<http://www.diva-portal.org>

This is the published version of a paper presented at *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*.

Citation for the original published paper:

Ward, E., Folkesson, J. (2015)

Multi-classification of Driver Intentions in Yielding Scenarios.

In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on* (pp. 678-685).

<http://dx.doi.org/10.1109/ITSC.2015.116>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-183496>

Multi-classification of Driver Intentions in Yielding Scenarios

Erik Ward¹, John Folkesson¹

Abstract—Predictions of the future motion of other vehicles in the vicinity of an autonomous vehicle is required for safe operation on trafficked roads. An important step in order to use proper behavioral models for trajectory prediction is correctly classifying the intentions of drivers. This paper focuses on recognizing the intention of drivers without priority in yielding scenarios at intersections – where the behavior of the driver depends on interaction with other drivers with priority. In these scenarios the behavior can be divided into multiple classes for which we have compared three common classification algorithms: k-nearest neighbors, random forests and support vector machines. Evaluation on a data set of tracked vehicles recorded at an unsignalized intersection show that multiple intentions can be learned and that the support vector machine algorithm exhibits superior classification performance.

I. INTRODUCTION

Vehicle autonomy has progressed to the point that commercially available systems can handle some simple scenarios such as parking and lane keeping, reducing the cognitive burden on the driver. More complex scenarios involving multiple agents interacting in more complex patterns is still a challenge. When dealing with multiple agents one would like to predict the future trajectories of the agents. The benefit of accurate prediction of the trajectories of other vehicles in these scenarios is primarily the correct appraisal of the risks associated with various driving decisions, such as entering the intersection.

In order to predict these trajectories the driver intention must be recognized so that a proper behavior model for the motion can be used. Often the question of recognizing the driver intention is simplified to a binary classification, for example, is a car stopping or not? However, the real situation can be more nuanced than that with several different possible intentions each producing different predicted future motions. We look at just this multiple intention recognition problem. The hypothesis is that by matching the complexity of real world scenarios with our models more closely, subsequent decisions based on the models can be both safer and less conservative.

In these multi agent scenarios the intention of the driver cannot be correctly estimated based on the history of the driver's action alone. The relation to the other agents actions and the context must be taken into account. This causes an increase of input variables and a commensurate increase in the difficulty of accurate classifications. A first step to understanding these multi-agent problems is to look at the two agent classification. This is a step up from the single

Fig. 1. Here we show an example of the yield scenario studied here. The white vehicle approaching the line of yield markings is required to stop, and does so two seconds after this frame, to allow the red car to pass through the intersection. This image is taken from the video sequence used in our experiments.

independent agent case but is still a tractable problem. We will examine such a two agent case.

We compare different classification algorithms for inferring the intentions of drivers in yield scenarios at an intersection. The algorithms tested are support vector machine (SVM), random forest (RF) and k-nearest neighbors (k-NN). Intersection scenarios feature complex interaction between agents and provide a challenging problem for autonomous vehicle systems or advanced driver assistance systems (ADAS) applications and a hard problem for which to evaluate the classification algorithms. In 2013, 23% of Swedish serious accidents with fatal or severe injury were in intersections [1]. For intelligent vehicle applications we are often interested in the posterior probability of classes given sensor data, for example when evaluating the expected risk of a plan, and therefore we compare extracted probability estimates from the classification algorithms. Our main contributions are:

- Formulating intention classification as a multiclass problem with four different intentions.
- Comparing classification performance and analyzing convergence to the correct intention in give-way scenarios using SVM, RF and k-NN classifiers.

Section II describes related work in this area, our problem formulation is listed in section III, in section IV we describe the algorithms evaluated, in section VI results are presented and section VII contains conclusions.

