

Driving Safety Monitoring Using Semisupervised Learning on Time Series Data

Jinjun Wang, Shenghuo Zhu, and Yihong Gong, *Senior Member, IEEE*

Abstract—This paper introduces a dangerous-driving warning system that uses statistical modeling to predict driving risks. The major challenge of the research is how to discover the safe/dangerous driving patterns from a sparsely labeled training data set. This paper proposes a semisupervised learning method to utilize both the labeled and the unlabeled data, as well as their interdependence to build a proper danger-level function. In addition, the learned function adopts a continuous parametric form, which is more suitable in modeling the continuous safe/dangerous-driving state transitions in a practical dangerous-driving warning system. Our comprehensive experimental evaluations reveal that, in comparison with driving danger-level estimation using classification-based methods, such as the hidden Markov model (HMM) or the conditional random field algorithm, the proposed method requires less training time and achieved higher prediction accuracy.

Index Terms—Driving safety monitoring, functional safety, semisupervised learning.

I. INTRODUCTION

THE AVAILABILITY of onboard electronics and in-vehicle information systems has accelerated the development of more intelligent vehicles. One possibility is to automatically monitor a driver's performance to prevent potential risks. Although protocols to measure a driver's workload have been developed by both government agencies and the automobile industry, they have been criticized as being too costly and difficult to obtain. In addition, existing uniform heuristics for driving risk prevention does not account for changes in individual driving environments [1]. Hence, the technology for understanding a driver's frustrations to prevent potential driving risks has been listed by many international automobile companies as one of the key research areas for realizing intelligent transportation systems.

In the past decade, there have been much research that has attempted to recognize the driving danger level by detecting a driver's vigilance level, using physiological and biobehavioral signals such as heart rate, blood volume pulse, respiration, etc. [2]–[4]. However, acquiring physiological data is intrusive because electrodes or sensors must be attached to the driver's body. To develop nonintrusive driving safety monitoring systems, two sets of features have been explored. The first set is

to extract visual features that are correlated to a driver's fatigue state [5]–[7] using cameras or infrared light-emitting diodes [8]. Unfortunately, these visual features cannot always be acquired accurately or reliably due to the large human/environment variations and the immaturity of current computer vision techniques. The other set of nonintrusive features relates to a vehicle's dynamic parameters, such as lateral positions, steering wheel movements, accelerations/decelerations [9], and environmental condition parameters, such as road-adhesion coefficient [10] and magnetic lane-marker detection [11]. Since these parameters can be extracted using hidden sensors and are strong indicators of a driver's state, they are promising features and have big potential for developing driving safety monitoring systems [2].

The task of detecting driving risks can be modeled as an anomaly detection problem. The most straightforward way is to use rule-based approaches. However, defining a comprehensive set of rules to cover all kinds of driving conditions/risks is an extremely difficult task. Therefore, many researchers applied statistical modeling methods, such as Fisher's linear discriminant analysis [3], [12], support vector machines [13], Bayesian networks [14], reinforcement learning with temporal Difference criteria [9], etc., to learn rules (or knowledge) for driving anomaly detections from training samples. Many of these reported works used classification-based methods to identify safe and dangerous states. However, classification-based methods have several inherent limitations for the problem. First, it is very difficult to collect sufficient samples for dangerous driving states. Second, there are no commonly agreed-upon criteria to draw a clear boundary between safe and dangerous driving states. For instance, should a dangerous/safe label be assigned to states of near misses or states with frequent corrective actions? Third, the entire driving state space is indeed a continuous space where states corresponding to driving accidents scatter in many places, and any transitions between safe and dangerous states occur rather continuously than discretely. Hence, classification-based methods are not an ideal model for the driving anomaly detection problem. To cope with these limitations, the reinforcement learning algorithm using temporal difference was examined in [9]. Since it is based on regression, very impressive results were obtained. However, the proposed method requires lengthy training time.

In this paper, we propose a novel semisupervised learning method to build a model to measure the danger level of each driving state using a vehicle's dynamic parameters. We assume that, in the entire driving state space, states that correspond to driving accidents are the most dangerous states and should be assigned the highest danger level, whereas states that are far

Manuscript received May 9, 2008; revised April 13, 2009 and April 24, 2010; accepted May 3, 2010. Date of publication June 1, 2010; date of current version September 3, 2010. The Associate Editor for this paper was N. B. Sarter.

The authors are with NEC Laboratories America, Inc., Cupertino, CA 95014 USA (e-mail: jjwang@sv.nec-labs.com; zsh@sv.nec-labs.com; ygong@sv.nec-labs.com).

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.2010.2050200

away from any accident states are the safest states and should be assigned the lowest danger level. We also assume that the danger level of any other state should be proportional to their spatiotemporal distances to the safe/accident states. The proposed learning method strives to estimate a danger-level model from time-series training samples that satisfies the aforementioned three assumptions. It requires labeling only the accident and the safe¹ states in the training data and learns an optimal model with respect to predefined constraints. Compared with the classification-based approaches, our method has several advantages. First, it does not need to label those nonaccident dangerous states, which are the most difficult and subjective part of the labeling task. Second, the model's performance will not be seriously affected by the imbalance between positive and negative training samples. Third, the model has explicit and smooth parametric forms so that it can be estimated with efficient algorithms. Fourth, the proposed method is more appropriate for modeling the continuous driving state space with very sparsely distributed accident states. Our comprehensive experimental evaluations reveal that the driving anomaly detection accuracy of the proposed method surpasses classification-based approaches by a large margin under all circumstances.

The rest of this paper is organized as follows: Section II describes the data-acquisition and feature-extraction setup in our work. Section III explains the problem in more detail. Section IV describes the mathematical model of the problem and the proposed learning methods. Section V briefly introduces the classification-based approach for comparison. Section VI lists the experimental results. Section VII draws conclusions and discusses some future work.

II. DATA ACQUISITION

In the current work, the set of features used is the vehicle's dynamic parameters, such as speed, accelerating/braking, steering, lateral lane position, and physical data from the vehicle. In our study, the vehicle dynamic parameters are collected from a driving simulator called "STISIM" by Systems Technology, Inc. STISIM is a PC-based interactive driving simulator that allows the user to control almost all aspects of driving such as throttling, breaking, and steering (see Fig. 1). The driving scenario, including weather, road condition, traffic light/sign, pedestrian, buildings, and so on, was carefully designed by professionals to make the simulation as close to reality as possible. During simulation, "STISIM" outputs 38 different dynamic parameters simultaneously. Table I lists the nine parameters used in our system. These parameters are selected because they are suggested to be highly correlated to driving safety performance and can be extracted reliably in real vehicles.

The raw vehicle dynamic parameter data stream is converted into a sequence of feature vectors for modeling. A sliding window with length T_w and step size T_s is applied on the raw data stream, and the raw data samples covered by the sliding window are converted into one single feature vector \mathbf{x} . We denote the obtained feature sequence by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_T]$. The process to convert the raw data in each

¹Safe states will be automatically discovered based on their temporal distances to the accident states.



Fig. 1. Driving simulator ("STISIM").

