Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Part C 16 (2008) 54–70

TRANSPORTATION RESEARCH PART C

www.elsevier.com/locate/trc

Incident detection algorithm based on partial least squares regression

Wei Wang [a], Shuyan Chen [a,b,*], Gaofeng Qu [c]

[a] *College of Transportation, Southeast University, Nanjing 210096, China*
[b] *Department of Electronic Information, Nanjing Normal University, Nanjing 210097, China*
[c] *College of Automation, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*

Received 14 May 2006; received in revised form 11 June 2007; accepted 14 June 2007

Abstract

We present the development of freeway incident detection models based on the partial least squares regression (PLSR), which has become a standard tool for modeling relations between multivariate measurements with flexibility, simplicity and strength. The PLSR models are built with the components extracted from the training dataset, and it distinguish incidents state from normal traffic state according to the output whether exceeding the threshold predefined. The performance of detection is evaluated using the common criteria of detection rate, false alarm rate, mean time to detection. Moreover, classification rate (CR), receiver operating characteristic (ROC) analysis and the area under the ROC (AUC) are also used to evaluate the model performance. Several experiments are performed to investigate the potential application of partial least squares regression to automatic incident detection. Simulated traffic data of Ayer Rajah Expressway (AYE) in Singapore and a real data collected at the I-880 Freeway in California were used in these experiments. The available traffic measurements, including speed, volume and occupancy collected at both upstream and downstream, are used to develop the PLSR model. The experiments conducted on the simulated traffic data studied the influence that the proportion of incident instances in training set and different length of time series of measured data have on the detection performance. In addition, empirical results are presented comparing with neural networks for freeway incident detection. The experiments conducted on the real traffic data discussed the problem resulted from imbalance data (incident instance is rare class in real world), and compares its detection performance with support vector machine (SVM). The experimental results have demonstrated that the PLSR model is comparative to a MLF neural networks and SVM implementation for AID applications, and PLSR has the potential for the application of automatic incident detection in the real world.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Automatic incident detection (AID); Partial least squares regression (PLSR); Receiver operating characteristic (ROC) analysis; The area under the ROC (AUC); Multi-layer feed forward neural network (MLF); Support vector machine (SVM)

[*] Corresponding author. Address: College of Transportation, Southeast University, Nanjing, Jiangsu 210096, China.
E-mail address: shuyan282@hotmail.com (S. Chen).

0968-090X/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trc.2007.06.005

1. Introduction

Freeway and arterial incidents often occur unexpectedly and cause undesirable congestion and mobility loss. If the abnormal condition cannot be detected and fixed up just in time, it increases traffic delay and reduces road capacity, and often causes the second traffic accidents. Therefore, automatic incident detection (AID) play an important role in most advanced freeway traffic management system. In the past decades, AID has attracted much attention from traffic researchers as an exciting research area, and a number of incident detection algorithms have been proposed and tested, a review of AID algorithms has been done by Parkany and Xie (2005).

A variety of detection/sensor technologies are used in detecting and providing real time traffic information for incident detection. These sensors commonly involve the use of inductive loop detectors (ILD) and video sensors, i.e., video image processors (VIPs). A VIP system consists of one or more video cameras, a micro-processor-based computer for digitizing and processing the video imagery, and software for interpreting the images and converting them into traffic flow data (Parkany and Xie, 2005). Some different vision-based automatic traffic incident detection algorithms have been put forward. According to the features applied to traffic incident detection, vision-based AID methods can be classified into two major categories: AID methods based on vehicular activities (Ikeda et al., 1999) and AID methods based on traffic flow abnormality (Jiang, 1997). The basic idea of the former is tracking each vehicle in the view of the camera and judging whether the vehicle is running without disturbances. The latter analyze and recognize traffic incidents based on abrupt abnormality of integrated traffic flow parameters such as average spatial speed, spatial occupancy, average distance of neighborhood vehicles, queue length, etc. Generally speaking, vision-based AID algorithms include three consecutive steps: object detection, vehicle tracking and activity understanding. Detailed information please refers to many studies on vision-based AID methods (Ikeda et al., 1999; Trivedi et al., 2000; Wang et al., 2005).

Video sensors become particularly important in traffic applications mainly due to their fast response, easy installation, operation and maintenance, and their ability to monitor wide areas (Kastrinaki et al., 2003). However, Inductive Loop Detectors (ILDs) are the most commonly used sensors in traffic surveillance and management applications. Currently, most incident detection systems and algorithms use traffic data derived from ILDs (Parkany and Xie, 2005). In this paper, we also discussed AID algorithms based on traffic data collected from ILDs.

Among existing AID algorithms, artificial neural networks, such as multi-layer feed forward neural network (MLF), probability neural network (PNN), constructive probability neural network (CPNN) have been investigated in freeway AID with good results (Chang, 1992; Ritchie and Abdulhai, 1997; Jin et al., 2001, 2002). The most significant advantage of using neural network models is that it can achieve high classification accuracy, or has a better transferability, that is adaptation capability to changing site traffic characteristics. The main drawback of it is its long back propagation training time (for MLF), or large network size requirement (for PNN), which could be a potential problem for its application in real world. Moreover, classification performance is not the only criterion by which to judge algorithms, another important criterion is simplicity and easy implementation.

More recently, Cheu et al. (2003) and Yuan and Cheu (2003) investigated the use of SVM for freeway incident detection and arterial incident detection, tested its performance on I-880 Freeway data in California and simulated data generated by Integration model, and compared the test results with the neural networks for freeway AID. These studies confirmed that SVM is a superior pattern classifier to be used in the AID problem. However, SVM has one drawback limiting its applications. Usually, the kernel function and parameters (specifically, a kernel parameter and a penalty parameter) of SVM have a great effect on the generalization performance, and setting the parameters of the SVM algorithm is a challenging task. At present, there is a lack of a structured way to choose them. Usually, an appropriate kernel function and parameters have to been chosen and tuned by trial and errors. Some study applied search techniques for this problem, such as simulated annealing (SA), genetic algorithm (GA) and Evolutionary algorithms (EA) (Quang et al., 2002; Imbault et al., 2004; Rojas and Fernandez-Reyes, 2005). However, a large amount of computation time will be still involved for such search techniques are themselves computationally demanding.

This paper investigates an alternative model, namely partial least squares regression (PLSR) model, for incident detection problem. PLSR is a multivariate data analysis technique which can be used to relate several response (Y) variables to several explanatory (X) variables. The method aims to identify the underlying factors, or linear combination of the X variables, which best model the Y dependent variables. Due to PLSR simplicity and easy interpretation, the applications of this approach can be found in an abundant literature (Leardi, 2000; Kim et al., 2001; VCCLAB, 2001; El-Feghi et al., 2004). However, the report of its application to traffic engineering is rare. It provides ample motivation to investigate this model performance on incident detection. The objective of this study is to develop PLSR model, and examine the applicability of this approach for AID. A traffic data simulated for Ayer Rajah Expressway (AYE) in Singapore and a real traffic data collected at the I-880 Freeway at San Francisco Bay area in California are used to investigate the powerfulness of PLSR for incident detection. We also showed that this approach is significantly comparative to a multi-layer feed forward (MLF) neural networks and a support vector machine (SVM) implementation of the application, and our experiments have demonstrated that PLSR has great potential to AID.

