Lane Changing Prediction Modeling on Highway Ramps: Approaches and Analysis

Yuan Zhang **Sheng Li** **Mengxuan Zhang**

Abstract

The lane-change maneuver prediction system for human drivers is valuable for advanced driver assistance systems (ADAS) in terms of avoiding unnecessary maneuver efforts or unsafe merging, as well as encouraging lane-change behaviors that could increase travel efficiency. Learning the decision-making process of an intended lane changing is essential to model semi/full autonomous vehicles control systems. However, it will be a highly complicated decision-making model if analyzed explicitly, since the observation space is of high dimension, and the transition states are connected sequentially. In this paper, we proposed general machine learning approaches to generalize the lane-change model, and classified the most indicating features that may lead to a lane-change maneuver in the near future. The detailed vehicle trajectory data on highway ramps was found from the Next Generation Simulation (NGSIM). The predictor was modeled on three different approaches: logistic regression, support vector machine (SVM) and artificial neural network (ANN). The predictor was trained on high dimensional features for each lane changing/keeping event, including six time steps of characteristic information from both the ego vehicle and three surrounding vehicles. The results show that the lane changing can be predicted with high accuracy in all three algorithms, among which ANN reaches the best performance.

.

1. Introduction

The lane changing prediction is critical in the field of human driving modeling for autonomous vehicles and traffic simulators. As the result of a study, improper lane changing and lane merging maneuvers may lead to serious problems, which accounts for 5% of all crashes and 7% of all crash fatalities [1]. Gipps in [2] concerns the human reasoning process of making a lane changing decision: the possibility, necessity, and desirability of performing lane changes. Ahmed [3] proposed a discrete dynamic environment indicator model, which defines the lane changing as a three-state process: decision to consider a lane change, choice of a target lane, and acceptance of a gap in the target lane. In the aspect of behavioral dynamics, Lee [4] models the lane changing environment and defines the relative motion estimator of the surrounded vehicles, which requires high fidelity trajectory data and continuous dynamic inputs.

On the other hand, we wish to generalize the model by extracting existing data instead of modeling the explicit reasoning process or vehicle dynamics model. Hou [5] investigates to ensemble learning method, random forest and AdaBoost, to develop the lane changing assistance system. AdaBoost iterates starting from a weak classifier and compute weighting distribution to strengthen the classifier, whereas random forest performs well for high-variance and low-bias procedures. Our approaches are similar from the method proposed in [6], where Dou utilized Artificial Neural Network and Support Vector Machine to classify the given features and compared their performance. However, they only took into consideration of 6-dimension features, and assumed single time step ahead of the lane changing behavior, which only accounts for instant lane change decisions. In our work, we increase dimensionality of the features, which takes account of three surrounded vehicles, in terms of distance, relative velocity, acceleration and vehicle type; as well as all information of multiple time steps before the actual lane changing maneuver.

This paper begins with an introducing the dataset, defining the features, and extracting the samples in Section 2. The inputs fed to the training algorithms are samples of lane changing/keeping events, which are constructed with observations of the traffic environment. In Section 3, we introduce the training approaches (logistic regression, SVM and ANN) and hyperparameter optimization methods. The outputs are binary predictions of whether a car will make a lane changing or not. The results for each approach and analysis are organized in Section 4. Section 5 concludes our paper, and provides directions for future work.

2. Data Extraction and Feature Selection

2.1 Introduction of Dataset

The training and testing data were found on Next Generation Simulation (NGSIM) website [7]. The NGSIM dataset contains the detailed vehicle trajectory information in ramp sections of US-101 and I-80 highways in California. Recorded in July 2005, a total of 45 minutes of data are available in the full dataset, segmented into three 15 minutes periods. The data includes a study area of 600 meters, where 8 synchronized digital cameras located on a 36-story building adjacent to the freeway, with sampling

rate of 0.1 second. The US-101 dataset contains 3,849,725 entries and the I-80 dataset has 3,145,725 entries. The dataset gives a unique ID to existed each vehicle, and keeps the information of longitudinal positions, velocities and accelerations, and lateral positions for each vehicle, as well as the vehicle type, lane ID, and time/space headways. The detailed description of the dataset can be found in [7]. We only brought some key identities that may relate to our study here. We selected and extracted relevant information from the dataset, and constructed the feature set used for training and testing.

