approximation was tested in the point prediction mode, in order to get an idea of the quality of approximation in general: a set of sample data is considered that has been embedded in lag space. All the elements of this data vector are stored except the non-lagged output dimension. We then predict the one output dimension with our model and compare the predicted values to the recorded output. Obviously we do not forecast with this procedure, but we gain insight into the quality of our predictor function

as well as in perceptional properties of re-synthesized sound.

Figures 4-5 and 4-6 show the results for one particular sound example at two different time scales. At the time scale of a basis period of the signal, the approximation does not very well. The reconstructed wave form is not smooth at all and creates artifacts. Yet, it conserves the basic frequency properties. Globally the reconstructed system looks pretty well. The model reproduces the original amplitude changes as well as the deviation from zero-mean. Also the model extrapolates amazingly well at the global time scale. Figure 4-5 indicates the interval of data that has actually been used for training. Although not even the whole amplitude range is covered by the training data, these never seen states of the violin have been perfectly reconstructed.

Perceptionally the signal is very close to the original sound. Apparently the local distortion does not affect the overall violin characteristics. This result confirms the known fact from research in musical perception: the characteristics of musical instruments are determined by the amplitude of the waveform envelope rather than by the composition of its frequency spectrum. Certainly the specific overtone series has impact on the sound characteristic, especially in quasi stationary regimes. Yet, the real distinction comes from the instrument's global behavior in the time domain. The main difference between a violin and a piano is the way the player interacts and controls the strings. This interaction is reflected by the overall waveform envelope.

The choice to keep the input device violin bow is therefore confirmed to be right. The other important result we got from polynomial point predictions concerns the partition of our resources: In order to obtain the perceptionally best model, we may concentrate our efforts on global characteristics rather than small local details of the wave form.

4.4 Prediction of the Envelope

We noticed so far, that the different time scales of the violin signal are difficult to match within a single model. The input series are unable to 'communicate' the driving energy to the system and to the output. Given furthermore that the perceptionally predominant characteristic of the signal is the wave envelope, we consider concentrating on predicting the waveform envelope separately. An envelope predictor could then be combined with a wave generator to synthesize a complete signal.

As can be seen from the time domain representation in figure 4-7, the overall shape of the violin signal is characterized by a deviation from zero mean, due to a temporarily constant component of the sound pressure, and by the energy envelope of the sound wave. We denote by $d(t)$ the deviation from zero mean and by $e(t)$ the energy envelope of the signal and define these two time series as follows:

$$\begin{aligned} d(t) &= \frac{1}{T}\int_{t-T}^{T} s(\tau)d\tau \\ e(t) &= \frac{1}{T}\int_{t-T}^{T} |s(\tau) - d(\tau)|^2 d\tau \end{aligned} \tag{4.1}$$

where $s(t)$ is the original signal and T is a constant which typically is of the order of the input sampling period. In the practical implementation these definitions were replaced by the following time discrete extraction rules, which reflect the same ideas, but are easier to calculate:

$$\begin{aligned} d(nT) &= \max_s\{s(\theta)\} + \min_s\{s(\theta)\} \\ e(nT) &= \max\{|\ s(\theta) - d(nT)\ |\} \end{aligned} \tag{4.2}$$

where T denotes the sample interval of the input signals and θ stands for the time interval $[(n-1)T, nT]$.

Figures 4-7 shows the original signal, the extracted deviation from zero mean and the extracted amplitude/energy. In figure 4-7 also the time synchronous series of bow velocity and bow pressure are shown. The correlation between the signals to predict and the inputs is significant, which creates hope for prediction. The envelope $e(nT)$ appears extremely sharp. It goes approximately to zero whenever bowing direction is changed. This extreme behavior indicates that a polynomial approximation of the envelope predictor might not be a very good idea. Indeed the attempts to reconstruct $e(nT) = f(v(nT), p(nT))$ with the polynomial model ended with rather poor results. However, the cluster based model dealt very well with the particular shape of $e(nT)$. For stability reasons we used only time lags on the input signals. Our 5-dimensional state vector was defined as

$$\mathbf{z}(nT) = \{e(nT), v(nT), v((n-k)T), p(nT), p((n-k)T))\}.$$

Using k=20, we reconstructed the two signals as shown in figures 4.4. The bow change edge appears slightly smoothed, but most of the details are detected and reproduced by the model. The reconstruction of out-of-sample signals of the same bowing type works as fine.

The two reconstructed signals were then recombined with a steady state violin wave $\tilde{s}(t)$ as synthesized in section 4.2. The reconstruction rule for $\hat{y}(t)$:

$$\hat{y}(t) = \tilde{s}(t) \cdot \tilde{e}(t) + \tilde{d}(t) \tag{4.3}$$

where $\tilde{e}(t)$ and $\tilde{d}(t)$ are filtered and re-sampled (audio sample rate) versions of $e(nT)$ and $d(nT)$ (figure 4.4). As expected $\hat{y}(t)$ perceptionally is clearly identified as a violin although the local waveform is maintained the same over the whole 5s sample. The reconstructed sound sounds 'cleaner' than the original. It is, e.g., missing the attack noise which is, though noise, an important part of the characteristics and the beauty of the violin tone. We hope to recover these characteristics of the local spectrum by differentiating between local experts.

4.5 Results, Problems and Potential Solutions

The experiments confirmed most of the preliminary assumptions and model priors. Most important, it has been shown empirically in the point prediction mode that the single valued prediction surface, foreseen by the embedding theorem exists. It has furthermore

been shown that cluster based iterated prediction models satisfy high stability criteria. Also it became clear that details of the violin waveform are perceptionally less important than the envelope behavior.

The experiments also clarified some problematic issues which imply a correction of the original approach: Although the embedded function exists, it seems to be computationally impossible to approximate it in a global state space. The measures proposed in the following will be part of the final model as developed in chapter 5.

4.5.1 Multi-Scale Space Splitting

There are obviously different time scales associated with the device violin. The waveform itself has a typical scaling at the audio sampling rate. The smallest perceptional resolution of a played note lies in the 10 ms domain. Other time scales can be identified, such as the scaling of a musical phrase or the formal structure of a movement. We are not interested in the latter elements, because they represent intentional supersets of smaller elements, which are intrinsic to the physics or control possibilities of the instrument. Yet, the distinction between the control and forming of a tone and the actual waveform could be very helpful for prediction and modeling.

Given the specific boundary conditions, work at different time scales seems to be even more appropriate, as we dispose already of two different sampling periods. The signals describe either the input observables or the audio signal. We recall that input/output embedding requires the input and the output signals to behave homogeneously. Homogeneous is not clearly defined, but we noticed that the physical 'communication' between input and output in our model was rather poor. We therefore might try to find some intermediate state space that selects and conditions lower order state spaces. For the data set to become computable we consider hard decisions, either in form of input partitioning (see next section) with hierarchical approximation techniques or in form of hierarchical spaces.

In terms of space hierarchy we propose a general framework as sketched in figure 4.5.1. The expert system comes close to the framework of gated experts [WMS95] and of hierarchical experts [JJ93]. Yet, there is some significant difference. Both of the cited model architectures are basically unsupervised learning approaches for prediction problems, that let the network partition the space autonomously. Partitioning is mainly done in the input space. No a priori meaning is associated with different experts. As opposed to this unsupervised approach to expert learning we believe that there are meaningful global states that characterize the system at the input scaling and that condition the subsystem which is working at the audio sample rate.

Let's denote:

T_{lt}: long term sampling period.

T_{st}: short term sampling period.

$\vec{x}_{lt}(l)$: Input vector (including time lags of the of the long term expert and of the final system's output).

$\vec{w}_{st}^{sel}(l)$: Internal vector characterizing meaningful long term behavior.

$\vec{w}_{st}^{in}(m)$: Input vector of the short term experts.

Q: total number of short term experts.

E_{st}^{q}: short term expert q.

y_i: output of expert E_{st}^{q}.

