On the Fidelity of Human Skill Models

Michael C. Nechyba and Yangsheng Xu

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Modeling dynamic human control strategy, or human skill, in response to real-time sensing is becoming an increasingly popular paradigm in many research areas. These models are learned from experimental data, and as such can be characterized despite the lack of a good physical model. Unfortunately, learned models presently offer few, if any, guarantees in terms of model fidelity to the source data. As such, we propose an independent, post-training model validation procedure based on Hidden Markov Models (HMMs). The proposed method generates a stochastic similarity measure comparing system trajectories for the source process and the learned models. Using this method, we are able to verify model fidelity. We demonstrate the proposed method in the validation of neural-network models for real-time human driving skill.

1. Introduction

Models of human skill, or human control strategy, which accurately emulate dynamic human behavior, have far reaching potential in areas ranging from robotics to virtual reality to the intelligent vehicle highway project. Significant challenges arise in the modeling of human skill, however. Defying analytic representation, little if anything is known about the structure, order or granularity of an individual's human controller. Human control strategy is both dynamic as well as stochastic in nature. In addition, the complex mapping from sensory inputs to control action outputs inherent in human control strategy can be highly nonlinear for given tasks. Therefore, developing an accurate and useful model for this type of dynamic phenomenon is frustrated by a poor understanding of the underlying basis for that phenomenon. Consequently, modeling by observation, rather than physical derivation, is becoming an increasingly popular paradigm for characterizing a wide range of complex processes, including human control strategy. This type of modeling is said to constitute learning, since the model is not derived from *a priori* laws of nature, but rather from observed instances of experimental data, known collectively as the training set.

The main strength of modeling by learning is that no explicit physical model is required; this also represents its biggest weakness, however. On the one hand, we are not restricted by the limitations of current scientific knowledge, and are able to model processes for which we have not yet developed adequate understanding. On the other hand, the lack of scientific justification detracts from the confidence that we can show in these learned models. This is especially true when the unmodeled process is (1) dynamic and/or (2) stochastic in nature, as is the case for human control strategy. For a dynamic process, model errors can feed back on themselves to produce trajectories which are not characteristic of the source process or are even potentially unstable. For a stochastic process, a static error criterion, based on the difference between the training data and predicted model outputs may be inadequate and inappropriate to gauge the fidelity of a learned model to the source process. Yet, most learning approaches today utilize some static error measure as a test of convergence for the learning algorithm. While this measure might be useful during training, it offers few, if any, guarantees about the dynamic behavior of the resulting learned model.

To counter these problems, we propose a post-training model-validation procedure, which characterizes the totality of the system trajectories generated by the learned model. Similar to methods developed in speech recognition [10], gesture recognition [11], and action learning [12], our approach centers around Hidden Markov Models (HMMs) as the primary tool for analyzing the multi-dimensional control trajectories generated by the human control strategy as well as the corresponding learned model. First, the source trajectories are characterized by training a validating HMM. Second, this HMM is cross-evaluated with the model-generated trajectories to arrive at a similarity measure between the demonstrated human control strategy and the learned model.

Although the proposed method generalizes to any learning paradigm and to the modeling of any dynamic process, we focus our attention in this paper on the learning of human control strategy through artificial neural networks. We first motivate the need for this kind of validation technique in observation-based modeling of unknown dynamic, stochastic processes. Second, we describe the proposed HMM-based validation method. Third, we discuss our approach for learning human control strategy models in the context of a driving simulator. Finally, we report results in validating the learned control strategy models.

2. Dynamic, stochastic model validation

2.1 Need for model validation

For most learning approaches, the modeling of an unknown process begins with the collection of experimental training data. This training data generally consists of static input/output vectors, where the inputs represent sensory data and the state of the system, and the outputs represent some desired control action. For dynamic systems where the state is not available or the order of the system is unknown, as in human control strategy modeling, the inputs are actually a time history of present and past sensory data as well as a time history of previous outputs. The learning process subsequently attempts to fit that training data to some general, adjustable model structure. Typically, the criterion for evaluating the quality of the model-in-training is some error measure, which is based on the difference between the training set outputs and the predicted model outputs. The root-mean-squared

(RMS) error is one such measure commonly used. While this static performance criterion can serve as a valuable gauge to evaluate model convergence during learning, it is significantly less instructive in how well the model matches the source process when actually employed in a dynamic feedback loop. That is, there are no guarantees, theoretical or otherwise, that a given RMS error over the training set will result in model trajectories similar to the source process. Even unstable trajectories can result.

