Probabilistic Driving Models and Lane Change Prediction

Tim Wheeler

CS229 Fall 2014

Abstract

Probabilistic traffic models, providing a statistical representation of the future behavior of traffic participants, are crucial for risk estimation in automotive collision avoidance systems. Current research has focused on large-scale behavior, primarily in the form of lane change prediction. These models are limited in their use for high-fidelity driving propagation. This paper investigates a methodology for dynamic model construction based on a Bayesian statistical framework successfully employed in aviation collision avoidance systems. Machine learning techniques are also used to develop a lane change predictor.

1 Introduction

As the automotive industry moves towards autonomous driving, it becomes increasingly necessary to develop advanced collision avoidance systems, crash prediction systems, and the tools for their rigorous analysis. The certification of any automated driving system will require a combination of driving tests and detailed simulation studies to ensure system effectiveness and safety. Driving tests are inherently expensive and time-intensive; simulation can be used to quickly and inexpensively test over the space of potential trajectories.

Recent developments in collision avoidance in civil aviation have allowed for the creation of rich encounter models based on a Bayesian statistical framework from which optimal collision avoidance strategies have been derived[1]. These encounter models used dynamic Bayesian networks for the correlated propagation of aircraft in a close encounter.

The purpose of this project is to work towards the development of such statistical models for highway driving and to apply machine learning techniques to develop a lane change predictor.

All work for this project was conducted with the Julia programming language[2].

2 Probabilistic Traffic Models

The driving prediction problem can be formalized as a joint probability distribution $P(s_{t+1} \mid s_t)$ over the future scene given the current scene configuation, where a scene s incorporates current and past information necessary to leverage the Markov assumption. This work focuses on obtaining a model of this probabilist distribution which best fits the existing training data and performs well in simulation. Prior work in collision avoidance for aviation leveraged dynamic Bayesian networks to model this state transition probability [3].

A dynamic Bayesian network (DBN) is a Bayesian network split into time slices, where indicators from time t are used to condition a distribution over target values in $t+1$. Using a DBN to directly model $P(s_{t+1} \mid s_t)$ is difficult due to the high variability in traffic structure and scene composition. One way to develop a model applicable accross varying traffic structures and participant counts is to model the distribution over individual vehicles' actions, $P(a_t^i \mid s_t)$ where the individual actions can be used to deterministically propagate the scene forward.

Vehicle pose and kinematics are modelled assuming a rigid body in the euclidean plane. Control is exerted through the pedals and the steering wheel. Thus, the target values for the DBN were chosen to be the constant acceleration and turn-rate over the next time step, a^{fut} and $\dot{\phi}^{\text{fut}}$. Deterministic propagation of a particular vehicle's state given these values is conducted using Euler integration.

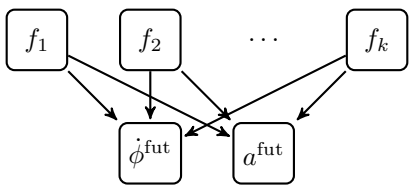

Figure 1: Probabilist dynamics model in the form of a Dynamic Bayesian Network

Modelling the distribution requires solving several issues simultaneously. The set of relevant features, their respective discretizations, the model structure, and the conditional probability tables must all be learned from a limited set of driving data.

3 Data Processing

This work uses data generated from Interstate 280 (I-280) in the San Francisco Bay Area of Northern California. I-280 presents a clean driving environment, lacking complicated merges and splits between Palo Alto and South San Francisco, and in particular always has four lanes in each direction for the sections used.

Group	Feature	Units	Description
GLOBAL	t	s	timestamp, Unix Epoch
EGO	p_x^G	m	northing in the global coordinate frame
	p_y^G	m	easting in the global coordinate frame
	v_x^B	$\mathrm{m\,s^{-1}}$	longitudinal velocity in the body frame
	v_y^B	$\mathrm{m\,s^{-1}}$	lateral velocity in the body frame
	ϕ^G	rad	vehicle heading in the global frame
OTHER	id	-	identification number, unique across frames
	p_x^B	m	longitudinal relative position from the ego car
	p_y^B	m	lateral relative position from the ego car
	$v_{x,oth}^B$	$\mathrm{m\,s^{-1}}$	longitudinal relative velocity from the ego car
	$v_{y,oth}^B$	$\mathrm{m\,s^{-1}}$	lateral relative velocity from the ego car

Table 1: A summary of raw data collected from drives

Raw features were obtained from drive log files and are summarized in table 1. Data had already been resampled to 20 Hz. High-accuracy lane curves were used to project the given data to a road-relative Frenet frame.