$\vec{w}_{lt}^{sel}(l)$ and $\vec{w}_{lt}^{in}(l)$ represent vectors of meaningful system characteristics, e.g. the energy level of the system. $\vec{w}_{lt}^{sel}$ and $\vec{w}_{lt}^{in}$ are predicted given the input $\vec{x}_{lt}(l)$. Iterative prediction is possible but should be avoided for stability reasons. Based on the state vector $\vec{w}_{lt}^{sel}(l)$ a local expert E_{st}^{q} is chosen that works at a faster sampling period. The expert E_{st}^{q} predicts the output y. The input of E_{st}^{q} $\vec{w}_{st}^{in}(m)$ consists of time lags of y, the global input $\vec{x}_{lt}(l)$ and the interior states $\vec{w}_{lt}^{in}(l)$. The elements of $\vec{w}_{lt}^{in}(l)$ may differ from those of $\vec{w}_{lt}^{in}(l)$. All interactions between the network agents are described within a soft probability model, which can easily be reduced to a hard decision version. Although we have a cluster based estimate in mind, the model works with any probabilistic estimate. The overall probability we are interested in is $p(y(m) \mid \underline{x}_{st})$

$$p(y(m) \mid \underline{x}_{st}) = \sum_{q=1}^{Q} p(E_{st}^{q} \mid \underline{x}_{lt}) \cdot p(y_i|\underline{x}_{lt}) \tag{4.4}$$

$$= \sum_{q=1}^{Q} p(E_{st}^{q} \mid \underline{w}_{lt}^{sel}) \cdot p(\underline{w}_{lt}^{sel} \mid \underline{x}_{lt}) \cdot p(y_i \mid \underline{w}_{lt}^{in}) \cdot p(\underline{w}_{lt}^{in} \mid \underline{x}_{lt}) \tag{4.5}$$

For prediction we would exploit $p(y(m) \mid \underline{x}_{st})$ in terms of maximization or in terms of the expected value. Obviously $p(y(m) \mid \underline{x}_{st})$ is calculated as a sum over a chain of conditional probability densities, where each PDF depends on the previous model unit only. We noted $\underline{x}$ and $\underline{w}$ as matrices, in order to indicate that theoretically any past value of $\vec{x}(l)$ and $\vec{w}(l)$ could be taken in account. Concretely each conditional PDF is based on a time-lagged-structure.

How can the system be trained? Clearly we are still looking for a maximum likelihood approximation of the data. A general learning algorithm in form of the EM algorithm or in form of explicit parameter search could be defined, but becomes almost impossible to understand without a concrete application (see [JJ93]. We therefore prefer to present a learning framework once a specific model architecture is defined (chapter 5).

4.5.2 Hierarchical Cluster Structures

As has been seen, our learning algorithms suffered from the fact, that all the training points and all the basis functions were related. Compared to the $M \times N$ proportionality of the proposed algorithms other parameters such as dimensionality and iterations have

relatively little effect on the computing time. More precisely the following computational cost proportionalities were found:

Approx.Technique/Learning Algorithm	cost proportionality
Polynomial Model	$M^2 \times N \times D$
Clustering by Melting	$M \times N \times D \times n_I \times n_\beta$
EM (no local model)	$M \times N \times D \times n_I$
EM (local model)	$M \times N \times D \times K_l^2$

with the Number of basis functions (clusters) M, the number of data points N, the dimension D, the number of iterations n_I (n_β) and the number of local basis functions K_l. For which ever model is used, these dependencies have to be reduced for the violin application, with regard to learning as well as to synthesis.

A commonly used method to organize huge data sets for classification and search are binary-trees. We do not review tree structures, but assume that the reader is familiar with their principle structure. Common trees do not fit very well our soft data structure. Yet, we consider hierarchical cluster trees with arbitrarily soft boundaries. In the original approach all clusters interact with all points. However, due to the Gaussian structure characterizing the domain of influence, some points do not matter at all for 'far away' clusters. Unfortunately we have no idea which points are 'far away', as long as we haven't calculated their distance or the probability of being associated with a cluster. In order to escape from this dilemma, the following tree oriented cluster architecture is proposed. We denote:

m : Number of Clusters per node.

$1+\epsilon$: relative number of points per layer with respect to the total number of points N.

N : Total number of points.

M : Total number of clusters at the lowest tree level.

$\alpha_{i,j}$: Probability of data point i being generated by cluster j.

The tree architecture is built up according to the following steps. For the clustering steps any cluster algorithm may be used.

1. $n = 0$
2. Cluster the data set with m clusters until convergence.
3. Evaluate the $\alpha_{i,j}$ and assign each point to the cluster j that maximizes $\alpha_{i,j}$. Assign those points to a second cluster j that share the next highest $\alpha^*_{i,j}$ with j^*, such that the number of points assigned becomes $(1+\epsilon)N$.
4. Store the clusters and their subsets of assigned points.

5. $n = n + 1$.

6. If the total number of points per cluster is too big go to 1. and do 1. for all m subsets of points.

The algorithm is meant to be recursive. Any subset of points serves as root for m new branches of the tree. It stops as soon as the cardinality of the point subsets becomes small enough, or as soon as the total number of clusters in the last layer of the tree corresponds to the desired number of clusters to describe the fine structure of the data set. Therefore the architecture assures the same fine resolution of description as non hierarchical clusters. Although we do have hard assignments between points and clusters, clusters overlap softly. The degree of softness is tuned by ϵ. A high ϵ results in slower convergence speed, but let subspaces of points and clusters overlap better.

The computation cost for the EM based search now becomes proportional to

$$\begin{aligned} Cost\,(hierarchical\,EM) &\propto N \cdot D \cdot [1 + (1+\epsilon) + (1+\epsilon)^2 + ...(1+\epsilon)^L] \cdot n \qquad (4.6) \\ &\leq N \cdot D \cdot n \frac{(1+\epsilon)^{L+1}}{\epsilon} \end{aligned}$$

where $L = \frac{\log M}{\log m}$.

Let's consider a data set of a million sample points, which corresponds to $30s$ audio signal. Furthermore 10^5 clusters are assumed to describe the lowest hierarchy level, which is realistically needed for a reasonable approximation. If we choose $m = 5$ and $\epsilon = 0.1$ our gain becomes

$$\begin{aligned} \frac{Cost\,(ordinary\,EM)}{Cost\,(hyrarchical\,EM)} &= \frac{M}{m} \frac{\epsilon}{(1+\epsilon)^{\frac{\log M}{\log m}+1}} \\ &\simeq 1000 \end{aligned}$$

The computational gain applies for the synthesis situation as well. The expert for an input constellation is found by jumping from node to node where at each node only a few probabilities are to be calculated. As it is our final goal to built an on-line violin, this feature will be most important.

If one is looking for a philosophical justification of hierarchical cluster trees, he/she may consider nature as making many successive (binary) decisions. Just as the algorithm is finding its way through the tree, nature decides which path to take at a number of alternatives that is still understandable. Even though this view might not quite correspond to reality, it helps to understand complex decision systems.

4.5.3 On-line Learning

It has been pointed out that it is necessary to break the cluster/point interaction, such that our overall model becomes computable. We proposed a hierarchical structure, that reduced the computational proportionality roughly by $M/\log M$.

A further improvement could be achieved by incremental learning or on-line learning. The principle idea is expressed by the following question: Given an optimal model with respect to a data set I, how do parameters have to be updated when a new data set -data set II- is considered? In other words, which is the optimal model with respect to a first data set, that had been used for initializing the model and then was thrown away, and a second data set which is considered later.

The concept that jumps to mind is the Kalman Filter. A Kalman Filter updates model parameters with respect to the old model, to the new data and to a predicted value (prediction error). However, there are a couple of problems associated with the implementation of such a filter. It requires very precise ideas of how the model has to look like and which task each model parameter is supposed to fulfill. Its implementation for fully non-linear models with complex architecture and unsupervised parameter allocation is certainly tricky, if not impossible or unstable.

We therefore propose an update algorithm that is very much oriented at the EM framework and could in fact be named on-line-EM. It does not take into account the prediction error of the old model, but simply acts as if it had the old data and the new data available for the update.

Assume that a set of data g_1 has been exploited to train a cluster based probability estimate via the EM algorithm. Once more $\Psi_j^y = \mu_j^y$ for clarity. The model $\mathcal{M}_1$ has been iterated until convergence into a least square minimum and then data set g_1 was thrown away. We keep the number of clusters M constant through the model change. Model $\mathcal{M}_1$ is to be updated given a data set g_2, generated by the same system. Further notations are needed to describe the update:

i: Index associated with the new points.

j_1: Index associated with the old clusters.

j_2: Index associated with the new clusters.

Θ_1: Parameter set of model $\mathcal{M}_1$, initialized with data set g_1. Θ_1 includes M mean vectors $\vec{\mu}_{j_1}^z$, as many variances $\vec{\sigma}_{j_1}^2$ and M weights ω_{j_1}.

Θ_2: Parameter set of model $\mathcal{M}_2$, updated with data set g_2. Θ_2 includes M mean vectors $\vec{\mu}_{j_2}^z$, as many variances $\vec{\sigma}_{j_2}^2$ and M weights ω_{j_2}.

C_1: Cardinality of g_1 $(= N_1)$.

C_2: Cardinality of g_2 $(= N_2)$.