For a simple illustration of this problem, consider the following example. Suppose that we wish to learn a dynamic process represented by the following simple difference equation,

$$y(k+1) = 0.75y(k) + 0.24y(k-1) + x(k) \qquad \text{(Eq. 1)}$$

where $y(k)$, $x(k)$ represent the output and input of the system, respectively, at time step k. The following input/output training data is provided,

Table 1: Input-output training data

Input			*Output*
$y(k-1)$	$y(k)$	$x(k)$	$y(k+1)$
-0.1	0.1	0.4	0.349
0.1	0.1	0.5	0.599
-0.3	0.2	0.3	0.123
0.3	0.2	0.4	0.673
0.2	0.0	0.5	0.650
0.0	0.2	0.3	0.348

Note that (Eq. 1) is asymptotically stable. Now, suppose, for example, that we train simple neural networks to learn three different approximations of the system in (Eq. 1):

$$\#1:\ y(k+1) = 0.76y(k) + 0.25y(k-1) + x(k) \qquad \text{(Eq. 2)}$$

$$\#2:\ y(k+1) = 0.76y(k) + 0.23y(k-1) + x(k) \qquad \text{(Eq. 3)}$$

$$\#3:\ y(k+1) = 0.74y(k) + 0.23y(k-1) + x(k) \qquad \text{(Eq. 4)}$$

The three models all have the same RMS error for the training set in Table 1. Nevertheless, the dynamic trajectories for the three models differ significantly. Consider the input,

$$x(k) = 0.1\sin\left(\frac{k\pi}{100}\right) \qquad \text{(Eq. 5)}$$

The resulting output for the system as well as the three models is shown in Figure 1. Model #2 approximates the system in (Eq. 1) well; model #3 remains stable, but approximates the system with significantly poorer accuracy; finally, model #1 diverges into an unstable trajectory.

The difference in the three models is the distribution of the error over the training set. Thus, a static error measure, such as RMS error, does not provide sufficiently satisfactory model validation for a dynamic process. Furthermore, for stochastic systems, one cannot expect equivalent trajectories for the process and the learned model, given the same initial conditions. Thus, we require a procedure, which examines the totality of the trajectories generated by the learned model and compares those to the trajectories of the unknown process. To this end, we are proposing the following HMM-based model-validation procedure.

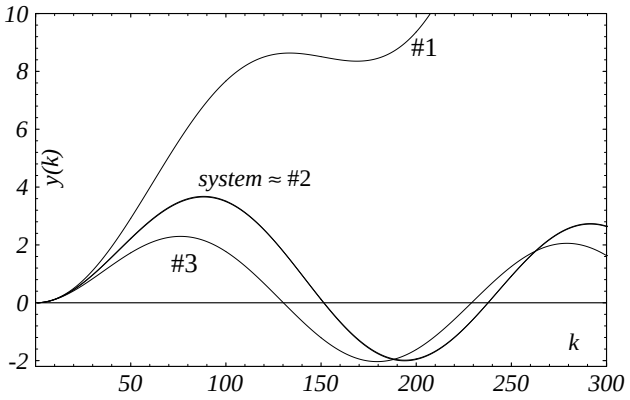

Fig. 1: The three models result in dramatically different (even unstable) trajectories.

2.2 HMM-based model validation

Rich in mathematical structure, HMMs are powerful parametric models which have been applied extensively in the area of stochastic signal processing. As such, HMMs can provide the proper framework for the validation analysis of dynamic, stochastic processes, such as those exhibited in human control strategy. In speech recognition, where HMMs have found their widest application, one-dimensional audio signals are analyzed as speech patterns [4][10]. Similarly, the multi-dimensional dynamic system trajectories in human control strategy can be analyzed as skill patterns using HMMs.