This paper is organized as follows. Section 2 introduces the algorithm of partial least squares regression. Section 3 discusses the measurements of detection performance. Section 4 presents experiments of PLSR for AID developed with AYE data, and compares its performance with MLF neural networks. Section 5 describes experiments conducted on I-880 real data, and compares PLSR with support vector machine to illustrate its performance. Finally, Section 6 gives some conclusions.

2. Partial least squares regression

This section will briefly introduce the algorithm of partial least squares regression. Details are readily obtained in many references on PLSR (Leardi, 2000; Kim et al., 2001; VCCLAB, 2001; El-Feghi et al., 2004). The PLSR method is usually presented as an algorithm and divided into a calibration and a prediction step.

Let the basic data be given by $X = [x_1, x_2, \ldots, x_m]$ and y, where each of the vectors $x_1, x_2, \ldots, x_m$ and $\mathbf{y}$ are $\boldsymbol{n}$ dimensional which is the size of samples. The data matrix $\mathbf{X}$ can be decomposed into a bilinear form as in Eq. (1)

$$X = t_1p'_1 + t_2p'_2 + \cdots + t_hp'_h + E_h \tag{1}$$

where p's are loading vectors, t's are latent variables (factors), and E_h is the residual matrix of $\mathbf{X}$ when the first h latent variables are included in the PLSR model. The basis for the PLSR method is that the relation between X and $\mathbf{y}$ is conveyed through the latent variables. This means that one also has a decomposition as in Eq. (2)

$$y = q_1t_1 + q_2t_2 + \cdots + q_ht_h + f_h \tag{2}$$

where the scalar q_h is the loading value of $\mathbf{y}$ and f_h is the residual vector of $\mathbf{y}$ when the first h latent variables are included in the PLSR model.

2.1. Calibration of PLSR

The algorithm specifies how to calculate the scores and loadings, which is formalized as follows:

Step 1: Scale the process variables.Both X and y are scaled to unite variance by dividing them by their standard deviation and centered by subtracting their average. This corresponds to giving X and y the same weight and same prior importance in the analysis

$$x^*_{ij} = \frac{x_{ij} - \overline{x_j}}{s_j}$$
$$y^*_i = \frac{y_i - \bar{y}}{s_y} \tag{3}$$

where $\overline{x_j} = \sum_{i=1}^{n} x_{ij}/n$ is the average of x_j, $s_j = \sqrt{(x_j - \bar{x}_j)'(x_j - \bar{x}_j)/(n-1)}$ is the standard deviation of x_j. Analogously, $\bar{y}$ and s_y are the average and the standard deviation of y, respectively. Then, set $h = 1, E_0 = (x^*_{ij})_{n*k}, f_0 = (y^*_i)_{n*1}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, m$.

Step 2: Calculate the weight vectors, w_h

$$w_h = E'_{h-1} f_{h-1} \tag{4}$$

Step 3: Calculate the score vectors, t_h

$$t_h = E_{h-1} w_h \tag{5}$$

Step 4: Calculate the loading values of X

$$p_h = E'_{h-1} t_h / (t'_h t_h) \tag{6}$$

Step 5: Calculate the loading values of y

$$q_h = F'_{h-1} t_h / (t'_h t_h) \tag{7}$$

Step 6: Find the residuals

$$\begin{aligned} E_h &= E_{h-1} - t_h p'_h \\ f_h &= f_{h-1} - q'_h t_h \end{aligned} \tag{8}$$

Step 7: Determine the stopping point.

Cross-validation (CV) is often used to fix stopping criterion. The most common used method is one-at-a-time form of cross-validation. The PLSR model is developed for all the instances save one, then tested on that hold-out instance. This is repeated n times, with each instance used as the validation instance in turn. The residual sum of squares (RSS) and the predicted error sum of squares (PRESS) for cross-validation are computed for different h-factor model. Choose the least number of factors whose residuals are not greater than the model with the lowest prediction error. Normally, define Q^2 as a stop criterion

$$Q^2 = 1 - \text{PRESS}_h / \text{RSS}_{h-1} \tag{9}$$

In practice normally, if $Q^2 \geqslant (1 - 0.95^2) = 0.0975$, continues extracting factors, otherwise stops.

However, such kind of CV involves a large amount of computation time to develop n PLSR models. Therefore, it is only suitable for the small training date set. Common alternatives are cross-validation by splitting the data into blocks or by reserved test set validation.

In our experiments, k-fold CV was used due to the large size of incident data set, that is, cross-validating the model with various numbers of factors, then choosing the number with minimum prediction error on the validation set. In k-fold CV, partition the training data set into k subsets of approximately equal size. The $k - 1$ of subsets is set aside to generate PLSR model with h components, while leaving the remainder as a testing data. Repeat this process k times, each time one different subset is used for testing. Suppose PERSS_{hi} $(i = 1, 2, \ldots, k)$ is the prediction error sum of squares obtained by testing the ith subset with the PLSR model generated with $k - 1$ groups

$$\text{PRESS}_{hi} = \sum_{j=1}^{n_i} (y_{ij} - \hat{y}_{ij})^2 \tag{10}$$

where y_{ij} and $\hat{y}_{ij}$ are the jth actual and prediction data in the ith subset, n_i is the size of the ith subset, and $\sum_{i=1}^{k} n_i = n$.

Next, compute the sum prediction error sum of squares over k times donated with PERSS_h

$$\text{PRESS}_h = \sum_{i=1}^{k} \text{PRESS}_{hi} \tag{11}$$

When PRESS declines only insignificantly when an additional factor is extracted, factor extraction is stopped. In other words, if the following condition is met, then select h as the optimal number of components to create PLSR model. Otherwise, increase h to $h + 1$, then repeat steps 2–7 unless the number of the optimal latent variables, h, is reached

$$\begin{aligned} &|\text{PRESS}_h - \text{PRESS}_{h+1}| \leqslant \sigma \\ &\text{or} \quad \text{PRESS}_h \leqslant \text{PRESS}_{h+1} \end{aligned} \tag{12}$$

where σ is a given level.

2.2. Prediction

Suppose the iteration stops at the hth iteration, and we obtain h principal components, $t_1, t_2, \ldots, t_h$, so f_0 can be expressed with them, written as

$$f_0 = q_1 t_1 + q_2 t_2 + \cdots + q_k t_h \tag{13}$$

Because the principal components are the linear combinations of original descriptors, factor model indirectly describes the effect of each descriptor on activity.

$$f_0 = q_1 E_0 w_1 + q_2 E_1 w_2 + \cdots + q_h E_{h-1} w_h = q_1 E_0 w_1^* + \cdots + q_h E_0 w_h^* \tag{14}$$

where $w_h^* = \prod_{j=1}^{h-1}(I - w_j p_j')w_h$, I is the identity matrix.

