STOCHASTIC SIMILARITY FOR VALIDATING HUMAN CONTROL STRATEGY MODELS

Michael C. Nechyba Yangsheng Xu

CMU-RI-TR-96-29

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

August, 1996

This material is based upon work partially supported under a National Science Foundation Graduate Research Fellowship, as well as a Department of Energy Doctoral Research Fellowship.

Table of Contents

List of Figures

List of Figures

List of Tables

Abstract

Modeling dynamic human control strategy (HCS), or human skill in response to real-time sensing is becoming an increasingly popular paradigm in many different research areas, such as intelligent vehicle systems, virtual reality, and space robotics. Such models are often 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 training data. This is especially true for dynamic reaction skills, where errors can feed back on themselves to generate state and command trajectories uncharacteristic of the source process. Thus, we propose a stochastic similarity measure — based on Hidden Markov Model analysis — capable of comparing and contrasting stochastic, dynamic, multi-dimensional trajectories. This similarity measure is the first step in validating a learned model's fidelity to its training data by comparing the model's dynamic trajectories in the feedback loop to the human's dynamic trajectories. In this paper, we first derive and demonstrate properties of the similarity measure for stochastic systems. We then apply the similarity measure to real-time human driving data by comparing different control strategies among different individuals. We show that the proposed similarity measure outperforms the more traditional Bayes classifier in correctly grouping driving data from the same individual. Finally, we illustrate how the similarity measure can be used in the validation of models which are learned from experimental data, and how we can connect model validation and model learning to iteratively improve our models of human control strategy.

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 (HCS). 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 HCS for which we have not yet developed adequate biological or psychological 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 (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 (such as RMS error), 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, including the cascade learning algorithm, utilize some static error measure as a test of convergence for the learning algorithm. While this measure is very useful during training, it offers no guarantees, theoretical or otherwise, about the dynamic behavior of the resulting learned model.

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: Sample 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. For example, 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

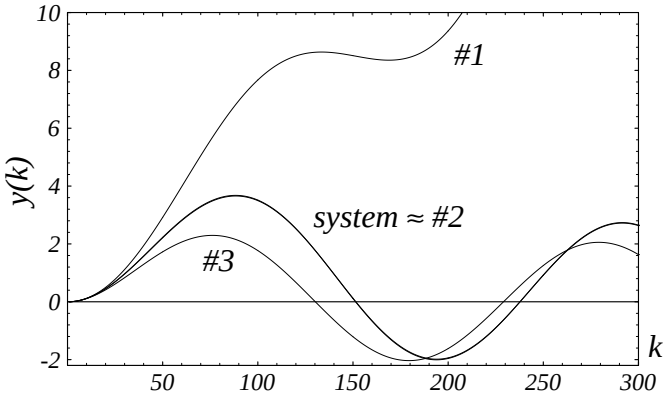

Figure 1: The three models result in dramatically different (even unstable) trajectories.

equivalent trajectories for the process and the learned model, given the same initial conditions. Thus, we require a stochastic similarity measure, with sufficient representational power and flexibility to compare multi-dimensional, stochastic trajectories. In this paper, we develop such a similarity measure, based on Hidden Markov Model (HMM) analysis, as a first step in validating models of dynamic human control strategy.

This paper is organized into three parts. First, using Hidden Markov Models, we derive a *stochastic similarity measure* capable of comparing aribtrary multi-dimensional, stochastic trajectories. This measure makes no *a priori* assumptions about the statistical distribution of the underlying data to be compared. We demonstrate certain properties of the proposed similarity measure through both mathematical proof and simulation of known stochastic systems. Second, we evaluate the similarity measure on human control data, by comparing driving strategies among different individuals. We show that the proposed similarity measure outperforms the more traditional Bayes classifier in correctly grouping driving data from the same individual. Finally, we illustrate how the similarity measure can be used in the validation of models which are learned from experimental data., and how we can connect model validation and model learning to iteratively improve our models of human control strategy.

2 Stochastic similarity

Similarity measures or metrics have been given considerable attention in computer vision [2, 4, 28], image database retrieval [11], and 2D or 3D shape analysis [14, 25]. These methods, however, generally rely on the special properties of images, and are therefore not appropriate for analyzing sequential trajectories. Other work has focussed on classifying temporal patterns using standard statistical techniques [5], wavelet analysis [27], neural networks [9, 24], and Hidden Markov Models (see discussion below). Much of this work, however,

analyzes only short-time trajectories or patterns, and, in many cases, generates only a binary classification, rather than a continuously valued similarity measure. Prior work has not addressed the problem of comparing long, multi-dimensional, stochastic trajectories, especially of human control data. Thus, we propose to evaluate *stochastic similarity* between two dynamic, multi-dimensional trajectories using *Hidden Markov Model (HMM)* analysis.

2.1 Hidden Markov Models

Rich in mathematical structure, HMMs are trainable statistical models, with two appealing features: (1) no *a priori* assumptions are made about the statistical distribution of the data to be analyzed, and (2) a high degree of sequential structure can be encoded by the Hidden Markov models. As such, they have been applied for a variety of stochastic signal processing. In speech recognition, where HMMs have found their widest application, human auditory signals are analyzed as speech patterns [10, 20]. Transient sonar signals are classified with HMMs for ocean surveillance in [13]. Radons, *et. al.* [22] analyze 30-electrode neuronal spike activity in a monkey's visual cortex with HMMs. Hannaford and Lee [7] classify task structure in teleoperation based on HMMs. In [29, 30], HMMs are used to characterize sequential images of human actions. Finally, Yang and Xu apply Hidden Markov Models to open-loop action skill learning [32] and human gesture recognition [31].

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 training HMMs with both discrete and continuous output probability distributions, and although most applications of HMMs deal with real-valued signals, discrete HMMs are preferred to continuous HMMs in practice, due to their relative computational simplicity and lesser sensitivity to initial random parameter settings [21]. Using discrete HMMs for analysis of real-valued signals requires that data be converted to discrete symbols through pre-processing and vector quantization (see Section 2.4 below). Thus, a discrete HMM is completely defined by the following triplet [20],

$$\lambda = \{A, B, \pi\} \qquad \text{(Eq. 6)}$$

where A is the probabilistic $n \times n$ state transition matrix, B is the $L \times n$ output probability matrix with L discrete output symbols $\chi \in \{1, 2, \ldots, L\}$, and π is the n-length initial state probability distribution vector for the HMM. Figure 2, for example, represents a 5-state HMM, where each state emits one of 16 discrete symbols, based on some probability distribution.

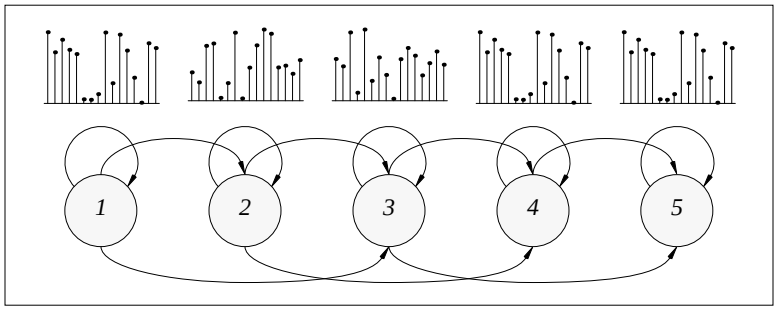

Figure 2: A 5-state Hidden Markov Model, with 16 observable symbols in each state.

For an observation sequence O of discrete symbols, we can locally maximize $P(\lambda|O)$ (i.e. the probability of the model λ given the observation sequence O) off-line using the Baum-Welch Expectation-Maximization (EM) algorithm [3, 20], or on-line, as presented in [1] and [12]. We can also evaluate $P(O|\lambda)$ (i.e. the probability that a given observation sequence O is generated from the model λ).

Finally. two HMMs λ_1 and λ_2 are defined to be *equivalent*,

$$\lambda_1 \sim \lambda_2, \text{ iff. } P(O|\lambda_1) = P(O|\lambda_2), \forall O \qquad \text{(Eq. 7)}$$

Note that λ_1 and λ_2 need not be identical to be equivalent. The following two HMMs are, for example, equivalent:

$$\lambda_1 = \left\{ \begin{bmatrix} 1 \end{bmatrix}, \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T, \begin{bmatrix} 1 \end{bmatrix} \right\}, \lambda_2 = \left\{ \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T \right\} \qquad \text{(Eq. 8)}$$

2.2 Similarity measure

Below, we derive a stochastic similarity measure, based on discrete-output HMMs. Let O_i, $i \in \{1, 2, \ldots\}$, denote a distinct observation sequence of discrete symbols with length T_i. Also let $\lambda_j = \{A_j, B_j, \pi_j\}$, $j \in \{1, 2, \ldots\}$, denote a discrete HMM locally optimized (using the Baum-Welch algorithm) to maximize $P(\lambda_j|O_j)$. Similarly let $P(O_i|\lambda_j)$ denote the probability of the observation sequence O_i given the model λ_j, and let,

$$P_{ij} = \hat{P}(O_i|\lambda_j) = P(O_i|\lambda_j)^{\frac{1}{T_i}} \qquad \text{(Eq. 9)}$$

denote the probability of the observation sequence O_i given the model λ_j, normalized with respect to T_i. In practice, we calculate P_{ij} as,

$$P_{ij} = 10^{\log P(O_i|\lambda_j)/T_i} \quad \text{(Eq. 10)}$$

to avoid problems of numerical underflow for long observation sequences.

Using the definition in (Eq. 9), Figure 3 illustrates our overall approach to evaluating similarity between two observation sequences. Each observation sequence is first used to train a corresponding HMM; this allows us to evaluate P_{11} and P_{22}. Furthermore, we subsequently cross-evaluate each observation sequence on the other HMM (i.e. $P(O_1|\lambda_2)$, $P(O_2|\lambda_1)$) to arrive at P_{12} and P_{21}. Given, these four normalized probability values, we now propose the following similarity measure between O_1 and O_2:

$$\sigma(O_1, O_2) = \sqrt{\frac{P_{21}P_{12}}{P_{11}P_{22}}} \quad \text{(Eq. 11)}$$

This measure takes the ratio of the cross probabilities over the training probabilities, and normalizes for the multiplication of the two probability values in the numerator and denominator by taking the square root. We note that in practice, we calculate the P_{ij} not on λ_j itself, but rather $\hat{\lambda}_j$, which is a smoothed version of λ_j, where zero elements in the matrices $\{A_j, B_j, \pi_j\}$ are replaced by $\varepsilon > 0$ and renormalized to fit probabilistic constraints. Throughout this paper, we use $\varepsilon = 0.0001$ as our smoothing value.