2.2 Introduction of Features and the Model

We selected the features that may most possibly affect the decision on resulting a lane changing event based on our recognition. We defined the ego vehicle to be the vehicle of interest, which is the predicted vehicle that we proposed to train and test, shown as the red car in Fig. 1. At each time step, we chose to observe a total of 14 characteristic features, including 2 features of the ego vehicle: its longitudinal velocity and longitudinal acceleration. The remaining 12 features characterize the surrounding environment of the ego vehicle, where the environment is composed of three surrounding vehicles (yellow vehicles in Fig. 1) that are located at: (*1*) the direct front in the current lane, which we named it the front center vehicle (fc); (*2*) the preceding vehicle in the target lane (ft); (*3*) the following vehicle at the rear of the target lane (rt). For each of the three surrounding vehicles, we chose to observe 4 features, which are the relative distances to the ego vehicle in both longitudinal (x) and lateral directions (y), its relative longitudinal velocities (v) and longitudinal accelerations (a) with respect to the ego vehicle. Note that the above description of 14 features are variables in a single time step.

Further, we assumed the movement of vehicles to be of Markov process, where the state at the next time step is independent from the past states. Therefore, we could model the feature set of a lane changing event as the concatenation of the features of multiple time steps before the lane changing event happens. In the current model, we used the concatenation of features from 6 time steps ahead of the lane changing happens. So that a feature set of a lane changing event is a feature vector with dimension of $14\times6 = 84$.

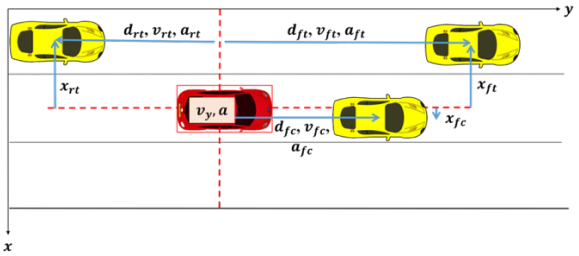

Figure 1: the demonstration of the model and the feature classes at one time step.

This model assumes that in a lane changing event, as demonstrated in Fig. 1, there is always a front vehicle in the same lane of the ego vehicle. Moreover, there always exists a front vehicle and a rear vehicle in the target lane in case of a lane changing. This restriction is strict, but this scenario accounts for the most dangerous case of all types of lane changing event, which is our focus to study.

2.3 Data Filtering and Extracting

The raw NGSIM sampled the data at a time step of 0.1 seconds. We resampled the dataset with a time step of 0.5 seconds, so as to record more sequential information in the feature vectors with limited memory space. As a result, the dataset reduced its size by a factor of 5. Then we chose a concatenation with information from previous 6 time steps to construct a feature vector, i.e. 3.0 seconds sequential information of the ego vehicle is recorded.

To collect data that satisfies the constraints of the lane changing model that there are exactly three surrounding cars, we performed data filtering and extraction on the original NGSIM dataset. The resulting samples contain all the lane changing events that satisfy our assumptions. To train the machine learning algorithms, lane keeping events were also needed to provide negative training samples. A lane keeping event was found 10 seconds prior to a lane changing event for each ego vehicle. Note that the lane-keep events due to traffic congestion were filtered away.

The resulting dataset is therefore ready for training and testing, which has nearly equal numbers of lane changing events (645 samples) and lane keeping events (663 samples).

