Understanding Musical Sound with Forward Models and Physical Models

Michael A. Casey

Perceptual Computing Group

MIT Media Laboratory

E15-401c, 20 Ames Street

Cambridge, MA 02139

Phone: (617) 253-0116

e-mail mkc@media.mit.edu

Abstract

This research report describes an approach to parameter estimation for physical models of sound-generating systems using distal teachers and forward models, [Jordan and Rumelhart, 1992, Jordan, 1990]. The general problem is to find an inverse model of a sound-generating system that transforms sounds to action parameters; these parameters constitute a model-based description of the sound. We first show that a two-layer feed-forward model is capable of performing inverse mappings for a simple physical model of a string. We refer to this learning strategy as direct inverse modeling; it requires an explicit teacher and it is only suitable for convex regions of the parameter space.

A model of two strings was implemented that had non-convex regions in its parameter space. We show how the direct modeling strategy failed at the task of learning the inverse model in this case and that forward models can be used, in conjunction with distal teachers, to bias the learning of an inverse model so that non-convex regions are mapped to single points in the parameter space. Our results show that forward models are appropriate for learning to map sounds to parametric representations.

1 Introduction

When we listen to music we perceive more than pitches, durations and a general sense of timbre; we also recover detailed gestural information, such as how hard an instrument is bowed, and even which part of the bow is on the string. As listeners we correlate what we hear with our gestural understanding of the perceived sound model. The interpretation of gestural information is part of the experience of listening to music and, more generally, that of listening to any sound. Our understanding of musical instruments, for example, is a combination of *implicit* and *explicit* knowledge of their input and output behaviors. Musicians may correlate gestural information with musical signals on a much finer scale than lay listeners because they possess a more detailed understanding of musical instruments, from direct physical experience, and are thus able to parameterize according to their *internalized* instrumental models. Lay listeners, however, may map musical information in a more abstract manner, relating to sound models other than musical instruments, voice for example, or perhaps not in terms of sound at all; however, such speculation is beyond the scope of this paper.

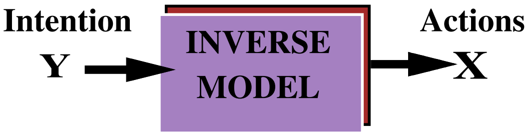
We present a general technique for recovering gestural information for sound models from audio signals. Our claim is that such information is part of the mechanism by which we learn to understand and recreate sounds. If we can parameterize a sound environment in terms of learned models of sound-generating systems, we have achieved a form of understanding. Our goal is to show that physically meaningful parameters can be recovered from audio signals if they closely fit a learned sound model. Although we develop our examples in terms of recovering physically parameterized data for a sound model, it is also possible that more abstract feature mappings could be learned, such as those described by [Grey, 1975]. The learning paradigms presented below are suitable for parameterized mappings to many classes of sound model.

2 Direct Inverse Modeling

The obvious starting point for the problem of learning to map a sound to a parametric representation is to use the direct inverse modeling strategy. The model learns the inverse mapping by reversing an observed set of inputs and outputs for the instrument, producing a functional mapping between them; instead of a physical action producing a sound, we want the sound to produce the physical action.

An example of a direct solution to the inverse modeling problem is classical supervised learning. The learner is explicitly presented with a set of *sound, action-parameter* pairs observed from the physical model, $\{\mathbf{y}^\star, \mathbf{x}^\star\}$, and is trained using an associative learning algorithm capable of non-linear mappings. Once trained, the learner has the ability to produce *actions* from sound *intentions*, hopefully with good generalization. This technique is only suitable for modeling data that is convex in the region of the solution space that we are interested in, see Figure 1.

Figure 1: Direct Inverse Modeling

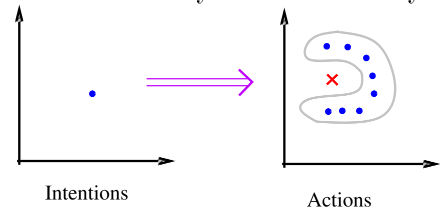


The direct inverse modeling strategy takes intentions as input and maps them to actions. This is the reverse function of a physical environment, which takes actions as input and maps them to an outcome. Classical supervised learning is an example of such a learning system.

2.1 Direct Inverse Modeling of a Convex Solution Space

Solution regions and learnability are well-studied characteristics in the machine learning community; see for example, [Anthony and Biggs, 1992, Haussler, 1989, Minsky and Papert, 1969]. A solution set X is said to be convex if and only if there exist three co-linear points p, q, r such that, if $p \in X$ and $r \in X$, then $q \in X$; otherwise the region is non-convex. See Figure 2.

Figure 2: Non-Convexity of One-to-Many Mappings

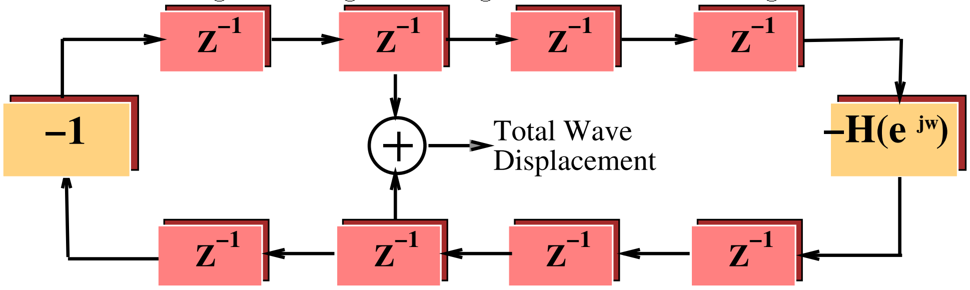


The point on the left should be mapped inside the bounded region on the right, but the multiple solutions are averaged to the cross in the center of the graph. Since the cross lies outside of the solution region (the bounded area) the problem is non convex.

As an illustration of modeling a convex solution space we implemented an inverse model to learn to map a sound waveform, generated by a physical model of a single violin string, to the physical stop position on the string. This mapping was unique, and therefore convex, since for every sampled waveform that was produced using the physical model, there was only one stop-parameter value.

The violin string model was implemented using a discrete version of the wave equation efficiently computed using digital waveguides and linear time-invariant filters for damping, dispersion and resonance characteristics: [Smith, 1986, Smith, 1992], see Figure 3. The parameterization of the violin model is given in Table 1.