A total of thirteen hours of driving data spread over twenty one drive files were available for processing, split roughly evenly between north- and south-bound. Data outliers were removed using a gaussian threshold test and then smoothed using a gaussian kernel. Resulting poses were projected to the lane-relative Frenet frame with the use of lane centerlines. This frame has an x-axis along the lane in the direction of travel and a y-axis perpendicular to the lane, positive towards the left. The resulting lane-relative positions and velocities (p_x^F, p_y^F, ψ^F, v_x^F, and v_y^F) were smoothed again to reduce noise from the projection process.

A set of 162 indicator features were extracted for the ego vehicle. Features included inherent vehicle properties, relative features between vehicles, roadway relative features, and aggregated features over a vehicle's history. Features were reflective of those used in the literature [4, 5]. For a complete list of features see table 8 in the appendix.

Due to limitations with the current sensing setup, data on other vehicles is not reliable figure 3. Vehicle positions are estimated by fitting to point clouds, resulting in problems with larger vehicles, occlusion, and doubling up when a new section of the car becomes visible. This, coupled with the natural limitation of not being able to see the entire environment, makes it difficult to create a predictor based on other vehicles.

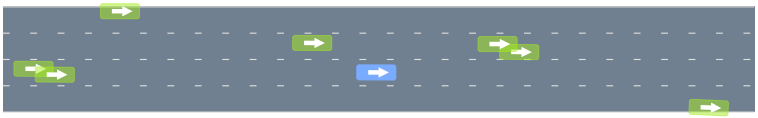

Figure 2: A projection to the Frenet frame showing vehicle overlap. Blue indicates the ego vehicle

Instead, learning methods were applied to the ego vehicle instead. A more detailed analysis of other car motion will be withheld until higher quality data can be obtained.[1]

The problem was chosen to be modelled using discretized probability distributions, simplifying the inclusion

[1]Information from other vehicles is still leveraged, such as when computing features like the distance to the car in front

of hybrid variables and allowing for standard structure learning packages to be used. For the purposes of this project discretization was conducted with manually chosen bin edges, typically five bins per variable.

4 Feature Selection for Probabilistic Traffic Models

One primary objective for this project is to identify a set of features to be used in a probabilistic traffic model. Formally, one must identify the set of features which maximize the likelihood of the graph given the data, $P(G \mid D)$. This is equivalent to maximizing the log-likelihood, also known as the Bayesian score:

$$\ln P(G \mid D) = \ln P(G) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left[\ln \left(\frac{\Gamma(\alpha_{ij0})}{\Gamma(\alpha_{ij0} + m_{ij0})} \right) + \sum_{k=1}^{r_i} \ln \left(\frac{\Gamma(\alpha_{ijk} + m_{ijk})}{\Gamma(\alpha_{ijk}} \right) \right]$$

where there are n random variables $X_{i:n}$, r_i is the number of bins for X_i, q_i is the number of parental instantiations of X_i, m_{ijk} is the number of times $X_i = k$ given the jth parental instantiation in the dataset $\mathcal{D}$. We use α_{ijk} for a Dirichlet prior (same idea as Laplace Smoothing) and $\alpha_{ij0} = \sum_{k=1}^{r_i} \alpha_{ijk}$, $m_{ij0} = \sum_{k=1}^{r_i} m_{ijk}$.

For general graphs and input data, learning the structure is NP-hard. We make an independence assumption between the two target variables to split the problem into two smaller structure learning problems:

Solving each reduced problem is still difficult. There are $2^{n_{\text{indicators}}} = 2^{162}$ possible graphs, one binary choice on whether to include a particular edge for each indicator. Note that the actual graph will likely only have a small number of parents, due to a built-in limitation on needing enough data to properfy specify such a large conditional probability distribution. Even if we restrict ourselves to eight parents there are $\sum_{i=0}^{8} \binom{162}{i} \sim 10$ trillion options.

Three heuristic feature search methods were used: feature ranking, forward search, and graph search. Each seeks to maximize the log-bayesian score on a reduced problem.

Feature ranking is the simplest of the three. Here the Bayesian score is computed for each indicator-target pair, and the highest ranked indicators are successively added until the Bayesian score ceases to increase.

Algorithm 1 Feature Ranking

$R \Leftarrow$ features sorted by indicator-target Bayesian component score
$\mathcal{F} \Leftarrow \emptyset$
while score$(\mathcal{F} \cup \{R_{|\mathcal{F}|+1}\}) >$ score$(\mathcal{F})$ **do**
  $\mathcal{F} \Leftarrow \mathcal{F} \cup \{R_{|\mathcal{F}|+1}\}$
end while

Forward search starts with an empty feature graph and continuously adds the next feature which results in the greatest increase in the Bayesian score. The algorithm terminates when there is no feature which can further increase the score. Forward search can get stuck in local optima but in practice outperforms feature ranking.