3. Approaches

3.1 Feature Selection

There are 84 features for each sample, but we suspected that some features are not important to the learning task, and we were also worried about the overfitting problem due to the considerable number of features. To select the features, we used filter feature selection methods. We computed simple score $S(i)$ that measures how informative each feature x_i is about the class labels y, and then ranked the features according to their scores in descending order. We chose $S(i)$ to be the mutual information (MI, defined by Kullback-Leibler (KL) divergence) between x_i and y.

$$MI(x_i, y) = KL(p(x_i, y) \,||\, p(x_i)\, p(y))$$

We chose the top n_f highest scores features in the entire feature set to form a new feature vector, where n_f is a hyperparameter to be optimized.

3.2 Logistic Regression

Binary logistic regression (LR) classifies data by estimating the probability that a class is present given the values of features. Binary logistic regression with regularized average empirical loss was used as the first approach. Newton's method was used as the solver. $y = 1$ represents a lane-change event and $y = -1$ represents a lane-keep event. Regularized empirical loss is given by

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + e^{-y^{(i)}\theta^T x^{(i)}}\right) + \frac{\lambda}{2}\|\theta\|^2,$$
$$\text{where } y^{(i)} \in \{-1, 1\}.$$

λ is a hyperparameter that controls the amount of regularization.

3.3 Support Vector Machine

Support Vector Machines (SVM) was used as the second approach. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. In addition, SVM can take advantage of kernels, which gives a way to apply SVMs efficiently in very high dimensional feature spaces. We used SVM with regularized hinge loss and Gaussian kernel, and SMO algorithm was used as the solver. Regularized hinge loss is given by

$$J(w, b) = \sum_{i=1}^{m} \left[1 - y^{(i)}\left(w^T x^{(i)} - b\right)\right]_+ + \frac{\lambda}{2}\|w\|^2,$$

Gaussian kernel is given by

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right).$$

There are two hyperparameters in our SVM model, λ, which controls the amount of regularization and σ, which is a free parameter.

3.4 Artificial Neural Network

Artificial Neural Network (ANN)is a biologically inspired network of artificial neurons configured to perform specific tasks. Feed forward neural network composed of two fully connected layers was used as the third approach. The weight and bias values were initialized by Nguyen-Widrow initialization method. A hyperbolic tangent sigmoid function was used as the activation function of the first layer (hidden layer), and log-sigmoid function was used as the activation function of the second layer (output layer). Gradient descent method with momentum weight and bias was used as the learning function. Cross-entropy loss with scaled conjugate gradient backpropagation was used to train the network. We set the number of neurons in second layer to be one and let the number of neurons in the first layer (n_{neuron}) to be a hyperparameter to be tuned. An example of neural network with one hidden layer is shown in Fig. 2.

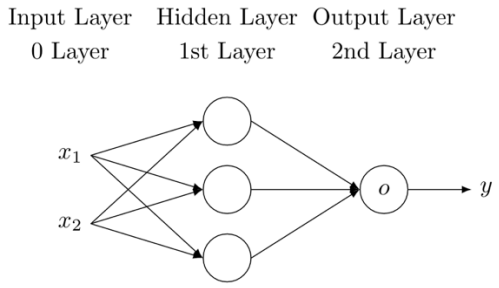

Figure 2: Schematic of a fully connected two layers feed forward neural network.

3.5 Hyperparameters Optimization

The values of hyperparameter have significant impact on the performance of a learning algorithm. For example, Fig. 3 shows the effect of n_f in the logistic regression for fixed λ. A small n_f causes high bias problem, while large n_f causes high variance problem. The Fig. 4 shows the effect of regularization in the logistic regression for fixed n_f. A small λ solves the high variance problem, whereas a large λ introduces the bias problem. By tuning the hyperparameters, an optimized model is achievable.

To find the optimal values of hyperparameters, we randomly split samples into three subsets: the training set (70% of the data), the hold-out cross validation set, which is also known as dev set (15% of the data), and the test set (15% of the data). For a given set of hyperparameters, we trained the model in the training set, and evaluated its performance according to a metric in the dev set. The set of hyperparameters that results the best performance in the dev set will be defined as the set of optimal hyperparameters. The optimized model is attained by retraining the model with the set of optimal hyperparameters in the set composed of the training set and the dev set. The test set is used to evaluate the performance of the optimized model.