TABLE I
VEHICLE DYNAMIC PARAMETERS F_3

ID	Description
p_1	Driver's lateral lane position (meter).
p_2	Longitudinal acceleration due to the throttle (m/s^2)
p_3	Longitudinal acceleration due to the brakes (m/s^2)
p_4	Driver's longitudinal velocity (m/s)
p_5	Steering angle
p_6	Throttle depression fraction
p_7	Braking depression fraction
p_8	Minimum range (meter) between the driver and all vehicles in the driver's direction
p_9	Minimum range (meter) between the driver and all vehicles in the driver's opposite direction

sliding window into a feature vector \mathbf{x}_i , $i = 1, \dots, T$ is as follows: Denoting each R^9 raw data sample by $f = [p_1, p_2, \dots, p_9]^T$, we first calculate its first-order forward difference $\Delta f = f_{t+1} - f_t$, square f^2 , and first-order forward difference of square Δf^2 . Then, we concatenate them together to obtain $\mathbf{f} = [f, \Delta f, f^2, \Delta f^2]^T$, which is a column vector in R^{36} . Next, denoting those \mathbf{f} 's covered by sliding window T_w by $\mathbf{F}_i = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_t]$ [see Fig. 2(b)], we calculate the maximal, minimal, average, and standard deviation of each row of \mathbf{F}_i , as well as the zero-crossing-rate (ZCR) of p_1 , p_5 , and their square values. Finally, these statistics are concatenated together to form an R^{148} feature vector $\mathbf{x}_i = [\text{ZCR}_i(p_1), \text{ZCR}_i(p_5), \text{ZCR}_i^2(p_1), \text{ZCR}_i^2(p_5), \max(\mathbf{F}_i), \min(\mathbf{F}_i), \text{mean}(\mathbf{F}_i), \text{std}(\mathbf{F}_i)]^T$ [see Fig. 2(c)]. With the \mathbf{x}_i sequence, the next system module uses semisupervised learning algorithms to mine specific patterns for driving risk prediction.

III. PROBLEM FORMATION

As previously explained in this paper, the training and testing data in our model are the multidimensional data streams generated by "STISIM" and collected from multiple driving sessions with different drivers. These multidimensional data streams contain various driving situations, but the only labels we have are the time instances of each crash accident.

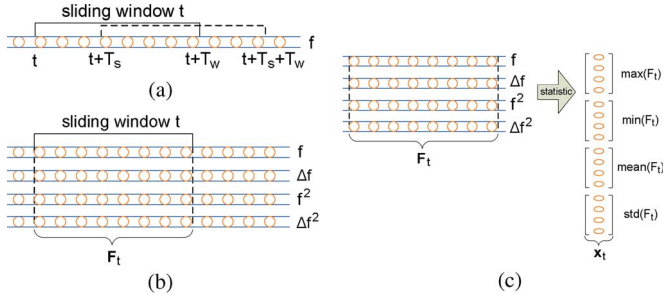


Fig. 2. Feature derivation. (a) Sliding window t ; each circle represents one data sample. (b) Obtaining F_t . (c) Obtaining \mathbf{x}_t .

To facilitate analysis, we regard those feature vectors derived from those windows that overlap any of the accidents as the *dangerous* samples and assign the highest danger-level score f_d to them. Next, we use a simple heuristic to identify the *safe* samples, which are the feature vectors that have sufficiently large temporal distances to all the accident instances, and label them with the lowest danger-level score f_s . The danger-level scores of the remaining feature vectors are unknown. As a result, the entire training data set \mathcal{X} can be denoted by $\mathcal{X} = L \cup U$, where $L = \{(\mathbf{x}_i, y_i) | i \in T_L\}$ is the labeled portion, $y_i \in \{f_s, f_d\}$, $U = \{\mathbf{x}_j | j \notin T_L\}$ is the unlabeled portion, and $|L| \ll |U|$. T_L is the index set that contains the time instances of all the *accident* and *safe* units.

From the training data set \mathcal{X} , we want to create a danger-level estimation model (function) $f(\mathbf{x})$ that can map each input feature vector \mathbf{x} into a numerical danger-level score. Our proposed learning method stipulates that the learned model $f(\mathbf{x})$ must satisfy the following two principles: 1) For labeled feature vectors $\mathbf{x}_i \in L$, the estimated danger-level scores $f(\mathbf{x}_i)$ must be as close to y_i as possible; and 2) for all the neighboring feature vectors $\mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}$, the difference between their estimated danger-level scores $f(\mathbf{x}_i) - f(\mathbf{x}_k)$ must be proportional to the spatiotemporal distance between \mathbf{x}_j and \mathbf{x}_k . Hence, the obtained function should generate low danger-level scores for normal driving states and higher scores for more dangerous driving states. As a typical semisupervised learning approach, our method does not need to label feature vectors $\mathbf{x} \in U$. Furthermore, our method produces a model that estimates a numerical danger-level score for each feature vector, which, we believe, is more appropriate for modeling the driving anomaly-detection problem than sequential classification-based approaches, because the entire driving state space is continuous rather than discrete. Our experimental evaluations show that our proposed method outperforms those classification-based methods in various aspects and is less affected by the imbalance between positive and negative training samples.

IV. SEMISUPERVISED LEARNING

As previously explained in this paper, we want to learn a danger-level estimation model $f(\mathbf{x})$ that assigns a danger-level score $f(\mathbf{x}_i)$ to a labeled feature vector $\mathbf{x}_i \in L$, which is as close to y_i as possible, and assigns scores $f(\mathbf{x}_j), f(\mathbf{x}_k)$ to neighboring feature vectors $\mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}$ with the difference

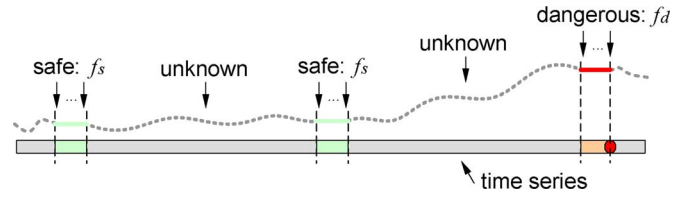


Fig. 3. Similarity propagation scheme.

proportional to the spatiotemporal distance between \mathbf{x}_j and \mathbf{x}_k . We strive to learn such a model by minimizing a predefined cost function using the quasi-Newton method. In this section, we propose two label-propagation schemes to utilize both the labeled and the unlabeled data for training and describe the corresponding optimization algorithms to compute the optimal $f(\mathbf{x})$.

A. Similarity Propagation

In the first scheme, the danger-level scores of the labeled feature vectors $\mathbf{x}_i \in L$ are propagated to the unlabeled feature vectors $\mathbf{x}_j \in U$ through the spatial similarity between neighboring feature vectors (see Fig. 3).

Using the notations defined in Section III, the cost function for this scheme can be defined as follows:

$$\begin{aligned} \phi(f) = \phi_1(f) + \lambda\phi_2(f) = & \sum_{\mathbf{x}_i \in L} (y_i - f(\mathbf{x}_i))^2 \\ & + \lambda \sum_{\mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}} (f(\mathbf{x}_j) - f(\mathbf{x}_k))^2 a_{jk} \end{aligned} \quad (1)$$

where a_{jk} is the spatial similarity between \mathbf{x}_j and \mathbf{x}_k , which is defined as

$$a_{jk} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_j - \mathbf{x}_k\|^2}{1.25}\right), & \text{if } k = j \pm 1 \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

In (1), the first term $\phi_1(f)$ penalizes the difference of an assigned score $f(\mathbf{x}_i)$ for a labeled feature vector $\mathbf{x}_i \in L$ to its label y_i , and the second term $\phi_2(f)$ penalizes the difference of assigned scores to two neighboring vectors $\mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}$ with high spatial similarity a_{jk} . This is an appropriate cost function for learning the desired $f(\mathbf{x})$ because it encodes both the two principles defined in Section III.

