# Chapter 15

# Cluster-Weighted Modeling: Probabilistic Time Series Prediction, Characterization and Synthesis

**Bernd Schoner**[1]
**Neil Gershenfeld**

*ABSTRACT Cluster-Weighted Modeling, a mixture density estimator around local models, is presented as a framework for the analysis, prediction and characterization of non-linear time series. First architecture, model estimation and characterization formalisms are introduced. The characterization tools include estimator uncertainty, predictor uncertainty and the correlation dimension of the data set. In the second part of this chapter the framework is extended to synthesize audio signals and is applied to model a violin in a data-driven input-output approach.*

## 15.1 Introduction

The list of time series worthwhile to be forecast is about as long as the first unsuccessful attempts to do so. It would be most helpful to know beforehand when a heart is about to stop beating, what the weather will be like tomorrow and when the stock market is going to crash. Unfortunately, these examples share the one property that they have nothing in common and that they don't fit into any familiar categories of system dynamics theory. Not only are they non-linear, non-Gaussian and non-stationary, they are essentially non-everything.

Linear systems theory has yielded a multitude of results that are widely applied in practically all engineering and scientific disciplines. The majority of signal processing, system engineering, control and characterization techniques rely on linear assumptions and use a theory that has matured in decades of research and implementations. However, the limitations of linear theory are clear: non-linear behavior of any kind can not be handled.

The reconstruction (embedding) theorem on the other hand provides

[1]Author for correspondence.

the theoretical means to handle highly non-linear behavior of arbitrary physical systems with hidden dynamics [22]. It shows that the system's state space can be mapped into a diffeomorphic space, constructed from any observable of the system, and that we can characterize the data with respect to dimensionality and dynamic behavior in the reconstructed space. The reconstruction theorem also detects low dimensional structure in a high dimensional data space, which lets us work in the space described by the effective degrees of freedom of a system, for example a violin, rather than its countless mechanical degrees of freedom.

Unfortunately it turns out to be rather difficult to use a reconstructed state space to predict the output of a complex system. While low dimensional systems are tractable (Fig.15.1), models become easily unstable given a complicated state space or an arbitrary prediction horizon. Driven systems should be easier to handle than autonomous systems. However, the model dimensionality of a driven system is significantly bigger, since input and output observables need to be considered at the same time[4]. The presence of noise in practically any real world system further complicates the embedding task. Due to these problems we end up with a fairly small number of examples where embedding, despite its theoretical promise, has been applied successfully to predict a signal.

In between linear and highly non-linear systems there is a large class of systems that are not easily classified as one or the other but combine characteristics from both worlds. The bow string interaction of a violin, for example, is strongly non-linear, since it transforms the slow actions of the player into a fast audio signal. At the same time the effect of the violin body is most efficiently described by a linear filter since there is only little non-linear effects in the bridge and body dynamics[2] [14]. Hence a violin combines linear and non-linear processing.

This chapter introduces Cluster-Weighted Modeling (CWM) as a modeling tool that allows one to characterize and predict systems of arbitrary dynamic character. The framework is based on density estimation around Gaussian kernels which contain simple local models describing the system dynamics of a data subspace. In the extreme case where only one kernel is used the framework collapses to a simple model that is linear in the coefficients. In the opposite extreme it allows one to embed and forecast data that may be non-Gaussian, discontinuous, high-dimensional and chaotic. In between CWM covers a multitude of models, each of which is characterized by a different local model and state representation. We create globally non-linear models with transparent local structures through the embedding of past practice and mature techniques in the general non-linear framework.

[2]The exception from this is the famous Wolf tone, a tone that periodically collapses despite constant bowing. The phenomenon is particularly strong on the cello and is caused by a non-linear coupling between of a string and body mode [6].

The limitations of Artificial Neural Networks (ANNs) have become apparent almost as quickly as their modeling power: networks take long to converge, coefficients are only meaningful in the context of the entire model and failure and success of an architecture are unpredictable beforehand. More recently a new family of networks has been developed, which interpret data probabilistically and are often represented in graphical networks[3, 9, 11]. As a meta-class of models, graphical models are conceptually unbounded. They unify existing network architectures, for example classical ANNs in a single theory [15], provide new insights and extensions to conventional networks and open up new application domains. Graphical models are also referred to as independence networks, since the graphical representation really describes dependence and independence among random variables. They are called Bayesian belief networks since dependencies between variables are expressed in terms of conditional probability functions that have implicit or explicit prior beliefs built into them. They are furthermore named influence diagrams since causal dependences between variables are clearly illustrated. "Influence" is meant probabilistically, which contains deterministic causality as a special case. Unfortunately graphical models lack a systematic search algorithm that maps a given problem into a network architecture. Instead, before the network parameters can be trained on new data, the architecture needs to be redesigned node by node from scratch.

Cluster-Weighted Modeling is a special case of a probabilistic model that gives up some of the generality of graphical models in favor of ease of use, a minimal number of hyper-parameters and a fast parameter search. It has been designed as an architecture that is as general as reasonably possible, but as specific to a particular application as necessary. We present a tool that allows us to do statistical time series analysis from a physicist's perspective and at the same time allows us to solve complicated engineering problems, for example the design of a digital musical instruments. As opposed to ANNs it provides transparent local structures and meaningful parameters, it allows one to identify and analyze data subspaces and converges quickly.

The first part of this chapter provides the basic architecture, estimation and characterization tools of CWM. The second part is concerned with the problem of building a data-driven input-output model of a violin. The violin is a complex driven device that in its socio-cultural, artistic and physical subtlety is hardly matched by any other human artifact. At the same time the violin provides a very clear error metric in that the model is just as good as it sounds. From a non-linear dynamics and statistics viewpoint the violin is a paradigmatic object, since it shows non-linear and linear, stochastic and deterministic behavior at the same time.

## 15.2 Cluster-Weighted Modeling

### *15.2.1 Architecture*

Cluster-Weighted Modeling (CWM) is an input-output inference framework based on probability density estimation of a joint set of input feature and output target data. It is similar to mixture-of-experts type architectures [10] and can be interpreted as a flexible and transparent technique to approximate an arbitrary function. Unlike conventional Kernel based techniques, CWM requires only one hyper-parameter to be fixed beforehand, and provides data parameters such as the length scale (bandwidth) of the local approximation as an output rather than an input of the algorithm [5].

We start with a set of discrete or real valued input features $\mathbf{x}$ which may be measured features or components in a time lagged embedding space, and an discrete or real valued output target vector $\mathbf{y}$. Given the joint input-output set $\{\mathbf{y}_n, \mathbf{x}_n\}_{n=1}^N$, the most general model infers the joint density $p(\mathbf{y}, \mathbf{x})$ of the data set, from which conditional quantities such as the expected $\mathbf{y}$ given $\mathbf{x}$, $\langle \mathbf{y}|\mathbf{x}\rangle$, and the expected covariance of $\mathbf{y}$ given $\mathbf{x}$, $\langle \mathbf{P}_y|\mathbf{x}\rangle$ can be derived.

