Pattern Recognition/Machine Learning Approach for Tunnel Detection in Automotive Radar

by

Garima Budhani

A thesis submitted in partial fulfillment for the
degree of Master of Technology

under supervision of
Dr. Shobha Sundar Ram
Department of Electronics and Communication Engineering
Indraprastha Institute of Information Technology, Delhi

November 2018

Pattern Recognition/Machine Learning Approach for Tunnel Detection in Automotive Radar

by

Garima Budhani

A thesis submitted in partial fulfillment for the degree of Master of Technology

to

Indraprastha Institute of Information Technology, Delhi

November 2018

Certificate

This is to certify that the thesis titled "Pattern Recognition/Machine Learning Approach for Tunnel Detection in Automotive Radar" being submitted by Garima Budhani (Roll No.- MT16092) to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original work carried out by her under my supervision. In my opinion, thesis has reached the standards fulfilling the requirements of the regulations relating to the degree. The results contained in the thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Date: _____

Dr. Shobha Sundar Ram
Assistant Professor
Department of Electronics and Communication Engineering
Indraprastha Institute of Information Technology, Delhi
New Delhi-110020

Dr. Karthikeyan Rajarathinam
Technical Architect (ADAS BU)
Continental Automotive Pvt Ltd
Bengaluru-560100

Mrs. Smita Nair
Technical Architect (ADAS BU)
Continental Automotive Pvt Ltd
Bengaluru-560100

Abstract

Adaptive cruise control (ACC) and forward collision warning (FCW) are the two main features of advanced driver-assistance systems (ADAS) enhancing safe driving. Forward looking radar detectors form the main building block of these systems. The functionality of such systems are affected by the frequently occurring environmental structures such as bridges, tunnels, guardrails, road signs or other overhead structures. The environmental changes degrades the radar sensors performance as these might cause false targets leading to unnecessary ACC and FCW actions. It is thus important to correctly classify such targets to avoid these kind of situations, thus improving the overall performance. This work proposes a novel method to understand radar data and apply machine learning algorithms to specifically detect tunnels which was identified as a binary classification problem. This work can be extended to other enclosed structures such as closed car-parks.

In our approach, we considered multiple-target detection count and height as the important parameters based on the subsequent observations after analyzing the available experimental radar data. With a suitable scheme, we mapped these parameters to two dimensional matrices using range information which are analogous to a radar grid map. We implemented logistic regression to model and train the classifier with the matrices being the input features. We compared the classifier performance for test data with different combinations of feature matrix dimensions and other parameter values. Results showed that unregularized logistic regression with mini-batch gradient descent using a feature matrix size of 5×5 provided the optimum results with an accuracy of 95%.

Acknowledgements

I sincerely thank my supervisors Dr. Shobha Sundar Ram and Dr. Karthikeyan Rajarathinam and co-supervisor Mrs. Smita Nair for their valuable guidance and constant support which led to the completion of this work.

I would like to dedicate this work to my parents Sh. Jagmohan Budhani and Smt. Deepa Budhani for all their love, care, hard work and the difficulties they faced in bringing me up and making me what I am today. Also, to my brother Vishal Budhani, who always guided me and has been my pillar of strength along all these years.

I would also like to thank Dr. Anshu Gupta for mentoring and supporting me at various stages throughout my tenure in Continental Automotive. Special thanks to Mr. Mangesh Desai and Mr. Abhijeet S. (Continental Automotive) for clearing all my doubts and help me understand various concepts. It was my privilege indeed to work with you all.

Contents

List of Figures

List of Tables

Dedicated to my beloved parents...

Chapter 1

Introduction

Advanced driver-assistance systems (ADAS) targeted towards increased car and road safety are being extensively researched and developed. These systems use a combination of sensors - camera, lidar and radar - whose outputs are processed to automatically generate suitable control actions in response to a dynamic environment. Front-facing systems comprising of these sensors are an integral part of ADAS and provide safety functions such as forward collision warning, adaptive cruise control, emergency brake assist, pedestrian detection and traffic sign recognition among others [1]. To ensure a smooth functioning, it is required that the environment surrounding the ego vehicle (vehicle on which sensor is mounted) should be modelled correctly to detect potential targets efficiently. While identification of targets like vehicles, pedestrians, etc is important to avoid collisions ensuring driver safety, structures like bridges, road-signs, tunnels, guardrails, road-side barricades, etc. are also necessary to be classified correctly as they often affect the functionality of ADAS. If not properly sensed, these might be wrongly identified as potential targets ahead of the ego vehicle resulting in unnecessary control actions, such as emergency brake assist or forward collision warning, thereby affecting the overall ADAS performance. To avoid such situations and to improve the performance, it is necessary to correctly classify scenarios such as a high-traffic lane in a city, a parking area or an under-bridge.

Through our work, we have approached this problem by focusing on classifying tunnels. We propose to use supervised machine learning algorithms on automotive radar data to identify the presence of tunnels. Though, machine learning techniques have been applied for several autonomous driving applications such as pedestrian detection [2, 3], street-side vehicle detection [4] and prediction of change in driver intentions [5, 6], there has not been any prior work on tunnel detection using machine learning approaches. On the other hand, camera images can be more helpful for classification provided their structure

details and color maps but we chose to use the radar data due to it's advantages over camera. Radar gives accurate position and Doppler information which is not obtained in camera. For example, using the Doppler information provided by the radar sensor, we can differentiate between stationary and moving targets but the same cannot be achieved with the help of a camera sensor. Secondly, performance of a camera sensor is limited by adverse weather conditions. Varying lighting conditions also affect the camera performance; same set of images cannot be used for training a classifier for all the time instances. For example, the camera pictures during the day time cannot be used for detecting the tunnels in the night.

1.1 Objective

The objective of our work is to perform a binary classification on the processed radar data obtained from Continental's proprietary simulation tool to identify the presence or absence of tunnels. Specifically, we implemented regularized and un-regularized logistic regression with two gradient descent methods namely batch and mini-batch GD. Classifier performance was evaluated in terms of performance metrics namely accuracy, precision, sensitivity and specificity. While we achieved a test accuracy as high as 95%, improved results can be obtained by further improving the input features and applying other classification algorithms like support vector machines and neural networks.

1.2 Proposed Method

Our proposed methodology includes three steps - data analysis, feature extraction and implementation - and is summarized in Figure (1.1).

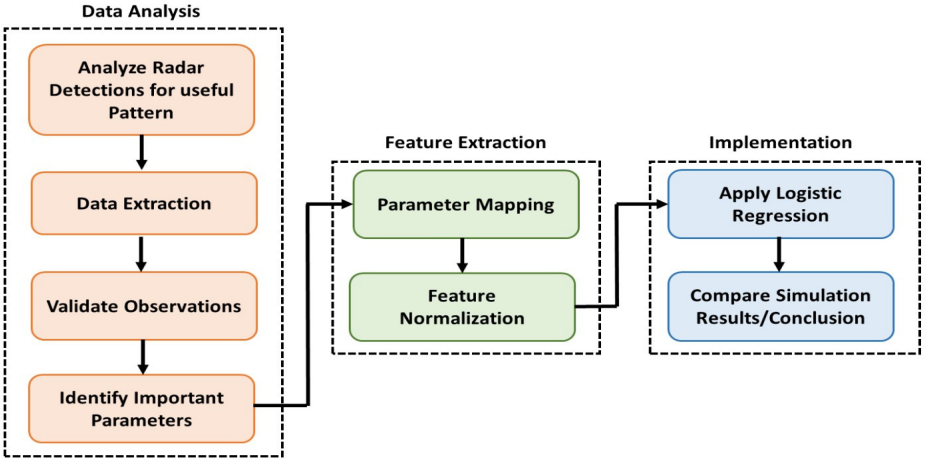


FIGURE 1.1: Flowchart depicting the process followed

- **Data Analysis:** The goal of this step was to analyze radar detections from available real-time recordings to find special patterns that can distinguish tunnel from open space. It was observed that multiple-target detections were significantly high inside the tunnel unlike the open space. Multiple-target detection count and height were identified as important parameters and considered for generating the required input feature. This has been explained in detail in Chapter 2.