Equation (1) can be rewritten in a compact matrix form. Define an $|L| \times N$ matrix \mathbf{X}_L where row i is the labeled feature vector $\mathbf{x}_i \in L$, an $|\mathcal{X}| \times N$ matrix \mathbf{X} where row j is a feature vector $\mathbf{x}_j \in \mathcal{X}$, and an $|L| \times 1$ column vector $\mathbf{y} = [y_1, y_2, \dots, y_{|L|}]^T$. In addition, define a $|\mathcal{X}| \times |\mathcal{X}|$ similarity matrix \mathbf{A} where each element a_{jk} is defined by (2), a diagonal matrix \mathbf{S} where each diagonal element is equal to $s_{jj} = \sum_k a_{jk}$, and a matrix $\mathbf{L} = \mathbf{S} - \mathbf{A}$. With these definitions, the first term of (1) becomes

$$\phi_1(f) = \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T f(\mathbf{X}_L) + f^T(\mathbf{X}_L) f(\mathbf{X}_L) \quad (3)$$

where $f(\mathbf{X}_L)$ is an $|L| \times 1$ column vector $f(\mathbf{X}_L) = [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_{|L|})]^T$, and $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{|L|}$ are the row

vectors of the matrix \mathbf{X}_L . Similarly, the second term of (1) can be rewritten as

$$\phi_2(f) = 2 \left(\sum_j s_{jj} f^\top(\mathbf{x}_j) f(\mathbf{x}_j) - \sum_{jk} a_{jk} f^\top(\mathbf{x}_j) f(\mathbf{x}_k) \right) = 2f^\top(\mathbf{X}) \mathbf{L} f(\mathbf{X}). \quad (4)$$

Taking (3) and (4) into (1), we have

$$\phi(f) = \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top f(\mathbf{X}_L) + f^\top(\mathbf{X}_L) f(\mathbf{X}_L) + 2\lambda f^\top(\mathbf{X}) \mathbf{L} f(\mathbf{X}) \quad (5)$$

and our problem becomes finding a model $f^*(\mathbf{x})$ that minimizes $\phi(f)$, i.e., $f^*(\mathbf{x}) = \arg \min_f \phi(f)$.

The model $f(\mathbf{x})$ can take any parametric or nonparametric form. For nonparametric $f(\mathbf{x})$, the problem becomes the same as that proposed by Zhu *et al.* [15], where the authors proposed a quadric energy function to constrain the learned model to generate similar labels for similar feature vectors. With our problem, we use parametric $f(\mathbf{x})$ because we can use the quasi-Newton method and matrix computations to estimate the optimal $f(\mathbf{x})$ efficiently, and the obtained model $f(\mathbf{x})$ will have a simple form that requires fewer computations. This will be very important for any in-vehicle information systems because of limited computation resources in vehicles.

We first consider a simple linear form for $f(\mathbf{x})$

$$f(\mathbf{x}) = \mathbf{x}\mathbf{w} + \epsilon \quad (6)$$

where \mathbf{w} is the coefficient vector that defines the linear function, and ϵ is an independent and identically distributed noise with normal distribution $\mathcal{N}(\epsilon|0, \sigma_1^2)$. It can be verified that the optimal \mathbf{w}^* is

$$\mathbf{w}^* = (\mathbf{X}_L^\top \mathbf{X}_L + 2\lambda \mathbf{X}^\top \mathbf{L} \mathbf{X} + \sigma_1^2 \mathbf{I})^{-1} \mathbf{X}_L \mathbf{y} / 2$$

where $\sigma_1^2 \mathbf{I}$ is the Tikhonov regularization term. σ_1^2 is set to 10^{-5} in our current implementation.

In our experiments, we also evaluated the logistic function, hyperbolic tangent (tanh) function, and radical basis function (RBF) for $f(\mathbf{x})$. Specifically, for both the logistic and tanh functions

$$f(\mathbf{x}) = \sum_{n=1}^{N_n} a_n \tanh(\mathbf{x}\mathbf{w}_n + b_n) + c + \epsilon \quad (7)$$

where \mathbf{w}_n , a_n , b_n , and c are the parameters to be estimated. N_n is the number of tanh functions used. If $N_n = 1$, (7) becomes a logistic function; if $N_n > 1$, (7), the multi-tanh function actually represents a simple neural network with N_n neurons.

For Gaussian RBF form $f(\mathbf{x})$

$$f(\mathbf{x}) = \sum_{n=1}^{N_r} a_n \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}_n\|^2}{0.2}\right) + \epsilon \quad (8)$$

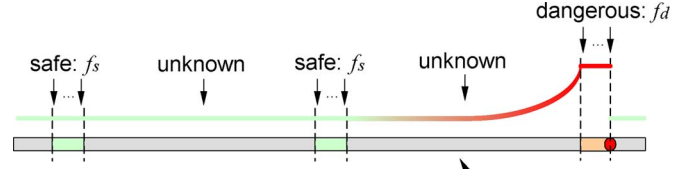


Fig. 4. AR propagation scheme.

where $\boldsymbol{\mu}_n$ and a_n are the parameters to be estimated, and N_r is the number of RBFs used. $N_r = 1$ represents a single RBF, and $N_r > 1$ represents multiple RBFs.

There are many techniques in the literature to optimize (5). In our implementation, we used the limited-memory Broyden–Fletcher–Goldfarb–Shanno (l-BFGS) method [16] to optimize the parameters of $f(\mathbf{x})$.

B. AR Propagation

The second propagation scheme aims to propagate the danger-level scores from the labeled data to those unlabeled data according to certain decaying rates (see Fig. 4). Hence, for unlabeled data, we want to find a danger-level model $f(\mathbf{x})$ that satisfies the following constraints:

$$f(\mathbf{x}_j) = \alpha f(\mathbf{x}_{j+1}) + \epsilon, \quad \mathbf{x}_j \in U \quad (9)$$

where $\alpha \in (0, 1]$ is a user-defined constant, and ϵ is an independent and identically distributed noise with normal distribution $\mathcal{N}(\epsilon|0, \sigma_1^2)$.