Figure 3: Digital Waveguide Model of a String



A single string can be modeled with a bi-directional delay line representing the right (top) and left (bottom) traveling components of the wave. The output of the string model is the sum of the two waves at a given output point. The two ends of the string reflect the wave back in the opposite direction. The dispersion, damping and resonance characteristics of the string can either be collected in a single filter, as in this diagram, or they can be distributed accross separate filters.

Table 1: String Model Parameters

Symbol	Description	Units
d	initial string displacement	m
l	total string length (nut to bridge)	m
c	string velocity	m/s
s	stop position	m

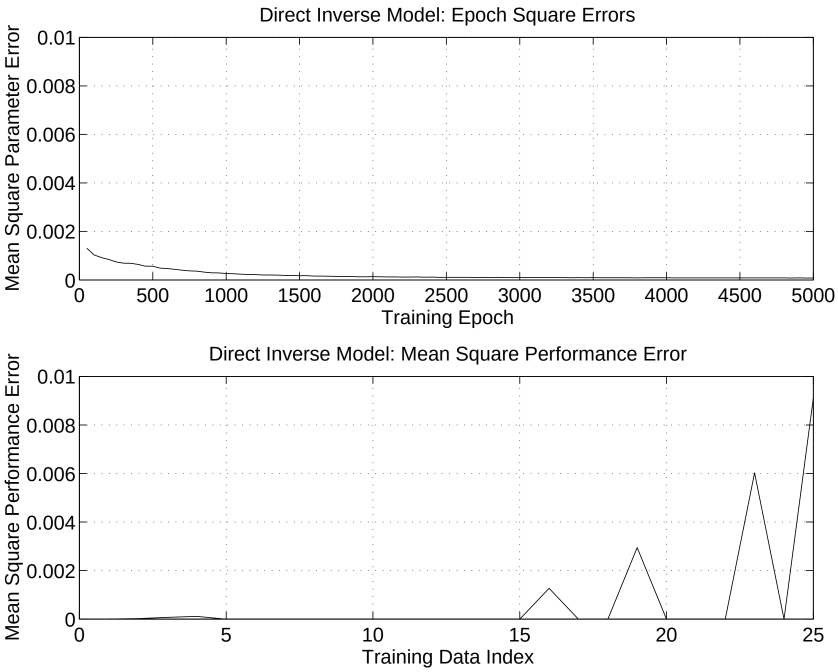
The first experiment required only the D string of the violin. The length of the violin from the nut to the bridge was 0.32 m and the model was calibrated so that the pitch class $A4$ was at 440.000 Hz, thus the open D string had a fundamental frequency of $f_0 = 293.665$ Hz. The speed of wave propagation in the string was determined by $c = \sqrt{\frac{K}{\epsilon}}$ where K was the string tension and ϵ was the linear mass density of the string; for the D string the wave propagation speed was 187.9456 m/s.

The training set for the direct inverse model comprised a set of time-domain waveforms generated by the violin model, $\mathbf{y}^{\star}$, and a set of target parameters that produced the waveforms, $\mathbf{x}^{\star}$. The original waveforms were represented at 16-bit resolution with floating-point values in the range 0-1. We used the first 61 samples generated by the physical model as the representative set for each of the waveforms; this allowed frequencies as low as 293.665 Hz ($D4$) to be uniquely represented. The waveforms were generated at frequencies spaced a half-step apart along the D string, spanning two octaves starting in open position (0.32 m). The stop position for each of the waveforms was expressed as distance along the string.

The direct inverse model was implemented as a two-layer feed-forward network with biases, utilizing the generalized delta-rule as a learning algorithm, [Rumelhart et al., 1986]. There were 61 linear input units, one for each sample of the sound intention $\mathbf{y}^{\star}$, 20 logistic hidden units and a single linear output unit for the stop position. The training pairs were presented in random order with the entire set of data being presented in each epoch. We used an adaptive learning-rate strategy and included a momentum term for faster convergence.

Figure 4 shows the convergence of the parameter errors in the inverse model for 5000 epochs of the training data, and the mean-squared performance error for each of the training patterns after the inverse model reached criterion. The parameter error is the difference between the target actions

Figure 4: Convergence and Mean Errors of Direct Inverse Model: Convex Data



The upper graph shows the convergence of the direct inverse model to the *parameter* error criterion ≤ 0.0001. The lower graph shows how the *performance* error is distributed across the training set. The mean-squared performance error was ≈ 5.6 bits.

$\mathbf{x}^\star$ and the output of the inverse model $\hat{\mathbf{x}}$:

$$\mathbf{J}_{param} = \frac{1}{2}(\mathbf{x}^\star - \hat{\mathbf{x}})^T(\mathbf{x}^\star - \hat{\mathbf{x}}) \tag{1}$$

The performance waveforms and the squared performance errors are shown in Figure 5. The performance outcome was computed by applying the outputs of the inverse model $\hat{\mathbf{x}}$ to the inputs of the physical model. The performance error compares the the input waveform $\mathbf{y}^\star$ to the outcome waveform $\mathbf{y}$:

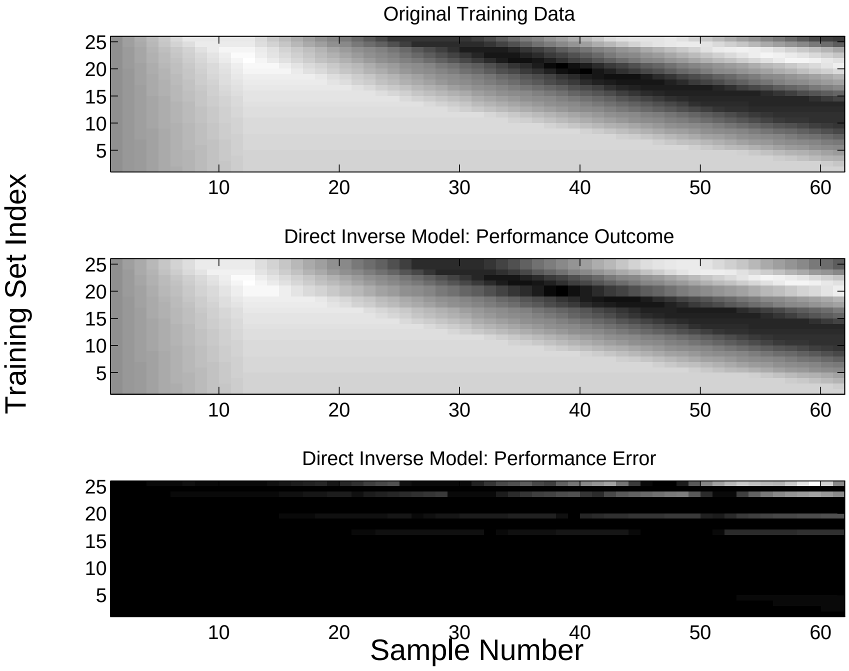
$$\mathbf{J}_{perf} = \frac{1}{2}(\mathbf{y}^\star - \mathbf{y})^T(\mathbf{y}^\star - \mathbf{y}) \tag{2}$$