A Hidden Markov Model consists of a set of *n* states, interconnected through probabilistic transitions; each of these states has some output probability distribution associated with it. Although algorithms exist for HMMs with both discrete and continuous output probability distributions, and although many applications for HMMs deal with real-valued signals, discrete HMMs are overwhelmingly preferred to continuous HMMs in practice, due to their relative computational simplicity and lesser sensitivity to initial random parameter settings. Therefore, we choose to work exclusively with discrete output HMMs. Figure 2 below shows an example of a 5-state HMM, where each state emits one of 16 discrete symbols, based on some probability distribution.

Thus, a discrete HMM is completely defined by the following triplet: $\lambda = \{A, B, \pi\}$, where A represents the probabilistic $n \times n$ state transition matrix, B represents the $l \times n$ output probability matrix with l discrete output symbols, and π represents the n-length initial state probability distribution vector.

Consider, for the moment, a given system trajectory, be it from the source process (i.e. the human), or the learned model. This system trajectory defines a multi-dimensional trajectory of real-valued signals, including the state variables and control action outputs. In order to be useful for HMM analysis, this multi-dimensional flow of information must be converted to a sequence of discrete observation symbols [11] as shown in Figure 3. First, each dimension of the trajectory is normalized and converted to a sequence of vectors, consisting of short-time FFT power spectral coefficients. We use the Fast Fourier Transform (FFT) as it im-

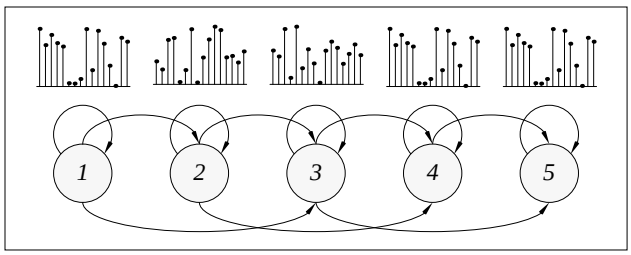

Fig. 2: A 5-state Hidden Markov Model (HMM), with 16 observable symbols in each state.

parts important dynamic information in characterizing the human's control strategy, and spectral conversion has been successfully used in other HMM-based applications. The short-time windows (k time-steps long) are overlapped by 50% and filtered through a Hamming window, so as to minimize the loss of information in the signal. For each short-time window, the individual spectral vectors are then joined into one long vector. This sequence of vectors is passed through a vector quantizer, which iteratively generates codebooks of size 2^m, $m \in \{0, 1, 2, \ldots\}$, stopping at an appropriate level of discretization [5] given the amount of available data and complexity of the trajectories. The original multi-dimensional, real-valued signal of length t has now been converted to a sequence of discrete symbols of length $T = \text{int}(2t/k)$.

Now, if this sequence of discrete symbols is generated from the source process (in our case, the human), it can be used to train an n-state HMM to maximize $P(\lambda|O)$ (i.e. the probability of the model, given the observation) using the well-known Baum-Welch Expectation-Maximization (EM) algorithm [1][10]. The resulting HMM can then be employed as a benchmark by which to judge the learned model of that individual's control strategy, by evaluating $P(O|\lambda)$ (i.e. the probability of the observation, given the model) for the learned models' trajectories. Let,

$$\bar{P}(O, T, \lambda) = 10^{\log P(O|\lambda)/T} \qquad \text{(Eq. 6)}$$

where O = observation sequence, T = length of O, and λ = HMM. $\bar{P}(O, T, \lambda)$ represents a probability measure normalized with respect to the length of the observation sequence. Now, define the similarity measure σ,

$$\sigma = \frac{\bar{P}(O^v, T^v, \lambda^*)}{\bar{P}(O^*, T^*, \lambda^*)} = \frac{P_2}{P_1}, \ \sigma > 0 \qquad \text{(Eq. 7)}$$

where λ^* = HMM trained on observation sequence O^* of length T^*, O^v = observation sequence of length T^v, which we wish to compare to O^* through λ^*. As shown in Figure 4 below, the measure σ stochastically compares the two observation sequences O^* and O^v through the trained HMM λ^*, and will generally range from 0 to 1. Larger values indicate greater probabilistic similarity, while smaller values (closer to zero) indicate dissimilarity.

