Project Number 288094

eCOMPASS

eCO-friendly urban Multi-modal route PlANning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 043

Use of Density-based Cluster Analysis and Classification Techniques for Traffic Congestion Prediction and Visualization

T. Diamantopoulos, D. Kehagias, F. Konig, D. Tzovaras

September 2013

Project Number 288094

eCOMPASS

eCO-friendly urban Multi-modal route PlANning Services for mobile uSers

STREP
Funded by EC, INFSO-G4(ICT for Transport) under FP7

eCOMPASS – TR – 043

Use of Density-based Cluster Analysis and Classification Techniques for Traffic Congestion Prediction and Visualization

T. Diamantopoulos, D. Kehagias, F. Konig, D. Tzovaras

September 2013

Use of Density-based Cluster Analysis and Classification Techniques for Traffic Congestion Prediction and Visualization

Themistoklis Diamantopoulos*, Dionysios Kehagias*, Felix G. König† and Dimitrios Tzovaras*
* Centre for Research & Technology Hellas,
Information Technologies Institute,
P.O. Box 60361, 57001 Thermi-Thessaloniki, Greece
Email: {thdiaman, diok, tzovaras}@iti.gr
† TomTom International BV, An den Treptowers 1, 12435 Berlin, Germany
Email: Felix.Koenig@tomtom.com

***Abstract*—The field of Intelligent Transportation Systems has raised increasing interest due to it socio-economic impact. The aim of this work concerns developing efficient techniques for traffic congestion prediction and visualization. We have designed a simple, yet effective and scalable model to handle sparse data from GPS observations and reduce the problem of congestion in a binary classification problem (jam, non-jam). An attempt to generalize the problem is performed by exploring the impact of discriminative versus generative classifiers when employed to produce results in a 30-minute interval ahead of present time. In addition, we present a novel traffic jam visualization approach based on cluster analysis that identifies dense congestion areas. The results of our analysis provide insight for further research.**

I. Introduction

Nowadays, there has been increasing interest in the field of Intelligent Transportation Systems. The socio-economic impact of traffic congestion is present in various real-life problems. The aim of this paper lies in forming the problem of predicting traffic jams, as well as visualizing congestion in a meaningful comprehensive way.

Congestion prediction is a challenging task that is broadly studied by several researchers [1]–[12]. In general, the congestion prediction techniques face the following problems:

- Data Acquisition and Manipulation: acquiring data in a particular form (e.g. loop detectors or probes) and creating certain metrics, usually traffic flow/volume, occupancy or even mean speed or travel time per link.
- Congestion Modeling: using the aforementioned metrics to define the jam and non-jam states of a road, done in a heuristic manner.
- Congestion Prediction: predicting jams, accomplished using classification algorithms, although regression and heuristics are also present in current literature.

Apart from the above problems, in this paper the problem of traffic congestion visualization is taken into account.

The rest of this paper is organized as follows. Section II of this paper summarizes certain state-of-the-art techniques of current literature, while denoting their approach on the problems described here. The dataset used and the provided metrics are described in Section III, while Section IV describes the model used to form the congestion prediction problem. Sections V and VI present the proposed approaches for the congestion prediction and congestion visualization problems respectively. The results of our evaluation are presented in Section VII, while Section VIII summarizes the main conclusions of this paper and suggests future directions.

II. State-of-the-art Congestion Prediction

Several traffic congestion prediction techniques, such as the approach by S. Yang [1] formulate the problem as a binary classification task. The author draws data from 4000 loop detectors, that each provide with the *traffic volume*, i.e. the number of vehicles passing through the detector per time unit. Upon defining an upper and a lower threshold, the state of a road is defined as *jammed* if its volume is above the upper threshold and *non-jammed* if it is below the lower threshold. Thus, the dataset is split into two sets, one having jammed and one having non-jammed roads, and those sets may be used to train a binary classifier. However, using data from all sensors to predict the class of a road's state is not possible in terms of dimensionality, thus the author also applies a p-test to identify which features actually affect the state of the road [1]. Upon identifying the most important features for each road (i.e. the ones with the highest p-score), the classification is performed assuming Gaussian distributions over the two datasets, so that the final probability for the jams state of road j at time t is given by the following equation:

$$S_{t,\tau}^{j} = \prod_{i=1}^{I} \frac{\Pr\{v_{t-\tau}^{i} \in Gaussian\{\mu_{ij}, \sigma_{ij}\}\}}{\Pr\{v_{t-\tau}^{i} \in Gaussian\{\bar{\mu}_{ij}, \bar{\sigma}_{ij}\}\}} \quad (1)$$

where the volume values for time $t-\tau$ are of course known and I is the total number of sensors. Finally, the author uses mean precision to evaluate the results and performs analysis to determine the optimal number of features required. The approach is quite effective since Gaussian models are generally strong for low number of features. However, they seem to struggle when there are many features. Thus, applying the algorithm to a scenario with speed probes would probably be both ineffective and inefficient.

A similar, yet less scalable approach is presented by G. Huisken and M. V. Maarseveen in [3], upon prior research in [2]. The authors collect data using 35 induction loops on the motorway A10 of Amsterdam. The detectors are "on" when a vehicle passes by them and they are "off" when no vehicle passes. The number of vehicles passing the detector per time unit (*volume*) as well as the percentage that the detector is "on" (*occupancy*) are known. In addition, the average and standard deviation of speed in a road segment is calculated using the respective series of loop detectors. Finally, the authors claim having an oracle indicating the presence of congestion. The volume, mean speed, occupancy and standard deviation of speed can be given as features to any classifier, whereas the output class comprises of the binary congestion indicators for the ring road, which were totally 6. The authors test different classifiers, including multi-linear regression, an ARMA model, a heuristic Fuzzy Logic classifier, and three neural network implementations. Although their line of work is interesting, the number of features is clearly not indicative of an urban scenario. The method may be effective for ring roads, however using it in a large road network is prohibitive.