The mean-squared performance errors are given by:

$$Perf_{mse} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{M}\sum_{j=1}^{M}\left(\mathbf{y}_j^{\star(i)} - \mathbf{y}_j^{(i)}\right)\right) \tag{3}$$

where N is the number of training patterns, M is the number of samples in the waveform.

Figure 5: Performance Outcome of Direct Inverse Model: Convex Data



The upper two images show gray-scale plots of the waveforms for the convex training set and the direct inverse model's performance outcome; dark regions are small values. The lower image shows the performance squared error.

The original waveforms had 16 bits of resolution; the mean-squared performance error of the direct inverse model after convergence to criterion was 7.8267×10^{-4}. The accuracy of the performance was given by $16 + \log_2 7.8267 \times 10^{-4} \approx 5.6$ bits. This was the *performance* accuracy of the inverse model when trained to a mean square *parameter* accuracy of ≤ 0.0001. The accuracy of the direct inverse model of the convex data set was acceptable for our purposes. (The typical noise margin for digital recording ≈ 6 bits).

The evaluation of the model in this manner was purely a matter of convenience for illustration purposes. If we were interested in developing a perceptual representation of auditory information we would not use the error critereon cited above, which reflects the ability of the system to *reconstruct* the original data. For more sophisticated applications of inverse modeling for audio data, we would need to develop perceptual error measures, ensuring that the machine makes judgements that are perceptually valid in human terms; see, for example, [Grey, 1975, Lee and Wessel, 1992].

2.2 Direct Inverse Modeling of a Non-Convex Solution Space

By adding another degree of freedom to the violin string model we made the task of learning the inverse model a much harder problem. In the next experiment we added a second string; the learning task was to map a set of waveforms to parameters representing the string stop positions (as in the previous experiment) as well as a unit representing string selection, in this case the D string ($f_0 = 293.665$ Hz) and the A string ($f_0 = 440.000$ Hz). In the physical-model implementation, the fundamental pitch f_0 of each string in open position was determined by the speed of propagation of the wave through the string, $c = \sqrt{\frac{K}{\epsilon}}$. For the D string the speed of propagation was 187.9456 m/s and for the A string it was 281.6000 m/s.

The set of stop positions spanned two octaves for the D string ($D3 - D5$) and an augmented eleventh for the A string ($A4 - D\sharp5$) spaced at half-step intervals. The pitch ranges were determined by the resolution of the physical model since it was implemented as a digital waveguide with unit delays. The highest frequency for half-step resolution, without adding fractional delays to the model, is given by $\frac{SR}{n}$, where SR is the sampling rate of the physical model and n is the number of delays used to model the string. The minimum number of delays required for a half-step resolution has to satisfy the inequality:

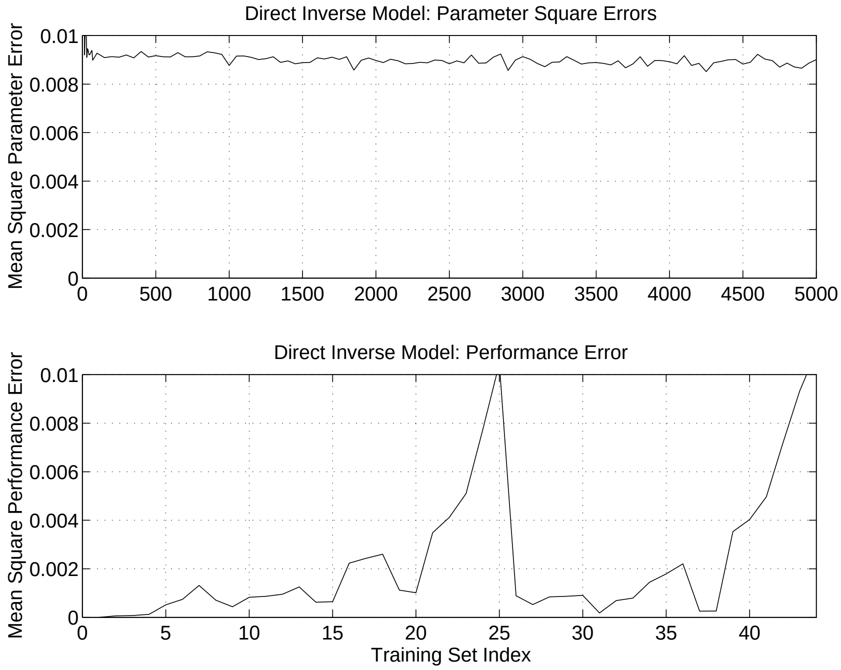
$$n \leq 2\lceil \frac{-1}{1-2^{\frac{1}{12}}} \rceil \tag{4}$$

This gave $n \leq 34$ for the required half-step interval resolution. The sample rate was 44100 Hz, thus the highest frequency was 1297.1 Hz, approximately $D\sharp5$.

There was considerable overlap in the training data because the waveforms from the string model for pitch classes $A4$-$D5$ on both strings were exactly equivalent. With this overlap the solution space was non-convex; thus solutions that averaged the multiple parameter sets for each duplicated waveform were not valid.

To show how this applies to the problem of inverse modeling we used the direct inverse modeling strategy of Section 2.1 on the the non-convex training data. Figures 6 and 7 show the results obtained using the two-layer feedforward network described above, with an extra output unit representing the choice of string. The model failed to converge to criterion over 5000 epochs, so the performance error was significant in some regions of the solution space. The mean-squared performance error was 0.0024; using the same calculation for the accuracy as for the convex data set we got $\log_2 0.0024 + 16 \approx 7.3$ bits of error, which was significantly worse performance than for the convex data set.