The cost function for the autoregression (AR) propagation can be defined as follows:

$$\begin{aligned} \phi(f) &= \phi_1(f) + \lambda \phi_2(f) \\ &= \sum_{\mathbf{x}_i \in L} (f(\mathbf{x}_i) - y_i)^2 + \sum_{\mathbf{x}_j \in U} (f(\mathbf{x}_j) - \alpha f(\mathbf{x}_{j+1}))^2 \end{aligned} \quad (10)$$

where the first term is the same as that used in the previous similarity propagation scheme, whereas the second term employs the AR approach to propagate the danger-level labels to unlabeled data. With the same definitions used in (3) and (4), (10) can also be rewritten in a compact matrix form, i.e.,

$$\begin{aligned} \phi(f) &= \mathbf{y}^\top \mathbf{y} - 2\mathbf{y}^\top f(\mathbf{X}_L) + f(\mathbf{X}_L)^\top f(\mathbf{X}_L) \\ &\quad + 2\lambda f(\mathbf{X})^\top \mathbf{D}^\top \mathbf{D} f(\mathbf{X}) \end{aligned} \quad (11)$$

where \mathbf{D} is a $|\mathcal{X}| \times |\mathcal{X}|$ sparse matrix with nonzero element on the diagonal and top-right off-diagonal locations. Specifically, for the diagonal element d_{jj} and the top-right off-diagonal element d_{jj+1}

$$[d_{jj} \quad d_{jj+1}] = \begin{cases} [1 & -\alpha], & \text{if } \mathbf{x}_j \in U \\ [0 & 0], & \text{otherwise.} \end{cases} \quad (12)$$

Since a parametric model $f(\mathbf{x})$ is preferred, we also evaluated the linear form in (6), the tanh form in (7), and the RBF form in (8). To optimize (11), again, we applied the l-BFGS method [16].

C. Inherent Relation Behind the Two

As can be seen from (5) and (11), the two propagation schemes result in very similar cost function forms. In fact, the ideas behind the two schemes are also very similar. They all require partial labeling and propagate the labeling information to the unlabeled feature vectors. The difference is the strategy of propagation: The similarity propagation scheme propagates the labeling information in proportion to the similarity between the neighboring vectors (4), whereas the AR propagation scheme propagates the information by a fixed decaying factor α in (9). This leads to different Laplacian matrices in the cost functions, i.e., \mathbf{L} in (5) and $\mathbf{D}^\top \mathbf{D}$ in (11). We can verify that, in (5)

$$\mathbf{L} = \begin{pmatrix} a_{1,2} & -a_{1,2} & & & \\ -a_{1,2} & a_{1,2} + a_{2,3} & -a_{2,3} & & \\ & -a_{2,3} & a_{2,3} + a_{3,4} & -a_{3,4} & \\ & & & \dots & \\ & & & & \dots \end{pmatrix}$$

and the $\mathbf{D}^\top \mathbf{D}$ term in (10) can be expanded as

$$\mathbf{D}^\top \mathbf{D} = \begin{pmatrix} 1 & -\alpha & & & \\ -\alpha & 1 + \alpha^2 & -\alpha & & \\ & -\alpha & 1 + \alpha^2 & -\alpha & \\ & & & \dots & \\ & & & & \dots \end{pmatrix}$$

which is also a Laplacian matrix. The first Laplacian matrix \mathbf{L} relates the propagated labeling values between the neighboring feature vectors \mathbf{x}_j and \mathbf{x}_k by their similarity a_{jk} , whereas the second Laplacian matrix $\mathbf{D}^\top \mathbf{D}$ relates the propagated labeling values by the fixed decaying factor α .

V. CLASSIFICATION-BASED METHODS

To make comparisons with the classification-based approaches, we also evaluated the hidden Markov model (HMM) and the conditional random field (CRF) algorithms for the purpose of danger-level estimations [17]. Since HMM and CRF are classification algorithms, we need to modify them so that comparisons can be made with our proposed method. The modifications are briefly explained as follows. More detailed explanations can be found in [17].

To obtain training samples for HMM and CRF, we extracted multiple nonoverlapping segments from the entire training data set. Each segment is 1 min long. These segments are extracted in such a way that they end with either an accident or a nonaccident state, and no accidents occur at the start or middle of any segments. The former segments are used as *dangerous* samples, and the later are used as *safe* samples. As in Section III, each segment is converted into a sequence of N -dimensional feature vectors.

To apply HMM, we built a *dangerous* HMM model Θ_d and a *safe* HMM model Θ_s using the *dangerous* and *safe* training segments, respectively. To apply CRF, we labeled all the units in a *dangerous/safe* training segment as *dangerous/safe*, respectively. Then, the entire set of training segments is used to train a CRF model Θ .

To train the HMM model, the expectation-maximization (EM) algorithm is applied. To train the CRF model, the same

l-BFGS method as that used in our proposed semisupervised method is applied.

To predict the danger-level score of a testing unit \mathbf{x}_i , we use the obtained HMM and CRF models to classify a 1-min-long segment before the current feature vector into either a *dangerous* or a *safe* class and use the classification confidence to derive a numerical value as the danger-level score. Letting $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i]$ be the segment of feature vectors, the danger-level score by HMM is computed as

$$f_{\text{HMM}}(\mathbf{x}_i) = \log \frac{p(\mathbf{X}|\Theta_d)}{p(\mathbf{X}|\Theta_s)}$$

and the danger-level score by CRF is computed as

$$f_{\text{CRF}}(\mathbf{x}_i) = \log \frac{p(\mathbf{x}_i \rightarrow \text{dangerous}|\mathbf{X}, \Theta)}{p(\mathbf{x}_i \rightarrow \text{safe}|\mathbf{X}, \Theta)}.$$

VI. EXPERIMENTAL RESULTS

In our experiment, 14 participants were invited to drive the “STISIM” simulator. Each subject performed two to three sessions, each session was 20 min long, and 40 sessions were conducted in total. In every driving session, the multi-channel sensor data stream was continuously captured, and a 148-dimension feature vector ($N = 148$) was derived for every data unit (see Section III). Both the proposed propagation schemes used $f_d = 300$, $f_s = 0$, and $\lambda = 1$. To make a fair comparison with the HMM and CRF methods, the *safe/dangerous* labels were given at a 1-min interval, i.e., any time instance with a given f_d or f_s values should be at least 1 min away from a neighboring label. For the AR propagation scheme, the decaying factor α in (10) was empirically set to 0.5.

A. Prediction Accuracy

First, we evaluated the danger-level prediction performance using a leave-one-out cross-validation strategy: The driving sessions from the same driver were left out in each trial for validation, and the remaining sessions were used for training. In each iteration, the validation data set was sequentially fed to the trained model to generate a continuous danger-level curve from which we calculated the receiver operating characteristic (ROC) curve. To proceed, we arbitrarily define an interval T_1 before each crash instance and regard this interval as the ground-truth dangerous time. This way, the set of real dangerous instances can be represented by $T_r = \{\mathbf{x}_j | \mathbf{x}_j \text{ to next crash} < T_1\}$. We varied T_1 between 1 and 30 s and calculated the ROC curve correspondingly. In addition, as mentioned in Section II, we tested the sliding window size T_w with 5 and 15 s, respectively. The following Figs. 5 and 6 illustrate the ROC curves when $T_w = 5$ and $T_1 = 10$ or 20 s, respectively, and Figs. 7 and 8 illustrate the ROC curves with $T_w = 15$ and $T_1 = 10$ or 20 s, respectively.