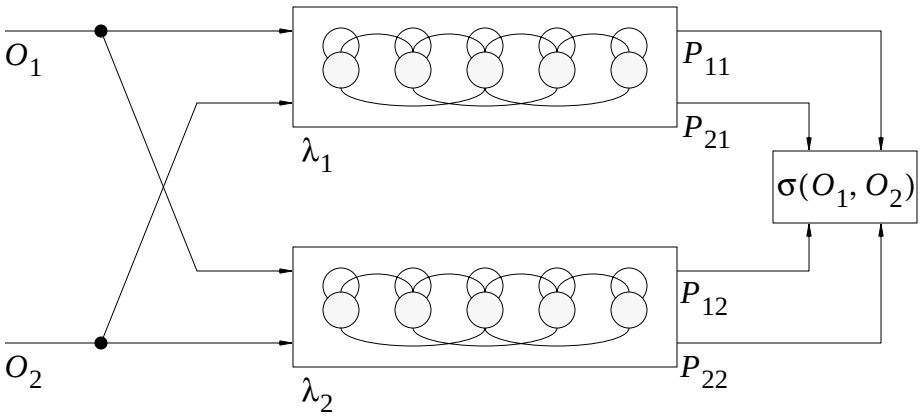

Figure 3: Four normalized probability values make up the similarity measure.

2.3 Properties

For now, we assume that P_{ii} is a global (rather than just a local) maximum. Then,

$$1/L \leq P_{ii}, \text{ and} \quad \text{(Eq. 12)}$$

$$0 < P_{ij} \leq P_{ii},\ \varepsilon > 0. \quad \text{(Eq. 13)}$$

The lower bound for P_{ii} in (Eq. 12) is realized for single-state discrete HMMs, and a uniform distribution of symbols in O_i. From (Eq. 11) to (Eq. 13), we can establish the following properties for $\sigma(O_1, O_2)$:[1]

$$\text{Property \#1: } \sigma(O_1, O_2) = \sigma(O_2, O_1) \text{ (by definition)} \quad \text{(Eq. 14)}$$

$$\text{Property \#2: } 0 < \sigma(O_1, O_2) \leq 1 \quad \text{(Eq. 15)}$$

$$\text{Property \#3: } \sigma(O_1, O_2) = 1 \text{ iff. } \lambda_1 \sim \lambda_2 \quad \text{(Eq. 16)}$$

When, P_{ii} is only a local maximum (as guaranteed by Baum-Welch), properties #2 and #3 are only approximately correct. This is not of significant concern, however, as the Baum-Welch algorithm converges to near-optimal solutions in practice [20].

For the class of single-state, discrete HMMs,

$$\lambda_j = \{A_j, B_j, \pi_j\} = \left\{ \left[1\right], \left[b_{j1} \ \ldots \ b_{jL}\right]^T, \left[1\right] \right\}, \quad \text{(Eq. 17)}$$

which encode only the distribution of symbols without capturing any of the sequential properties of observation sequence O_j, properties #2 and #3 are easy to show. The normalized probability values are given by,

$$P_{ij} = \hat{P}(O_i | \lambda_j) = \left(\prod_{k=1}^{L} (b_{jk})^{(T_i b_{ik})} \right)^{\frac{1}{T_i}} = \prod_{k=1}^{L} (b_{jk})^{b_{ik}} \quad \text{(Eq. 18)}$$

and the resulting similarity measure reduces to,

1. In [18], we proposed a different similarity measure for which properties #1, #2, and #3 do not hold. Thus, the previous similarity measure gives potentially inconsistent and misleading results for certain data. The similarity measure in (Eq. 11) corrects these problems.

$$\sigma(O_1, O_2) = \sqrt{\frac{P_{21}P_{12}}{P_{11}P_{22}}} = \left[\frac{\left(\prod_{k=1}^{L}(b_{1k})^{b_{2k}}\right)\left(\prod_{k=1}^{L}(b_{2k})^{b_{1k}}\right)}{\left(\prod_{k=1}^{L}(b_{1k})^{b_{1k}}\right)\left(\prod_{k=1}^{L}(b_{2k})^{b_{2k}}\right)}\right]^{\frac{1}{2}} \quad \text{(Eq. 19)}$$

$$\sigma(O_1, O_2) = \prod_{k=1}^{L}\left(\frac{b_{1k}}{b_{2k}}\right)^{\frac{b_{2k}}{2}} \prod_{k=1}^{L}\left(\frac{b_{2k}}{b_{1k}}\right)^{\frac{b_{1k}}{2}} \quad \text{(Eq. 20)}$$

$$\sigma(O_1, O_2) = \prod_{k=1}^{L}\left(\frac{b_{1k}}{b_{2k}}\right)^{\frac{(b_{2k}-b_{1k})}{2}} \quad \text{(Eq. 21)}$$

(Eq. 21) is a product of terms of the following form,

$$\xi = \left(\frac{x}{y}\right)^{\frac{(y-x)}{2}}, \; 0 < x, y < 1 \quad \text{(Eq. 22)}$$

Hence, in order to maximize $\sigma(O_1, O_2)$, we must maximize each of the multiplicands in the product of (Eq. 21) such that,

$$\left.\begin{aligned}\frac{\partial\xi}{\partial x} &= \frac{1}{2}\xi\left(\frac{y-x}{x} - \ln\frac{x}{y}\right) = 0\\ \frac{\partial\xi}{\partial y} &= \frac{1}{2}\xi\left(\frac{x-y}{y} + \ln\frac{x}{y}\right) = 0\end{aligned}\right\} \qquad \left.\frac{y-x}{x} + \frac{x-y}{y} = 0\right\} \qquad x = y \quad \text{(Eq. 23)}$$

$$\xi\big|_{x=y} = 1 \quad \text{(Eq. 24)}$$

In other words, $\sigma(O_1, O_2)$ reaches a maximum when $b_{1k} = b_{2k}$, $\forall k \in \{1, 2, \ldots, L\}$, or simply, $B_1 = B_2$ (Eq. 23), and that maximum is equal to one (Eq. 24). As an example, consider the case where,

$$B_1 = \begin{bmatrix} p_1 & 1-p_1 \end{bmatrix}^T \qquad B_2 = \begin{bmatrix} p_2 & 1-p_2 \end{bmatrix}^T \quad \text{(Eq. 25)}$$

For this special case, equation (Eq. 21) reduces to,

$$\sigma(O_1, O_2) = \left[\left(\frac{p_1}{p_2}\right)\left(\frac{1-p_2}{1-p_1}\right)\right]^{\frac{(p_2-p_1)}{2}} \qquad \text{(Eq. 26)}$$

which is graphed in Figure 4 as a contour plot for $0 < p_1, p_2 < 1$.

Finally, we show one example of how the proposed similarity measure changes, not as a function of different symbol distributions, but rather as a function of varying HMM structure. Consider the following Hidden Markov Model,

$$\lambda(\alpha) = \left\{ \begin{bmatrix} \frac{1+\alpha}{2} & \frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & \frac{1+\alpha}{2} \end{bmatrix}, \begin{bmatrix} \frac{1+\alpha}{2} & \frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & \frac{1+\alpha}{2} \end{bmatrix}, \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T \right\}, -1 < \alpha < 1 \qquad \text{(Eq. 27)}$$

and corresponding observation sequences, $O(\alpha)$, stochastically generated from model $\lambda(\alpha)$. For all $\alpha \in (-1, 1)$, $O(\alpha)$ will have an equivalent aggregate distribution of symbols 0 and 1 — namely 1/2 and 1/2. As $|\alpha|$ increases, however, $O(\alpha)$ will become increasingly structured. For example,

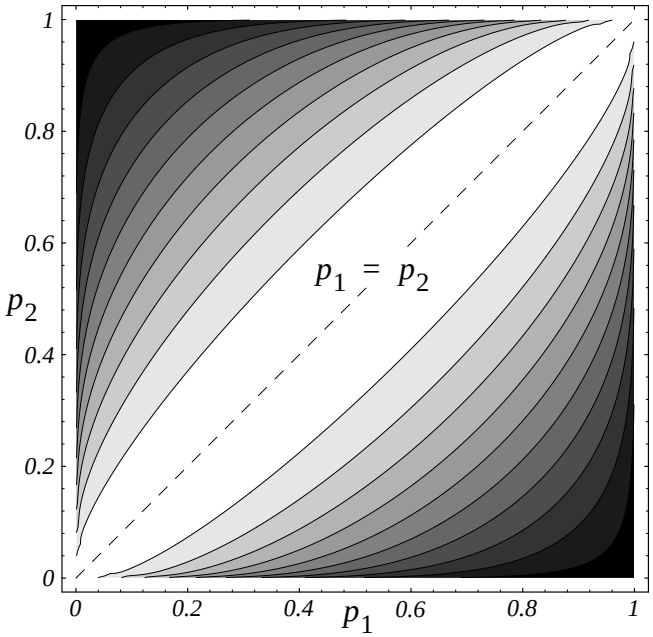

Figure 4: Similarity measure for two binomial distributions. Lighter colors indicate higher similarity.

$$\lim_{\alpha \to -1} O(\alpha) = \{\ldots, 1, 0, 1, 0, 1, 0, 1, 0, \ldots\} \quad \text{(Eq. 28)}$$

$$\lim_{\alpha \to 0} \lambda(\alpha) = \left\{ \left[1\right], \left[0.5 \;\; 0.5\right]^T, \left[1\right] \right\} \text{ (equivalent to unbiased coin toss)} \quad \text{(Eq. 29)}$$

$$\lim_{\alpha \to 1} O(\alpha) = \{\ldots, 1, 1, 1, 0, 0, 0, \ldots, 0, 0, 0, 1, 1, 1, \ldots\} \quad \text{(Eq. 30)}$$

Figure 5 graphs $\sigma[O(\alpha_1), O(\alpha_2)]$ as a contour plot for $-1 < \alpha_1, \alpha_2 < 1$, where each observation sequence $O(\alpha)$ of length $T = 10,000$ is generated stochastically from the corresponding HMM $\lambda(\alpha)$. Greatest similarity is indicated for $\alpha_1 = \alpha_2$, and around the origin, $(\alpha_1 = 0, \alpha_2 = 0)$, while greatest dissimilarity occurs for $(\alpha_1 \to 1, \alpha_2 = -1)$, and $(\alpha_1 \to -1, \alpha_2 = 1)$.

