Title: 4-Way-Stop Wait-Time Prediction
Group members (1): David Held

As part of my research in Sebastian Thrun's autonomous driving team, my goal is to predict the wait-time for a car at a 4-way intersection. This will enable us to plan what our autonomous car should do at such an intersection. By running the classifier on our car at in intersection, we could use the result to determine when we should enter the intersection.

Data was collected from actual cars approaching 4-way intersections. Our car was parked near the intersection of Stock Farm and Oak Road, and data was recorded of cars passing through the intersection. The data was collected using a Velodyne laser, which returns a point-cloud of position and intensity information for points surrounding the laser. Using this sensor, we can track the position of each car that is driving through the intersection, and observe how its position changes over time. Although work has been done to automatically detect and track cars from the point-cloud data, for this project, I manually labeled the car positions at each frame (with linear interpolation where possible) to ensure clean and accurate data.

Work is also being done to predict which direction the car is turning, based on lane position, turn-signal, and other cues. However, for the purposes of this project, I am assuming that we know the turn-direction perfectly. Rather than manually label this information, I automatically check in future frames which direction the car actually turns and use that information a priori, as if we predicted it in real-time. The only information that we extract from these future frames is the turn-direction (left, right, straight); we do not take advantage of knowing any future position information.

The results of the car labeling can be seen in figure 1. The position of each car is manually marked using a visual car marking and interpolation tool. Each yellow square in figure 1 indicates the position of a car in a given frame. After marking the positions of cars for some small number of frames, linear interpolation is used to predict the position of the car at all time frames. The blue car shown at the bottom of the figure is a stationary car which holds the laser used to record the above data.

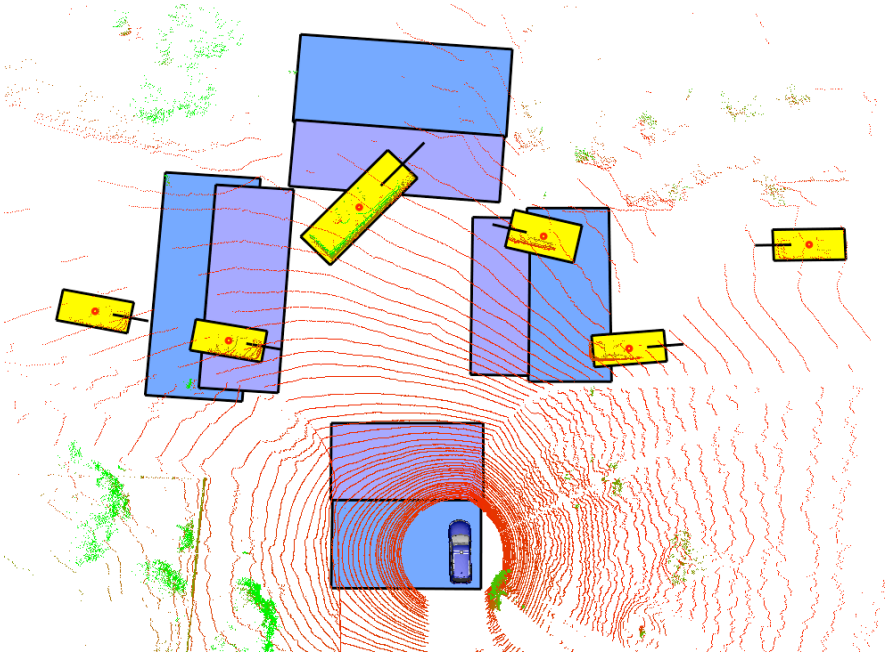

Figure 1. Yellow rectangles indicate car positions Light blue rectangles are “wait regions”, in which a car is considered to be waiting to enter the intersection. Light purple rectangles are pedestrian crosswalks.

I have marked the tracks of 74 cars using the above tool. Of these, 52, will be used for training, and 22 will be used for test. Each of the 52 training cars represents 544 positive examples (for each time frame in which the car will be entering the intersection within the next second) and 945 negative examples. Each of the 22 cars in the test set represent 232 positive examples and 412 negative examples.

To quantify how long each car has waited at the intersection, a region is delineated in each direction which is marked as the "wait region." These regions are shown in light blue in Figure 1. Any car within this region is considered to be "waiting" at the intersection. I am attempting to predict how long each car spends in this region, measured from the time that the car first enters the region, until the time that the car leaves the region. Using this "wait region" allows us to measure wait-times using position information alone, instead of relying on noisy velocity estimates. The distribution of wait times can be seen in figure 2. The mean wait time for all cars is 2.8215 seconds, with values ranging from 0.99 seconds to 6.93 seconds.