We expand this joint density in clusters labeled $c_m$, each of which contains an input domain of influence, a local model, and an output distribution. In a first step the joint density is separated into an unconditioned cluster probability and a conditional probability of a data given a cluster, which is then further expanded into an input domain of influence and an output distribution,

$$\begin{aligned} p(\mathbf{y}, \mathbf{x}) &= \sum_{m=1}^{M} p(\mathbf{y}, \mathbf{x}, c_m) \\ &= \sum_{m=1}^{M} p(\mathbf{y}, \mathbf{x}|c_m)\ p(c_m) \\ &= \sum_{m=1}^{M} p(\mathbf{y}|\mathbf{x}, c_m)\ p(\mathbf{x}|c_m)\ p(c_m)\ . \end{aligned} \tag{15.1}$$

Many problems require a distinction between slowly varying state variables describing the global boundary conditions and state of the system and fast varying variables describing the fast dynamics of the system. If this is the case we decompose $\mathbf{x}$ into $\mathbf{x}_s$ and $\mathbf{x}_f$ and obtain for the density

$$p(\mathbf{y}, \mathbf{x}) = \sum_{m=1}^{M} p(\mathbf{y}|\mathbf{x}_f, c_m)\ p(\mathbf{x}_s|c_m)\ p(c_m)\ , \tag{15.2}$$

where $\mathbf{x}_s$ and $\mathbf{x}_f$ may be identical, overlapping in some dimensions or completely distinct.

The input distribution is taken to be a Gaussian distribution,

$$p(\mathbf{x}|c_m) = \frac{|\mathbf{P}_m^{-1}|^{1/2}}{(2\pi)^{D/2}} e^{-(\mathbf{x}-\mu_m)^T \cdot \mathbf{P}_m^{-1} \cdot (\mathbf{x}-\mu_m)/2} \ , \tag{15.3}$$

where $\mathbf{P}_m$ is the cluster-weighted covariance matrix in the feature space. It can be reduced to variances in each dimension, when computational complexity is an issue.

The output distribution is taken to be

$$p(\mathbf{y}|\mathbf{x}, c_m) = \frac{|\mathbf{P}_{m,y}^{-1}|^{1/2}}{(2\pi)^{D_y/2}} e^{-(\mathbf{y}-\mathbf{f}(\mathbf{x},\beta_m))^T \cdot \mathbf{P}_{m,y}^{-1} \cdot (\mathbf{y}-\mathbf{f}(\mathbf{x},\beta_m))/2} \ , \tag{15.4}$$

where the mean value of the output Gaussian is replaced by the function $\mathbf{f}(\mathbf{x}, \beta_m)$ with unknown parameters $\beta_m$. Again the off diagonal terms in the output covariance matrices $\mathbf{P}_{m,y}$ can be neglected if needed.

To understand this form, consider the conditional forecast of the expected $\mathbf{y}$ given $\mathbf{x}$,

$$\begin{aligned} \langle \mathbf{y}|\mathbf{x}\rangle &= \int \mathbf{y}\ p(\mathbf{y}|\mathbf{x})\ d\mathbf{y} \qquad (15.5) \\ &= \int \mathbf{y}\ \frac{p(\mathbf{y},\mathbf{x})}{p(\mathbf{x})}\ d\mathbf{y} \\ &= \frac{\sum_{m=1}^{M} \int \mathbf{y}\ p(\mathbf{y}|\mathbf{x}, c_m)\ d\mathbf{y}\ p(\mathbf{x}|c_m)\ p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m)\ p(c_m)} \\ &= \frac{\sum_{m=1}^{M} \mathbf{f}(\mathbf{x},\beta_m)\ p(\mathbf{x}|c_m)\ p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m)\ p(c_m)} \ . \end{aligned}$$

We observe that the predicted $\mathbf{y}$ is a superposition of all the local functionals, where the weight of each contribution depends on the posterior probability that an input point was generated by a particular cluster. The denominator assures that the sum of the weights of all contributions equals unity.

Likewise we compute the conditional error in terms of the expected covariance of $\mathbf{y}$ given $\mathbf{x}$

$$\begin{aligned} \langle \mathbf{P}_y|\mathbf{x}\rangle &= \int (\mathbf{y} - \langle \mathbf{y}|\mathbf{x}\rangle)(\mathbf{y} - \langle \mathbf{y}|\mathbf{x}\rangle)^T\ p(\mathbf{y}|\mathbf{x})\ d\mathbf{y} \qquad (15.6) \\ &= \int (\mathbf{y}\mathbf{y}^T - \langle \mathbf{y}|\mathbf{x}\rangle\langle \mathbf{y}|\mathbf{x}\rangle^T)\ p(\mathbf{y}|\mathbf{x})\ d\mathbf{y} \\ &= \frac{\sum_{m=1}^{M} \int \mathbf{y}\mathbf{y}^T p(\mathbf{y}|\mathbf{x}, c_m) d\mathbf{y}\ p(\mathbf{x}|c_m)\ p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m)\ p(c_m)} - \langle \mathbf{y}|\mathbf{x}\rangle\langle \mathbf{y}|\mathbf{x}\rangle^T \\ &= \frac{\sum_{m=1}^{M} [\mathbf{P}_{m,y} + \mathbf{f}(\mathbf{x},\beta_m)\mathbf{f}(\mathbf{x},\beta_m)^T]\ p(\mathbf{x}|c_m)\ p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m)\ p(c_m)} - \langle \mathbf{y}|\mathbf{x}\rangle\langle \mathbf{y}|\mathbf{x}\rangle^T \end{aligned}$$

which equals the expected variance if only a single output dimension is considered,

$$\langle \sigma_y^2 | \mathbf{x} \rangle = \frac{\sum_{m=1}^{M} [\sigma_{m,y}^2 + f(\mathbf{x}, \beta_m)^2] \; p(\mathbf{x}|c_m) \; p(c_m)}{\sum_{m=1}^{M} p(\mathbf{x}|c_m) \; p(c_m)} - \langle y | \mathbf{x} \rangle^2 \; .$$

There are two parameters to be determined beforehand: the number of clusters $M$ and the form of the local models $f$ which together control the model resources and hence under versus over-fitting. We trade off the complexity of the local models against the complexity of the global architecture, which is nicely illustrated in the case of a local polynomial expansion(Equ.15.7): If we use locally constant models together with a large number of clusters, the predictive power is determined by the number of Gaussian kernels. If, alternatively, we use a high-order polynomial model and a single kernel, the model reduces to a global polynomial model.