- **Feature Extraction:** Using the range information, multiple-target detection count and height were mapped to two dimensional matrices which served as the input feature for classification. Feature matrix size was chosen for the optimal classifier performance. This has been discussed in Chapter 2.

- **Implementation:** The problem of tunnel detection was identified as a binary classification problem. We implemented logistic regression and tested the performance for various cases as discussed later in Chapter 4. Batch and mini-batch gradient descent were used to test for an optimal performance with different values of other relevant parameters. Chapter 3 discusses the classifier model for logistic regression and the gradient descent methods.

We have showed that unregularized logistic regression with mini-batch gradient descent and a feature matrix size of 5×5 provided the optimum results. It has also been shown that smoothing the predicted output reduced the fluctuations in the output.

Chapter 2

Parameter Selection for Pattern Recognition

In this chapter, we discuss the experimental data obtained from Continental's proprietary simulation tool for pattern recognition for tunnel detection. The simulation tool processes radar sensor data pertaining to specific camera recordings of real-time scenarios to generate a set of radar detections. The video recordings provide the necessary vision while analyzing the output detections on the simulation tool. The radar data are modeled after the ARS-441 sensor manufactured by Continental Automotive [7]. It is an automotive radar with an operating frequency of 77 GHz. The ARS-441 is used for realizing several important functions including adaptive cruise control, emergency brake assist and forward collision warning. For an approximate cycle time of 60 ms, the radar sweeps two separate scans:

- **Far Range (FR) Scan:** Maximum range is 250 m with a field of view (FOV) of $\pm 9^{\circ}$
- **Near Range (NR) Scan:** Maximum range is 70 m with a FOV of $\pm 45^{\circ}$

The processed output of the tool includes two separate lists of detections, FR detections and NR detections, per scan. Each detection is attributed with the following parameters:

- Longitudinal range (m)
- Lateral range (m)
- Relative velocity (m/s)
- Radar cross-section (RCS, dBsm)

- Azimuth (ϕ, rad)
- Elevation (θ, rad)
- Probability of detection (P_d)

A valid detection is one which is within the FOV and whose RCS is greater than the threshold. Any two detections are considered as distinct single target detections if they can be resolved along range, azimuth or velocity. In other words, if

$$|r_1 - r_2| > \Delta r$$
$$|v_1 - v_2| > \Delta v$$
$$|\phi_1 - \phi_2| > \Delta \phi.$$

Here, r_n, v_n, and ϕ_n represent the range, velocity and azimuth of the n^{th} detection. Δr, Δv and $\Delta \phi$ are the sensor resolutions in terms of range, velocity and azimuth respectively. When they do not satisfy the above criteria, the detections are termed as *multiple-target* detections. The output set of detections provided by the simulation tool can be visualized in a grid view [8] as shown in Figure. 2.1. Each data point in the grid is a radar detection and the position of the ego vehicle is at the origin. The number of detections and their associated parameter values vary scan to scan. Single-target and multiple-target detections are represented with different colors.

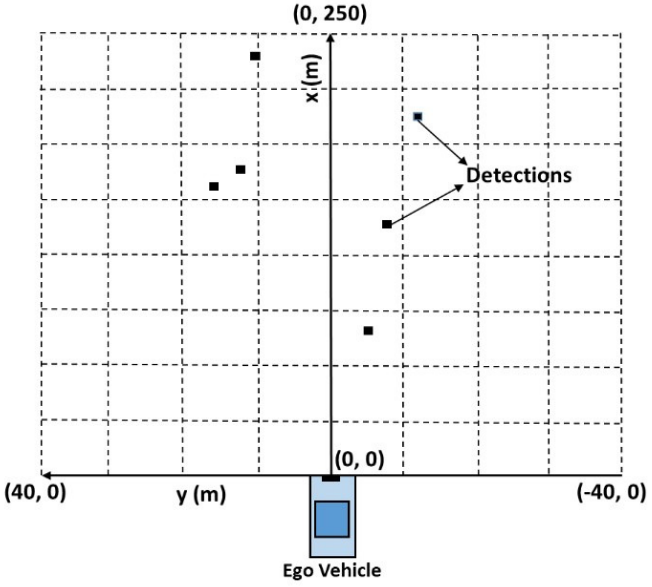


FIGURE 2.1: Radar detections as seen in a grid view

For our work, we analyzed 11 camera recordings of different durations. The subsequent observations, discussed in the following section, are with respect to valid stationary

detections since the tunnels form static targets. Moving detections have not been considered.

2.1 Choice of parameters

2.1.1 Count of total target detections

Figure. 2.2 show the distribution of target detections obtained from FR scans for five different scenarios. The lateral range of the detections, along y axis, spans from 0 to 80m and the position of the ego vehicle is now shifted to $(40, 0)$ m. Figure. 2.2(a,b) and (c,d) show two scenarios when the ego vehicle is in open space and is approaching a tunnel ahead. Total number of multiple-target detections in the first instant is 8 and in the second case it is 17. Figure. 2.2(e,f) shows the instant when the ego vehicle is at the entrance of the tunnel. The total detection count increases to 45 in this case. Figure. 2.2(g,h) depicts the vehicle inside the tunnel with total detections equal to 50. Figure. 2.2(i,j) depicts the vehicle in open space again, just after exiting the tunnel. The total detections reduces to 23. The detections in this case are slightly more towards the right side of the vehicle and can be attributed to the presence of a metallic barricade.

2.1.2 Multiple target detections

The pattern that we observe here is that the multiple-target detections are significantly higher when the ego vehicle is inside the tunnel. These detections start increasing as the vehicle approaches the entrance of the tunnel, remains high in numbers while the vehicle is inside the tunnel and decrease towards the exit. This pattern was found to be consistent for other available recordings as well. To verify the visual observations, we extracted the processed data provided by the simulation tool and segregated the multiple-target detections. Figure. 2.3a-c show the variation of multiple-target detection count for three different recordings. Here, x axis shows the different FR scans that were recorded. The time resolution between two subsequent frames is approximately 60 ms. The speed of the ego vehicle varies between 100 km/h (27.8 m/s) to 120 km/h (33.3 m/s). We observe that the multiple-target detections remain consistently high inside the tunnel as compared to the open space.