$\mathcal{M}_2$ is first initialized with $\mathcal{M}_1$[1]. Then the following update-rules for $\mathcal{M}_2$ are applied:

[1]Conceivably $\mathcal{M}_2$ could be initialized arbitrarily. However, the proposed solution seems to be more efficient, especially as we assume $C_2 < C_1$ and as we assume $\mathcal{M}_2$ not very different from $\mathcal{M}_1$.

E-step: We need to define the mutual probabilistic dependences between the new points and the new clusters, and between the old clusters and the new clusters.

$$\begin{aligned} p_{i,j_2} &= \omega_{j_2} \cdot \chi^z_{j_2}(\mathbf{z}_i) \qquad (4.7) \\ &= \omega_{j_2} \chi^y_{j_2}(y_i|\mathbf{x}_i) \cdot \chi^x_{j_2}(\mathbf{x}_i) \\ p_{j_1,j_2} &= \lambda \cdot \omega_{j_2} \cdot \frac{C_1}{C_2} \cdot M_1 \cdot \omega_{j_1} \cdot \chi^z_{j_2}(\vec{\mu}^z_{j_1}) \\ &= \lambda \cdot \omega_{j_2} \cdot \frac{C_1}{C_2} M_1 \cdot \omega_{j_1} \cdot \chi^y_{j_2}(\vec{\mu}^y_{j_1}|\vec{\mu}^z_{j_1}) \cdot \chi^x_{j_2}(\vec{\mu}^x_{j_1}) \\ &. \qquad (4.8) \end{aligned}$$

where the χ's have been defined in equation 3.17 and 3.20 and $\lambda \leq 1$. The corresponding αs follow directly:

$$\begin{aligned} \alpha_{i,j_2} &= \frac{p_{i,j_2}}{\sum_{j_2=1}^{M} p_{i,j_2} + p_{j_1,j_2}} \qquad (4.9) \\ \alpha_{j_1,j_2} &= \frac{p_{j_1,j_2}}{\sum_{j_2=1}^{M} p_{i,j_2} + p_{j_1,j_2}} \end{aligned}$$

M-step: The M-step is pretty straightforward as well:

$$\begin{aligned} \omega_{j_2} &= \frac{\sum_{i=1}^{N_2} \alpha_{i,j_2} + \sum_{j_1=1}^{M} \alpha_{j_1,j_2}}{\sum_{j_2=1}^{M}(\sum_{i=1}^{N_2} \alpha_{i,j_2} + \sum_{j_1=1}^{M} \alpha_{i,j_2})} \qquad (4.10) \\ \vec{\mu}^z_{j_2} &= \frac{\sum_{i=1}^{N_2} \mathbf{z}_i \alpha_{i,j_2} + \sum_{j_1=1}^{M} \vec{\mu}^z_{j_1} \alpha_{j_1,j_2}}{\sum_{i=1}^{N_2} \alpha_{i,j_2} + \sum_{j_1=1}^{M} \alpha_{j_1,j_2}} \\ \vec{\sigma}^z_{j_2} &= \frac{\sum_{i=1}^{N_2} (\vec{\mu}^z_i - \mathbf{z}_i)^2 \alpha_{i,j_2} + \sum_{j_1=1}^{M} \vec{\sigma}^z_{j_1} \alpha_{i,j_2}}{\sum_{i=1}^{N_2} \alpha_{i,j_2} + \sum_{j_1=1}^{M} \alpha_{j_1,j_2}} \end{aligned}$$

Note that there is a slight inconsistency concerning the update of $\vec{\sigma}^z_{j_2}$. In fact not all he information given by the data set is conserved in the clusters. Therefore problems with algorithm steps occur, which result in some compromises. The errors associated with these compromises are kept small under the condition $C_1 \gg C_2$. λ denotes a decay parameter that reduces the weight of the old data at each E-step. Therefore new data is always taken a little bit more serious than data that has been used some time ago. This also means that the model can change or adopt with time. The role of λ within the Bayesian framework has been pointed out above (chapter 3.1). E- and M-step are repeated until convergence. It might be possible to add new data sets, before a model has completely been converged. The update rules for hierarchical clusters would obviously be more complex, but there is an analytical solution as well. See [JJ93] for the general case.

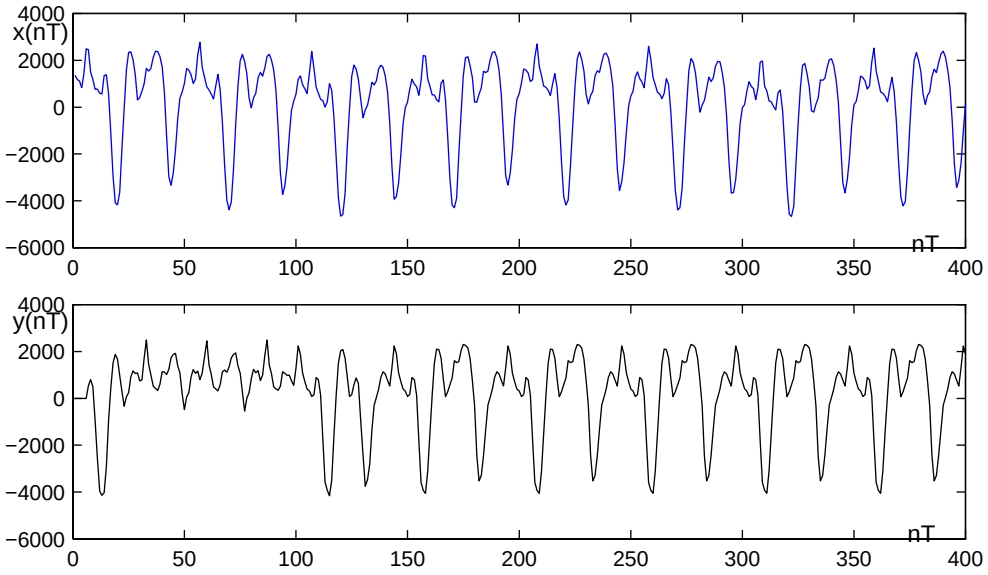

Figure 4-3: Steady state violin signal (400 samples). (a) original and (b) re-synthesized.