In some cases, it may be more convenient to represent the similarity between two trajectories with a *distance measure* $d(O_1, O_2)$, rather than a probability-based measure. Such a conversion is straightforward, given the similarity measure $\sigma(O_1, O_2)$. Let,

$$d(O_1, O_2) = -\log\sigma(O_1, O_2) = \frac{1}{2}[\log(P_{11}P_{22}) - \log(P_{21}P_{12})] \quad \text{(Eq. 31)}$$

such that,

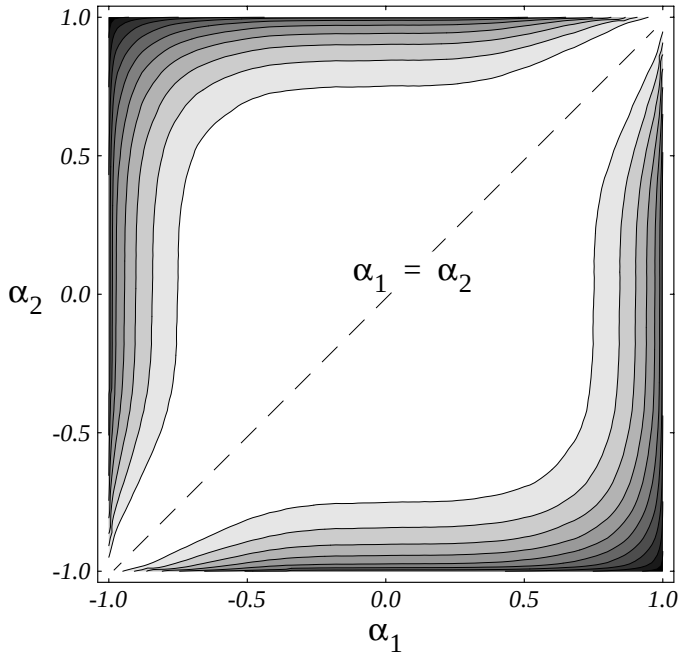

Figure 5: The similarity measure changes predictably as a function of HMM structure.

$$d(O_1, O_2) = d(O_2, O_1) \quad \text{(Eq. 32)}$$

$$d(O_1, O_2) \geq 0\,. \quad \text{(Eq. 33)}$$

$$d(O_1, O_2) = 0 \text{ iff. } \lambda_1 \sim \lambda_2\,. \quad \text{(Eq. 34)}$$

Equations (Eq. 32), (Eq. 33), and (Eq. 34) follow directly from (Eq. 14), (Eq. 15), and (Eq. 16), respectively.

2.4 Signal-to-symbol conversion

Multi-dimensional, real-valued human control data must be converted to a sequence of discrete symbols in order to apply the similarity measure defined in (Eq. 11). Such conversion generally involves two steps: (1) spectral preprocessing and (2) vector quantization, as illustrated in Figure 6. The primary purpose of the spectral preprocessing is to extract meaningful feature vectors for the vector quantizer. In this work, we rely on the fast Fourier transform (FFT) and the fast Walsh transform (FWT), the $O(n\log n)$ algorithmic counterparts of the discrete Fourier transform (DFT) and the discrete Walsh transform (DWT), respectively. Instead of sinusoidal basis functions, the Walsh transform decomposes a signal based on the orthonormal *Walsh functions* [23]. The first eight Walsh-ordered Walsh functions are shown in Figure 7(a). In Figure 7(b), we show an example of human control data (see Section 3.1 below) which can be characterized better through the Walsh transform, rather than the Fourier transform, due to its sharply discontinuous, step-like profile. Consider, for example, the power spectral densities (PSD's) for the square wave in Figure 8(a). The Walsh PSD in Figure 8(b) is a more concise feature vector than the corresponding Fourier PSD in Figure 8(c).

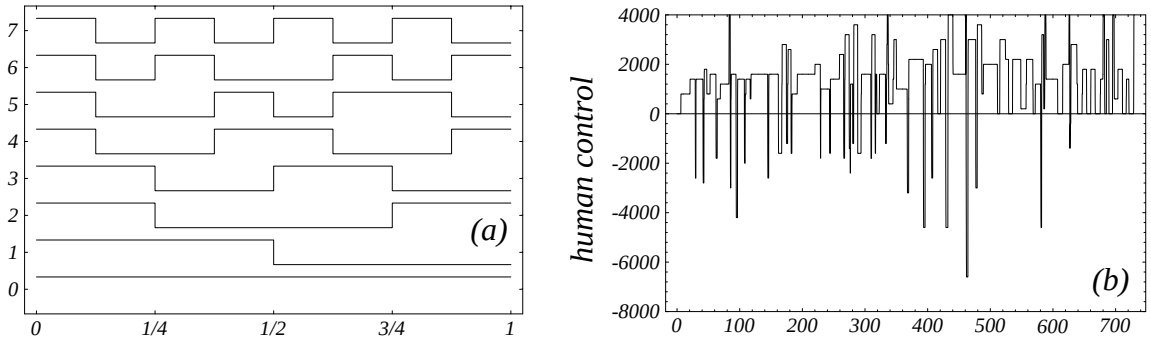

Figure 7: (a) The first eight Walsh-ordered Walsh functions, and (b) some sample human control data.

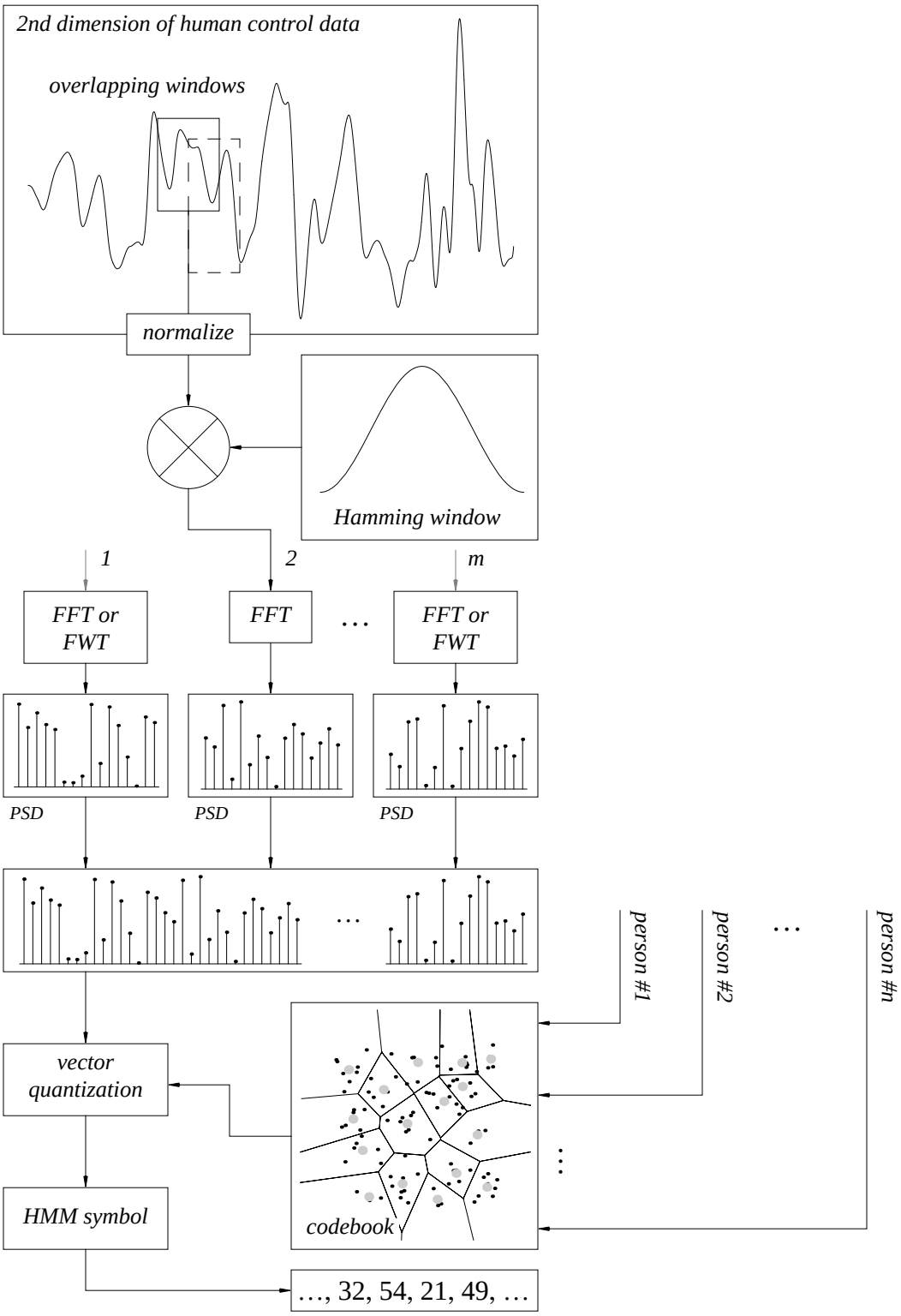

Figure 6: Conversion of multi-dimensional human control data to a sequence of discrete symbols.

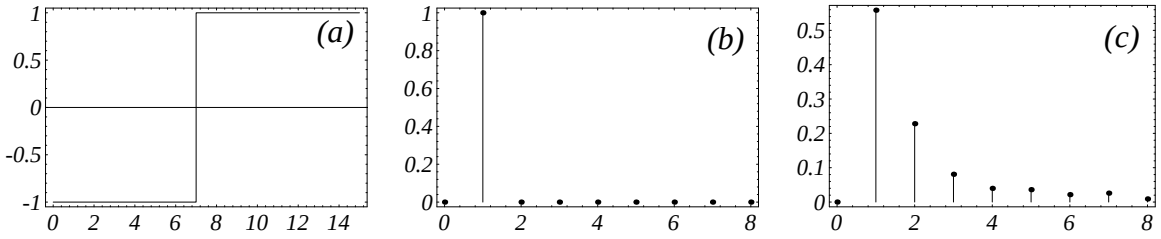

Figure 8: (a) Sample square wave and it's corresponding (b) Walsh and (c) Fourier PSD's.

For each dimension of the human control data, we partition the data into overlapping window frames, and perform either a short-time FFT or FWT on each frame. Generally, we select the FFT for *state trajectories*, and the FWT for *command trajectories*, since these trajectories tend to have sharp discontinuities for the experimental data in this paper. In the case of the FFT, the data in each frame is filtered through a Hamming window before applying the FFT, so as to compensate for the windowing effect. The spectral coefficients are then converted to power spectral density (PSD) vectors. In preparation for the vector quantization, the PSD vectors along each dimension of the system trajectory are normalized and concatenated into one long feature vector per frame. We quantize the resulting sequence of long feature vectors using the iterative LBG VQ algorithm [15]. This vector quantizer generates codebooks of size 2^m, $m \in \{0, 1, 2, \ldots\}$, and can be stopped at an appropriate level of discretization given the amount of available data and complexity of the system trajectories. Assuming that we segment the data into window frames of length k with 50% overlap, the original multi-dimensional, real-valued signal of length t is thus converted to a sequence of discrete symbols of length $T = \text{int}(2t/k)$.