If sensory or environmental inputs to the system outside the control of the system itself (e.g. road characteristics for driving) are part of the dynamic process, care must be taken that these inputs are similar for both the source data and the learned model when a comparison is made. For example, in modeling human driving, a sufficient equivalency criterion could be that the roads in both cases exhibit similar characteristics, such as the frequency

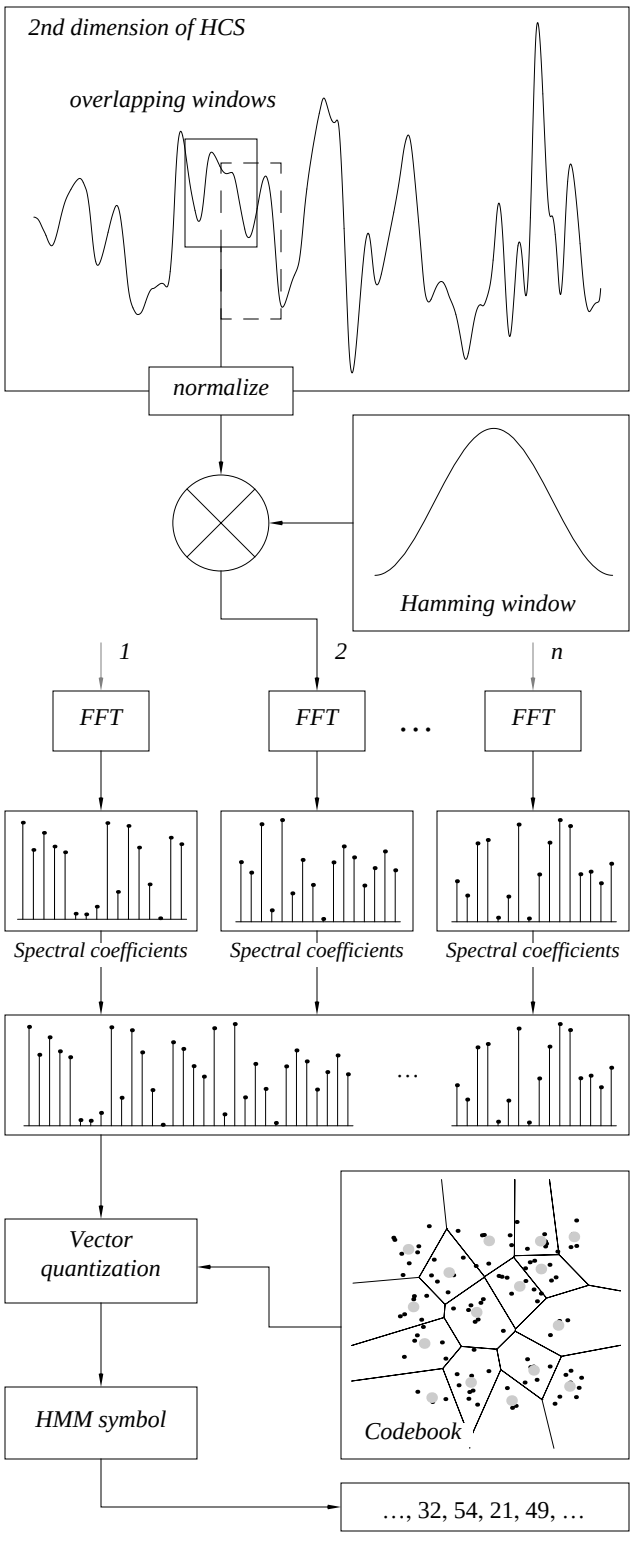

Fig. 3: To analyze human control strategy, the multi-dimensional control signal must be converted into a suitable form for HMM-based analysis. The above diagram illustrates the steps from real-valued signal to HMM symbol.