A rather different approach by W. Labeeuw et al. [4] concerns predicting congestion in different points of a ring road, using a generated speed probe dataset. The authors initially address the problem as a regression problem, attempting to predict future velocity values using Bayesian regression over Gaussian processes. Due to performance issues, the problem is once again reduced to classification. Three different cases are determined for each road as slowdown, congested, and normal. Velocity data from each sensor and its local sensor are used to formed the features and two classifiers are tested: an SVM and a C4.5 tree classifier. The results indicate that the C4.5 classifier is quite faster while it has better precision concerning the $Congested$ and $Slowdown$ sets. Finally, the paper hints applying the algorithms in a distributed multi-agent manner. Although the paper indicates that classification is actually a viable option when considering jams, the dataset is small and the algorithms do not seem to scale. As previously, since the dataset is generated for a ring road, the approach is rather problem-specific, while noise and sparse data are not taken into account.

By contrast with the aforementioned approaches, the *IEEE ICDM Contest: TomTom Traffic Prediction for Intelligent GPS Navigation* [13] that took place in 2010 provided with interesting insight concerning the raw form of the data and the noise it may have. Despite relying on simulated data for the city of Warsaw, the scenarios of the contest are quite realistic. The 2nd task of the contest, described in [13], refers to jams. The dataset consisted of sequences of road segments. In the training set, the first 5 road segments reflected roads that are closed due e.g. to roadwork, while the others are road segments that where jammed during a 20-minute interval. The goal was to create an algorithm that could identify the jams in the next 40-minute interval (of course given for the training set), as well as the length of the output road sequence (number of jams). The output road sequence should be ordered according to jam appearance (from earlier to later).

The winning algorithm of the contest by L. Romaszko was a modified version of the k-Nearest Neighbors (kNN) algorithm. The algorithm compares sequences of the training set with the respective ones in the test set. Intuitively, assuming the most similar training sequences to a specific testing sequence, the road jammed in training sequences are probably jammed also for the testing sequence. The algorithm also outputs jam probability for each road, considering its position in the training sequences. The average length of the k nearest training sequences determines the length of the test sequence.

The runner-ups of the competition, J. He et al. [5], created an ensemble-based method which combines the scores of different base predictors. Two types of predictors were created: the geographic propagation predictors and the nearest neighbor predictors. The former track the flow of a jam based on the connectivity of the road segments while the latter are based on comparing the training sequences with the testing sequence at hand. Each of the tested predictors assigns every jammed road with a score, which denotes its ranking, based on the distance given by kNN as well as several heuristic parameters. The scores of all predictors are combined in a linear fashion to form the final sequences. Concerning the length of each testing sequence, the authors use a simple average over the respective nearest training sequences. Both this and the winning solution were powerful. yet relatively inefficient since kNN has to store data instead of the model and its execution time is rather slow. Although both approaches are quite interesting, they are largely specific to the dataset of the contest.

Finally, current literature in congestion prediction includes several other lines of work, such as the one proposed by G. Marfia et al. [6]–[9]. The authors suggest a distributed *Advanced Traveler Information System (ATIS)*, where cars send in their travel times upon traversing each road. Upon defining suitable thresholds, the authors employ univariate heuristics to predict future states. Both this line of work and the one by Y. Ando, O. Masutani et al. [10]–[12], where cars emit pheromone proportional to the traffic, are mainly directed towards distributed systems, thus deviate from the scope of this paper.

III. Data Acquisition

We used road network and traffic data for the city of Berlin, in order to evaluate our methods. Modeling traffic jams implies defining their nature with respect to the data given, which in our case consists of instantaneous vehicle speeds in different moments as well as free flow speeds, reported when road traffic is quite sparse.

IV. Modeling Traffic Congestion

According to current literature (see Section II), modeling traffic congestion usually comes down to defining appropriate heuristics with respect to the metrics provided. As described in Section III, the metrics used in this paper are instantaneous speed measurements for the roads of the network in different moments in time. The model described in this section is not only effective and efficient but also quite noise tolerant. This is crucial since GPS observation data is sparse, thus a one-by-one comparison with a system using loop detectors is impossible. Thus, the speed probes for each road are not stored. Instead, certain metrics such as their arithmetic average, their harmonic average, their standard deviation, and the number of samples are calculated for 5-minute time intervals. As a result, noise tolerance and scalability are handled satisfactorily.

Let $\mu_r(t)$ be the mean speed for road r at time interval t and $\sigma_r(t)$ be its standard deviation, we define two thresholds μ_T and σ_T respectively. Thus, the presence or not of a jam in road r at time t is determined using the following equation:

$$S_r(t) = \begin{cases} Jam, \text{ if } \dfrac{100 \cdot \mu_r(t)}{FF_r} \leq \mu_T \text{ and } \sigma_r(t) \leq \sigma_T \\ NonJam, \text{ otherwise} \end{cases} \quad (2)$$

where FF_r is the free flow speed of road r. Thus, the mean speed threshold μ_T is defined as a percentage of the free flow speed, while the standard deviation threshold σ_T in the order of road speed.

Intuitively, equation (2) provides quite realistic distinction of jams. Assuming values 60 and 40 for μ_T and σ_T respectively, any road is considered jammed if its current average speed is 60% of its free flow speed as long as most speed probes are close to this speed within an (approximate) $[-40, +40]$ range area. Hence, although simplistic, the model is effective and realistic. Finally, note that all of the above computations are performed in a real-time manner, e.g. running mean and running standard deviation are calculated using Welford's method [14].

V. Traffic Congestion Prediction

Following the congestion modeling procedure of Section IV, the problem of congestion prediction is reduced in a binary classification problem, with the class labels (jam, non-jam). The selection of features to be used in the classifier and the proposed approaches to the classification problem are discussed in the following subsections.