3 Comparing human control strategies

3.1 Experimental set-up

Figure 9 shows the real-time graphic driving simulator, from which we collect human control data. In the interface, the human operator has full control over the steering of the car (mouse movement), the brake (left mouse button) and the accelerator (right mouse button); the middle mouse button corresponds to slowly easing off whichever pedal is currently being "pushed." The vehicle dynamics are given in (Eq. 35) through (Eq. 54) below (modified from [8]):

$$\ddot{\theta} = (l_f P_f \delta + l_f F_{\xi f} - l_r F_{\xi r})/I \qquad \text{(Eq. 35)}$$

$$\dot{v}_\xi = (P_f \delta + F_{\xi f} + F_{\xi r})/m - v_\eta \dot{\theta} - (\operatorname{sgn} v_\xi) c_D v_\xi^2 \quad \text{(Eq. 36)}$$

$$\dot{v}_\eta = (P_f + P_r - F_{\xi f}\delta)/m + v_\xi \dot{\theta} - (\operatorname{sgn} v_\eta) c_D v_\eta^2 \quad \text{(Eq. 37)}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_\xi \\ v_\eta \end{bmatrix} \quad \text{(Eq. 38)}$$

where,

$\dot{\theta}$ = angular velocity of the car (Eq. 39)

v_ξ = lateral velocity of the car, v_η = longitudinal velocity of the car (Eq. 40)

$$F_{\xi k} = \mu F_{zk}(\tilde{\alpha}_k - (\operatorname{sgn}\delta)\tilde{\alpha}_k^2/3 + \tilde{\alpha}_k^3/27)\sqrt{1 - P_k^2/(\mu F_{zk})^2 + P_k^2/c_k^2}, \quad \text{(Eq. 41)}$$

$$\tilde{\alpha}_k = c_k \alpha_k/(\mu F_{zk}),\ k \in \{f, r\} \quad \text{(Eq. 42)}$$

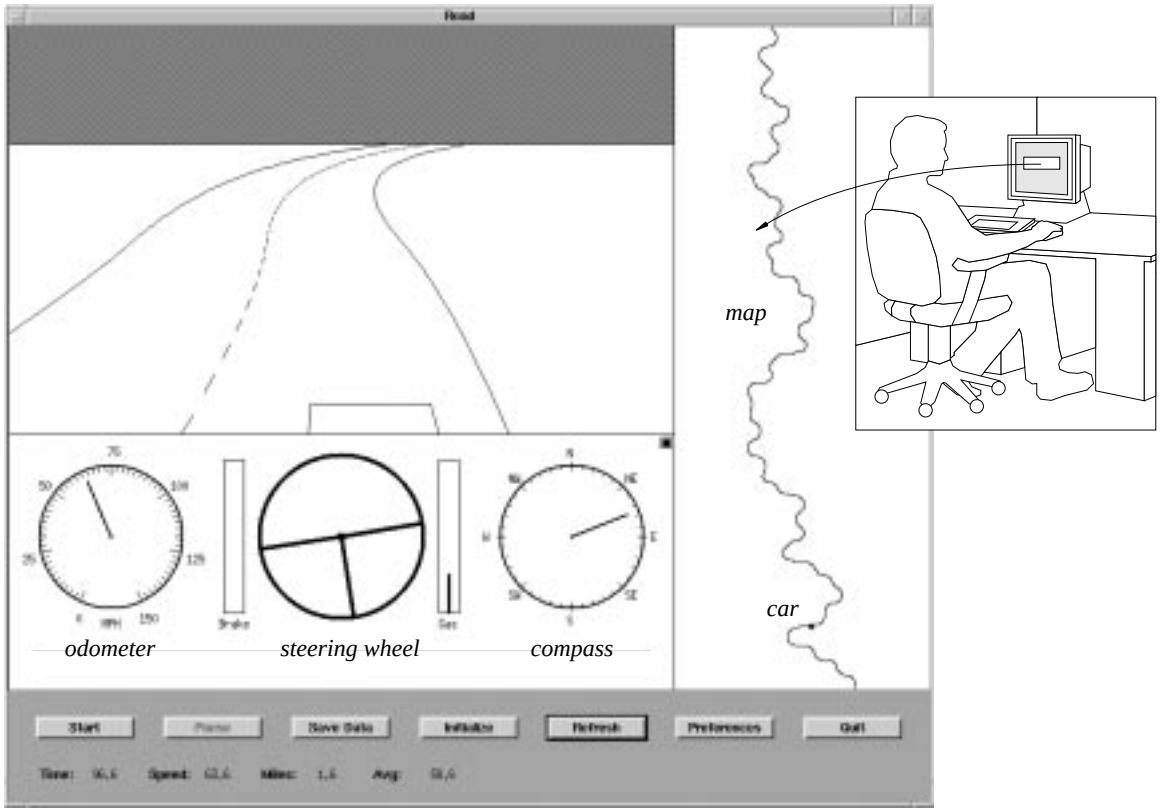

Figure 9: The driving simulator generates a perspective view of the road for the user, who has independent control over steering, braking, and acceleration (gas).

$$\alpha_f = \text{front tire slip angle} = \delta - (l_f\dot{\theta} + v_\xi)/v_\eta, \quad \text{(Eq. 43)}$$

$$\alpha_r = \text{rear tire slip angle} = (l_r\dot{\theta} - v_\xi)/v_\eta, \quad \text{(Eq. 44)}$$

$$F_{zf} = (mgl_r - (P_f + P_r)h)/(l_f + l_r), \quad \text{(Eq. 45)}$$

$$F_{zf} = (mgl_f + (P_f + P_r)h)/(l_f + l_r) \quad \text{(Eq. 46)}$$

$$\xi = \text{body-relative lateral axis}, \ \eta = \text{body-relative longitudinal axis} \quad \text{(Eq. 47)}$$

$$c_f, c_r = \text{cornering stiffness of front, rear tires} = 50000\text{N/rad}, 64000\ \text{N/rad} \quad \text{(Eq. 48)}$$

$$c_D = \text{lumped coefficient of drag (air resistance)} = 0.0005\text{m}^{-1} \quad \text{(Eq. 49)}$$

$$\mu = \text{coefficient of friction} = 1 F_{jk} = \text{frictional forces} j \in \{\xi, z\} k \in \{f, r\} \quad \text{(Eq. 50)}$$

$$P_r = \text{longitudinal force on rear tires} = \begin{cases} 0, & P_f \geq 0 \\ k_b P_f, & P_f < 0, k_b = 0.34 \end{cases} \quad \text{(Eq. 51)}$$

$$m = 1500\text{kg}, I = 2500\text{kg-m}^2, l_f = 1.25\text{m}, l_r = 1.5\text{m}, h = 0.5\text{m}, \quad \text{(Eq. 52)}$$

and the controls are given by,

$$-8000\text{N} \leq (P_f = \text{longitudinal force on front tires}) \leq 4000\text{N} \quad \text{(Eq. 53)}$$

$$-0.2\text{rad} \leq (\delta = \text{steering angle}) \leq 0.2\text{rad} \quad \text{(Eq. 54)}$$

Note that the separate brake and gas commands for the human are, in fact, the single P_f variable, where the sign indicates whether the brake or the gas is active (Figure 7(b), for example, illustrates one person's P_f profile for part of one run). The entire simulator is run at 50 Hz.

3.2 Similarity results

For the first set of experiments, we ask five people — (1) Larry, (2) Curly, (3) Moe, (4) Groucho, and (5) Harpo — to practice driving in the simulator for a period of up to fifteen minutes to become accustomed to the simulator's dynamics. We then record driving data for each person on three different, randomly generated 20km roads, show in Figure 10(a), (b), and (c), with only short breaks in between each run. Thus, we have a total of 15 runs. For notational convenience, let,

Figure 10: (a) Road 1, (b) Road 2, (c) Road 3.

$$R_{ij} = R(i, j),\ i \in \{1, 2, 3, 4, 5\},\ j \in \{1, 2, 3\} \quad \text{(Eq. 55)}$$

denote the run from person (*i*) on road #*j*. Table 2 below reports some aggregate statistics for each of the 15 runs.

Table 2: Aggregate statistics for human driving data

	Run	*v (mph)*	$\dot{\theta}$ *(rad/s)*	δ *(rad)*	P_f *(N)*
Larry	*R(1, 1)*	63.1 ± 12.2	0.00 ± 0.17	0.00 ± 0.06	1383 ± 2430
	R(1, 2)	62.7 ± 9.5	0.00 ± 0.17	0.00 ± 0.06	1309 ± 1852
	R(1, 3)	64.0 ± 8.6	0.00 ± 0.18	0.00 ± 0.06	1285 ± 1367
Curly	*R(2, 1)*	70.8 ± 8.3	0.00 ± 0.20	0.00 ± 0.07	1901 ± 3260
	R(2, 2)	69.1 ± 7.7	0.00 ± 0.19	0.00 ± 0.07	1849 ± 3342
	R(2, 3)	71.5 ± 7.7	0.00 ± 0.20	0.00 ± 0.08	1966 ± 3144
Moe	*R(3, 1)*	73.1 ± 9.5	0.00 ± 0.24	0.00 ± 0.09	2189 ± 2772
	R(3, 2)	71.9 ± 9.0	0.00 ± 0.25	0.00 ± 0.09	2238 ± 2622
	R(3, 3)	74.5 ± 9.4	0.00 ± 0.29	0.00 ± 0.11	2573 ± 2648
Groucho	*R(4, 1)*	66.8 ± 12.4	0.00 ± 0.18	0.00 ± 0.08	1851 ± 3826
	R(4, 2)	65.1 ± 13.2	0.00 ± 0.21	0.00 ± 0.09	1937 ± 3985
	R(4, 3)	69.8 ± 12.3	0.00 ± 0.23	0.00 ± 0.11	2293 ± 3755
Harpo	*R(5, 1)*	52.3 ± 12.2	0.00 ± 0.17	0.00 ± 0.05	894 ± 1480
	R(5, 2)	51.7 ± 4.2	0.00 ± 0.16	0.00 ± 0.04	702 ± 250
	R(5, 3)	56.1 ± 5.7	0.00 ± 0.20	0.00 ± 0.06	1014 ± 335

Our goal here is to see (1) how well the similarity measure classifies each individual's runs across different roads, and (2) how the classification performance of the proposed similarity measure compares with a more conventional statistical technique, namely the Bayes optimal classifier.