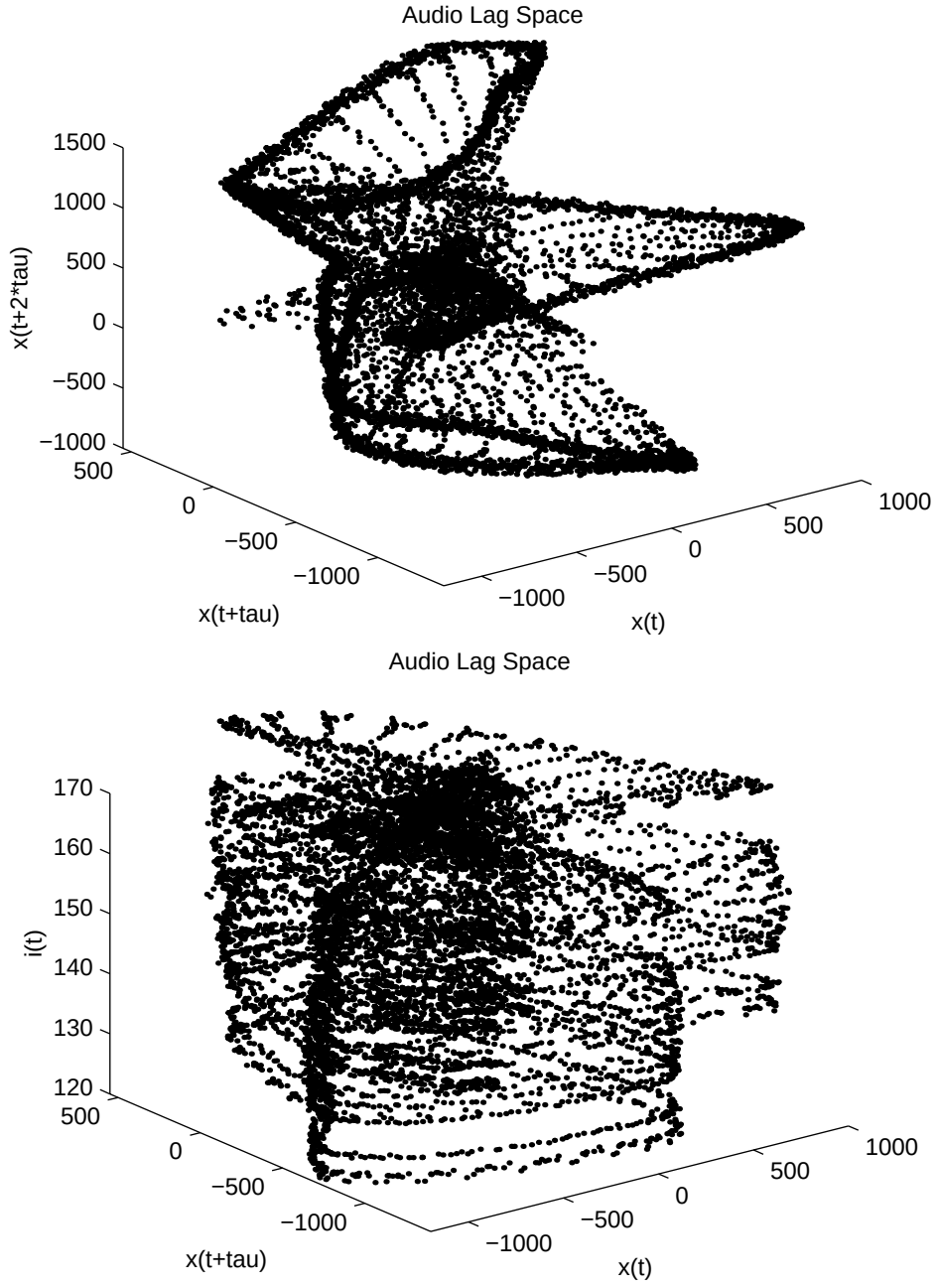

Figure 4-4: (a) Violin audio signal embedded in a three dimensional time-lag space,(b)violin audio signal (x and y axes) and x-position (z axis).

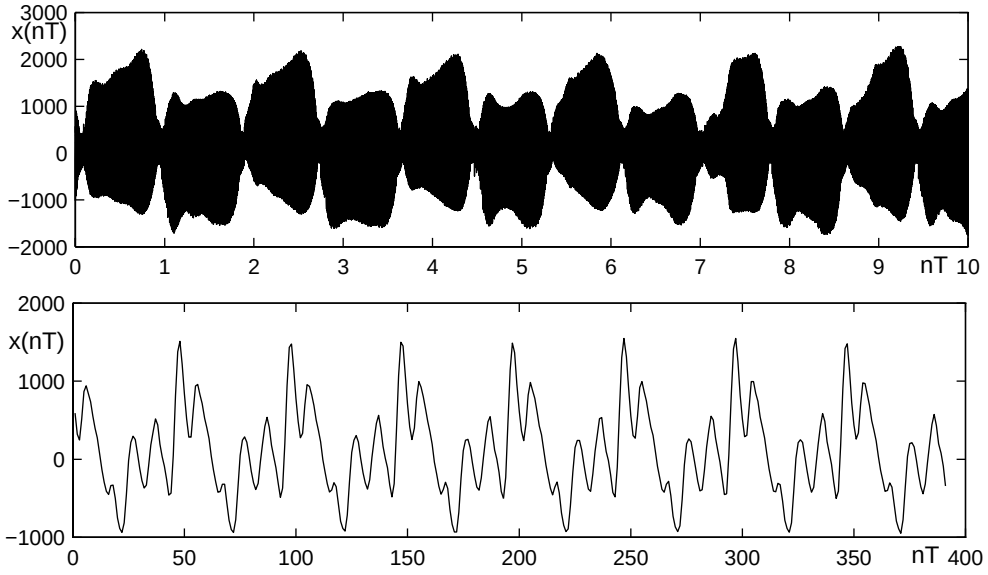

Figure 4-5: Training signal for the point prediction mode (10 seconds). The training was done on sample 2000-5000, which corresponds to the interval [0.2,1] of plot(a).

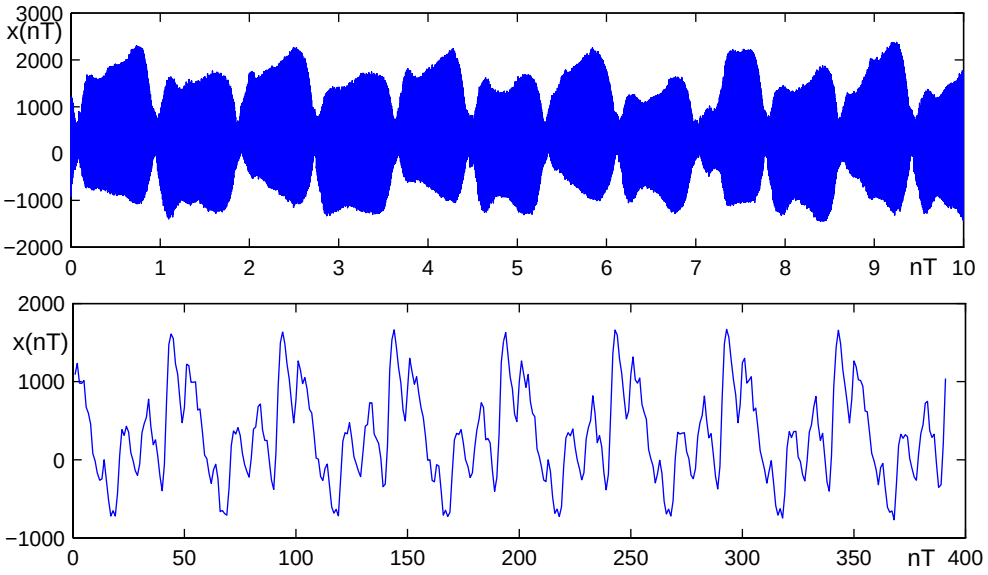

Figure 4-6: Prediction of the input/output signal in the point prediction mode.

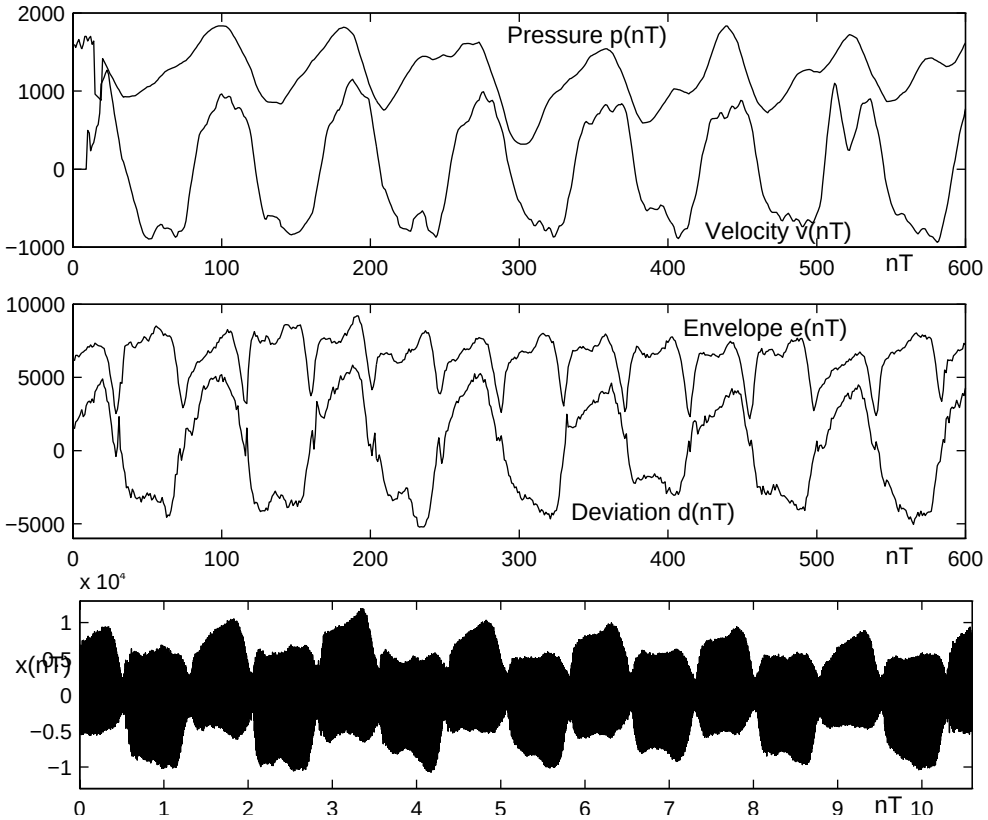

Figure 4-7: sample (6 seconds) of détaché bowing. Input series (a). Extracted envelope $e(t)$ and deviation $d(t)$ (b). Full signal (c).