¹ Erik Ward and John Folkesson are with the Centre for Autonomous Systems at KTH Royal Institute of Technology, Stockholm, SE-100 44, Sweden. {eward, johnf}@kth.se

II. RELATED WORK

The complex relationship between the intention of a driver along with the current context and the resulting trajectory is impossible to define, even for an expert. Therefore, machine learning methods have been used extensively to discriminate between different intended maneuvers. In [2] Aoude classified the behavior of drivers as compliant or violating to a red light at an intersection at different prediction times. He compared the traditional methods, required deceleration parameter (RDP), time to intersection (TTI) thresholding and speed-distance regression (SDR) to machine learning methods based on SVMs and HMMs. The behaviors were defined such that a violating driver does not stop even though the light is red and a compliant driver stops. A comparison of recall at 5% false positive rate showed that the SVM and HMM based models significantly outperformed the traditional methods. Each model was trained on data from a time window of measurements taken before the estimated time-to-intersection, TTI, fell below a threshold – the time instant that the classification performance was evaluated. The methods used did not account for any interaction between vehicles and for each prediction time a separate set of model parameters must be fitted to the data. In our approach, we do not focus the learning on a single short time window but instead evaluate the algorithms on longer data sequences.

In the case of the first car in a lane approaching a stop-sign or a red light one could argue that the correct behavior of the agent is independent of other traffic participants since the driver should always stop. For yield scenarios, the relative positions and velocities of multiple agents dictate the correct behavior and hence this information should be included in the models used to infer the behavior of agents. Lefèvre argued that the assumption of independent motion of vehicles does not hold in most situations, “Vehicles share the road with other vehicles, and the maneuvers performed by one vehicle will necessarily influence the maneuvers of the other vehicles. Inter-vehicle dependencies are particularly strong at road intersections, where priority rules force vehicles to take into account the maneuvers performed by the other vehicles” [3].

Graf studied behavior prediction for situations in intersections without traffic lights, involving two agents using an on-line learning approach. For each training trajectory, the correct label was found by looking at the whole trajectory. The goal was to classify if an agent would stop for another agent with precedence or not, corresponding to our behaviors *Stop* and *Go*, respectively, defined in section III. Graf used relative distances and speeds of the two vehicles to extract features. The features consisted of derivative information, means, variances and qualitative features of the time series by dividing the possible values into three levels (low, medium, high) and three trends (falling, constant, rising). Classification was performed by ranking a similarity score of all training examples [4].

Binary classification of agent intentions is inherently problematic in that it makes a crisp decision even in the

face of significant ambiguity. It is not helpful to know for example that it is most likely that the car will stop. By estimating probabilities of the type $P(intention|input)$ when planning a suitable action one is able to minimize the expected risk or weigh risks of collision against risks associated with overly conservative driving. Several authors have included a Bayesian filter component in order to filter the outputs of the underlying classification algorithm [2] [5]. However, this serves mostly as a way to avoid transient false detections and a separate threshold variable is fit in order to perform classification, while the quality of the estimated probability is not evaluated. Bayesian network models, which give probability estimates of intentions, have been explored by [6], [7] and others. These types of probabilistic models have the potential to explicitly encode domain knowledge but often one needs to include classification type models as local distributions in the network which are similar to the algorithms evaluated in this paper. In order to discriminate speed profiles of *Stop* and *Go* classes a likelihood function was created in [6] that compared the speeds of test examples with speeds of a data set of training trajectories while in [7] a logistic regression component fitted to a training set was used in order to estimate the probability of a subset of the possible intentions.

In [8] Gindele used a random forest for regression to fit a policy function for accelerations and yaw-rates of agents in intersection scenarios, where relations between agents such as distances between agents and difference in orientation was included in the feature set. Using a sampling based approach one can then estimate the probability distribution of future trajectories. Only simulated data was used for evaluation and the method was compared to Kalman filter tracking with respect to tracking performance. They did not directly perform intention classification.

III. PROBLEM STATEMENT

Fig. 2. Example scenario where agent A_y should yield to agent A_p . d_y denotes the distance to the yield line for A_y and d_p the distance until A_p clears the intersection.

In scenarios where there is a yield sign or marking on a lane leading into the intersection, each agent traveling on this *yield-lane*, $A_y \in Y$, should let agents traveling on other lanes, $A_p \in P$, pass through the intersection first if they are sufficiently close, where Y and P denote the set of agents traveling on the yield-lane and set of agents with precedence, respectively. An example scenario is shown in Fig. 2.

By examination of data from such an intersection we found that there are four classes of behaviors for agent A_y :

- 1) *No action* A_y determines that A_p will pass through the intersection before it with sufficient margin without any need for A_y to slow down or stop.
- 2) *Creep* A_y slows down but does not stop completely in order to let A_p pass through the intersection before it.
- 3) *Stop* A_y slows down and finally stops before the intersection and waits until A_p has passed.
- 4) *Go* A_y passes through the intersection before A_p . This could incur significant risk since it potentially contradicts the rules of the road.