Algorithm 2 Forward Search

$\mathcal{I} \Leftarrow$ set of all indicators
$\mathcal{F} \Leftarrow \emptyset$
while $\arg\max_{f \in \mathcal{I}}$ score$(\mathcal{F} \cup \{f\}) >$ score$(\mathcal{F})$ **do**
  $\mathcal{F} \Leftarrow \mathcal{F} \cup \{f\}$
  $\mathcal{I} \Leftarrow \mathcal{I} \setminus \{f\}$
end while

Graph search is similar to forward search but allows for the removal of a feature. It also terminates when no action will result in a higher score. Graph search has greater freedom than forward search in traversing the space but can also get stuck in local maxima. It tends to perform the best.

Algorithm 3 Graph Search

$\mathcal{I} \Leftarrow$ set of all indicators
$\mathcal{F} \Leftarrow \emptyset$
while TRUE **do**
  $\text{score}_{\text{add}} = \max_{f \in \mathcal{I}} \text{score}(\mathcal{F} \cup \{f\})$
  $\text{score}_{\text{delete}} = \max_{f \in \mathcal{F}} \text{score}(\mathcal{F} \setminus \{f\})$
  if $\text{score}_{\text{add}} > \text{score}_{\text{delete}}$ AND $\text{score}_{\text{add}} > score(\mathcal{F})$ **then**
    $f = \arg\max_{f \in \mathcal{I}} \text{score}(\mathcal{F} \cup \{f\})$
    $\mathcal{F} \Leftarrow \mathcal{F} \cup \{f\}$
    $\mathcal{I} \Leftarrow \mathcal{I} \setminus \{f\}$
  else if $\text{score}_{\text{add}} > \text{score}_{\text{delete}}$ AND $\text{score}_{\text{add}} > \text{score}_{\text{delete}} > score(\mathcal{F})$ **then**
    $f = \arg\max_{f \in \mathcal{F}} \text{score}(\mathcal{F} \setminus \{f\})$
    $\mathcal{F} \Leftarrow \mathcal{F} \setminus \{f\}$
    $\mathcal{I} \Leftarrow \mathcal{I} \cup \{f\}$
  else
    BREAK
  end if
end while

These algorithms were run on two separate feature sets, the full indicator set including all 162 features and a reduced indicator set lacking the template features, totalling 60. A runtime comparison is given in table 8.1. Four target values were used, $\dot{\phi}^{\text{fut}}$ and a^{fut} each for a quarter second and half second horizon. The results for $\dot{\phi}^{\text{fut}}_{250ms}$ are given below. See the rest in the appendix.

60 Features	Score	Indicators
Feature Ranking	-54532	$\dot{\psi}$, v_y^F, ψ, a_y^F
Forward Search	-54506	$\dot{\psi}$, v_y^F, d_{cl}
Graph Search	-54506	$\dot{\psi}$, v_y^F, d_{cl}

Table 2: Feature selection for $\dot{\phi}^{\text{fut}}_{250ms}$ with 60 features

162 Features	Score	Indicators
Feature Ranking	-52141	$\dot{\psi}$, a_y^F, $\hat{\dot{\psi}}_{100ms}$, $\hat{\dot{\psi}}_{150ms}$, $\hat{a}^F_{y,100ms}$, $\bar{\dot{\psi}}_{100ms}$
Forward Search	-50433	$\dot{\psi}$, v_y^F, $\hat{\dot{\psi}}_{150ms}$, $\hat{a}^F_{y,750ms}$
Graph Search	-49957	$\dot{\psi}$, v_y^F, $\hat{\dot{\psi}}_{100ms}$, $\bar{\dot{\psi}}_{750ms}$

Table 3: Feature selection for $\dot{\phi}^{\text{fut}}_{250ms}$ with 162 features

It was necessary to decimate the training data by a factor of two in order to get decent results. Using the full dataset resulted in over-fitting and massive computation times due to the large number of features selected. Recall that the dataset is not iid, but sampled at 20Hz from continuous trajectories, so tossing every other sample does not result in much loss of information and greatly reduces data redundancy.

These results are promising. The smaller feature sets contain values which can be extracted from a small state space (they do not rely on a long history) and are thus candidates for online motion planning. The larger feature sets can be used for higher-fidelity models in the validation of safety systems.