The choice of local models depends on the application. In general $f$ expresses prior beliefs about the nature of the data or insights in the mechanics of a system and thus functions as a regularizer of the model. Machine learning architectures and estimation algorithms typically depend on global regularizers that handle prior beliefs about what is a good model. This is problematic since global statements may not apply locally. For example, the maximum entropy principle is good at handling discontinuities, but has no notion of local smoothness, whereas integrated curvature is good in enforcing local smoothness but rounds out discontinuities. In our approach the model is constrained only by the local architecture which may enforce local smoothness but at the same time allows for discontinuities where needed.

### *15.2.2 Model estimation*

Non-linear function fitting uses models with linear coefficients $\beta_m$ and non-linear basis functions $f(\mathbf{x})$,

$$\mathbf{y}(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \mathbf{f}_m(\mathbf{x}) \; , \tag{15.7}$$

for example a polynomial expansion, or models that have the coefficients inside the nonlinearities,

$$\mathbf{y}(\mathbf{x}) = \sum_{m=1}^{M} \mathbf{f}_m(\mathbf{x}, \beta_m) \; , \tag{15.8}$$

for example a neural network. In the case of a generalized linear model (Equ.15.7) only a single matrix pseudo-inverse is needed to find the set of coefficients yielding the minimum mean-square error. However, the number of coefficients in Equ.15.7 is exponential in the dimension of $\mathbf{x}$. A model with non-linear coefficients (Equ.15.8) has more expressive power, which can reduce the number of coefficients needed for a given approximation error to linear in the dimension of $\mathbf{x}$ [2]. Yet, the non-linear parameters of Equ.15.8 require an iterative search [8].

CWM uses simple local models, which satisfy (15.7), to create globally powerful models as described by (15.8) and hence combines the efficient estimation of the former with the benefits of the latter models. We fit the local model parameters by a matrix inversion of the local covariance matrix and find the remaining cluster parameters in charge of the global weighting, using a variant of the Expectation-Maximization (EM) algorithm [7]. EM is an iterative search that maximizes the model likelihood given a data set and initial conditions [16, 1]. We pick a set of starting values for the cluster parameters and then enter the iterations with the Expectation step.

In the **E-step** we assume the current cluster parameters correct and evaluate the posterior probabilities that relate each cluster to each data point. These posteriors can be interpreted as the probability that a particular data was generated by a particular cluster or as the normalized responsibility of a cluster for a point:

$$\begin{aligned} p(c_m|\mathbf{y},\mathbf{x}) &= \frac{p(\mathbf{y},\mathbf{x}|c_m)\ p(c_m)}{p(\mathbf{y},\mathbf{x})} \\ &= \frac{p(\mathbf{y},\mathbf{x}|c_m)\ p(c_m)}{\sum_{l=1}^{M} p(\mathbf{y},\mathbf{x}|c_l)\ p(c_l)} \ , \end{aligned} \tag{15.9}$$

where the sum over clusters in the denominator causes clusters to interact, fight over points and specialize in data they best explain.

In the **M-step** we assume the current data distribution correct and find the cluster parameters that maximize the likelihood of the data. The new estimate for the unconditioned cluster probabilities is

$$\begin{aligned} p(c_m) &= \int p(c_m|\mathbf{y},\mathbf{x})\ p(\mathbf{y},\mathbf{x})\ d\mathbf{y}\ d\mathbf{x} \\ &\approx \frac{1}{N}\sum_{n=1}^{N} p(c_m|\mathbf{y}_n,\mathbf{x}_n) \ , \end{aligned} \tag{15.10}$$

Here the idea is that an integral over a density can be approximated by an average over variables drawn from the density.

Next we compute the expected input mean of each cluster which is the estimate of the new cluster means:

$$\begin{aligned}
\mu_m &= \int \mathbf{x}\ p(\mathbf{x}|c_m)\ d\mathbf{x} \qquad (15.11)\\
&= \int \mathbf{x}\ p(\mathbf{y},\mathbf{x}|c_m)\ d\mathbf{y}\ d\mathbf{x}\\
&= \int \mathbf{x}\ \frac{p(c_m|\mathbf{y},\mathbf{x})}{p(c_m)}\ p(\mathbf{y},\mathbf{x})\ d\mathbf{y}\ d\mathbf{x}\\
&\approx \frac{1}{N\ p(c_m)} \sum_{n=1}^{N} \mathbf{x}_n\ p(c_m|\mathbf{y}_n,\mathbf{x}_n)\\
&= \frac{\sum_{n=1}^{N} \mathbf{x}_n\ p(c_m|\mathbf{y}_n,\mathbf{x}_n)}{\sum_{n=1}^{N} p(c_m|\mathbf{y}_n,\mathbf{x}_n)}
\end{aligned} \qquad (15.12)$$

The apparently formal introduction of $\mathbf{y}$ into the density as a variable to be integrated over has the important result that cluster parameters are found with respect to the joint input-output space. Clusters get pulled based on both where there is data to be explained and how well their model explains the data. In a similar way we can define a cluster-weighted expectation of any function $\Theta(\mathbf{x})$,

$$\begin{aligned}
\langle\theta(\mathbf{x})\rangle_m &\equiv \int \theta(\mathbf{x})\ p(\mathbf{x}|c_m)\ d\mathbf{x} \qquad (15.13)\\
&\approx \frac{1}{N}\sum_{n=1}^{N} \theta(\mathbf{x}_n)\ \frac{p(c_m|\mathbf{y}_n,\mathbf{x}_n)}{p(c_m)}\\
&= \frac{\sum_{n=1}^{N} \theta(\mathbf{x}_n)\ p(c_m|\mathbf{y}_n,\mathbf{x}_n)}{\sum_{n=1}^{N} p(c_m|\mathbf{y}_n,\mathbf{x}_n)}\ ,
\end{aligned}$$

which lets us update the cluster weighted covariance matrices,

$$[\mathbf{P}_m]_{ij} = \langle(x_i-\mu_i)(x_j-\mu_j)\rangle_m \qquad (15.14)$$

It also lets us compute the matrices needed for the update of the local models. The model parameters are found by taking the derivative of the *log* of the total likelihood function with respect to the parameters,

$$0 = \frac{\partial}{\partial\beta}\ \log \prod_{n=1}^{N}\ p(\mathbf{y}_n,\mathbf{x}_n)\ . \qquad (15.15)$$