Of these four it is correct classification of go scenarios that is safety critical. The other classes could allow us to drive in a less conservative manner in some situations.

For a specific intersection and each pair of interacting agents, $(A_y, A_p) \in I$, traveling through it, our goal is to classify the behavior of A_y when it has not entered the intersection and A_p has not cleared the intersection. Only scenarios when both agents are close in time to drive through the intersection are considered which corresponds the set of pairwise interacting agents $I \subseteq Y \times P$. Pairwise interactions are assumed independent. Each scenario has a start time t_0 and end time t_N and for each time instance $t \in [t_0, t_N]$ we wish to infer the correct class for the behavior of A_y : $y \in \{1, 2, 3, 4\}$, using only observations available up until t . The possible classes of behaviors for A_y correspond to different modes of longitudinal motion and not which route the agent takes – we assume that we know the routes of A_y and A_p . A detailed definition of what constitutes a scenario is listed in section V.

IV. ALGORITHMS

We evaluate the performance of three common classification algorithms: k-NN, SVM and RF. We are given a digital map of the intersection and estimated state of moving objects in the area. We extract the features, f , defined in section IV-B, that are input to the classification algorithm to determine the current intention y . At every time step the tracking system provides an estimate of the state of each tracked agent $x_t = (x, y, \theta, v, a)^T$, where x, y are coordinates in a fixed frame of reference, θ the heading, v the speed in the direction of θ and a the acceleration. If we have the full trajectory for each pair of agents in each situation, i.e. $x_{t_0:t_N}^y, x_{t_0:t_N}^p$, the behavior can be determined using a simple rule based labeling algorithm listed in section V. However, our goal is to provide a prediction of the behavior, or the intention, given only partial trajectories, with data from a starting time t_0 to the current time t . Fig. 3 shows an overview of the evaluation system.

The instance-based k-NN classifier works by directly comparing the training data and test examples to be classified – each test example is classified as having the label as the majority of its k nearest neighbors using a suitable distance metric in the feature space – we have used the euclidean distance. The algorithm is impractical for large data sets, one must store all the training data and compare against it, and

Fig. 3. System overview, features are extracted using a digital map representation of the intersection and filtered object tracks that are input to the classifier.

it is poor at generalizing. However, using a sufficiently large data set, it serves as a baseline for comparison with sparse methods. Probabilities are extracted by counting the number of neighbors for each class out of the k nearest neighbors as $P(y = i) = |nn_i|/k$.

A Support Vector Machine (SVM) is a discriminative binary classifier first introduced by Corinna and Vapnik [9]. It is a sparse classifier where the model parameters are a subset of the training set of feature vectors f_i , the support vectors. Training the SVM model is formulated as finding the separating hyper-plane between the two classes, assumed to have class labels $y \in \{-1, 1\}$, which has the maximum margin. Classification amounts to checking at which side of the hyper-plane a test instance is located. In its base form the SVM is a linear classifier, but it can be extended to a nonlinear classifier using the kernel trick, where dot products between the hyper-plane and feature vectors are implicitly performed in a higher dimensional space, mapped to by $\phi(f)$, using a kernel function $K(f_i, f_j) \triangleq \phi(f_i)^T \phi(f_j)$ which can allow separation of classes that are not linearly separable in the original data space. For training feature vectors $f_i, i = 1, \dots, l$ with labels y_i we optimize the maximum margin hyper plane specified by the normal vector w and distance from the origin b . To accommodate mislabeled training examples, slack variables ξ_i are introduced and the optimization problem becomes:

$$\begin{aligned}
 \min_{w, b, \xi} \quad & \frac{1}{2} w^T w + c \sum_{i=1}^l \xi_i \\
 \text{s.t.} \quad & y_i (w^T \phi(f_i) + b) \geq 1 - \xi_i \\
 & \xi_i \geq 0, i = 1, \dots, l
 \end{aligned}$$

The dual of this formulation is a convex quadratic optimization problem where $\phi(f_i)$ is not directly calculated, instead only the kernel function is used. Here c acts as a regularization parameter and for the experiments in section VI we have used the radial basis function (RBF) kernel $K(f_i, f_j) = \exp(-\gamma \|f_i - f_j\|^2)$, where γ is a parameter. Classification of a test example f amounts to checking the sign of the score $s = w^T \phi(f) + b$. Multi-class extensions of SVM are done by training several classifiers and voting –

we have used the one-against-one approach where for each unordered pair out of the k classes a SVM is trained and the final classification is generated by tallying the votes from the $k(k-1)/2$ classifiers. For unbalanced data sets, where some classes have far more data points, we can set different regularization parameters for each class using the parameter vector $C = (c_1, \dots, c_k)^T$ [10].