Finally, we have

$$\hat{y}^* = \alpha_1 x_1^* + \cdots + \alpha_m x_m^* \tag{15}$$

where $\alpha_j = \sum_{h=1}^{k} q_h w_{hj}^*$ is the coefficient of x_j^*, and w_{hj}^* is the jth element of w_h^*.

Perform anti-operation of normalization, we have

$$y = \bar{y} + s_y \left(\sum_{i=1}^{m} \alpha_i x_i^* \right) = \bar{y} + s_y \left(\sum_{i=1}^{m} \alpha_i \frac{x_i - \bar{x}_i}{s_i} \right) \tag{16}$$

This is the PLSR model. When fed with new vector of X, it gives the corresponding results. In our research, $\boldsymbol{X}$ is a feature matrix of size $n * m$ representing the extracted feature vectors from n instances, and y represents the traffic condition, 1 for incident and -1 for non-incident of a point time t. We used it to decide whether an incident happened or not for current measured parameters by comparing its output with a threshold predefined. In our experiments, zero is used as the threshold.

3. Performance measures

3.1. Definition of DR, FAR, MTTD and CR

The common criteria of detection rate (DR), false alarm rate (FAR), and mean time to detection (MTTD) are the key indicators of detection performance. DR and MTTD are written as

$$\text{DR} = \frac{\text{number of incident cases detected}}{\text{total number of incident cases}} \tag{17}$$

$$\text{MTTD} = \frac{t_1 + \cdots + t_m}{m} \tag{18}$$

here, t_i is the length of time between the start of the incident and the time the alarm is initiated, m is the number of incident cases detected successfully. If a single incident instance is identified within the period of actual occurrence of one incident case, which often consists of many continuous incident instances, the incident case is regarded as detected.

FAR is calculated to determine how many incident alarms were falsely set. There are several different formulas for this measure, following are two calculations commonly used (Jin et al., 2002; Cheu et al., 2003)

$$\text{FAR} = \frac{\text{number of false alarm cases}}{\text{total number of input instances}} \tag{19}$$

$$\text{FAR} = \frac{\text{number of false alarm cases}}{\text{total number of non-incident instances}} \tag{20}$$

The number of false alarm cases is computed differently. A continuous cluster of instances that were incorrectly classified as incident instances is taken to be one false alarm case (Cheu et al., 2003), another calculation is taking one instance misclassified as incident instance as one false alarm case. In our study, we used Eq. (19) to achieve FAR and adopted the first definition to compute the number of false alarm cases.

In addition to the above performance measures, the conventional index of classification rate (CR) is also used in this paper to measure the classification accuracy. CR is defined as the proportion of instances that were correctly classified based on total instances in data set, written as

$$\mathrm{CR} = \frac{\text{number of instances correctly classified}}{\text{total number of input instances}} \tag{21}$$

There exist trade offs between the DR, FAR and MTTD. A more sensitive detection model has a higher DR, short MTTD but also higher FAR. Persistence check offers an opportunity to adjust the DR, FAR and MTTD of an incident detection model. With the persistence test, an AID algorithm triggers an incident alarm only after a number of consecutive incident patterns have been classified correctly.

3.2. ROC analysis

Receiver operating characteristic (ROC) (Witten and Frank, 1999; Flach, 2004; Macskassy and Provost, 2004; Patel and Markey, 2005; Hopley and Schalkwyk, 2001) analysis is a widely used method for analyzing the performance of two-class classifiers. Advantages of ROC analysis include the fact that it explicitly considers the tradeoffs in sensitivity and specificity, includes visualization methods, and has clearly interpretable summary metrics. An ROC curve is a plot of the sensitivity vs. (1 − specificity) or equivalently the true positive rate (TPR) vs. the false positive rate (FPR), a frequently used measure of relative risk. TPR is the fraction of positive cases that are correctly classified as positive and FPR is the fraction of negative cases that are incorrectly classified as positive. Suppose incident instance is positive and non-incident instance negative, TPR and FPR can be computed as follows:

$$\mathrm{TPR} = \frac{\text{number of correctly classified incident instances}}{\text{total number of incident instances}} \tag{22}$$

$$\mathrm{FPR} = \frac{\text{number of misclassified non-incident instances}}{\text{total number of non-incident instances}} \tag{23}$$

ROC graphs plot false positive rates on the x-axis and true positive rates on the y-axis. A simple approach easy to implement to generate ROC curves is to collect the probabilities for all the various tests, along with the true class labels of the corresponding instances, and generate a single ranked list based on this data (Witten and Frank, 1999). This assumes that the probability estimates from the classifiers built from the different training sets are all based on equally sized random samples of the data.

The steps how to construct the points of the ROC curve from the output of testing data on PLSR developed in our experiments can be described in detail as follows. Suppose o_i is the output of PLSR testing on the ith instance, $y_i \in \{1, -1\}$ is the true class label of the corresponding instance, and $\hat{y}_i \in \{1, -1\}$ is the classified label derived from o_i, and p_i is the probability belonging to positive class, i.e., incident state, derived from o_i.

Firstly, according to the outputs of each testing data o_i compute the probability of each testing data belonging to positive class by linear transform. Next rank descendent all the testing instances by their probabilities. Then compute TPR and FPR along with each instance, which can be illustrated by the following pseudo codes. Last, from the first to the last instance, plot each point according to TPR_i vs. FPR_i.

```
$tp = 0, fp = 0;$
for $i = 1$ to $n$
  if $y_i = 1$ then $\{tp = tp + 1, \mathrm{TPR}_i = tp/n\}$
  else $\{fp = fp + 1, \mathrm{FPR}_i = fp/n\}$
end
```

where tp and fp are the number of true positive and false positive instances, n is the number of instances in the testing data set.

In AID problems, another common ROC curve is transfiguration plot by DR against FAR. These two kinds of ROC curves are different, because DR is not equal to TPR, while FAR is equal to FPR if FAR is computed with Eq. (20).

3.3. AUC

Often the comparison of two or more ROC curves consists of either looking at the area under the ROC curve (AUC) or focusing on a particular part of the curves and identifying which curve dominates the other in order to select the best-performing algorithm. The area under the ROC curve (AUC) (Macskassy and Provost, 2004; Hopley and Schalkwyk, 2001) is a numeric performance metric, which represents how separable two objects are. From the performance point of view, the detection performance of AID algorithm can be assessed using the area under the ROC curve which provides a singleton value for the assessment of performance. An AUC of 1.00 suggests that the classifier would always be able to distinguish a positive from a negative, and AUC of 0.5 indicates chance classification. Chance classification means that when posed with the task of distinguishing a positive from a negative, the classifier could at best "guess" what state the object was. Empirically, values of $\text{AUC} > 0.9$ indicate excellent detective power, while $\text{AUC} < 0.7$ indicate the lack of classification power. We consider value $\text{AUC} = 0.8$ as the threshold for the good accuracy of detections. The real beauty of using AUC is its simplicity. The visual and numeric metrics associated with this method allow for great flexibility in performance analysis.

