A General Framework to Detect Unsafe System States From Multisensor Data Stream

Huazhong Ning, Wei Xu, Yue Zhou, Yihong Gong, and Thomas S. Huang, *Life Fellow, IEEE*

***Abstract*—This paper proposes a general framework for detecting unsafe states of a system whose basic real-time parameters are captured by multiple sensors. Our approach is to learn a danger-level function that can be used to alert the users of dangerous situations in advance so that certain measures can be taken to avoid the collapse. The main challenge to this learning problem is the labeling issue, i.e., it is difficult to assign an objective danger level at each time step to the training data, except at the collapse points, where a definitive penalty can be assigned, and at the successful ends, where a certain reward can be assigned. In this paper, we treat the danger level as an expected future reward (a penalty is regarded as a negative reward) and use temporal difference (TD) learning to learn a function for approximating the expected future reward, given the current and historical sensor readings. The TD learning obtains the approximation by propagating the penalties/rewards observable at collapse points or successful ends to the entire feature space following some constraints. This avoids the labeling issue and naturally allows a general framework to detect unsafe states. Our approach is applied to, but not limited to, the application of monitoring driving safety, and the experimental results demonstrate the effectiveness of the approach.**

***Index Terms*—Driving safety, labeling issue, multisensor, temporal difference (TD) learning, unsafe system state.**

I. Introduction

THE INCREASING advance in sensor technologies has enabled applications that automatically capture a variety of parameters of a complex system in real time, such as vehicles and computer networks. By analyzing multisensor data streams, we can detect the state/behavior of the system and react proactively to prevent certain states/behaviors from happening. An effective detector of unsafe states can largely save us from economic losses (e.g., collapse of a computer network) and from exposure to dangerous situations (e.g., a driving accident). We call it "unsafety detector."

Manuscript received May 1, 2008; revised November 11, 2008, March 28, 2009, and May 6, 2009. The Associate Editor for this paper was Q. Ji.

H. Ning is with the AdCenter Laboratories, Microsoft Corporation, Redmond, WA 98052 USA (e-mail: huaznin@microsoft.com).

W. Xu and Y. Gong are with the NEC Laboratories America, Inc., Cupertino, CA 95014-8505 USA (e-mail: xw@sv.nec-labs.com; ygong@sv.nec-labs.com).

Y. Zhou is with Microsoft Corporation, Redmond, WA 98052 USA (e-mail: yue.zhou@microsoft.com).

T. S. Huang is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA, and also with the Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: huang@ifp.uiuc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2009.2026446

At first, we use an example to make the problem clear, which is related to computer networks and service-level agreements (SLAs). Usually, a service provider signs an SLA with a client that guarantees certain quality of services (e.g., a guarantee of 5 ms or less response time of the network). The service provider is subject to a fine when the SLA is violated. Therefore, they need to monitor the network system by capturing and analyzing the basic parameters, such as CPU usage, network traffic, and memory usage, at every time stamp. The unsafe states could be predicted in advance from abnormal temporal patterns of the system parameters. Proactive measures need to be taken before the system collapses into unsafe states to avoid the SLA violations.

In this paper, the term "unsafe" refers to the high probability of system collapse in the succeeding time interval. The ideal unsafety detector works like a function that takes the sensor readings as input and outputs a real value at any time stamp that indicates how safe/unsafe the system state is. In this paper, this real value is called *danger level* and correspondingly the function called *danger level function*. In other words, the danger level should reveal, either explicitly or implicitly, the probability that the system will collapse from the current state in the following time interval. Measures should be taken to avoid a collapse when the danger level exceeds a threshold.

Usually, the multisensor readings amount to dozens, or even hundreds, of dimensions when the system is complex. A rule-based approach to detect the unsafe states is impractical, and machine learning methods are more desirable. However, the standard supervised learning methods such as classification and regression cannot directly be applied here because these methods require sample input–output pairs from the system to be learned, and the pairs are usually unavailable in our problem due to the labeling issue (see Section III). In our problem, most of the time, we do not have direct measurement of the danger level, except when the system collapses or successfully ends.

To avoid the labeling issue, we model the danger level as the expected future reward. A large value of negative expected future reward indicates a highly dangerous state. At some specific points, the reward can objectively be assigned to the system. For example, the penalty (negative reward) assigned to a system crash can be decided based on its economic loss. After the reward is suitably defined for the system, we propose to use the *temporal difference (TD)* learning [2] to learn the danger level function. Roughly speaking, TD learning aims at estimating the expected future reward of the system, given the current and historical sensor readings. It was originally applied to the reinforcement learning [2] to estimate the so-called value function given current and historical observations. Although

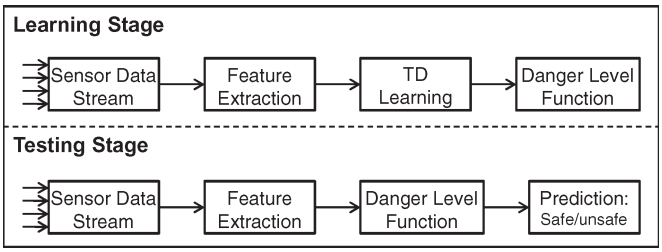


Fig. 1. Block diagram of our framework.

our task differs from the reinforcement learning in that we do not need to learn the optimal policy, they share the same problem of estimating expected future reward.

To demonstrate how much our approach can gain from avoiding the labeling issue, we also manually (and subjectively) label the training data. Then, classifiers and regressors trained on this data set serve as baselines. We choose the classifiers/regressors as baselines because both these and our TD learning approaches try to learn a direct mapping from the observations to the danger level (i.e., the danger level function). The essential difference is that the classification/regression approaches learn the relationship from the labeled input–output pairs, while the TD learning estimates the parameters by propagating the penalties/rewards at ends to the entire feature space so that it requires no labeled input–output pairs. Therefore, comparing with these baselines can reflect the benefit of avoiding the labeling issue. Experimental results show that our approach achieves a significant improvement over the baselines in the application of predicting driving safety.

Fig. 1 illustrates the block diagram of our framework. In the learning stage, streams of sensor readings are captured and stored. Instead of using the original sensor readings for training, we create a compact and informative feature vector from each short time sliding window (see Section IV). The danger-level function is trained on the new feature space using the TD learning algorithm, which is expected to avoid the labeling issue. In the testing stage, feature vectors are extracted in the same way as in the learning stage and are then fed into the danger-level function that outputs a danger level for each feature vector. Unsafe system states are identified by thresholding the danger level.

To demonstrate the effectiveness of our framework, we apply it to the real application of predicting driving safety. In this application, some sensors are installed on a vehicle to collect state parameters of the vehicle, such as braking frequency/strength, throttle frequency/strength, wheel turning angle, and lane position. Some sensors are attached to the human body to collect physiological parameters of the driver (such as electrocardiogram, skin temperature, and skin conductance). Cameras can also be configured right before the driver to capture biobehavioral information, such as percent eye closure (PERCLOS) and facial expression. The in-vehicle information system monitors the driving state by analyzing these sensor readings and alerts the driver to avoid crashes when the danger level exceeds a fixed threshold. We developed a live prototype system of driving safety prediction based on the proposed framework that further validates the robustness and efficiency of our approach.

Our approach is a general framework for monitoring unsafe system states. It has great potential to be applied to a wide range of applications that require real-time monitoring, such as predicting driving safety, that is validated by the experiments in this paper. Validation of our framework on some other applications will be part of our future work. The contributions of this paper are summarized as follows.