The system trajectory for the driving task is defined by the three state variables $\{v_\xi, v_\eta, \theta\}$ and the two control variables $\{\delta, P_f\}$. For all the similarity results below (unless otherwise noted), we choose the following spectral preprocessing,

$$\{v_\xi, v_\eta, \dot{\theta}, \delta, P_f\} = \{\text{F-16, F-16, F-16, W-16, W-16}\} \quad \text{(Eq. 56)}$$

where F-*k*, $k \geq 1$, denotes the *k*-point FFT, and W-*k*, $k \geq 1$, denotes the *k*-point Walsh transform, with 50% overlapping window frames. Note that the PSD vector for a *k*-point transform has length (*k*/2+1); thus, the feature vector to be quantized is of length 45. Also, note that we choose the Walsh transform for the two control variables $\{\delta, P_f\}$, since the user's control is generally discontinuous and step-like in profile (as shown in Figure 7(b) above).

From the preprocessed data, we build three vector codebooks Q_k, $k \in \{1, 2, 3\}$, each with 128 levels, and corresponding to data from R_{ik}, $\forall i \in \{1, 2, 3, 4, 5\}$. For these experiments, the number of levels in the VQ codebook is primarily constrained by the amount of available data to train the HMMs, since we want,

$$(T = \text{length of each observation sequence}) \gg (n = \#\text{ model parameters}), \quad \text{(Eq. 57)}$$

and the number of model parameters increases with the number of levels *L* in the VQ codebook. Now define,

$$O_{ij}^k = O(i, j, k),\ i \in \{1, 2, 3, 4, 5\},\ j, k \in \{1, 2, 3\} \quad \text{(Eq. 58)}$$

as the observation sequence of discrete symbols vector quantized from the preprocessed feature vectors of run R_{ij}, using the codebook Q_k. The observation sequences O_{ik}^k can be thought of as observation sequences which characterize the control strategy of known individuals, while the observation sequences O_{ij}^k, $j \neq k$, can be thought of as observation sequences of unknown origin, in need of classification. Tables 3, 4, and 5 classify each of the O_{ij}^k, $j \neq k$ based on O_{lk}^k for $k = 1$, $k = 2$, and $k = 3$, respectively, for eight-state HMMs. Note that the maximum value in each row is highlighted. We consider O_{ij}^k, $j \neq k$, classified correctly if and only if,

$$\sigma(O_{ik}^k, O_{ij}^k) > \sigma(O_{lk}^k, O_{ij}^k),\ \forall (j \neq k, l \neq i) \quad \text{(Eq. 59)}$$

In other words, we expect that two runs from the same individual (but on different roads) will yield a higher similarity measure than two runs from two different individuals. From the

tables, we observe that the similarity measure correctly classifies all 30 comparisons, by assigning the highest similarity between runs from the same individual, and significantly lower similarity between runs from two different individuals.

Now we compare these classification results with the Bayes optimal classifier. Define class ω_{ik} as,

$$\omega_{ik} = \omega(i,k) = \{\mu_{ik}, \Sigma_{ik}\},\ i \in \{1,2,3,4,5\},\ k \in \{1,2,3\}, \qquad \text{(Eq. 60)}$$

where μ_{ik} is the mean vector for R_{ik}, and Σ_{ik} is the covariance matrix for run R_{ik}. For each road *k*, we have five classes, one corresponding to each individual. Each data point $\bar{x} = \{v_\xi, v_\eta, \dot{\theta}, \delta, P_f\}$ in R_{ij}, $j \neq k$, is now classified into class ω_{lk} according to the Bayes decision rule [5],

$$g_{lk}(\bar{x}) > g_{ik}(\bar{x}),\ \forall l \neq i \qquad \text{(Eq. 61)}$$

where,

$$g_{ik}(\bar{x}) = -\frac{1}{2}\bar{x}^T\Sigma_{ik}^{-1}\bar{x} + \Sigma_{ik}^{-1}\mu_{ik}\bar{x} - \frac{1}{2}\mu_{ik}^T\Sigma_{ik}^{-1}\mu_{ik} - \frac{1}{2}\log|\Sigma_{ik}| + \log P(\omega_{ik}) \qquad \text{(Eq. 62)}$$

$$P(\omega_{ik}) = 1/5,\ i \in \{1,2,3,4,5\},\ k \in \{1,2,3\} \qquad \text{(Eq. 63)}$$

In Tables 3, 4, and 5 we report the percentage of data points in R_{ij}, $j \neq k$, which are classified in class ω_{lk} for $k = 1$, $k = 2$, and $k = 3$, respectively. We consider R_{ij} to be classified correctly when a plurality of the data from R_{lj} falls into class ω_{lk}, and observe, from the tables, that the Bayes optimal classifier misclassifies 7 out 30 (23%) of the runs. The performance of the similarity measure (0% error) therefore compares quite favorably.

Next, we present results for task-based classification. We select from each run R_{ij} all the left-turn maneuvers, and all the right-turn maneuvers. We get two resulting sets of maneuvers for each person,

$$\alpha_i, \beta_i,\ i \in \{1,2,3,4,5\} \qquad \text{(Eq. 64)}$$

where α_i corresponds to all the left-turn maneuvers for person *i*, and β_i corresponds to all the right-turn maneuvers for person *i*. We then split each of these sets into two — one to train a VQ codebook (or calculate the Bayesian statistics), the other to determine a similarity value (or the Bayesian classification). Tables 6 and 7 report the results for the similarity measure

Table 3: Classification: similarity measure *vs.* Bayes optimal classifier for road #1 data

	Stochastic similarity measure				
	O(1, 1, 1)	*O(2, 1, 1)*	*O(3, 1, 1)*	*O(4, 1, 1)*	*O(5, 1, 1)*
O(1, 2, 1)	0.765	0.172	0.117	0.099	0.315
O(1, 3, 1)	0.585	0.131	0.097	0.062	0.309
O(2, 2, 1)	0.187	0.857	0.358	0.287	0.012
O(2, 3, 1)	0.179	0.797	0.395	0.279	0.011
O(3, 2, 1)	0.255	0.407	0.845	0.343	0.019
O(3, 3, 1)	0.153	0.208	0.726	0.318	0.010
O(4, 2, 1)	0.073	0.189	0.287	0.644	0.006
O(4, 3, 1)	0.065	0.164	0.336	0.628	0.007
O(5, 2, 1)	0.038	0.004	0.003	0.009	0.428
O(5, 3, 1)	0.089	0.012	0.011	0.013	0.602

	Bayes optimal classifier				
	ω(1, 1)	*ω(2, 1)*	*ω(3, 1)*	*ω(4, 1)*	*ω(5, 1)*
R(1, 2)	0.338	0.131	0.017	0.059	0.455
R(1, 3)	0.295	0.102	0.033	0.033	0.538
R(2, 2)	0.249	0.353	0.053	0.288	0.057
R(2, 3)	0.172	0.405	0.095	0.294	0.034
R(3, 2)	0.134	0.256	0.344	0.232	0.034
R(3, 3)	0.067	0.160	0.539	0.220	0.014
R(4, 2)	0.216	0.208	0.150	0.357	0.069
R(4, 3)	0.129	0.173	0.285	0.374	0.039
R(5, 2)	0.003	0.000	0.003	0.001	0.994
R(5, 3)	0.051	0.005	0.032	0.004	0.908

Table 4: Classification: similarity measure *vs.* Bayes optimal classifier for road #2 data

	Stochastic similarity measure				
	O(1, 2, 2)	*O(2, 2, 2)*	*O(3, 2, 2)*	*O(4, 2, 2)*	*O(5, 2, 2)*
O(1, 1, 2)	0.753	0.176	0.108	0.115	0.353
O(1, 3, 2)	0.626	0.143	0.094	0.062	0.272
O(2, 1, 2)	0.209	0.860	0.350	0.276	0.013
O(2, 3, 2)	0.180	0.786	0.384	0.270	0.010
O(3, 1, 2)	0.236	0.406	0.826	0.340	0.018
O(3, 3, 2)	0.139	0.206	0.705	0.288	0.010
O(4, 1, 2)	0.082	0.185	0.267	0.674	0.007
O(4, 3, 2)	0.076	0.171	0.331	0.637	0.007
O(5, 1, 2)	0.038	0.003	0.003	0.008	0.429
O(5, 3, 2)	0.106	0.013	0.009	0.011	0.599

	Bayes optimal classifier				
	ω(1, 2)	*ω(2, 2)*	*ω(3, 2)*	*ω(4, 2)*	*ω(5, 2)*
R(1, 1)	0.579	0.202	0.046	0.096	0.077
R(1, 3)	0.635	0.090	0.041	0.025	0.209
R(2, 1)	0.225	0.490	0.152	0.130	0.003
R(2, 3)	0.232	0.500	0.150	0.117	0.002
R(3, 1)	0.111	0.329	0.407	0.150	0.002
R(3, 3)	0.061	0.203	0.539	0.196	0.001
R(4, 1)	0.271	0.365	0.102	0.261	0.001
R(4, 3)	0.115	0.267	0.294	0.322	0.000
R(5, 1)	0.341	0.010	0.035	0.123	0.491
R(5, 3)	0.256	0.006	0.044	0.053	0.641

Table 5: Classification: similarity measure *vs.* Bayes optimal classifier for road #3 data

	Stochastic similarity measure				
	O(1, 3, 3)	*O(2, 3, 3)*	*O(3, 3, 3)*	*O(4, 3, 3)*	*O(5, 3, 3)*
O(1, 1, 3)	0.773	0.144	0.131	0.106	0.376
O(1, 2, 3)	0.597	0.108	0.096	0.053	0.352
O(2, 1, 3)	0.249	0.869	0.340	0.333	0.015
O(2, 2, 3)	0.190	0.803	0.394	0.285	0.009
O(3, 1, 3)	0.280	0.442	0.806	0.397	0.025
O(3, 2, 3)	0.143	0.218	0.680	0.307	0.014
O(4, 1, 3)	0.095	0.210	0.297	0.676	0.010
O(4, 2, 3)	0.065	0.163	0.317	0.679	0.009
O(5, 1, 3)	0.047	0.003	0.005	0.011	0.503
O(5, 2, 3)	0.091	0.009	0.011	0.011	0.614

	Bayes optimal classifier				
	ω(1, 3)	*ω(2, 3)*	*ω(3, 3)*	*ω(4, 3)*	*ω(5, 3)*
R(1, 1)	0.458	0.264	0.032	0.096	0.150
R(1, 2)	0.513	0.172	0.014	0.057	0.245
R(2, 1)	0.185	0.575	0.078	0.132	0.030
R(2, 2)	0.208	0.588	0.037	0.129	0.037
R(3, 1)	0.086	0.451	0.270	0.181	0.012
R(3, 2)	0.108	0.424	0.255	0.196	0.018
R(4, 1)	0.201	0.456	0.046	0.292	0.004
R(4, 2)	0.114	0.440	0.083	0.361	0.001
R(5, 1)	0.173	0.010	0.015	0.109	0.693
R(5, 2)	0.059	0.000	0.001	0.000	0.940