How to calculate the area under the ROC curve? One method is integrating the area under the ROC curve (Bettinger, 2003). In our study, we obtained the AUC values by the trapezoidal rule. Slice the area into vertical segments, each segment would be trapezoid. The total AUC is calculated by adding these areas of segments together.

4. Case study with AYE simulated data

In the next two sections, we presented the development of freeway incident detection models based on PLSR, which was used to relate the extracted feature of traffic flow to the traffic state, that is, to map the relation between the traffic flow and the traffic state.

Several experiments were performed on AYE simulated data to investigate the performance of the PLSR method. The first experiment focused on the influence that the proportion of incident instances has on the detection performance. The second experiment used different length of time series data to build PLSR models, the idea behind this was to examine whether the previous traffic data has influence on the performance of the proposed PLSR. The last one emphasized on a comparison between the PLSR and MLF neural networks. We implemented the algorithm in Matlab code to construct the PLSR model, and used neural networks toolbox of Matlab for individual neural network training and classification.

4.1. Data description

The traffic data used in this study for the development of the incident detection models was produced from a traffic simulated system. A 5.8 km section of the Ayer Rajah Expressway (AYE) in Singapore has been selected to simulate incident and incident-free conditions. The selection of this site for incident detection study was due to its diverse geometric configurations that can cover a variety of incident patterns (Cheu et al., 1998; Jin et al., 2002).

The simulation system generated volume, occupancy and speed data at upstream and downstream for both incident and incident-free traffic conditions. The traffic dataset consisted of 300 incident cases that had been simulated based on AYE traffic. The simulation of each incident case consisted of three parts. The first part was the incident-free period that lasted for 5 min. This was after a simulation of 5 min warm-up time. The second part was the 10-min incident period. This was followed by a 30 min post-incident period.

The above 300 incidents were split in two mutually exclusive partitions, training data set and testing data set. Each data set had 3000 input patterns for incident state and 10 500 patterns corresponding to incident-free state. Each input pattern included traffic volume, speed and lane occupancy accumulated at 30-s intervals, averaged across all the lanes, as well as the label of traffic state. The value of label is -1 or 1, referred to the incident-free or incident, respectively.

4.2. Experiment 1: test influence of the proportion of incident instances on performance

Very often, a PLSR-based AID application involves two general steps: building a PLSR model with the training data, and then using this PLSR to classify. First, we use all the training data to build PLSR model, here $\boldsymbol{X}$ is a feature matrix of size $13\,500 * 6$ representing the extracted feature vectors, and Y represents the label of traffic state, 1 for incident and -1 for non-incident of a point time t. Therefore, the formal description of matrix X and Y can be written as follows:

$$\begin{aligned} X &= [x_1 \quad x_2 \quad \cdots \quad x_6] \\ &= \begin{bmatrix} \text{occupancy}_{\text{up1}} & \text{occupancy}_{\text{dn1}} & \text{volume}_{\text{up1}} & \text{volume}_{\text{dn1}} & \text{speed}_{\text{up1}} & \text{speed}_{\text{dn1}} \\ \text{occupancy}_{\text{up2}} & \text{occupancy}_{\text{dn2}} & \text{volume}_{\text{up2}} & \text{volume}_{\text{dn1}} & \text{speed}_{\text{up2}} & \text{speed}_{\text{dn2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{occupancy}_{\text{upn}} & \text{occupancy}_{\text{dnn}} & \text{volume}_{\text{upn}} & \text{volume}_{\text{dnn}} & \text{speed}_{\text{upn}} & \text{speed}_{\text{dnn}} \end{bmatrix} \\ Y &= \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \end{aligned}$$

where $n = 13\,500$ is the number of instances, and $y_i \in \{-1, 1\}$.

It is found with cross-validation (CV) described in Section 2.1 that three principal components are needed to describe the data set. The model is showed as follows:

$$Y = 0.35 + 0.01x_1 - 0.02x_2 - 0.01x_5 + 0.01x_6$$

Here, x_1 and x_2 indicate occupancy at upstream and downstream respectively, x_3 and x_4 indicate volume, x_5 and x_6 referring to speed at upstream and downstream, respectively. Y is the label of traffic state.

When the PLSR model fed with the testing matrix of X, it gives the corresponding results Y. We used it to decide whether an incident happened or not for current measured parameters by comparing its output with the threshold predefined, here set as 0. If the output y is larger than 0, it indicates the occurrence of incidents for this instance otherwise non-incident.

Then evaluate its performance on the corresponding testing set which is not used to construct this PLSR model. High DR and AUC, low FAR and MTTD are major requirements for the development of AID models. However, this PLSR model performed poor, for it gave very low DR, only 69.33%. The reason maybe there are much fewer incident instances, only 22.22% in the training set.

In order to find whether the proportion of incident instances in training set has any influence on the performance of PLSR model, we generate several new training sets, and each included all 3000 incident instances, the number of non-incident instance randomly chosen varied from 1500 to 10 500. Now, there is different proportion of incident instances in different training set, each training data set was used to build one PLSR model. Fig. 1 showed the trend of DR, FAR, MTTD and CR along with the change of the proportion of incident instances, where persistence test = 1. Increase the proportion of incident instances in training set can improve DR and reduce MTTD, however, it also yielded high FAR and reduce CR. It seems that using the dataset containing 41–56% of incident instances to build PLSR can obtain relative satisfactory performance. If there is too fewer incident instances, it gives too lower DR and too large MTTD; If there is too many incident instances, it gives too high FAR and too lower CR.

Table 1 showed the testing results obtained with 22.22%, 50.00% and 55.56% of incident instances in the training dataset. For comparison, the results obtained by persistence test of 2 were also listed in this table.

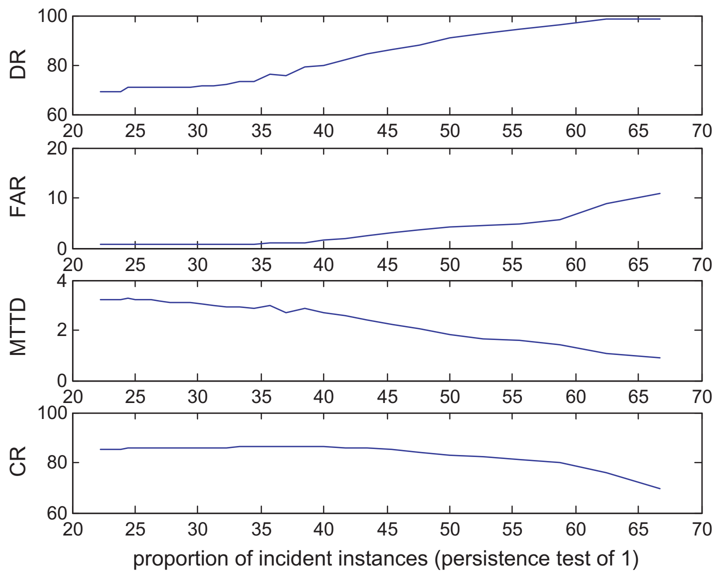

Fig. 1. Performance of PLSR changed along with the proportion of incident instances (AYE data).