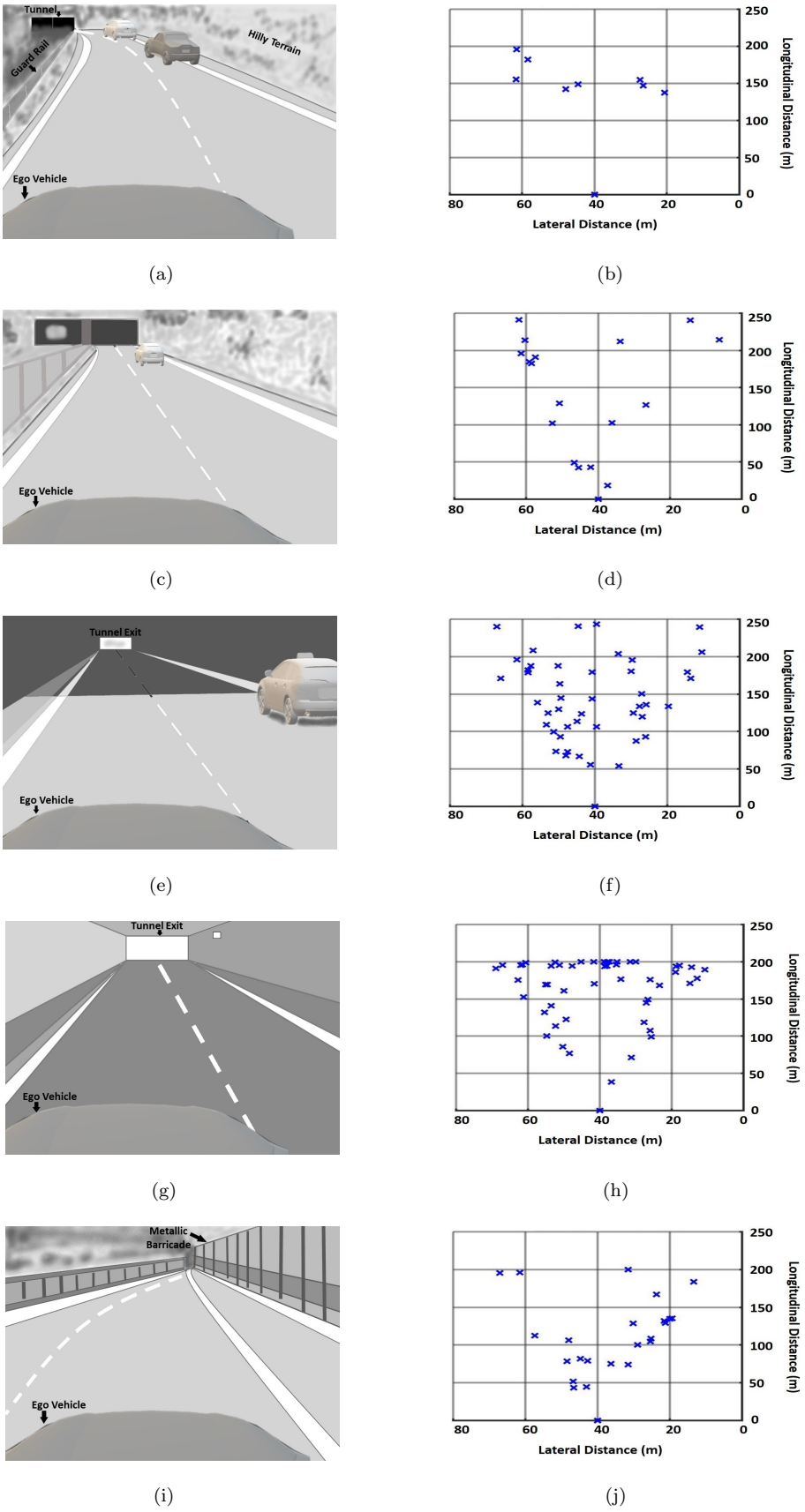
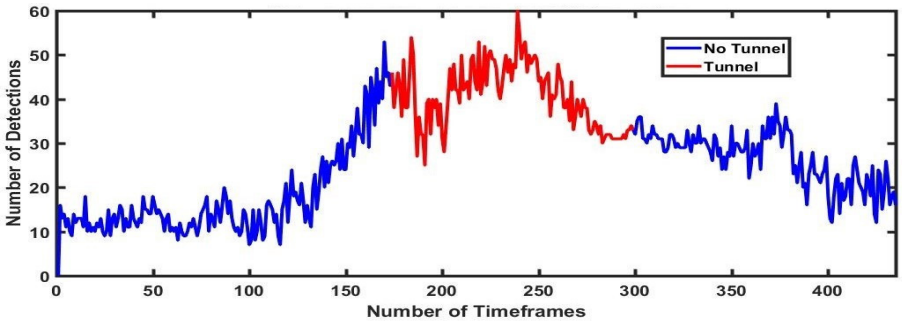
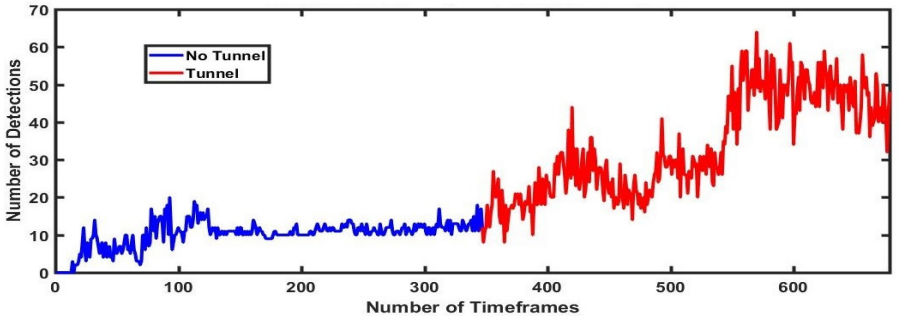


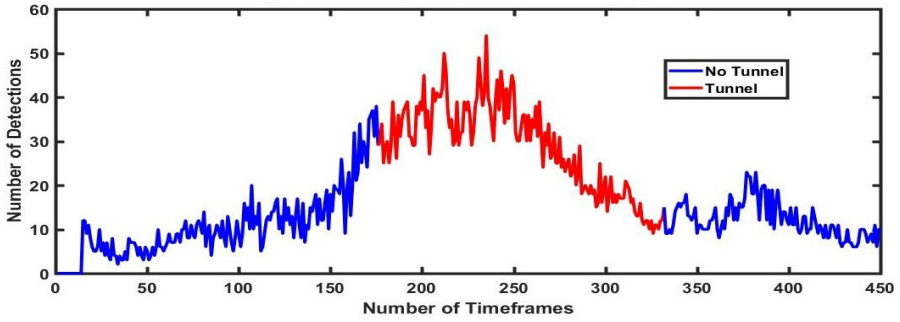
FIGURE 2.2: Scenarios: Ego vehicle (a,b) in open space, (c,d) approaching tunnel, (e,f) entering tunnel, (g,h) inside the tunnel and (i,j) leaving the tunnel. Figures (a,c,e,g,j) show the camera recordings. Figures (b,d,f,h,j) show the radar detections.



(a)



(b)



(c)

FIGURE 2.3: Multiple-target detection count variation for three different recordings

2.1.3 Single target detections

The single-target detection counts of three recordings are studied in Table 2.1 to see if there is any pattern of increase in counts when the car is inside a tunnel. In the first recording, the average number of single-target detections both inside and after leaving the tunnel are comparable. For the second recording, the average is highest when before the vehicle enters the tunnel and has decreased significantly inside the tunnel. There is similarly no consistent pattern in the third recording. Hence, we conclude that the variation in the single-target detection count cannot be used for tunnel detection.

TABLE 2.1: Average Number of Single-Target Detections

	Rec 1	**Rec 2**	**Rec 3**
Before Tunnel	24	44	34
Inside Tunnel	41	18	32
After Tunnel	43	–	45

2.1.4 Target height

Another parameter that showed some type of correlation with the presence of a tunnel was the target height that we calculated using the elevation and radial range of these detections. We plotted the sum of heights of the multiple-target detections per scan against total number of scans for the above three recordings. It showed a similar variation as observed for the detection count remaining significantly high inside the tunnel. Target height was hence used as the second important parameter.

2.1.5 RCS

We also verified the possibility of using RCS as a parameter. Figure. 2.4a-c show the average RCS variation for the three recordings discussed earlier. It can be seen that the variations of RCS do not show any consistent pattern. Hence RCS was not considered.It should be noted that the results included here are for only three recordings but have been verified for others as well.

2.2 Formation of feature vectors

Consider a radar scenario presented in Figure.2.5a. The resulting multiple-target detection counts distribution for one FR scan is shown in Figure.2.5b. We map the multiple-target detection count into a two-dimensional matrix. The size of the matrix is chosen for optimal classification performance and is discussed in further chapters. The mapping process to convert the multiple-target detection count to a $n_x \times n_y$ matrix is explained in the following steps.

