EPiC Series in Computing
Volume 75, 2021, Pages 32–40
CAINE 2020. The 33rd International Conference on Computer Applications in Industry and Engineering

Classifying Driver Attention Level Using Logistic Regression and Support Vector Machine

Ana Farhat[1] and Ka C Cheok[2]
[1,2] Oakland University, Rochester, MI, 48309-4401, USA
anafarhat@oakland.edu, cheok@oakland.edu

Abstract

One of the major reasons of road related accidents is driver distraction. The aim of this study is to classify driver attention level which can be extended to improving driver warning systems by generating adaptive driver alert warning systems. In this paper logistic regression (LR) and support vector machine (SVM) classifiers are used to classify driver attention level. The performance of mentioned classifiers is illustrated and compared via the figures of predicted decision boundaries. Also, in comparison to LR, higher accuracy of SVM has been verified.

1 Introduction

Inattentiveness while driving is one of the major reasons of road accidents, and it would endanger the lives of the driver and passengers. Several studies have confirmed that around 20% of road accidents are fatigue-related [1, 2]. This fact has been the main motivation for engineers to come up with the idea of safety improvements in the vehicles especially in Advanced Driver Assist Systems (ADAS) areas. In 2011 Ford has presented such a challenge of designing a classifier that detects the driver alertness [3]. They used a binary classifier to make decision whether drivers are in an alert state or not. Another approach in driver inattention detection system is getting the information not only from the driver's facial features but also from the road. [4] represents an approach that driver assistance system is not only responsive to the road environment and the driver's actions but also designed to correlate the driver's eye gaze with road events to determine the driver's observations. In this way it is possible to detect missed road events and alarm the driver.

In this paper driver attention level has been predicted and classified using logistic regression (LR) and support vector machine (SVM) classifiers. The performance of these classifiers has been compared via visualizing the estimated decision boundaries which separate different categories of data. This paper is organized as follows. Section II discusses in detail about collecting and processing the data. In Section III logistic regression classifier has been used to classify the data. In Section IV, support vector machine has been implemented on the same dataset, and the performance of two classifiers has been compared. Conclusion is presented in Section V, and at the end references are given.

2 Collecting and Labeling the DATA

In this paper we implement two well-known classification methods, LR and SVM, for driver's attention level classification. For this purpose, we collect the data from a simulated model in Matlab Simulink. In this driving scenario, we assume two cars with varying speed and acceleration on the road. Distance between two cars is measured by a LiDAR sensor mounted in front of ego car, and by processing the distance, we calculated relative speed and acceleration of two cars. Later these features, distance, relative speed and acceleration will be used as the input data to train the classifiers. This data for N data points is presented in (1).

$$X = \begin{bmatrix} d_1 & v_1 & a_1 \\ \vdots & \vdots & \vdots \\ d_N & v_N & a_N \end{bmatrix} \tag{1}$$

where first column illustrates the distance between two cars, and second and third columns show relative speed and acceleration. N is the data size. To use the data for training the classifiers, it needs to be labeled. In this research we label the data by calculating time to collision (TTC). TTC is the time required for two vehicles to collide if they continue at their present speed and on the same path [5]. Using (2) which is the relation between position, velocity, and acceleration, we calculate TTC for each row of data in X.

$$0.5at^2 + vt + d = 0 \tag{2}$$

Therefore, as shown in (3), we have TTC vector corresponding to N data points in X.

$$\text{TTC\_vector} = \begin{bmatrix} ttc_1 \\ \vdots \\ ttc_N \end{bmatrix} \tag{3}$$

Before labeling the data we need to consider one more piece of information which is driver's reaction time (RT). Driver's RT is the time takes for driver to react to a certain situation, and it varies from person to person. Through a test which measures driver's RT to react to size and color change of different objects, we measured this factor. In this test driver is asked to choose the object once it has the right color and size. A view of this test is presented in Figure 1. The time that takes for driver to choose the right objet is considered as driver's RT. In this paper measured RT is equal to 0.83 sec.

To apply TTC and driver's RT for labeling the data, through different tests we came up with a labeling logic presented in (4). According to (4), by comparing TTC and driver's RT, if TTC > 4RT, because the time that takes for driver to collide is larger than four times of driver's RT, it can be considered as a completely safe situation, and we label it as Perfect, meaning driver's focus level is very high. Similarly if 2RT < TTC $\leq$ 4RT we label it as Normal, and if TTC $\leq$ 2RT, because the collision time is very close to driver's RT and most probably driver will collide, we label it as Poor focus level of driving. This logic has been summarized in (4).