Table 1
Ratio of incident instances has influence on performance of PLSR developed with AYE data

Proportion (%)	Persistent test	DR (%)	FAR (%)	MTTD (min)	CR (%)	AUC (%)
22.22	1	69.33	0.75	3.23	85.42	86.29
	2	64.00	0.55	3.59		
50.00	1	90.67	4.27	1.82	82.99	85.62
	2	79.33	1.68	2.45		
55.56	1	94.67	5.02	1.61	81.34	85.83
	2	88.67	2.04	2.52		

From this table, it is seen obviously making use of persistence test can reduce significantly the false alarm rate.

The detection performance of PLSR is encouraging if we use the training data set with proper ratio of incident instance. It illustrates that PLSR models are capable of capturing the dynamic relationship between the process variables and detecting the occurrence of the incident. Moreover, from the PLSR models, we can find the relative importance of different measured variables in the system being studied, for example, the model built from 50% of incident instances in the training dataset is follows:

$$Y = 0.99 + 0.01x_1 - 0.03x_2 - 0.01x_5$$

The variables, included in the model, implicitly suggested their importance to detect incident. In this model, x_1, x_2 and x_5, that is, occupancy at upstream and downstream, speed at upstream are important indicators for detecting incident.

4.3. Experiment 2: with time series data

In this experiment, we built the PLSR models with traffic measures from the previous time period $(t - n)$ to the current time slice (t), thus, the length of time series is $n + 1$. We built five PLSR models with different length of time series data from the time period $(t - 5)$ to t, from $(t - 4)$ to t, from $(t - 3)$ to t, from $(t - 2)$ to t, and from $(t - 1)$ to t, respectively. The proportion of incident instances in the training data set is 43.48. The models expressed with coefficients were shown in Table 2, and all the models were built with five

Table 2
PLSR models built with time series data of AYE data set

n	$Y = f(X)$
2	$1.01 - 0.01x_1(t-1) + 0.01x_5(t-1) - 0.01x_6(t-1) + 0.02x_1(t) - 0.03x_2(t)$
3	$0.94 - 0.01x_1(t-2) + 0.01x_5(t-2) - 0.01x_2(t-1) + 0.01x_6(t-1) + 0.01x_1(t) - 0.02x_2(t) - 0.02x_5(t) - 0.01x_6(t)$
4	$0.92 - 0.01x_1(t-3) + 0.01x_5(t-3) - 0.01x_2(t-2) + 0.01x_6(t-2) + 0.01x_1(t-1) - 0.01x_2(t-1) + 0.01x_1(t) - 0.02x_2(t) - 0.01x_5(t) - 0.01x_6(t)$
5	$1.11 - 0.01x_1(t-4) + 0.01x_5(t-4) + 0.01x_6(t-3) + 0.01x_1(t-1) - 0.01x_2(t-1) - 0.01x_5(t-1) - 0.01x_6(t-1) + 0.01x_1(t) - 0.02x_2(t) - 0.01x_5(t) - 0.01x_6(t)$
6	$0.96 + 0.01x_5(t-5) + 0.01x_6(t-5) + 0.01x_6(t-4) - 0.01x_2(t-2) - 0.01x_6(t-2) + 0.01x_1(t-1) - 0.01x_2(t-1) - 0.01x_5(t-1) - 0.01x_6(t-1) + 0.01x_1(t)$ $- 0.01x_2(t) - 0.01x_5(t) - 0.01x_6(t)$

Note: x_1 and x_2 indicate occupancy at upstream and downstream respectively, x_3 and x_4 indicate volume, x_5 and x_6 referring to speed at upstream and downstream, respectively.

Table 3
Performance of PLSR built with time series data in AYE data with 43.48% incident instance (persistence test = 1)

Length of time series	DR (%)	FAR (%)	MTTD (min)	CR (%)	AUC (%)
1	83.00	2.33	2.36	85.83	86
2	88.00	2.29	1.70	87.89	88
3	89.33	1.93	1.68	89.30	90
4	90.00	1.60	1.51	90.66	91
5	87.33	1.48	1.25	91.18	92
6	86.67	1.31	1.17	91.89	93

components. These models implied that volume at upstream and downstream in all the time period is not necessary to incident detection, because they did not included in any models.

The testing results of these PLSR models with persistent test of 1 were shown in Table 3, the data in first row yielded by the model built with current time period (t) is list for comparison. It is clear found that, with the length of time series increase, FAR and MTTD decrease, and CR and AUC increase, however, DR increase at first, achieve top at four, then drop when the length of time series exceeds four. Therefore, increasing the length of time series does not mean the improvement of performance, moreover, it requires much computing time. Considering all five evaluating index, the model built with time series data from time period ($t - 3$) to (t) performed best.

4.4. Experiment 3: comparison with neural networks

The previous studies have shown that neural networks are successful methods for AID. The neural network models mainly focused on the applications of multi-layer feed forward (MLF) neural networks for incident detection. In this paper, it is used as a benchmark for comparison. We trained 10 network classifiers using the training data set containing 50.00% of incident instances, and tested them on the testing data set. The networks fall into two different structures, one has three layers with six neurons in input layer, three neurons in hidden layer, and one neuron in output layer, another has the same number of input neurons and output neuron, but has six neurons in hidden layer. As before, the output indicates the traffic state compared to a predefined threshold, here set as zero, that is, if it is larger than 0, it indicates the occurrence of incident for this instance otherwise non-incident. The parameters for training network was set as follows, learning rate is 0.1, the maximal training epochs is 1500, and the learning goal is 0.04. The performance including running time averaged over 10 networks was listed in Table 4 for comparison.

Build PLSR with the same training data, and test it on the same testing data set. Table 4 showed the testing results, and the running time (s) is also listed in this table for comparison.

Compared PLSR with MLF networks, the PLSR seems inferior significantly on all indexes excluding the measurement of running time, which is the only one better than MLF networks.

Now, generate time series data with length of 4 from the previous training data set, and use them to build one PLSR again. The testing results without persistent test are listed in the last row. Compared with the previous PLSR, DR keep the same level, MTTD dropped slightly, but FAR decreased dramatically from 4.13% to 1.94%, CR raised from 82.99% to 89.82%, and AUC increased from 86% to 91%. Although the need time for building PLSR is larger than before, from 1.18 raised to 9.00 s, it is still very small compared to the training time that MLF networks needs, which is 344.70 s.