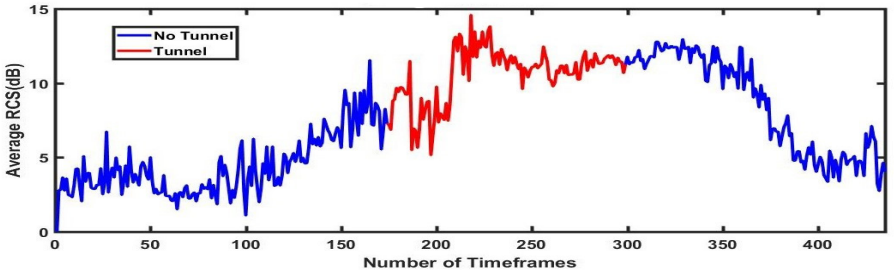
1. For each time frame, the lateral range of a multiple-target detection is shifted from $(-40, 40)$ m to $(0, 80)$ m

2. The longitudinal and lateral range axes are of size 80×250. The axes values are scaled with f_x and f_y which are given in

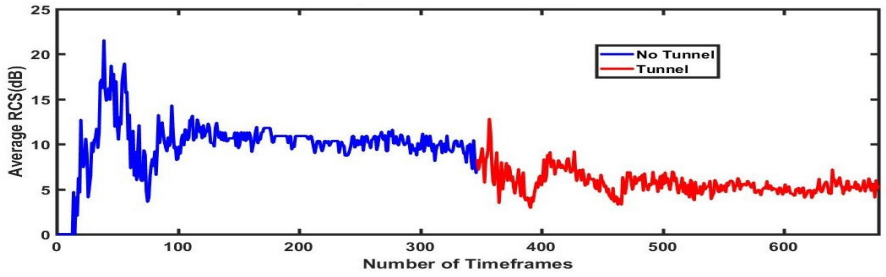
$$f_x = \frac{250}{n_x}$$
$$f_y = \frac{80}{n_y}$$

3. Multiple-target detections are mapped to a particular cell of the matrix if the corresponding scaled down range values lie within that cell. Each k^{th} cell, thereby, consists of n_k number of multiple-target detections.

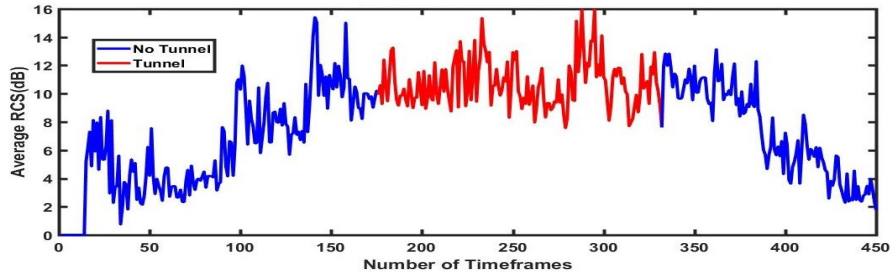
Target height parameters are also mapped in a similar manner to a similar sized two-dimensional matrix and h_k represents the sum of heights of all the multiple-target detections lying in the k^{th} cell. The matrices now represent two images which will be used as input features for training the classifier discussed in the next chapter.



(a)



(b)



(c)

FIGURE 2.4: Average RCS variation of multiple-target detections for three different recordings

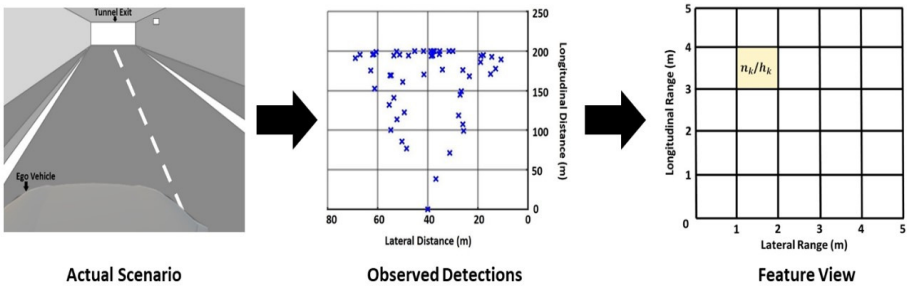


FIGURE 2.5: Schematic of transformation from actual scenario to the feature level

Next chapter discusses logistic regression and gradient descent methods.

Chapter 3

Logistic Regression

We identify the problem of tunnel detection as a binary classification problem where we need to classify whether or not the ego vehicle is inside a tunnel. We use logistic regression which is a simple and commonly used supervised learning algorithm for classification. Other algorithms like support vector machines and neural networks may be considered in future.

For a binary classification problem, given a set of input values represented by a one dimensional vector $x = [x_1, x_2, ...x_n]$, the predicted (or estimated) output represented by $\hat{y}$ should be either 0 or 1. While '0' represents negative class, '1' represents positive class. $\hat{y}$ will be predicted based on a mathematical function which maps the input vector to the output [9]. This mathematical function is called a hypothesis function.

The general form of the hypothesis function is given as

$$h_\theta(x) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i \tag{3.1}$$

where, $\theta = [\theta_0, \theta_1, ...\theta_n]$ is called the learning parameter vector. For a given θ, the hypothesis function $h_\theta(x)$ tries to map the input variable to the output. Here, θ_0 is called the bias variable. For binary classification, $h_\theta(x)$ is required to be modified such that it satisfies $0 \leq h_\theta(x) \leq 1$. Hence, (3.1) modifies as below

$$h_\theta(x) = g(\theta_0 + \sum_{i=1}^{n} \theta_i x_i) \tag{3.2}$$

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

$$z = (\theta_0 + \sum_{i=1}^{n} \theta_i x_i) \tag{3.4}$$

Here, $g(z)$ represents a sigmoid function whose value lies between 0 and 1 for any value of z. Figure. 3.1 shows a graphical plot of $g(z)$. In vector form, $z = x * \theta^T$.

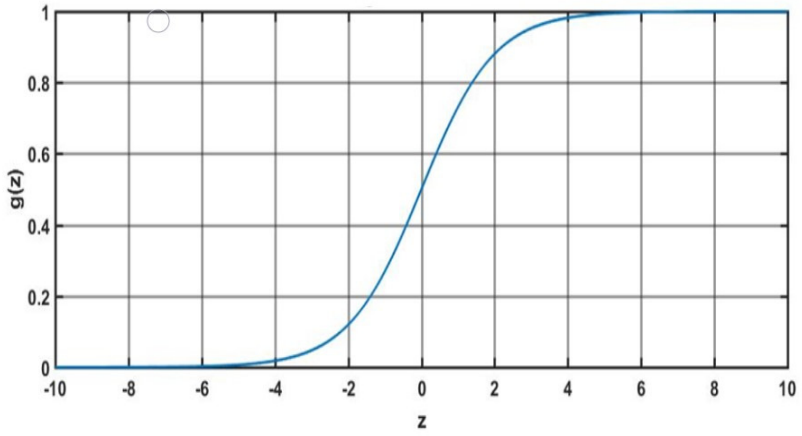


FIGURE 3.1: A typical graph of a sigmoid function

Thus $h_\theta(x)$ gives the probability that the output is 1. For example, $h_\theta(x) = 0.65$ signifies that the probability of output being 1 is 65%.

$$h_\theta(x) = P(y = 1|x; \theta) = 1P(y = 0|x; \theta) \tag{3.5}$$

The output vector $\hat{y}$ is now defined as

$$\hat{y} = 1, for h_\theta(x) \geq 0.5 \tag{3.6}$$

$$\hat{y} = 0, for h_\theta(x) < 0.5 \tag{3.7}$$

where, $h_\theta(x) = 0.5$ is the decision threshold. The training data x is of size $(m, n + 1)$, where m is the size of the data set and n is the total number of features for one data sample. An additional column is added with all values equals to 1 to compensate for the bias term as mentioned earlier. Now, $h_\theta(x)$ and $\hat{y}$ are the hypothesis and output vector respectively of size $(m, 1)$. Similarly, the label vector y (ground truth) will also be of size $(m, 1)$.