As can be seen from the aforementioned figures, the proposed method outperforms the HMM- and CRF-based methods. This is because, as explained in Section V, the danger-level scores computed by the classification-based methods are derived from the classification confidence. However, the

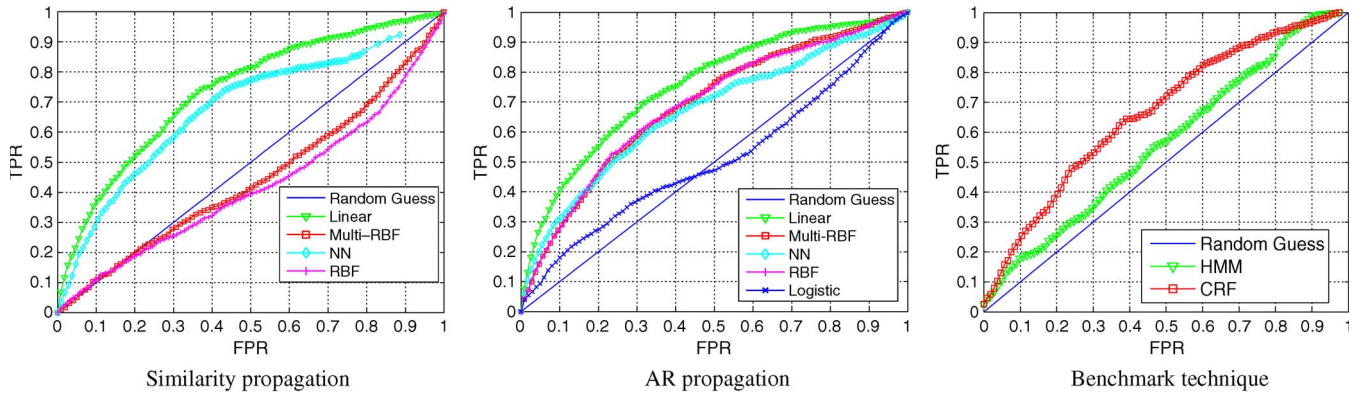


Fig. 5. ROC curve comparison ($T_1 = 10$ s, $T_w = 5$ s).

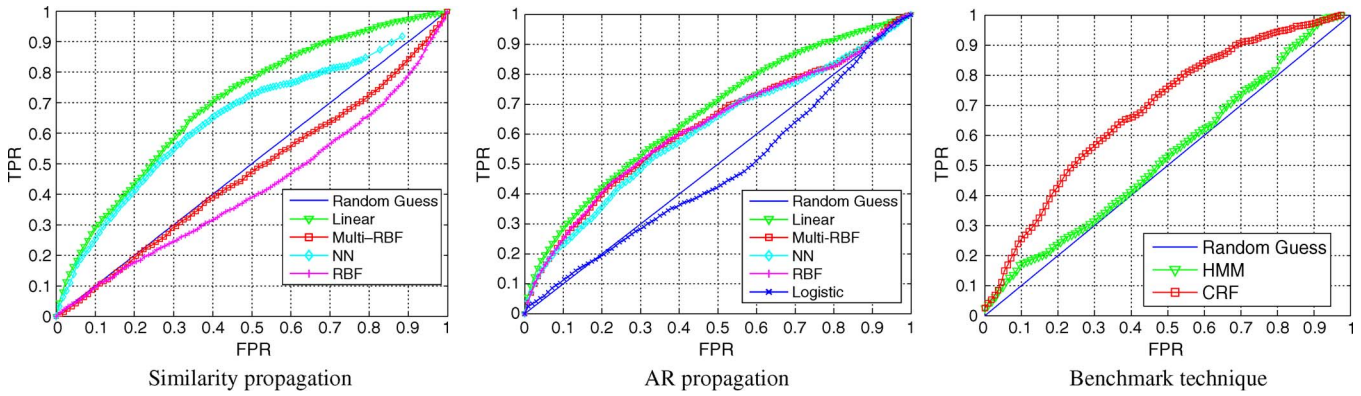


Fig. 6. ROC curve comparison ($T_1 = 20$ s, $T_w = 5$ s).

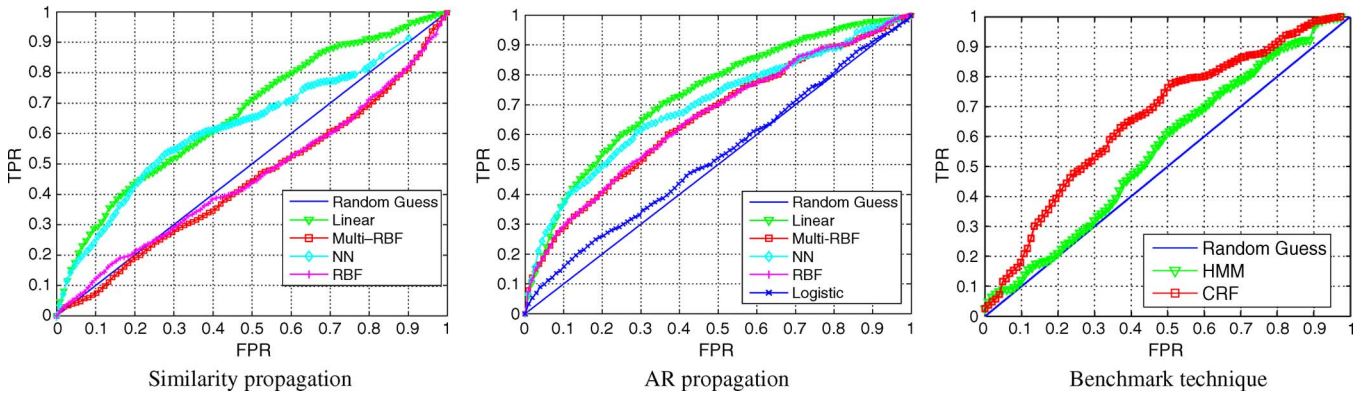


Fig. 7. ROC curve comparison ($T_1 = 10$ s, $T_w = 15$ s).

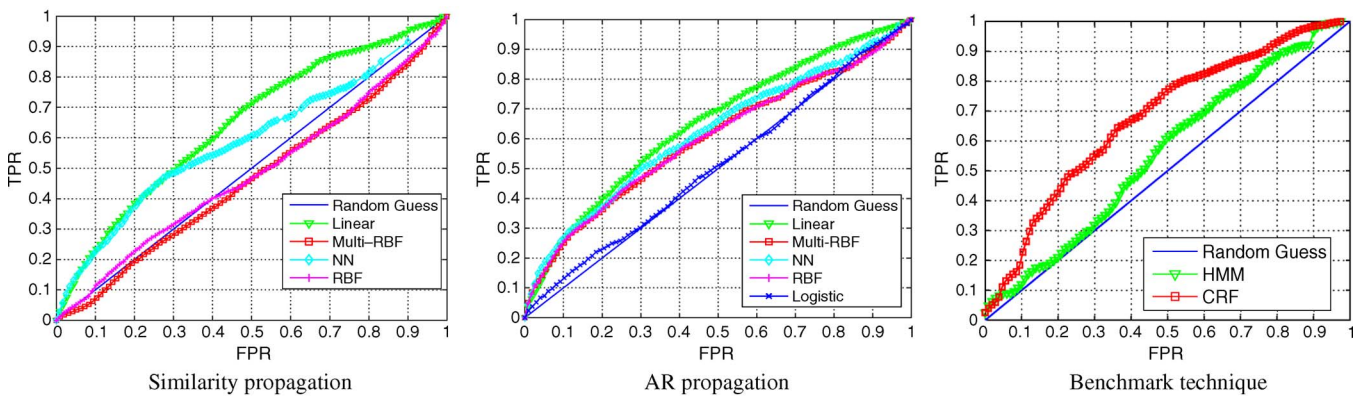


Fig. 8. ROC curve comparison ($T_1 = 20$ s, $T_w = 15$ s).

