On-line Memory-based Detection of General Purpose Systems

Kan Deng **Andrew W. Moore** **Michael C. Nechyba**

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Kdeng@cs.cmu.edu

Abstract

By combining memory-based learning methods with likelihood analysis, we provide a technique which can distinguish the underlying system given a set of the system input and output observations. It is surprising that this straightforward technique is capable of doing many hard jobs. As a demonstration, we use it to distinguish different styles of manipulating a tennis simulator. Also, we apply it to a driving style detection domain, using both simulation and real world data.

1 INTRODUCTION

Let's begin with explaining the title. A system usually has inputs and outputs. The output is a function of the input, plus noise. We assume both the input and output are fully observable. Some systems are dynamic, because they have feedback. Others are static: simply mappings between inputs and outputs.

By "general purpose" we mean the input and output of a system can be any type of value. They can be continuous, or discrete, including categorical, or even a mixture of them. More importantly, given a specific input, the output can be of any distribution. For some systems, the outputs corresponding to the same input may scatter around a center, so that the conditional distribution of the output with respect to an input is reasonable to be formed as Gaussian. However, for a general purpose system, we don't need this Gaussian assumption. Later in this paper, we will find that the multinomial discrete distribution is very interesting.

The task of system detection is that given a set of observations of input and output signals, we want to figure out who generates these signals among a finite set of system candidates.

In some situations, we don't have the closed-form knowledge of the candidate systems; instead, we have stored previous observations of the input and output signals for each of them. To detect a set of unlabeled observations generated by a unknown system, we compare the unlabeled signals with the observations of each candidate system. If the unlabeled observations are similar to the observations from a candidate, the underlying system is likely to be the corresponding candidate system. That is the principle of "memory-based system detection". To be on-line, we need some tricks to make the processing sufficiently fast.

For the sake of convenience, we name our method On-line Memory-based GenerAl purpose system detection (*OMEGA)*.

2 RELATED WORK

The related work in literature can be referred to system identification, system recognition, system verification, prediction, novelty detection, etc. Each of them has its own goal. The comparison between them and OMEGA is only in the point of view of system detection.

Classification Based on Features.

A simple way to detect a system is to calculate the distance from the centroid of the unlabeled observations to the centroid of a certain candidate system's memory observations, then compare the distances with respect to different candidates to figure out the most likely one. In literature, this method is called Bayes classifier [Duda, 73].

However, this method assumes that the distributions of the unlabeled observations and those of the candidate systems are Gaussian; but maybe with different parameters. In many domains, this assumption doesn't hold. Of course, Bayes classifiers can be improved with the help of complicated distribution approximating techniques, but that may increase the computational cost, and begs the question of who decides what these distributions should be.

Linear System Identification

System identification from control tries to figure out the parameters of a system, given a set of observation and a priori knowledge of the type of the system, e.g. linear. Once we have known the parameter values of the system, it is straightforward to compare the unlabeled system with the candidate ones [Makhoul, 75], [Eykhoff, 74], [Basseville, 93]. However, there are two restrictions in this technique.

1. Because of the linear assumption, this technique performs badly in case the underlying system is non-linear. Also because of the linearity, the output must be uni-modally distributed. This assumption doesn't hold in many domains.

2. In many cases, the system model is assumed to be global. That is, for different inputs, the system model's parameters must keep the same values.

Neural Detection

Because of the restrictions of the linear model, some researchers use neural nets to approximate the system models. Before the detection work starts, a neural network is configured and trained off-line for each of the candidate systems. To detect a set of unlabeled observations, one can compare the observed outputs with those predicted by the neural nets [Narendra, 90]. Since each candidate system needs a neural net, it is hard to update them when new training data keep coming. This problem becomes more serious when the number of candidate systems is large.

Hidden Markov Model

HMM [Rabiner, 89] plays an important role in time series analysis [Nechyba, 98]. HMM assumes the input and output signals of a system are decided by some hidden states, and the transition probabilities of these states determine the similarity of different time series observations. This assumption works well with speech; but it is arguable whether or not in other domains the transition probability is the only principal characteristic of a time series. Besides, the training of HMM is data-consuming. And similarly to neural nets, each candidate system needs a HMM model. Hence, it is not easy to update the HMM models on-line, too. Further discussion on the comparison between HMM and OMEGA refers to [Deng, 98].