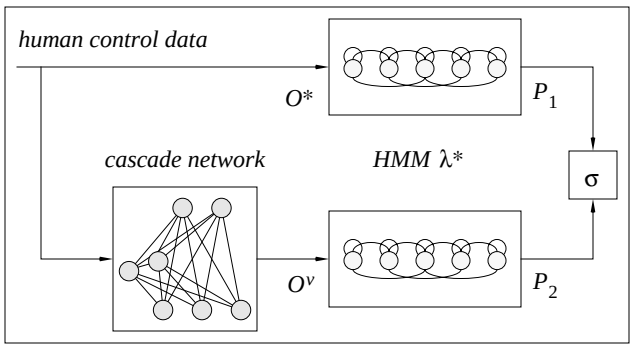

Fig. 4: The source process trajectory trains an HMM. Both the source process and the model output are then assigned separate probabilities to arrive at the similarity measure.

and sharpness of turns in the road. Alternatively, multiple HMMs can be trained for a single individual in differing environmental conditions.

3. Learning human control strategy

Below, we briefly review our general approach to abstracting human control strategy into computational models. We then narrow in on driving as one particular human control strategy. We describe our experimental driving simulator used to collect data from human subjects and to test learned models of driving strategy. Finally, we describe results for control strategy models from two different individuals, Oliver and Stan, which we use in the subsequent section to illustrate the model-validation procedure.

3.1 Cascade architecture

In learning human control strategy, we wish to approximate the functional mapping between sensory inputs and control action outputs which guide an individual's actions. Human control strategy is dynamic, stochastic, and often highly nonlinear in nature. Little, if anything, is known *a priori* about the underlying structure, order, or granularity of an individual's internal controller. Consequently, we require a flexible, nonlinear learning architecture, capable of generating a wide spectrum of mappings from smooth to discontinuous, linear to nonlinear. Cascade neural networks [2] with variable activation functions [6][7][9] offer such flexibility.

Cascade neural networks are feed-forward neural networks. In cascade learning, the structure of the network is adjusted as part of the training process by adding hidden units one at a time to an initially minimal network. Hidden units are added in a cascading fashion, with a new hidden unit taking input from all previous hidden units. In addition, each of these hidden units can assume a variable activation function, which is not restricted to simply the sigmoidal nonlinearity. This flexibility in functional form leads to efficiency in learning speed and good function approximation properties [9]. Thus, we use cascade neural networks with variable activation functions to learn models of human control strategy [7][8].

3.2 Experimental set-up

Figure 5 below illustrates the graphic driving simulator, which we used to model the control strategy of driving from human example. In the interface, the user has full control over the steering ($-0.2\text{rad} < \delta < 0.2\text{rad}$) of the car, as well as the brake ($-8000\text{N} < P < 0\text{N}$) and accelerator ($0\text{N} < P < 4000\text{N}$) controls. By moving the mouse in the horizontal direction, the user turns the steering wheel back and forth. Through the three mouse buttons, the user has independent control over the brake and steering. The left mouse button corresponds to pushing on the brake; the right mouse button corresponds to pushing on the accelerator; the middle mouse button corresponds to slowly easing off the current pedal being pushed. The vehicle dynamics are given in [3][7], and are simulated at 50Hz, where,

$$\dot{\theta} = \text{angular velocity of the car}, \qquad \text{(Eq. 8)}$$

$$v_\xi = \text{lateral velocity of the car}, \qquad \text{(Eq. 9)}$$

$$v_\eta = \text{longitudinal velocity of the car}. \qquad \text{(Eq. 10)}$$

The road to be navigated is defined by a random sequence of variable-length straight-line segments, and circular arcs of variable radius and sweep angle [7].

3.3 Human control strategy models

Here, we report modeling results for two different individuals, Oliver and Stan. The left sides of Figure 6 and Figure 7 show part of the driving data collected from Oliver and Stan, respectively. Note that the driving styles for the two individuals are quite different for the same road. Each person "drove" in the driving simulator for about 10 minutes, yielding approximately 30,000 data points, 10,000 of which are randomly selected for training the cascade network models.