Table 6: Classification of left-turn control

	Stochastic similarity measure				
	α*(1)*	α*(2)*	α*(3)*	α*(4)*	α*(5)*
α*(1)*	0.691	0.172	0.132	0.047	0.121
α*(2)*	0.072	0.832	0.263	0.200	0.007
α*(3)*	0.051	0.202	0.785	0.326	0.011
α*(4)*	0.026	0.161	0.280	0.720	0.013
α*(5)*	0.390	0.011	0.012	0.009	0.792

	Bayes optimal classifier				
	α*(1)*	α*(2)*	α*(3)*	α*(4)*	α*(5)*
α*(1)*	0.478	0.223	0.037	0.050	0.212
α*(2)*	0.165	0.559	0.130	0.131	0.016
α*(3)*	0.042	0.275	0.527	0.143	0.012
α*(4)*	0.161	0.339	0.163	0.328	0.009
α*(5)*	0.122	0.015	0.012	0.034	0.817

Table 7: Classification of right-turn control

	Stochastic similarity measure				
	β*(1)*	β*(2)*	β*(3)*	β*(4)*	β*(5)*
β*(1)*	0.713	0.118	0.109	0.036	0.223
β*(2)*	0.119	0.801	0.276	0.199	0.008
β*(3)*	0.109	0.390	0.742	0.324	0.009
β*(4)*	0.032	0.173	0.275	0.773	0.003
β*(5)*	0.244	0.006	0.014	0.005	0.875

	Bayes optimal classifier				
	β*(1)*	β*(2)*	β*(3)*	β*(4)*	β*(5)*
β*(1)*	0.253	0.188	0.015	0.052	0.492
β*(2)*	0.147	0.616	0.075	0.108	0.054
β*(3)*	0.105	0.330	0.412	0.137	0.015
β*(4)*	0.163	0.364	0.174	0.292	0.008
β*(5)*	0.105	0.014	0.024	0.028	0.830

(seven-state HMMs), and the Bayesian classification. Note again that the similarity measure classifies all 10 sets (5 left-turn, 5 right-turn) correctly, while, the Bayes classifier misclassifies 3 out of 10 (30%).

Finally, we present classification results for data which is more difficult to classify. Moe is asked to drive over the *same* road on two different days, two times each day, generating four runs (#1, #2, #3, #4). Because the runs are recorded on the same road, Moe is able to improve his skill relatively quickly. As recorded in Table 8, his average speed improves from 65.9 mph to 71.9 mph from run #1 to run #4. We take two additional data sets, one from Larry and one from Curly, over the same road. These data sets have similar aggregate statistics compared to at least some of Moe's runs.

Now we generate a 64-level VQ codebook with data from Larry's and Moe's fourth run, and another 64-level VQ codebook with data from Curly's and Moe's fourth run. We also generate corresponding classes for the Bayes-classifier comparison. We now classify each of Moe's first three runs as either similar to Larry or Moe #4, or as either similar to Curly or Moe #4.

Table 8: Aggregate statistics for additional human data

Name	v *(mph)*	$\dot{\theta}$ *(rad/s)*	δ *(rad)*	P_f *(N)*
Larry	61.9 ± 8.1	0.00 ± 0.18	0.00 ± 0.06	1297 ± 1485
Curly	65.6 ± 8.3	0.00 ± 0.20	0.00 ± 0.08	1829 ± 3531
Moe #1	66.1 ± 8.2	0.00 ± 0.23	0.00 ± 0.07	1743 ± 1816
Moe #2	65.9 ± 8.1	0.00 ± 0.21	0.00 ± 0.07	1771 ± 2432
Moe #3	67.4 ± 9.5	0.00 ± 0.23	0.00 ± 0.08	1917 ± 2633
Moe #4	71.9 ± 7.4	0.00 ± 0.24	0.00 ± 0.09	2182 ± 2149

Table 9 shows the classification results based on the similarity measure and Bayesian statistics. We observe that the similarity measure misclassifies one out of six (17%), while the Bayes classifier misclassifies five out of six (83%), some quite badly.

3.3 Discussion

Here, we discuss two issues which have not yet been addressed in the above results. First, we demonstrate why the Bayes classifier fails in some cases, where the similarity measure

Table 9: Difficult Classifications

	Stochastic similarity measure				
	Larry	*Moe #4*		*Curly*	*Moe #4*
Moe #1	0.572	0.528	Moe #1	0.315	0.616
Moe #2	0.435	0.540	Moe #2	0.495	0.603
Moe #3	0.258	0.728	Moe #3	0.550	0.760

	Bayes optimal classifier				
	Larry	*Moe #4*		*Curly*	*Moe #4*
Moe #1	0.609	0.391	Moe #1	0.569	0.431
Moe #2	0.589	0.411	Moe #2	0.663	0.337
Moe #3	0.416	0.583	Moe #3	0.567	0.433

succeeds. Figure 11(a) plots the distribution (over v and P_f) of Curly's data (Table 8), and the Gaussian approximation of that distribution. Likewise, Figure 11(b) and (c) show similar comparisons for Moe's second run and Moe's fourth run, respectively. It is clear that the Bayes classifier is doomed to fail, since the human data is distributed in a decidedly non-Gaussian manner. The similarity measure, on the other hand, succeeds because the HMMs are trained on the underlying distributions of the data sets, and make no *a priori* assumptions about each individual's distribution. We should also note that despite various attempts at improving the Bayes classifier's performance over the similarity measure — by only classifying on a subset of the vector $\{v_\xi, v_\eta, \dot{\theta}, \delta, P_f\}$ — we have yet to identify an example where the Bayes classifier succeeds and the similarity measure fails.

Second, we show that the increasing the number of states in the Hidden Markov Models improves the discrimination capability of the similarity measure. Consider the results in Tables 3, 4, and 5, and define the following measure of discrimination ζ_k:

$$\zeta_k = \frac{\sum_{i,\, l \neq k} \sigma(O_{ik}^k, O_{il}^k)}{\frac{1}{(n-1)} \sum_{i \neq j,\, l \neq k} \sigma(O_{ik}^k, O_{jl}^k)} \quad l, k \in \{1, 2, 3\} i, j \in \{1, 2, 3, 4, 5\} n = 5 \quad \text{(Eq. 65)}$$

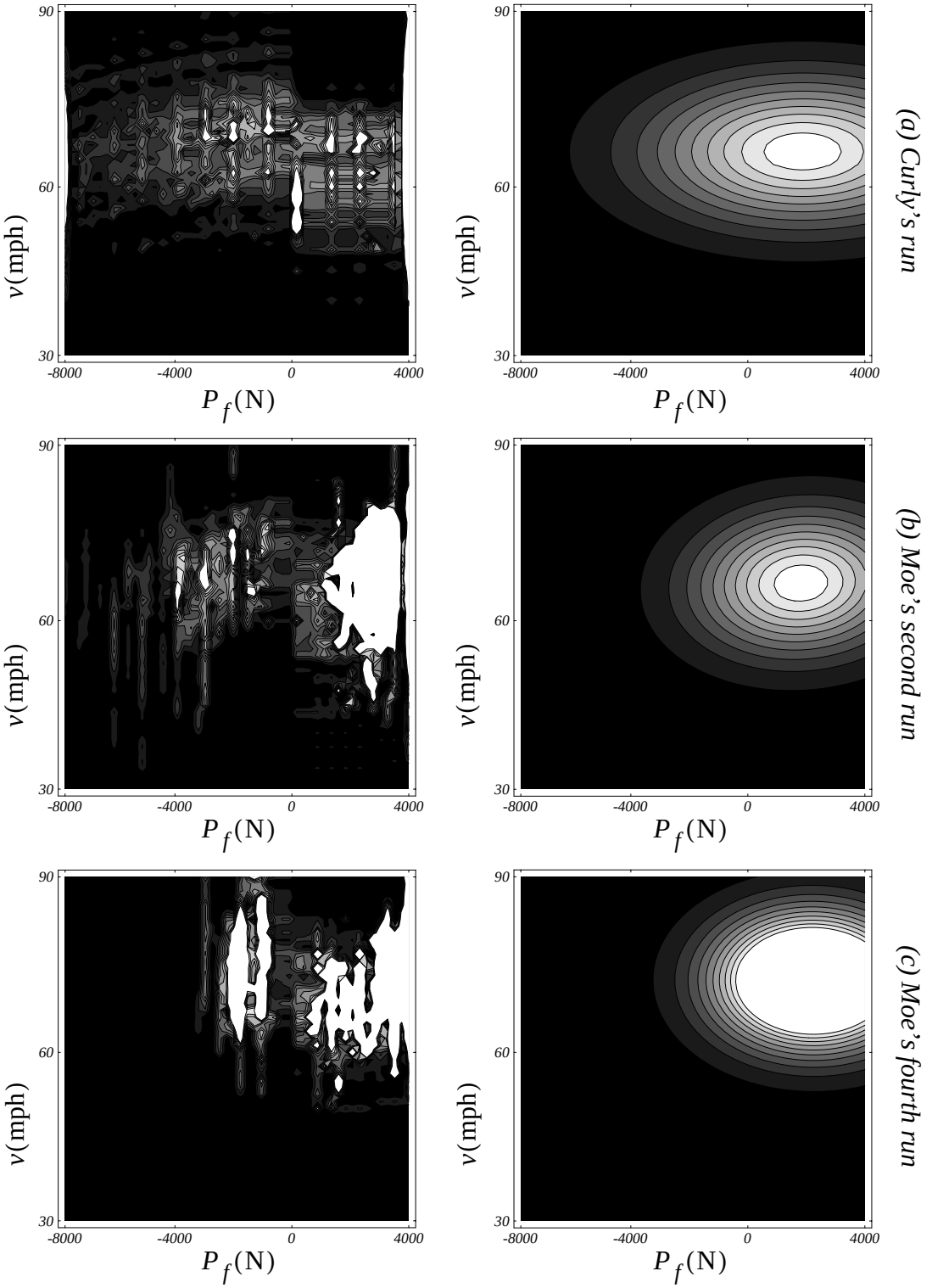

Figure 11: Moe's second run (b) is badly misclassified by the Bayes classifier as Larry's run (a) and not as Moe's fourth run (c). This happens because the assumption of Gaussian distributions (right) is not warranted by the real data (left). The similarity measure classifies Moe's second run correctly.

