Tree-based Cluster Weighted Modeling: Towards A Massively Parallel Real-Time Digital Stradivarius

Edward S. Boyden III
e@media.mit.edu
Physics and Media Group
MIT Media Lab
20 Ames St.
Cambridge, MA 02139 USA

Abstract
Cluster-weighted modeling (CWM) is a versatile inference algorithm for deriving a functional relationship between input data and output data by using a mixture of expert clusters. Each cluster is localized to a Gaussian input region and possesses its own trainable local model. The CWM algorithm uses expectation-maximization (EM) to find the optimal locations of clusters in the input space and to solve for the parameters of the local model. However, the CWM algorithm requires interactions between all the data and all the clusters. For a violin, whose training might easily require over a billion data points and a hundred thousand clusters, such an implementation is clearly undesirable. We use a variant of CWM that makes "pseudosoft" splits in the data to model the time-series relationship between time-lagged values of human input data and digital audio output data. We describe how this tree implementation would lend itself to a multiprocessor parallelization of the CWM algorithm and examine the expected reduction in time and space requirements. We also consider how this method could be used to perform intelligent training of the violin.

1. Introduction

Among all great mechanical creations, classical instruments alone have been unsurpassable by modern technology; the great violins of Stradivarius and Guarneri have not seen their popularity fade with the years. At the MIT Media Lab, hyperinstruments, augmented instruments, and instruments with unusual interfaces have been developed with abandon, but for the past two years the Physics and Media group, and in particular Neil Gershenfeld, Bernd Schoner, and Eric Metois, have tried to recreate a 'digital violin' with traditional inputs and realistic sound. The theoretical basis for this agenda of reconstruction is solid, and we start with a fact from the theory of nonlinear dynamics.

In 1981 Floris Takens proved that for a smoothly flowing classical dynamical system, one can embed its d-dimensional phase space manifold into a $2d+1$-dimensional real space consisting of time-lagged values of a system observable y [1]. Since y is a function of the state, this implies that it is possible to predict the value of the observable y from $2d+1$ time-lagged values of itself.

For systems with inputs, it suffices to take 2*d*+1 lags of each input as well [2]. Therefore it is, in theory, possible to find a mapping between the time-lagged inputs and outputs of a system and future values of the output of the system. The solving of such inference problems has been a very active area of research for the past two decades.

Among Bayesian learning techniques, which try and maximize the probability of a model given observed data, one method has been suggested as particularly suitable for the violin problem. If we consider the problem of a one-string violin, we might naively attempt to map the instantaneous values of the finger position and the bow velocity to a sample of an output audio waveform. Takens' theorem suggests that we should actually take 2*d*+1 values of the finger position, bow velocity, and waveform amplitude. This representation is fraught with difficulty, including the fact that gigaflops of computing power would be required, but two facts limit the actual dimension that we have to emulate: 1) there is limited information contained in the digital waveform that represents the output sound (16 bit, 22kHz at the current moment), and 2) we only have to recreate the signal up to the perceptual limits of the human ear. Much personal experimentation over the past month has demonstrated that a mere three lags in bow velocity, two lags in finger position, and no lags in the output audio suffices to recreate the in-sample sound of the violin.

Two years of experimentation have also suggested an optimal way to represent a chunk of sound [3]. Our preprocessing program chops up the training data into 256/22050 ~ 1/86-second frames, and the program averages the finger position and velocity data over each of these intervals. As for the audio, we extract the frequencies and amplitudes of the 25 most important harmonics, so that the output space is 50-dimensional [3]. Thus these 1/86-second frames represent the input and output data at a discrete "instant," and they are the data that we use to train our network. When we want to synthesize new sound from new input data, we simply take time-lagged sets of bow velocity and finger position data, find the 50-dimensional output corresponding to these inputs, and synthesize a 1/86-second frame of sound from these harmonics. The relevant signal processing was developed by [4].

The method of machine learning used to solve the inference problem is cluster-weighted modeling, which is described in the next section.

2. Cluster-weighted modeling

The cluster-weighted modeling (CWM) algorithm was developed by Gershenfeld, Schoner and Metois [5]. We wish to reconstruct the probability density *p*(*y*, *x*) from *y* and *x*; we assume that we can expand the distribution over M clusters as follows:

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

We assume that the clusters have the following Gaussian probabilities:

$$p(x|c_m) = \frac{1}{(2p)^{d/2}|C_m|^{1/2}}\,e^{-(x-m_m)C^{-1}{}_m(x-m_m)/2} \tag{2}$$

$$p(y|x,c_m) = \frac{1}{(2p)^{d/2}|C_{ym}|^{1/2}}\,e^{-(f(b,x)-y)C^{-1}{}_m(f(x,b)-y)/2} \tag{3}$$

where *b* is a set of parameters such that *f(x,b)* is linear in the parameters *b*. One can show that (2) is equal to the assumption of softmax gating functions in the mixture of experts network, which is described in [6]. We can now apply expectation-maximization to (1) by letting the cluster means and variances be hidden variables. This gives the following equations for the cluster means and variances, equivalent to the analogous EM derivation of Gaussian mixture parameters in Bishop [7], although Gershenfeld derives it in a completely original way, using Monte Carlo integration, in [5]:

$$m_m = \frac{\sum x_n\,p(c_m|x_n,y_n)}{\sum p(c_m|x_n,y_n)} \tag{4}$$

$$s^2{}_m = \frac{\sum (x_n - m_m)^2\,p(c_m|y_n,x_n)}{\sum p(c_m|y_n,x_n)} \tag{5}$$

We derive the parameters for each cluster *m* using maximum likelihood:

$$0 = \P_b \left\langle \log(p(y,x)) \right\rangle_m = \P_b \left\langle \log(p(y|x,c) \right\rangle_m = \left\langle [y - f(x,b)]\frac{\P}{\P b} f(x,b) \right\rangle_m \tag{6}$$

which gives us, for a function *f* that is linear in the parameters *b*, i.e.,

$$f(x,b_m) = \sum b_{m,i}\,f_i(x) \tag{6}$$

the following solution

$$0 = \left\langle y f_j(x) \right\rangle - \sum b_{m,i} \left\langle f_j(x) f_i(x) \right\rangle \equiv a_j - \sum b_{m,i} B_{ij} \tag{7}$$

which yields the solution

$$b_m = B^{-1} a \tag{7}$$

which is just the pseudoinverse solution to the normal equations.

