Temporal Difference Learning to Detect Unsafe System States

Huazhong Ning*, Wei Xu†, Yue Zhou*, Yihong Gong†, Thomas Huang*

*ECE Department, U. of Illinois at Urbana-Champaign, Urbana, IL 61801. {hning2, yuezhou, huang}@ifp.uiuc.edu.

†NEC Laboratories America, Inc., 10080 N. Wolfe Road, Cupertina, CA 95070. {xw,ygong}@sv.nec-labs.com

Abstract

This paper proposes a general framework to detect unsafe states of a system whose basic realtime parameters are captured by multi-sensors. Our approach is to learn a danger level function which can be used to alert the users in advance of dangerous situations. The main challenge to this learning problem is the labelling 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 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 expected future reward (penalty is regarded as negative reward) and use temporal difference (TD) learning [2] to learn a function to approximate the expected future reward. The TD learning obtains the approximation by propagating the penalty/reward observable at collapse points or successful ends to the entire feature space following some constraints. Our approach is applied to, but not limited to, the application of monitoring of driving safety and the experimental results demonstrate the effectiveness of the approach.

Keywords: *Temporal Difference Learning, Unsafe System State, Multi-sensor, Driving Safety*

1 Introduction

The increasing use of sensor technologies enables applications of increasing importance that automatically capture and process the basic parameters of and make decision for complex systems, such as in-vehicle information systems, computer networks, and so on. The direct goal is to monitor the performance of the complex system by analyzing the multi-sensor output, especially to detect unsafe states of the system before it collapses, so that certain measures can be taken in advance to avoid the collapses. We call it "unsafety detector". "Unsafe" here means high probability of system collapse in the following time interval. However, there is no quantitative definition yet. 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*.

Our work is a special case of anomaly detection. There are mainly three categories of methods of anomaly detection, that are more or less related to our approach. Rule-based methods [10] check the current and/or past sensor readings against a set of predefined rules. A combination of violations of the rules is regarded as an anomaly. A large amount of literature falls into the category of statistical methods [8, 9] that learn generative or discriminative models to identify the abnormal events. The third category of methods involves signal processing techniques [7, 5]. However, these methods usually require sample input-output pairs to derive rules or learn models, and the pairs are usually unavailable in our problem due to the labelling issue.

To avoid the labelling issue, we model the danger level as the expected future reward. A large value of negative expected future reward indicates highly dangerous state. On some specific points, the reward can be objectively assigned to the system. For example, the penalty (negative reward) assigned to a system crash can be decided based on its economic loss. With the reward suitably defined for the system, we use *temporal difference (TD) learning* [2] to learn the danger level function. Roughly speaking, TD learning aims to estimate the expected future reward of the system given the current and historical sensor readings. TD learning is originally applied to reinforcement learning. Although our task differs from reinforcement learning in that we do not need to find the optimal action, they share the same problem of estimating future reward.

In this paper, our approach is validated by the experiments of driving safety. But our approach is a general framework for monitoring unsafe system states. It has great potentials to be applied to a wide range of applications that require realtime monitoring.

2 Labelling Issue

Suppose that the feature at time t extracted from the sensor readings is X_t. Let J be the danger level function and $J(X_t)$ be the danger level given all past sensor readings. Our goal is to learn the danger level function J from the training data. An intuitive idea is to parameterize the danger level function and then estimate the parameters by supervised learning. However, supervised learning requires manually labelled training data, i.e., assigning a danger level to the sensor readings at any time t. Then a problem arises: can human do it objectively? The answer is **no**. Therefore most of previous work labels the training data manually and subjectively, *e.g.*, by questionnaires [4] or by simulating drowsy behaviors [1]. In these cases, the results are doomed to be subjective since the learning relies on the subjectively labelled training data.

However, it is determinable that it is definitely unsafe at a collapse point and that it is safe if the system is ended successfully, although the safeness is uncertain at any other time. Furthermore, as illustrated in Figure 1(a), it is natural to assume that it becomes more and more unsafe when approaching to the collapse point and that it is safer when being far away from the collapse point. Instead of being predefined by some subjective rules, this constraint should be learnt from the training data. From above, it is reasonable to assign an extremum penalty (without loss of generality, it is negative and minimum in this paper) to the collapse point and to assign a certain positive reward when the system has successfully ended. Then the penalty and reward are propagated to any other time according to the above learned constraint, which in turn forms the danger level function. Here the TD learning is used to accomplish the propagation and avoid the labelling issue.

The above constraint itself gives cues of how to manually (subjectively, too) label the training data. The *hard label* has discrete values $\{1, -1\}$: 1 means a safe state and -1 indicates a unsafe state. The *soft label* takes continuous value and is consistent with Figure 1(a). Both *hard* and *soft label* approaches are compared with our TD learning method in the experiments.

