Data-driven Detection of Congestion-affected Roads

Technical Report TUD–KE–2014–02

Timo Nolle
Technische Universität Darmstadt
Immanuel Schweizer
Telecooperation Group, Technische Universität Darmstadt
Frederik Janssen
Knowledge Engineering Group, Technische Universität Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Knowledge
Engineering

Abstract

Traffic congestion wastes time, energy, and, thus, money. While well understood on highways, detecting congestion in dense urban centers is much harder. Recently, floating car data has been proposed to provide congestion information even in urban centers. However, this is still not fine-grained enough to distinct different lanes and sometimes actually also directions. Yet the more basic question of *Which lane is even affected by congestion?* cannot be answered today without unsustainable manual effort.

We will present an approach that uses inductive loop data to automatically classify congestion-affected roads, i.e., ones that are likely to be affected by traffic jams. Relying on measurements of both *load* and *count* for a given sensor, we found that most sensors exhibit one of two different patterns. One shows a linear correlation between the number of cars and the utilization. These roads are not affected by congestion. Congestion manifests as an atypical increase in *load* without increase in the number of cars.

By mathematically modeling these two patters, we are able to distinct both classes (congestion-affected sensors from non-congestion-affected sensors). Using this approach, we were able to correctly identify about 90% of the 499 sensors that are installed on roads in Darmstadt (Germany). While there is still room for improvement, this paper can be seen as a first step towards a fully automatic detection of congestion and congestion-affected roads in dense urban centers.

Contents

1 Introduction

The Texas A&M Transportation Institute has estimated that in 2011 congestion caused around 427 million extra hours of travel time and cost around $10 billion[1] in the US alone. However, congestion in dense urban areas is indeed still not understood well. Mostly, because a manual gathering of the data is unsustainable given the number of streets.

In recent years floating car data has been proposed as a means to detect congestion even in urban areas [3, 2]. Given the lack in data density and location accuracy, this data is still not fine-grained enough to understand congestion on a per-lane and per-direction level. Even basic questions, e.g., *Is this road even affected by congestion during its lifetime?* are, thus, hard to answer.

In this paper we want to introduce an approach to model congestion using data from inductive loop sensors embedded into the streets. Due to the known location and high accuracy of these sensors we can detect congestion on a per-sensor level yielding the most fine-grained method for detecting congestion yet. Given data about sensor utilization and the number of cars, we are able to model congestion using a custom error function and linear regression. Based on this model we classify congestion-affected sensors with almost 90% accuracy using different threshold-based approaches.

The remainder of this paper is organized as follows. The data and mathematical model is introduced in Section 2. Section 3 introduces the classifier, which is then evaluated in Section 4. The paper is concluded in Section 6.

[1] http://mobility.tamu.edu/ums/

2 Data Model

The challenge of this paper is the automatic classification of congestion-affected roads. Which roads in a city are actually prone to congestion? Before answering this question, we need to understand the input data and how to model congestion given that input data.

In the following sections we will, thus, first introduce and understand the data sources available. Next, we introduce our approach to mathematically model congestion given this data. More precisely, we will mathematically model the difference between roads that are affected or not affected by congestion.

Thus, the following two sections are dedicated to the input data sources and the congestion model respectively.

2.1 Data Sources

All traffic lights in Darmstadt, Germany are equipped with inductive loops. Each inductive loop is one sensor reporting *count* and *load* for each one minute interval. Here, *count* is the number of cars that have passed over the sensor during that one minute interval. *load* is the percentile of time the sensor was active. E.g., if the sensor was active for 30 seconds during the last minute interval the *load* is reported as 50%.