Table 4
Comparison between PLSR and MLF neural networks with AYE data

	Persistent test	DR (%)	FAR (%)	MTTD (min)	CR (%)	AUC (%)	Running time (s)
MLF	1	98.80	5.88	1.39	85.38	89	344.70
	2	90.27	1.44	2.29			
PLSR	1	90.67	4.13	1.82	82.99	86	1.18
	2	79.33	1.64	2.45			
PLSR built with time series	1	90.67	1.94	1.44	89.82	91	9.00

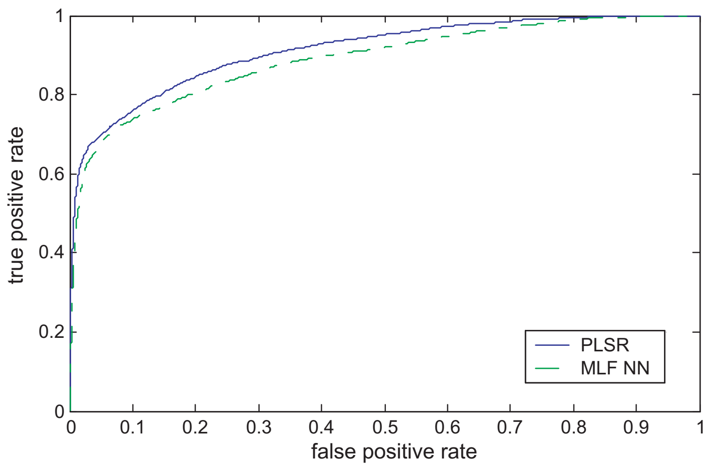

Fig. 2. Comparison of ROC curve between PLSR and MLF neural networks derived from AYE data.

The MLF networks without persistent test yield too high FAR to be accepted, thus, we selected the MLF networks with persistent test of 2 for further comparison. This PLSR model yielded similar DR, and better MTTD, CR and AUC than MLF networks. As FAR as concerned, 1.94 of FAR is slightly higher, compared to 1.44 yielded by MLF networks. Overall, the PLSR model built with time series data exhibits similar or modest better detection performance to MLF networks. Moreover, the PLSR converge very fast although it has much more input variables, while the MLF networks require too much time to train, 344.70 s far bigger than 9 s of PLSR. For this reason, we did not use time series data to develop MLF networks for AID model for comparison in this study, for it would demand far much more time to train.

Fig. 2 depicted the ROC curves of PLSR and MLF neural networks. In general, one point on the ROC curve is better than another if it is closer to the "northwest corner" point of perfect classification, (0, 1) (Bettinger, 2003). It can be seen that the ROC curve of PLSR is slightly higher than and at the left of ROC curve of MLF neural networks, it means that TPR (True Positive Rate) is higher than that of MLF neural network at the same FPR (False Positive Rate), and FPR is lower than that of MLF neural network at the same TPR. The ROC curve of PLSR dominating that of MLF neural network illustrates the fact that the PLSR approach has higher classification accuracy than MLF neural networks.

5. Case study with I-880 real data

This experiment focused on the performance with I-880 real traffic data. The influence that the proportion of incident instances has on the detection performance has been studied also in this experiment, and the comparison between PLSR and SVM has been made. The *N*-way Toolbox for Matlab (Andersson and Bro, 2000) was used to build PLSR model, whereas the LIBSVM Matlab toolbox (Chang and Lin, 2001) was used for SVM training and classification.

5.1. Data description

The second data set used in our research is the loop detector data collected at the I-880 Freeway in the San Francisco Bay area, California (Petty et al., 1996). The database has been used in many similar incident detection researches (Jin et al., 2002; Yuan and Cheu, 2003; Cheu et al., 2003). The loop detector data, in the form of lane specific volume, speed and occupancy were collected at 30-s intervals. The average values computed from all the lanes at a station were fed into the incident detection models. The incident data has 45 incident cases, in which 22 incident cases (2100 instances) were randomly selected as the training set, the remaining 23 incident cases (2036 instances) were used as the test set. The incident-free data collected on 16 February 1993 (43 418 instances) was used as the training set, while the incident-free data gathered on 17 Feb 1993 (43 102 instances) were used as the test sets.

5.2. Experimental results and analysis

We adopted I-880 real traffic data to build PLSR, here

$$X = [x_1 \quad x_2 \quad \cdots \quad x_6] = \begin{bmatrix} \text{speed}_{\text{up1}} & \text{occupancy}_{\text{up1}} & \text{volume}_{\text{up1}} & \text{speed}_{\text{dn1}} & \text{occupancy}_{\text{dn1}} & \text{volume}_{\text{dn1}} \\ \text{speed}_{\text{up2}} & \text{occupancy}_{\text{up2}} & \text{volume}_{\text{up2}} & \text{speed}_{\text{dn1}} & \text{occupancy}_{\text{dn2}} & \text{volume}_{\text{dn2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{speed}_{\text{upn}} & \text{occupancy}_{\text{upn}} & \text{volume}_{\text{upn}} & \text{speed}_{\text{dnn}} & \text{occupancy}_{\text{dnn}} & \text{volume}_{\text{dnn}} \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

where $n = 43418 + 2100 = 45518$, is the number of instances in training data set, and $y_i \in \{-1, 1\}$.

The PLSR is built with four components fixed by CV. Now evaluate its performance on the testing data. For comparison, we trained several SVM classifiers with different kernel functions and different parameters using the same training data, and tested them on the corresponding dataset. It is observed that the radial basis kernel function in general produced good results, so it was selected for further comparison with PLSR. The penalty parameter C was set to 1.0 and the parameter gamma for the radial basis kernels function was set to 1 while constructing SVM. The testing results about the performance of PLSR and SVM are listed in Table 5, with persistent check of 1 and 2.

Compared with the performance of SVM, the PLSR is inferior to SVM on DR, MTTD and CR. Moreover, DR of PLSR is too lower to be accepted, although its FAR is quite satisfactory. The reason maybe the training set is imbalanced data set, for there are only 2100 incident instances, compared to 43418 non-incident instances in the training set, it amounts to 4.84%. In order to find whether we can improve the detect performance of PLSR model by increasing the proportion of incident instances in training set, we discard randomly non-incident instances to improve the proportion of incident instances, then build PLSR models and test its performance. Fig. 3 depicted DR, FAR, MTTD and CR against the proportion of incident instances in the training set.

It is seen evidently that DR raised along with the proportion of incident instances, achieved the largest point at about 20% in our experiment, FAR become large while MTTD become small with the raising of incident instances in the training set. For CR, it rises until the proportion of incident instances reach about 18–20%, then drops down with the increase of incident instances. Therefore, the PLSR is sensitive to the proportion of positive and negative class in training data.