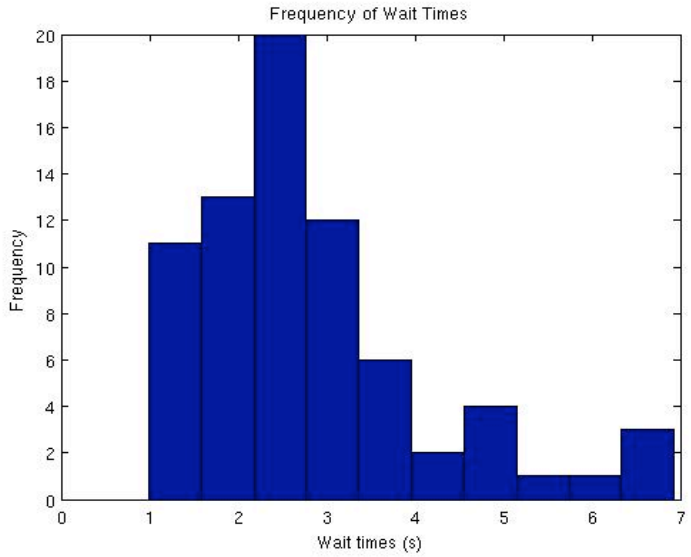

Figure 2. Distribution of wait-times at an intersection, for all cars in the training set.

My hypothesis will attempt to predict whether a car will leave the wait region in the next 1 second. Thus, this task is framed as a classification task, in which the positive class indicates that the car will enter the intersection (leave the wait region) in the next second, and the negative class indicates that the car will continue to wait in the wait-region for the next second. Each car is classified at each time step, based on the updated feature values at that time step.

The problem could alternatively have been framed as a regression problem, attempting to predict the number of seconds until the car enters the intersection. However, this seems both more difficult as well as less useful. For example, if the car is going to wait 15 seconds before entering the intersection, it will not be very helpful to attempt to predict that the car will be waiting more than 10 seconds and less than 20 seconds; the important information to extract from this scenario is that the car cannot enter the intersection at the present moment.

As a baseline, I used as a sole feature the number of seconds the car has already been waiting to enter the intersection. Intuitively, cars that have been waiting for longer are more likely to leave the wait region and enter the intersection. The distribution of this feature (scaled by a factor of 1/6) for both positive and negative examples can be seen in figure 3 below.

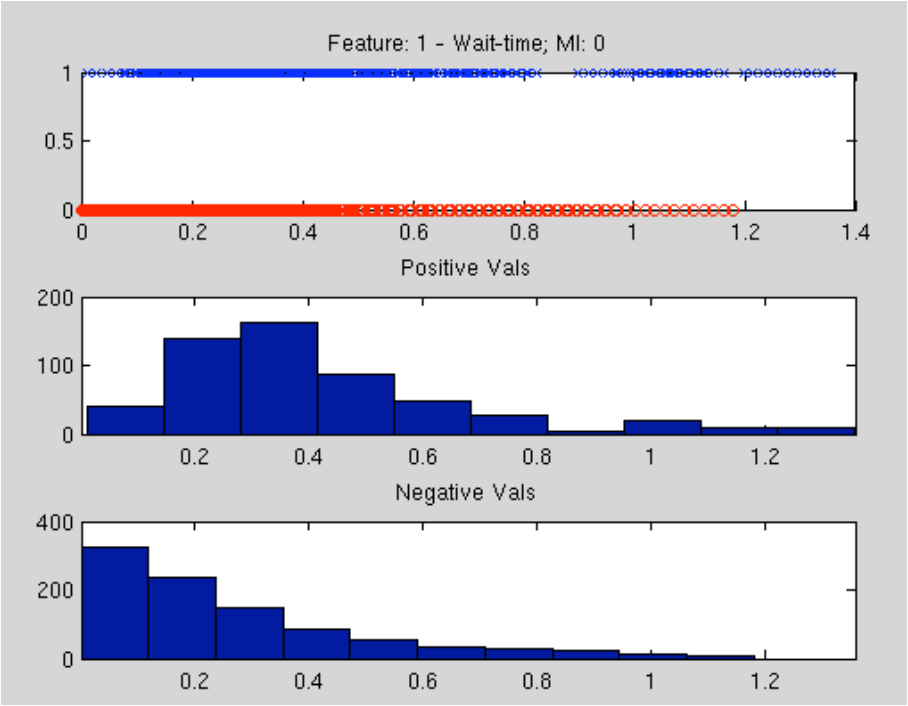

Figure 3. Wait time, for a car waiting to enter an intersection. The top graph shows the range of values for positive examples ("1"), for a car that is about to enter the intersection in the next second, or negative examples, "0", for a car that is still waiting at the intersection. The middle and bottom plots show the distribution of wait-times for positive and negative examples.