5 Maximum Likelihood Structure Learning

The performance of three general structure learning algorithms were compared on a variety of indicator feature sets. Two of these algorithms used the Smile.jl package[2], the third, a greedy hill-climbing method using the K2 prior, was implemented based on a method directly in Julia[3].

Algorithm	K2 Score	BDeu Score	$deg^-(\dot{\psi}^{\text{fut}}_{250ms})$	$deg^-(a^{\text{fut}}_{250ms})$	Run Time
Bayesian Search	−867 523	−868 873	3	3	6.80s
Greedy Thick Thinning	−775 182	−779 545	4	4	1.29s
K2 Hill Climbing	−775 149	−779 545	4	2	32.9s

Table 4: Algorithm comparison for full structure search with eight indicators

Algorithm	K2 Score	BDeu Score	$deg^-(\dot{\psi}^{\text{fut}}_{250ms})$	$deg^-(a^{\text{fut}}_{250ms})$	Run Time
Bayesian Search	−1 138 535	−1 138 677	3	3	4.09s
Greedy Thick Thinning	−1 138 506	−1 138 993	4	4	0.43s
K2 Hill Climbing	−1 138 483	−1 138 993	4	2	30.9s

Table 5: Algorithm comparison for constrained structure search with eight indicators

Tables 4 and 5 show results using the 250ms targets and eight indicators: ψ, $\dot{\psi}$, v_y^F, a_x^F, a_y^F, $|v|$, d_{cl}, and $ttcr_{mr}$. The first is a full structure search in which tiering was enforced: edges from target variables to indicator variables were forbidden. The second table has the additional constraint of preventing edges between indicator variables as they are going to be observed in the final dynamic model.

We observe that the K2 Hill Climbing algorithm consistently results in the highest K2 score, which it is designed to maximize. The Bayesian Search method sometimes outperforms it in terms of the BDeu score, which it is designed to maximize. Greedy Thick Thinning is the fastest algorithm and produces the lowest-scoring graphs.

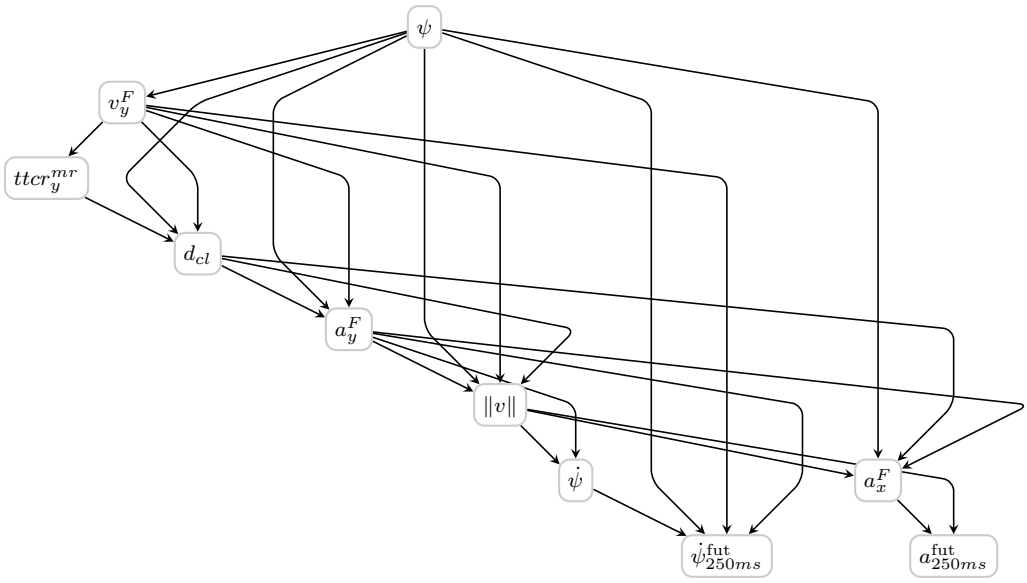

Figure 3: Resulting graph structure from K2 Hill Climbing under full search

[2]Written by the author to provide access to the SMILE C++ library[6]

[3]Method adapted from code written by Edward Schmerling

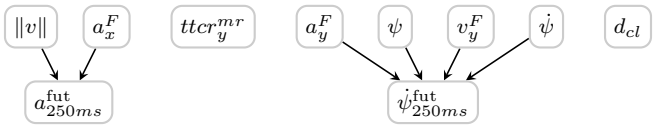

Figure 4: Resulting graph structure from K2 Hill Climbing under constrained search

The corresponding graph structures for K2 Hill Climbing are given in figures 3 and 4. Note that the parents of the target values remain unchanged. The advantage of the second approach is the drastically reduced search space. The resulting model is sufficient for modelling the posterior distribution when indicators are observed but cannot be used for reverse inference.