A random forest classifier uses an ensemble of decision trees that are trained using bootstrap aggregation. Each decision tree classifier in the ensemble represents a disjoint partitioning of the feature space into a set of axis aligned cuboids, each assigned one of the class labels, and classification corresponds to checking which cuboid contains a test example while the ensemble as a whole provides a classification by voting. The partitioning is learned for each decision tree by recursively dividing the feature space using thresholds on one dimension at a time. The thresholds are found using a greedy optimization strategy where an information theoretic measure such as the cross entropy is optimized to determine which dimension and what threshold to use at each step [11]. We have used Gini's diversity index. In a random forest, tree construction of the ensemble of decision trees is modified compared to standard decision tree algorithms – the dimensions for which to evaluate a split on at each step of the greedy optimization algorithm are chosen from random subset of size N_{sample} of the available features and no pruning is performed. This strategy performs well compared to many other classifiers and reduces over-fitting [12].

During training of the random forest ensemble, each of the N_{tree} decision tree classifiers is trained using a separate training set sampled from the original training set. For each decision tree, N data points are sampled from the original training set, with replacement, where N is the size of the original training set. Samples are selected so that the probability of selecting a sample from each of the is classes equal, compensating for unbalanced data sets by over-sampling minority classes to a larger extent than majority classes.

A. SVMP Classification

Besides the standard SVM formulation we also looked at a more refined multi-classification method using the SVM where we estimate the posterior probability $P(y|f)$. Starting from each of our trained SVMs, we fit a sigmoid $1/(1 + \exp\{A(s - B)\})$ to the output scores, s , of each pairwise classifier for classes i, j , as in [13] to estimate $r_{ij} \approx P(y = i|y = i \vee y = j, f)$, for example see fig. 4. Then we estimate the class probabilities $P(y = i|f)$ using the second approach of Wu, Lin and Weng [14] where a linear-equality constrained quadratic optimization problem is solved for $P(c = i|f)$. In order to fit the parameters A and B cross validation is performed on the training data. A classifier is trained on a disjoint subset of the training data and evaluated on the other subset in each fold to give a score for each training example. Using $\arg\max_i P(c = i|f)$

Fig. 4. Sigmoid fit to extract a probability, $P(y = 3) = 1 - P(y = 4)$, from the SVM classifier between classes 3 (*Stop*) and 4 (*Go*). Plus signs show $P(y = 3|score)$ extracted using Bayes rule from the histograms in Fig. 5.

Fig. 5. Histogram of SVM scores for classifier between classes *Go* (blue) and *Stop* (red)

as the classification rule, henceforth called SVMP, can give different results than the standard SVM formulation.

B. Features

In order to capture relevant information about the current context and the behavior of A_y , features are defined with respect to the lane where the car travels, its position, velocity and acceleration. The longitudinal velocity, v_l^y , and acceleration a_l^y are used as features and are calculated by projecting the velocity and acceleration estimated by the tracking system onto the center of the road, defined by a digital map. In order to capture the tendency of more aggressive drivers to take a wider turn, we use the lateral position in the lane p_{lat} as a feature. p_{lat} is defined as +1 when the center of gravity of the car is at the right lane boundary and -1 when it is at the left lane boundary as follows $p_{lat} = (-d_R + w/2)/(w/2)$ where d_R is the distance to the right lane boundary and w the current width of the lane. Using digital map data to extract features such as

distances to stop lines, longitudinal position and velocity and lateral deviation with respect to specific lanes is common [6], [8], [15] and by assuming that virtual lanes that encode the different possible routes through an intersection are known, lane-related features can be defined inside intersections.