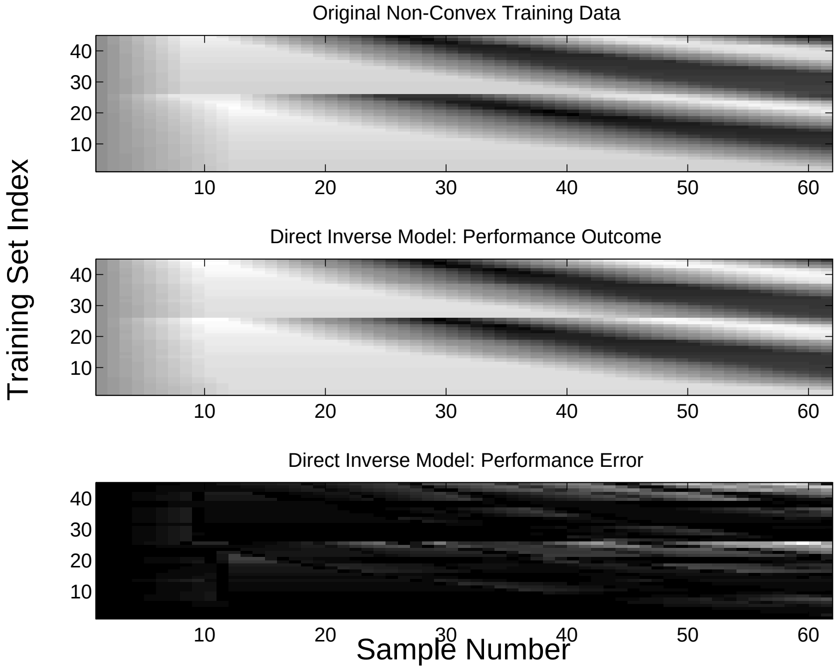
Figure 6: Convergence and Mean Errors of Direct Inverse Model: Non-Convex Data



The upper graph shows the mean-square parameter error for 5000 epochs of the non-convex training data. The inverse model failed to converge to a criterion of 0.0001 for this data. The lower plot shows the performance errors resulting from the parameters given by the direct inverse model. The mean-squared performance error was ≈ 7.3 bits.

These results show that direct inverse modeling using a two-layer feed-forward network gave unsatisfactory results for non-convex training data. We improved on the accuracy of the inverse model by implementing a learning technique that was better suited to non-convex data set.

Figure 7: Performance Outcome of Direct Inverse Model: Non-Convex Data

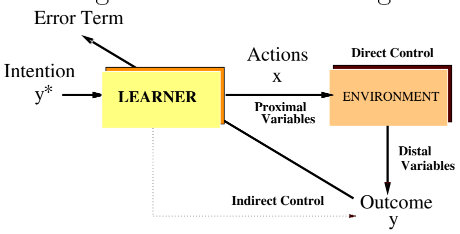


The upper two images show gray-scale plots of the waveforms for the non-convex training set and the direct inverse model's performance outcome; dark regions are small values. The lower image shows the performance squared error.

3 Forward Models for Non-Convex Data

A forward model is a learned approximation of the physical environment and it is used in series with an inverse model to form a composite learning system. This learning system is capable of solving the inverse mapping for non-convex regions of the solution space,[Jordan, 1990, Jordan and Rumelhart, 1992]. The training technique for the composite model is called *distal learning* and it is illustrated in Figure 8. The learner controls a distal outcome via a set of proximal variables which are inputs to a physical environment, in our case a physical model of two violin strings. The variable names and their functions for the composite system are outlined in Table 3.

Figure 8: Distal Learning



The distal learning paradigm uses errors collected at the output of the environment to drive the learning of the proximal variables. The learner's task is to find the set of action parameters that produce the intended outcome.

Table 2: Simulation Input and Output Variables

Symbol	Variable	Description
$\mathbf{y}^\star$	Intention (Target Outcome)	Sampled Waveform
$\hat{\mathbf{y}}$	Predicted Outcome	Approximated Waveform
$\mathbf{y}$	Actual Outcome	Waveform Output from Physical Model
$\hat{\mathbf{x}}$	Actions	Estimated Action Parameters
$\mathbf{x}^\star$	Target Actions	Training Action Parameters

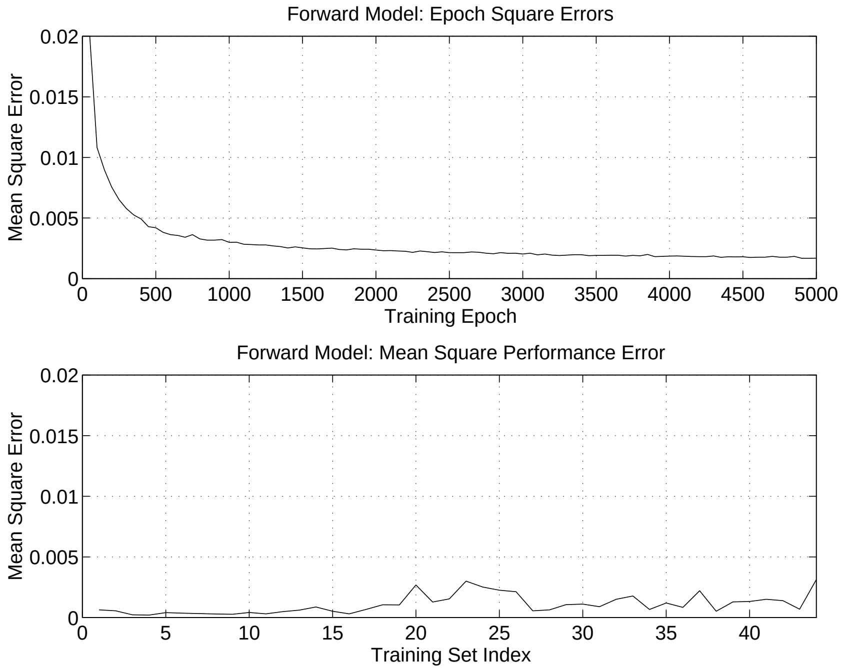
3.1 Training the Forward Model

The training set for the forward model comprised pairs of *action parameters* and *sound outcomes*. We used the same non-convex training set as for the direct inverse model but with the inputs and outputs reversed. Once learned, the forward model was able to approximate the input/output behavior of the physical model. The output of the forward model is called the *predicted outcome* $\hat{\mathbf{y}}$ and the difference between the sound intention $\mathbf{y}^\star$ and the predicted outcome $\hat{\mathbf{y}}$ is the *predicted performance* error:

$$\mathbf{J}_{pred} = \frac{1}{2}(\mathbf{y}^\star - \hat{\mathbf{y}})^T(\mathbf{y}^\star - \hat{\mathbf{y}}) \tag{5}$$