3 MEMORY-BASED SYSTEM DETECTION.

Given an observation of input and output, (x_t, y_t), the probability that this observation is generated by a system S_p, $P(S_p \mid x_t, y_t)$, is proportional to $P(y_t \mid x_t, S_p)$. If y_t is continuous and its distribution is uni-modal, one can always roughly treat y_t as Gaussian distributed.

$$P(y_t \mid S_p, x_t) = \frac{1}{\sqrt{2p}\, s} \exp\{-\frac{[y_t - E(y \mid S_p, x_t)]^2}{2s^2}\}$$

In the situation that the closed-form function relationship between x_t and y_t is unknown, but there are sufficient previous observations of (x_t, y_t) generated by the system S_p, memory-based machine learning methods can be used to approximate $E(y \mid S_p, x_t)$ and s. Kernel regression is one of these methods. If there are N data points generated by S_p, (x_i, y_i), $i = 1, 2, \ldots, N$,

$$E(y_t \mid S_p, X_t) = \sum_{i=1}^{N} w_i y_i \Big/ \sum_{i=1}^{N} w_i \qquad s^2 = E(y_t^2 \mid S_p, x_t) - E(y_t \mid S_p, x_t)$$

Now let's consider the situation that y_t is continuous but not uni-modal. Theoretically, we should use state-of-art techniques to approximate the distribution of $P(y \mid S_p, x_t)$. But in practice, we simply discretize y and treat it as a multinomial distribution.

If y is a discrete variable whose distribution is multinomial,

$$P(y_t \mid S_p, x_t) = \sum_{i=1}^{N} w_i \Phi(y_t, y_i) \Big/ \sum_{i=1}^{N} w_i \qquad \Phi(y_t, y_i) = \begin{cases} 1 & \text{when } y_t = y_i \\ 0 & \text{otherwise} \end{cases}$$

Note that multinomials are usually used to model the distribution of categorical variables. One cannot say one value of a categorical variable is closer to another value, but further from a third one. Therefore, the multinomial approximation of a continuous multi-modal distributed variable is in fact the lower bound of the correct approximation.

If there are a set of observations, the *likelihood* is a good metric of the overall probability that the observations are generated by an assigned system. For the sake of computational convenience, we use the average of the negative log likelihood in our system,

$$-lik(S_p) = -\sum_{t=1}^{T} \log P(y_t \mid x_t, S_p) \Big/ T$$

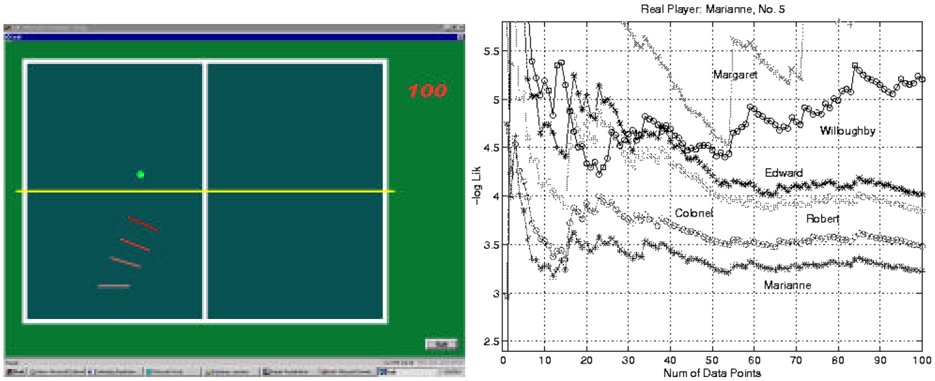
When $-lik(S_p)$ is close to zero, $P(y_t \mid x_t, S_p)$ is near *1.0*, therefore, the underlying system which generates (x_t, y_t) is very likely to be S_p.

This formula works for most cases. However, when the density of data points varies greatly with different x_t, the approximation of $P(y_t \mid x_t, S_p)$ may have different confidence. To adjust the overall log likelihood, we should insert a weight into the sum. The weight is a function of the density of data points near x_t. One possible implementation is,

$$w(x_t) = \sum_{i=1}^{N} w_i \Big/ N \qquad N \text{ is the number of data points in memory}$$

To make the process sufficiently fast to be on-line, we use a kd-tree for kernel regression [Deng, 95]. Kd-tree has other good features, for example, it can help us focus on the more promising candidate systems, while ignoring the others, from the very beginning of the process.

4 EXPERIMENT TWO: TENNIS SIMULATION.



(a) (b)

Figure 1 a. Tennis simulator interface. b. Likelihood curves of six human

In this experiment, we designed a simple simulator of tennis, to study different people's performance styles. The ball is served automatically from a random position in the upper half field with a random speed and a random direction towards the bottom line. A human player can control the racket by moving the mouse. The line segments in Figure 1.a illustrate the recent movement of the racket. Notice that this is not a dynamic system, because every serving and hit is independent without feedback.