In order to capture more than just the information contained in the current snapshot of the trajectory, statistics are calculated over a moving window of the last K_t time-steps and used as features as in [2]. We have also included the maximum of the feature since the first time-step: $t = 1$.

We argue that the necessity to act is strongly influenced by how close A_y is to the yield-line and how close A_p is from clearing the intersection and this is reflected in the features d_y , d_p , v^p , where v^p is the speed of agent A_p . The full list of features for time t are:

- v_t^y , $\mu_{t-K_t:t}(v_t^y)$, $\text{var}_{t-K_t:t}(v_t^y)$, $\text{max}_{1:t}(v_t^y)$
- a_t^y , $\mu_{t-K_t:t}(a_t^y)$, $\text{var}_{t-K_t:t}(a_t^y)$, $\text{max}_{1:t}(a_t^y)$
- p_{lat} , $\mu_{t-K_t:t}(p_{lat})$, $\text{var}_{t-K_t:t}(p_{lat})$, $\text{max}_{1:t}(p_{lat})$
- d_y
- d_p
- v^p

Here $\mu_{t-K_t:t}(x)$, $\text{var}_{t-K_t:t}(x)$, denotes the means and variance over for feature x over the last K_t time-steps and $\text{max}_{1:t}(x)$ the maximum of feature x since the onset of the scenario.

For all algorithms we use scaled features where for each dimension we scale the feature value by sorting all the training data and assigning the smallest value to 0 and the largest to 1 with a fixed increment in between and linearly interpolating between the scaled values of each the two closest feature values in the training data for each test example.

V. DATA SET

The data set consist of tracked vehicles in a four-way intersection, shown in Fig 1, near central Gothenburg recorded by a video camera placed on the roof of a nearby building looking down at the intersection. We have used scenarios where A_y is southbound (to the left in Fig. 1), i.e. turning right, and A_p is coming from the east and driving straight through the intersection. Tracks were generated using a Kalman filter based approach based on estimated 3D box shapes for vehicles [16]. The tracking system delivers data at 20Hz. A more detailed description of the sensor setup as well as the distribution of route-choices, path-crossing scenarios and other statistics are in [17]. From the original data set that contained both false and very short tracks, we excluded tracks that had a total track time of less than one second, were more than one meter outside road bounds or covered less than 25 meters of road.

To define what constitutes a scenario we define thresholds on the time margin of when both agents occupy the intersection. The time margin for collision, TMC is calculated using the distance to the yield line for A_y , d_y , and the distance until A_p clears the intersection, d_p , as in Fig 2. Specifically, $TMC = |d_y/v_y - d_p/v_p|$ where v_y , v_p are the longitudinal speeds for agent A_y and A_p , respectively. The time interval

Fig. 6. Speed profile for A_y in *Stop* (solid, red) and *Go* (dashed, blue) examples. We see that beyond 10 m the two classes are impossible to distinguish based on these velocities alone.

for the scenario, $[t_0, t_N]$ has starting time t_0 when $TMC < \tau_{TMC}$ or when $d_p/v_p < \tau_{TMC} \wedge v_y < 2m/s$. The latter case represents the case where the precedence agent, A_p , is close to the intersection but A_y has slowed down significantly. When either the precedence agent has cleared the intersection or when the yield agent has entered the intersection the scenario ends, corresponding to $d_p < 0 \vee d_y < -2m$. Some overlap is allowed for A_y since not all drivers stop before the yield line – some stop after it.

We looked at the distribution of speeds for all drivers traveling through the intersection and turning right when no other cars influence its speed. A threshold is calculated for the minimum velocity during creeping behavior as $\tau_{creep} = \bar{v}_{min} - \sigma_{v_{min}}$, where $\bar{v}_{min}$ is the average minimum speed of all agents and $\sigma_{v_{min}}$ is the standard deviation.