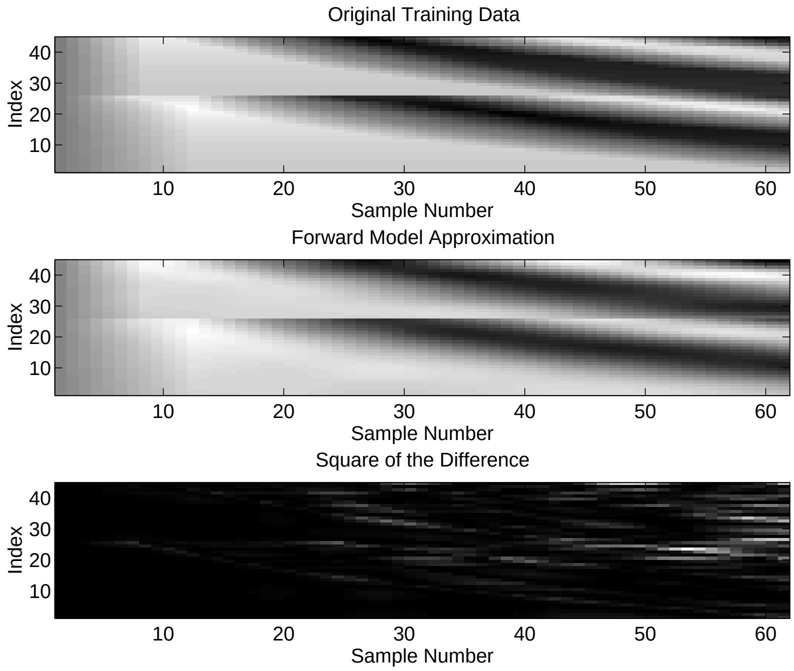
We used this error for optimizing the forward model. As in the two previous experiments, the forward model was implemented as a two-layer feed-forward network with 2 linear units for inputs, representing string selection and stop positions, 20 logistic hidden units and 61 output units corresponding to the sample estimates of the violin string models. The forward model was trained until the predicted performance error reached an accuracy of $\approx$ 6 bits, (mean-squared predicted performance error ≤ 0.001). Figure 9 shows the convergence of the forward model to within the chosen treshold. The lower graph in figure 9 shows the mean-squared predicted performance error for each of the training patterns.

Figure 9: Convergence and Mean Errors of Forward Model



The upper graph shows the convergence of the forward model's mean-squared *predicted* performance error. The lower figure shows the mean-squared predicted performance error, $\approx$ 6 bits.

Figure 10: Predicted Performance Outcome of Forward Model

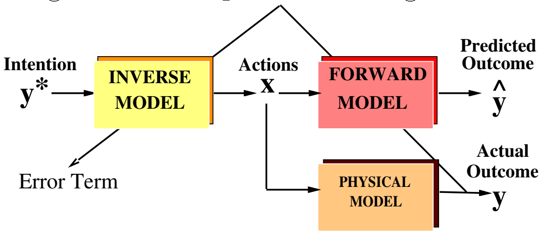


The upper two images show the waveforms for the training set and the forward model's approximation; i.e. the predicted performance. The lower image shows the predicted performance squared error.

3.2 Training the Inverse Model using the Forward Model

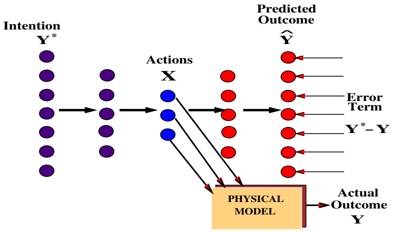
Once the forward model was trained we placed it in series with the inverse model to form a composite learning system of the type shown in Figure 11. Again, the inverse model was implemented as a two-layer feed-forward network with 61 input units, 20 logistic hidden units and 2 output units. The composite model can also be thought of as a single four-layer feed-forward network, see Figure 12. First the intention waveform $\mathbf{y}^{\star}$ was given to the input units of the inverse model. The activations were fed forward, through the inverse model, to the forward model. The activations passed completely through the four layers of the network until values were obtained for the output of the forward model. We then recursively computed the deltas for each of the 4 layers and adjusted the weights of the inverse model, leaving the forward model unchanged since it had already converged to a satisfactory solution.

Figure 11: Composite Learning Scheme



A forward model is placed in series with the inverse model and in parallel with the physical environment. The performance errors are propagated back through the forward model to the inverse model but the forward model's weights are left unchanged. The entire composite system is designed to produce accurate *actions* by optimizing the outcome.

Figure 12: Connectionist Implementation of Composite Learning Scheme



The distal learning architecture is implemented as two feed-forward networks placed in series; the first is the inverse model and the second is the forward model. The system can also be thought of as a single four-layer network. The forward model's outputs are replaced by the physical model's outputs so that the performance error is used for optimization. The outputs of the composite network occur in the middle since it is the *actions* that we want to obtain.

There are three approaches to training the inverse model using a forward model: training from random initial conditions, training from the direct inverse model's final condition and training with the *predicted performance error* of Equation 5.

Training from random initial conditions is standard practice for many applications of connectionist networks. The strength of the connections are initialized to small ($|w| \leq 0.1$) uniformly distributed values with zero mean.

As the network converges to a globally satisfactory solution, the weights get larger, representing a progressively higher-order fit to the training data. Initializing the network with small weights ensures that the model does not over-fit the data.

If we initialize the distal inverse model with the weights obtained from the direct inverse model the task of the composite learning system is made somewhat easier and we observe faster convergence than for random initial conditions. This technique works because the direct inverse model is good for convex regions of the solution space; if the inverse modeling problem has relatively small non-convex regions the difference between the direct inverse model and distal inverse model will be small.