A. Feature Selection

Proper feature selection is vital since the performance of the classifier is highly dependent on it. Intuitively, multivariate analysis should yield quite meaningful results since the state of a road depends on its local neighborhood and possibly on several other important roads all over the road network. As proved also in [15], the effect of network-wide metrics in forecasting is quite significant. Intuitively, this is rather expected since e.g. a congestion on a ring road could affect several major roads of the city even though they may not be in its local neighborhood.

Thus, the problem is formulated as predicting congestion for a road, given the time series of speed values for its neighboring roads as well as time series drawn from global roads of the network. Forecasting congestion, or traffic in general, comes down to applying multivariate analysis using both global and local features. However, real networks, such as the one described in Section III, may contain thousands of roads, and furthermore some roads may have sparse or noisy data, or simply data that has no impact on traffic. As a result, an important step of global multivariate traffic prediction approaches is determining the roads that affect the traffic of the road of interest. Current literature is directed towards the use of correlation metrics such as *Coefficient of Determination (CoD)* [15]–[17], or transformation procedures, such as *Principal Component Analysis (PCA)* [18].

In terms of this paper, three series are constructed for each road corresponding to three metrics for every interval: the number of samples per interval, the arithmetic mean and the standard deviation of speed. Upon performing PCA over these three series, the most important global features for these three dimensions are isolated. Hence, these are the first set of features given to the classifier.

Apart from global features, the classifier is also given local ones, containing arithmetic means of the time series of the road to be predicted as well as its neighboring roads. Let $n_1, n_2, \ldots, n_s$ be the neighboring roads of road r, for each time interval the number of local features is equal to the number of neighboring roads plus one for the road itself $(s+1)$. The final feature set is shown in Figure 1.

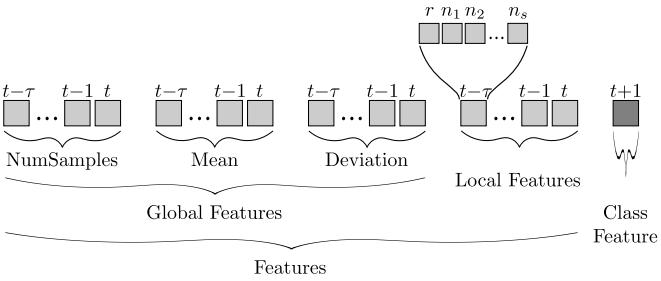


Fig. 1. Feature set used to classify road state at time $t+1$ to *Jam* or *NonJam*.

Thus, Figure 1 actually illustrates the feature selection procedure required by any classifier. As mentioned above, the three global feature groups correspond to different metrics, and were derived upon performing PCA. Local features contain only the arithmetic mean of the road and its neighboring roads for different time intervals.

Finally, the scenario can be defined as binary classification with scalar features. Applying any classifier is straightforward. The training set consists of data in the form depicted in Figure 1, while the testing set is in the same form, excluding the value of the class feature which has to be determined.

B. Classification

Classification is defined as the problem of determining the class of a new observation, given a set of observations for which the class is known (i.e. the training set). Classifiers are divided into two different categories: *generative* and *discriminative* classifiers. A generative classifier attempts to model the probability distribution of the data in order to classify a new observation. By contrast, discriminative classifiers classify observations without creating such a model. For instance, using a generative classifier to classify a text to a language would include learning every possible language model and classifying the text using the learned models. A generative classifier, on the other hand, would determine the discriminating elements among languages, and thus classify the text according to them.

The comparison of generative and discriminative classifiers is a broadly studied topic in current literature [19], [20]. In general, generative classifiers are known to perform better in small datasets since they approach their asymptotic error much faster. By contrast, since discriminative classifiers exhibit lower asymptotic error, they are better suited towards large diverse datasets. Since data of real-world classification scenarios are usually rather large, there is strong preference for discriminative classifiers not only because of their effectiveness but also

due to their efficiency, since computational power is consumed for classifying instead of constructing a model. Generative classifiers, on the other hand, are preferable when data is sparse, since their models exhibit better noise tolerance.

In terms of the problem at hand in this paper, both generative and discriminative techniques can be expected to perform satisfactorily, since the former would tolerate better the sparse and noisy data, whereas the latter would be effective concerning the large size of the dataset. Consequently, we implemented four techniques, two of each kind, in order to investigate their effectiveness on the problem. These techniques are analyzed in the following paragraphs.

1) Gaussian Bayes: Probably the most representative generative classifier is *Naïve Bayes*. The classifier is a generalization of the Bayes theorem for modeling beliefs and it is called naïve since conditional independence between the features is assumed. Let the features be $x_1, x_2, \ldots, x_n$, a probability is derived for each possible value y_i of the class attribute[1] y according to the following equation:

$$P(y_i|x_1, x_2, \ldots, x_n) = \frac{\prod_k \{P(x_k|y_i)\} \cdot P(y_i)}{\sum_j \left\{ \prod_k \{P(x_k|y_j)\} \cdot P(y_j) \right\}} \quad (3)$$

Given the values of all attributes, $P(x_k|y_i)$ is the probability of an attribute having value x_k given that the class is y_i. Thus, the nominator of the above equation is defined as the product of those probabilities for all attributes, multiplied by $P(y_i)$ which is the prior probability of the class being y_i. The denominator serves only as a normalizer so that the probabilities of all class attribute values sum up to 1.

Since the Naïve Bayes classifier is used when the attributes of the dataset are nominal, using it in a scalar-attributes scenario requires making an assumption about the distribution of the data. In other words, the problem is reduced to approximating the $P(x_k|y_i)$ term of equation (3). A common technique involves assuming the data is distributed according to a Gaussian distribution. Thus, the probability of an attribute value when the class is y_i can be computed as:

$$P(x_k|y_i) = \frac{1}{\sqrt{2\pi\sigma_{y_i}^2}} \cdot e^{-\frac{(y_i-\mu_{y_i})^2}{2\sigma_{y_i}^2}} \quad (4)$$

where μ_{y_i} and σ_{y_i} are the distribution's mean and standard deviation respectively. Finally, using equation (3) the probability for each class value is computed and the class of an instance may be selected either as the one with the maximum probability or even in a probabilistic way with respect to the class attribute values.