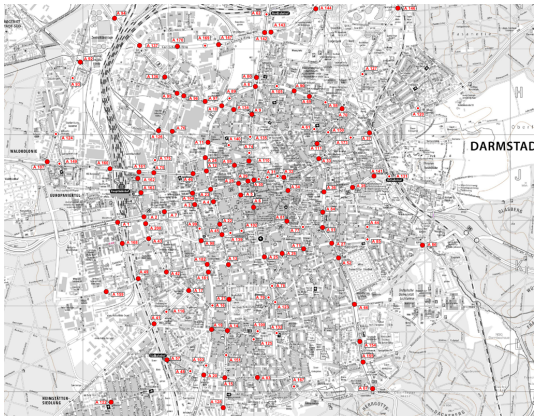

Figure 2.1: Overview over all traffic lights with sensors in the city center of Darmstadt (Germany)

Figure 2.1 illustrates the amount of traffic lights for Darmstadt center. Each traffic light is equipped with multiple inductive loops (sensors) for the different directions and lanes. Each sensor reports two values per minute. Using the complete data set between the 1st and the 31st of January 2014 the results of this paper are based on 20.254.589 raw measurements.

This data is preprocessed to filter faulty data and to discretize the time.

2.1.1 Preprocessing

The inductive loops record data for one minute intervals. This time interval is too small to measure traffic accurately, hence preprocessing of these values had to be performed. Roundtrip times for traffic lights can be as high as 90 seconds, i.e., using one minute intervals on such roads would not even capture one round trip. To deal with this, we average *load* and *count* over the last 15 consecutive one minute intervals for one specific sensor, leaving us with more stable 15 minute intervals. During preprocessing, we also removed sensors that did not produce any values at all. After this initial preprocessing 499 sensors are left for classification.

Additionally, for the mathematical model, the *count* per sensor has been normalized. Each *count* is divided by the maximum *count* for this sensor. Hence the normalized counts are between 0 and 1.

2.1.2 Observations

Before we model congestion it is important to note a number of important observations.

We start by taking a closer look at one crossing (cf. Fig. 2.2). First, congestion is a local phenomenon. While a road might be congested, it makes more sense to classify congestion on a sensor level. It might be that only one lane

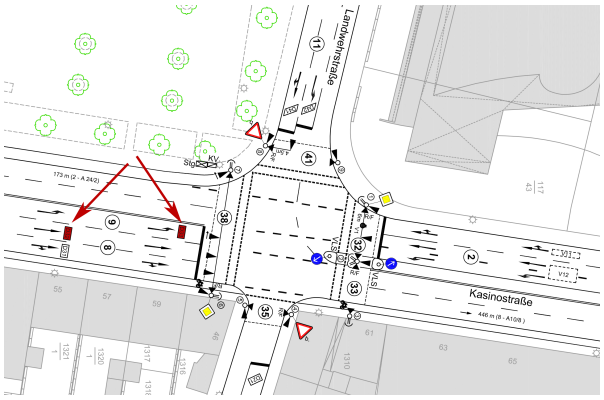

Figure 2.2: Overview over all traffic lights with sensors in the city center of Darmstadt (Germany)

is congested (e.g., the lane for turning left); it might be that only one direction is congested. Classifying each sensor allows for the most fine-grained congestion analysis. Second, considering only the *count* is not enough. Depending on speed and other factors each street (segment) has a unique capacity. Hence, we need to combine *load* and *count* to model congestion (more in Section 2.2). Third, the distance from sensor to traffic light matters. In Figure 2.2 two sensors are highlighted. One sensor is placed close to the traffic light. During red light one car is enough to keep the sensor active, thus increasing the *load*. The other sensor is placed further away from the traffic light. If only one car needs to stop at the red light the *load* will not increase.

Given this observations (especially the third), is it even possible to detect patterns in the data?

Figure 2.3 and Figure 2.4 plot all measurements for two different sensors. These two sensors show different patterns in their distribution. In Figure 2.3 we observe a near linear relation between the *count* and the *load*. Each additional car seems to have the same impact on *load*.