In order to achieve PLSR with better performance, how to choose training set should be given more attention. We chose randomly 12200 instances including all incident instances to generate a new training set, which contains 20.6% incident instances. Built PLSR with this new dataset, and got the following PLSR model:

$$Y = 0.68 - 0.02x_1 + 0.04x_2 - 0.03x_5$$

For comparison, construct a SVM model with the same parameters mentioned above on the same training data set. Next test PLSR and SVM models on the same testing data, and the results were shown in Table 6.

Table 5
Performance comparison between PLSR and SVM (I-880 data, 4.84% incident instances)

	Persistent check	DR (%)	FAR (%)	MTTD (min)	CR (%)	AUC (%)	Running time (s)
PLSR	1	52.17	0.06	7.54	95.90	94.29	1.16
	2	47.83	0.05	9.86			
SVM	1	86.96	0.08	4.20	97.08	92.36	203.28
	2	82.61	0.06	4.47			

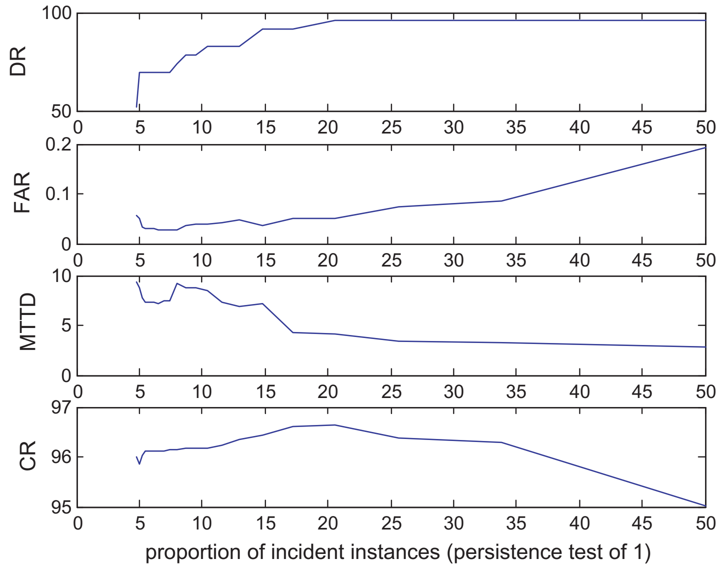

Fig. 3. DR, FAR and MTTD against the proportion of incident instances.

Table 6
Performance comparison between PLSR and SVM (I-880 data, 20.6% incident instances)

	Persistent check	DR (%)	FAR (%)	MTTD (min)	CR (%)	AUC (%)	Running time (s)
PLSR	1	95.65	0.06	4.66	96.65	94.26	2.32
	2	91.30	0.05	4.88			
SVM	1	91.30	0.24	2.55	94.65	93.60	13.69
	2	86.96	0.17	2.80			

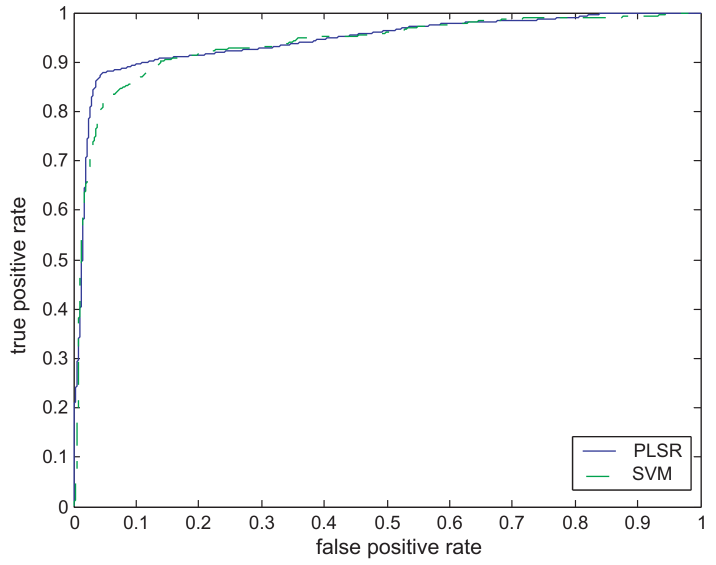

Fig. 4. Comparison of ROC curve between PLSR and SVM derived from I-880 data.

PLSR is superior to SVM on DR and CR, for DR, 95.65% compared to 91.30% for persistent test of 1, and 91.30% compared to 86.96% for persistent test of 2; for CR, 96.65% compared to 94.65%. PLSR also yielded better FAR, 0.06% compared to 0.24% for persistent test of 1, and 0.05% compared to 0.17% for persistent test of 2, respectively, almost one-third of SVM. A significant difference is that PLSR converges very fast, only 2.32 s, compared to 13.69 s of SVM. However, MTTD of PLSR is inferior to SVM, 4.66 compared to 2.55, and 4.88 compared to 2.80 yielded by SVM.

Fig. 4 gives the comparison of ROC curves between PLSR and SVM for incident detection. It can be seen the two curves are very close each other at the most left that we cannot tell which is better in detection performance.

However, it deserves attention that we chose the best performance of SVM for comparison. In practice, we do not know which kernel function and parameter has the best detection performance beforehand. For this reason, it is believed that PLSR models have a similar or better ability to the SVM methods in detecting incident, but with no parameter need to be adjusted.

6. Conclusions

The purpose of this research is to investigate thoroughly the detection performance of PLSR. With PLSR, there exists the capability to extract the relationship between the inputs and outputs of a process. Thus, this property of PLSR is well suited to the problem of incident detection under consideration. From our experiments conducted on AYE simulated traffic data and I-880 real traffic data, it is found that the relation between incident and traffic flow on a freeway could be represented by a PLSR model, and this model can be used to discriminate between normal traffic state and incident state compared to predetermined thresholds.

Our experiments show that PLSR is sensitive to imbalanced training data, that is, the proportion of incident samples in training set has much influence on detection performance, increasing this ratio, will yield high detection rate and lower MTTD, however, it also produce more false alarms. Therefore, it should be noticed about the proportion of incident samples in the training data set. At what ratio for incident instances contained in the training set so that PLSR model will perform better in terms of incident detection performance, our experiments showed that it depends on the different training data set.

If build the PLSR models based on current and previous time series data, the models build with $(t-3)$ to (t) data perform better. Increasing the length of time series data to build model do not assure the improvement of performance.

The comparison between PLSR and MLF neural network, between PLSR and SVM for AID indicates that the proposed PLSR models for AID perform similar to or better than the MLF model and SVM methods, but PLSR needs very little time to convergence, and there is no parameter need tuning in PLSR for detect incident. However, the performance of MLF neural network depends on the choice of structure and training parameters, and the performance of SVM depends on the choice of kernel function and several parameters for SVM. In addition, SVM, especially neural networks are very slow to converge.