Stopping is defined as when A_y has a speed lower than $\tau_{stop} = 0.25m/s$. We also store whether or not A_p clears the intersection before A_y enters it using the indicator function $first(A_p)$ and the behavior y of A_p is defined as follows:

$$y = \begin{cases} 1 & \text{if } v_{min}^y \geq \tau_{creep} \wedge first(A_p) \\ 2 & \text{if } v_{min}^y < \tau_{creep} \wedge v_{min}^y \geq \tau_{stop} \wedge first(A_p) \\ 3 & \text{if } v_{min}^y < \tau_{stop} \wedge first(A_p) \\ 4 & \text{otherwise} \end{cases} \quad (1)$$

A tracked object in the yield lane is only assigned as A_y if there are no preceding vehicles in front of the intersection. The part of the total data set that we used consists of 369 situations where 171 have the class *No action*, 90 *Creep*, 95 *Stop* and 13 *Go* while the average situation last for 4.5 seconds. The numerical values of the thresholds are: $\tau_{TMC} = 6s$, $\tau_{creep} = 2.32m/s$.

The intersection features a fairly sharp turn to the right for the situations considered and therefore all A_y in our data set slow down considerably in order to make the turn, making it harder to distinguish the different classes the further the agent is from the intersection. This can be seen in Fig. 6 which

Fig. 7. Predicted class labels at different distances by SVM. Each histogram shows how the examples of a particular true class were classified as the vehicle approached the intersection.

TABLE I

PRECISION AND RECALL AT DIFFERENT d_y , 2-CLASS

d_y	Precision		Recall	
SVMP	$\neg$ Go	Go	$\neg$ Go	Go
2.5	1.00	0.50	0.92	1.00
5.0	0.99	0.33	0.89	0.92
7.5	0.99	0.23	0.90	0.75
10.0	0.99	0.21	0.92	0.67
SVM	$\neg$ Go	Go	$\neg$ Go	Go
2.5	0.99	0.83	0.99	0.83
5.0	0.97	0.50	0.97	0.50
7.5	0.96	0.00	0.98	0.00
10.0	0.98	0.25	0.98	0.17
k-NN	$\neg$ Go	Go	$\neg$ Go	Go
2.5	0.99	0.91	0.99	0.83
5.0	0.95	0.33	0.97	0.25
7.5	0.96	0.00	1.00	0.00
10.0	0.97	0.00	0.99	0.00
RF	$\neg$ Go	Go	$\neg$ Go	Go
2.5	0.96	0.86	0.99	0.50
5.0	0.95	0.43	0.98	0.25
7.5	0.97	1.00	1.00	0.12
10.0	0.98	1.00	1.00	0.17

TABLE II

PRECISION AND RECALL AT DIFFERENT d_y , 4-CLASS

d_y	Precision				Recall			
SVMP	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
2.5	0.90	0.82	0.81	0.55	0.82	0.83	0.66	1.00
5.0	0.88	0.65	0.62	0.43	0.76	0.71	0.47	1.00
7.5	0.86	0.63	0.64	0.23	0.79	0.68	0.33	0.88
10.0	0.84	0.60	0.62	0.15	0.82	0.54	0.33	0.83
SVM	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
2.5	0.85	0.80	0.78	0.67	0.82	0.79	0.70	1.00
5.0	0.86	0.67	0.67	0.64	0.83	0.71	0.62	0.75
7.5	0.81	0.62	0.62	0.25	0.85	0.68	0.42	0.38
10.0	0.81	0.56	0.58	0.18	0.89	0.58	0.35	0.33
k-NN	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
2.5	0.74	0.64	0.64	0.91	0.68	0.67	0.66	0.83
5.0	0.80	0.60	0.57	0.40	0.80	0.72	0.45	0.33
7.5	0.74	0.49	0.36	1.00	0.84	0.53	0.29	0.12
10.0	0.75	0.40	0.24	0.00	0.86	0.42	0.17	0.00
RF	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
2.5	0.97	0.82	0.72	0.80	0.91	0.83	0.77	0.67
5.0	0.86	0.64	0.55	0.36	0.86	0.72	0.47	0.33
7.5	0.80	0.60	0.45	0.40	0.87	0.60	0.42	0.25
10.0	0.79	0.51	0.36	0.50	0.86	0.52	0.30	0.33