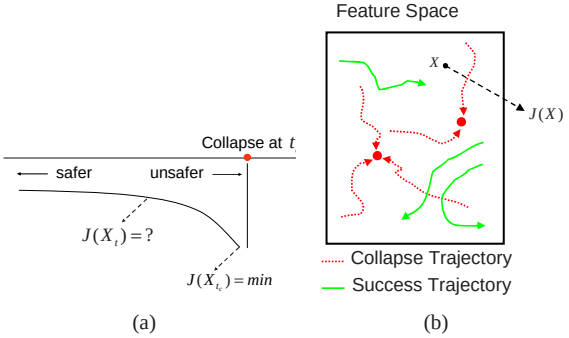


Figure 1. (a) Labelling issue. (b) Trajectories and sink points in feature space.

3 Temporal Difference (TD) Learning

We first give an intuitive idea of estimating the danger level function J by TD learning. In the feature space, each segment of a system running course corresponds to a trajectory which is ended with either collapse or success. As in Figure 1(b), the red dotted curves indicate collapse trajectories, and green solid curves success trajectories. The collapse (red) points can be viewed as sink points, and a moving point in the feature space will be either absorbed into a sink point or be stopped by external force, which forms collapse or success trajectory respectively. The training data correspond to sparse trajectories in the feature space. And J can take values of either a minimum penalty or a positive reward at the end of each trajectory. But the values at any other points in the feature space are uncertain. The TD learning propagates the penalty/reward so that J has values in the entire feature space.

3.1 Estimation by TD(λ)

Here the learning involves two main choices [2]: 1) the choice of an approximation architecture to represent the danger level function; 2) the choice of a training algorithm. We replace 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 non-linear function of r or even a neural network with r as the weights. In this paper, both the linear and non-linear representations are used,

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

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

In non-linear representation, S is the number of kernels and $r = (\alpha_1, \cdots, \alpha_S, r_1, \cdots, r_S, \beta)$. It is originated from the idea of sink points in Figure 1(b). The linear representation requires low computational cost and may have good generalization ability when feature dimension is high. In the following, we give a detailed description of the algorithm to learn the parameter r.

Assume that the system transition from state X_t to X_{t+1} incurs a danger level change $g(X_t, X_{t+1})$, then the danger level function $J(X_t)$ should satisfy the *Bellman's equation* [2, 6]

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

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

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

where $M^{(n)}$ is the penalty/reward at the end of the n-th trajectory. We consider the linear/nonlinear function of $J(X_t, r)$ 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]

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

The above least square 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 $(X_1, X_2, \cdots, X_T)$ 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 Eqn. 2 into Eqn. 4 and after a few manipulations, Δr is rewritten as

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

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. d_s is called *temporal difference* [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 weight $\lambda^{s-t}, 0 \leq \lambda \leq 1$, is used to decrease the influence of further temporal difference on $\nabla_r J(X_t, r)$. The above equation provides a family of algorithms, known as TD(λ).

Figure 2(a) gives a danger level function of a testing driving course constructed by linearly parameterized approximation. The crash (red) point and the preceding frames have very small value, which is consistent with that the smaller the danger level the more unsafe the system. There are some other points having very small values but without crash. Manually checking some of these points by playing back the driving courses in STISIM system shows that the drivers at these points usually take risk of unsafe driving. Since these points are not labelled as "unsafe", it deteriorates the performance of our approach to some extent.

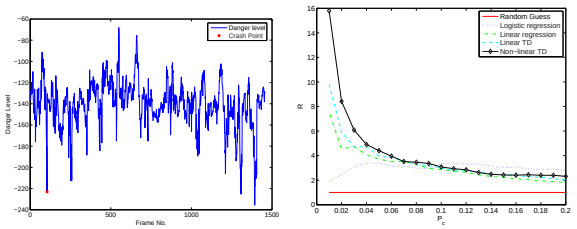
4 Experiments

Our general framework has a wide range of applications that attempt to detect unsafe system states by analyzing multi-sensor data. This paper applies it to driving safety to demonstrate its effectiveness.

Data collection. The multi-sensor data were collected through the STISIM simulator, including time stamp, distance, lane position, acceleration due to the throttle and brake, velocity, steering input, throttle input, brake input, and so on. We have 36 drivers, and each drives for about 20 minutes, having $1 \sim 3$ crashes per driving course. Each time one driver is selected out for testing and the other drivers for training. We compute the average performance over the 36 drivers.