1) We propose a general framework for monitoring unsafe system states. In this framework, a danger-level function is learned from training data. The function outputs in real time an indicator of the system states when new senor readings are arriving. This framework is scalable to a large number of sensors.
2) We identify the labeling issue that exists in the domain of monitoring unsafe system states. We formulate the problem as a function approximation by TD learning so that the labeling issue is avoided. To the best of our knowledge, we are the first to apply TD learning to this class of problems.
3) A large real data set was collected to validate our approach. Our live prototype system of driving safety further demonstrates the robustness and efficiency of our approach.

This paper is organized as follows. Related work is given in Section II. Section III describes the labeling issue. Section IV focuses on feature extraction. The TD learning framework is specified in Section V. This framework is applied to the real application of driving safety in Section VI. Section VII concludes this paper.

II. Related Work

The proposed work is mainly related to the following three research areas: 1) anomaly detection; 2) data stream processing; and 3) prediction of driving safety. Here, we briefly review the literature in these topics one by one, although some literature may pertain to two or more of them.

A. Anomaly Detection

The proposed work is a special case of anomaly detection. Anomaly detection has the potential to impact a wide range of important real-world applications, e.g., security, finance, public health, medicine, biology, environmental science, manufacturing, astrophysics, business, and economics. Accordingly, there is a huge volume of literature that pertains to all of the above areas. Here, we summarize only three categories of anomaly detection methods that are more or less related to our approach.

Rule-based methods check the current and/or past sensor readings against a set of predefined rules. A violation or a combination of violations of the rule(s) is regarded as an anomaly. Wong *et al.* [31] used a complicated rule-based approach that characterizes each anomalous pattern with a rule. The significance of each rule is carefully evaluated using *Fisher's exact test* and a randomization test. Rule-based methods may work well on simple scenarios, but it is impractical to predefine rules for complex applications, such as the prediction of driving safety, in this paper.

A large amount of literature falls into the category of statistical methods [7], [10], [26], [30]. Many of them learn a generative model from the training data that are labeled as one of the following two classes: 1) normal or 2) anomalous [7], [10], [26]. The Bayesian approach is used to estimate the probability of each class for new incoming data points, and anomaly is alarmed if the probability of anomalous class exceeds a predefined threshold. Sometimes, only the generative model of the normal data is learned because the anomalous data are very rare [10]. In this case, the Bayesian approach can still be used to find the outliers. Some statistical methods train two-category classifiers using a support vector machine [30] or a neural network [6] that classifies the incoming data as normal or anomalous. The above statistical methods require huge labeled data for training. However, labels are unavailable or not objective in our problem due to the labeling issue (see Section III). Another drawback of the above methods is that they overlook the long-term temporal correlations of the sensor data. These correlations are explored in this paper to predict the anomaly as early as possible (see Section IV).

The third category of methods involves signal processing techniques [16], [23]. For instance, Jiang *et al.* [16] proposed an approach to automatically model and search relationships between the flow intensities measured at various points across the system. If the modeled relationships hold all the time, they are regarded as invariants of the underlying system; Lotze *et al.* [23] investigated the use of wavelets for detecting disease outbreaks in temporal syndromic data.

B. Data Stream Processing

The data stream model is motivated by the emerging applications of massive data sets, such as customer click streams, retail transactions, telephone records, multimedia data, as well as the multisensor data streams discussed in this paper [8].

A typical data stream model is a clustering model. The stream data arrive at such a high rate that it is impossible to fit them in the memory or scan them multiple times, as conventional clustering does. Hence, it demands for efficient incremental or online methods that cluster massive data by using limited memory and by one-pass scanning. A large portion of such algorithms dynamically updates the cluster centers [9], medoids [8], or a hierarchical tree [5] when new data points are inserted, using limited memory. More results on data stream can be found in [22], [24], [25], and [27]. However, in our applications, data points are extremely unbalanced between the "safe" and "unsafe" clusters because computers and vehicles normally do not crash very often, and therefore, the unsafe states are very rare. This poses extreme difficulties for currently available clustering algorithms. Another drawback of these algorithms is that as unsupervised learning algorithms, they do not utilize information in the training data (such as historic crashes), while this information is helpful for future prediction in supervised methods.

Another set of data stream models are time series models that are often specifically designed to forecast future events based on known past events: to forecast future data points before they are measured. When the prediction greatly deviates from the measurement, a possible anomaly may occur [4]. Time series models can have many forms and represent different stochastic processes. Three broad classes of practical importance are given as follows: 1) the *autoregressive* model; 2) the *integrated* model; and 3) the *moving average* model [3]. The three classes linearly depend on the previous data points. Nonlinear dependence is also of interest because of the possibility of producing a chaotic time series and some advantages over linear models in practice. The time series models, both linear and nonlinear, try to analytically model the internal structure (such as autocorrelation, trend, or seasonal variation) of a time series. However, for complex applications such as the prediction of driving safety, the internal structures of multisensor streams cannot be analytically modeled in either linear or nonlinear form because drivers' behaviors are subjective and unpredictable and because driving environments have large variations (such as moving pedestrians, traffic lights, and passing-by vehicles). Like the clustering approaches, times series models also have the drawback of not utilizing prior information on unsafe states (e.g., historic crashes) because they are usually trained on normal data.

A third set of approaches, i.e., the data stream classification/regression approaches, can avoid the above limitations associated with the clustering and time series models. First, classifiers and regressors are learned by supervised learning so that they can utilize the prior information. Second, by learning direct mappings from the observations to the system danger level, they avoid the tough problem of modeling internal structures of the stream data, which may limit the time series models in our applications. However, the classification/regression approaches suffer from the labeling issue, i.e., they have to be trained on manually (and subjectively) labeled data. Our approach shares the advantages (utilizing prior information and learning direct mappings) with the classification/regression approaches, while avoiding the labeling issue through TD learning.

C. Prediction of Driving Safety

Another set of literature that is closely related to our work pertains to monitoring stream data for the prediction of driving safety. It is a special case of the aforementioned research topics. We specifically review it because we use the application of the prediction of driving safety to validate our framework.

This set of literature often uses traditional classification [32], clustering [19], or data-mining techniques [20], [21] to detect unsafe driving patterns.

Zhou *et al.* [32] trained a graphical model, namely, the conditional random field (CRF) on the multisensor stream data, which is expected to detect unsafe driving patterns. The CRF model has the advantage of representing the variable dependences in an undirected graph. Ubiquitous data mining, which fuses and analyzes different types of information from crash data and physiological sensors to diagnose driving risks in real time, is used in [21]. Kargupta *et al.* [19] presented a mobile and distributed data stream mining system that allows real-time vehicle-health monitoring and driver characterization. They rely on an incremental clustering algorithm to identify the safe operating regimes. For the task of monitoring vehicle

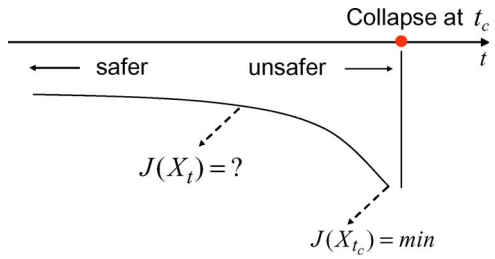


Fig. 2. Labeling issue. Safeness is uncertain at any time, except at the ends of driving courses. A subjective assumption can be made that it becomes more unsafe when approaching a collapse point.

data streams that requires frequent computation of correlation matrices in a resource-constrained environment, Kargupta *et al.* [20] further proposed a fast algorithm that uses a divide-and-conquer strategy to reduce computations. Our framework is also applied to driving safety prediction, but it differs from the above approaches. First, we estimate a danger-level function that enables real-time and continuous monitoring of system states. Second, the use of TD learning naturally avoids the labeling issue that a huge number of manually labeled training sets is needed in most of other learning-based approaches [32].