Six people were invited to do the experiment. Each of them played sixteen runs, and during each run, they hit the ball one hundred times. For each hit, we recorded eight variables: the position where the ball was served, the ball's speed and direction, the position where the racket hit the ball, and also the speed and direction of the ball after the contact. We assigned the first six variables as the inputs, and the latter two as the outputs. For the same input, different people's outputs may be of distinguishable distributions. We didn't evaluate the merit of the performance, we only focused on the different styles.

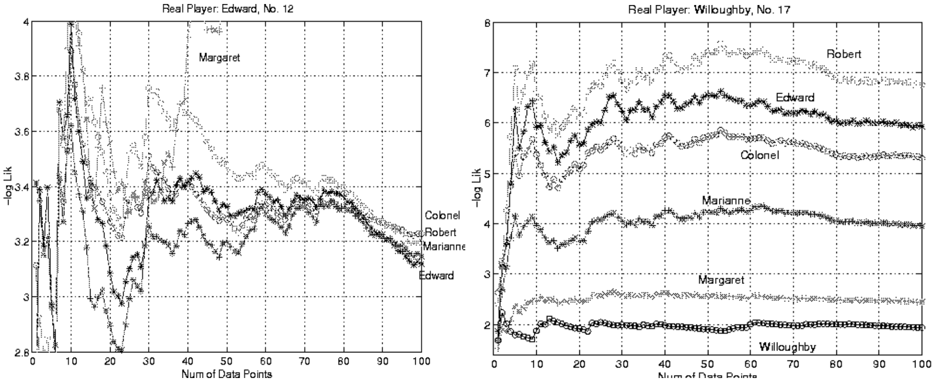
One run from each player was used as a test set, with the other nineteen used for training. To detect who was the hidden player of a certain test data set, we tried OMEGA with respect to all the players' memory data sets one by one; hence, we got six average negative log likelihood curves. The curve closest to the horizontal axis indicates the hidden player, because this curve’s average negative log likelihood is closest to zero, in other words, its likelihood is closest to the maximum, *1.0*.

Figure 1.b is a typical picture of the likelihood curves, which detects Marianne is the hidden player. This result is correct. We tried ninety-six cases. Among them, eighty-five are correct, four are wrong and seven are confused. By confusion, we mean this situation: although the lowest curve does correspond to the correct player, there is another curve very close it, so that they are not distinguishable from each other. Figure 2.a shows a confused case.

The likelihood curves are bumpy some time. This is because the player performed in a way that hasn't been observed in memory. If a performance was so strange that it never happened to all the players, then all the likelihood curves are bumpy, in a way roughly paralleling each other. Therefore, the bumpiness implies the consistency of the players. Comparing Figure 2.a with Figure 2.b, it is obvious that Willoughby is more consistent than Edward.

The distances among the likelihood curves imply whose performances are similar. In this experiment, Margaret and Willoughby behaved similarly, referring to Figure 2.b. But they are different from the others. As in Figure 2.a, their curves were so much higher than the others that they are off the graph.

Table 1 is the comparison of the accuracy between OMEGA and Bayes classifier. It is obvious that OMEGA outperforms Bayes classifier by a large margin. HMM cannot be used in this domain, because it is not a dynamic system.



(a) (b)

Figure 2. The likelihood curves of the tennis simulation experiments by six players.

Table 1 The comparison of the accuracy between Bayes classifier and OMEGA

	Correct	Wrong	Confused
Bayes	34	40	22
OMEGA	85	4	7

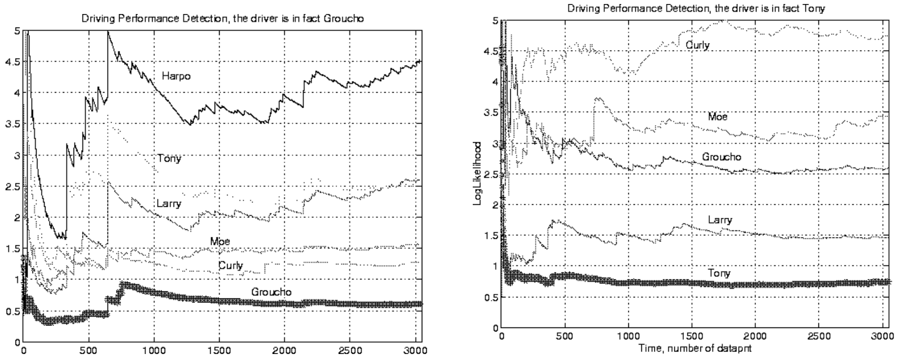


Figure 3. Simulation driving style detection.

Table 2. The comparison of Bayes classifier, HMM and OMEGA.