We used the performance error for optimization during learning with these models, see Equation 2, which is different than the *predicted* performance error of Equation 5.

However, we also obtained good results with faster convergence by using the predicted performance error; the difference was that the predicted performance error used the forward model's outputs as an error measure so there was no need to present the action parameters to the physical model. An inverse model trained in this way is biased by the inaccuracies of the forward model, thus we switched to performance error optimization for the last few epochs of training. This technique is only effective if the forward model has converged to a good approximation of the physical model. We initialized the inverse model with the direct inverse model's final values. See Table 3 for a summary of the various error functions and training sets that were used for the above models.

Table 3: Training Sets and Error Terms for the Various Models

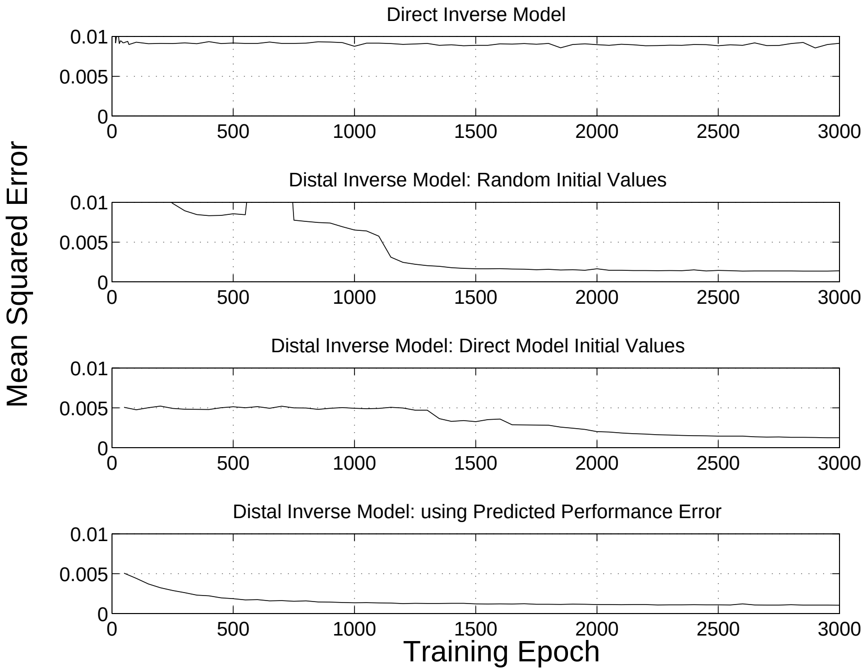
	Forward Model	Direct Inverse Model	Distal Inverse Model
Training Set	Actions, Outcomes $\{\mathbf{x}^\star, \mathbf{y}^\star\}$	Intentions, Actions $\{\mathbf{y}^\star, \mathbf{x}^\star\}$	Intentions $\{\mathbf{y}^\star, \mathbf{y}\}$
Optimization Error	Predicted Performance Error $\frac{1}{2}(\mathbf{y}^\star - \hat{\mathbf{y}})^T(\mathbf{y}^\star - \hat{\mathbf{y}})$	Parameter Error $\frac{1}{2}(\mathbf{x}^\star - \hat{\mathbf{x}})^T(\mathbf{x}^\star - \hat{\mathbf{x}})$	Peformance Error $\frac{1}{2}(\mathbf{y}^\star - \mathbf{y})^T(\mathbf{y}^\star - \mathbf{y})$

Figures 13, 14 and 15 show the results of training the inverse model using the three forward-modeling strategies outlined above, as well as the results for the direct inverse modeling technique.

We can see from Figure 13 that the fastest convergence was given by the the third of the non-direct techniques, which used the predicted performance error for most of the training epochs. After 3000 trials the three distal-inverse models had converged to a solution that met the error criterion, the direct inverse model had not.

The mean-squared performance errors for the entire training set of the two-string violin model are shown in Figure 14. The performance errors, shown in Figure 15, are concentrated in smaller regions for the three distal-inverse models than for the direct inverse model. The inverse model with the best overall performance was the distal inverse model trained from the direct inverse model's final values. The mean-squared performance error for this model was 5.1136×10^{-4} which gives an accuracy of $\log_2 5.1136\times10^{-4}+16 \approx$

Figure 13: Convergence of the Inverse Models: Non-Convex Data



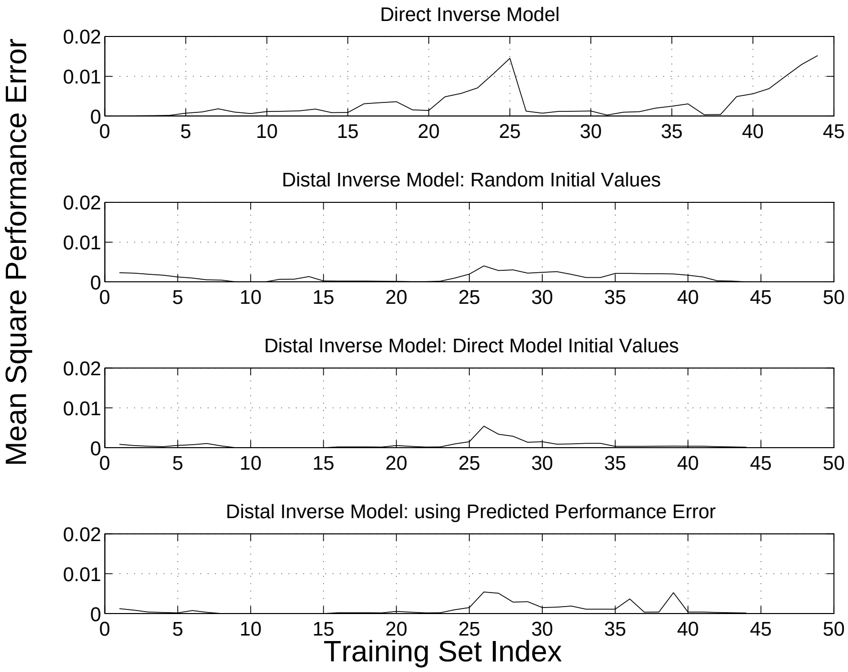
Each of the four figures shows the evolution of the mean-squared error for 3000 epochs of the non-convex training set. The upper graph shows the mean-squared parameter error for the direct model, the remaining three graphs show the mean-squared performance error for the distal inverse models.