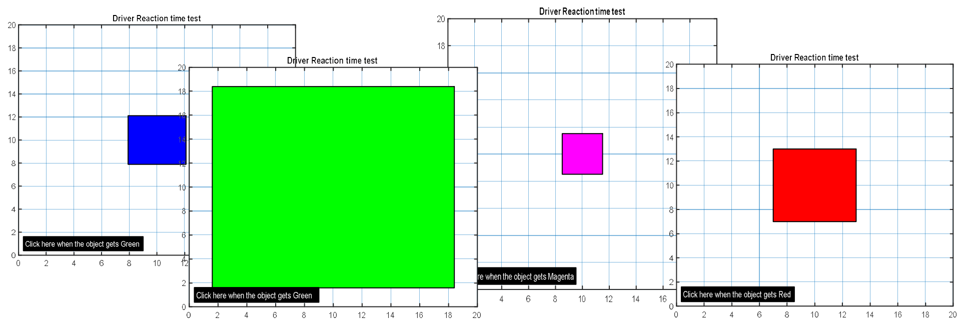
Driver Reaction time test
Click here when the object gets Green
Click here when the object gets Magenta
Click here when the object gets Red

Figure 1. Driver's RT test

$$\begin{cases}\text{if TTC} > 4\text{RT} & \text{label as Perfect}\\ \text{if } 2\text{RT} < \text{TTC} \leq 4\text{RT} & \text{label as Normal}\\ \text{if TTC} \leq 2\text{RT} & \text{label as Poor}\end{cases} \tag{4}$$

Using TTC vector, driver's RT, and logic above, we were able to label our data in X to three different classes as Perfect, Normal, and Poor. Labels are stored in vector y as shown in (5).

$$X = \begin{bmatrix} d_1 & v_1 & a_1 \\ \vdots & \vdots & \vdots \\ d_N & v_N & a_N \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{5}$$

A sample of above dataset with considering driver's RT equal to 0.83 sec is represented in Figure 2. Green labels represent Perfect zone which TTC > 4RT, blue labels represent Normal zone that 2RT < TTC ≤ 4RT, and red labels illustrate Poor zone where TTC ≤ 2RT.

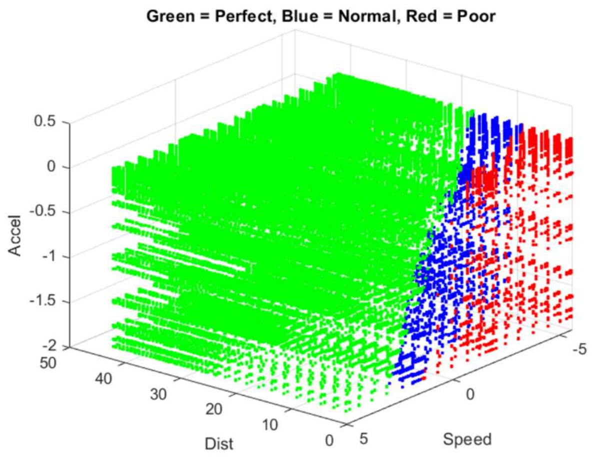

Figure 2: Sample data with three features: Dist, Speed, Accel

3 Classifying using Logistic Regression and visualizing decision boundaries

To predict driver's focus level, or in other words, to classify the entire dataset into three different categories, we need a classifier which helps us to come up with the equations of decision boundaries as accurate as possible. In this section we are applying LR as our data classifier. LR is a machine learning algorithm which is based on the concept of probability. The hypothesis function of LR is defined in (6) which is estimated probability on a binary classification.

$$h_\theta\left(X^{(i)}\right) = g(\theta^T X^{(i)}) \tag{6}$$

Function $g(.)$ is a sigmoid function which limits $h_\theta\left(X^{(i)}\right)$ between 0 and 1. A graph of sigmoid function is represented in Figure 3. θ includes the unknown parameters of decision boundaries' equations which need to be trained, and $X^{(i)}$ is the i^{th} row of matrix X. The general form of θ and $X^{(i)}$ for a simple binary classification problem by considering the bias term is as follows:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}, \; X^{(i)} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{bmatrix}$$

where m is the number of input features and in this paper $m = 3$. In general, θ is $(m + 1) \times 1$ and in this paper it will be a 4×1 vector. By applying θ and $X^{(i)}$ in (6) we have:

$$g(\theta^T X^{(i)}) = g\left(\begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_m \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{bmatrix} \right) = g(\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m) \tag{7}$$

where term $\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m$ represents the equation of decision boundary for a binary classifier. Knowing $g(.)$ is a sigmoid function, and considering 0 and 1 as two classes of the binary classifier, we have

- if $\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m \geq 0$:
  $g(\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m) \geq 0.5 \Longrightarrow$ label as 1
- if $\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m < 0$:
  $g(\theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m) < 0.5 \Longrightarrow$ label as 0.