Considering a single output dimension $y$ and a single coefficient $\beta_m$, we get:

$$\begin{aligned} 0 &= \sum_{n=1}^{N} \frac{\partial}{\partial \beta_m} \log\ p(\mathbf{y}_n, \mathbf{x}_n) \qquad (15.16) \\ &= \sum_{n=1}^{N} \frac{1}{p(y_n, \mathbf{x}_n)} p(\mathbf{y}_n, \mathbf{x}_n, c_m) \frac{y_n - f(\mathbf{x}_n, \beta_m)}{\sigma_{m,y}^2} \frac{\partial f(\mathbf{x}_n, \beta_m)}{\partial \beta_m} \\ &= \frac{1}{N p(c_m)} \sum_{n=1}^{N} p(c_m | \mathbf{y}_n, \mathbf{x}_n) [y_n - f(\mathbf{x}_n, \beta_m)] \frac{\partial f(\mathbf{x}_n, \beta_m)}{\partial \beta_m} \\ &= \left\langle [y - f(\mathbf{x}, \beta_m)]\ \frac{\partial f(\mathbf{x}, \beta_m)}{\partial \beta_m} \right\rangle_m \end{aligned}$$

Plugging (15.7) into (15.16) we obtain an expression to update $\beta_m$,

$$\begin{aligned} 0 &= \langle [y - f(\mathbf{x}, \beta_m)] f_j(\mathbf{x}) \rangle_m \qquad (15.17) \\ &= \underbrace{\langle y f_j(\mathbf{x}) \rangle_m}_{a_{j,m}} - \sum_{i=1}^{I} \beta_{m,i}\ \underbrace{\langle f_j(\mathbf{x}) f_i(\mathbf{x}) \rangle_m}_{\mathbf{B}_{ji,m}}\ , \\ \Rightarrow \beta_m &= \mathbf{B}_m^{-1} \cdot \mathbf{a}_m\ , \end{aligned}$$

where the matrix inverse should be done by a Singular Value Decomposition to avoid numerical problems with singular covariance matrices.

Considering the full set of model parameters we get

$$\beta_m = \mathbf{B}_m^{-1} \cdot\ \mathbf{A}_m\ , \qquad (15.18)$$

with

$$\begin{aligned} [\mathbf{B}_m]_{ij} &= \langle f_i(\mathbf{x}, \beta_m) \cdot f_j(\mathbf{x}, \beta_m) \rangle_m \\ [\mathbf{A}_m]_{ij} &= \langle y_i \cdot f_j(\mathbf{x}, \beta_m) \rangle_m\ . \end{aligned} \qquad (15.19)$$

Finally the output covariance matrices associated with each model are estimated,

$$\begin{aligned} \mathbf{P}_{y,m} &= \langle [\mathbf{y} - \langle \mathbf{y} | \mathbf{x} \rangle]^2 \rangle_m \qquad (15.20) \\ &= \langle [\mathbf{y} - \mathbf{f}(\mathbf{x}, \beta_m)] \cdot [\mathbf{y} - \mathbf{f}(\mathbf{x}, \beta_m)]^T \rangle_m\ . \end{aligned}$$

Clusters should not be initialized arbitrarily because the algorithm is only guaranteed to terminate in a local likelihood maximum. Also, initializing clusters in places that are close to their final position saves time, since they don't have to walk their way through the data set. We use a method that performs well empirically: Choose $1/N$ as the initial cluster probabilities.

Pick randomly as many points from the training set as there are clusters and initialize the cluster input means, as well as the cluster output mean with these points. Set the remaining output coefficients to zero. Use the size of the data set in each space dimension as the initial cluster variances. It is also a good idea to normalize the training data to zero mean and unit variance since arbitrary data values may cause probabilities to become too small.

To summarize the model estimation process: (1) pick some initial conditions; (2) then evaluate the probability of the data $p(\mathbf{y}, \mathbf{x}|c_m)$; (3) from those find the posterior probability of the clusters $p(c_m|\mathbf{y}, \mathbf{x})$; (4) then update the cluster weights $p(c_m)$, the cluster-weighted expectations for the input means $\mu_m^{new}$ and variances $\sigma_{m,d}^{2,new}$ or covariances $\mathbf{P}_m^{new}$, the maximum likelihood model parameters $\beta_m^{new}$, and finally the output variances ${\sigma_{m,y}^2}^n ew$; go back to (2) until the total data likelihood does not increase anymore [7].

### 15.2.3 Error Estimation and Characterization

From the probability density of the training data set (15.1) several error estimates and statistics can be derived, each of which provides useful insights as well as a self-consistency check on the model. The density itself indicates the model uncertainty in that we can't expect to obtain a valid model where the data density is low. The certainty of the model estimate is proportional to the data density in a subspace.

The conditional covariance (15.6) on the other hand indicates the prediction uncertainty given an input $\mathbf{x}$. It can be related to other characterizations of uncertainty, such as entropy and Lyapunov exponents. The differential entropy of a Gaussian process is $H = \log_2(2\pi e \sigma^2)/2$. Because only differences in a differential entropy matter, we ignore the additive and consider $H = \log_2(\sigma)$. The asymptotic rate of growth of the entropy with time is equal to the source entropy $h$, which in turn is equal to the sum of positive Lyapunov exponents times the time lag $\tau$ between samples, $h = \tau \sum \lambda^+$. Therefore, assuming that the prediction errors are roughly Gaussian, the asymptotic value of the log of the output width as the input dimension is increased provides a local estimate of the source entropy of the system. The sum of the negative exponents can similarly be found by analyzing the time series in reverse order (thereby exchanging positive and negative exponents).

Because clusters find the subspace that is occupied by data, we can use the cluster parameters to find the dimension of the data set even in a high-dimensional space. Intuitively, the number of significant eigenvalues of the local covariance matrices provides an estimate of the dimensionality of the data manifold. For example, we obtain three significant eigenvalues for the Lorenz attractor embedded in 6 dimensions (Fig.15.2). To quantify this further we use the eigenvalues of the local covariance matrices $E_m =$