TABLE III

PRECISION AND RECALL AT DIFFERENT t_{pred} , 4-CLASS

t_{pred}	Precision				Recall			
SVMP	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
1.0	0.92	0.70	0.94	1.00	0.93	0.81	0.77	1.00
1.5	0.91	0.65	0.87	0.92	0.91	0.73	0.77	0.92
2.0	0.91	0.64	0.79	0.76	0.92	0.64	0.74	1.00
SVM	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
1.0	0.91	0.74	0.95	1.00	0.95	0.81	0.78	0.92
1.5	0.90	0.70	0.85	1.00	0.94	0.72	0.80	0.69
2.0	0.88	0.58	0.77	1.00	0.92	0.58	0.77	0.46
k-NN	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
1.0	0.87	0.66	0.82	1.00	0.95	0.66	0.73	0.62
1.5	0.88	0.63	0.73	1.00	0.94	0.56	0.78	0.38
2.0	0.85	0.55	0.63	0.00	0.92	0.51	0.67	0.00
RF	No A.	Creep	Stop	Go	No A.	Crp.	Stop	Go
1.0	0.93	0.76	0.93	1.00	0.95	0.87	0.81	0.62
1.5	0.89	0.66	0.83	1.00	0.93	0.74	0.75	0.23
2.0	0.87	0.63	0.72	0.50	0.94	0.64	0.68	0.08

shows the speed profiles of A_y in *GO* and *Stop* examples.

The digital map used to define some of the features used in section IV-B has been created by inspecting a satellite photo of the intersection and manually outlining the edges of the lanes and positions of yield-lines in the lanelet format [18].

VI. RESULTS

The SVM, k-NN and RF classifiers have been evaluated on the data set described in section V using a Matlab implementation which used libSVM [10] for the SVM classifier. We train on data for all times $[t_0, T_N]$ and for each situation, we classify each data point separately. By grouping the results across all situations based on the time until the car passes the intersection, the prediction time, $t_{pred} = t_N - t$ or the distance d_y we can inspect the classifier performance. Since our data set is rather limited, especially considering the number of *Go* examples, 10-fold cross validation has been used where in each fold 90% of the situations are used for training and 10% for testing such that each trajectory is used for evaluation once.

Some parameters for the different classifiers are set a-priori, these are $K_t = 1s$ and for RF: $N_{sample} = 4$. Other parameters are set using grid search and an additional, nested, 3-fold cross-validation procedure for *each* 90%-10%

cross validation fold. For SVM the γ and C_{shared} parameters are set such that $C = C_{shared}(1, 1, 1, 10)^T$, for k-NN the number of nearest neighbors, k , and for RF the number of trees, N_{trees} . Table IV shows the parameter options that were evaluated and the macro-averaged F_1 -score (2) was used as the selection criteria.

TABLE IV
PARAMETER VALUES CONSIDERED DURING CROSS VALIDATION

γ	2^i	$i = -13, -11, \dots, 3$
C_{shared}	2^i	$i = -3, -2, \dots, 9$
k	$2^i + 1$	$i = 1, 2, \dots, 9$
N_{trees}	$10i$	$i = 1, 2, \dots, 30$

Table I shows precision and recall for the two class case, where classifiers are trained and evaluated on the classes *go* or *not go*, and tables II and III show the performance when using all four classes: *No action*, *Creep*, *Stop* and *Go*. The rows show the predictions at different times and distances. The most critical column in the tables is the far right, highlighted in red, which shows the recall of *go* examples for all cross validation folds. Each example here of a missed *go* class could lead to a collision if it is not corrected in time. We see that the SVM variants seems to perform best overall, where we trade of precision for recall with SVMP. The recall for the critical *Go* class is seen to be 100% by 1 second or 2.5 m of the collision point. The 2-class case is roughly equal in terms of classification performance, except for RF which performs better in the 4-class case. Not surprisingly the classification becomes better the closer to the intersection the vehicle is.

Table V shows the macro-averaged F_1 -score calculated as

$$F_1 = \text{mean}(F_i) \quad (2)$$

$$F_i = 2 \frac{PPV(i) * TPR(i)}{PPV(i) + TPR(i)}$$

where PPV is a vector of precision for the different classes and TPR is a vector of recall for the different classes. When looking at the F_1 scores, we can see that both SVM variants outperform the k-NN and RF classifiers. The axis aligned decision boundaries of the decision trees in the random forest ensemble are inadequate to discriminate between the different classes and the use of an RBF kernel improves the performance of the SVM classifier compared to the k-NN classifier.