The inputs to the cascade networks for each individual include (1) current and previous state information $\{v_\xi, v_\eta, \dot{\theta}\}$, (2) previous command information $\{\delta, P\}$, and a description of the road. For the road description, we discretize the visible view of the road ahead into 15 equivalently spaced, body-relative (x, y) coordinates of the road median, and provide the sequence of coordinates as input to the network. The outputs for the control strategy model are, of course, $\{\delta(k+1), P(k+1)\}$ (i.e. the steering and acceleration command for the next time step).

Since Stan's control strategy is relatively simple, his control strategy model requires 30 hidden units and only the previous two states as input. Oliver's more complicated control strategy model, on the other hand, requires 32 hidden units and relies on the previous ten states [7] to stay on the road. These input histories were determined experimentally to achieve stable road following for each control strategy model. The right sides of Figure 6 and Figure 7 show part of the model-generated command trajectories for Oliver and Stan, respectively.

4. Validating human control strategy models

4.1 Implementation issues

In using the validation procedure, several implementation choices are required. First, what HMM structure should be used for the validating HMM (λ^*)? Second, which dimensions of the signal trajectory should be included in the signal-to-symbol conversion? Third, what data is to be used to generate the vector quantization codebook? Finally, to how many levels should we quantize the spectral vectors?

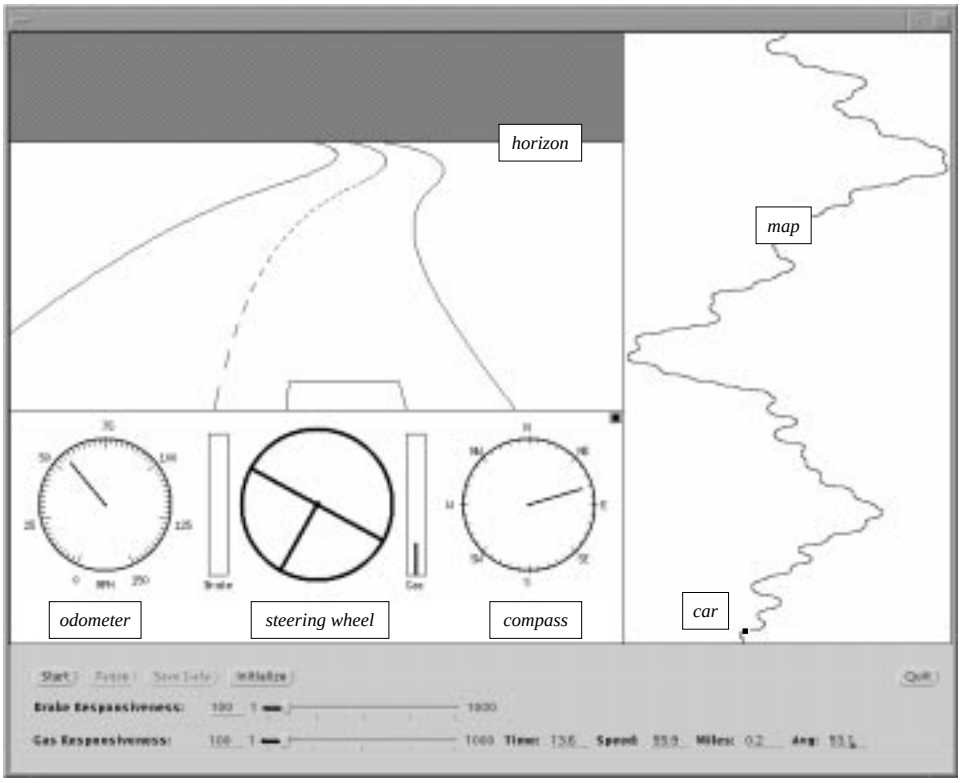

Fig. 5: The driving simulator gives the user a perspective preview of the road ahead. The user has independent controls of the steering, brake, and accelerator (gas).