2) Gaussian Mixture Models: As shown in the previous technique, a generative classifier can be generally based on defining models that fit the data and deriving probabilities for new observations. Although assuming the data follows a Gaussian distribution may seem naïve, it can be noted that using a mixture of Gaussians should satisfactorily model most datasets. Given d-dimensional distributions of the form:

$$P_G(\vec{x}) = \frac{1}{(2\pi)^{d/2} \cdot \sqrt{|\vec{\sigma}|}} \cdot e^{-\frac{(\vec{x}-\vec{\mu})^T \cdot \vec{\sigma}^{-1} \cdot (\vec{x}-\vec{\mu})}{2}} \quad (5)$$

[1]The terms "feature" and "attribute" are used interchangeably.

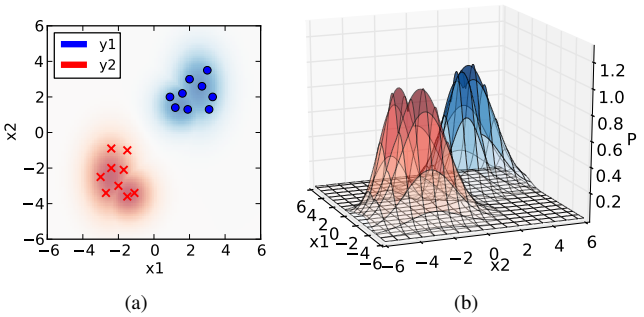


Fig. 2. A classification example with 2 sample dimensions (x_1, x_2) using Gaussian Mixture Models, shown in (a) two and (b) three dimensions. The blue Gaussian Mixture Model has two distributions starting at $(1.5, 1.5)$ and $(2.5, 2.5)$, while their mean and standard deviation vectors are [1 0.5], [1 1] and [1 1], [1 1.5] respectively. The red model has two distributions at $(-2.5, -2.5)$ and $(-1.5, -3.5)$, with means and deviations [2 0.5], [1 1.5] and [1 1], [1 1] respectively.

where $\vec{\mu}$ is the mean and $\vec{\sigma}$ is the covariance matrix of the distribution, the probability in a mixture of K Gaussians is:

$$P(\vec{x}) = \sum_{j=1}^{K} w_j \cdot P_{G_j}(\vec{x}) \quad (6)$$

where w_j is the prior probability of the j-th Gaussian. A modeling example using *Gaussian Mixture Models (GMMs)* is shown in Figure 2 for two dimensions x_1 and x_2 and a class feature y. Assuming the red $\times$'s correspond to a dataset, the mixture of Gaussians corresponding to sample y_2 seems to successfully cover them. The model is probabilistic, thus any new observation could be assigned to model y_2 according to the probability given by equation (6). Thus, as a result of the above analysis, the problem is reduced to creating the models for the two classes of data, y_1 and y_2. Upon having constructed such models, the class of a new instance may be selected using the probabilities derived from the two models, once again either by selecting the maximum probability or simply by returning a probabilistic result.

Given the sample $\vec{x}$, equations (5) and (6) produce the following optimization problem:

$$\theta^* = \arg\max_{\theta} \log\left(P(\vec{x}|\theta)\right) = \arg\max_{\theta} \prod_{j=1}^{K} P_{G_j}(\vec{x}|\theta) \quad (7)$$

where θ contains all the parameters of the model, i.e. the weights $w_1, w_2, \ldots, w_K$, the means $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_K$ and the standard deviations $\vec{\sigma}_1, \vec{\sigma}_2, \ldots, \vec{\sigma}_K$. The optimal θ^* is achieved when the probability is maximized, so that it covers the sample as much as possible. A popular approach of maximizing θ is to use the *Expectation-Maximization (EM)* algorithm.

The EM algorithm is a two step procedure. The expectation step calculates the probability of sample $\vec{x_i}$ to belong to mixture k using the available parameters:

$$P_{G_k}(\vec{x_i}|\theta) = \frac{w_k \cdot P_{G_k}(\vec{x_i}|\vec{\mu}_k, \vec{\sigma}_k)}{\sum_{j=1}^{K} w_j \cdot P_{G_j}(\vec{x_i}|\vec{\mu}_j, \vec{\sigma}_j)} \quad (8)$$

The maximization step involves estimating the mixture parameters using the computed probabilities. Let N be the total

number of samples, the weight w_k is approximated as follows:

$$w_k = \frac{\sum_{i=1}^{N} P_{G_k}(\vec{x_i}|\theta)}{N} \tag{9}$$

while the mean $\vec{\mu}_k$ and the standard deviation $\vec{\sigma}_k$ are:

$$\vec{\mu}_k = \frac{\sum_{i=1}^{N} P_{G_k}(\vec{x_i}|\theta) \cdot \vec{x_i}}{\sum_{i=1}^{N} P_{G_k}(\vec{x_i}|\theta)} \tag{10}$$

$$\vec{\sigma}_k = \frac{\sum_{i=1}^{N} P_{G_k}(\vec{x_i}|\theta) \cdot (\vec{x_i} - \vec{\mu}_k) \cdot (\vec{x_i} - \vec{\mu}_k)^T}{\sum_{i=1}^{N} P_{G_k}(\vec{x_i}|\theta)} \tag{11}$$

Continuously iterating over the two steps results in a usually strong model, as long as the number of mixture models and the number of iterations are carefully chosen to avoid overfitting the algorithm. Finally, note that any step can be chosen as the first one. When, however, the algorithm starts with the maximization step, it is necessary to provide the algorithm with a prior estimate over the sample (i.e. the initial values for $P_{G_k}(\vec{x_i}|\theta)$, for every k).