Fig. 7 shows histograms of predicted class labels at different distances from the intersection for the SVMP classifier. Each of the histograms is for test data of a given class and shows the classification 'confusion' as the intersection

TABLE V
MACRO-AVERAGED F_1 SCORES.

d_y	SVM	SVM-P	k-NN	RF
2.5	0.79	0.78	0.72	0.81
5.0	0.72	0.66	0.58	0.60
7.5	0.57	0.57	0.46	0.54
10.0	0.52	0.52	0.35	0.52

is approached. We see that the general trend is that classification accuracy increases as the situation progresses. The poor precision for *Go* constitutes cases where A_p is far away from clearing the intersection, which represents cases with low risk as can be seen in Fig 8. When A_p gets closer the false positives decrease, which can also be seen in table III where precision and recall is listed for different t_{pred} : the time until either A_p clears the intersection, or A_y enters it. From an algorithm evaluation standpoint it is regrettable that we have no *Go* examples where A_y enters the intersection when A_p is very close. Evaluation of such high risk scenarios using simulated data is left as future work.

Fig. 8. Predicted class labels when the true class is *Stop* at different d_p by SVMP.

Until close to the stop-line there is very little that differs between a trajectory featuring braking aggressively or one that follows through with a *Go* maneuver. Across all scenarios, all drivers slow down considerably to make the turn. This explains the poor precision of *Go* classifications at long distances or long prediction times. However as the cars get closer to the intersection our classification performance improves.

Fig. 9 shows the fraction of test examples with class $y = i$ at different estimated posterior class probabilities $P(y = i|f)$. This gives some indication of how biased the probability estimates are. We can see that the k-NN and the RF classifiers performs well and that the SVMP gives very poor estimates.

VII. CONCLUSIONS

We have shown a more refined multiple class description, four classes compared to classes stop or not stop, of driver intentions in intersections with interactions can be learned and have indication that it should be possible to extract meaningful probabilities from k-NN and RF while fitting a sigmoid to a SVM classifier can lead to poor probability estimates, even though SVM has superior classification performance. These results need to be investigated further before firm conclusions can be made.

ACKNOWLEDGMENT

We gratefully acknowledge that this work was carried out within the iQMatic, VINNOVA funded project, number

Fig. 9. Fraction of class examples out of the four different classes at different estimated probabilities. Top left is for k-NN, top right RF and bottom center SMVP.

2012-04626. We would also like to thank Professor Michael Felsberg at Linköpings Universitet for providing the data for this paper.

REFERENCES

- [1] B. Saxton, "Road traffic injuries 2013," Trafikanalys, Torsgatan 30, 113 21 Stockholm, Tech. Rep., 6 2014. [Online]. Available: [\url{http://trafa.se/PageDocuments/Vaegtrafikskador_2013.pdf}](http://trafa.se/PageDocuments/Vaegtrafikskador_2013.pdf)
- [2] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, "Driver behavior classification at intersections and validation on large naturalistic data set," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 2, pp. 724–736, 2012.
- [3] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech Journal*, vol. 1, no. 1, pp. 1–14, 2014.
- [4] R. Graf, H. Deusch, F. Seeliger, M. Fritzsche, and K. Dietmayer, "A learning concept for behavior prediction at intersections," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 939–945.
- [5] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 797–802.
- [6] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, "Risk assessment at road intersections: Comparing intention and expectation," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 165–171.
- [7] S. Klingelschmitt, M. Platho, H.-M. Gros, V. Willert, and J. Eggert, "Combining behavior and situation information for reliably estimating multiple intentions," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 388–393.
- [8] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *Intelligent Transportation Systems Magazine, IEEE*, vol. 7, no. 1, pp. 69–79, 2015.
- [9] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [10] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [11] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 4, no. 4.
- [12] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [13] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.
- [14] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *The Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.
- [15] D. Petrich, T. Dang, G. Breuel, and C. Stiller, "Assessing map-based maneuver hypotheses using probabilistic methods and evidence theory," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 995–1002.
- [16] B. Johansson, J. Wiklund, P.-E. Forssén, and G. Granlund, "Combining shadow detection and simulation for estimation of vehicle size and position," *Pattern Recognition Letters*, vol. 30, no. 8, pp. 751–759, 2009.
- [17] J. Cano, J. Kovaceva, M. Lindman, and M. Brannstrom, "Automatic incident detection and classification at intersections," *Volvo Car Corporation*, pp. 09–0245, 2009.
- [18] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 420–425.