Model validation and tuning was the focus of work in an AA228 project and is subject to further research.

6 Predicting Lane Changes

Given a traffic scene at time t, predict whether a target vehicle will be in the same lane, in a lane to the left of its current lane, or in a lane to the right of its current lane at a time $t + H$ for some horizon H on the order of several seconds.

Most methods in the literature focus on lane change detection for the ego vehicle due to the same limitations of poor sensor measurements and incomplete scene information. The ego vehicle has very good knowledge of its own dynamics and fairly good knowledge of its immediate surroundings. For other vehicles this is not the case; for instance the ego vehicle may not know whether the car in front of itself has another car in front of it which affects its behavior.

It was common in previous work to include features such as driver eye tracking to predict lane changes. Note that there was no human driver in this case, and other features such as blinker status is not available. Only the original position and velocitiy estimates, coupled with the lane curves, were used to derive the feature set.

Let us consider applying a Naive Bayes classifier. The Naive Bayes classifier is well defined for discrete and continuous features, but it is not clear how to handle more complicated hybrid features. Consider $d_{x,fo}^{rel}$, the distance to the car in front. This feature is typically continuous, but when there is no car in front it takes on ∞. Similarly, features like $v_{x,fo}^{rel}$, the velocity of the car in front, is typically continuous but is simply missing if no vehicle is seen.

6.1 Baseline Algorithm Comparison

We can establish a lower bound on performance using only well-behaved indicators. Table 6.1 was computed using Sci-Kit learn from Julia via PyCall. Decimation by a factor of 10 was employed to avoid the problem of non-iid samples. There were a total of 272 samples with a lane change in a two-second horizon, out of a total of 30 441 samples. Results are averages over 10-fold cross validation.

Classifier	AUC	Precision	Accuracy	Parameters
RBF SVM	0.8129	0.9613	0.9961	default
Nearest Neighbors	0.8263	0.9181	0.9961	Nearest 2
AdaBoost	0.7809	0.8789	0.9934	default
Random Forest	0.5987	0.8373	0.9928	10 estimators, max depth = 5
Decision Tree	0.8182	0.8134	0.9954	max depth = 5
Linear SVM	0.5197	0.3833	0.9911	C = 0.025
Naive Bayes	0.9296	0.0968	0.9193	gaussian

Table 6: Rough Sci-Kit Learn algorithm comparison on well-behaved indicators

We see that Naive Bayes paradoxially has the highest AUC metric score but by far the lowest precision. It appears to be primarily predicting lane holds, which as we can see does fairly well in terms of AUC but poorly in terms of precision.

The highest precision was obtained by an SVM with an RBF (gaussian) kernel. This prompted further efforts to find a high-qualitfy classifier using a larger feature set. All 162 indicator features were used to train a

better RBF SVM classifier. Features with undefined behavior were arbitrarily set to their maximum values when undefined values were present.

6.2 Tuned RBF SVM Classifier

Recall that the radius basis kernel function is given by:

$$K_{\text{rbf}}(x, x') = \exp\left(-\gamma \|x - x'\|^2\right)$$

Given training vectors $x_{i:n} \in \mathcal{R}^p$ in two classes, and a truth vector $y \in \mathcal{R}^n$ such that $y_i \in \{-1, 1\}$, SVM for classification (SVC) solves the primal problem[7]:

$$\begin{aligned} \min_{w,b,\zeta} \ & \tfrac{1}{2} w^T w + C \textstyle\sum_{i=1}^{n} \zeta_i \\ \text{subject to } & y_i(w^T \phi(x) + b) \geq 1 - \zeta_i \\ & \zeta_i \geq 0 \quad \forall\, i = 1, \ldots, n \end{aligned}$$

The resulting dual formulation is:

$$\begin{aligned} \min_{\alpha} \ & \tfrac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{subject to } & y^T \alpha = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall\, i = 1, \ldots, n \end{aligned}$$

where e is the vector of all ones, $C > 0$ is the upper bound, and Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv K(x, x')$.

The classification decision function is:

$$\text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

where ρ is an intercept term also obtained during training.

The advanced RBF SVM used $\gamma = 0.001$. We find an increase in AUC outperforming all previous models. We do see a slight drop in accuracy but an increase in precision. Values extracted by averaging across 10 cross validation folds.