First, the human control strategy signals we wish to compare are temporal signals. As such, we decide to use a five-state left-to-right model (as shown in Figure 2) for the HMM structure. Since this model only allows state transitions in one direction, we split the overall sequence of symbols into multiple, shorter subsequences, which then train the Hidden Markov Model using the Baum-Welch reestimation formulas for multiple observation sequences. Second, for the driving example, we include $\{\delta, P\}$, the steering and acceleration commands, in the multi-dimensional signal that we wish to characterize. The two commands capture the demonstrated human control strategy. Third, for the results reported below, we utilized a single vector quantization codebook, which is generated from Oliver and Stan's source data. This ensures that the results reported herein are not the consequence of biased preprocessing. Finally, the number of levels to quantize is obviously dependent on the amount of data available for training.

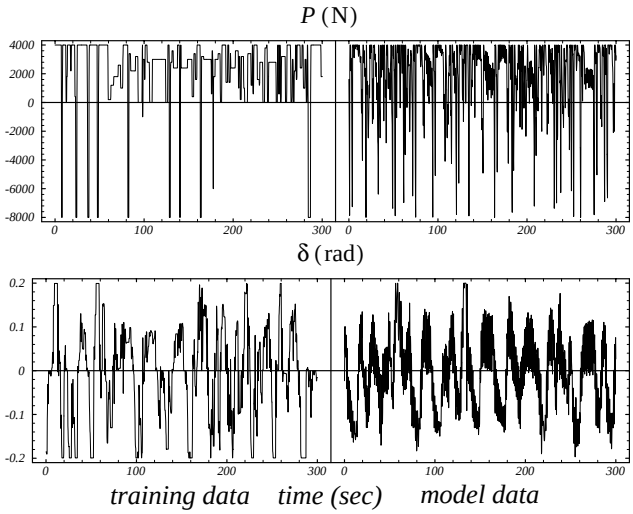

Fig. 6: Oliver's driving data. On the left, the source training data; on the right, the model-generated data.

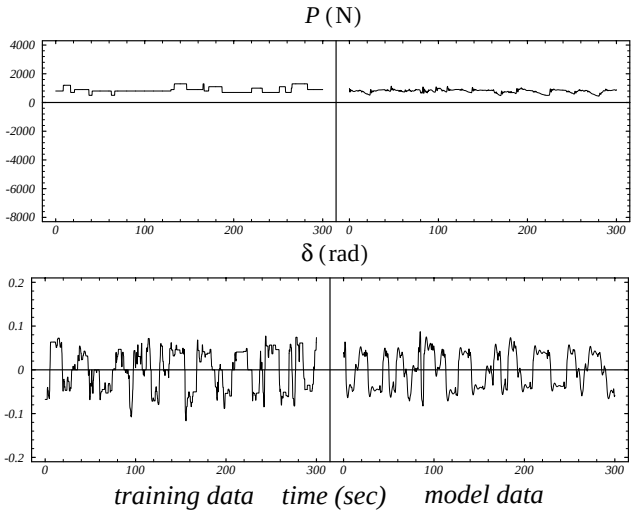

Fig. 7: Stan's driving data. On the left, the source training data; on the right, the model-generated data.

That is, the number of quantized levels directly affects model size, and therefore the number of nonzero parameters in the HMM. The number of nonzero parameters should be much less than the length of the observation sequence used in training. For our case, this restricts us to 64 levels.

4.2 Validation results

Table 2 below summarizes the similarity measure σ for some different scenarios. The top row indicates whose data is used to train the validating HMM. In Oliver's case, the HMM is trained on a sequence of 2900 observables, while in Stan's case, the HMM is trained on a sequence of 3710 observables.