4. Tree-based algorithm

We devised the following tree-based algorithm to reduce the interactions between the various points and clusters, after Schoner [8]. Let *n* be the number of clusters that each cluster is subdivided into, *d* the fraction of duplicated points in the pseudosoft step, *L* the number of levels of the tree, *N* the total number of data points, and *M* the total number of clusters on the lowest level.

1) Run cluster-weighted modeling with *n* clusters on the current data set.
2) Assign each data point to the cluster for which the likelihood $p(x|c_m)$ is greatest.
3) Choose the *d*% next highest likelihoods and assign these points to their corresponding clusters, so that the total number of points on any level *L* is $N(1+d)$. This effects a pseudosoft sharing of commonly held clusters. (One point may be assigned to more than two clusters.)
4) With these assignments in mind, reiterate, running CWM on each of the smaller data sets and dividing the training data as appropriate.
5) On the last level, run ordinary CWM so that each small set of data is effectively clustered with *n* clusters.

The reconstruction step is analogous:

1) Given a data point *x*, evaluate the likelihoods $p(x|c_m)$ for each cluster in the next lower level.
2) Choose the most likely cluster, and repeat step 1 until you reach the last level.
3) On the last level, do ordinary CWM (i.e., evaluate the weighted sum of functions $f(x,b)$ and return the answer.

Thus instead of one CWM operation with *M* clusters and *N* data points, one performs on the order of *M/2* CWM operations, each with *n* clusters and a fraction of the number of data points. According to [2], this gives a speedup of $\frac{m}{M}(1+d)^{\log M/\log m}$ which can easily be over a thousand for the value of *M* we gave in the introduction, and reasonable values for *m* (say, 5) and *d* (~.1).

5. Implications

On a sample scale fragment, we found rough speedups of 50%-400% for choices of M in the range 25-125, with almost no decline in perceptual sound quality as compared to the ordinary cluster-weighted modeling method. Note that these are values of M that are much smaller than the actual expected values, so the eventual relative speedup will probably be much greater. Also, the tree driver code was completely written in Matlab, while the CWM code itself was written in C. The real speedup for an algorithm completely written in C could be significantly greater.

We also were able to use sufficient numbers of clusters such that synthesis of full A-major scales was possible (before this algorithm was developed, only single notes were efficiently synthesized). Complex signal analysis and synthesis is now accomplishable in a matter of minutes, as opposed to a matter of hours. Clearly this will be important as continued work on a realistic violin continues.

Furthermore, this tree-model allows the CWM code to be parallelized. The implications for parallelization are obvious: once a pseudosoft split of a batch of input data has been effected, the analysis trees are separate from there onward. Thus different processors can process the different data subsets in parallel. In the reconstruction stage, the likelihoods of each data point with respect to the different clusters on a level can be evaluated on separate processors. We have acquired a 6-processor SGI and plan to experiment with MPI acceleration of the routines.

Also, there remains the possibility of more intelligent training of the violin. What if the initial split was done with respect to finger position alone, and subsequent “fine-tuning” divisions were accomplished with respect to all the data? This coarse split would effectively be an additional set of priors based on finger position, and may improve the accuracy of the violin training. The effectiveness of this method has yet to be explored, but it looks promising.

In conclusion, this algorithm is a useful extension to current cluster-weighted-modeling and mixture-of-experts algorithms, when applied to datasets involving vast quantities of data and many different regions of operation.

References and Notes

1. Takens, F., "Detecting Strange Attractors in Turbulence", *Lecture Notes in Mathematica*, 366-381, Springer, 1981.
2. Schoner, B., *State Reconstruction for Determining Predictability in Driven Nonlinear Acoustical Systems*, Diploma Thesis, MIT Media Lab, Massachusetts Institute of Technology, May 1996.
3. Schoner, B. (private communication)
4. Bernd Schoner (Physics and Media, MIT Media Lab) wrote all of the signal processing code for creating frames, Kalman-filtering the bow velocity and finger position data, and synthesizing sound. We will start organizing these elements into coherent libraries soon.
5. N. Gershenfeld, B. Schoner, E. Metois, "Cluster-Weighted Modeling for Time Series Prediction and Characterization", preprint (1997).
6. Jordan, M. I., and Jacobs, R. A., "Hierarchical Mixtures of Experts and the EM Algorithm", *Neural Computation*, **6**, 181-214, 1994.
7. Bishop C. M., *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.
8. An outline of this algorithm is given in Schoner's thesis [4], for a different situation (at this point in the history of the violin, the output representation was still a pure waveform, not a reduced 50-dimensional representation, so the need for vast cluster-crunching algorithms was even more necessary!).

Thanks to Neil Gershenfeld for the original inspiration behind the project, for developing the CWM algorithm, and for writing the original implementation; to Bernd Schoner for lots of signal-processing code and help in understanding it, and for many interesting discussions; to Chris Douglas for helping assemble a coherent infrastructure for the violin software; and to Charles Cooper for many interesting discussions of the relevant music physics. (Digital Stradivarius Team 1997.)