In general, the advantage of PLSR models is it converges fast, without parameters to adjust, bring to light the relative importance of different variables, and is easy to implement with low computation load. The authors believe that PLSR models for AID possess the potential for real-time implementation and adaptation, and promise a significant improvement in operational performance.

Although the experiments have proved the strength of this approach, there are some problems in its proper application and several future works are still needed. As stated earlier, PLSR is sensitive to rare class, thus, if use PLSR in practice, the most important thing to do is instance selection, which chooses typical instances to construct PLSR to improve PLSR performance, this is worth studying. In addition, the linear form used in our experiments is the simplest PLSR form, next we plan to build non-linear PLSR models and test their performance to detect incident. Recently, there has been a trend away from data processing algorithms based on loop detector systems toward considering video-based technologies, thus, further work also includes developing PLSR for AID with video-based traffic data.

Acknowledgements

Shuyan Chen would like to thank Prof. Dr. Ruey Long Cheu (National University of Singapore), who provided the traffic incident data and related research materials for this study, also make a grateful acknowledge to Prof. Dr. Luc De Raedt (Freiburg University, Germany) for his many useful suggestions and help in this study. The authors are very grateful to the anonymous reviewers for their valuable suggestions and comments to improve the quality of this paper.

This work is supported by National Basic Research Program of China, under the Grant 2006CB705500, National Natural Science Foundation of China (No. 50608018), and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, Nanjing Normal University (Grant No. 2006102XLH0134).

References

Andersson, C.A., Bro, R., 2000. The N-way toolbox for MATLAB. Chemometrics and Intelligent Laboratory Systems 52, 1–4, Software available on the internet at http://www.models.kvl.dk/sourcer .

Bettinger, R., 2003. Cost-sensitive classifier selection using the ROC convex hull method. In: The Second Annual Hawaii International Conference on Statistics and Related Fields, pp. 1–12.

Chang, E.C.P., 1992. A neural network approach to freeway incident detection. In: The 3rd International Conference on Vehicle Navigation and Information Systems (VNIS), pp. 641–647.

Chang, C.C., Lin, C.J., 2001. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cheu, R.L., Jin, X., Ng, K.C., Ng, Y.L., Srinivasan, D., 1998. Calibration of FRESIM for Singapore expressway using genetic algorithm. Journal of Transportation Engineering, ASCE 124 (6), 526–535.

Cheu, R.L., Srinivasan, D., Teh, E.T., 2003. Support vector machine models for freeway incident detection. In: Proceedings of Intelligent Transportation Systems, 1, pp. 238–243.

El-Feghi, I., Alginahi, Y., Sid-Ahmed, M.A., Ahmadi, M., 2004. Craniofacial landmarks extraction by partial least squares regression. In: Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS '04) 4, pp. 45–48.

Flach, P.A., 2004. Tutorial on the many faces of ROC analysis in machine learning. In: The Twenty-First International Conference on Machine Learning, Canada.

Hopley, L., Schalkwyk, J.V., 2001. The magnificent ROC, Available at http://www.anaesthetist.com/mnm/stats/roc/.

Ikeda, H., Matsuo, T., Kaneko,Y., Tsuji, K., 1999. Abnormal incident detection system employing image processing technology. In: Proceedings of the IEEE International Conference on Intelligent Transportation Systems, pp. 748–752.

Imbault, F., Lebart, K, 2004. A stochastic optimization approach for parameter tuning of support vector machines. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004), vol. 4, pp. 597–600.

Jiang, Z.F., 1997. Macro and micro freeway automatic incident detection (AID) methods based on image processing. In: The IEEE Conference on Intelligent Transportation Systems, pp. 344–349.

Jin, X., Srinivasan, D., Cheu, R.L., 2001. Classification of freeway traffic patterns for incident detection using constructive probabilistic neural networks. IEEE Transaction on Neural Networks 12 (5), 1173–1187.

Jin, X., Cheu, R.L., Srinivasan, D., 2002. Development and adaptation of constructive probabilistic neural network in freeway incident detection. Transportation Research Part C 10, 121–147.

Kastrinaki, V., Zervakis, M., Kalaitzakis, K., 2003. A survey of video processing techniques for traffic applications. Image and Vision Computing 21, 359–381.

Kim, Y.S., Yum, B.J., Kim, M., 2001. A hybrid model of partial least squares and artificial neural network for analyzing process monitoring data. In: Proceedings of International Joint Conference on Neural Networks (IJCNN'2001), Washington, DC, USA, pp. 2292–2297.

Leardi, R., 2000. Application of genetic algorithm-PLS for feature selection in spectral data sets. Journal of Chemometrics 14, 643–655.

Macskassy, S.A., Provost, F., 2004. Confidence bands for ROC curves: methods and an empirical study. In: First International Conference on Machine Learning, Canada.

Parkany, E., Xie, C., 2005. A complete review of incident detection algorithms and their deployment: what works and what doesn't. Transportation Center, University of Massachusetts, Technical Report, NETCR37.

Patel, A.C., Markey, M.K, 2005. Comparison of three-class classification performance metrics: a case study in breast cancer CAD. In: Proceedings of Medical Imaging 2005: Image Perception, Observer Performance, and Technology Assessment, Bellingham, WA, vol. 5749, pp. 581–589.

Petty, K., Noeimi, H., Sanwal, K., Rydzewski, D., Skabardonis, A., Varaiya, P., 1996. The freeway service patrol evaluation project: database support programs and accessibility. Transportation Research 4C (3), 71–86.

Quang, A.T., Zhang, Q.L., Li, X., 2002. Evolving support vector machine parameters. In: Proceedings of International Conference on Machine Learning and Cybernetics, vol.1, pp. 548 – 551.

Ritchie, S.G., Abdulhai, B., 1997. Development testing and evaluation of advanced techniques for freeway incident detection, California PATH Working Paper, UCB-ITS-PWP-97-22, pp. 1–37.

Rojas, S.A., Fernandez-Reyes, D., 2005. Adapting multiple kernel parameters for support vector machines using genetic algorithms. In: The 2005 IEEE Congress on Evolutionary Computation, 1, pp. 626–631.

Trivedi, M.M., Mikic, I., Kogut, G., 2000. Distributed video networks for incident detection and management. In: Proceedings of IEEE International Conference on Intelligent Transportation Systems, Dearborn (MI), USA, pp. 155–160.

Wang, K.F., Jia, X.W., Tang, S.M., 2005. A survey of vision-based automatic incident detection technology. In: Proceedings of IEEE International Conference on Vehicular Electronics and Safety, pp. 290–295.

Witten, Ian H., Frank, E., 1999. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, San Francisco, 89–97, 125–127, 159–161.

VCCLAB (Virtual Computational Chemistry Laboratory), 2001. Partial least squares regression (PLSR). Available at http://146.107.217.178/lab/pls/m_description.html.

Yuan, F., Cheu, R.L., 2003. Incident detection using support vector machines. Transportation Research Part C 11, 309–328.