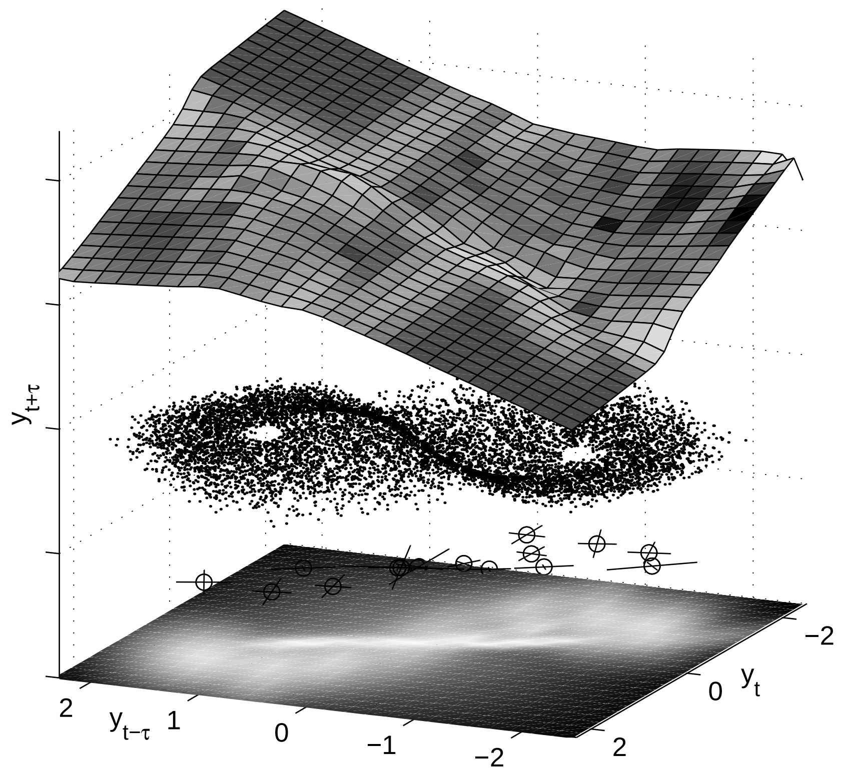

**FIGURE 15.1. The plot shows the Lorenz set, embedded in a three dimensional lag space. The dense dots show the embedded data. Below it are the cluster means and covariances, and the derived input density estimate; above it is forecasting surface shaded by the conditional uncertainty, showing the maxima associated with the orbit re-injection.**

$\{e_{1,m}, e_{2,m}, ..., e_{3,m}\}$ to evaluate the radial correlation integral

$$\begin{aligned} C_m(r) &= \int_{-r}^{r} ... \int_{-r}^{r} p(x_1, \ldots, x_D | C_m)\ dx_1 \ldots dx_D \qquad (15.21) \\ &= \text{erf}\left(\frac{r}{\sqrt{2e_{1,m}^2}}\right) \cdots \text{erf}\left(\frac{r}{\sqrt{2e_{D,m}^2}}\right) \end{aligned}$$

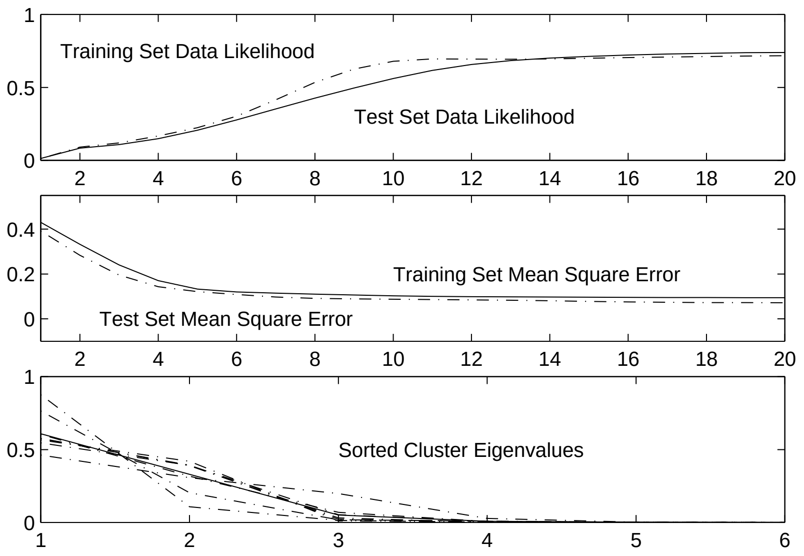

**FIGURE 15.2. Fitting the Lorenz set.** ***Top*****: Data likelihood as a function of iterations.** ***Middle*****: Mean square error as a function of iteration:** ***Bottom*****: Sorted eigenvalues of the local covariance matrices.**

which in turn lets us compute the cluster's correlation dimension [8] as

$$\begin{aligned}\nu_m &= \frac{\partial \log C_m(r)}{\log r} \qquad (15.22)\\ &= \sum_{d=1}^{D} \frac{1}{\operatorname{erf}\left(\frac{r}{\sqrt{2e^2_{d,m}}}\right)} \sqrt{\frac{2}{\pi e^2_{d,m}}}\; e^{-r^2/2e^2_{d,m}}\; r\end{aligned}$$

In the limit $r \to 0$, this dimension is equal to the dimension of the space , because locally the curvature of the clustered space can not be seen. If it is evaluated at $r = 0.1\sigma_{\max}$, for the $e_{\max}$ direction the contribution is still 0.997, but for a direction with variance $e_{\max}/100$ the contribution drops to $10^{-21}$. The expected dimension of the whole data set is finally given by the expectation

$$\langle \nu \rangle = \sum_{m=1}^{M} \nu_m \; p(c_m) \qquad (15.23)$$

Unlike a conventional $O(N^2)$ calculation of the dimension of a data set from all the inter-point pairs, the clusters find the significant places to evaluate the dimension, as well as the appropriate length scale at which to test the scaling.

## 15.3 Application: How to build a digital Strad

Mimic synthesis of musical instruments tries to infer models that behave and sound like the original instrument ideally to the extend that original and model become indistinguishable. Given this general goal there have been a variety of different modeling approaches. Global sampling, for example, has been particularly successful in commercial keyboard synthesizers. Each single note of a piano is recorded at many different volume levels and with varying duration and these sounds are replayed during synthesis. Since memory is cheap only very little interpolation between samples is required and the sound quality is close to the original recordings. However, the method works only for instruments with low dimensional control space, namely keyboard instruments. Since the model does not know about the instrument's internal state, but only reuses what it has seen before, there is no notion of control on part of the player.

Another successful synthesis technique is physical modeling [21]. It is based on first principles analysis of the acoustics of the instrument which are implemented in numerical methods. This method provides a lot of flexibility, for example it allows one to create new instruments that are derived from physical mechanisms but could not be implemented physically. However, the approach has also serious limitations. Current computers can barely run a full-scale model of the violin as can be shown in a simple calculation on a finite element approximation of a violin. Assumed 10 body modes per body axis and 10 finite element nodes per cycle, we get $10^4$ nodes per violin plate and in the order of $10^5$ nodes per instrument. If we multiply this by a CD quality sample rate of 40 kHz, we end up with roughly 10 Giga instructions per second needed to run a model in real time.

As a further fundamental problem of physical models there is no systematic parameter search within a model structure and an instrument family. Given a basic model of a violin there is no way to find the parameters that distinguish a Guanerius from a Stradivarius instrument other than trying out combinations of parameters in a high dimensional space.

The method we are presenting here lies conceptually in between the sampling and the physical modeling approach and hence is best described as a "physics-sampler". Although we infer our model from recorded data and even use stored samples, we create a model that has the flexibility of a physical model, since we synthesize the physics of the instrument, not the sound. At the same time we are doing computational compression on data, since the physical device is represented in an efficient description.

It was mentioned before that the mechanics of violin playing involve stochastic behavior. The stochastic aspects become clear when one considers player and instrument jointly. The violinist only partially controls her instrument. While she has an idea of the spectral characteristics she wants to achieve, she has no means to hear and control the phase of the produced signal. Naturally there is a causal relationship between the player action and the spectral content of the sound, whereas the phase of the different partials is random and hence unpredictable.