RBF SVM	Feature Count	AUC	Precision	Accuracy	Parameters
Original	60	0.8129	0.9613	0.9961	default
Improved	162	0.9718	0.9702	0.9419	$\gamma = 0.001$

Table 7: Comparison of tuned RBF SVM with full indicator set to baseline

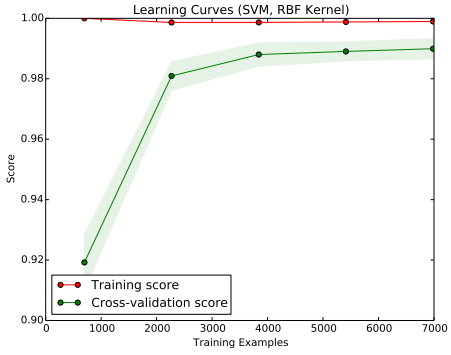

Figure 5: lane change AUC performance on a Naive Bayes classifier versus horizon.

An inspecton of the learning curve shows that the algorithm performance is converging on the training error, which remains close to one. This is very good and suggests that a larger number of iterations on a larger feature set would produce better results.

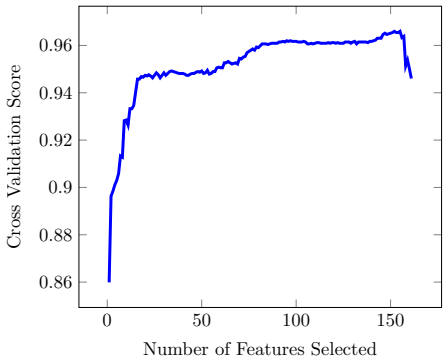

Figure 6: Automated feature reduction for RBF SVM

Running automated feature reduction shows that having access to most of the features results in best results, but fairly good results can still be obtained with as few as twenty. This makes sense; most of the vehicle's motion is captured in a few variables, but the full description is often better.

6.3 Predictive Power vs. Horizon

We can also get a sense for predictive power for certain horizons, see figure 6.3. Here we see strong results for a horizon of up to two seconds, after which the ability to predict begins to drop off.

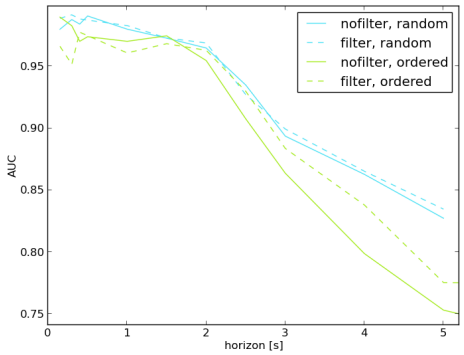

Figure 7: lane change AUC performance on a Naive Bayes classifier versus horizon.

7 Conclusion

This paper introduces a method for developing probabilistic driving models for use in traffic modelling and automotive safety systems. Dynamic Bayesian networks were used to represent the distribution over acceleration and turn-rate for the next time step. Graph structure and parameters were learned from real-world driving data using existing Bayesian methods and leveraging existing toolkits. In particular, the Greedy Hill-Climbing algorithm with a K2 prior consistently provided the best results in terms of maximizing the Bayesian Score.

A lane change predictor using support vector machine classification was trained using a radius basis kernel function. The classifier possessed an AUC metric of 0.9718 with the possibility of improvement given additional driving data. An analysis showed the decline in feature predictive performance with increased horizons above two seconds.

References

[1] M. Kochenderfer, L. Espindle, J. Kuchar, and J. D. Griffith, "Correlated encounter model for cooperative aircraft in the national airspace system version 1.0," *Project Report ATC-344, Lincoln Laboratory*, 2008.

[2] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv preprint arXiv:1209.5145*, 2012.

[3] M. J. Kochenderfer, M. W. M. Edwards, L. P. Espindle, J. K. Kuchar, and J. D. Griffith, "Airspace encounter models for estimating collision risk," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 487–499, 2010.

[4] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 108–114, IEEE, 2014.

[5] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-oriented bayesian networks for detection of lane change maneuvers," *Intelligent Transportation Systems Magazine, IEEE*, vol. 4, no. 3, pp. 19–31, 2012.

[6] M. J. Druzdzel, "Smile: Structural modeling, inference, and learning engine and genie: a development environment for graphical decision-theoretic models," in *AAAI/IAAI*, pp. 902–903, 1999.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search*, vol. 81. MIT press, 2000.

[9] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[10] M. Kochenderfer, *AA228: Decision Making under Uncertainty*. Stanford Bookstore, 2014.

[11] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[12] A. S. Hesar, H. Tabatabaee, and M. Jalali, "Structure learning of bayesian networks using heuristic methods," *International Proceedings of Computer Science & Information Technology*, vol. 45, 2012.