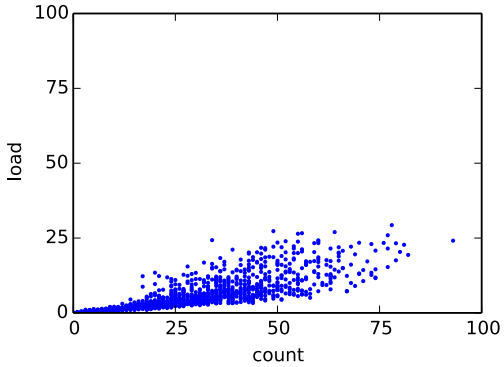

Figure 2.3: A sensor that is not congestion-affected (linear pattern)

The second sensor (cf. Fig. 2.4) shows a different pattern, which we could recognize for many other sensors as well. At some point the linear relation between *count* and *load* breaks. Each new car suddenly has a much higher impact on *load* until the road hits its capacity limit and the *count* goes down even though the *load* further increases. Hence, this turning point is exactly where the maximum capacity of the sensor, hence the road, is reached. Up to 100 cars in 15 minutes the traffic is dense but still flowing. And this indicates a reduction of flowing speed or, more specifically, congestion.

Now we have a visual pattern of how congestion looks like. What we are still missing is a model for congestion. This will be introduced in the next section.

In summary, almost all sensors in Darmstadt show one of these two patterns. The few exceptions will be discussed later. We are going to refer to these patterns as “congestion-affected” and “non-congestion-affected” sensor. The classifier will assign one of these two classes based on the following mathematical model.

2.2 Modeling congestion

Given this input data and the abovementioned observations, how can we mathematically describe the characteristics of congestion? Describe them in such a way that classification is possible. In the following sections, we first try to

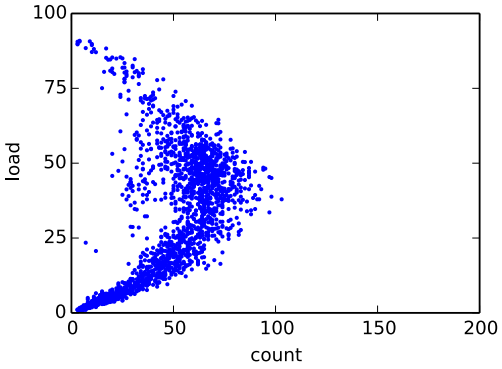

Figure 2.4: A sensor that is congestion-affected (curved pattern)

capture the expected behavior. How would a given sensor behave, if the corresponding street or lane would have infinite capacity? Then we will introduce an error function to quantify the deviation from this expected behavior.

2.2.1 Fitting a Linear Regression

In order to calculate the deviation from the expected behavior of a sensor, we first have to describe the expected behavior. Given infinite capacity, we expect that *count* correlates linearly with *load*. Hence, the most intuitive approach is fitting a straight line to the point cloud using linear regression.

A straight line is given by the following equation:

$$y = \alpha x + \beta \tag{2.1}$$

We assume that the *load* is zero, if no car is present. Hence, the y-intercept is $\beta = 0$.

This is accurate for sensors that are not affected by congestion, since the point cloud is approximately linear anyway. However, this approach cannot be used directly on congestion-affected sensors. Figure 2.5 plots the linear regression for the above mentioned congestion-affected sensor. Indeed, this does not capture the characteristics of this sensors. What we actually want is a regression line that approximates the linear behavior found before hitting the capacity.