3.1 Cost Function

The cost function, also called as loss function, is a measure of error incurred while predicting the output using the hypothesis function. It needs to be optimized to obtain a minimum value. For a single input data sample, we define our cost function as

$$\text{Cost}(h_\theta(x), y) = -log(h_\theta(x)) \text{ , if y = 1;}$$

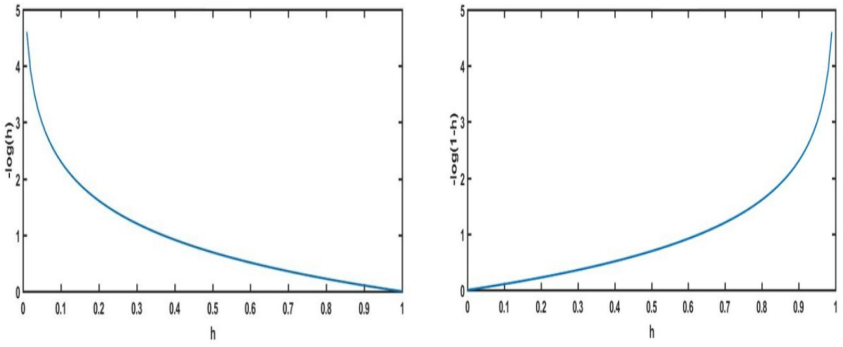
$$\text{Cost}(h_\theta(x), y) = -log(1 - h_\theta(x)) \text{ , if y = 0}$$

Figure. 3.2(a,b) show the graphical plot for these two. This can be explained as follows. When $y = 1$, $h_\theta(x))$ should also be closer to 1, thus keeping the cost to minimum. If $h_\theta(x) << 1$ for$y = 1$, the error will be high. For $y = 0$, it is exactly opposite. When combined together, an overall cost function $J(\theta)$ is obtained as

$$J(\theta) = -y * log(h_\theta(x)) - (1 - y) * log(1 - h_\theta(x)) \tag{3.8}$$

Figure. 3.3 shows the graphical plot for the overall cost function $J(\theta)$. For m number of input data samples (3.8) modifies to

$$J(\theta) = -\frac{1}{m}\sum_{j=1}^{m}(y^j * log(h_\theta(x^j)) + (1 - y^j) * log(1 - h_\theta(x^j))) \tag{3.9}$$



(a) (b)

FIGURE 3.2: Graphical plot for cost function for (a) y = 1, and (b) y = 0

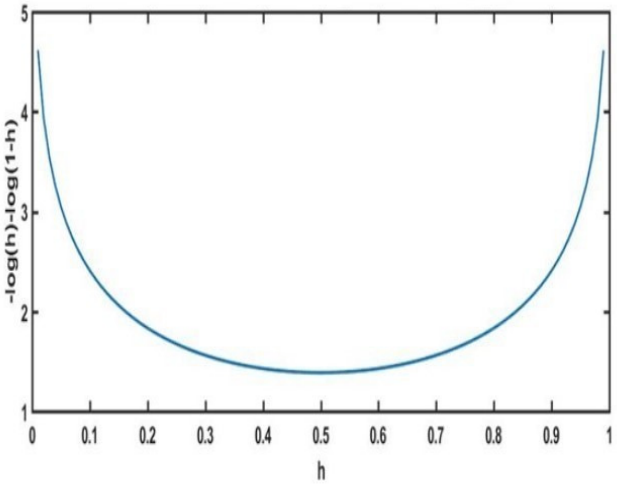


FIGURE 3.3: Graphical plot for overall cost function

3.2 Gradient Descent

Gradient descent (GD) is an optimization algorithm which is used to find the minimum value of a function. Starting from a random point, it takes steps proportional to negative of the approximate gradient of the function at the current point till the minimum value is reached. A convex function is generally preferred which has a single minima, instead of a non-convex function which might have a number of local minima. This is because, for a non-convex function, the algorithm may get stuck on a local minima instead of converging to a global minima.

In logistic regression for binary classification, GD serves two purposes. It optimizes the cost function as given by (3.9) and estimates the best possible value of the learning parameter vector which is initially set to 0. For an input data set x of size $(m, n+1)$, GD algorithm updates the learning parameter vector θ until the minimum value of cost function $J(\theta)$ is reached. It is represented in vector form as

Repeat until convergence:

$$\theta := \theta - \frac{\alpha}{m} * (x^T * (x * \theta^T - y)) \tag{3.10}$$

$$\frac{1}{m} * (x^T * (x * \theta^T - y) = \Delta J(\theta) = \text{gradient ;}$$
$$\alpha = \text{learning rate, m = number of input data samples}$$

This algorithm is also called as batch GD where the batch size for each iteration is equal to m. The value of the cost function after updating θ in each iteration is recorded. A plot of the cost function against number of iterations is used to check whether or not the function value is converging to a minimum. A very small value of α can make GD slow whereas too high of a value can make it to overshoot or undershoot making it either slow to converge to the minima or not converge at all even after many iterations. Another variation is mini-batch GD in which instead of the whole input data, 'b' number of samples are considered at each time for each iteration. The mini-batch size is b. It is represented as

Repeat until convergence:

$$\text{for } (1 : b : m)\{\theta := \theta - \tfrac{\alpha}{b} * (x_b^T * (x_b * \theta^T - y_b))\}$$

Generally the values of b are taken as power of 2. Mini-batch GD requires lesser number of iterations as compared to batch GD and α is also small. If the input data size is not too large, mini-batch GD can perform better than batch GD. It is not preferred for large data sets as it slows down the convergence.

Due to varying range of input feature values, GD may oscillate unnecessarily as it descends faster for small ranges and slower for large ranges. This can be overcome by normalizing the feature values for entire data set and thus speed-up GD. This is achieved as

$$x = \frac{x-\mu}{\sigma} \text{ ; } \mu = \text{mean , } \sigma = \text{standard deviation}$$

Regularization is a measure to overcome the problem of over-fitting or high variance in case of which, the classifier performs quite well for the training data but poorly for the test data. Applying regularization modifies (3.9) to

$$J(\theta) = -\frac{1}{m}\sum_{j=1}^{m}(y^j * log(h_\theta(x^j)) + (1-y^j) * log(1-h_\theta(x^j))) + \frac{\lambda}{2m} * \sum_{i=1}^{n}\theta_i^2 \quad (3.11)$$

and gradient $\Delta J(\theta)$ is modified as

$$\Delta J(\theta) = \frac{1}{m} * [(x^T * (x * \theta^T - y) + \lambda\theta]$$

Here, λ is the regularization parameter. If a certain feature value is causing the hypothesis function to over fit, it's effect is reduced by inflating the cost for the corresponding weight (learning parameter) using regularization. As a result, the equivalent gradient value will also increase, there by reducing the weight while updating it's value in a GD iteration. The bias term θ_0 is not updated with regularization.

Parameters like gradient descent iterations, batch size, learning rate (α), regularization factor (λ), etc. are called hyper-parameters. There is no fixed set of values of hyper-parameters and they need to be tuned to optimize the output for a specific algorithm. This is done by trying different set of values randomly and choosing the one which gives the optimal result.

Next chapter discusses the simulation results.

Chapter 4

Simulation Results

We implemented logistic regression for detecting tunnels using the classifier model defined earlier. The training set was generated by combining the data samples for three different sensor recordings. The samples were labelled with a file that contained the beginning and end time stamps for each tunnel. Different test sets were also generated using the available data and labelled in a similar way. For each training and test sample, we had two feature matrices obtained from multiple-target detection counts and target heights. Each matrix was rolled out into a one dimensional vector and by combining both, we obtained a single input feature vector of size $(2(n_x \times n_y))$. Hence, for m_{train} training and m_{test} test samples, the corresponding data set size would be $(m_{train}, n+1)$ and $(m_{test}, n+1)$. An additional bias value equal to 1 was added in each feature vector.

4.1 Evaluation Metrics

Figure. 4.1 shows a confusion matrix for classification outcomes identified as true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) . These are defined as follows:

- **TP**: Number of samples actually belonging to class 1 and predicted as class 1 by the classifier.
- **FP**: Number of samples actually belonging to class 0 but predicted as class 1 by the classifier.
- **TN**: Number of samples actually belonging to class 0 and predicted as class 0 by the classifier.