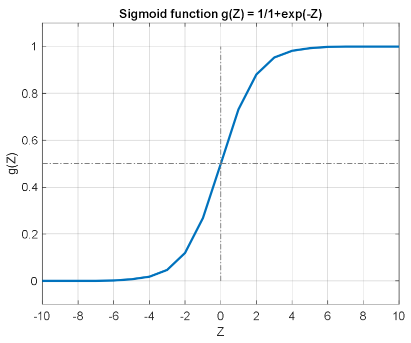

Figure 3: Sigmoid function graph

In cases that we have multi classes, θ, unknown parameters of decision boundaries, will be modified to a $(m+1)\times n$ matrix as below

$$\theta = \begin{bmatrix} \theta_0^1 & \cdots & \theta_0^n \\ \vdots & \ddots & \vdots \\ \theta_m^1 & \cdots & \theta_m^n \end{bmatrix} \tag{8}$$

where n and m are the numbers of classes and features, respectively. As mentioned earlier, in this research our data has been labeled to three classes (Perfect, Normal, Poor), and input vector $X^{(i)}$ has three features (distance, relative speed, relative acceleration). Therefore, matrix θ will be a 4×3 matrix as below:

$$\theta = \begin{bmatrix} \theta_0^1 & \theta_0^2 & \theta_0^3 \\ \theta_1^1 & \theta_1^2 & \theta_1^3 \\ \theta_2^1 & \theta_2^2 & \theta_2^3 \\ \theta_3^1 & \theta_3^2 & \theta_3^3 \end{bmatrix}. \tag{9}$$

To train the classifier and update the parameters, we define our cost function for each class c as below [6]

$$J\left(\theta^{(c)}\right) = \frac{1}{N}\sum_{i=1}^{N}\left[-y^{(i)}\log\left(h_\theta\left(X^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(X^{(i)}\right)\right)\right] + \frac{\lambda}{2N}\sum_{j=1}^{m}{\theta_j^{(c)}}^2 \tag{10}$$

where $c = 1,\dots,n$, and N is the number of training examples. The second part of cost function has been added for regularization purposes. With the right choice of regularization parameter λ, the algorithm will be robust against under-fitting and over-fitting. According to (10), if λ is set to an extremely large value, algorithm results in under-fitting (fails to fit even the training set), or if it is set to extremely small value, the algorithm will face over-fitting [6]. In this paper we have $\lambda = 0.1$. To get the optimized values of θ, we need to minimize the cost function J with respect to θ, $min_\theta J(\theta)$. To find the minimum of cost function, several gradient-based algorithms such as Levenberg-Marquadt Algorithm (LMA) [7, 8] have been applied. Eventually, according to the results, `fminunc` matlab function has been used which finds the minimum of unconstrained function. `fminunc` calculates the minimum of a problem specified by

$$min_x f(x)$$

where $f(x)$ is a function that returns a scalar [9]. In this research `fminunc` has been used in the form of (15).

```
x = fminunc(fun, x0, options)                (15)
```

This minimizes `fun` with the optimization options specified in `options`. Because `fminunc` is faster and more reliable when we provide derivatives [9], we define the objective function in a way that it returns both the gradients and function values. In this study, `fun` is the cost function defined in (10) and `x` is $\theta^{(c)}$.

Because we have multi classes, we need to train multiple classifiers; this method is called one-vs-all classification. One-vs-all classification is a method which involves training n distinct binary classifiers, each designed for recognizing a particular class [10]. By training the classifier we get the values of θ as shown in (16), then as illustrated in (7), by using the equation of decision boundary for each binary classifier, we can plot and visualize the decision boundaries for different classes. Considering the updated values of θ, we have the plots of black decision boundaries for three classes represented in Figure 4 and Figure 5.

$$\theta = \begin{bmatrix} -0.24 & 0.62 & 2.14 & 0.9 \\ -0.72 & -0.2 & -0.35 & -0.69 \\ 0.11 & -2.09 & -3.86 & -0.51 \end{bmatrix}^T \tag{16}$$