3) k-Nearest Neighbors: The first discriminative classifier that we implemented is the kNN classifier. The classifier is initially given a training set in the form of features $x_{t1}, x_{t2}, \ldots, x_{tn}$ and the (known) class feature y_t for all different time moments t of the training set. Given a sample $x_1, x_2, \ldots, x_n$ to classify, the algorithm initially finds the distances between the sample and each of the known training samples, in our case using the Euclidian distance metric, although other distance metrics may be used accordingly. The k "nearest" samples are found and the output y is determined using an average metric of their respective outputs (y_t for all t that belong to the k nearest samples). In our work, the output of the classifier is given by the arithmetic mean, and is later given the value 0 or 1 with respect to a given threshold.

4) Support Vector Machines: Support Vector Machines (SVMs) have been widely used in literature for various classification tasks. The main idea is to construct a hyperplane that sets apart the classes of a sample. A classification example is shown in Figure 3.

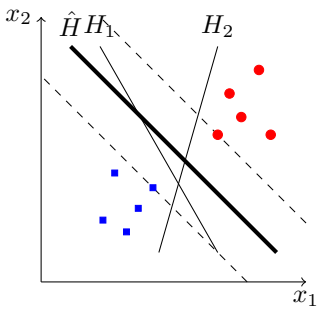


Fig. 3. A Support Vector Machine, separating blue (■) from red (●) data points. H_1 and H_2 are two valid hyperplanes, and $\hat{H}$ is the optimal hyperplane (dashed lines denote maximum distances).

The example of Figure 3 concerns a two-dimensional space (with dimensions x_1 and x_2), i.e. the data instances are classified according to two attributes. For two dimensions, the hyperplanes are reduced to single lines. Although there may be various hyperplanes that separate a dataset, SVMs attempt to approximate the optimal hyperplane, i.e. the hyperplane that has maximum distance from instances on both sides. Maximizing the margin comes down to solving the *Quadratic Programming (QP)* problem. Current state-of-the-art suggests solving it using *Sequential Minimal Optimization (SMO)*.

In classifying traffic jams, SVMs are initially given training data in the form of attributes $x_{t1}, x_{t2}, \ldots, x_{tn}$ and the class attribute y_t in order to construct the optimal hyperplane in a n-dimensional space. Classifying a new instance is rather simple; given the features $x_1, x_2, \ldots, x_n$, the SVM returns whether the instance is on the Jam or on the $NonJam$ side of the hyperplane, as well as its distance from the hyperplane which may be used as a quantitative metric of how discriminated is the instance. Upon cross-validating, we decided to use a *Radial basis function (RBF)* kernel with γ equal to 0.1, and the penalty multiplier C of the SVMs was set to 10.

VI. Traffic Congestion Visualization

Upon forecasting congestion, we present a novel traffic jam visualization approach that illustrates congestion areas. Intuitively, jams spread in an area-wise manner. If several roads of an area are congested, then a small *dense* area covering them should also be considered congested or at least under slow-down. However, jam indications in sparse areas may indicate noise in data or even insignificant micro-jams. Based on the aforementioned intuitive remarks, we perform cluster analysis using modifications of density-based clustering algorithms in order to identify dense congestion areas. Our algorithm, which is based on *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)* [21], is shown in Figure 4.

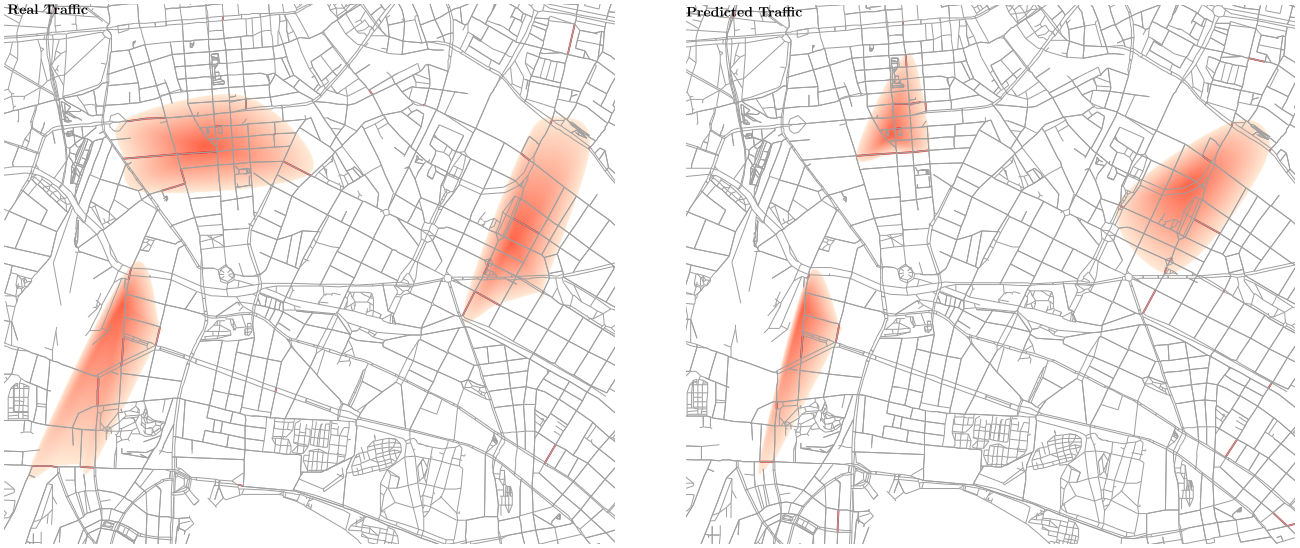
```
JAMS_DBSCAN(NeighOrder, MinRoads)
  clusters = new Cluster[]
  foreach road R
    mark R as visited
    if R is jammed
      NeighRoads = GetJammedNeighbors(R, o)
      if sizeof(NeighRoads) ≥ MinRoads
        C = new Cluster
        ExpandC(R, NeighRoads, C, o, MinRoads)
        clusters.add(C)

ExpandC(R, NeighRoads, C, o, MinRoads)
  C.add(R)
  foreach road R' in NeighRoads
    if R' is not visited
      mark R' as visited
      NeighRoads' = GetJammedNeighbors(R', o)
      if sizeof(NeighRoads') ≥ MinRoads
        NeighRoads = NeighRoads ∪ NeighRoads'
    if R' not in any cluster
      C.add(R')

GetJammedNeighbors(R, o)
   return all neighbor roads of R within
          maximum order o that are jammed
```

Fig. 4. The density-based clustering algorithm that accepts as input the minimum number of roads that form a cluster (`MinRoads`) ad the order of neighboring roads to be considered (o) and creates the `clusters`.