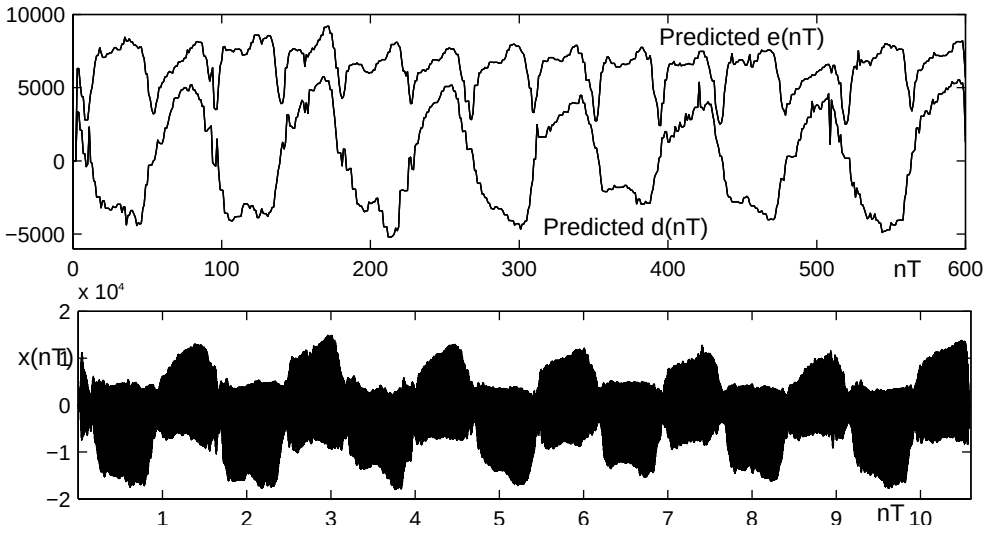

Figure 4-8: Re-synthesized envelope and deviation (a). Re-combination with an arbitrary steady state signal (b).

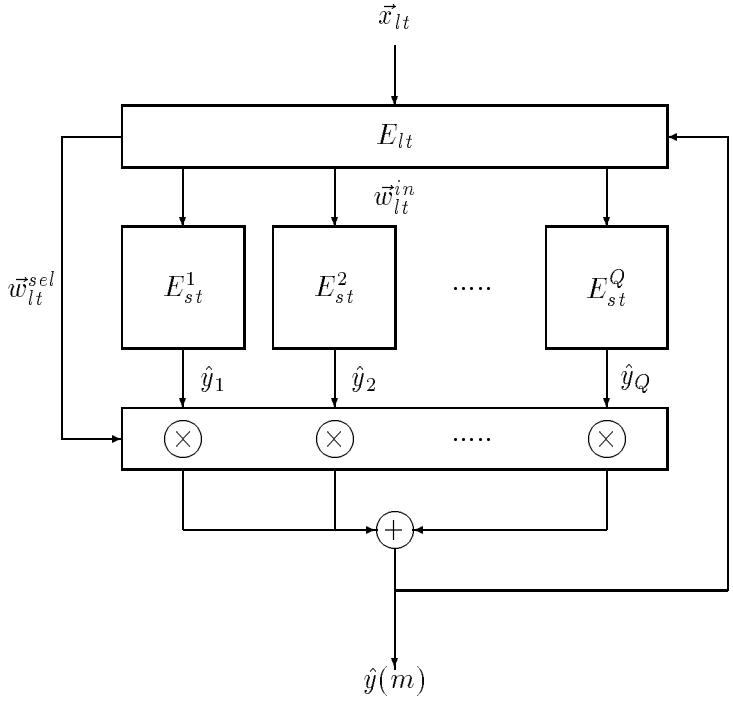

Figure 4-9: System of hierarchical experts for multi-time-scale prediction.

Chapter 5

Toward a Final Model

5.556 Eine Hierarchie der Formen der Elementarsätze kann es nicht geben. Nur was wir selbst konstruieren, können wir vorraussehen.

5.5561 Die empirische Realität ist begrenzt durch die Gesamtheit der Gegenstände. Die Grenze zeigt sich wieder in der Gesamtheit der Elementarsätze.
Die Hierarchien sind und müssen unabhängig von der Realiät sein.
L. Wittgenstein, TRACTATUS

5.1 The Architecture

Based on the experimental results and the potential architectural improvements discussed in the last chapter we propose a two-scale two-expert predictor model of the violin:

The long-term predictor (expert E_{lt}) reconstructs the amplitude envelope and the deviation from zero mean of the violin signal. The long term expert is also in charge of selecting the most appropriate short term predictor, given a particular state of the violin and a particular input. The short term predictor (expert E_{st}) synthesizes the most likely normalized wave form. The amplitude modulation of the output of the two experts results in the final audio signal.

Both experts are implemented as cluster weighted estimators. They are distinct, but do interact especially during the learning process. The predicted audio signal can be expressed as a product of conditional probabilities.

Figure 5.1 indicates the overall architecture. The model is running at two different sampling periods. The long-term sample period T_{lt} is determined by the sampling of the input $\vec{i}_l$. We define $\vec{i}_l = \vec{i}(lT_{lt})$, where $\vec{i}(lT_{lt})$ contains bowing data and information about pitch. Associated with T_{lt} is the long term behavior e_l, d_l and the long term expert E_{lt}. The short term sampling period T_{st} is equal to the audio sampling period. It is associated with the predicted wave form s_m and the audio signal y_m, defined as $s_m = s(mT_{st})$ and $y_m = y(mT_{st})$.

The Long Term Expert

Expert E_{lt} works at T_{lt}. It fulfills two basic tasks: It predicts the signal's long-term amplitude envelope e_l and its deviation from zero mean d_l[1]. The prediction concept has been proposed and discussed in section 4.4. The long term prediction space may or may not contain time lags of the output, depending on how stable the predictor behaves.

Secondly E_{lt} has a discrete classification task. It decides which short term expert E_{st} best fits the global state of the violin. Classification has not been formally introduced in this paper. So far we used cluster based models for the prediction of a continuous variable y. When y is chosen from a discrete and finite set of Q elements, the problem becomes a classification problem. The model is supposed to find the most likely element out of the Q possible elements for y. The original density space is replaced by Q discrete density spaces.

Let's denote $p(E_{st}^q|\mathbf{x}_l)$ the probability that short term expert E_{st}^q is responsible for the wave generation given a certain long term state $\mathbf{x}(l)$. This probability is proportional to the value of the PDF associated with expert E_{st}^q. Each of the associated PDF's is approximated by weighted superpositions of Gaussian basis functions. We define the expert PDF

$$p_q(\mathbf{x}) = \sum_{j=1}^{M_q} \omega_j^q \cdot \chi_j^{xq}(\mathbf{x}) \tag{5.1}$$

with

$$\chi_j^{xq}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^D |\mathbf{P}_j|}} \cdot e^{-\frac{1}{2}(\mathbf{x}-\vec{\mu}_j)^T P_j^{-1}(\mathbf{x}-\vec{\mu}_j)}.$$

We then obtain

$$p(E_{st}^q|\mathbf{x}_l) = \frac{p_q(\mathbf{x}_l)}{\sum_{q=1}^{Q} p_q(\mathbf{x}_l)} \tag{5.2}$$

as the overall probability of expert E_{st}^q being in charge of the wave prediction. E_{lt} has to maximize equation 5.2 with respect to q and selects the corresponding E_{st}^q. The two decision spaces handled by expert E_{lt} are either completely separate or they share the input vector $\vec{x}$.

The Short Term Expert

The selected short term expert E_{st}^q works as an autonomous predictor at the sampling period T_{st}. As has been shown in section 4.2, stable iterated prediction is very well possible with clear trajectories in space. We choose such a autonomous system to predict a normalized signal $s_m = s(mT_{st})$. The E_{st}^q are trained on normalized signal pieces, for constant amplitude assures that short term models can be summarized independently from their energy level. We hope to allocate resources more efficiently this way.

Out of E_{lt} comes the predicted amplitude e_l and the predicted deviation d_l. Both series are sampled at T_{lt}. In order to fit with the short term signal they have to be re-sampled at

[1] d_l might perceptionally not be important enough to spend resources on this detail. Further perceptional test will be made on that question.