III. Labeling Issue

Let X_t be the sensor readings at time t and $X_{0:t}$ be all the sensor readings from time 0 to t. Let J be the danger level function and $J(X_{0:t})$ be the danger level given all past sensor readings. Our goal is to learn the danger level function J from the training data (sensor data). An intuitive idea is to parameterize the danger level function and then estimate the parameters by linear or nonlinear regression/clasification, i.e., by supervised learning. However, supervised learning requires manually labeled training data, i.e., assigning a danger level to the sensor readings at any time t. Here, a problem arises: Can a human being do it objectively? The answer is **no**. It is hard to judge if speeding is more or less unsafe than weaving. Furthermore, it is impractical for human to claim that the computer network is in a safe or unsafe state by just looking at the basic parameters such as CPU usage, network traffic, and memory usage. Therefore, the training data are labeled manually and subjectively in most of the previous work. For instance, Healey and Picard [14] obtained the driving stress labels by asking the drivers to fill out the subjective rating questionnaires immediately after the driving course was ended. In another paper on driving safety prediction [1], the drivers simulated some drowsy behaviors according to the physiology study of the U.S. Department of Transportation [13]. In these studies, the results are doomed to be subjective since the learning relies on the subjectively labeled training data.

By intuition, it is natural to assume that a driving state becomes more unsafe when it is closer to a collapse point and that the state is safer when it is farther away from the collapse point (see Fig. 2). This constraint itself gives cues of how to manually (subjectively) label the training data. There are two kinds of labels that result in two categories of supervised learning approaches: 1) The *hard label* has discrete values $\{1, -1\}$, where 1 means a safe state and -1 an unsafe state. Intuitively, we can label all the states t_u seconds before a collapse point as "unsafe" (-1) and any other states as "safe." Then, a two-category classifier can be trained on the hard label data. It results in a danger level function that outputs discrete values. 2) The *soft label* takes continuous values that is consistent with Fig. 2, i.e., the bigger the value is, the safer the system state becomes. The soft label suggests a linear or nonlinear regression of the danger-level function. The parameters of the regressor are estimated from the soft label data.

Instead of predefining some subjective rules, it is more desirable to learn the above constraints from the training data. We assuredly know the following: 1) A collapse point is definitely unsafe; 2) it is safe when the system is ended successfully; and 3) the safeness is uncertain at any other points. Hence, it is reasonable to assign an extremum penalty (without loss of generality, it is negative and minimum in this paper) to a collapse point and to assign a positive reward to a successful system ending. The penalty and reward can be propagated to any other states according to the learned constraints, which, in turn, forms the danger-level function. Here, TD learning is used to accomplish the propagation and to avoid the labeling issue. TD learning aims at estimating the expected future reward of the system given the current and historical sensor readings, while the above soft label can be regarded as an empirical expectation of the future reward. In Section V-C, we argue that TD learning provides a less noisy and, thus, better supervision than the empirical expectation provided by the soft label.

In the experiments, we demonstrate how much TD learning can gain from avoiding the labeling issue. More specifically, we train a two-category classifier on the hard label data using logistic regression [12] and a linear regressor on the soft label data. Both the classifier and the regressor are used as baselines and are compared with our TD learning approach. The experiments of driving safety prediction in Section VI show that our approach outperforms the classification/regression approaches. We did not use the data stream clustering approaches or time series models as baselines because they are not appropriate for our applications (see discussions in Section II-B).

IV. Feature Extraction

Theoretically, the danger-level function should depend on all the past and current sensor data. However, it is impractical and inconvenient to examine the entire sensor data stream that accumulates over time. In practice, it can empirically be assumed that the danger-level function depends only on the sensor readings in the past s seconds, i.e., $J(X_{t-s:t}) \approx J(X_{0:t})$. In other words, J predicts danger level by analyzing the sensor data in a time window with the length of s seconds. Usually, J is expected to make a prediction after each time interval t_0. This can be done by sliding the time window with step size t_0.

Instead of using the original sensor data, we create a new feature vector for each time window using a transformation $\widetilde{X}_t = T(X_{t-s:t})$, where $\widetilde{X}_t$ is the new feature vector at time t. The transformation T reduces the dimension of the raw data and summarizes the information meaningful to "unsafety" detection. Usually, T is empirically determined but should extract as much information as possible. In the application of

the prediction of driving safety, T extracts the *mean*, *max*, *min*, and *variance* from each dimension of the sensor readings and the *weave* from the dimensions of lane position and steering wheel movement. Here, *weave* is the frequency of vehicle oscillations on the lane, which reveals the driver's skill, fatigue, and drowsiness to some extent. These measurements are stacked together to form the new feature vector $\widetilde{X}_t$.

Although the danger level function is basically determined by the new feature vector $\widetilde{X}_t$, a more precise approach should consider the variables that have effects much longer than the window size s. These variables are called *long-term variables*. They can be either hidden states of the system or explicit measurements. For example, in the driving safety prediction application, the driver's emotional state is a hidden variable of the entire system, which affects driving performance for hours/days [17]; stop-sign violation is another long-term variable that reveals the driver's carelessness over a relatively long period. In the experiment, we consider the long-term variables of individual mistakes (such as speeding, stop-sign violation, and red-light ticket). Each individual mistake gives a negative penalty that vanishes exponentially over time. All of the penalties are added together to form a long-term variable that becomes a dimension of the feature vector $\widetilde{X}_t$. In other words, the danger-level function will depend not only on the feature vector extracted from the time window but on the long-term variable as well. From now on, $\widetilde{X}_t$ is simply represented as X_t for simplicity.

V. Temporal Difference Learning

As we mentioned in Section I, TD learning was originally used in reinforcement learning, and its primary goal is to provide effective suboptimal solutions to complex problems of planning and sequential decision making under uncertainty. TD learning draws on the theories of function approximation, dynamic programming, and iterative optimization [2]. More importantly, it combines two disciplines of dynamic programming and supervised learning to successfully solve the problems that neither discipline can address individually [11]. This paper utilizes this property to avoid the labeling issue and to estimate the danger-level function by iterative optimization.

We first give an intuitive description of estimating the danger level function J by TD learning. In the new feature space, each individual system running course corresponds to a trajectory that is ended with either a collapse or a success. As shown in Fig. 3, the red dotted curves correspond to collapse trajectories, and the green solid curves correspond to success trajectories. The collapse points can be viewed as sink points (red points in Fig. 3). A moving point in the feature space will be either absorbed into a sink point or stopped by an external force, with the former leading to a collapse trajectory and the latter leading to a success trajectory. The training data correspond to sparse trajectories in the feature space. As discussed in Section III, the danger-level function can take values of a minimum (negative) penalty at the end of each collapse trajectory and a positive reward at the end of each success trajectory, whereas the values at any other points in the feature space are uncertain. The TD learning propagates the penalties/rewards in the feature space so that the danger-level function has values in the entire feature space.

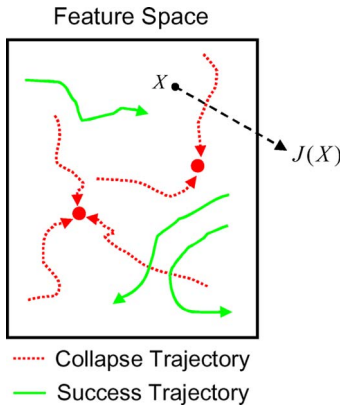


Fig. 3. Trajectories and sink points (red dots) in feature space. Each segment of a system running course corresponds to a trajectory.