The area under the receiver operating characteristic (ROC) curve (AUROC) was chosen as the metric to evaluate the performance of the model, since the AUROC measures the overall classification performance. A better classifier should have a larger AUROC value. A perfect classifier has the AUROC value equaling one.

As shown in Table 1, each learning algorithm has more than one hyperparameters, so the multivariate global optimization method needs to be applied to find a set of hyperparameters that maximizes the AUROC value in the dev set. Several optimization methods were applied, and genetic algorithm performed the best according to the metric of finding global maximum [8].

LR	SVM	ANN
n_f, λ	n_f, λ, σ	n_f, n_{neuron}

Table 1: Hyperparameters of each algorithm

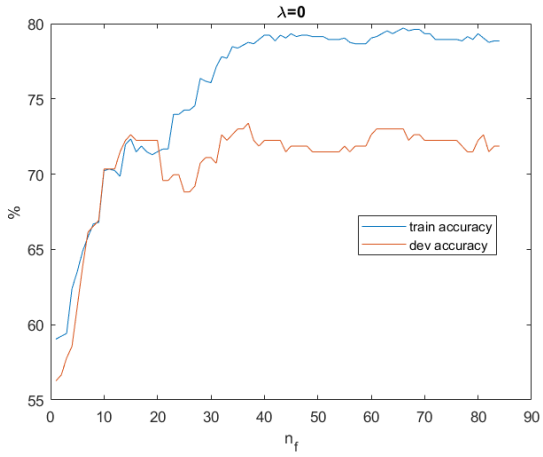

Figure 3: Effect of n_f in LR for fixed λ

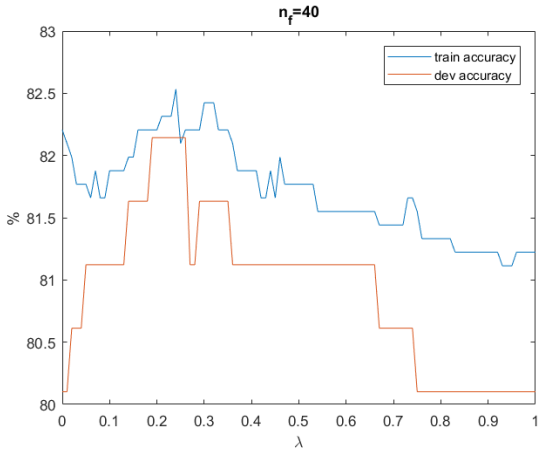

Figure 4: Effect of λ in the LR for fixed n_f

4. Results and Analysis

4.1 Feature Selection

The top 5 high-score feature categories are:

1. d_{ft}: longitudinal distance to the preceding vehicle in the target lane
2. x_{fc}: Lateral distance to the front center vehicle
3. v_{fc}: Relative longitudinal velocity to the front center vehicle
4. v_{rt}: Relative longitudinal velocity to the following vehicle in the target lane
5. v_y: Ego vehicle longitudinal velocity

4.2 Logistic Regression

Fig. 5 shows the test set results from the logistic regression. The test set accuracy is 82.1%, sensitivity is 83%, precision is 80.4% and F1-Socre is 0.817. The optimized n_f is 82, and the optimized λ is 0.0254. AUROC value is 0.9047.

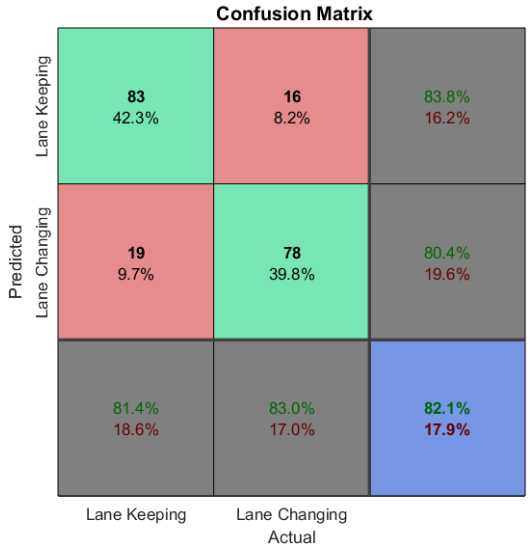

Figure 5: The test set results of logistic regression.