Fortunately, since phase is not perceived as a discriminating feature in a typical playing situation, we may pick it arbitrarily as long as we avoid discontinuities in the signal components. The general lesson to learn is that we need to model the process, not an instantiation of a particular process. While we can predict deterministic aspects of the signal, stochastic behavior needs to be summarized in appropriate statistics such as the power spectrum.

The violin, as most musical instruments, is characterized by slowly varying boundary conditions that map into a fast audio signal. The non-linear interaction between bow and string causes the slow player motion to be turned into the famous Helmholtz motion which contains the frequency components of the final audio signal [6]. The slow and fast elements describe two different times scales which, if mixed, confuse each other. Instead, fast and slow dynamics and the corresponding state variables need to be treated differently. CWM provides the means to implement such distinction: The slowly varying boundary conditions are used to select the domain of operation (cluster) in the configuration space (Equ.15.3), while the fast dynamics are handled by the local models and the associated state variables (Equ.15.4).

The previous section introduced CWM as a machine learning framework that allows one to predict and characterize arbitrary input-output data. Given this inference tool we need to consider a second important aspect of data analysis and prediction, which is data representation. Although linear transforms such as Fourier or wavelet transforms do not change the information content of the data, it makes a considerable difference in which domain we try to predict. CWM lets us embed a variety of specific local representations. In this section we discuss Cluster-Weighted Spectral Modeling and Cluster-Weighted Sampling as examples of two local implementations of CWM. We also introduce ways of higher order factorization and show how the CWM structure can be included in a Hidden-Markov Model to the end of explicitly encoding timing in the model.

### *15.3.1 Cluster-Weighted Spectral Modeling*

It is our goal to build an input-output model of a violin given a data set that contains physical input features measured on the bow and the finger-board along with synchronized audio data. In the training process the network learns the mapping between the physical input and the sound. After training the network knows how do generate appropriate audio, given new input.

We decompose the audio training signals into spectral frames at a frame rate that equals the sampling rate of the slowly varying physical input. Each frame contains of the coefficients of a Short Term Fourier Transform

(STFT) applied to a fixed number of audio samples weighted by a Hamming Window. The underlying assumption is that the player operates on the spectral composition of the sound and that these spectral characteristics do not change faster than the actual control. From those frames we retain only the harmonic partials of the violin signal. The amplitudes of the harmonic partials are taken to be the magnitude of the power spectrum in the frequency bin, while precise frequency estimates are obtained from the phase difference in closely spaced sample windows [13]. Given a total of $P$ partials the output vector $\mathbf{y}$ has $2P$ components.

The input vector $\mathbf{x}$ consists of physical input data, such as bow velocity, pressure, finger position, and bow-bridge position. Driven by the belief that past input conditions the current state of the instrument the input vector is augmented with respect to past input data. Adding time lagged input samples to $\mathbf{x}$, we balance the need to include the past and the burden of a big input space. While the model scales linearly in the output dimension, it is very sensitive to large input spaces, since the required amount of training data increases exponentially the input data dimension. Also the model is more sensitive to over-fitting given a bigger input space.

In training we use the set of vector pairs $\{\mathbf{y}_n, \mathbf{x}_n\}_{n=1}^N$ to train a CWM input-output model using simple linear local models of the form $\mathbf{y} = \beta_m \cdot \mathbf{x}$. In the synthesis process the vector of spectral information $\mathbf{y}$ is predicted from new input data $\mathbf{x}$. Given the spectral data we compute the time domain audio data by sinusoidal additive synthesis, where phase and amplitude of the partials are taken to be the predicted components, linearly interpolating between frames [19]. The final signal is obtained from summing the different components [18].

### *15.3.2 Cluster-Weighted Sampling*

Global sampling has been a successful synthesis technique for instruments with low dimensional control space, such as the piano [12]. However, the technique is less appropriate for instruments with continuous complex control, such as the violin. In the violin case the amount of data required to cover all possible playing situations is prohibitive, since control possibilities are essentially unlimited. To overcome this problem we parameterize the available sample material in an efficient way. CWM learns how to select the appropriate samples, but also to predict the parameters needed to reassemble the sound from the raw material.

Clusters now have multiple output models covering sample selection, amplitude prediction and pitch prediction. The first expert is a pointer into sample space. The cluster that most likely generated a control data takes over and its sequence of samples stand in for the particular playing situation. The cluster is replayed until another cluster becomes more likely and takes over with its own samples. We will come back to the issue of sequencing time domain sound samples below.

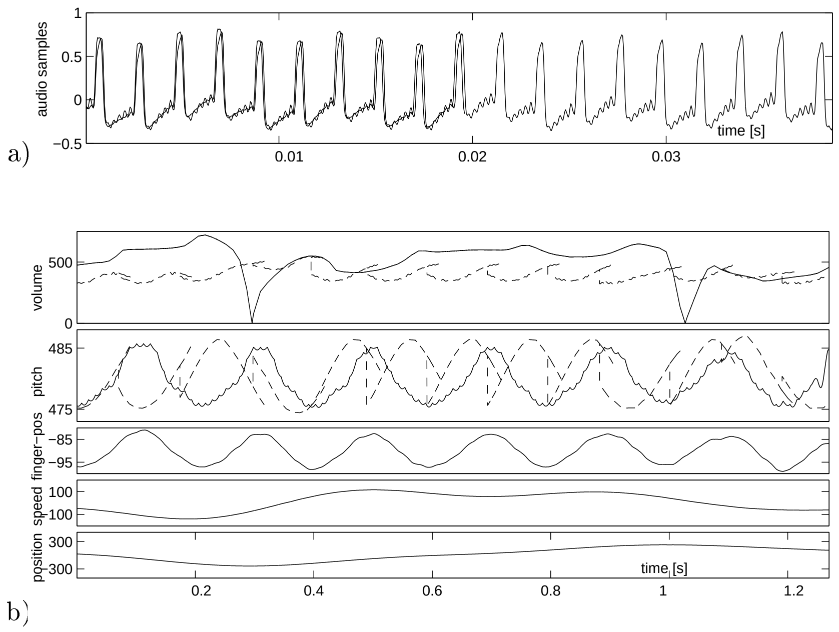

**FIGURE 15.3. Cluster-weighted sampling: a) overlapping samples of the string signal. b) input-output model,** *from the bottom*: **bow position; bow speed; finger position; predicted out-of-samples amplitude (solid) and given sampled amplitudes (dashed); predicted out-of-samples pitch (solid) and given sampled pitch (dashed); the doubled dashed lines indicate overlapping sample windows: the old window is slowly faded out while the new window is faded in, in such a way that the total weight of data adds up to unity at any given moment.**