A. Estimation by TD(λ)

Here, the learning involves the following two main choices [2]: 1) the choice of an approximation architecture to represent the danger-level function and 2) the choice of a training algorithm, i.e., a method to update the parameters of the architecture. For simple cases of function approximation, the function can simply be represented by a lookup table, and a training algorithm approximates the function by iteratively updating the table [11], [18]. However, in this paper, the danger level function takes values on a continuous and high-dimensional feature space, and it is impractical to represent it using a lookup table. Therefore, we focus on a suboptimal method that replaces the optimal danger-level function $J(X_t)$ with a suitable approximation $J(X_t, r)$, where r is a vector of parameters. $J(X_t, r)$ can be a linear or nonlinear function of r or even a neural network with r as the weights. In the following, we give a detailed description of the algorithm to learn the parameters r.

The danger-level function $J(X_t)$ implicitly gives the maximum probability that the system will collapse from the current state X_t. Assuming that the system transition from state X_t to X_{t+1} incurs a danger-level change $g(X_t, X_{t+1})$, $J(X_t)$ should satisfy the following *Bellman's equation* [2], [11], [18]:

$$J(X_t) = \min_{X_{t+1}} \left(g(X_t, X_{t+1}) + J(X_{t+1}) \right). \tag{1}$$

Suppose there are N trajectories in the training data $(X_1^{(n)}, X_2^{(n)}, \ldots, X_{T_n}^{(n)})$, $n = 1, 2, \ldots, N$, where T_n is the length of the nth trajectory. Denote the **real** danger level at $X_t^{(n)}$ of the nth trajectory as $D(X_t^{(n)})$. Note that $J(X_t, r)$ is an approximation of $D(X_t)$. According to (1), we have

$$D\left(X_t^{(n)}\right) = \sum_{s=t}^{T_n - 1} g\left(X_s^{(n)}, X_{s+1}^{(n)}\right) + M^{(n)} \tag{2}$$

where $M^{(n)}$ is the penalty/reward at the end of the nth trajectory. We consider the linear/nonlinear function of the form $J(X_t, r)$ that approximates $D(X_t)$, where r is a parameter

vector. In our problem, r can be estimated by solving the least-square optimization problem [2] as follows:

$$\min_r \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(J\left(X_t^{(n)}, r\right) - D\left(X_t^{(n)}\right)\right)^2. \quad (3)$$

The above least-squares problem can be solved by an incremental gradient method [2]. For convenience, only one trajectory is considered for each iteration, i.e., the parameter vector r is updated iteratively by

$$\Delta r = -\gamma \sum_{t=1}^{T-1} \nabla_r J(X_t, r) \left(J(X_t, r) - D(X_t)\right) \quad (4)$$

where $(X_1, X_2, \ldots, X_T)$ is a trajectory, $\nabla_r J(X_t, r)$ is the partial differentiation with respect to r, and γ is a step size. By inserting (2) into (4) and after a few manipulations, Δr can be rewritten as

$$\Delta r = \gamma \sum_{t=1}^{T-1} \nabla_r J(X_t, r) \sum_{s=t}^{T-1} d_s \quad (5)$$

where the quantities d_s are defined by

$$d_s = g(X_s, X_{s+1}) + J(X_{s+1}, r) - J(X_s, r). \quad (6)$$

Here, $J(X_T, r) = M$ is fixed as the penalty/reward at the end of that trajectory. The term d_s is called *TD* [2]. The key idea of TD learning is that $g(X_s, X_{s+1}) + J(X_{s+1}, r)$ is a sample of the value $J(X_s, r)$, and it is more likely to be correct because it is closer to $J(X_T, r)$.

The TD provides an indication as to whether or how much the estimate r should be increased or decreased. Usually, d_s in (5) is multiplied by a weight λ^{s-t}, $0 \leq \lambda \leq 1$ to decrease the influence of farther TD on $\nabla_r J(X_t, r)$, i.e.,

$$\Delta r = \gamma \sum_{t=1}^{T-1} \nabla_r J(X_t, r) \sum_{s=t}^{T-1} d_s \lambda^{s-t}. \quad (7)$$

The above equation provides a family of algorithms, one for each choice of λ, is known as TD(λ), and it is an offline version. An online version of updating that corresponds to the transition (X_s, X_{s+1}) is

$$\Delta r = \gamma d_s \sum_{t=1}^{s} \lambda^{s-t} \nabla_r J(X_t, r), \qquad s = 1, \ldots, T-1. \quad (8)$$

We may add a discount factor α, $0 < \alpha \leq 1$, to the TD, i.e.,

$$d_s = g(X_s, X_{s+1}) + \alpha J(X_{s+1}, r) - J(X_s, r) \quad (9)$$

to weight a near term more heavily than a distant one. Then, (8) can be rewritten as

$$\Delta r = \gamma d_s \sum_{t=1}^{s} (\alpha\lambda)^{s-t} \nabla_r J(X_t, r), \qquad s = 1, \ldots, T-1. \quad (10)$$

The discount factor is required if the length of the trajectory is infinite (or very long); otherwise, the danger level will be infinite (or very large) for some feature points.

B. Implementation of TD(λ)

In this paper, the approximate danger-level function $J(X_t, r)$ has both linear and nonlinear representations. We have

$$\text{Linear}: \quad J(X_t, r) = X_t r \quad (11)$$

$$\text{Nonlinear}: \quad J(X_t, r) = \sum_{i=1}^{S} \alpha_i e^{-\frac{\|X_t - r_i\|^2}{2\sigma^2}} + \beta. \quad (12)$$

In the nonlinear representation, S is the number of kernels, and $r = (\alpha_1, \ldots, \alpha_S, r_1, \ldots, r_S, \beta)$. It is originated from the idea of sink points in Fig. 3, i.e., each r_i corresponds to a sink point, and S represents the number of sink points. The linear representation requires low computational cost and may have good generalization ability when feature dimension is high.

The partial differentiation of $J(X_t, r)$ with respect to r, i.e., $\nabla_r J(X_t, r)$, can easily be computed when $J(X_t, r)$ has the above representations. For example, consider the online TD(λ) with linear representation: $J(X_t, r) = X_t r$. In this case, we have

$$\nabla_r J(X_t, r) = X_t. \quad (13)$$

The TD

$$d_s = \begin{cases} g(X_{T-1}, X_T) + \alpha M - X_{T-1} r, & s = T-1 \\ g(X_s, X_{s+1}) + \alpha X_{s+1} r - X_s r, & 1 \leq s < T-1 \end{cases} \quad (14)$$

where M is the penalty/reward at the end of that trajectory. By substituting (13) and (14) into (10), we can rewrite (10) as

$$\Delta r = \gamma d_s \sum_{t=1}^{s} (\alpha\lambda)^{s-t} X_t, \qquad s = 1, \ldots, T-1. \quad (15)$$

Then, at each iteration, r can be updated by $r \leftarrow r + \Delta r$. However, the parameters γ, α, and λ will be decided according to the specific applications. Other cases with offline TD(λ) and/or nonlinear $J(X_t, r)$ can be similarly implemented.

In this paper, both the offline and online TD(λ) with a discount factor α are used to iteratively update the parameter vector r, with each iteration dealing with one training trajectory. The two versions produce similar performances. Fig. 4 shows a danger-level function trained on two trajectories: One ends with car crash, and the other does not. The crashed curve takes values that are increasingly smaller when approaching the crash point, and the danger level of the noncrash trajectory is basically constant. This result is exactly what we expect.