- **FN**: Number of samples actually belonging to class 1 but predicted as class 0 by the classifier.

Based on these values, the performance of a classifier is then evaluated in terms of different performance metrics as described below.

1. **Accuracy:** It refers to the total number of correct predictions out of the total number of predictions made by the classifier.

$$\text{Accuracy} = \frac{TP+FP}{TP+FP+FN+TN}$$

Accuracy is a good measure when the data is balanced in terms of the classes, i.e. data contains equal number of samples for each class. But in case of skewed data, where samples of one class are more than the other, accuracy cannot alone be counted upon. For example, in a binary classification problem, if the training data set contains 98% of negative (class 0) samples and 2% of positive (class 1) samples, a training accuracy as high as 98% can be achieved even if the classifier model predicts all samples to be negatives. Performance in such cases is evaluated with other metrics which are explained next.

2. **Sensitivity:** Also known as recall or true positive rate, it refers to the total number of correct predictions out of the total actual positive.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.1}$$

3. **Specificity:** Also known as true negative rate, it refers to the total number of correct predictions out of the total actual negatives.

$$Specificity = \frac{TN}{TN + FP} \tag{4.2}$$

4. **Precision:** It refers to the total number of correct predictions out of the total positive predictions made by the classifier.

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Sensitivity, specificity and precision should be high, ideally equal to 1, for an efficient classifier.

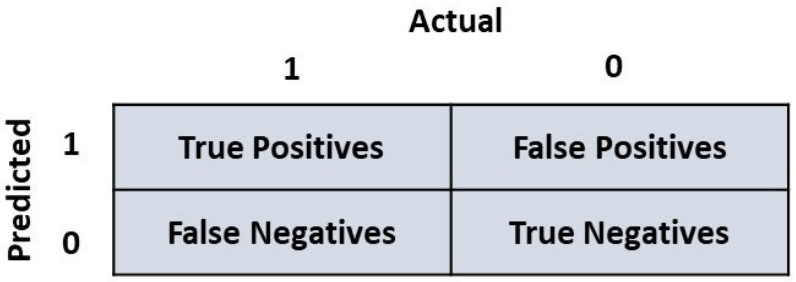


FIGURE 4.1: Confusion matrix for binary classification

4.1.1 Receiver operating characteristic curve

The receiver operating characteristic (ROC) curve is another way to examine the diagnostic ability of a classifier [10]. It is a graphical plot of sensitivity or true positive rate (TPR) against the false positive rate (FPR) for different values of decision thresholds varying between 0 and 1. FPR is also called as fall-out or probability of false alarm and is given as:

$$\text{FPR} = 1 - \text{Specificity} = \frac{FP}{TN+FP}$$

Figure. 4.2 depicts ROC curves for different classifiers. For a perfect classifier the area under the curve is equals to 1 and (0, 1) represents the best prediction [11]. In the ideal condition, when the decision threshold is equals to 0, both TPR and FPR will be equal to 1 as all the samples in the data set will be detected as positives. This is denoted by (1, 1) in the graph. As the decision threshold is increased gradually, the FPR decreases without affecting the TPR and finally both converge to 0 as the threshold becomes equal to 1. The ROC curve of a good classifier should be as close to the ideal curve as possible with the area under the curve closer to 1. In Figure. 4.2, the diagonal line is called as the no-discrimination line representing a classifier that makes random predictions. In this case, TPR and FPR are always equal for all the values of threshold and the area under the curve is 0.5. A good classifier must provide an ROC curve that is above the diagonal.

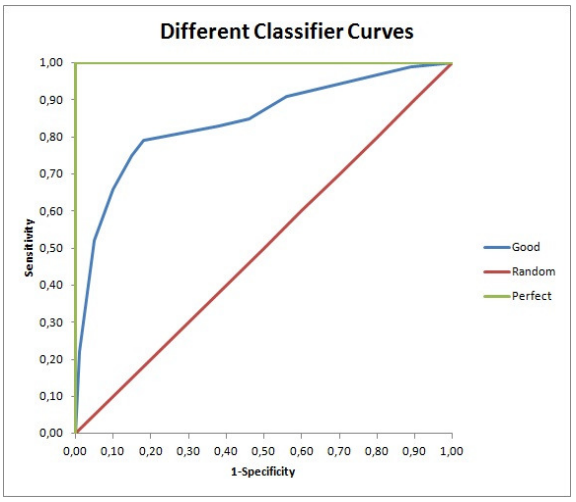


FIGURE 4.2: ROC curve for different classifiers

4.2 Results

The simulations results presented here include the following:

1. **Case 1:** Performance comparison for two different feature matrix dimensions for first test set.
2. **Case 2:** Effect of applying smoothing on the predicted output.
3. **Case 3:** Performance comparison for second test set before and after filtering the height data with optimal parameters.

4.2.1 Effect of matrix size, regularization and batch size

In this subsection, we compare the classifier performance for the first recording test with two different feature matrix dimensions: 5×5 and 7×4 respectively. Results were first obtained using logistic regression (LR) with batch gradient descent (GD). Then the exercise was repeated using regularization and mini-batch GD. The training data set consists of 2443 samples and the test set of 743 samples. Table 4.1 shows the values of the various hyper-parameters used with a matrix size of 5×5. Table 4.2 (a) compares the metrics obtained for the test set for unregularized LR with batch and mini-batch GD and regularized LR with batch GD. Table 4.3 (a) shows the confusion matrices for these three cases. Except a slight reduction in accuracy, other metrics are almost same for regularized LR with batch GD as compared to unregularized LR with batch GD. Regularization does not improve the results here. Unregularized LR with mini-batch

GD shows a slight improvement in terms of accuracy, specificity and precision as compared to unregularized LR with batch GD. Figure. 4.3 shows the ground truth versus estimated output for the test set obtained using unregularized LR with mini-batch GD. The result plots for remaining two cases did not show many distinctions and hence are not shown. Fluctuations can be noticed at the entrance and exit of the tunnel. This might be attributed to the sudden increase in multiple-target detections as the vehicle approaches the entrance and then decrease towards the exit as observed earlier.

TABLE 4.1: Hyper-parameter values for case 1

Hyper-Parameters	**Un-Reg/Batch GD**	**Un-Reg/Mini-Batch GD**	**Reg/Batch GD**
Learning Rate (α)	0.1	0.01	0.1
No. of GD Iterations	2500	1500	2500
No. of Training Samples	2443	2443	2443
No. of Test Samples	743	743	743
Feature Matrix Dimension	5 x 5	5 x 5	5 x 5
Regularization Parameter (λ)	-	-	30
Mini-Batch Size	-	64	-

TABLE 4.2: Performance metrics for test set in case 1

(a) Matrix dimension: 5x5

Metric (%)	**Un-Reg/Batch GD**	**Un-Reg/Mini-Batch GD**	**Reg/Batch GD**
Accuracy	95	95.2	94.8
Sensitivity	87.7	87.3	87.2
Specificity	97.9	98.3	97.9
Precision	94.4	95.4	94.3

(b) Matrix dimension: 7x4

Metric (%)	**Un-Reg/Batch GD**	**Reg/Batch GD**	**Reg/Mini-Batch GD**
Accuracy	94.7	95	95.6
Sensitivity	92	92	91.5
Specificity	95.8	96.4	97.2
Precision	89.8	91	92.8

TABLE 4.3: Confusion matrices for test set in case 1

(a) Matrix dimension: 5x5

Un-Reg/Batch GD		**Un-Reg/Mini-Batch GD**		**Reg/Batch GD**	
TP: 186	**FP:** 11	**TP:** 185	**FP:** 9	**TP:** 185	**FP:** 11
FN: 26	**TN:** 520	**FN:** 27	**TN:** 522	**FN:** 27	**TN:** 520

(b) Matrix dimension: 7x4

Un-Reg/Batch GD		**Reg/Batch GD**		**Reg/Mini-Batch GD**	
TP: 195	**FP:** 22	**TP:** 195	**FP:** 19	**TP:** 194	**FP:** 15
FN: 17	**TN:** 509	**FN:** 17	**TN:** 512	**FN:** 18	**TN:** 516

For feature matrix size 7×4, Table 4.2 (b) compares the metrics obtained using unregularized LR with batch and regularized LR with batch and mini-batch GD. Table 4.3 (b) shows the confusion matrices for these three cases. Accuracy for unregularized LR with batch GD is slightly decreased to to 94.7% as compared to 5×5 where it was 95%. Though sensitivity increased from 87.7% to 92% due to decreased false negatives and slightly increased true positives, false positives doubled to 22 which lead to decrease in specificity and precision. Figure. 4.4(a) shows the fluctuations in the open space before the vehicle's entry in the tunnel. Applying regularization with regularizing parameter value equals to 50 slightly improves the performance in terms of accuracy, specificity and precision. Regularized LR with mini-batch GD with regularizing parameter value equal to 5 gave the optimum result for this matrix size. Figure. 4.4(b,c) show the reduced fluctuations in the predicted output for regularized LR with batch and mini-batch GD respectively.