The second output model is a pitch predictor. Given a control input that typically includes the left hand finger position on the finger board a local linear model predicts the appropriate pitch at any moment in time. The samples selected for synthesis almost certainly won't match this desired pitch exactly. Therefore they are re-sampled with respect to the predicted target pitch. The resampling is done in real time according to

$$\hat{s}(t) = \sum_{n=-N}^{n=N} s(n \cdot T_s)\ h_s(t - n \cdot T_s)\ , \tag{15.24}$$

with

$$h_s = (\min\{F_s/F'_s\}\mathrm{sinc}(\min\{F_s, F'_s\}t)\ , \tag{15.25}$$

where $F_s$ is the stored sampling frequency and $F'_s$ is the target sampling frequency [20]. Sample pitch and target should not differ too much, since big pitch shifts results in audible artifacts. However, resampling can easily compensate for effects such as vibrato. Since we cannot hope to record any possible vibrato sequence and frequency, we choose to superpose the desired vibrato behavior on the sampled material.

The third output model predicts the sound volume at any moment in time using, once again, simple locally linear predictors. The selected samples are re-scaled with respect to the target volume. Strong modifications of the sample volume should be avoided in order for the correct timbre not to be altered.

This approach requires a number of preprocessing steps that extract the high level properties from the audio data. We need both pitch and volume to label, parameterize and correct the audio data at any moment in time. These properties are easier to obtain than it may seem. Although pitch extraction is a problem that has not been solved in full generality, it turns out to be surprisingly simple in our approach. Since we are measuring physical input data, we have a rather good estimate of pitch to start with. Given a certain finger position, the possible pitch is within a very small frequency interval which makes it practically impossible for a pitch tracker to get confused in the audio analysis.

An important detail is the sequencing of pieces of audio when there is looping within a sample interval or when a change of cluster occurs. We choose to match samples by minimizing the least square error between the old and the new samples. Additionally we fade out the old sound and fade in the new sound using a Hamming window overlap-add.

Sine we re-sample the audio material anyway, we can increase the resolution of our fit allowing for non-integer alignment of sounds without increasing the complexity of the synthesis algorithm. The success of the overlap-add depends on the length of the permissible fading interval and on the character of the sound. Fig.15.3 shows the overlap of two highly phase coherent pieces of the string signal of a violin describing a Helmholtz motion. In that case the partials line up nicely with the fundamental and discontinuities are not a problem. However, the sound signal loses its regularity after the filtering by the bridge and the resonant body of the instrument, which makes it much harder to deal with.

### *15.3.3 Higher order factorization: Hierarchical mixture models and Hidden-Markov Models*

We have demonstrated a flat network structure that is easily applied to many problems and sufficiently complex for most applications. However, there are cases where additional hierarchical structure is helpful if not crucial. Identical models may want to be reused in different areas in the configuration space or systems may have long-term temporal dependences. [10] introduce mixture models of arbitrary hierarchical depth. Similarly we can add higher level factorization describing global states of our system. For

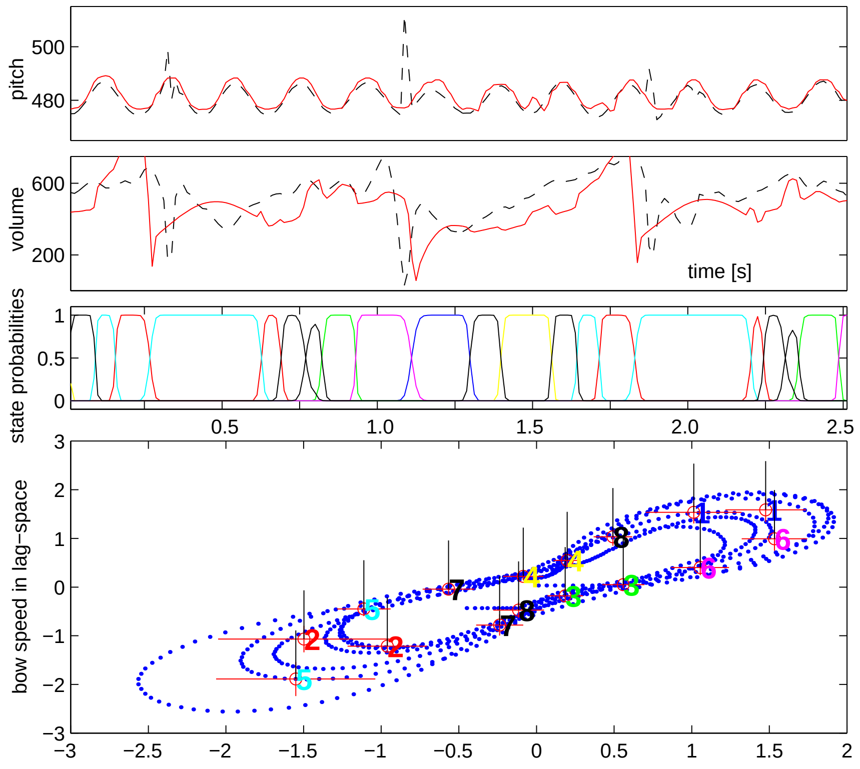

**FIGURE 15.4. Hidden-Markov model,** ***from bottom*****: cluster/model input space, two clusters per state; state probabilities; predicted out-of-samples amplitude (measured-dashed and predicted-line); predicted out-of-samples pitch (measured-dashed and predicted-line). Although measured and predicted data are visibly different, the reconstructed audio sounds very similar to the original audio data, since the spectral characteristics and the basic characteristics of the sound envelope are preserved.**

example, the top-level state of a violin model could distinguish global playing conditions such as pizzicato and arco playing or the use of a particular string. The probability density is then expanded as

$$p(\mathbf{y}, \mathbf{x}) = \sum_k \sum_m p(\mathbf{y}, \mathbf{x}, c_m, \text{Model}_k) \ . \tag{15.26}$$

In the previous sections we used time lags of the input signal to encode temporal structure and memory of the system. Another way of stating this dependence is to say that the current state depends on the past state and the current input. Hidden-Markov-Models (HMMs) have been developed to precisely implement this dependence in a probabilistic framework [17]. If we embed CWM in a HMM structure we obtain an input-output synthesis model with an explicit time dependence built into it.

HMMs are typically defined in terms of the number of distinct states $q_1, q_2, ..., q_N$; the state transition probability matrix $\mathbf{A} = \{a_{i,j}\}$, where $a_{i,j}$ denotes the probability that state $i$ follows state $j$ and the emission probability $b_j(k)$, which denotes the probability that the system generates observation $k$, given that it is in state $j$. We replace the discrete emission probabilities by a continuous probability density function of the form of $p(\mathbf{x}, \mathbf{y}|q_j)$, which means the cluster probabilities $p(c_m)$ become effectively time dependent, conditioned on past system states.