C. Discussion of the Algorithm

We need to stress the difference between the standard reinforcement learning and the TD learning used in our framework. A standard reinforcement learning needs to learn a *policy* that is a mapping from states to actions. A policy determines which action should be performed in each state [11]. In our

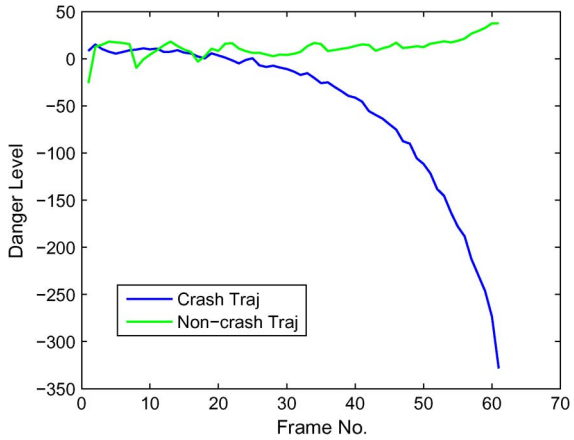


Fig. 4. Danger level function trained on two trajectories.

case, we do not need to learn a policy. A policy is a random sample from the policy space with an unknown fixed sampling distribution. Our goal is to learn a common (average) value function for this policy space. TD learning exactly satisfies our purpose of approximating the value function by propagating the penalties/rewards to the entire feature space.

Convergence of TD(λ) and selection of the value λ is essential in TD learning. TD(λ) combined with function approximation may converge to a different limit for different values of λ, and even the issue of convergence is complex. The only available convergence result refers to the linearly parameterized approximation architecture but requires constraints on the choice of the eligibility coefficients [2]. One constraint is that the step size γ_n should be nonnegative and should satisfy $\sum_{n=0}^{\infty}\gamma_n = \infty$ and $\sum_{n=0}^{\infty}\gamma_n^2 < \infty$, where n means the nth trajectory. A popular choice is to let $\gamma_n = c/(d+n)$ in the iteration on the nth trajectory, where c and d are some positive constants. The convergence issue of nonlinear architecture is far from well studied. Furthermore, there exist cases where TD(λ) may fail to converge. However, there is empirical evidence that for many problems, nonlinear approximation does converge and leads to better performance than the linear approximation when the linear model cannot fit the function well. This statement also holds in our experiments where the nonlinear approximation comes from the idea of sink points.

The motivation for TD(λ) does have strong theoretical basis. It argues that as λ becomes smaller, there is a tendency for the quality of approximation to deteriorate. Nevertheless, empirical evidence shows that for many problems, TD(λ) with $\lambda < 1$ converges faster and achieves better performance than that obtained from TD(1) [29]. If the algorithm for a certain problem is convergent, then the quality of the estimation should improve as the algorithm progresses iteratively. Therefore, we could rely increasingly more on the values of the nearest downstream states as we should use a smaller λ. This suggests an empirical strategy that we start with a large λ, and as learning progresses, we decrease it to zero [2]. TD(1) uses the discounted future reward as supervision for the current time step; hence, it is nearly the same as using a soft label for supervised regression. TD(0) uses the predicted future reward at next time steps as the supervision for the current step. The difference between the two types of supervision is that the predicted future reward is usually a much more smoothed target than the discounted future reward and, hence, can make learning easier.

The algorithm is scalable to a large number of sensors. First, increasing the number of sensors does not affect the approximation architecture of the danger-level function. Second, the number of iterations needed by the learning process probably increases when more sensors are added, but the learning convergence is guaranteed for linear approximation and can usually be controlled for nonlinear approximation by adjusting the parameter λ. Third, for both linear and nonlinear approximations in (11) and (12), the computational cost of prediction is linear with the number of sensors.

In addition, we need to discuss the initialization problem of the learning algorithm. Any random initialization of the parameter vector r satisfies the linearly parameterized approximation constructed by TD(λ) since the convergence is guaranteed as long as some constraints are satisfied. However, the initialization is crucial to the quality of the nonlinear function approximation, and even the convergence may depend on the choice of initial values. In this paper, the nonlinear representation of (12) comes from the idea of sink points so that the intuitive choice of initial r_i can be the cluster centers of the features of collapse trajectories.

VI. Experiments

Our general framework has a wide range of applications for the problems of detecting unsafe system states based on the analysis of multisensor data streams. This paper applies it to the driving safety prediction problem to demonstrate its effectiveness. Currently, the ever-increasing traffic accidents due to a driver's diminished vigilance level poses a serious problem for society. Drivers with a diminished vigilance level suffer from a decline in their perception, recognition, and vehicle-control abilities and, therefore, pose a serious danger on their and other people's lives [1], [15]. According to the National Highway Traffic Safety Administration, each year, there are at least 100 000 crashes that are caused by drowsy drivers, which results in more than 1500 fatalities and 71 000 injuries in the U.S. [28]. Therefore, developing systems that actively monitor the state of both the driver and the car and are able to alert the driver of any unsafe driving is essential for increasing driving safety.

A. Data Collection

Our experimental data were collected through the STISIM simulator that is a product of over three decades of research by Systems Technology, Inc. STISIM is a personal computer-based interactive driving simulator that allows the driver to control all aspects of driving, including the vehicle's speed and steering. The whole system includes a computer running the STISIM software, a projector displaying the driving scenarios, a steering wheel, a brake, and a throttle pedal (see Fig. 5[1]). The driving scenario, including windy weather, slippery road, red light, stop sign, speed limit, pedestrian, policeman, buildings,

[1]The picture comes from the STISIM website.

Fig. 5. Interface of the STISIM driving simulator.

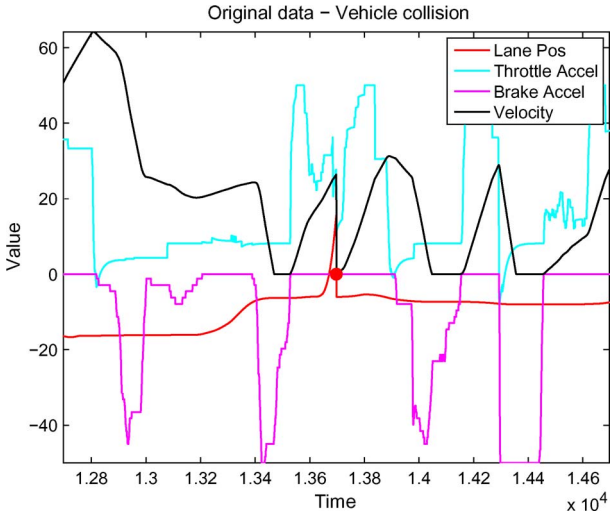


Fig. 6. Segment of data samples of lane position, throttle acceleration, brake acceleration, and velocity.

and so on, was carefully designed to make the simulation as close to reality as possible.

One capability of the STISIM simulator is that it can perform real-time capture of various dynamic parameters related to the driving states, including time stamp, distance, lateral lane position, acceleration due to the throttle, deceleration due to the brake, velocity, steering input counts, throttle input counts, brake input counts, the minimum range between the driver and all vehicles in the driver's direction, the minimum range between the driver and all vehicles opposite the driver's direction, etc., with a resolution as high as 30 samples/s. These sensor readings are the original features. Fig. 6 shows a segment of data samples containing lane position, throttle acceleration, brake acceleration, and velocity, where the red mark indicates a vehicle collision. STISIM also captures the individual mistakes including speeding, stop-sign violation, red-light ticket, etc., which form a long-term variable affecting the driving state (see Section IV).