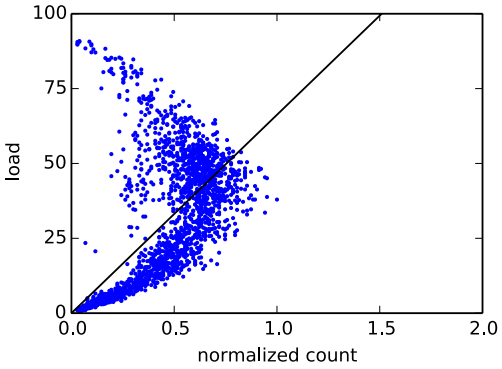

Figure 2.5: Linear regression for a congestion-affected sensor

Since congestion-affected sensors behave linearly when close to origin, we can approximate the expected behavior by applying linear regression only on values close to the origin. We tried different cut-off points and using the bottom 70% values for a single sensor led to stable results. This way we can produce usable regression lines for both kinds of sensors using the same method. Figure 2.6 shows a regression line that was fitted using only the bottom 70% values (blue colored points). Note that the points near the origin have a high density, which means that free flowing traffic is more likely than dense traffic.

What this regression line really does is describing the expected linear relation between *count* and *load* of any given sensor, if the capacity would be infinite. Hence, deviation from this expected behavior implies congestion. Deviation is best described using error functions and the following section will shortly introduce the applied error function.

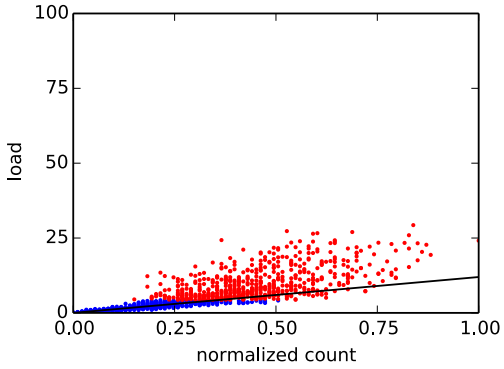

Figure 2.6: Regression line on 70% of the data

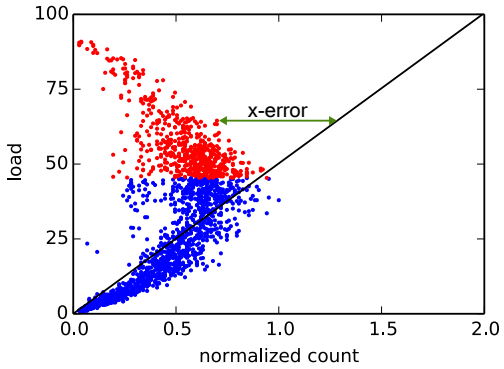

Figure 2.7: Regression line and x-error for a single value of a congestion-affected sensor

2.2.2 X-Error

To describe how far away from linear behavior a sensor actually is we need a measure that is able to reflect this. The error for a given sensor measurement $P = (count, load)$ and a linear regression function $f(x) = ax$ is defined as given in Equation 2.2, where f^{-1} is the inverse function of f.

$$E(P) = \begin{cases} f^{-1}(P_{load}) - P_{count} & P_{count} \leq f^{-1}(P_{load}) \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The error function is based on the following observation: Given an infinite capacity (the expected behavior function f) we can calculate an expected number of cars for a given *load*. If the number of cars is lower the traffic flow performs worse than expected and a positive error is calculated. If the number of cars is higher than expected the traffic flow is even better than expected and the error is set to 0.

This error is calculated using the mean squared error (MSE) over all x-errors for the remaining 30.0% not used for calculating the linear regression. The idea is to quantify how good the linear regression calculated on the bottom 70% fits the top 30%.

$$MSE(P) = \frac{1}{|P|} \sum_{p \in P} E(p)^2 \tag{2.3}$$

When we compare Figure 2.6 and Figure 2.7 we can see that congestion-affected sensors will have a higher x-error than non-congestion-affected ones. Hence it seems that the x-error is a good indicator for congestion-affected sensors.

2.2.3 Possible Shortcomings of the X-Error

Using a cut off point of 70% works well for most sensors. However, it does not work for some sensors. Especially, if alot of measurements are densely clustered around the origin. This can lead to a suboptimal x-error for that sensor that does no longer reflect the desired property, namely to detect congestion-affected sensors.

This could be mitigated by, for example, using dynamic cut off points based on density. However, given the variety of different sensors and the quality of the employed method (cf., Section 4), we postponed this for future work.