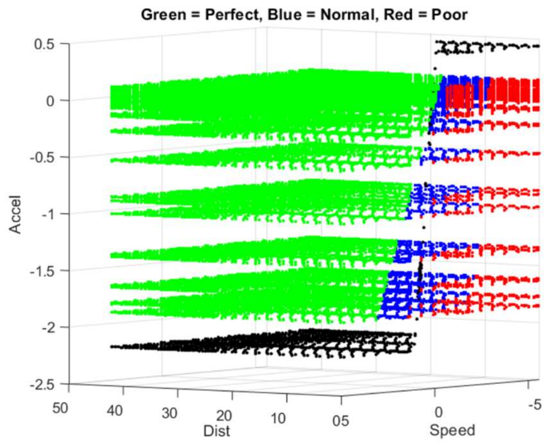

Figure 4: Black LR decision boundary between class Perfect and Normal

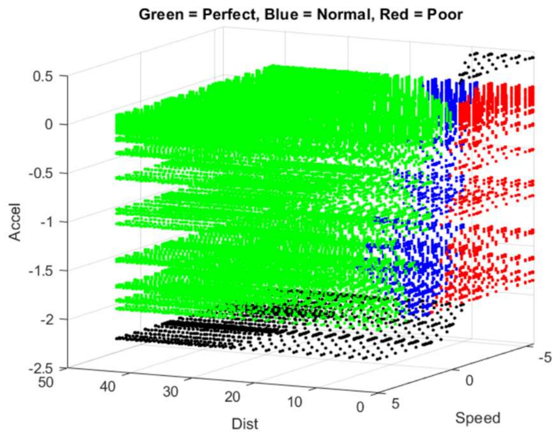

Figure 5: Black LR decision boundary between class Normal and Poor

As it is visible in mentioned figures, these decision boundaries which have been plotted based on updated values of θ, do not have an acceptable precision since they do not clearly separate different zones of data. The prediction accuracy on testing data is 0.76.

4 Classifying Using Support Vector Machine and visualizing decision boundaries

In this section we implement SVM classifier for the same dataset. SVM can efficiently perform both linear and non-linear classifications using kernel trick. In this research, according to the linear nature of data, and also because training SVM with linear kernel is faster than with any other kernels, we train SVM with linear kernel. In the case of fully separable dataset, the main idea of SVM is constructing a hyperplane called decision boundary such that the margin of separator is maximized [11]. This hyperplane can be defined as follow:

$$y(X^{(i)}) = \theta^T X^{(i)}. \tag{17}$$

Hyperplane will separate the data into two classes as positive and negative. For that purpose, if we consider entire data has the label of either 1 or -1, the constraints can be expressed as shown below.

- if $\theta^T X^{(i)} \geq 1 \Rightarrow y(X^{(i)}) = 1$
- if $\theta^T X^{(i)} \leq -1 \Rightarrow y(X^{(i)}) = -1$

which can be rewritten as presented below [12].

$$y(X^{(i)})(\theta^T X^{(i)}) \geq 1 \tag{18}$$

where $X^{(i)}$ is the input vector and θ is a vector of weight parameter values. Given training dataset, here the goal is to find the optimal values of θ to construct a hyperplane which maximizes the margin of separation.

To define the SVM classifier we used `fitcecoc` matlab command which defines multiclass models. This classifier defines a binary learner for each class. For each binary learner, one class is positive, another is negative, and the software ignores the rest. This design performs all combinations of class pair assignments. The objective function for this classifier is log (1 + cross validation loss). Using this function, classifier will be trained and the updated values of unknown parameters of decision boundaries will be calculated. Similar to estimated decision boundaries in LR method, here also we are expecting to have hyper planes separating different zones of dataset. Updated values of θ which are the coefficients of the hyperplanes are shown in (19).

$$\theta = \begin{bmatrix} 0.13 & 3.28 & 10.9 & 18.38 \\ 0.65 & 1.09 & 3.19 & 3.68 \\ -0.39 & 5.65 & 9.43 & 7.45 \end{bmatrix}^T \tag{19}$$

In comparison to values of θ in (16), we got completely different set of parameters calculated by SVM. Using above coefficients, and what is explained in (17), decision boundaries estimated by SVM have been illustrated in Figure 6 and Figure 7. As we see in these figures, the accuracy of SVM classifier is good enough to separate the different zones of dataset precisely. Calculated prediction accuracy on testing data is 0.992 which can be verified via mentioned figures.