Feature extraction. Instead of using the original sensor data Z_t, we extract new features from each window $Z_{t-s:t}$ by a transformation $X_t = T(Z_{t-s:t})$. The transformation T reduces the dimension of the original feature and summarizes the information meaningful to unsafety detection. Usually T is empirically determined but extracting information as much as possible. In driving safety, T extracts *mean*, *max*, *min*, and *variance* from each dimension of the sensor reading and *weave* from the dimensions of lane position and steering wheel. Here *weave* is the frequency of vehicle oscillation on the lane, which reveals the driver's skills, fatigue, and drowsiness to some extent. These measures are stacked and form the new feature X_t.

In the experiments, each trajectory contains $L = 60$



(a) (b)

Figure 2. (a) Danger level function constructed by linearly parameterized approximation. The red mark indicates a crash point. (b) P_c-R curves of our approach and the baselines. Best viewed in color.

frames, and each frame corresponds to a feature extracted from a window of 15 seconds. The consecutive frames have 80% overlapping. The non-crash trajectories are randomly selected from each driving course but they have no overlapping with any crash trajectory. The ratio of the number of non-crash trajectories to that of crash trajectories is kept to $10 : 1$ for each driving course. So we have about 1050 non-crash trajectories and 105 crash trajectories for training, and leave about 30 non-crash and 3 crash trajectories for testing.

Evaluation. We compare our algorithm with two baselines. The first baseline is a two-category classifier using logistic regression that is generally regarded as a safe and robust classifier relying on few assumptions [3]. Its training data are manually obtained *hard labels*. The second baseline is a linear regressor. Its training data are *soft labels*. Note that the training data for both baselines are labelled manually. Therefore, comparing our approach with the baselines demonstrates our approach effectively avoids the labelling issue.

We specifically define an evaluation metric to measure the performance. The metric is to measure how much the probability of accidents increases when we only look at the claimed danger time. The higher the increase, the more power of detecting unsafe driving. We do not have ground truth for the testing data except at the crash points. To make the evaluation feasible, t_r seconds before each crash are directly defined as *real danger time*. Let T_r be the total real danger time in the testing data and T be the total time. The *claimed danger time*, denoted as T_c, is the time when the value of danger level is below 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* 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 $R = p/p_r$ expresses how much the probability of accidents increases when we only look at the claimed danger time. The greater the R the better the performance. R is calculated at a certain level of $P_c = T_c/T$, the percentage of claimed danger time in total time. Lower P_c is preferred because high P_c often means high 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. Actually, a threshold corresponds to a value of P_c and, in turn, of R. Therefore, we can generate a P_c-R curve over all possible thresholds. The curve represents the performance of the corresponding approach. Note that the real number output, instead of the binary output, is used for the first baseline (logistic regression). Figure 2(b) gives the P_c-R curves of both our approach and the baselines. Obviously, the curve for the random guess is always one. From the figure, the order of performance at low $P_c(< 5\%)$, is random guess $<$ logistic classifier $<$ linear regression $<$ linear TD learning $<$ nonlinear (multi-RBF) TD learning. It shows that our approach, both linear and nonlinear TD learning, outperforms the baselines because our approach can avoid the labelling issue. Especially, the performance of nonlinear TD learning is much better than those of other methods. We compare the performance at low $P_c(< 5\%)$ because the in-vehicle warning system is preferred to work at low P_c. When we only look at the claimed danger time, the possibility of accidents is much higher than looking at the total time ($R \gg 1$), which means that our approach does have much power to detect unsafe driving.

5 Conclusions

This paper proposes a general framework to detect unsafe system states. It uses temporal difference learning to approximate a danger level function that indicates how safe/unsafe the system is. The main challenge to the learning procedure is the labelling issue. This paper uses TD learning to avoid the labelling issue. The approach is applied to, but not limited to, the application of driving safety and the experimental results demonstrate the effectiveness of the approach.

References

[1] L. Bergasa, J. Nuevo, M. Sotelo, R. Barea, and M. Lopez. Real-time system for monitoring driver vigilance. *IEEE Trans. Intelligent Transportation System*, 2006.

[2] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[3] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[4] J. A. Healey and R. W. Picard. Detecting stress during real-world driving tasks using physiological sensors. *IEEE Trans. Intelligent Transportation System*, 2005.

[5] G. Jiang, H. Chen, and K. Yoshihira. Discovering likely invariants of distributed transaction systems for autonomic system management. In *ICAC*, 2006.

[6] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.

[7] T. Lotze, G. Shmueli, S. Murphy, and H. Burkom. A wavelet-based anomaly detector for early detection of disease outbreaks. In *ICML*, 2001.

[8] T. Menzies and D. Allen. Bayesian anomaly detection. *ICML Workshop on MLASED*, 2006.

[9] T. Singliar and M. Hauskrecht. Towards a learning traffic incident detection system. *ICML Workshop on MLASED*, 2006.

[10] W. Wong, A. Moore, G. Cooper, and M. Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. *AAAI*, 2002.