4.3 Support Vector Machine

Fig. 6 shows the test set results from the support vector machines. The test set accuracy is 85.7%, sensitivity is 80.9%, precision is 88.4%, and F1-Socre is 0.845.

The optimized n_f is 83, optimized λ is 1978 and the optimized σ is 30.3861. AUROC value is 0.9078.

4.4 Artificial Neural Network

Fig. 7 shows the test set results from the artificial neural network. The accuracy is 96.9%, sensitivity is 98.9%, precision is 94.9%, and F1-Score is 0.969.

The optimized n_f is 25, and the optimized n_{neuron} is 23. The AUROC value is 0.9961.

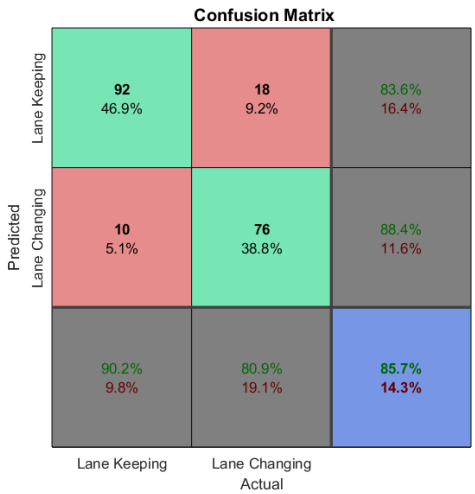

Figure 6: The test set results of support vector machine

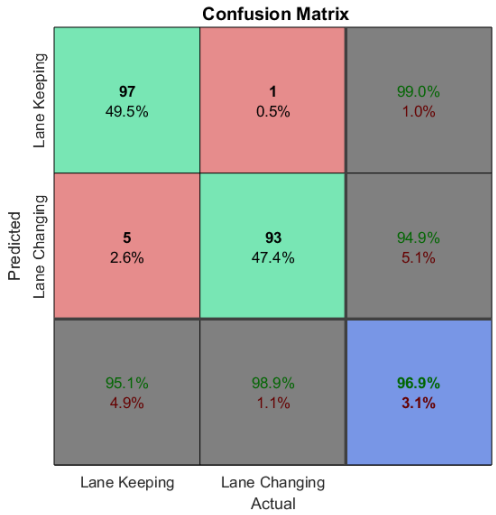

Figure 7: The test set results of Artificial Neural Network.

4.5 Analysis

Table 2 and Fig. 8 show the comparison of LR, SVM, and ANN. In the context of our problem formulation, the cost of false negative is much higher than the cost of false positive because the failure in predicting lane changing event may cause traffic accident. Thus, we use F1-score to measure the test set performance (threshold = 0.5), and AUROC to measure the overall classification performance (threshold from 0 to 1). For the test set performance measured by F1-score, ANN outperforms SVM, and SVM outperforms LR. For overall classification performance measured by AUROC, ANN has the best performance, and SVM and LR have similar overall classification performance.