We hired 36 drivers, each of whom drove for about 20 min, with an average of three crashes per driving session. In estimating the danger-level function, we used the cross-validation approach that at each iteration, one driver is left out for testing, and the remaining drivers are used for training. The average performance of the 36 drivers is compared with the baselines.

B. Setup and Convergence

In our experiments, each trajectory consists of $L = 60$ frames, with each frame corresponding to a feature vector extracted from a time window of 15 s (see Section IV). There is an 80% overlap between each pair of neighboring frames. The ratio of the number of noncrash trajectories to that of crash trajectories is kept to 10 : 1 for each driving session so that at each iteration, we have about 1050 noncrash trajectories and 105 crash trajectories (extracted from any 35 driving sessions) for training and leave about 30 noncrash trajectories and three crash trajectories (extracted from the remaining one driving session) for testing.

The discount factor α is set to $e^{\ln w/L}$, where L is the length of the trajectory, so that the farthest frame is discounted to at most w. In the nth iteration, we choose $\gamma_n = 100/(100 + n^{0.8})$ instead of $\gamma_n = c/(d + n)$ because it makes the updating of r converge more quickly while achieving similar performance. The parameter λ of TD(λ) is set to $\lambda = e^{-n/1000}$. This setting gives relatively large values to λ at the beginning of the learning process so that the penalties/rewards can quickly be propagated backward from the end of the trajectories. As the learning process progresses, λ becomes increasingly smaller, and hence, we can rely increasingly more on the values of the nearest downstream. In the experiments, the parameter r converges to a stable value after about 10 000 iterations if the approximation architecture of $J(X_t, r)$ is linear and about 100 000 iterations if $J(X_t, r)$ is nonlinear. In each iteration, a trajectory is randomly selected from the training set and then put back, i.e., each trajectory can be chosen for multiple times.

C. Evaluation

After the parameter vector r is estimated from the training data, the danger level for any test driving session can be calculated by extracting features and feeding the features to the danger level function. Fig. 7 gives the curves of the danger level function on two test driving sessions. The function is the linearly parameterized approximation. The crash points (indicated by the red marks) and their preceding frames have very small values of danger level. This is consistent with the fact that the smaller the danger level is, the more dangerous the system state will be. There are some other noncrash points that have very small danger level values. Manually checking some of these points by playing back the driving sessions using the STISIM system shows that the drivers at these points usually risk unsafe driving. Since these points are not labeled as "unsafe," it underestimates the performance of our approach to some extent.

We compare our algorithm with two baselines. The first baseline is a two-category classifier using logistic regression. Logistic regression is generally considered as a safe and robust classification method that relies on few assumptions [12]. The training data used for training this classifier are manually labeled with *hard labels*, as described in Section III. The second baseline approximates the danger level function with a general linear regression. In this case, the training data are manually labeled with *soft labels*. Note that the training data for both baselines are labeled manually, and therefore, comparing our approach with the baselines demonstrates the effectiveness of avoiding the labeling issue for our approach.

We specifically define an evaluation metric to measure the performance of both our approach and the baselines. The goal

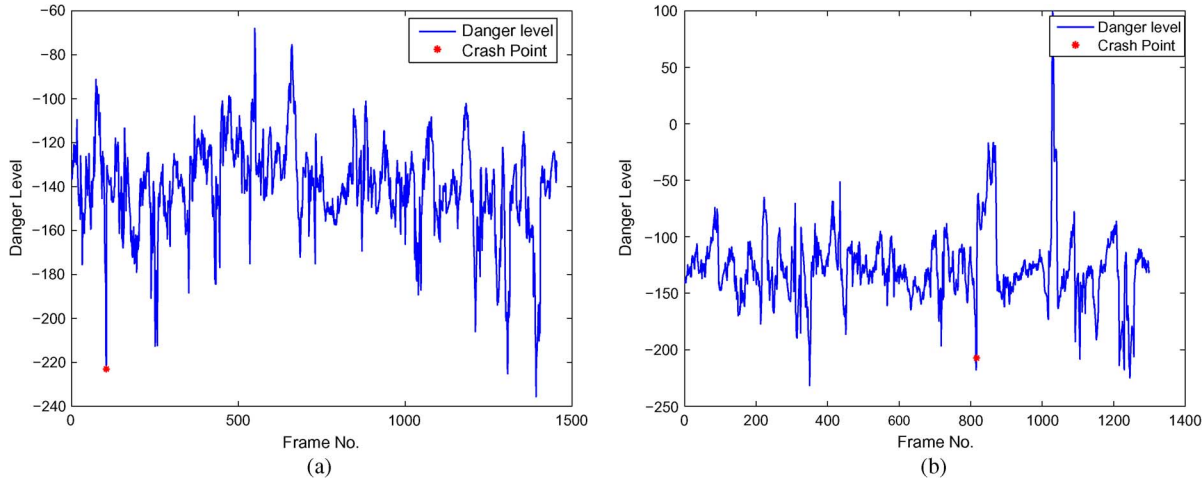


Fig. 7. Curves of the danger level function on two testing driving sessions. The function is a linearly parameterized approximation. The red marks indicate crash points.

of the evaluation metric is not to compare the "intelligent" versus "unintelligent" systems but to first determine whether our approach is an improvement over no crash warning system and the baselines. The intuitive idea of the metric is to measure how much the probability of accidents increases when we only look at the alerted danger time instead of the total time. The higher the increase there is, the more power of detecting unsafe driving there will be. However, we need to exactly specify some important terms to quantitatively define the metric.

As we know, we do not have ground truth for the test data, except at the crash points. To make the evaluation feasible, t_r seconds before each crash is directly defined as *real danger time*, assuming that the real danger level at these time stamps is very high. Let T_r be the total real danger time in the test data and T be the total test driving time. The *claimed danger time*, which is denoted as T_c, is the time when the value of danger level exceeds a threshold. Let T_{cr} be the real danger time that is also claimed as danger time by our approach or the baselines. With these definitions, the *precision* denoted by p can be defined as $p = T_{cr}/T_c$, and $p_r = T_r/T$ is the percentage of the real danger time in the total time. Then, the ratio $R = p/p_r$ expresses how much the probability of accidents increases when we only look at the claimed danger time instead of the total time. The greater the value of R is, the better the performance will be. R is calculated at a certain level of percentage of claimed danger time in total time, which is denoted as $P_c = T_c/T$. Usually, a lower P_c is preferred because a high P_c often means a high rate of false alarm that may result in a "cry wolf" effect where drivers cease to trust automation.