As shown in Figure 4, the algorithm initially defines a dynamic array of type `Cluster`. Iterating over all roads, the algorithm selects only the ones that are congested and finds also neighboring roads that are congested. If the number of congested



(a) (b)

Fig. 5. Visualization of traffic congestion using convex hulls that encircle clusters of jammed roads. Real jams are shown in (a) and predicted jams are shown in (b). The gradient level of each convex is proportional to the expected jam strength at the corresponding area, while its most strong area is at the centroid of the cluster.

roads in the nearby area is higher than `MinRoads`, then a new cluster is initialized and expanded using the function `ExpandC`. `ExpandC` initially adds the road at hand (`R`) to the cluster and then iterates over each of the neighboring roads (`NeighRoads`) and adds both the neighboring road (`R`$'$) and the latter's new neighbors (`NeighRoads`$'$) as long as they form a congested area with more than `MinRoads`.

Upon identifying dense congestion areas, the new problem lies in visualizing them comprehensively. At first, the clusters are actually buckets holding nearby roads. These roads have coordinates of the form $(x_1, y_1) \rightarrow (x_2, y_2)$, i.e. from their start to their end point (node). Each cluster contains the start and end points of all its roads, i.e. $(x_1^1, y_1^1), (x_2^1, y_2^1), (x_1^2, y_1^2), (x_2^2, y_2^2), \ldots$. The problem of finding the smallest convex shape that contains these points is known in literature as finding the *convex hull* of the points. We used *Graham Scan*, a simple algorithm named after R. L. Graham [22], to find the points of the convex hull. The algorithm initially finds the point with the lowest y coordinate and sorts all points according to the angle of the line formed between them and the lowest point with the x axis. Upon sorting, the algorithm iterates counter-clockwise over all points (i.e. in the way they were sorted) and discards any points that are inside the hull by measuring their angle. The algorithm finalizes when it returns to the initial point (more on this algorithm in [22]).

An example of the visualization of traffic jams is shown in Figure 5. The congested areas are shown with red color, while the gradience of the color is a metric of the concentration of congestion in the specific area. Intuitively, the strongly-colored areas are the ones close to the *centroid* of the cluster, which is defined as the arithmetic mean position of all the points of the cluster. Note that this is different from the centroid of the convex shape we created, since the former corresponds to the road coordinates that belong to the cluster, whereas the latter corresponds to the convex hull.

VII. Evaluation

This section presents the results of our evaluation for the algorithms used to predict traffic congestion, as well as certain qualitative comments for congestion visualization.

A. Prediction Evaluation

Concerning congestion prediction, the binary classifiers analyzed in subsection V-B are evaluated against data for the city of Berlin (see Section III). The data is split into two weeks, thus data for one day of the first week (e.g. Thursday of week 1) was used to train the algorithms and data from the subsequent day on the next week (e.g. Thursday of week 2) was used to test them. Note also that designing an appropriate evaluation framework is not trivial, since the distribution of the data is skewed. Congestion is expected to be rarer than free flow conditions. Thus, the classifiers require appropriate adaptation to the nature of the data.

As mentioned in subsection V-B, all classifiers can actually produce scalar outputs in a pre-specified interval. For example, Gaussian Bayes provides a probability of a road being congested, i.e. a value between 0 and 1, while SVMs provide not only the subspace (divide by the hyperplane) that the sample belongs but also its distance from the hyperplane. Converting this output to a nominal $(Jam, NonJam)$ value involves creating an appropriate threshold. Let θ be this threshold, θ shall receive decimal values from 0 to 1 with step 0.1.

Given a value of θ, a classifier provides nominal values $(Jam, NonJam)$ for all the roads of the dataset. According to *Information Retrieval*, evaluating the performance of the algorithm initially involves defining the four metrics:

- *True Positives (TP)*: the number of roads that are predicted to be congested and actually are congested
- *True Negatives (TN)*: the number of roads that are predicted to be on free flow and traffic flows freely

- *False Positives (FP)*: the number of roads that are predicted to be congested but traffic flows freely
- *False Negatives (FN)*: the number of roads that are predicted to be on free flow but actually are congested

Given these metrics, one may devise useful conclusions for the algorithms. Calculating the *precision* P and the *recall* R of an algorithm is trivial:

$$P = \frac{TP}{TP + FP} \tag{12}$$

$$R = \frac{TP}{TP + FN} \tag{13}$$

Intuitively, precision denotes the percentage of predicted jams that were correctly classified as jams, and recall denotes the percentage of jams that were successfully predicted.

Although precision and recall are useful metrics, in a skewed distribution scenario, such as the congestion prediction one, they may easily be deceiving. Achieving high recall is trivial; an algorithm may just classify all roads as congested. In accordance, predicting few jams may result in very high precision since false positives are minimized. The accuracy of the algorithms, given as:

$$A = \frac{TP + TN}{N} \tag{14}$$

where N is the total number of roads, is also rather skewed. Classifying all roads as free flow achieves quite high accuracy since congested roads are generally much fewer. A rather more appropriate metric for such distributions is the *F-measure*[2], given using precision P and recall R:

$$F = \frac{2PR}{P + R} \tag{15}$$

All of the above metrics were computed for the algorithms that we implemented. The bar graph of Figure 6a illustrates their values for $\theta = 0.4$. As shown in that figure, the SVMs appear to outperform all other algorithms in terms of accuracy. Their precision is also the highest, while having rather satisfactory recall. The kNN classifier is also satisfactory, with high accuracy and high precision. Thus, discriminative classifiers seem to preform slightly better in terms of accuracy and precision. In terms of recall, however, GMMs outperform other algorithms, while the performance of Gaussian Bayes is also satisfactory. The F-measure is rather too close to successfully discriminate among the algorithms. In any case, comparing the algorithms over only one theta value is rather hasty, since the θ's are not fully normalized, and their effect on the performance of the algorithms is not fully compared.