3 Sensor Classification

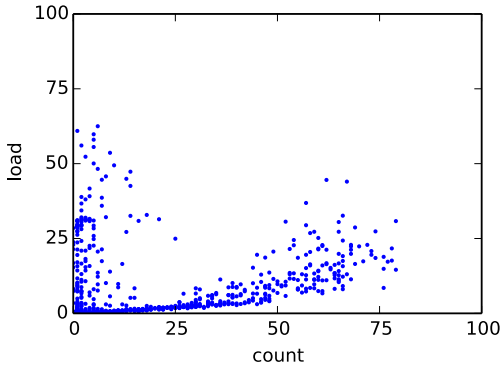

Figure 3.1: Example for a sensor with the *uncertain* label

Last section we introduced the x-error, which is designed as an indicator for congestion-affected sensors. A sensor is more likely to be congestion-affected if the x-error is high.

Given a labeled training data set, we can sort all sensors by their x-error and find the optimal threshold to separate the two class (congestion-affected and non-congestion-affected). This would yield an optimal threshold for our training data set. This is not desirable since it would overfit. Therefore, the calculation of the threshold must be independent from our training set to be still able to generalize to new previously unseen sensors.

Hence, a more general process is introduce in this section. First, we describe how we obtained a labeled data set. Second, a approach is introduced yielding an upper and lower threshold bound. Third, given the two threshold the process for interpolating the final threshold is given.

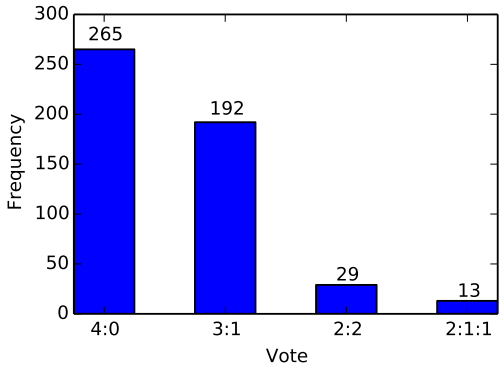

Figure 3.2: Statistics of the merging process

3.1 Manual Labeling

To evaluate any threshold, we must generate a labeled training data set. These labels are not available, hence, we decided to conduct a manual labeling. The manual labeling was performed by all three authors independently. The main author labeled twice. Thus, each sensor received four labels total. Given the visual pattern of each sensors one of three labels could be assigned: (i) *Congestion-affected*, (ii) *non-congestion-affected*, and (iii) *uncertain*. Some sensors are uncertain, due to different possible errors (e.g., physical relocation of sensors etc.). An example for such a sensor can be seen in Figure 3.1, which was unanimously labeled uncertain. The sensor seems to have two linear patterns instead of a single one. Uncertain sensors are filtered out, leaving us with only two classes to distinguish.

Table 3.1: Example of the process for determining the top threshold

Win.	Sensor	Cong.
	1	Yes
	2	Yes
*	3	Yes
*	4	Yes
*	5	No
*	6	Yes
*	**7**	**No**
	8	No
	9	Yes

The actual training set was then generated by assigning the label where the majority of the labelers agreed. In case of a collision, i.e., the ratio of labels was 2:2, the label of the most experienced labeler was used, which in this case is the person that labeled the data twice, where then the second label was used.

Figure 3.2 shows the result of the merging process and the frequency of the different possible ratios. Collisions happened only in 29 of 499 cases, whereas most of the sensors were unanimously labeled with a ratio of 4:0.

This led to 252 congestion-affected sensors, 177 non-congestion-affected ones, and 70 sensors could not be assigned one of the groups and therefore are uncertain. Please note that we removed these sensors as they contain no information for our purposes.

3.2 Top and Bottom Thresholding

When we sort all sensors by their corresponding x-error in descending order it is expected that congestion-affected sensors are more likely to be at the top of the ranking. The non-congestion-affected sensors should be at the bottom and there should be a part in the middle where both are mixed up.