	Correct	Wrong	Confused
Bayes	0	5	10
HMM	13	0	2
OMEGA	13	0	2

5 EXPERIMENT THREE: DRIVING STYLE.

We have also done experiments to distinguish driving styles using both simulation and real world data. For simulation, we asked five people to operate a driving simulator by moving the mouse and pushing the buttons to control direction and speed. The input contains the vehicle's speed, direction, its position in the road, and their previous values. Besides, the curvature of the road, the previous values of the driver's control of the steering angle and gas/brake are also included in the input. The output is assigned to be the driver steering control. Due to the limitation of space, we cannot explain the experiment and its results in detail. Figure 3 demonstrates the success of OMEGA in this simulation domain. Table 2 compares OMEGA with Bayes classifier and HMM. Both HMM and OMEGA significantly outperform Bayes classification, but it is hard to tell who is better between HMM and OMEGA.

Figure 4 is OMEGA’s performance in a real world driving domain. The data was collected by NavLab of CMU [Pomerleau, 96]. Eight people drove a vehicle with many sensors from Pittsburgh to Grove City and back. This experiment is more difficult than the simulation one, not only because the data is from real world, but also does it involve traffic conditions as a part of the input

The result in Table 3 is easy to understand, but the interesting thing is that OMEGA even outperforms HMM which is a more complicated method. Further discussion on the comparison of OMEGA and HMM goes to [Deng, 98].

CONCLUSION.

By combining memory-based learning methods with likelihood analysis, we come to a general purpose system detection technique. Although this technique is

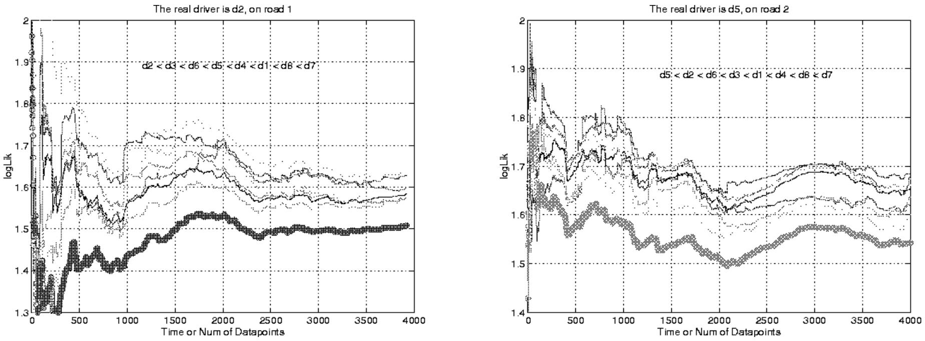


Figure 4. NavLab Real world driving style detection.

Table 3. The comparison of Bayes classifier, HMM and OMEGA.

	Correct	Wrong	Confused
Bayes	0	5	11
Global Linear	4	7	5
HMM	11	2	3
OMEGA	13	0	3

straightforward, it is a surprise that it is capable of doing many difficult jobs with competitive quality.

Reference:

[Basseville, 93] Detection of Abrupt Changes: Theory and Application. By M. Basseville, I.Nikiforov, 1993.

[Deng, 95] Multiresolution Instance-based Learning. By K.Deng, A.W.Moore. Proc. of IJCAI, 1995.

[Deng, 98] On the Comparison of Memory-based System Detection Method and Hidden Markov Model Approach. By K.Deng, A.W.Moore. In preparation.

[Duda, 73] Pattern Classification and Scene Analysis. By R.O.Duda, P.E.Hart. Published by John Wiley & Sons, New York, 1973.

[Eykhoff, 74] System Identification: Parameter and State Estimation. By P.Eykhoff. Published by: Wiley-Interscience New York. 1974.

[Makhoul, 75] Linear Prediction, by J.Makhoul: Proc. of IEEE. Vol 63, No. 4, April, 1975.

[Narendra, 90] Identification and Control of Dynamical Systems Using Neural Networks. By K.S.Narendra, K.Parthasarathy. IEEE Trans. On Neural Networks, vol. 1, pp. 4 –27, March, 1990.

[Nechyba, 98] On Discontinuous Human Control Strategies. By M.C.Nechyba, Y.Xu, Proc. IEEE Int. Conference on Robotics and Automation, May, 1998.

[Pomerlear, 96] Rapidly Adaptive Machine Vision for Automated Vehicle Steering. By D.A.Pomerleau, T.Jochem. IEEE Expert, vol. 11, No.2, pp. 19-27, 1996.

[Rabiner, 89] A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, By L.R.Rabiner, Proc. of IEEE, vol. 77, No. 2, Feb. 1989.