5 bits. This is significantly better performance than for the direct inverse model and is well within the required criterion of 6 bits of error.

4 Performance on Novel Data

In order to evaluate the generalization capabilities of each of the inverse modeling techniques we constructed a novel data set comprising target sounds that required parameter values that were not in the original training set but that were generalized as a result of the learning. We produced a new set of waveforms using the D string on the violin with stop positions that were in quarter-tones with the training set. The frequencies available at a quarter-tone resolution we were limited by:

Figure 14: Mean Performance of the Inverse Models: Non-Convex Data



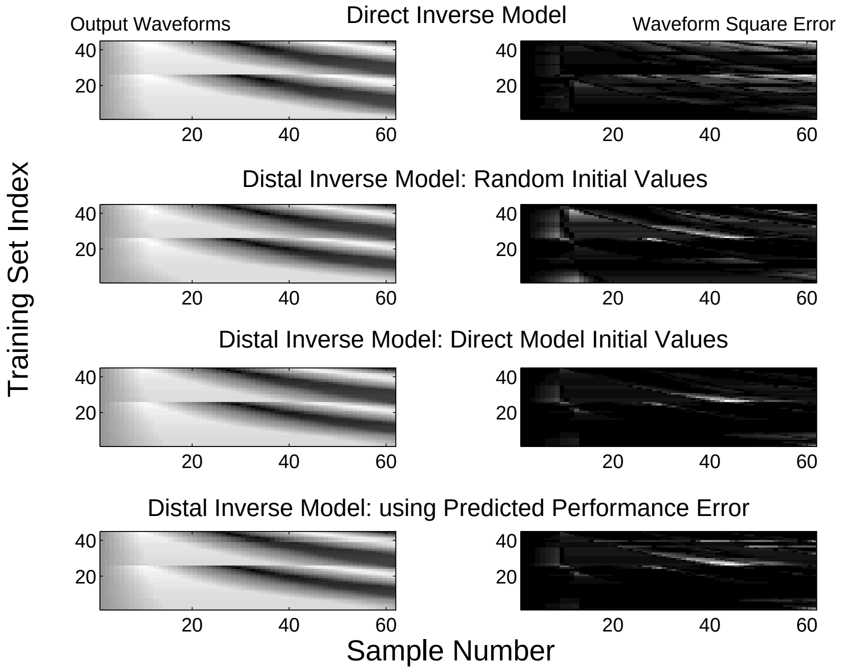
The four figures show the mean-squared performance errors of each of the direct inverse modeling strategies for all waveforms in the convex training set. The best performance was the distal inverse model initialized with the direct model's final state, the mean-squared error for this model was ≈ 5 bits.

$$n \leq 2\lceil \frac{-1}{1 - 2^{\frac{0.5}{12}}} \rceil \tag{6}$$

which gave $n = 69$. Therefore $f_{max} = \frac{44100}{69} = 639.13Hz$ ($D\sharp 5$). So we had to limit the testing set to 14 waveforms computed in the range $D4$ - $D\sharp 5$.

Figures 16 and 17 show the distribution of errors in the output of each of the inverse models. The inverse model with the best overall performance on the novel data was the model trained using the predicted performance error. The mean-squared performance error for this model was 1.5724×10^{-4} giving an error of $\log_2 1.5724 \times 10^{-4} + 16 \approx 3.4$ bits. The accuracy is better

Figure 15: Performance Outcomes of the Inverse Models: Non-Convex Data



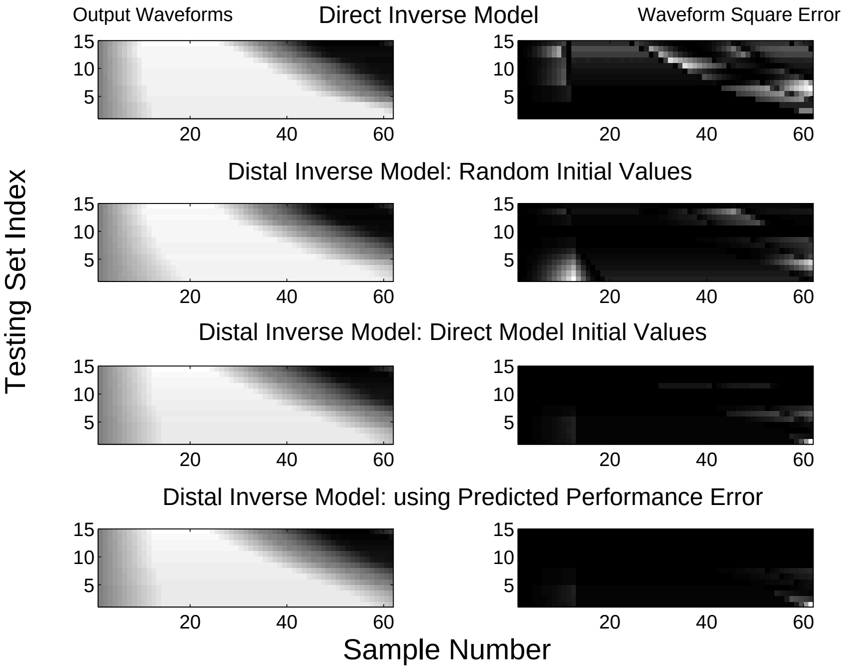
The images on the left show gray-scale plots of the output waveforms of the physical model given the action parameters from each of the inverse models. The images on the right show the squared performance error; dark regions are small values.

than for the original training data because we were testing the model in a small range of the problem space, due to the limited frequency resolution of the physical model. The results show that the generalization capabilities of distal inverse models are good for the given problem domain.