Further investigating the relative effectiveness of the algorithms involves analyzing the effect of θ on the performance of the algorithms. This is accomplished using a *Receiver Operating Characteristic (ROC)* curve. Drawing the curve involves the calculation of *sensitivity* and *specificity*, given different values of θ. Sensitivity is defined equally to recall, as in equation 13, and specificity is defined as:

$$Specificity = \frac{TN}{FP + TN} \tag{16}$$

The x-axis of the ROC curve is $1 - Specificity$ and the y-axis is $Sensitivity$. The curve for the algorithms we implemented is shown in Figure 6b. A metric of the performance of a classifier is the distance of its curve from the random curve.

As shown in Figure 6b, the SVM classifier outperforms all other algorithms, regardless of the value of theta. In addition, kNN is very stable, clearly outperforming GMMs for low values of $1 - Specificity$, and achieving better performance than Gaussian Bayes for high values of $1 - Specificity$. The Gaussian Bayes classifier is strong only for low values of $1 - Specificity$, indicating its inability to identify jams above a certain threshold. GMMs, on the other hand, appear to overgeneralize, thus not achieving high $Sensitivity$ scores for low values of $1 - Specificity$. Hence, the main conclusion dictates that discriminative classifiers perform better than generative ones. This is also indicated by the *Area Under the Curve (AUC)* metric, which is shown in Figure 6c. Both kNN and SVMs outperform the generative classifiers, with the latter's curve achieving to cover more than 75% of the total area.

B. Visualization Evaluation

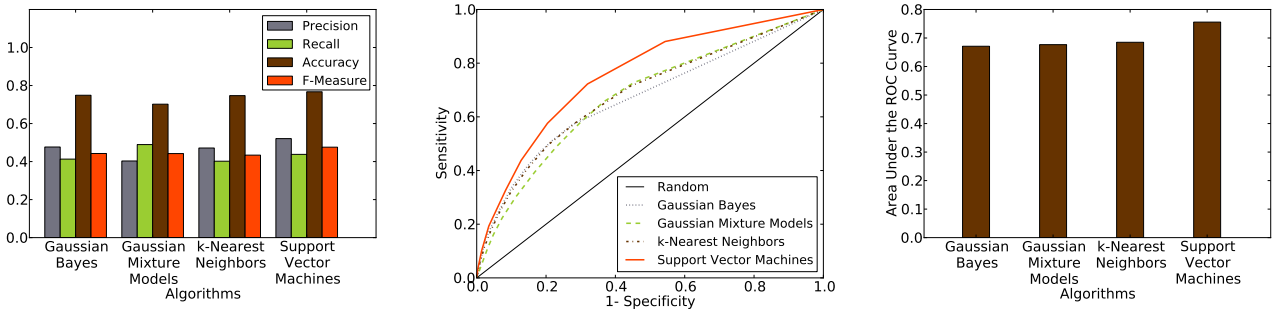
Evaluating cluster analysis is a task well-known in literature. Although there are several metrics, selecting an appropriate metric is difficult, since problem specifics are usually quite narrow. In this work, we evaluate the clustering algorithm given in Figure 4 both internally, based on the data itself in a quantitative manner, and externally, in a qualitative manner.

Internal evaluation should provide with an indication of properly selected clusters in terms of density. A widely used metric that determines whether the clusters are dense enough and properly separated from each other, is the *Davies Bouldin (DB) index*, named after its creators D. L. Davies and D. W. Bouldin [23]. The DB index is calculated as follows:

$$DB = \frac{1}{N} \sum_{x=1}^{N} \max_{x \neq y} \left(\frac{\bar{d}_x + \bar{d}_y}{d(c_x, c_y)} \right) \tag{17}$$

where x (and y) denote indexes of the x-th (and y-th) cluster out of the total number of clusters N. Also, $\bar{d}_x$ is the average distance of all the points of cluster x from the centroid of the same cluster c_x, and $d(c_x, c_y)$ is the distance between the centroids of clusters x and y. When the nominator of equation (17) is small, the clusters can be considered quite densely connected since the *intra-cluster* distances are small, so the points are quite close to each other. Concerning the denominator, the larger it is the better, since it reflects the *inter-cluster* distances, i.e. the distances between the clusters themselves. Thus, a low DB index is considered generally satisfactory. In our case, the value of the index is 0.37. This is actually a very low value (indicatively most literature algorithms expect values higher than 0.5), indicating our cluster analysis algorithm is effective. Having such low DB index is rather expected, since jams are mostly easily distinguishable. Congestion areas are dense and our algorithm manages to identify whether a new cluster should be formed in each case.

[2]This is also known as the *F1-measure*, since it is originally defined as $F_\beta = (1 + \beta^2 PR)/(\beta^2 P + R)$ for $\beta = 1$. Although other values of β are also possible, the value 1 is usually preferred.



(a) (b) (c)

Fig. 6. Evaluation metrics for the four classifiers. Precision, recall, accuracy and F-Measure for $\theta = 0.4$ are shown in 6a. The ROC curve shown in (b) is drawn using values of θ from 0 to 1 with step 0.1, and the total Area Under the Curve (AUC) is shown in (c).

Internal evaluation is certainly useful to provide with a general measure of the performance of clustering algorithms. However, since clustering is an unsupervised task, there is no safe way to determine the effectiveness of an algorithm without human interference. Thus, it is useful to evaluate the results of our algorithm externally in a qualitative manner. An example of the congestion areas found by the algorithm is shown in Figure 5. As shown in this figure, our algorithm is robust, constructing clearly distinguished clusters. In addition, the visualization for each cluster convex indeed covers the cluster and indicates its centroid.