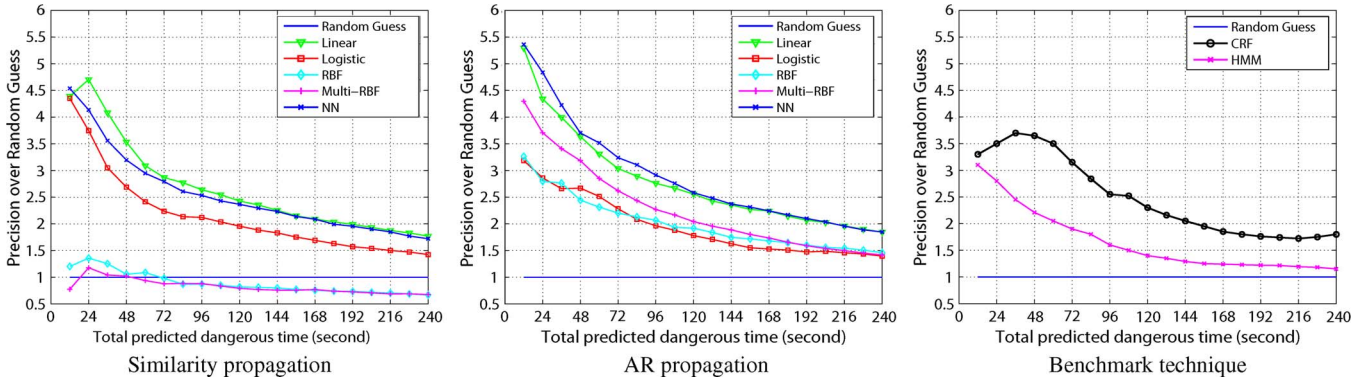


Fig. 9. Relative prediction precision.

confidence score has no direct connection to the danger-level score. For instance, higher confidence of being the dangerous class only implies that the stated segment is more similar to those dangerous training samples and does not necessarily represent that the driving state is more dangerous. This observation consolidates our statement in Section I that the classification-based methods are not an ideal model for the driving anomaly-detection problem, whereas our proposed model is more appropriate for modeling the driving anomaly-detection problem.

It is also observed that, compared with the similarity propagation scheme, the AR propagation scheme resulted in higher accuracy, because the similarity propagation scheme enforces the output of $f(\mathbf{x})$ to be proportional to the temporal distance to the accident states, whereas the AR model correlates the output of $f(\mathbf{x})$ to its neighboring predictions, which is more robust against noise. This fits the practical situation better.

Note that, in a 20-min driving course, the real dangerous instances only take a very small portion of the whole session. For instance, each session contains averagely five crashes. The real dangerous instances, denoted by T_r , add up to only 50–100 s when T_1 is set to 10 s. These instances take up only 4.17% to 8.33% of the whole driving session. Hence, it is appropriate to adopt another relative measure to examine how much the system can improve over random guess. Let us define a threshold T_2 , where the set of predicted dangerous instances can be represented by $T_p = \{\mathbf{x}_i | f(\mathbf{x}_i) > T_2\}$. Then, the precision is calculated as $P = |T_p \cap T_r| / |T_p|$. By adjusting T_2 , the total predicted dangerous time $|T_p|$ will change accordingly. In such a manner, a precision score can be calculated from each curve with respect to T_2 (or $|T_p|$), and the ratio between the score and the random guess accuracy can be assessed accordingly. Fig. 9 shows an example of the relative precision with $|T_p|$ spanning in 12–240 s. Fig. 9 can be interpreted as follows: When T_2 is large, i.e., $|T_p|$ is small, only instances that lead to peak danger-level scores are counted as predicted danger. The predicted dangerous instances mostly fall into the real dangerous period defined by T_1 , and the precision score is around 4.5 times higher (e.g., by the *similarity propagation* model in Fig. 9) than random guess. As T_2 decreases, $|T_p|$ will increase, and the system flags other dangerous instances that do not all lead to a crash. Hence, the precision score will decrease as the ground-truth danger time is defined solely based on a crash. Further

TABLE II
COMPARISON OF AVERAGE TRAINING TIME

method		time
Similarity prop.	Linear	92 sec
	Logistic	95 sec
	Multi-tanh	1191 sec
	RBF	176 sec
	Multi-RBF	188 sec
AR prop.	Linear	36 sec
	Logistic	46 sec
	Multi-tanh	974 sec
	RBF	160 sec
	Multi-RBF	169 sec
HMM		28007 sec
CRF		42222 sec

investigation is presented later to verify if some of the predicted dangerous instances are false alarm or near miss.

B. Convergence Speed

Next, we examined the computational time required for training our model in comparison with the classification-based methods. All the algorithms are implemented in MATLAB and run on a Dell Precision 650 PC with dual 3.2-GHz CPU and 2-GB RAM. Table II lists the average training time used in each cross-validation iteration. It can be seen that our method converged much faster than the HMM and CRF methods. The slow convergence of the HMM algorithm is due to the excessive number of iterations by the EM algorithm. The CRF algorithm is slow due to the computation of the partition function during the optimization process. On the other side, our method quickly converged because the defined cost function has a simple parametric form, such that the gradient can be easily obtained for updating the Hessian during the optimization process [16], and hence, the 1-BFGS method quickly converged.

C. Case Study

In this section, we further analyzed several predicted dangerous situations by looking at both the real driving conditions and the computed danger-level scores. This way, we want to gather more insight about how the system performs. First, a danger-level curve from a whole driving course is listed in Fig. 10 for three danger-level functions, i.e., the linear function in (6), the

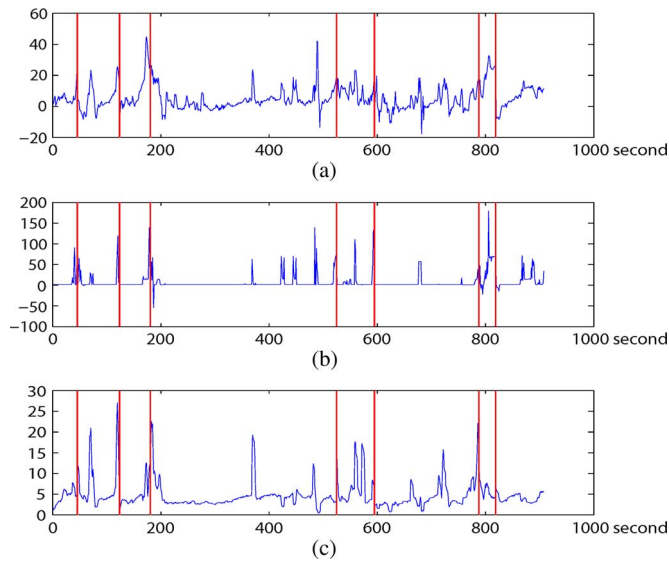


Fig. 10. Danger-level curve examples. (a) Linear model. (b) Multi-tanh model. (c) Multi-RBF model.

multi-RBF function in (8), and the multi-tanh function in (6). The vertical red lines represent the time instances when crashes were logged. Then, we selected three predicted dangerous situations for discussion, i.e., two leading to true crashes and one false alarm, as presented in the following sections.