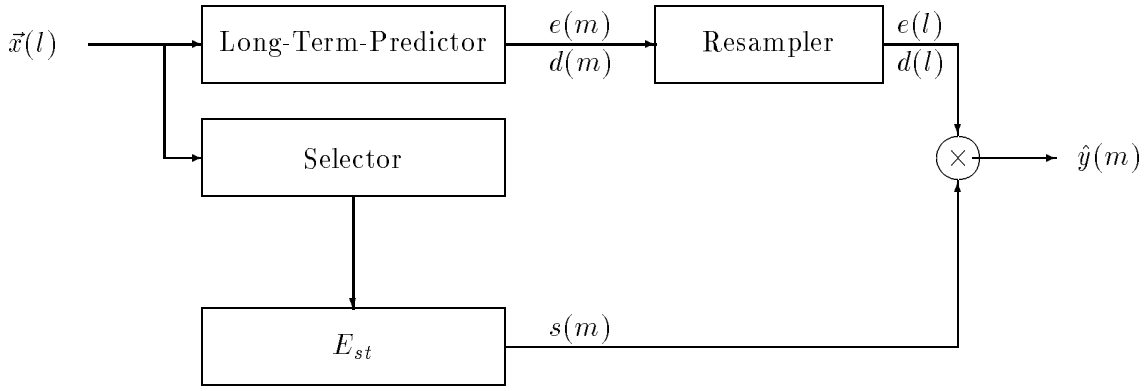

Figure 5-1: Model architecture for a multi-time-scale audio predictor

T_{st} and filtered. Eventually a slight time delay between input and output will be built into the system, which would allow a non causal filter. Finally the three time series s_m, d_m and e_m are recombined according to the simple rule:

$$\hat{y}_m = s_m \cdot e_m + d_m \tag{5.3}$$

$\hat{y}$ is now supposed to be equal to the original output.

Any one cluster model in this expert system is realized as a hierarchical cluster tree, as proposed in section 4.5.2. So learning and on-line synthesis become feasible.

5.2 Learning

We propose a learning algorithm close to the actual implementation. The learning is done incrementally (see section 4.5.3). We first initialize the model with a reasonable amount of data and then add successively new pieces of signal to the model. The learning algorithm for the envelope expert is independent from the classification and the short term expert. It is done as described in section 4.4. The classification expert and the short term predictors do interact during the learning process.

One more important definition is needed. In the last section the likelihood has been defined, that expert E_{st}^q is in charge of prediction given a certain long term state $\mathbf{x}_l$. For the training we need to define the conditional probability $p(E_{st}^q \mid g_{st})$ of a piece of training signal being generated by an expert E_{st}^q, where g_{st} denotes the set of signal points represented in lag space, of the piece of signal in issue. Fortunately we dispose already of a density approximation of the expert space q, and by now we are familiar with the two steps that lead to a conditional probability distribution. As we are dealing with a set of sample points

we need to define a product of probability distributions:

$$p(E_{st}^q \mid g_{st}) \propto \prod_{\vec{z} \in g_{st}} p_z(E_{st}^q | \mathbf{z}) = \prod_{\vec{z} \in g_{st}} p_z^q(\mathbf{z}) \tag{5.4}$$

where $p_z^q(\vec{z})$ denotes the short term PDF associated with short term expert E_{st}^q. For the final distribution we obtain

$$p(E_{st}^q \mid g_{st}) = \frac{\prod_{\mathbf{z} \in g_{st}} p_z^q(\mathbf{z})}{\sum_{q=1}^{Q} \prod_{\mathbf{z} \in g_{st}} p_z^q(\mathbf{z})} \tag{5.5}$$

We propose the algorithm, split into initialization and the on-line update rules. Note that the initialization step by itself is an iterated EM procedure.

Initialization:

1. Take a set of audio samples and the corresponding input samples. Extract the envelope information from the signal, normalize it and cut it into Q pieces.[2]

2. Initialize the envelope predictor. Build a lag space with the whole set of inputs and run a parameter search algorithm with a fixed number of basis functions.

3. Built Q autonomous short term predictors with the Q subsets of normalized audio points.

4. Initialize the classification expert by assigning a different classification PDF in $\mathbf{x}_{lt}$ to each of the experts E_{st}^q. Each point in the envelope predictor space is associated with the corresponding E_{st}^q. The ensemble of points $\mathbf{x}_{lt}$ that belongs to the same short term expert is estimated in a cluster based classification PDF $p_q(x_{lt})$.

On line update:

1. Take the next input sample k_{new} and the corresponding set of audio samples g_{new}. Extract the envelope information and normalize g_{new}.

2. Update the envelope expert E_{lt} with the new sample k according to the rules from section 4.5.3.

3. Find the $E_{st}^{q^*}$ that maximizes $p(E_{st}{}^q \mid g_{new})$. Ad g_{new} to the lag space that defines $E_{st}^{q^*}$ and update $E_{st}^{q^*}$ according to the update rules defined in section 4.5.3.

4. Update the classification PDF p_{q^*}associated with expert $E_{st}^{q^*}$ according to section 4.5.3.

[2]Clearly the set of data used for initialization has to be reasonable big. Concretely it should be at least bigger than $\frac{T_{lt} \cdot Q}{T_{st}}$.

5. Go to 1.

Most of the update rules have been explained in section 4.5.3. One detail is to be added: We cannot hope to do a very good job by initializing the different experts with random data. Therefore a decay parameter λ is introduced that lowers the relative weight of the old data at every update step. The mechanisms behind this concept has been explained in sections 3.1 and 4.5.3.

For the same reason the whole available data set may be passed through the algorithm several times. We cannot expect the algorithm to converge into the optimal solution at the first run, unless the data is itself repetitive. Any further run through should result in an improvement of the model in terms of maximizing the model likelihood, as well as in terms of perception.

We have not formally proved global convergence of the proposed architecture and the associated learning concept. However, convergence has been investigated in related contexts. In fact, our model could be interpreted as a special candidate of mixture experts, the convergence of which in fact has been proved [JJ93].

Chapter 6

Conclusion

6.5 Zu einer Antwort, die man nicht aussprechen kann, kann man auch die Frage nicht aussprechen.
Das Rätsel gibt es nicht.
Wenn sich eine Frage überhaupt stellen läßt, so *kann* sie auch beantwortet werden.
L. Wittgenstein, TRACTATUS[1]

It is the goal of this paper to give an overview over modeling techniques, model architectures and search algorithms to eventually build a complete synthesis system, which might deserve the title 'The Digital Stradivarius'. In fact the pieces described in this thesis need to be put together, in order to get a prototype for an input/output sound generator. A working synthesis system is now a matter of implementation and further tuning of model parameters.

In chapter 2 the theory of state-space reconstruction for unknown multidimensional input/output systems has been reviewed and related to the specific goal of musical synthesis. In the next chapter function approximation techniques and parameter search techniques were proposed. In particular cluster weighted local modeling was presented in a new formalism which includes the information about the global clusters and the local functions in

[1]

1.1 The world is the totality of facts, not of things.

1.12 For the totality of facts determines both what is the case, and also all that is not the case.

2.0123 If I know an object, then I also know all the possibilities of its occurrences in atomic facts. (Every such possibility must lie in the nature of the object.) A new possibility cannot subsequently be found.

2.01231 In order to know an object, I must know not its external but all its internal qualities.

3.032 To present in language anything which "contradicts logic" is as impossible as in geometry to present by its co-ordinates a figure which contradicts the laws of space; or to give the co-ordinates of a point which does not exist.

3.01231 We could present spatially an atomic fact which contradicted the laws of physics, but not one which contradicted the laws of geometry.

a single probability density estimate. The EM algorithm was then extended to a parameter search algorithm for local polynomials.

It has been shown in chapter 4 that the reconstruction of the violin's state space in time lag space indeed leads to a single valued function, as foreseen by the embedding theorem. In the stationary case we obtained clear orbits that were successfully used for stable cluster based prediction. In this mode the system gets confused only if damping effects in the training signal become to strong. At the time scale of the waveform envelope the correlation between the bowing data and the energy output became obvious. We used this feature for efficient prediction of the global signal behavior at a sampling rate in the millisecond range. The overall input/output state-space was reconstructed with polynomial basis functions. Although perfect extrapolation was achieved in the point-prediction mode, stable iterated prediction turned out to be impossible.

Finally hierarchical model structures have been proposed, that either split spaces into different input spaces or define prediction tasks at different time scales. In fact 'repartition of labor' in terms of predicting experts comes out of this paper as the main maxim for prediction. The principle of repartition applies at multiple levels. Efficient prediction needs division of global spaces into subspaces, hierarchical models within spaces and input domains, and multi-time-scale representations. The model proposed in chapter 5 represents a first attempt to integrate these features with respect to musical synthesis.

What remains to be done? Eventually, completely different function approximation techniques should be tested as well. We did not work with neural network architectures so far, for the transparency of cluster based models seemed to be too appealing as opposed to the hidden functionalism of neural nets. However, we might have to make use of the latter's compact description and prediction power, when a final on-line model is to be built. Also further research needs to be done on efficient parameter search algorithms. The stability problems that occured with the EM-algorithm could possibly be overcome by explicit search algorithms.