5 Conclusions

In this paper we have shown that inverse modeling techniques can be used to map representations of sound to physical parameters for sound-generating models. The inverse modeling strategy depends on the geometry of the solution space. If the solution region is convex we can use a direct inverse-

Figure 16: Mean Performance of the Inverse Models: Novel Data

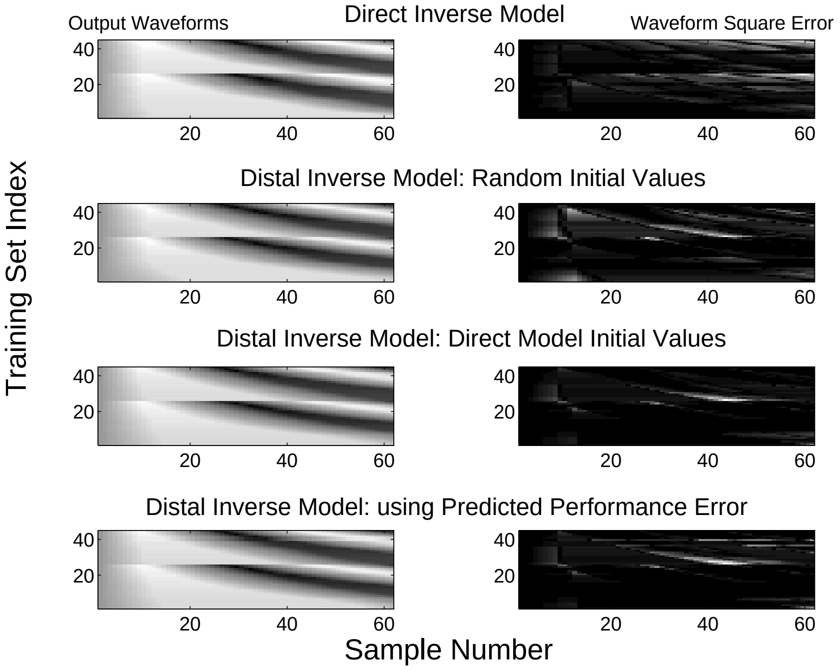


The four figures show the mean-squared performance error for each of the inverse models given novel data as input. The distal inverse model trained using the predicted performance error gave the best performance with ≈ 3.4 bits error.

modeling strategy, such as back-propagation in a two-layer, feed-forward network. However, non-convex solution regions require a more sophisticated approach to deriving the inverse model. One such approach is that of using distal teachers with forward models. We implemented such a system and obtained satisfactory results for recovering physical parameters for models of violin strings.

With careful implementation, the forward modeling strategy is general enough to be applied to many inverse modeling problems in the auditory domain. We are currently expanding the scope of the current research to include models of other sounding systems; e.g. single-reed, brass and vocal-tract models. The outputs of these inverse models can be treated as features to which we can apply pattern recognition techniques for source classification

Figure 17: Performance Outcomes of the Inverse Models: Novel Data



The images on the left show gray-scale plots of the output waveforms of the physical model given the action parameters of each of the inverse models in response to novel data. The images on the right show the squared performance error; dark regions are small values.

and gesture recognition. An example of this is to use parameters recovered from real musical performances to classify different playing styles or performance techniques, perhaps creating a machine listening system that can understand the subtleties of musical performance.

Future work will include the development of distance functions for auditory data that take into account human perceptual factors. Time-domain representations are unsatisfactory for many applications of audio inverse modeling; there are many different time-domain representation of a signal that produce a single auditory percept. This is due, in large part, to the low salience of phase in the human auditory system. We have experimented with a constant-Q frequency representation which better represents the perceptual distance between auditory stimuli. [1]

[1] The author would like to acknowledge the help of Michael Jordan of the MIT Brain and Cognitive Sciences department during the early development of this work, Eric Scheirer of the MIT Media Lab for his insightful comments during the revision process and Barry Vercoe of the MIT Media Lab for providing continual support for this research.

References

[Åström and Wittenmark, 1984] Åström, K.J. & Wittenmark, B.W. *Computer Controlled Systems.* Englewood Cliffs, NJ: Prentice Hall.

[Anthony and Biggs, 1992] Anthony, M., & Biggs, N. "Computational Learning Theory", chapter 8. *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press.

[Duda and Hart, 1973] Duda, R.O, Hart, P.E. *Pattern Classification and Scence Analysis.* New York: Wiley

[Elman, 1990] Elman, J.L. "Finding Structure in Time." *Cognitive Science* 14, pp. 179-211.

[Grey, 1975] Grey, J.M. "An Exploration of Musical Timbre." Ph.D Dissertation, Department of Psychology, Stanford University.

[Haussler, 1989] Haussler, D. "Generalizing the PAC model: Sample size bounds from metric dimension-based uniform convergence results." In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science.* IEEE Computer Society Press.

[Jordan, 1990] Jordan, M.I. "Motor learning and the degrees of freedom problem." In M. Jeannerod (Ed.), *Attention and Performance, XIII.* Hillsdale, NJ: Erlbaum.

[Jordan and Rumelhart, 1992] Jordan, M.I. & Rumelhart, D.E. "Forward models: Supervised learning with a distal teacher", *Cognitive Science* (in press).

[Kohonen, 1989] Kohonen, T. *Self Organization and Associative Memory* (3rd Ed.). Berlin: Springer-Verlag

[Lee and Wessel, 1992] Lee, M., & Wessel, D., "Connectionist Models for Real-Time Control of Synthesis and Compositional Algorithms.", *Proceedings of the International Computer Music Conference.* San Fransisco: International Computer Music Association.

[McIntyre et al., 1983] McIntyre, M.E., Schumacher, R.T., and Woodhouse, J. "On the Oscillations of Musical Instruments." *Journal of the Acoustical Society of America*, 75:5, pp.1325-1345

[Minsky and Papert, 1969] Minsky, M.L., & Papert, S.A., *Perceptrons: An Introduction to Computational Geometry*, Cambridge: MIT

[Risset and Mathews, 1969] Risset, J.C. & Matthews, M.V. "Analysis of Musical Intrument Tones." *Physics Today* 22:2, pp.23-40

[Rumelhart et al., 1986] Rumelhart, D.E., Hinton, G.E., & Williams, R.J. "Learning Internal Representations by Error Propagation". In D.E.

Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, pp. 318-363, Cambridge, MA: MIT

[Smith, 1986] Smith, J.O. "Efficient Simulation of the Reed-Bore and Bow-String Mechanism."*Proceedings of the International Computer Music Conference*. San Fransisco: International Computer Music Association.

[Smith, 1992] Smith, J.O. "Physical Modeling Using Digital Waveguides." *Computer Music Journal* 16:4, pp. 74-87