The threshold needs to be set such that it splits the sensors optimally into the two classes, i.e., the number of misclassified sensors should be minimal. A possible approach to find such a threshold is what we call top and bottom thresholds. It works as follows: We iterate over every sensor in the list and remember the last ten labels. Starting at the top of the ranking, we expect these labels to be *congestion-affected* first. Whenever the number of *non-congestion-affected* labels, within the window of ten, becomes greater than four we stop the process and remember the last x-error we saw[1]. This will be the top threshold t_{top}. The process is repeated from the bottom (with label expectations reversed) and the resulting threshold is called t_{bottom}. The hypothesis is that the optimal threshold is located somewhere in the middle between the top and bottom threshold.

Table 3.1 shows a simplified example of this process for the top threshold, where the window size is five and the process stops when the number of wrong labels exceeds one. In this example the process would stop at sensor 7 (shown in bold) and would return the corresponding x-error.

Please observe that these two threshold are both biased. The top threshold will have a high precision on congestion-affected sensors but a lower precision on non-congestion-affected sensors. The bottom threshold consequently behaves the other way around. It will have a high precision for non-congestion-affected sensors but a low precision for non-congestion-affected sensors. The following section will deal with finding the optimal threshold between t_{top} and t_{bottom}.

[1] Note that we experimented with different values for the window and the maximum error, but a window size of ten and a maximum error of four gave the best results.

3.3 Interpolated Thresholds

We expect the optimal threshold t, the threshold yielding the highest precision for both classes, to lie somewhere between t_{top} and t_{bottom}. To find the optimal threshold we linearly distributed thresholds in between t_{top} and t_{bottom} (cf. Equation 3.1).

$$step = \frac{1}{n+1}\left(t_{top} - t_{bottom}\right)$$

$$\begin{aligned} t_1 &= t_{top} + step \\ t_2 &= t_1 + step \\ &\dots \\ t_n &= t_{n-1} + step \end{aligned} \tag{3.1}$$

Each threshold is evaluated independently to find the overall best threshold relative to the top and bottom threshold. This relative threshold can then be applied to any given set of sensors, without overfitting to the specific set.

4 Evaluation

The process of finding the optimal threshold is now evaluated using ten-fold cross-validation for three and eight interpolated thresholds together with the corresponding t_{top} and t_{bottom} thresholds. Each cross-validation is repeated 100 times; each time with a different random split of the training set. We then calculated the accuracy for each of the thresholds as $acc = {}^{corr}/_{incorr}$.The scheme just labels every sensor with an x-error greater than the threshold as congestion-affected and non-congestion-affected otherwise.

Table 4.1: The results for using five different thresholds, averaged over 1000 iterations

Threshold	Accuracy	Frequency
t_{top}	87.0505%	0
$\mathbf{t_1}$	**89.9960%**	**979**
t_2	88.9220%	20
t_3	88.6095%	1
t_{bottom}	87.9057%	0

This led to the results from Table 4.1 where the best performing threshold is depicted in bold. We also counted how often a specific threshold had the highest accuracy which can be seen in the *frequency* column of the two tables. One would naively expect the threshold t_2 which lies exactly in the middle of t_{top} and t_{bottom} to perform the best. Interestingly this is not the case and instead the threshold t_1 wins. t_1 can be written as

$$t_1 = t_{top} + 0.25\left(t_{top} - t_{bottom}\right).$$

We then increased the amount of interpolated thresholds to eight and repeated the process, which led to the results from Table 4.2.