The goal of re-synthesis based on state-space reconstruction hangs in between physics and engineering. We are primarily interested in the phenomenological reconstruction of a physical system. We then use this information for the engineering like simulation and control of a system that is supposed to synthesize sound, indistinguishable from the original. Clearly

---

4.014 The gramophon record, the musical thought, the score, the waves of sound, all stand to one another in that pictorial internal relation, which holds between language and the world. To all of them the logical structure is common. (Like the two youths, their two horses and their lilies in the story. They are all in a certain sense one.)

5.5561 Empirical reality is limited by the totality of objects. The boundary appears again in the totality of elementary propositions. The hierarchies are and must be independent of reality.

6.5 For an answer which cannot be expressed the question too cannot be expressed. *The riddle* does not exist. If a question can be put at all, then it *can* also be answered.

Ludwig Wittgenstein. Tractatus Logico-Philosophicus

control theory from an engineering point of view, historically mostly linear control theory, and state-space-reconstruction based on time lags are very much related, as the feedback structure of states is at the heard of the two approaches. However, we might have to gain a clearer idea how results from the two domains can be combined for nonlinear control. Eventually intelligent combinations of linear and nonlinear techniques might be realized, such as a linear feedback together with a nonlinear prediction function.

Along with the physical and mathematical improvements more perceptional research should be done. Not all the signal features have the same importance for perceptionally good re-synthesis. The relevant features need to be identified in a more systematic way. This is where our approach to musical synthesis meets other research areas, e.g. audio compression.

Looking back to the very first sentence of this paper, we agree with the violin maker, that many questions arise when the violin becomes subject of discussion and research. However, we believe that some answers have been found working on this thesis and Wittgenstein (see quote) confirms our optimism that extraordinary synthesis will be possible soon.

This paper is to be finished with only a few remarks on the aesthetic implications of the 'Digital Stradivarius'. We will not replace the violin, but create for it a digital brother, which shares features, such as the basic control space and output behavior. Given the direct access to the input as well as to the output, the control space of the violin player can be enlarged and transformed in many directions. The violin may sound like a cello or it may trigger pre-sampled or synthesized music by decision rules based on the control data. Any such measure will enlarge the creation space accessible by composer and player. The digital violin is no longer image, but becomes an instrument on its own. It defines a new world of aesthetic experience both from a player's and an audience point of view.

Appendix A

Violin and Bow Hardware

The violin bow which we used for data sampling was developed by Joe Paradiso and Neil Gershenfeld [PG96]. Originally it served in a performance project with violinist Ani Kevaffian and the St. Paul Chamber Orchestra in 1993. The Paradiso bow behaves like an ordinary violin bow, apart from a slightly increased weight. It is loaded with sensors that sensor the bow position (the lateral position of the bow relative to the strings), the bow/bridge distance and the finger pressure. In the original music project this sensor data was used to translate the players action into computed sounds, either triggering presampled sound files or adjusting parameters of on-line synthesizers. The violin has been mutated into a hyperinstrument as its musical spectrum was enlarged compared to its traditional possibilities. Besides these artistic aspects the violin bow became very helpful for our scientific tasks .

We shortly describe the principle bow architecture (see figure A): A resistive strip is attached to the bow over the hole length of the pole. At its ends at frog and tip it is connected to battery powered low power oscillators, working at constant amplitudes and frequencies of 50 and 100 kHz. The resistive strip serves as antenna and as voltage divider. The two signals are received by an antenna mounted close to the violin bridge. The strength of signals is therefore roughly proportional to the distance bridge/frog respectively the distance bridge/tip.

Figure A shows the principle hardware components. The received signal first passes a FET-source-follower, placed on the violin. It is then transmitted through a shielded cable to the 'ground-station' containing the signal processing electronics, where it is filtered by second order band-passes of 50 and 100 kHz mid-frequencies. Finally envelope followers detect the amplitude of each of the two broadcasted frequencies V_L and V_R (see figure A).

It has been shown by Gershenfeld and Paradiso [PG96], that the x-position of the bow is approximately proportional to $(V_L - V_R)$. Moreover the distance bridge/bow goes linearly with the inverse sum of the signal $1/(V_L - V_R)$. Therefore we define the following variables

The violin bow is a sort of prototype for the upcoming project with Kronos-Quartet in Summer 1997, although more research is going to be done on the sensor technology. In fact we would like to replace the battery powered electric field oscillators by passive tags that work either as electric or as magnetic resonators.

as two of the potential input variables.

$$x(t) = \alpha_x \cdot \frac{(V_L - V_R)}{(V_L + V_R)} + \beta_x \quad \text{(A.1)}$$

$$y(t) = \alpha_y \cdot \frac{1}{(V_L + V_R)} + \beta_y \quad \text{(A.2)}$$

$$\text{(A.3)}$$

where α and β are linear normalization factors.

Furthermore a piezoresistive strip placed on the bow between index finger and pole changes its resistance with finger pressure and makes the corresponding low-impedance voltage cause a frequency change for a third oscillator. This additional oscillator was mounted on the bow, running at 25 kHz and transmitting through an antenna covering the full bow length. The 'ground-station' filters the signal, detects the frequency change in a phase-locked-loop and recovers the original pressure change.

The raw analog signals are converted into 8-bit samples by a HC11 A/D-converter, which was connected to the PC serial port. The sample rate for the input data was 125 Hz, which allows to cover almost all possible human motorics. Eventually the sampling rate will be reduced to 2 samples/ms. 1000 Hz seems to be the upper most resolution of human motion. Given that the control of a musical instrument requires as much sensibility as possible we would like to cover any detail of the 1kHz control motion.

Simultaneously with the bowing data we recorded the audio signal of the violin at either 22050 or 44100 Hz. The acoustic violin (German, $\sim$ 1900) is recorded by a microphone driven by a preamplifier which is connected to the PC sound card. The electrical violin we used was built by Richard Armin in Toronto. It uses piezoelectric polymer pickups to sense the vibration of the violin top plate. The picked up signal is directly connected to the PC sound device. In fact very little direct sound is radiated. The electrical violin can clearly be identified as an instrument of the violin family, although it does not provide the richness of sound and the control variety as the acoustical violin does. As there is no intermediate noise source between the vibrating violin corpus and the recording system, the final noise level of the electrical violin is much smaller than that of the acoustical recording. This feature was very useful for some of our experiments.

Bibliography

[ABST93] Henry D.I. Abarnel, Reggie Brown, John J. Sidorowich, and Lev Sh. Tsimring. The analysis of observed chaotic data in physical systems. *Reviews of Modern Physics*, 65(4):1331–1392, 1993.

[BPV84] Pierre Berge, Yves Pomeau, and Christian Vidal. *L'ordre dans le chaos. Vers une approche deterministe de la turbulence*. Hermann, Paris, 1984.

[Buc73] Alexander Buchner. *Geigenverbesserer*. Das Musikinstrument, Frankfurt/Main, 1973.

[Cas92] Martin Casdagli. A dynamical approach to modeling input-output systems. In *Nonlinear Modeling and Forecasting*. Addison-Wesley, 1992.

[CJE+] Martin Casdagli, Deirdre Jardins, Stephen Eubank, J.Doyne Farmer, John Gibson, Norman Hunter, and James Theiler. Nonlinear modeling of chaotic time series: Theroy and application.

[DES94] F.R. Drepper, R. Engbert, and N. Stollenwerk. Nonlinear time series analysis of empirical population dynamics. *Ecological Modelling*, 1994. to appear.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[Eck85] J.-P. Eckmann. Ergodic theory of chaos and strange attractors. *Review of Modern Physics*, 57(3 Part I):617–656, 1985.

[fW93] Yiu fai Wong. Clustering data by melting. *Neural Computation*, 5:89–104, 1993.

[fWP] Yiu fai Wong and Edward C. Posner. A new clustering algorithm applicable tp multispectral and polarimetric sar images. *IEEE Transactions on Geoscience and Remote Sensing*.

[Ger88] Neil Gershenfeld. An experimentalist's introduction to the observation of dynamical systems. In *Directions in Chaos*, volume II. 1988.

[Ger93] Neil Gershenfeld. Information in dynamics. In Doug Matzke, editor, *Proceedings of the Workshop on Physics of Computation*, Dallas, Texas, 1993. IEEE Press.

[Ger96] Neil A. Gershenfeld. Nonlinear inference and cluster-weighted modeling. In *Proceedings of the 1995 Florida Workshop on Nonlinear Astronomy*, 1996. to appear.

[Ger97a] Neil Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge Press, 1997. to appear.