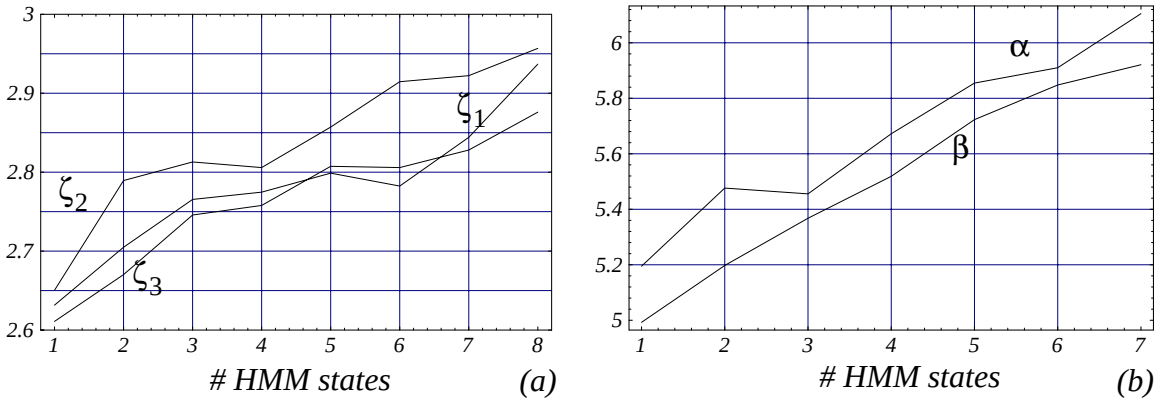

Figure 12: The discrimination between people improves as a function of the number of states in the Hidden Markov Model. Note also that the discrimination for the task-based classification (b) is twice that for the classification across different roads (a).

Essentially, ζ_k forms a ratio of self-similarities over averaged cross-similarities between individuals for observation sequences vector quantized on codebook Q_k. Figure 12(a) below plots this ratio as the number of states in the respective HMMs is varied from 1 to 8. Figure 12(b) plots a similar ratio for the left-turn/right-turn classifications. From Figure 12, we make two observations: (1) the discrimination of the similarity measure is affected positively by imparting structure onto the statistical model (i.e. the HMM) in the form of an increased number of HMM states, and (2) the discrimination of the similarity measure improves significantly (two-fold in this example), when we train on specific maneuvers (i.e. left turns and right turns) rather than arbitrary roads.

4 Validating human control strategy models

4.1 Learning human control strategy

In learning human control strategy, such as driving, 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 [6] with variable activation functions [16, 17, 19] 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 (see Figure 13). Moreover, 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 [19].

Below, we report cascade modeling results for human control data collected from the driving simulator described in Section 3.1 above. The inputs to the cascade networks for each individual include (1) current and previous state information (Eq. 66), (2) previous control information (Eq. 67), and (3) a description of the road (Eq. 68),

$$\left\{ \begin{array}{l} v_\xi(k-n_s), \ldots, v_\xi(k-1), v_\xi(k), \ldots \\ v_\eta(k-n_s), \ldots, v_\eta(k-1), v_\eta(k), \dot{\theta}(k-n_s), \ldots, \dot{\theta}(k-1), \dot{\theta}(k) \end{array} \right\}, \qquad \text{(Eq. 66)}$$

$$\{\delta(k-n_c), \ldots, \delta(k-1), \delta(k), P_f(k-n_c), \ldots, P_f(k-1), P_f(k)\}, \qquad \text{(Eq. 67)}$$

$$\{x(1), x(2), \ldots, x(n_r), y(1), y(2), y(n_r)\}, \qquad \text{(Eq. 68)}$$

where n_s = length of state history to include as input, and n_c = length of command history to include as input. For the road description, we discretize the visible view (given a specific horizon) of the road ahead into n_r equivalently spaced, body-relative (x, y) coordinates of the road median, and provide that sequence of coordinates as input to the network. Thus, for fixed n_s, n_c, and n_r, the total number of inputs to the network will be,

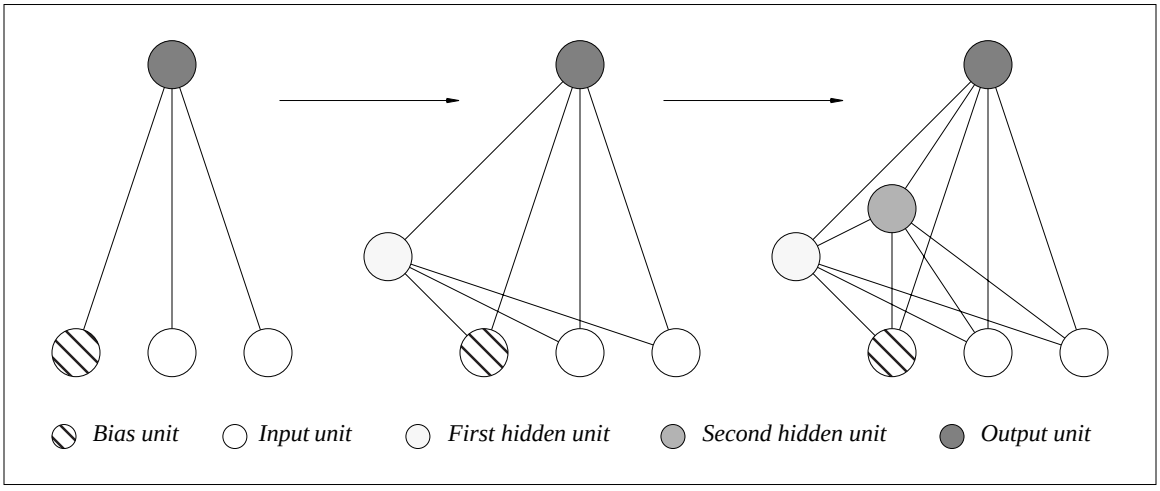

Figure 13: The cascade learning architecture adds hidden units one at a time to an initially minimal network. All connections in the diagram are feed-forward.

$$n_{inputs} = 3n_s + 2n_c + 2n_r \quad \text{(Eq. 69)}$$

The outputs for the cascade network are, of course, $\{\delta(k+1), P_f(k+1)\}$ (i.e. the steering and acceleration command for the next time step).

4.2 Similarity results

The left sides of Figure 14(a) and (b) show part of the driving data collected from two individuals, Oliver and Stan. Note that the driving styles for the two individuals are quite different for the same road. The right sides of Figure 14(a) and (b) show part of the cascade model-generated command trajectories for Oliver and Stan. Since Stan's control strategy is relatively simple, his control strategy model requires 30 hidden units and only the previous two states as input (i.e. $n_s = n_c = 2$). Oliver's more complicated control strategy model, on the other hand, requires 32 hidden units and relies on the previous ten states ($n_s = n_c = 10$) to stay on the road. We determine these input histories experimentally to achieve stable road following for each HCS model. For both models, we let $n_r = 15$.

Table 10 below summarizes the similarity results for the data in Figure 14. All pre-processing and vector quantization was performed in the same manner as in Section 3.2.

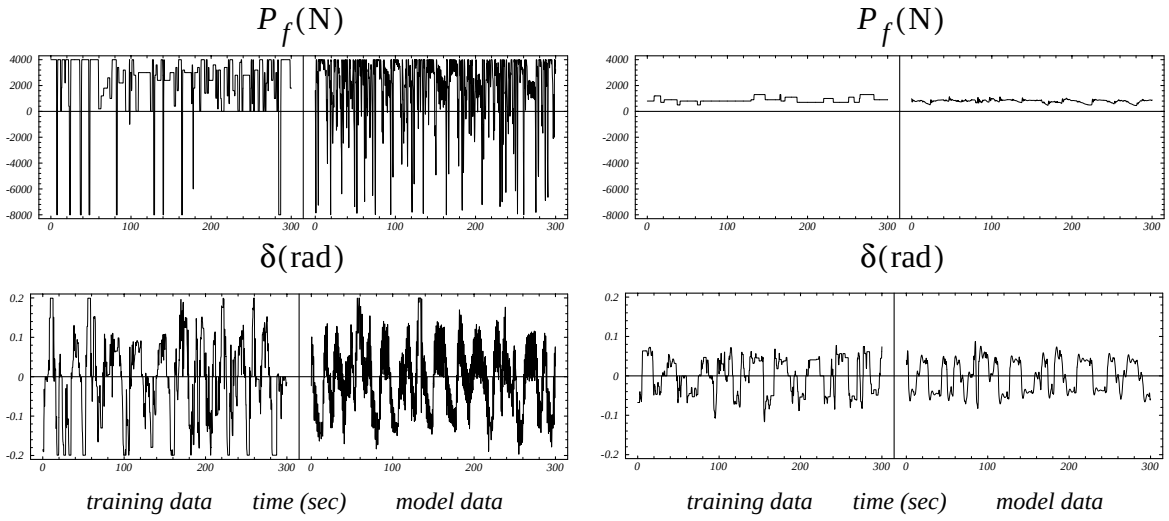

Figure 14: (a) Oliver's driving data, and (b) Stan's driving data. On the left of each figure, we show part of the source training data; on the right of each figure we show the corresponding model-generated data.

Table 10: Human-to-model similarity results

σ	*Stan*	*Stan's HCS model*	*Oliver*	*Oliver's HCS model*
Stan	1.000	0.748	0.009	0.007
Stan's HCS model	0.748	1.000	0.006	0.004
Oliver	0.009	0.006	1.000	0.349
Oliver's HCS model	0.007	0.004	0.349	1.000

4.3 Discussion and future work

The similarity results in Table 10 confirm two qualitative assessments of the data in Figure 14(a) and (b). First, we observe that the two driving styles are objectively quite different. This fact is reflected in the low similarity measures between one individual's model and the other individual's source and 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, 0.748 and 0.349. This is indicative that Oliver's sharply discontinuous driving strategy is more difficult to learn by a single cascade network than Stan's calmer approach. Indeed, Oliver's model generates significant oscillatory behavior, of which Oliver himself is not guilty.

Thus, the relatively low similarity measure between Oliver and Oliver's model points to a problem in the model itself, including possible improper assumptions about the controller order, granularity and control delay of the actual HCS. As a matter of fact, the values for n_s, n_c, and n_r for each model were arrived at in an essentially *ad hoc* manner. To correct this problem, we are at present working on an algorithm to improve a model's input representation based on the similarity measure. We propose combining the similarity measure with *simultaneously perturbed stochastic approximation (SPSA)* [26] to select the best model input representation.