An SVM is trained using the training data described above, using a Gaussian kernel. The baseline result is shown as the blue line in figure 4. Using only this feature, we obtain an average precision of 0.657. By adding 5 more features, we are able to obtain an average precision of 0.795. Finally, by changing the regularization parameter C, we are able to achieve an average precision of 0.811.

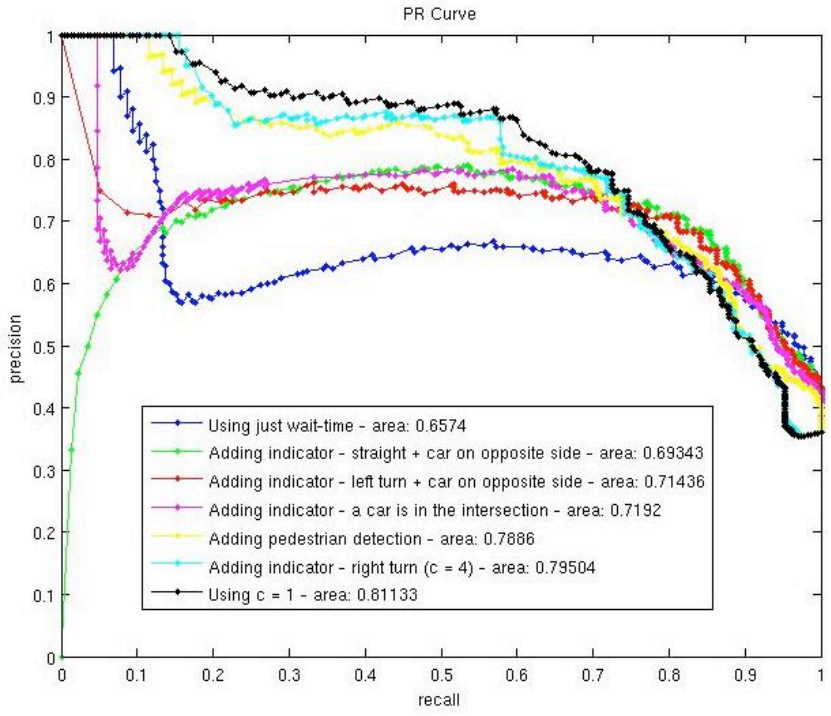

Figure 4. Precision-recall curves after training on an SVM. Each line is the result after adding a new feature.

Six features were used to train the SVM. The first feature, mentioned above, is the number of seconds that a car has already been waiting at the intersection. Each of the 5 other features that are added are indicator variables. To produce the best results for the Guassian kernel, this indicator is multiplied by the number of seconds that the car has been waiting at the intersection (which is itself also used as a feature). This seemed to produce better results than using the indicator variables directly, even with a quadratic kernel, which would represent a feature vector that includes the product terms between individual features. The improved performance using the Gaussian kernel is probably because a quadratic kernel would include other second-order terms that are less useful.

The first indicator represents if the car being classified wants to drive straight (not turn) and if there is a car waiting on the opposite side of the intersection. Presumably, if the target car thinks that the opposite car might turn left, then the target will wait for a little bit longer before entering the intersection. The next feature used is an indicator of whether the target car is turning left and if there is a car waiting on the opposite side of the intersection (which presumably might want to go straight). Another indicator represents whether a car is currently in the middle of the intersection. Next, we have an indicator of whether there is a pedestrian crossing the road in a place that blocks the target car from proceeding through the intersection. Crosswalks are shown as the purple regions in figure 1 above and figure 5 below.

Figure 5. A pedestrian is marked in the top center as a small white square. The two cars (on the left and right) need to wait for the pedestrian to cross before they can proceed through the intersection.

Last, an indicator is used to represent whether a car wants to make a right-turn (presumably this will cause it to go faster through the intersection).

Adding additional features causes the average precision to drop dramatically. This is most likely because we do not have enough training examples to learn in a more expressive feature-space. This can be demonstrated by observing the change in training error as we add these features. Interestingly, before adding these new features, the training error is higher than the test error. This is most likely because the training set has many more examples, and represents a much more diverse range of scenarios than the more limited scenarios that appear in the test set. Thus, while our features may be expressive enough to accurately classify cars in the test set, they are not expressive enough to predict car wait times in the wide array of scenarios from the training set.