[Ger97b] Neil Gershenfeld. *The Physics of Information Technology*. Cambridge Press, 1997. to appear.

[Gib88] V. Gibiat. Phase space representations of accoustical musical signals. *Journal of Sound and Vibration*, 123(3):529–536, 1988.

[GJP95] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization thory and neural networks architectures. *Neural Computation*, 7, 1995.

[GW93] Neil A. Gershenfeld and Andreas S. Weigend. The future of time series. In A.S. Wiegend and N.A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*. 1993.

[HBTS94] Hanspeter Herzel, David Bery, Ingo Titze, and Marwa Saleh. Analysis of vocal disorders with methods from nonlinear dynamics. *Journal of Speech and Hearing Research*, 37:1–37, 1994.

[Hun92] Norman F. Hunter. Application of nonlinear time-series models to driven systems. In M. Casdagli and S. Eubank, editors, *Nonlinear Modeling and Forecasting*, pages 467–491. Addison-Wesley, 1992.

[JJ93] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the —protectEM algorithm, 1993.

[LP88] W. Lauterborn and U. Parlitz. Methods of chaos physics and their application to acoustics. *Journal of the Accoustical Society of America*, 84(6):1975–1993, 1988.

[Lue75] H.D. Lueke. *Signalübertragung*. Springer, 1975.

[Met96] Eric Metois. *Music, Sound and Information*. PhD thesis, MIT Media Lab, 1996. to appear.

[Mey93] H. Meyr. *Regelungstechnik und Systemtheorie I/II*. RWTH Aachen, 1993.

[MF53] P.M. Morse and H. Feshbach. *Methods of Thoretical Physics*. McGraw-Hill Book Company, New York, 1953.

[MLS93] Fabrice Mortessagne, Olivier Legrand, and Didier Sornette. Transient chaos in room acoustics. *Chaos (American Institute of Physics)*, (3(4)):529–541, 1993.

[MS94] Jean Stephane Messonier and Bernd Schoner. Methode d'analyse et de characterisation des signaux de piano reel et de synthese, 1994. IRCAM Paris/ Ecole Centrale Paris.

[NH93] Radford M. Neal and Geoffrey E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. 1993.

[PCG91] J. Puaud, R. Causse, and V. Gibiat. Quasi-periodicitie et bifurcation dans la note de loupe. *J.Acoustique*, 4:253–259, 1991.

[PG] Joseph A. Paradiso and Neil Gershenfeld. Musical applications of electric field sensing. *Computer Music Journal*.

[PP93a] K. Popat and R.W. Picard. Novel cluster based probability model for texture synthesis, classification, and compression. *Proceedings of the SPIE. Visual Communications and Image Processing '93*, 2094, pt.2:756-68, 1993.

[PP93b] Kris Popat and Rosalind W. Picard. Cluster-based probability model and its application to image and texture processing. 1993.

[RGF90] Kenneth Rose, Eitan Gurewitz, and Geoffrey C. Fox. Statistical mecanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948, 1990.

[RJ86] L.R. Rubiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSp Magazine*, January 1986.

[Sac] Simone F. Sacconi. *Die 'Geheimnisse' Stradivaris (German translation)*. Das Musikinstrument, Frankfurt/Main.

[SYC] Tim Sauer, James Yorke, and Martin Casdagli. Embedology. *Journal of Statistical Physics*.

[Tak81] Floris Takens. Detecting strange attractors in turbulence. In *Lecture Notes in Mathematica*, pages 366–381. Springer, New York, 1981.

[WC90] Gabriel Weinreich and Rene Causse. Elementary stability considerations for bowed string motion. *Journal of the Acoustical Society of America*, (89(2)):887–895, 1990.

[WG93] Andreas Weigend and Neil Gershenfeld. Time series prediction. forecasting the future and understanding the past, 1993.

[Wit19] Ludwig Wittgenstein. *Tractatus logico-philosophicus*. 1919.

[WMS95] Andreas S. Weigend, Morgan Mangeas, and Ashok N. Srivastava. Nonlinear gated experts for time series: discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 1995.

[WZN95] Andreas Weigend, Hans Georg Zimmermann, and Ralph Neuneier. Clearning. 1995.

Nomenclature

$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{D}$ System matrices

a, b, c coefficients of linear ARMA models

B_i Mass function of data point $\mathbf{z}_i$

D space dimension (without the output dimension)

d index (indicating a space direction)

$d(t)$ signal deviation from zero mean

$d(n)$ discrete signal deviation from zero mean

C_j Cluster

$C()$ correlation dimension

$E_{i,j}$ cost function associated with data point $\mathbf{x}_i$ and cluster C_j

E_{st}^q short term expert

$e(t)$ energy envelope

$e(n)$ discrete energy envelope

e_k^d exponent associated with polynomial basis function k and space direction d

F effective cost function (free energy)

F_j effective cost function associated with cluster C_j

$f()$ prediction function

$\tilde{\mathbf{F}}(\mathbf{s})$ Fourier transform of $f(\mathbf{x})$

g set of data points

$H_d(\tau)$ Entropy of a d-dimensional lag-vector

$H(f)$ variation term

$h(\tau)$ source entropy

i index (data points)

$i(l)$ input time series

j index (basis functions)

K number of polynomial basis terms

K_l number of local basis terms

M number of basis terms (clusters)

m number of clusters per node (cluster-tree)

N number of points

n number of iterations

O order of polynomial approximation

P prediction function in the state space

$P(x)$ probability function

P_j covariance matrix associated with cluster Cj

$p(x)$ probability density function

$p(nT)$ discrete bow pressure

$p_{y|x}$ conditional probability density function

$p_{i,j}$ relative probability of data point $\mathbf{z}$ having been generated by cluster C_j

$<p(\mathbf{x})>$ expectation of $p(\mathbf{x})$

$<p(y\,|\,\mathbf{x})>$ conditional expectance of y

q index (experts)

Q number of experts

$R_d(\tau)$ Redundancy of a d-dimensional lag-vector

$s(t)$ signal function

$s(nT)$ *or* $s(n)$ time-discrete audio signal

$s(t-\tau)$ time lagged signal

$s((n-k)T)$ time-discrete lagged signal

T sampling period

T_{st} short term sampling period

T_{lt} long term sampling period

$v(nT)$,$v(n)$ discrete bow velocity

$\vec{w}_{lt}(l)$ internal long term state vector

$\vec{w}_{st}(m)$ internal short term state vector

$\vec{u}(t)$,$\mathbf{u}(t)$ time-continuous input vector

$\vec{u}(n)$,$\mathbf{u}(n)$ time-discrete input vector

$\vec{x}(t)$,$\mathbf{x}(t)$ reconstructed time-continuous state vector

$\vec{x}(t)$,$\mathbf{x}(t)$ reconstructed time-discrete state vector]

$\vec{x}_{lt}(l)$ long term input vector

$\mathcal{V}$ complete data(EM)

$\mathcal{W}$ unobserved data (EM)

$\mathbf{X}$ particular realization of $\mathbf{x}$

$y(t)$,$\hat{y}(t)$ scalar system output

$y(n)$,$\hat{y}(n)$ time discrete system output

$\vec{z}$,$\mathbf{z}$ state vector combining input and output dimensions

$\mathbf{Z}$ particular realization of $\mathbf{z}$

Z total partition function

Z_j partition function associated with cluster C_j

$\mathcal{Z}$ subspace of possible state vectors / observed data (EM)

$\alpha_{i,j}$ probability of data point $\mathbf{z}_i$ having been generated by cluster C_j

β Lagrange multiplier (1/T)

η_j weighted covariance matrix with respect to cluster C_j

Θ set of parameters

Θ^z set of parameters relating $\mathcal{Z}$ and $\mathcal{W}$

Θ_j^y set of parameters determining Ψ_j^y

κ tuning parameter

λ Lagrange multiplier or decay parameter (on-line-learning)

λ_i Lyapunov coefficient

$\vec{\mu}_j^x$ vector of cluster centers for cluster C_j

$\chi_j^x(\mathbf{x})$ local density estimate of $\mathbf{x}$

$\chi_j^y(y\,|\,\mathbf{x})$ conditional local density estimate of $\mathbf{y}$

χ_j^z local density estimate of $\mathbf{z}$

Φ diffeomorphic mapping between mechanical and the reconstructed state space

$\Phi(f)$ approximation prior

$\Psi_k(\mathbf{x})$ polynomial basis term

$\Psi_j^y(\mathbf{x})$ local function to approximate the output dimension y

ω_j weight of cluster C_j

$\vec{\sigma}_j^2$ vector of variances associated with cluster C_j

τ time lag