8 Appendix

		Core Features	
ψ^F	rad	cont.	heading angle in the Frenet frame
$\dot{\psi}$	$\mathrm{rad\,s^{-1}}$	cont.	turn rate
$\|v\|$	$\mathrm{m\,s^{-1}}$	cont.	speed
v_x^F	$\mathrm{m\,s^{-1}}$	cont.	longitudinal speed in lane
v_y^F	$\mathrm{m\,s^{-1}}$	cont.	lateral speed in lane
a_x^F	$\mathrm{m\,s^{-2}}$	cont.	longitudinal acceleration in lane
a_y^F	$\mathrm{m\,s^{-2}}$	cont.	lateral acceleration in lane
		Roadway Features	
$lane$	-	disc.	index of the closest lane
nll	-	disc.	number of lanes to the left
nlr	-	disc.	number of lanes to the right
d_{cl}	m	cont.	lateral distance between center of car and closest centerline
d_{ml}	m	cont.	lateral distance between center of car and the left marker
d_{mr}	m	cont.	lateral distance between center of car and the right marker
d_{onramp}	m	cont.	distance along the RHS until next onramp
$d_{offramp}$	m	cont.	distance along the RHS until next offramp
$ttcr_{ml}$ m	cont.	time to crossing of left lane marker	
$ttcr_{mr}$ m	cont.	time to crossing of right lane marker	
$v_{x,scene}^F$	$\mathrm{m\,s^{-1}}$	cont.	the mean velocity of vehicles in the scene
a_y^{req}	$\mathrm{m\,s^{-2}}$	cont.	the acceleration required to stay in lane
κ	$\mathrm{m^{-1}}$	cont.	local lane curvature
		Vehicle Relative Features	
$d_{x,r}^{rel}$	m	cont.	longitudinal distance between observer and related vehicle r
$d_{y,r}^{rel}$	m	cont.	lateral distance between observer and related vehicle r
$v_{x,r}^{rel}$	$\mathrm{m\,s^{-1}}$	cont.	longitudinal relative speed between observer and related vehicle r
$v_{y,r}^{rel}$	$\mathrm{m\,s^{-1}}$	cont.	lateral relative speed between observer and related vehicle r
ψ_r	rad	cont.	heading of the related vehicle
$\dot{\psi}_r$	$\mathrm{rad\,s^{-1}}$	cont.	turn rate of the related vehicle
a_r^{req}	$\mathrm{m\,s^{-2}}$	cont.	longitudinal acceleration required to avoid a collision with the related vehicle r
$ttc_{x,r}$	s	cont.	time to a longitudinal collision with related vehicle r
$\tau_{x,r}$	s	cont.	timegap between observer and related vehicle r
		Templated Features	
$\hat{a}_{x,H}^F$	$\mathrm{m\,s^{-2}}$	cont.	maximum longitudinal acceleration over history H
$\hat{a}_{y,H}^F$	$\mathrm{m\,s^{-2}}$	cont.	maximum lateral acceleration over history H
$\hat{\dot{\psi}}_H$	$\mathrm{rad\,s^{-1}}$	cont.	maximum turn rate over history H
$\bar{a}_{x,H}^F$	$\mathrm{m\,s^{-2}}$	cont.	mean longitudinal acceleration over history H
$\bar{a}_{y,H}^F$	$\mathrm{m\,s^{-2}}$	cont.	mean lateral acceleration over history H
$\bar{\dot{\psi}}_H$	$\mathrm{rad\,s^{-1}}$	cont.	mean turn rate over history H
$\sigma(a^F)_{x,H}$	$\mathrm{m\,s^{-2}}$	cont.	standard deviation of longitudinal acceleration over history H
$\sigma(a^F)_{y,H}$	$\mathrm{m\,s^{-2}}$	cont.	standard deviation of lateral acceleration over history H
$\sigma(\dot{\psi})_H$	$\mathrm{rad\,s^{-1}}$	cont.	standard deviation of turn rate over history H

Table 8: All indicator features

8.1 Feature Selection Results

Algorithm	60 Features	162 Features
Feature Ranking	0.62 s	1.68 s
Forward Search	2.37 s	8.97 s
Graph Search	2.43 s	8.94 s

Table 9: Feature selection algorithm runtime comparison. Note that this is an offline process so the timing difference for such small values is not of great concern.