By comparing the results represented in Figures 4 to 7, it can be concluded that for a particular sat of data, SVM provides a higher accuracy. This is because SVM can divide the data points belonging to different categories through a clear gap which is as wide as possible, while LR can have different decision boundaries that are near the optimal point.

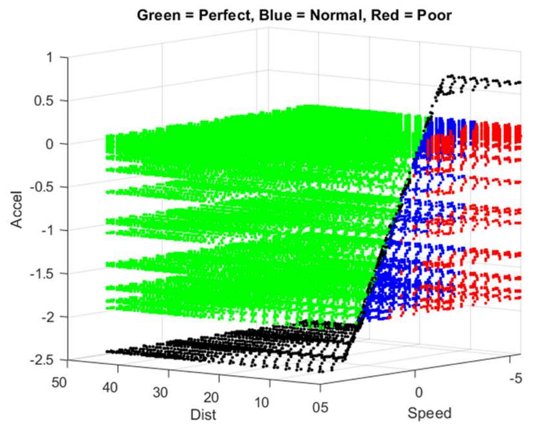

Figure 6: Black SVM decision boundary between class Perfect and Normal

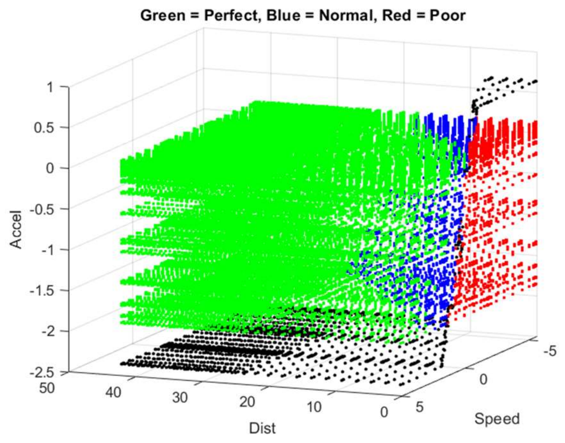

Figure 7: Black SVM decision boundary between class Normal and Poor

5 Conclusion

This paper applies LR and SVM methods to predict and classify driver's attention level. These classifiers are discussed through this paper and their performances are illustrated and compared. According to the figures and testing data accuracies, it is verified that SVM represents better performance; in other words, more accuracy. This is mainly because linear SVM divides different categories of data through some hyperplanes which maximize the margin of separation.

References

[1] Royal Society for the Prevention of Accidents. "DRIVER FATIGUE AND ROAD ACCIDENTS A LITERATURE REVIEW and POSITION PAPER". February 2001. Archived from the original (PDF) on 2017-03-01. Retrieved 2017-02-28.

[2] Wikiped "4.1.03. Driver Drowsiness Detection System for Cars". Retrieved 2015-11-05.

[3] Shen Xu; Ruoqian Liu; Dai Li; Murphey, Y. L,' A hybrid system ensemble based time series signal classification on driver alertness detection' The 2011 International Joint Conference on Neural Networks, July 2011, pp.2093-2099

[4] Fletcher, Luke; Zelinsky, Alexander, 'Driver Inattention Detection based on Eye Gaze—Road Event Correlation' The International Journal of Robotics Research, June 2009, Vol.28(6), pp.774-801

[5] https://www.hindawi.com/journals/cin/2014/761047/#literature-review

[6] https://www.coursera.org/learn/machine-learning

[7] A. Farhat and K. C. Cheok, "Improving adaptive network fuzzy inference system with Levenberg-Marquardt algorithm," 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, 2017, pp. 1-6, doi: 10.1109/SYSCON.2017.7934787

[8] A. Farhat, K. Hagen, K. C. Cheok, B. Boominathan, "Neuro-fuzzy-based electronic brake system modeling using real time vehicle data," EPiC Series in Computing. 2019;58:444-453. DOI: 10.29007/q7pr

[9] https://www.mathworks.com/help/optim/ug/fminunc.html

[10] https://utkuufuk.com/2018/06/03/one-vs-all-classification/

[11] A. Osman Kusakci, B. Ayvaz, E. Karakaya, "Towards an autonomous human chromosome classification system using Competitive Support Vector Machines Teams (CSVMT)," Expert Systems with Applications Volume 86, 15 November 2017, Pages 224-234

[12] A. Bustamama , A. Bachtiara , D. Sarwinda, "Selecting Features Subsets Based on Support Vector MachineRecursive Features Elimination and One Dimensional-Naïve Bayes Classifier using Support Vector Machines for Classification of Prostate and Breast Cancer," Procedia Computer Science, Vol.157, 2019, p.450–458