Table 4.2: The results for using ten different thresholds, averaged over 1000 iterations

Threshold	Accuracy	Frequency
t_{top}	87.0505%	0
t_1	89.1607%	42
t_2	89.5962%	390
$\mathbf{t_3}$	**89.6359%**	**455**
t_4	89.2466%	75
t_5	89.1197%	36
t_6	88.8149%	0
t_7	88.3402%	0
t_8	88.6779%	2
t_{bottom}	87.9057%	0

You can see that this time t_2 and t_3 perform very similar, with t_3 taking the lead. This means that the optimal threshold lies somewhere in between t_2 and t_3, which is close to the t_1 for three interpolated thresholds. Sticking to the simpler approach with less interpolated thresholds we expect the optimal threshold to be

$$t = t_1 = t_{top} + 0.25\left(t_{top} - t_{bottom}\right).$$

5 Related Work

The problem of traffic congestion is very frequently researched. Many ideas have been published of which only a few can be mentioned here.

In his Master's thesis [3] Jitesh Pati Tripathi used the idea of tracking the speed of a probe vehicle through GPS and then using this information determine the current traffic situation. He used this method to find what he calls hot spots, which are places where the traffic gets very congested. This is similar to the congestion-affected roads in our work.

Rebecca Ong et al. used a similar approach in [2], where they also used the GPS data of vehicles to find flock patterns, where vehicles are moving very slow, to detect potential traffic jams. Also using GPS as their input data were Ke Zhang and Guangtao Xue in [5], but they introduced a new method of representing the network of intersections which they call spatio-temporal original-destination matrices. These matrices are calculated from GPS data sets and can be used to find similarly behaving road segments in terms of traffic congestion.

A different approach was shown by Milos Milojevic and Veselin Rakocevic in [1], where they use a wireless connection between vehicles to make them self-aware of their surrounding. The vehicles interchange their information about traffic speed to calculate the current traffic situation.

Chenqi Wang and Hsin-Mu Tsai [4] also used a vehicle to vehicle network in their approach. The advantage of their approach is that only very few vehicles have to be equipped with their system in order to reach good results. Here equipped vehicles communicate with a local backend server to measure the current traffic situation. Using this approach they were able to achieve a detection accuracy of 88.94%.

As shown floating car data is an interesting direction for detecting congestion. They do offer data where no loop sensors are available. However, the accuracy and density of loop sensors in urban deployments is still unmatched. We see great potential in both directions and, even more so, hybrid approaches.

6 Conclusion and Future Work

The automatic detection of congestion-affected sensors is possible. Even using only this simple classification approach yields 90% accuracy already.

In the future we would like to take more features into account, e.g., grouping sensors together that are all installed on the same road. We would also like to work on the fitting of the regression line as mentioned earlier. The selection of values for the fitting could be based on the density of the values, which would lead to better fitting of the linear part. And this is just the first step on what can be done with inductive loop data in the future. Given the data the next step is to assign different traffic levels (light, medium, dense, congestion) per sensor. And then detect the current traffic situation for given *count* and *load* values in real-time.

In summary, this work is a first step into the direction of automatic detection of congestion-affected roads, based on inductive loop data.

Bibliography

[1] M. Milojevic and V. Rakocevic. *Distributed Vehicular Traffic Congestion Detection Algorithm for Urban Environments*, 2013. 13

[2] R. Ong, F. Pinelli, R. Trasarti, M. Nanni, C. Renso, S. Rinzivillo, and F. Giannotti. Traffic jams detection using flock mining. In *Machine Learning and Knowledge Discovery in Databases*, pages 650–653. Springer, 2011. 3, 13

[3] J. P. Tripathi. Algorithm for detection of hot spots of traffic through analysis of gps data. Master's thesis, Thapar University, 2010. 3, 13

[4] C. Wang and H.-M. Tsai. Detecting urban traffic congestion with single vehicle. In *Connected Vehicles and Expo (ICCVE), 2013 International Conference on*, pages 233–240, Dec 2013. 13

[5] K. Zhang and G. Xue. *A Real-Time Urban Traffic Detection Algorithm Based on Spatio-Temporal OD Matrix in Vehicular Sensor Network*, 2010. 13