A cluster (or more than one) now represents a specific state $q_j$ given a set of possible states $q_1...q_N$. The likelihood of a sequence of input-output observations $(\mathbf{X}, \mathbf{Y}) = \{\mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2, ..., \mathbf{x}_T, \mathbf{y}_T\}$ is

$$p(\mathbf{X}, \mathbf{Y}) = \sum_Q p(\mathbf{X}, \mathbf{Y}|Q) \cdot p(Q) \ , \tag{15.27}$$

with

$$\begin{aligned} p(Q) &= \pi_{q_1} a_{q_1 q_2} a_{q_1 q_2} ... a_{q_{T-1} q_T} \ , & (15.28) \\ p(\mathbf{X}, \mathbf{Y}|Q) &= b_{q1}(\mathbf{x}_1, \mathbf{y}_1) \cdot b_{q2}(\mathbf{x}_2, \mathbf{y}_2) ... b_{qT}(\mathbf{x}_T, \mathbf{y}_T) \ . & (15.29) \end{aligned}$$

$b_{qi}(\mathbf{x}, \mathbf{y})$ is the emission probability of a pair $(\mathbf{x}, \mathbf{y})$ given the state $q_i$. These probability densities may be simple clusters or themselves a sum over clusters,

$$b_{qi}(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^{M} p(\mathbf{y}|\mathbf{x}, c_m) \cdot p(\mathbf{x}|c_m) \cdot p(c_m) \ , \tag{15.30}$$

where the probability distributions are identical to (15.3) and (15.4).

The model estimation is more complicated but is based on the very same probabilistic ideas as shown earlier. HMMs are typically trained in a forward-backward procedure which is a special implementation of EM and makes the estimation problem tractable. In synthesis the model is evaluated in a forward procedure since output has to be generated causally [17]. The output sequence at any moment in time is taken to be the expected value of $\mathbf{y}$ given estimated past states and current observed input,

$$\begin{aligned} p(q_{j,t}) &= \frac{\sum_{i=1}^{N} p(q_{i,t-1}) \cdot a_{j,i} \cdot b(\mathbf{x}_t|q_j)}{\sum_{j=1}^{N} \sum_{i=1}^{N} p(q_{i,t-1}) \cdot a_{j,i} \cdot b(\mathbf{x}_t|q_j)} & (15.31) \\ \langle \mathbf{y}_t | \mathbf{x}_t, q_{t-1} \rangle &= \sum_{j=1}^{N} \mathbf{f}(\mathbf{x}_t, \beta_j) \cdot p(q_{j,t}) \ . \end{aligned}$$

A particular sequence of states now reflects a sequence of input gestures and internal states of the violin. Fig.15.4 illustrates a state sequence for simple détaché bowing. We can follow a note from the attack, to the sustained part, to the next bow change.

## 15.4 Summary

The valuable insights that are possible into signals from complex systems have not penetrated into routine data analysis and engineering practice because of algorithms with limited applicability or reliability. The Cluster-Weighted Modeling framework that we have presented cannot of course solve all problems, but it does handle nonlinearity and stochasticity in a transparent fashion that provides a clear connection to past practice in a domain (through the choice of the local models), with just a single hyperparameter (the number of clusters). A natural extension exists for problems that require internal states in the model, without needing to incur the architectural uncertainty of more general graphical probabilistic networks.

One of the most valuable consequences of this probabilistic setting is the range of statistics that can be derived from the underlying model. Rather than impose a cost function for a learning algorithm at the outset, prediction questions can be answered directly from the density estimate. This is possible with reasonable amounts of data because the estimate is constrained by the local models. Further, the many possible kinds of characterization of the data are done more reliably in a context that can also make falsifiable predictions about the data, including internal consistency checks such as predicting the model's own errors.

The resulting models are efficient in storage and computation because the model resources are allocated only where there is data to describe, and the out-of-sample generalization is limited to the reasonable behavior of the local models. These features point to the possibility of broadly applicable "physics sampling," building phenomenological models of driven systems in the space of effective internal degrees of freedom, thereby enabling new applications that figuratively and literally sound great.

## Acknowledgments

We are grateful for support from the MIT Media Lab Things That Think Consortium.

## References

[1] Shunichi Amari. Information Geometry of the EM and em Algorithms for Neural Networks. *Neural Networks*, 8(9):1379–1408, 1995.

[2] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39:930–945, 1993.

[3] W.L. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 1996.

[4] Martin Casdagli. A dynamical systems approach to modeling input-output systems. In M. Casdagli and S. Eubank, editors, *Nonlinear Modeling and Forecasting*, Santa Fe Institute Studies in the Sciences of Complexity, pages 265–281, Redwood City, 1992. Addison-Wesley.

[5] W.S. Cleveland and S.J. Devlin. Regression analysis by local fitting. *J. A. Statist. Assoc.*, 83:596–610, 1988.

[6] Lothar Cremer. *The Physics of the Violin.* MIT Press, Cambridge, Massachusetts, 1984.

[7] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood From Incomplete Data via the EM Algorithm. *J. R. Statist. Soc. B*, 39:1–38, 1977.

[8] Neil Gershenfeld. *The Nature of Mathematical Modeling.* Cambridge University Press, New York, 1999.

[9] D. Heckerman and M. Wellman. *Bayesian Networks.* Communications of the Association Machinery. 1995.

[10] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[11] Michael Jordan, editor. *Learning in Graphical Models.* MIT Press, Cambridge, Massachusetts, 1998.

[12] Dana C. Massie. Wavetable sampling synthesis. In Mark Kahrs and Karlheinz Brandenburg, editors, *Applications of Digital Signal Processing to Audio and Acoustics*, pages 311–341. Kluwer Academic Publishers, 1998.

[13] R.J. McAulay and T.F. Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34 No.4:744–754, 1986.

[14] M.E. McIntyre and J. Woodhouse. On the fundamentals of bowed-string dynamics. *Acustica*, 43(2):93–108, 1979.

[15] Radford M. Neal. *Bayesian Learning for Neural Networks.* Springer, New York, 1996.

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

[17] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[18] B. Schoner, C. Cooper, C. Douglas, and N. Gershenfeld. Data-driven modeling of acoustical instruments. *Journal for New Music Research*, 28(2):81–89, 1999.

[19] Xavier Serra and Julius O. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[20] J. Smith and P. Gosset. A flexible sampling-rate conversion method. In *Acoustics, Speech, and Signal Processing, San Diego*, volume 2, 1984.

[21] Julius O. Smith. Physical modeling using digital waveguides. *Computer Music Journal*, 6(4), 1992.

[22] Floris Takens. Detecting strange attractors in turbulence. In D.A. Rand and L.S. Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381, New York, 1981. Springer-Verlag.