In general, a given control strategy can be approximated by,

$$\bar{u}[(k+1)\tau + \delta] = \Gamma\begin{pmatrix}\bar{x}[k\tau], \bar{x}[(k-1)\tau], \ldots, \bar{x}[(k-m)\tau] \\ \bar{u}[k\tau], \bar{u}[(k-1)\tau], \ldots, \bar{u}[(k-n)\tau]\end{pmatrix} \qquad \text{(Eq. 70)}$$

where $\Gamma(\ \cdot\ )$ is some arbitrary unknown function, $\bar{u}(k)$ is a vector of control outputs, $\bar{x}(k)$ is a vector of sensory inputs (including both state and environment variables) at time step *k*, τ indicates the controller resolution or granularity, and δ indicates the control delay. The

order of the dynamic system is given by the constants *n* and *m*. In order to select the best input representation for the HCS model, we need to optimize the parameter vector,

$$\theta = \left[m \; n \; \tau \; \delta \right]^T, \qquad \text{(Eq. 71)}$$

for some given experimental data. Let, $\Gamma(\theta)$ denote a trained HCS model with input representation θ, and let $\sigma[\Gamma(\theta)] = \sigma(\theta)$ denote the similarity measure for model $\Gamma(\theta)$. Now, the best input representation θ^* is defined in terms of the similarity measure σ, such that,

$$\sigma(\theta^*) > \sigma(\theta), \; \forall \theta \neq \theta^* \qquad \text{(Eq. 72)}$$

This optimization is difficult in principle since (1) we have no explicit gradient information,

$$G(\theta) = \frac{\partial}{\partial \theta} \sigma(\theta) \qquad \text{(Eq. 73)}$$

and (2) each measurement of σ is computationally expensive. Thus, we resort to *simultaneously perturbed stochastic approximation* to perform the optimization. We propose to evaluate the similarity measure σ for two values of θ in order to arrive at a new estimate for θ at iteration *k*. Figure 15 illustrates the overall loop, where Δ_k is a vector of mutually independent, mean-zero random variables (e.g. symmetric Bernoulli distributed), the sequence $\{\Delta_k\}$ is independent and identically distributed, and the $\{a_k\}$, $\{c_k\}$ are positive scalar sequences.

5 Conclusion

Model validation is an important problem in the area of machine learning for dynamic systems, if learned models are to be exploited to their full potential. In this paper, we have derived a stochastic similarity measure, based on Hidden Markov Model analysis, by which we can compare and contrast arbitrary, multi-dimensional stochastic trajectories. We have shown that this method performs significantly better than traditional statistical techniques in classifying human control strategy data. Furthermore, we have shown that the similarity measure offers a feasible means of validating a learned HCS model's performance to its training data. Finally, we have proposed an iterative algorithm, based on the similarity measure, which allows us to refine a model's input representation to improve model fidelity. Such an algorithm is of special relevance in learning human control strategy, where little is known *a priori* about the structure, order, or granularity of each individual's controller.

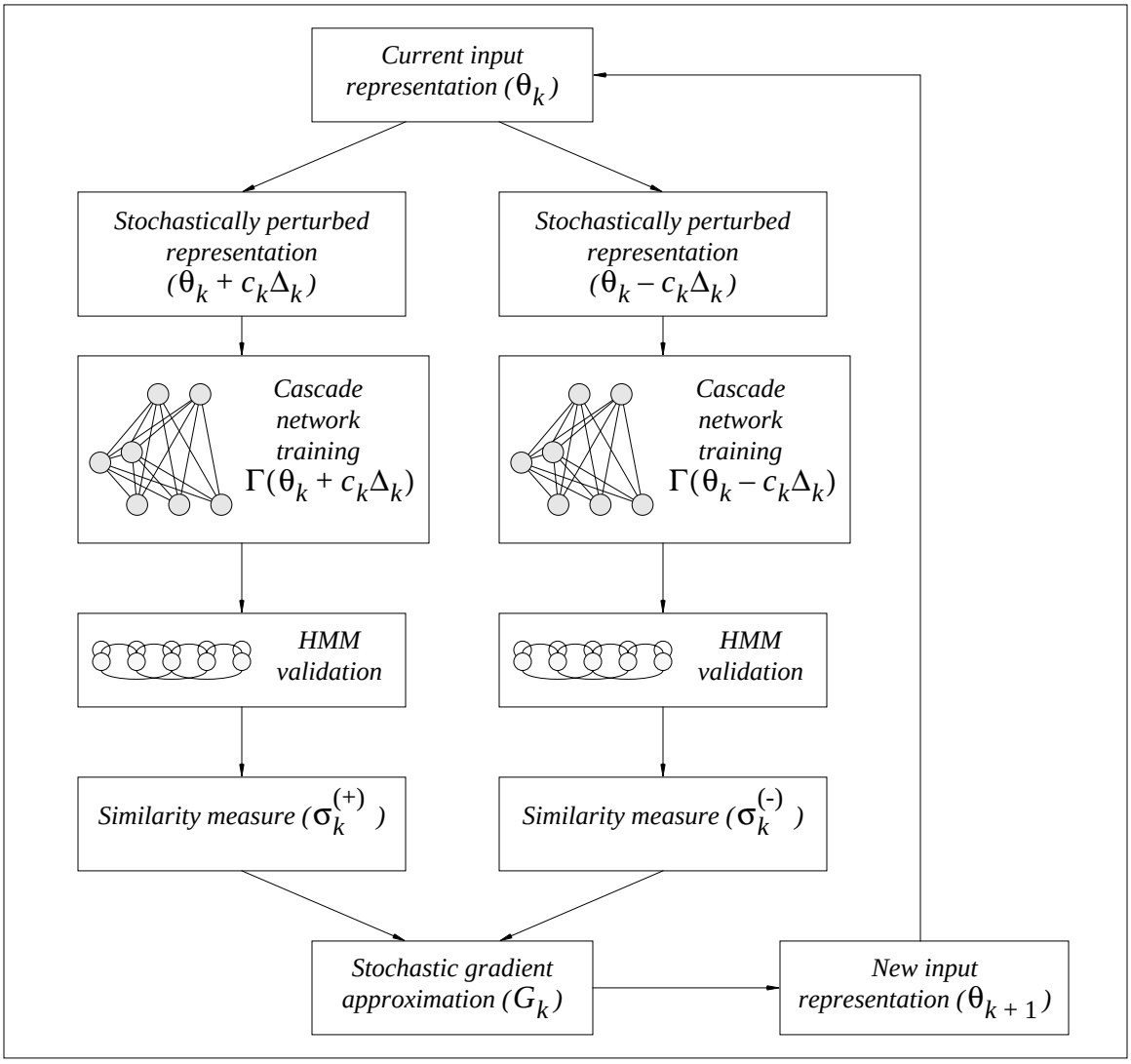

Figure 15: Overall approach to stochastic optimization of controller structure.

6 Acknowledgments

We would like to thank the many people who patiently “drove” through our driving simulator. Special thanks go to “Larry”, “Curly”, “Moe”, “Groucho” and “Harpo” for their patience as they navigated through some 50 miles of simulated road.

7 References

[1] P. Baldi and Y. Chauvin, “Smooth On-Line Learning Algorithm for Hidden Markov Models,” *Neural Computation*, Vol. 6, No. 2, pp. 307-318, 1994.

[2] R. Basri and D. Weinshall, “Distance Metric Between 3D Models and 2D Images for Recognition and Classification,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 4, pp. 465-479, 1996.

[3] L. E. Baum, T. Petrie, G. Soules and N. Weiss, “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.

[4] M. Boninsegna and M. Rossi, “Similarity Measures in Computer Vision,” *Pattern Recognition Letters*, Vol 15., No. 12, pp. 1255-1260, 1994.

[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

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

[7] B. Hannaford and P. Lee, “Hidden Markov Model Analysis of Force/Torque Information in Telemanipulation,” *Int. Journal of Robotics Research*, Vol. 10, No. 5, pp. 528-539, 1991.

[8] 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.

[9] J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing, Redwood City, 1991.

[10] X. D. Huang, Y. Ariki and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990.

[11] R. Jain, S. N. J. Murty, *et. al.*, “Similarity Measures for Image Databases,” *Proc. IEEE Int. Conf. on Fuzzy System*s, Vol. 3, pp. 1247-1254, 1995.

[12] V. Krishnamurthy and J. B. Moore, “On-Line Estimation of Hidden Markov Model Parameters Based on the Kullback-Leibler Information Measure,” *IEEE Trans. Signal Processing*, Vol. 41, No. 8, pp. 2557-2573, 1993.

[13] A. Kundu, G. C. Chen and C. E. Persons, “Transient Sonar Signal Classification Using Hidden Markov Models and Neural Nets,” *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 1, pp. 87-99, 1994.

[14] K. Y. Kupeev and H. J. Wolfson, “On Shape Similarity,” *Proc. of 12th IAPR Int. Conf. on Pattern Recognition*, Vol. 1, pp. 227-231, 1994.

[15] Y. Linde, A. Buzo and R. M. Gray, “An Algorithm for Vector Quantizer Design,” *IEEE Trans. Communication*, Vol. COM-28, No. 1, pp. 84-95, 1980.

[16] M. C. 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.

[17] M. C. 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.

[18] M. C. Nechyba and Y. Xu, “On the Fidelity of Human Skill Models,” *Proc. IEEE Int. Conf. on Robotics and Automation*, Vol. 3, pp. 2688-2693, 1996.

[19] M. C. 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.

[20] 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.

[21] L. R. Rabiner, B. H. Juang, S. E. Levinson and M. M. Sondhi, “Some Properties of Continuous Hidden Markov Model Representations,” *AT&T Technical Journal*, Vol. 64, No. 6, pp. 1211-1222, 1986.

[22] G. Radons, J. D. Becker, B. Dulfer and J. Kruger, “Analysis, Classification, and Coding of Multielectrode Spike Trains with Hidden Markov Models,” *Biological Cybernetics*, Vol. 71, No. 4, pp. 359-373, 1994.

[23] K. R. Rao and D. F. Elliott, *Fast Transforms: Algorithms, Analyses, Applications*, Academic Press, New York, 1982.

[24] L. G. Sotelino, M. Saerens and H. Bersini, "Classification of Temporal Trajectories by Continuous-Time Recurrent Nets," *Neural Networks*, Vol. 7, No. 5, pp. 767-776, 1994.

[25] H. Y. Shum, M. Hebert and K. Ikeuchi, "On 3D Shape Similarity," Technical Report, CMU-CS-95-212, Carnegie Mellon University, 1995.

[26] J. C. Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Trans. Automatic Control*, Vol. 37, No. 3, pp. 332-341, 1992.

[27] M. Sun, G. Burk and R. J. Sclabassi, "Measurement of Signal Similarity Using the Maxima of the Wavelet Transform," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 583-586, 1993.

[28] M. Werman and D. Weinshall, "Similarity and Affine Invariant Distances Between 2D Point Sets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, pp. 810-814, 1995.

[29] J. Yamato, S. Kurakakae, A. Tomono and K. Ishii, "Human Action Recognition Using HMM with Category Separated Vector Quantization," *Trans. Institute of Electronics, Information, and Communication Engineers D-II*, Vol. J77D-II, No. 7, pp. 1311-1318, 1994.

[30] J. Yamato, J. Ohya and K. Ishii, "Recognizing Human Action in Time-Sequential Images Using Hidden Markov Models," *Trans. Institute of Electronics, Information, and Communication Engineers D-II*, Vol. J76D-II, No. 12, pp. 2556-2563, 1993.

[31] 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.

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