60 Features	Score	Indicators
Feature Ranking	-54532	$\dot{\psi}$, v_y^F, ψ, a_y^F
Forward Search	-54506	$\dot{\psi}$, v_y^F, d_{cl}
Graph Search	-54506	$\dot{\psi}$, v_y^F, d_{cl}

Table 10: Feature selection for $\dot{\phi}^{\text{fut}}_{250ms}$ with 60 features

162 Features	Score	Indicators
Feature Ranking	-52141	$\dot{\psi}$, a_y^F, $\hat{\dot{\psi}}_{100ms}$, $\hat{\dot{\psi}}_{150ms}$, $\hat{a}^F_{y,100ms}$, $\bar{\dot{\psi}}_{100ms}$
Forward Search	-50433	$\dot{\psi}$, v_y^F, $\hat{\dot{\psi}}_{150ms}$, $\hat{a}^F_{y,750ms}$
Graph Search	-49957	$\dot{\psi}$, v_y^F, $\hat{\dot{\psi}}_{100ms}$, $\bar{\dot{\psi}}_{750ms}$

Table 11: Feature selection for $\dot{\phi}^{\text{fut}}_{250ms}$ with 162 features

60 Features	Score	Indicators
Feature Ranking	-148717	$\dot{\psi}$, v_y^F, ψ, a_y^F, d_{cl}, $ttcr_{mr}$
Forward Search	-148814	$\dot{\psi}$, v_y^F, ψ, $v^F_{scene,x}$
Graph Search	-148641	$\dot{\psi}$, v_y^F, ψ, d_{mr}, a^{req}_{stay}

Table 12: Feature selection for $\dot{\phi}^{\text{fut}}_{500ms}$ with 60 features

162 Features	Score	Indicators
Feature Ranking	-151118	$\dot{\psi}$, v_y^F, $\hat{\dot{\psi}}_{150ms}$, $\hat{\dot{\psi}}_{200ms}$, $\hat{a}^F_{y,750ms}$
Forward Search	-144201	$\dot{\psi}$, v_y^F, $\hat{\dot{\psi}}_{200ms}$, $\hat{\dot{\psi}}_{500ms}$, $\hat{a}^F_{y,1s}$
Graph Search	-143977	$\dot{\psi}$, a_y^F, $\hat{\dot{\psi}}_{100ms}$, $\hat{\dot{\psi}}_{150ms}$, $\hat{\dot{\psi}}_{200ms}$, $\hat{a}^F_{y,100ms}$, $\bar{\dot{\psi}}_{100ms}$, $\bar{\dot{\psi}}_{150ms}$

Table 13: Feature selection for $\dot{\phi}^{\text{fut}}_{500ms}$ with 162 features

60 Features	Score	Indicators
Feature Ranking	-80806	a_x^F, v_x^F
Forward Search	-80731	a_x^F, v_x^F, nll
Graph Search	-80731	a_x^F, v_x^F, nll

Table 14: Feature selection for a^{fut}_{250ms} with 60 features

162 Features	Score	Indicators
Feature Ranking	-79107	a_x^F, $\hat{a}^F_{x,100s}$, $\hat{a}^F_{x,150s}$, $\hat{a}^F_{x,200s}$, $\hat{a}^F_{x,250s}$, $\bar{a}^F_{x,100ms}$, $\bar{a}^F_{x,150ms}$, $\bar{a}^F_{x,200ms}$, $\bar{a}^F_{x,250ms}$, $\bar{a}^F_{x,500ms}$
Forward Search	-76500	a_x^F, $\lvert v\rvert$, $\hat{a}^F_{x,200s}$, $\hat{a}^F_{x,750s}$, $\bar{a}^F_{x,500ms}$
Graph Search	-76615	a_x^F, $\lvert v\rvert$, $\hat{a}^F_{x,150s}$, $\hat{a}^F_{x,500s}$, $\bar{a}^F_{x,200ms}$

Table 15: Feature selection for a^{fut}_{250ms} with 162 features

60 Features	Score	Indicators
Feature Ranking	-46145	a_x^F, v_x^F
Forward Search	-45285	a_x^F, v_x^F, nll
Graph Search	-45285	a_x^F, v_x^F, nll

Table 16: Feature selection for a_{500ms}^{fut} with 60 features

162 Features	Score	Indicators
Feature Ranking	-41826	a_x^F, $\hat{a}_{x,100ms}^F$, $\bar{a}_{x,100ms}^F$, $\bar{a}_{x,150ms}^F$, $\bar{a}_{x,200ms}^F$, $\bar{a}_{x,250ms}^F$, $\bar{a}_{x,500ms}^F$, $\bar{a}_{x,750ms}^F$
Forward Search	-39838	a_y^F, v_x^F, $\bar{a}_{x,100ms}^F$
Graph Search	-39838	a_y^F, v_x^F, $\bar{a}_{x,100ms}^F$

Table 17: Feature selection for a_{500ms}^{fut} with 162 features