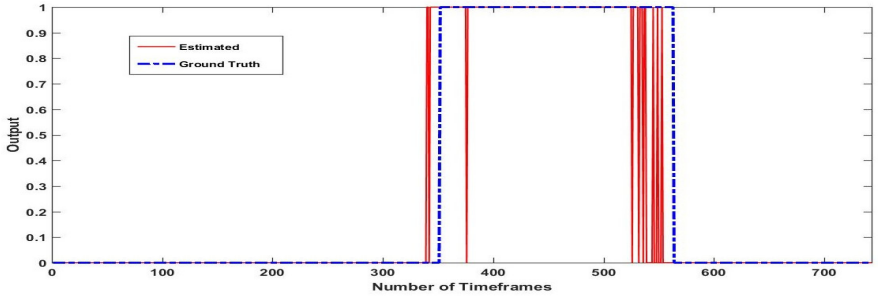
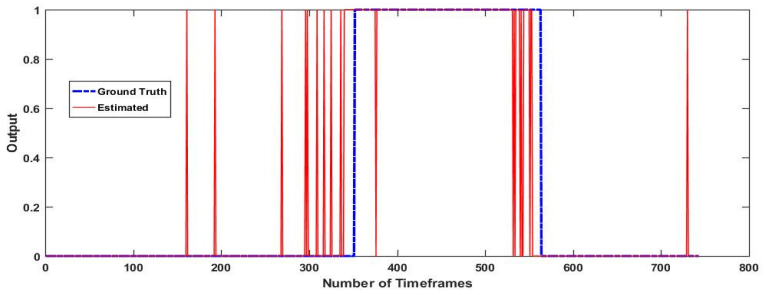
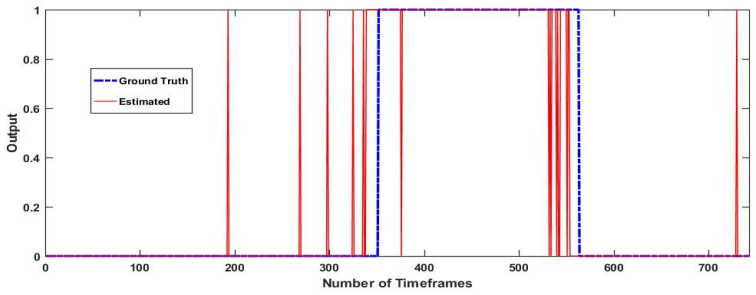


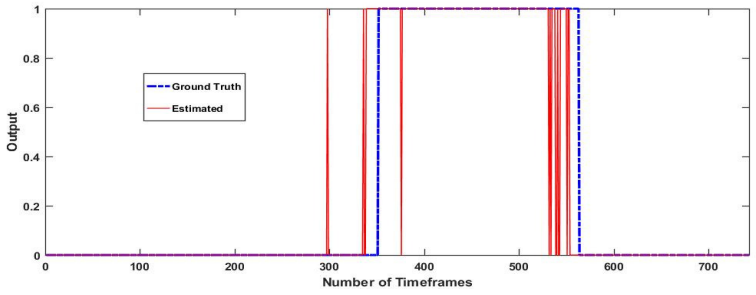
FIGURE 4.3: Estimated output v/s ground truth for first test set using unregularized LR with mini-batch GD and feature matrix size 5x5



(a)



(b)



(c)

FIGURE 4.4: Estimated output v/s ground truth for first test set with feature matrix size 7x4 for (a) un-regularized LR with batch GD, (b) regularized LR with batch GD, and (c) regularized LR with mini-batch GD

We tried using other feature matrix sizes like 5×4, 8×5 and 9×6, with different combinations of hyper-parameter values. The optimum result was obtained with matrix size 5×5 using unregularized LR with mini-batch GD. We concluded that increasing the matrix size beyond a certain limit makes the distribution of matrix values more sparse and this might be a reason of no improvements in the classifier performance.

4.2.1.1 ROC curve for optimal feature size

Figure. 4.5 shows the ROC curve generated using the training data with a feature matrix size of 5×5 and other parameters being same as discussed earlier (Table 4.1). A

zoomed-in section shows the curve for unregularized LR and regularized LR with batch GD lie below the curve for unregularized LR with mini-batch GD, signifying it's better performance over the other two.

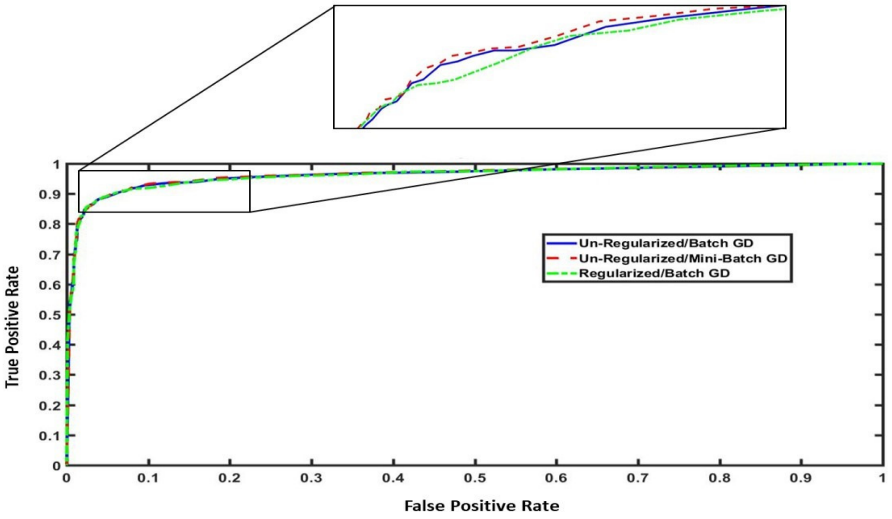


FIGURE 4.5: ROC curve for optimal feature matrix size 5×5

4.2.2 Effect of smoothing

In order to overcome the fluctuations in the predicted output, we applied moving average method of smoothing [12] (see Appendix A) on the predicted classification probability before comparing it to the detection threshold. Table 4.4 (a) and (b) shows the performance metrics and confusion matrices respectively for the first test set with the optimal parameters (as discussed earlier), before and after smoothing. Though the metrics are comparable, a smooth predicted output was obtained as shown in Figure. 4.6(b).