Figure 6 shows the training error. Run 242 (blue) is a result of training with the 6 features described above, and the SVM gives an average precision of 0.688. Adding 6 more features increases the average precision on the training set to 0.82. The first new feature added is the area of the target car. Generally, larger cars tend to wait for a longer amount of time than smaller cars. Next, the car's approach velocity is roughly estimated by

computing the derivative of its position for successive frames. A car with a faster approach velocity might be expected to drive more aggressively and wait for less time at the intersection. The remaining features indicate the priority of the target car compared to other cars at the intersection. Priority is determined by the order in which the cars arrived at the intersection. Although of these features improved the training error, we do not currently have enough data to be able to use these features to reduce the test error; adding in any of these features results in increasing the test error due to overfitting.

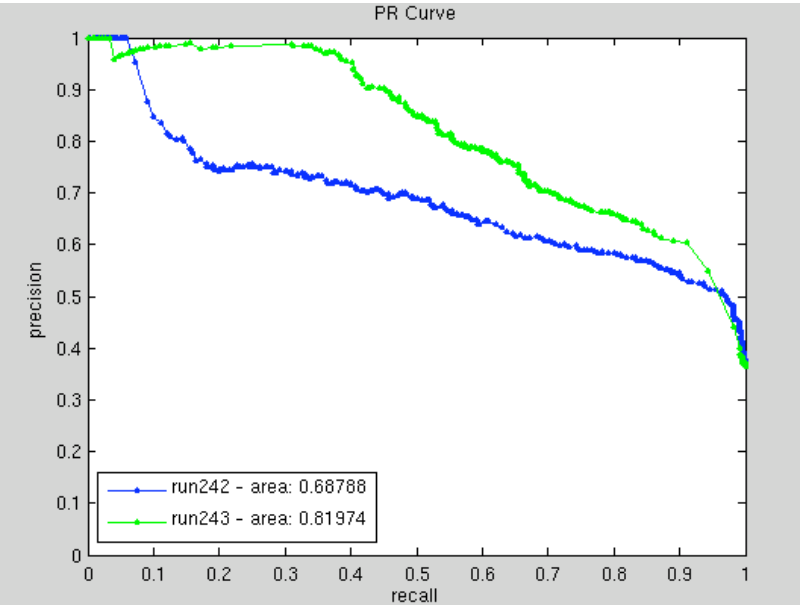

Figure 6. Training error. Run 242 (blue) is the training error using the 6 features described above and used for obtaining the test-error in figure 4. Run 243 is the training error resulting from training the SVM on 6 additional features.

More features can additionally be added to further reduce the test error. More accurate information of pedestrian location can be used to determine when a pedestrian is sufficiently out of the intersection to allow a car to proceed (even if the pedestrian has not entirely left the cross-walk). Further, more detailed information about the direction that a car is turning and the priority information of nearby cars can also be included. Finally, a more complicated framework can be used in which each car attempts to estimate the aggressiveness of each other car, based on its approach velocity, behavior (such as “inching forward” toward the intersection), car type, color, etc. Based on this estimated aggressiveness, a car will determine whether it can safely enter the intersection, or whether it will need to wait for the other cars.

It is also significant to mention that for the purposes of controlling an autonomous car, precision in the above task is more important than recall. A high precision indicates that, if the classifier predicts that a car should go in the next second, then indeed, a large percentage of the time, the target (human-driven) car will actually go in the next second. Used on an autonomous car, the classifier can be used to predict if the autonomous car can safely enter the intersection. A low precision corresponds to many false positives; this would be dangerous, because in this case, the classifier would predict that the autonomous car should enter the intersection at an unsafe time. On the other hand, a low recall corresponds to many “misses”; this indicates that the target (human-driven) car is entering the intersection, but we fail to predict this event. For classifying an autonomous car, this would simply mean that our autonomous car would wait at an intersection longer than necessary. On the other hand, if we are building a framework where we attempt to predict the actions of other cars, then we would want high recall – this would correspond to accurately predicting when each of the nearby cars might enter the intersection.

This research will hopefully be useful as part of a longer-term goal of building a planner to control an autonomous car approaching an intersection, to decide how long it should wait. By continuing to further improve the results, we will hopefully be able to accurately predict when our autonomous car can safely enter the intersection.