	Accuracy	Precision	Sensitivity	F1-Score
LR	82.1%	80.4%	83.0%	0.817
SVM	85.1%	88.4%	80.9%	0.845
ANN	96.9%	94.9%	98.9%	0.969

Table 2: Comparison of LR, SVM, and ANN Performance

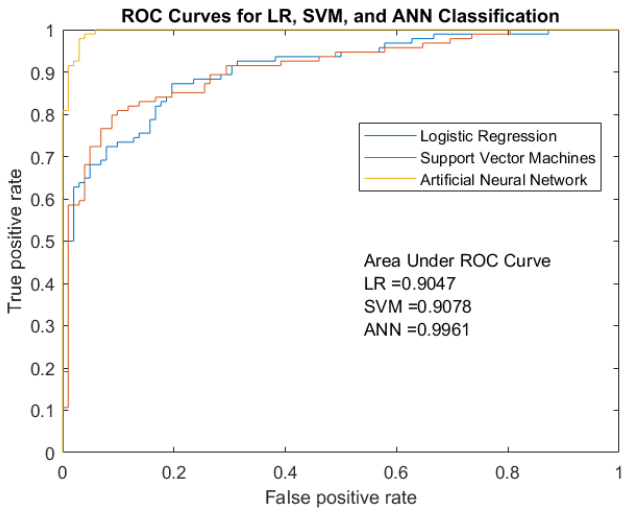

Figure 8: ROC Curves for LR, SVM, and ANN Classification.

4.6 Error Analysis

Though neural network has high predicting accuracy, there are still several misclassification examples exist. To analyze the prediction errors, we extracted the wrong prediction cases observed during running the classifier on the test set. After analyzing these cases, we found that they can be summarized into two categories:

1. Data set error due to sensor error

We observed that in some feature vectors which led to misclassification have discontinuity in the continuous variables (e.g. sudden dramatic change in distance). This can be explained by data set error due to sensing error. Our filter failed to filter these data.

2. Ego car hesitates over lane changing

Some of the wrong predictions show the lateral positions of the ego cars sway between in lateral directions. Our classifier failed to make correct lane-changing predictions for this type of confusing maneuver.

5. Conclusion and Future Work

In this paper, we proposed a lane changing predictor on highway ramps. We extracted lane changing and lane keeping samples from NGSIM, which records vehicle trajectories of the entire traffic on US-101 and I-80 correspondingly. We used logistic regression, SVM and neural network in separately to train our classifiers, and optimized their hyperparameters to obtain better models. The results show that a lane changing maneuver is indeed predictable, where the average accuracy of three algorithms reaches over 85%. The ANN outperforms the other two algorithms in all measurements.

This project may be extended in terms of generalizing the lane changing model, and extending the lane changing prediction to multiple outputs including lane changing directions, velocity, path planning, etc.

Contributions

Yuan Zhang: Implementing training algorithms, optimizing hyperparameters, testing and evaluating classifier performance

Sheng Li: extracting data, filtering data, building model, preliminary training algorithm testing

Mengxuan Zhang: Literature review, analyzing dataset, constructing observations.

References

[1] Chovan, J.D.. Tijerina, L., Alexander, G., & Hendricks, D.L. (1994). Examination of lane change crashes and potential IVHS countermeasures. National Highway Traffic Safety Administration, US Department Transportation, Technical Report.

[2] Gipps, P.G. (1986). A model for the structure of lane-changing decisions. Transp. Rea. Part B, Methodology., col. 20, no. 5, pp. 403-414.

[3] Ahmed, K.I. (1999.) Modeling driver's acceleration and lane changing behaviors. Ph.D. dissertation. Department of Civil and Environmental Engineering, MIT.

[4] Lee, Junyung et al. "Design Of A Strategy For Lane Change Assistance System." IFAC Proceedings Volumes, vol 46, no. 21, 2013, pp. 762-767. Elsevier BV, doi:10.3182/20130904-4-jp-2042.00134.

[5] Hou, Y., Edara, P., & Sun, C. (2015). Situation assessment and decision making for lane change assistance using ensemble learning methods. *Expert Systems with Applications*, *42*(8), 3875-3882.

[6] Dou, Y., Yan, F., & Feng, D. (2016, July). Lane changing prediction at highway lane drops using support vector machine and artificial neural network classifiers. In *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on* (pp. 901-906). IEEE.

[7] Next Generation Simulation Fact Sheet (2017, Nov). Next Generation Simulation Vehicle Trajectories. online: https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories

[8] Davis, L. (1991). Handbook of genetic algorithms.