Table 2: Similarity measure σ

	$\lambda^* = Oliver$	$\lambda^* = Stan$
Oliver	1.000	0.000
Oliver's Model	0.487	0.001
Stan	0.052	1.000
Stan's Model	0.022	0.911

5. Discussion

The similarity results in Table 2 confirm two qualitative assessments of the data in Figure 6 and Figure 7. First, we note that the two driving styles are objectively quite different. This fact is clearly reflected in the low similarity measures between one individual's HMM and the other individual's source or model-generated data. Second, Stan's model is a better reflection of his driving style, than Oliver's model is of his, as reflected in the two respective similarity measures, 0.911 and 0.487. This is indicative that Oliver's sharply discontinuous driving strategy is more difficult to learn by a cascade network than Stan's calmer approach. Indeed, Oliver's model generates significant oscillatory behavior, of which Oliver himself is not guilty.

A low similarity measure can be indicative of some problem in the model itself. In this case, perhaps too few or too many inputs are fed into the model. Or the inputs are not space sufficiently in time. Or more hidden units are called for. In a sense learning and validating a model are two tasks which feedback on each other and offer the potential for improvement. This is especially true in the area of human control strategy modeling, since little is known about the underlying structure of the human controller. The proposed procedure can aid in the selection of the best model architecture and most relevant input representation for a given human's control strategy. For the input representation, both the time scale and length of the time histories are variable. The best length can reveal the approximate order of a given person's control strategy, while the best time scale can reveal a given person's approximate reaction time.

When modeling human control strategy, the similarity measure can also serve a purpose besides model validation. It can be used to compare the control strategies of different individuals. For our example, it is qualitatively apparent that the driving strategies are quite different. Consequently, the similarity measure evaluates to very small values across individuals.

6. Conclusion

Model validation is an important problem in the area of machine learning for dynamic systems, if learned models are to be exploited for their full potential. We have described a stochastic model validation scheme based on Hidden Markov Models. The method introduces a similarity measure, by which we can verify model fidelity to source process trajectories. Such a measure is especially relevant in the learning and modeling of human control strategy, where little is known about the structure, order, or granularity of the underlying human controller. In fact, we have demonstrated the viability of the proposed method in the validation of human control strategy for the driving task, and have proposed further avenues of research along this direction.

Reference

[1] L. E. Baum, *et. al*., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains,", *Annals of Mathematical Statistics*, Vol. 41, No. 1, pp. 164-171, 1970.

[2] S. E. Fahlman, and C. Lebiere, "The Cascade-Correlation Learning Algorithm," Technical Report, CMU-CS-90-100, Carnegie Mellon University, 1990.

[3] H. Hatwal, and E. C. Mikulcik, "Some Inverse Solutions to an Automobile Path-Tracking Problem with Input Control of Steering and Brakes," *Vehicle System Dynamics*, Vol. 15, pp. 61-71, 1986.

[4] X. D. Huang, *et. al.*, "Hidden Markov Models for Speech Recognition," Edinburgh University Press, 1990.

[5] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communication*, Vol. COM-28, pp. 84-95, 1980.

[6] M. Nechyba, and Y. Xu, "Neural Network Approach to Control System Identification with Variable Activation Functions," *Proc. IEEE Int. Symp. on Intelligent Control*, Vol. 1, pp. 358-363, 1994.

[7] M. Nechyba, and Y. Xu, "Human Skill Modeling and Transfer: Neural Networks as Learners and Teachers," (submitted to *IEEE Trans. on Robotics and Automation*), 1995.

[8] M. Nechyba, and Y. Xu, "Human Skill Transfer: Neural Networks as Learners and Teachers,", *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, Vol. 3, pp. 314-319, 1995.

[9] M. Nechyba, and Y. Xu, "Towards Human Control Strategy Learning: Neural Network Approach with Variable Activation Functions," Technical Report, CMU-RI-TR-95-09, Carnegie Mellon University, 1995.

[10] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. of the IEEE*, Vol. 77, No. 2, pp. 257-286, 1989.

[11] J. Yang, Y. Xu, and C. S. Chen, "Gesture Interface: Modeling and Learning," *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 2, pp. 1747-52, 1994.

[12] J. Yang, Y. Xu, and C. S. Chen, "Hidden Markov Model Approach to Skill Learning and its Application to Telerobotics," *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 5, pp. 621-631, 1994.