Slow Danger Approaching Case: The first studied case corresponds to the second vertical red line in Fig. 10. It is a true crashed case, and the driving condition is shown in Fig. 11. In this case, bollards were placed on the driver's lane. Although the driver became aware of the situation 5.25 s before the crash [see Fig. 11(a)], he failed to bypass these roadblocks and finally hit one of the roadblocks. Now, back to the danger-level curves in Fig. 10, in all the three predicted curves, the generated system significantly increased the danger-level scores as the vehicle approaches the roadblock; hence, by using a simple thresholding boundary, all the three functions can predict this accident shortly before the crash. In addition, since the roadblock is observable with plenty of time before the crash, we expect that the danger can be detected as early as possible. This is achievable from both the linear danger-level function [see Fig. 10(a)] and the multi-RBF function [see Fig. 10(c)], whereas in Fig. 10(b), the multi-tanh function responded only when the vehicle was very close to crash.

Fast Danger Approaching Case: The second studied case is also a true crashed case, and it corresponds to the fifth vertical red line in Fig. 10. Its driving condition is shown in Fig. 12. In this case, the driver was driving normally until he entered a slippery road at 2.75 s before the crash [see Fig. 12(d)]. He tried to make a right bend, but the vehicle slipped to the left and shortly crashed. Now back to Fig. 10, in all the three prediction curves, the generated system significantly increased the danger-level scores; hence, they all can predict this accident. In addition, according to the driving condition, the instances before entering the slippery road should belong to the normal driving condition; hence, the curves by both the multi-tanh danger-level function [see Fig. 10(b)] and the multi-RBF function [see Fig. 10(c)] are more desirable because they remain low until close to the

crash. On the other side, in Fig. 10(a), the predicted danger-level score by the linear function started to rise much earlier. This obviously has nothing to do with entering the slippery road later, and hence, the linear function seems a bit more sensitive to noises.

False Alarm Case: The third studied case is illustrated in Fig. 13. It corresponds to 700–725 s in Fig. 10, with a predicted peak danger-level score at 725 s. In this case, the driver was driving at 60 mi/h when he entered a left bend. He did not reduce his speed and hence poorly remained the vehicle in his own lane. Then, he found the red traffic light and made an abrupt brake to reduce the speed from 45 mi/h to 0 within 2 s. During the last of this 2 s, the predicted danger-level score reached a local peak value [see Fig. 13(h)]. This is obviously a dangerous driving case because either reducing speed abruptly or invading into other's lane may cause a serious accident. However, since this very case did not result into a crash, the simulator did not record it as a dangerous instance for training. Going back to the three danger-level curves in Fig. 10, both the linear and multi-RBF functions generated increased danger-level curves for this period; hence, this dangerous driving portion could be correctly predicted. This result shows that, in spite of noises in training samples, our system robustly discovered the true patterns from dangerous driving behaviors. In addition, compared with the linear and multi-RBF functions, the danger-level curve generated by the multi-tanh function gave no predictions in this stated dangerous portion. Hence, although the ROC and relative precision metrics show that the multi-tanh gave the best performance, it is overfitted to the training samples, whereas the linear and multi-RBF functions have higher generalization ability.

D. Live System

Based on the offline cross-validation results, a live driving danger-level prediction system using the AR propagation scheme was developed, as shown in Fig. 14. It captures the driving simulator output (the left screen) and predicts the danger-level score in real time (the right screen). Several participants were invited to operate the system. Their online driving test results show that the system can predict driving risks due to sharp turning, sudden acceleration/deceleration, continuous weaving, approaching object, etc., events accurately. It is sensitive to the changes of vehicle's condition resulted from the driver's inattention or the road condition change such as a windy and slippery road. Overall, when the participant drove in good manner, the system stayed in silence unless a certain traffic condition changed, e.g., an ambulance or a police car approaching. Hence, the developed system worked as a reliable driving assistant with no noticeable false alarming. On the other side, if the participant wanted to crash the car on purpose, such as suddenly turning into the incoming vehicle, the alarm in our system did not go off because these types of dangerous situations do not occur in the collected training samples.

The first feedback compares between the linear and multi-RBF danger-level prediction functions. Most subjects felt that the linear function is too sensitive, whereas the multi-RBF function gives satisfactory stability and performance. Hence,

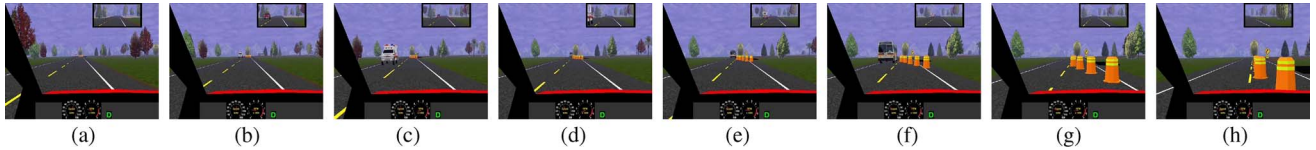


Fig. 11. Case 1 illustration. Hitting the bollard. (a) -5.25 s. (b) -4.5 s. (c) -3.75 s. (d) -3.00 s. (e) -2.25 s. (f) -1.5 s. (g) -0.75 s. (h) Crash.

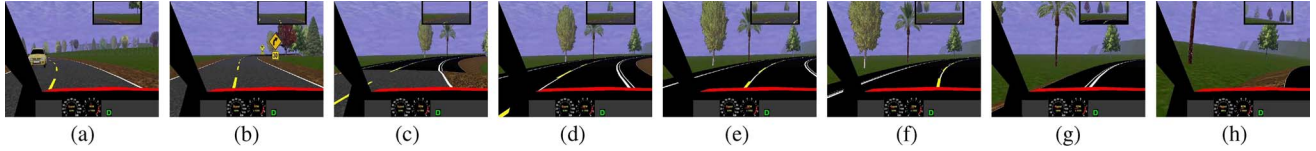


Fig. 12. Case 2 illustration. Off-road crash. (a) -9.60 s. (b) -5.73 s. (c) -3.64 s. (d) -2.75 s. (e) -2.05 s. (f) -1.68 s. (g) -0.84 s. (h) Crash.

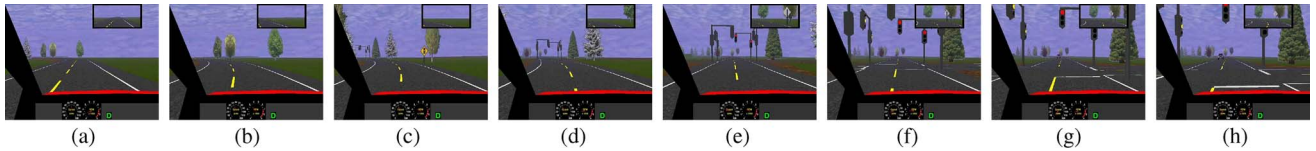


Fig. 13. Case 3 illustration. Lane violation and abrupt brake. (a) -14.66 s. (b) 11.42 s. (c) -6.79 s. (d) -5.38 s. (e) -3.98 s. (f) -2.80 s. (g) -1.40 s. (h) Peak danger-level (DL).