P_c can be lowered down by adjusting the threshold for the danger level. Each threshold corresponds to a value of P_c, and, in turn, P_c determines R. Therefore, we can generate a $P_c{-}R$ curve by adjusting the threshold from the minimum to the maximum. The curve represents the performance of the corresponding approach. Note that the real number output, instead of the binary output, of the first baseline (logistic regression) is used so that we can plot a continuous curve for it. Fig. 8 gives the $P_c{-}R$ curves of both our approach and the baselines. Obviously, the curve for the random guess is always one, which is marked as red. From the figure, the order of performance

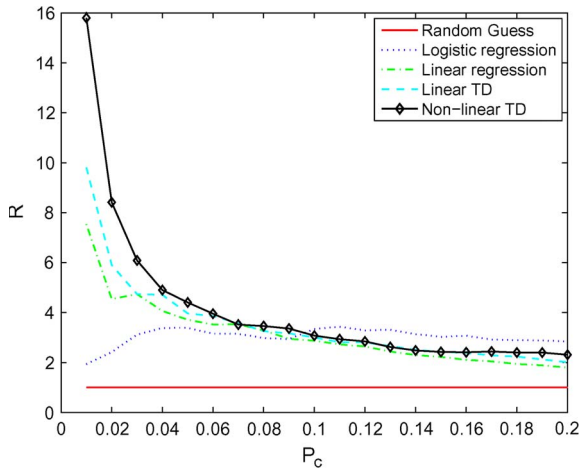
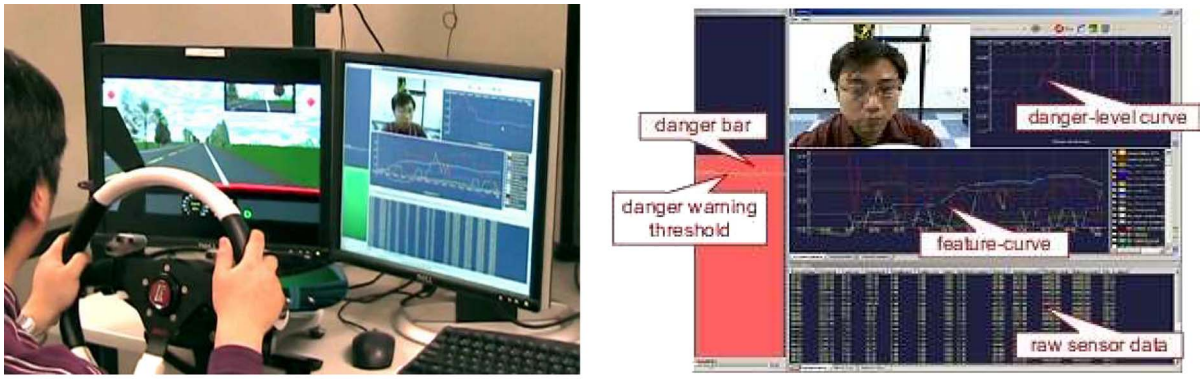


Fig. 8. $P_c{-}R$ curves of our approach and the baselines. R expresses how much the probability of accidents increases when we only look at the claimed danger time. P_c is the percentage of claimed danger time in total time.

at low $P_c(< 5\%)$, from low to high, is described as follows: random guess $<$ logistic classifier $<$ linear regression $<$ linear TD learning $<$ nonlinear TD learning. It shows that our approach, both linear and nonlinear TD learning, outperforms the baselines because our approach can avoid the labeling issue. In particular, the performance of nonlinear TD learning is much better than those of other methods. We compare the performance at low $P_c(< 5\%)$ because it is preferable for the in-vehicle warning system to work at low P_c. If we only look at the claimed danger time, the possibility of accidents is much higher than the average $(R > 1)$, which means that our approach does have much of an ability to detect unsafe driving.

D. Time and Space Complexity

It is worthwhile mentioning the time and space requirement for the application of driving safety prediction. The TD learning takes about several minutes to approximate the linear danger-level function and about 1 h for nonlinear approximation. Fortunately, the danger-level function can be learned offline and once for all. Once the danger-level function is well-trained, testing



(a) (b)

Fig. 9. (a) Live prototype system. The left screen displays the driving scenario, and the right screen displays the interface of the prediction module. (b) Detailed description of the prediction interface.

speed is much faster than real time on a personal computer desktop since the only computation is feature extraction and calculation of (11) or (12). The memory required in the testing stage mainly depends on the length of the time window because the feature is extracted from sensor readings in a window right before the current time stamp. These historical sensor readings should be buffered in memory. In this work, where the window length is 15 s, the buffer memory is only 70 kB. Considering its fast testing speed and low memory requirement, our approach is very suitable for resource-constrained environments.

E. Live Prototype System

As mentioned in Section VI-D, if the danger-level function is well-trained and ready for use, detection of unsafe driving from sensor stream data can be realized in real time. In accordance to this, we built a live prototype system to predict unsafe driving states. The system consists of the following three major modules.

1) **Data acquisition module**: This module captures the vehicle's dynamic parameters in real time. We use an STISIM driving simulator whose interface is shown in Fig. 5. The STISIM software provides programming interfaces that allow access to the sensor stream data.
2) **Feature extraction module**: This module converts the raw sensor readings and individual mistakes into predefined statistical features (see Section IV).
3) **Prediction module**: This module feeds the extracted features to the danger-level function and outputs numerical danger-level scores live. The score triggers the warning interface if it exceeds a predefined threshold learned from training samples.

Fig. 9(a) shows our prototype system where the left screen displays the driving scenario, and the right screen is the user interface of the prediction module. A detailed illustration of this interface is given in Fig. 9(b). On the interface, the sensor data, feature dimensions, and danger level curve are displayed in real time. A bar representation of the danger level is also given on the left side of the interface. The height and color of the bar reflect the magnitude of the danger level. If the danger level exceeds the threshold, a voice warning message is immediately announced to alert the driver. The driver's face image is also captured by a camera installed in the vehicle and displayed on the user interface. At the current stage, we do not utilize the information of facial expression and PERCLOS. However, it can easily be incorporated into our framework, although we leave this for future work.

Note that the danger level function used in this prototype system is trained on the data set described in Section VI-A using nonlinear TD learning. Hence, the corresponding evaluation curve in Fig. 8 basically gives the objective performance of this system. To further evaluate the entire system, 11 participants were invited to freely operate the system. An expert simultaneously observed the driver state, driving state, and danger level and gave an subjective evaluation of this system at the end of each driving session. Results showed that the system can predict main driving risks due to sharp turning, sudden acceleration/deceleration, continuous weaving, approaching objects, etc. In addition, the system is sensitive to the changes in the driver's state (e.g., fatigue) and changes in road conditions (e.g., windy and slippery road), because these changes usually result in changes of driving states. We did not evaluate the user experience of operating a vehicle with this prototype system turned on to alert the driver in case of unsafe driving. Psychological experiments are needed to obtain a statistic of the drivers who benefit from the warning messages given by the system, who do not benefit from the system, and who are annoyed by the warning messages. We leave this for future work.

VII. Conclusion

This paper has proposed a general framework to detect unsafe system states. It uses TD learning to approximate a danger-level function that indicates how safe/unsafe the system is by analyzing the multisensor data that capture the basic parameters of the system. The main challenge to the learning procedure was the labeling issue, i.e., it was nearly impossible to label the training data with an objective danger level, except at the collapse points, where an extremal penalty can be assigned, and at the successful ends, where a reward can be assigned. This paper used TD learning to avoid the labeling issue. The TD learning approximated the danger-level function by propagating the penalty and reward to the entire feature space following some constraints. The approach was applied to, but not limited

to, the application of driving safety, and the experimental results and the live prototype system demonstrate the effectiveness of the approach.

ACKNOWLEDGMENT

The authors would like to thank Toyota InfoTechnology Center Company, Ltd., for providing the driving data and J. Wang for his work on the live prototype system.

REFERENCES

[1] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, "Real-time system for monitoring driver vigilance," *IEEE Trans. Intell. Trans. Syst.*, vol. 7, no. 1, pp. 63–77, Mar. 2006.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[3] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA: Holden-Day, 1990.

[4] P. K. Chan and M. V. Mahoney, "Modeling multiple time series for anomaly detection," in *Proc. 5th IEEE Int. Conf. Data Mining*, 2005, pp. 90–97.

[5] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, 1997, pp. 626–635.