VIII. Conclusion

Although congestion prediction has been analyzed by several researchers, our approach in comparing generative versus discriminative classifiers is novel and shall provide new insight on the problem. Discriminative classifiers seem to outperform generative ones. Although Gaussian Bayes and GMMs are quite strong in certain time intervals, kNN and SVMs appear much more robust with satisfactory results in the whole dataset. The ROC curve actually indicates that discriminative classifiers are more effective overall, both in a sensitive and an insensitive scenario.

Concerning traffic jam visualization, our algorithm, based on DBSCAN, yields quite satisfactory results. Apart from the internal evaluation, the visualization seems informative and comprehensive. Future work in congestion visualization includes devising new algorithms, based mainly but not restricted to density-based cluster analysis. Furthermore, comparing the algorithms in different datasets could also yield interesting results.

Concerning traffic congestion prediction, the question whether using discriminative over generative classifiers is preferable requires further work. Performing evaluation on different datasets could yield more distinguishing results. In any case, this work sets an open question for further research on this interesting problem.

Acknowledgment

This work was supported by the EU FP7/2007-2013 (DG INFSO.G4-ICT for Transport), under grant agreement no. 288094 (project eCOMPASS).

References

[1] S. Yang, "On feature selection for traffic congestion prediction," *Transportation Research Part C: Emerging Technologies*, vol. 26, no. 0, pp. 160 – 169, 2013.

[2] G. Huisken, "Short-term congestion forecasting: Time series versus fuzzy sets," in *Proceedings of the 19th Annual South African Transport Conference*, 2000.

[3] G. Huisken and M. van Maarseveen, "Congestion prediction on motorways: A comparative analysis," in *Proceedings of the 7th World Congress on Intelligent Transport Systems*, 2000.

[4] W. Labeeuw, K. Driessens, D. Weyns, T. Holvoet, and G. Deconinck, "Prediction of congested traffic on the critical density point using machine learning and decentralised collaborating cameras," in *New Trends in Artificial Intelligence : 14th Portuguese Conference on Artificial Intelligence pages*, 2009, pp. 15–26.

[5] J. He, Q. He, G. Swirszcz, Y. Kamarianakis, R. Lawrence, W. Shen, and L. Wynter, "Ensemble-based method for task 2: Predicting traffic jam," in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, 2010, pp. 1363–1365.

[6] M. Roccetti and G. Marfia, "Modeling and experimenting with vehicular congestion for distributed advanced traveler information systems," in *Computer Performance Engineering*, ser. Lecture Notes in Computer Science, A. Aldini, M. Bernardo, L. Bononi, and V. Cortellessa, Eds. Springer Berlin Heidelberg, 2010, vol. 6342, pp. 1–16.

[7] G. Marfia and M. Roccetti, "Vehicular congestion modeling and estimation for advanced traveler information systems," in *Wireless Days (WD), 2010 IFIP*, 2010, pp. 1–5.

[8] G. Marfia and M. Roccetti, "Vehicular congestion detection and short-term forecasting: A new model with results," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 7, pp. 2936–2948, 2011.

[9] G. Marfia, M. Roccetti, and A. Amoroso, "A new traffic congestion prediction model for advanced traveler information and management systems," *Wireless Communications and Mobile Computing*, vol. 13, no. 3, pp. 266–276, 2013.

[10] O. Masutani, H. Sasaki, H. Iwasaki, Y. Ando, Y. Fukazawa, and S. Honiden, "Pheromone model: application to traffic congestion prediction," in *AAMAS'05*, 2005, pp. 1171–1172.

[11] Y. Ando, O. Masutani, H. Sasaki, H. Iwasaki, Y. Fukazawa, and S. Honiden, "Pheromone model: Application to traffic congestion prediction," in *Engineering Self-Organising Systems'05*, 2005, pp. 182–196.

[12] Y. Ando, Y. Fukazawa, O. Masutani, H. Iwasaki, and S. Honiden, "Performance of pheromone model for predicting traffic congestion," in *AAMAS'06*, 2006, pp. 73–80.

[13] M. Wojnarski, P. Gora, M. Szczuka, H. S. Nguyen, J. Swietlicka, and D. Zeinalipour, "Ieee icdm 2010 contest: Tomtom traffic prediction for intelligent gps navigation," *2012 IEEE 12th International Conference on Data Mining Workshops*, vol. 0, pp. 1372–1376, 2010.

[14] B. P. Welford, "Note on a Method for Calculating Corrected Sums of Squares and Products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[15] T. Diamantopoulos, D. Kehagias, F. G. König, and D. Tzovaras, "Investigating the effect of global metrics in travel time forecasting," to appear in *Proceedings of 16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[16] T. Cheng, J. Haworth, and J. Wang, "Spatio-temporal autocorrelation of road network data," *Journal of Geographical Systems*, vol. 14, no. 4, pp. 389–413, 2012.

[17] Y. Kamarianakis and P. Prastacos, "Space-time modeling of traffic flow," *Comput. Geosci.*, vol. 31, no. 2, pp. 119–133, Mar. 2005.

[18] B. Hamner, "Predicting travel times with context-dependent random forests by modeling local and aggregate traffic flow," in *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ser. ICDMW '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1357–1359.

[19] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, vol. 14. MIT Press, 2002, p. 841.

[20] G. Bouchard and B. Triggs, "The trade-off between generative and discriminative classifiers," in *Proceedings in Computational Statistics, 16th Symposium of IASC*. Physica-Verlag, 2004, pp. 721–728.

[21] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Second International Conference on Knowledge Discovery and Data Mining*, E. Simoudis, J. Han, and U. Fayyad, Eds. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[22] R. L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set," *Information Processing Letters*, vol. 1, pp. 132–133, 1972.

[23] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-1, no. 2, pp. 224–227, 1979.