Fig. 14. Real-time driving DL prediction.

combining the objective comparison from Figs. 5–8 and the subjective user feedback, the multi-RBF danger-level prediction function is recommended.

The second feedback is with regard to the transparency of the danger-level function: When the system alarms, the drivers were often unclear about which action caused the risk. Although a precise danger reason probe module is not available at the current stage, it is able to analyze the trained parameters to roughly know the contribution of each feature for the increasing of the danger-level score. Our analysis is based on the linear function because it gave top performance in objective evaluation (see Section VI-A) and is easy to interpret. To proceed, recall that each dimension of the derived feature vectors is normalized (see Section II), in the learned w of the linear function, if any dimension has a greater positive value, and an increase in the corresponding feature vector dimension will lead to a greater increase in the danger-level score, and *vice versa*. Table III lists the top-10 feature dimensions that differ most between safe and crash.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a semisupervised learning method to mine the safe/dangerous driving patterns from time-

TABLE III
TOP-10 IMPORTANT FEATURES

ID	Description	DL \uparrow if
x_{98}	Square Mean Driver's longitudinal velocity	\uparrow
x_{54}	Max Delta Braking input counts	\downarrow
x_{50}	Max Delta Longitudinal acceleration due to the brakes	\uparrow
x_{27}	Mean Braking input counts	\downarrow
x_{28}	Mean Minimum range between the driver and all vehicles in the driver's direction	\uparrow
x_{24}	Mean Driver's longitudinal velocity	\downarrow
x_{144}	Square Std Delta Steering input counts	\downarrow
x_{89}	Square Max Driver's longitudinal velocity	\downarrow
x_6	Min Driver's longitudinal velocity	\uparrow
x_9	Min Braking input counts	\uparrow

series data with very limited labeling information. The proposed method achieved higher prediction accuracy and less training time compared with classification-based methods. In addition, although the labeling information is only given for accidents, the learned model is able to predict noncrash dangers such as near miss or other dangerous driving maneuvers. Based on the result, we further developed a real-time driving danger-level prediction prototype. The subjective driving test result was promising.

We have begun investigating the next stage of the proposed system. The future framework includes several parts: 1) to integrate the model into a more realistic driving simulator or real vehicles to perform an additional subjective user study; 2) to incorporate the driver's visual-behavior-based features to further improve the performance; 3) to evaluate additional machine learning algorithms on larger data sets so that more driving safe/dangerous patterns can be modeled; and 4) to improve the user interface in real vehicles and have a good way to notify the driver of the urgency of the information, as well as the confidence of it, all in a very short time in which he would not have time to interpret other notifications such as speech.

ACKNOWLEDGMENT

The authors would like to thank I. M. Jonsson, B. Reaves, and M. Kresse from Toyota InfoTechnology Center, USA, Inc., for designing the driving scenarios, providing data-acquisition devices, and conducting data collection for the research.

REFERENCES

- [1] Proposed Driver Workload Metrics and Methods Project, Dept. Transp., Nat. Highway Traffic Safety Admin., Washington, DC, 2000.
- [2] A. Kircher, M. Uddman, and J. Sandin, *Vehicle Control and Drowsiness*. Linköping, Sweden: Swedish Nat. Road Transp. Res. Inst., 2002.
- [3] J. Healey and R. Picard, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 2, pp. 156–166, Jun. 2005.
- [4] L. Bergasa, J. Nuevo, M. Steloand, R. Barea, and M. Lopez, "Real-time system for monitoring driver vigilance," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 63–77, Mar. 2006.
- [5] Y. Matsumoto and A. Zelinsky, "An algorithm for real-time stereo vision implementation of head pose and gaze direction measurements," in *Proc. IEEE ICVR*, 2000, pp. 499–505.
- [6] T. Victor and A. Zelinsky, "Automating the measurement of driver visual behaviours using passive stereo vision," in *Proc. Int. Conf. Series IV*, 2001, pp. 19–22.
- [7] S. Boverie, A. Giralt, J. M. L. Quellec, and A. Hirl, "Intelligent systems for video monitoring of vehicle cockpit," in *Proc. Int. Congr. Expo. ITS, Adv. Controls Veh. Navig. Syst.*, 1998, pp. 1–5.
- [8] S.-W. Shih, Y.-T. Wu, and J. Liu, "A calibration-free gaze tracking techniques," in *Proc. ICPR*, 2000, pp. 201–204.
- [9] H. Ning, W. Xu, Y. Zhou, Y. Gong, and T. S. Huang, "A general framework to detect unsafe system states from multisensor data stream," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 4–15, Mar. 2010.
- [10] N. M. Enache, S. Mammari, M. Netto, and B. Lusetti, "Driver steering assistance for lane-departure avoidance based on hybrid automata and composite Lyapunov function," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 28–39, Mar. 2010.
- [11] M. Amodeo, A. Ferrara, R. Terzaghi, and C. Vecchio, "Wheel slip control via second-order sliding-mode generation," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 122–131, Mar. 2010.
- [12] J. Healey and R. Picard, "Smartcar: Detecting driver stress," in *Proc. ICPR*, 2000, pp. 218–221.
- [13] T. Singliar and M. Hauskrecht, "Towards a learning incident detection system," in *Proc. ICML Workshop Mach. Learn. Surveillance Event Detection*, 2006, pp. 1146–1153.
- [14] 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.
- [15] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th ICML*, 2003, pp. 912–919.

- [16] C. Zhu, R. H. Byrd, and J. Nocedal, "L-BFGS-B: Algorithm 778: L-BFGS-B: Fortran routines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, Dec. 1997.
- [17] Y. Zhou, W. Xu, H. Ning, Y. Gong, and T. Huang, "Detecting unsafe driving patterns using discriminative learning," in *Proc. IEEE ICME*, 2007, pp. 1431–1434.



Jinjun Wang received the B.E. and M.E. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2000 and 2003, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2008.

In 2007, he joined NEC Laboratories America, Inc., Cupertino, CA, as a Research Scientist. His research interests include machine intelligence, pattern recognition, and content-based multimedia analysis, retrieval, and editing.



Shenghuo Zhu received the Ph.D. degree in computer science from the University of Rochester, Rochester, NY, in 2003.

He is a Research Staff Member with NEC Laboratories America, Inc., Cupertino, CA. His primary research interests include information retrieval, machine learning, and data mining. In addition, he is interested in customer behavior research, game theory, robotics, machine translation, natural language processing, computer vision, pattern recognition, bioinformatics, etc.



Yihong Gong (SM'10) 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.

In 1992, he joined 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 a Project Scientist with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. He was a Principal Investigator for both the Informedia Digital Video Library project and the Experience-On-Demand project funded by the National Science Foundation, the Defense Advanced Research Projects Agency, the National Aeronautics and Space Administration, and other government agencies. He is currently the Department Head of the Department of Information Analysis and Management, NEC Laboratories America, Inc., Cupertino, CA. His research interest include image and video analysis, multimedia database systems, and machine learning.