[6] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2000.

[7] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proc. Int. Conf. Mach. Learn.*, 2000, pp. 255–262.

[8] S. Guha, N. Mishra, R. Motwani, and L. OCallaghan, "Clustering data streams," in *Proc. IEEE Annu. Symp. Foundations Comput. Sci.*, 2000, pp. 359–366.

[9] C. Gupta and R. Grossman, "GenIc: A single pass generalized incremental algorithm for clustering," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 137–153.

[10] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 202–209.

[11] M. E. Harmon and S. S. Harmon, *Reinforcement Learning: A Tutorial*, 1996. [Online]. Available: http://citeseer.ist.psu.edu/harmon96reinforcement.html

[12] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, Aug. 2001.

[13] J. Healey and R. Picard, "Smartcar: Detecting driver stress," in *Proc. Int. Conf. Pattern Recog.*, 2000, vol. 4, pp. 218–221.

[14] J. A. Healey and R. W. Picard, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Trans. Intell. Trans. Syst.*, vol. 6, no. 2, pp. 156–166, Jun. 2005.

[15] Q. Ji, Z. Zhu, and P. Lan, "Real-time nonintrusive monitoring and prediction of driver fatigue," *IEEE Trans. Veh. Technol.*, vol. 53, no. 4, pp. 1052–1068, Jul. 2004.

[16] G. Jiang, H. Chen, and K. Yoshihira, "Discovering likely invariants of distributed transaction systems for autonomic system management," in *Proc. IEEE Int. Conf. Autonomic Comput.*, 2006, pp. 1–8.

[17] I.-M. Jonsson and H. Harris, "Emotions, driving and driving performance," Toyota ITC, Palo Alto, CA. Tech. Rep.

[18] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[19] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy, "Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 300–310.

[20] H. Kargupta, V. Puttagunta, M. Klein, and K. Sarkar, "On-board vehicle data stream monitoring using minefleet and fast resource constrained monitoring of correlation matrices," *New Gener. Comput.*, vol. 25, no. 1, pp. 5–32, Nov. 2006.

[21] S. Krishnaswamy, S. W. Loke, A. Rakotonirainy, O. Horovitz, and M. M. Gaber, "Towards situation-awareness and ubiquitous data mining for road safety: Rationale and architecture for a compelling application," in *Proc. Conf. Intell. Veh. Road Infrastructure*, 2005, pp. 16–17.

[22] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani, and N. Mishra, "Streaming-data algorithms for high-quality clustering," in *Proc. Int. Conf. Data Eng.*, 2002, pp. 685–694.

[23] T. Lotze, G. Shmueli, S. Murphy, and H. Burkom, "A wavelet-based anomaly detector for early detection of disease outbreaks," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 1–6.

[24] G. Manku, S. Rajagopalan, and B. G. Lindsay, "Random sampling techniques for space efficient online computation of order statistics of large datasets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1999, pp. 251–262.

[25] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, "Approximate medians and other quantiles in one pass and with limited memory," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1998, pp. 426–435.

[26] T. Menzies and D. Allen, "Bayesian anomaly detection," in *Proc. ICML Workshop Mach. Learn. Algorithms Surveillance Event Detection*, 2006, pp. 1–8.

[27] J. Munro and M. Paterson, "Selection and sorting with limited storage," *Theor. Comput. Sci.*, vol. 12, pp. 315–323, 1980.

[28] D. Royal, "Volume ifindings report; national survey on distracted and driving attitudes and behaviours, 2002," Gallup Org., Washington, DC, Tech. Rep. DOT HS 809 566, 2003.

[29] S. Singh and P. Dayan, "Analytical mean squared error curves for temporal difference learning," *Mach. Learn.*, vol. 32, no. 1, pp. 5–40, Jul. 1998.

[30] T. Singliar and M. Hauskrecht, "Towards a learning traffic incident detection system," in *Proc. ICML Workshop Mach. Learn. Algorithms Surveillance Event Detection*, 2006, pp. 1–8.

[31] W. Wong, A. Moore, G. Cooper, and M. Wagner, "Rule-based anomaly pattern detection for detecting disease outbreaks," in *Proc. Assoc. Adv. Artif. Intell.*, 2002, pp. 217–223.

[32] A. Zhou, F. Cao, W. Qian, and C. Jin, "Tracking clusters in evolving data streams over sliding windows," *Knowl. Inf. Syst.*, vol. 15, no. 2, pp. 181–214, May 2008.

Huazhong Ning received the B.Sc. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2000, the M.Sc. degree in pattern recognition and intelligence systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in electrical engineering from University of Illinois at Urbana-Champaign in 2008.

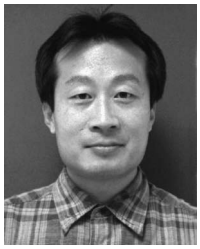
He is currently working as an applied researcher with AdCenter Laboratories, Microsoft Corporation. His current research interests include video/image processing, machine learning, clustering, audio analysis, data mining, etc.

Wei Xu received the B.S. degree from Tsinghua University, Beijing, China, in 1998 and the M.S. degree from Carnegie Mellon University (CMU), Pittsburgh, PA, in 2000.

From 1998 to 2001, he was a Research Assistant with the Language Technology Institute, CMU. In 2001, he joined NEC Laboratories America, Inc., Cupertino, CA, working on intelligent video analysis. His research interests include computer vision, image and video understanding, machine learning, and data mining.

Yue Zhou received the B.Sc. degree in computer science from Tsinghua University, Beijing, China, in 2000 and the Ph.D. degree in electrical engineering from University of Illinois at Urbana-Champaign in 2008.

He is currently with Microsoft Corporation, Redmond, WA. His current research interests include video/image retrieval, machine learning, data mining, etc.



Yihong Gong received the B.S., M.S., and Ph.D. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1987, 1989, and 1992, respectively.

He then joined the Nanyang Technological University, Singapore, where he was an Assistant Professor with the School of Electrical and Electronic Engineering for four years. From 1996 to 1998, he was with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, as a Project Scientist. He was a Principal Investigator for both the Informedia Digital Video Library project and the Experience-On-Demand project, which was a multimillion-dollar project funded by the National Science Foundation, the Defense Advanced Research Projects Agency, the National Aeronautics and Space Administration, and other government agencies. In 1999, he joined NEC Laboratories America, Inc., Cupertino, CA, where he has been leading the Multimedia Processing Group. In 2006, he became the Site Manager, leading the Cupertino branch of the laboratories. His research interests include multimedia content analysis and machine-learning applications. Major research achievements from his group include news video summarization, sports highlight detection, data clustering, and SmartCatch video surveillance that has led to a successful spin-off.

Thomas S. Huang (S'61–M'63–SM'76–F'79–LF'01) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, and the M.S. and D.Sc. degrees in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge.

He was with the Faculty of the Department of Electrical Engineering, MIT, from 1963 to 1973 and with the School of Electrical Engineering, Purdue University, West Lafayette, IN, as the Director of its Laboratory for Information and Signal Processing from 1973 to 1980. In 1980, he joined the University of Illinois, Urbana, where he is currently the William L. Everitt Distinguished Professor of electrical and computer engineering, a Research Professor of the Coordinated Science Laboratory, and the Head of the Image Formation and Processing Group and a Cochair of Human Computer Intelligent Interaction research with the Beckman Institute for Advanced Science and Technology. He is the author of 20 books and over 500 papers on network theory, digital filtering, image processing, and computer vision. His research interests include the broad area of information technology, particularly the transmission and processing of multidimensional signals.