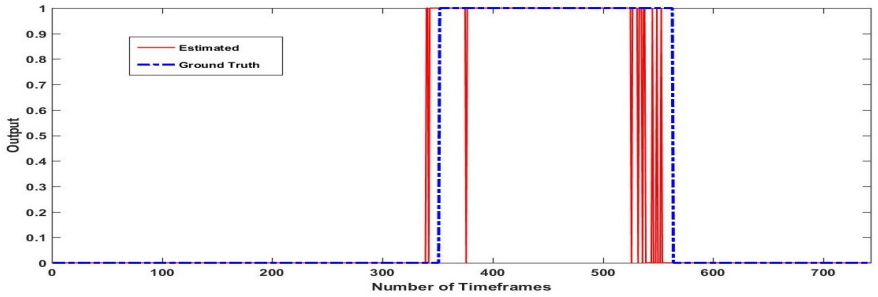
TABLE 4.4: Performance comparison for case 2

(a) Metrics

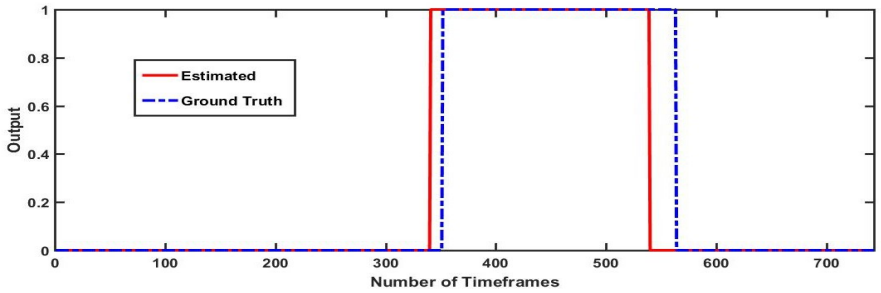
Metric (%)	**Before Smoothing**	**After Smoothing**
Accuracy	95.2	95.3
Sensitivity	87.3	88.7
Specificity	98.3	98
Precision	95.4	94.5

(b) Confusion matrices

Before Smoothing		**After Smoothing**	
TP: 185	**FP:** 9	**TP:** 188	**FP:** 11
FN: 27	**TN:** 522	**FN:** 24	**TN:** 520



(a)



(b) After smoothing

FIGURE 4.6: Estimated output v/s ground truth (a) before smoothing, and (b) after smoothing

4.2.3 Effect of target height

In this section, we compared the results for a second test before and after filtering target height for the optimal parameter values as discussed in case 1. This test set consisted of 1038 samples. Here, filtering target height refers to considering detections with height between 6 to 8 meters. The predicted output is a smoothed one. It can be seen in Table 4.5 (a) and (b) that accuracy, specificity and precision are reduced slightly with filtered height with an increase in the number of false positives. This can be improved by applying regularization. We concluded that filtering the height did not contribute much towards improving the performance.

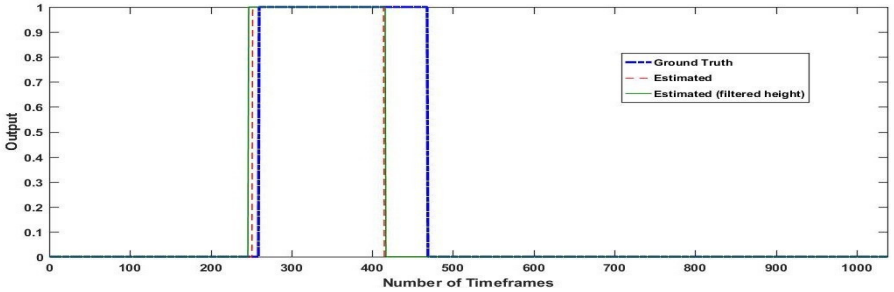


FIGURE 4.7: Estimated output v/s ground truth compared with and without filtering height

TABLE 4.5: Performance comparison for case 3

(a) Metrics

Metric (%)	Without Filtering Height	After Filtering Height
Accuracy	94	93.7
Sensitivity	74.2	75
Specificity	99	98.4
Precision	95	92.3

(b) Confusion matrices

Without Filtering Height		After Filtering Height	
TP: 155	**FP:** 8	**TP:** 157	**FP:** 13
FN: 54	**TN:** 821	**FN:** 52	**TN:** 816

Chapter 5

Conclusion

In this work, we proposed a machine learning based method for detecting tunnels using processed radar data. We determined that increased multiple-target detection counts and target heights were the appropriate patterns for identifying the presence of tunnels. We implemented logistic regression to model and train the classifier. We compared the classification accuracy, precision, specificity and sensitivity for test sets with different combinations of hyper-parameter values and feature matrix sizes. We showed that the unregularized logistic regression with mini-batch gradient descent using a feature matrix size of 5×5 provided the optimum results. Also, smoothing the predicted output helped in reducing the fluctuations in the output.

5.1 Limitations

There are certain limitations in this work. Firstly, mapping a one dimensional parameter into a two dimensional feature matrix results into values of many cells being zero leading to sparse matrix. This limits the classifier's performance. Second, the classification relies on only two parameters - multiple target detections and target height. We need to identify additional parameters to improve the performance. The target height parameter is estimated from the sensor's range and elevation. Since, the parameter is derived from two other measurements, its error bounds are high. The availability of a sensor that directly provides height information of a target would result in lower errors. The classifier was unable to detect those tunnels that did not strongly reflect the radar signals possibly due to their geometries or construction material. Lastly, the algorithm was unable to detect multiple tunnels that were separated by short distances.

Appendix A

Moving Average Filter

A moving average filter smooths data by replacing each data point with the average of the neighboring data points defined within the span [13]. It is ensured that span must be odd and the data point to be smoothed must be at the center of the span. This process is equivalent to low-pass filtering and the filter response is given as

$$y_s(n) = \frac{1}{2N+1} \sum_{k=-N}^{N} y(n+k)$$

where, $y_s(n)$ is the smoothed output value for n^{th} data point, N = number of neighboring data points on either side of the n^{th} data point and 2N+1 is the span.

The span is adjusted for data points that cannot accommodate the specified number of neighbors on either side. Since span cannot be defined for end points, they are not smoothed.

Bibliography

[1] Ann-Katrin Batzer, Christian Scharfenberger, Michelle Karg, Stefan Lueke, and Jürgen Adamy. Generic hypothesis generation for small and distant objects. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 2171–2178. IEEE, 2016.

[2] P Vasuki and S Veluchamy. Pedestrian detection for driver assistance systems. In *Recent Trends in Information Technology (ICRTIT), 2016 International Conference on*, pages 1–4. IEEE, 2016.

[3] Chia-Chen Li, Pei-Chen Wu, and Chang Hong Lin. Pedestrian detection using heuristic statistics and machine learning. In *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*, pages 1–5. IEEE, 2013.

[4] Wen Xiao, Bruno Vallet, Konrad Schindler, and Nicolas Paparoditis. Street-side vehicle detection, classification and change detection using mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:166–178, 2016.

[5] Ürün Dogan, Johann Edelbrunner, and Ioannis Iossifidis. Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1837–1843. IEEE, 2011.

[6] Il-Hwan Kim, Jae-Hwan Bong, Jooyoung Park, and Shinsuk Park. Prediction of drivers intention of lane change by augmenting sensor information using machine learning techniques. *Sensors*, 17(6):1350, 2017.

[7] `https://www.continental-automotive.com/en-gl/Passenger-Cars/Chassis-Safety/Advanced-Driver-Assistance-Systems/Radars/Long-Range-Radar/Advanced-Radar-Sensor-ARS441`.

[8] Klaudius Werber, Matthias Rapp, Jens Klappstein, Markus Hahn, Jürgen Dickmann, Klaus Dietmayer, and Christian Waldschmidt. Automotive radar gridmap

representations. In *Microwaves for Intelligent Mobility (ICMIM), 2015 IEEE MTT-S International Conference on*, pages 1–4. IEEE, 2015.

[9] Christopher M Bishop. Pattern recognition and machine learning (information science and statistics) springer-verlag new york. *Inc. Secaucus, NJ, USA*, 2006.

[10] `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`, .

[11] `https://crsouza.com/2009/12/28/discriminatory-power-analysis-by-receiver-operatin` .

[12] `https://in.mathworks.com/help/matlab/ref/smoothdata.html`.

[13] `https://in.mathworks.com/help/curvefit/smoothing-data.html`.