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

Themistoklis Diamantopoulos\textsuperscript{a}, Dionysios Kehagias\textsuperscript{a,1}, Felix G. König\textsuperscript{b}, Dimitrios Tzovaras\textsuperscript{a}

\textsuperscript{a}Centre for Research & Technology Hellas, Information Technologies Institute, P.O. Box 60361, 57001 Thessaloniki, Greece
\textsuperscript{b}TomTom International BV, An den Treptowers 1, 12435 Berlin, Germany

Abstract

The field of Intelligent Transportation Systems has lately raised increasing interest due to its high socio-economic impact. This work aims on developing efficient techniques for traffic congestion prediction and visualisation. We have designed a simple, yet effective and scalable model to handle sparse data from GPS observations and reduce the problem of congestion prediction to a binary classification problem (jam, non-jam). An attempt to generalise 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 congestion prediction algorithm based on using correlation metrics to improve feature selection. Concerning the visualisation of traffic jams, we present a traffic jam visualisation approach based on cluster analysis that identifies dense congestion areas.

Keywords: traffic congestion prediction; classification; clustering

Résumé

Le domaine des systèmes de transport intelligents a suscité un vif intérêt en raison de son impact socio-économique élevé. Des techniques efficaces pour la prédiction et la visualisation de la congestion du trafic ont été développées. Ce projet a permis de concevoir un modèle simple, mais efficace et évolutif de traiter données de observations GPS et de réduire le problème de la prédiction de la congestion à un problème de classification binaire (congestion, non-congestion). Nous essayons de généraliser le problème en explorant l'impact des classifiants discriminants par rapport aux classifiants génériques lorsqu'ils sont utilisés pour produire des résultats dans un intervalle de 30 minutes d'avance sur le temps présent. En outre, nous présentons un nouvel algorithme de prédiction de la congestion qui utilise des mesures de corrélation pour améliorer la sélection de paramètres. Concernant la visualisation des embouteillages, nous présentons une approche en grappes de visualisation qui identifie les zones de congestion dense.

Mots-clé: prédiction de la congestion du trafic; classification; clustering

---

\textsuperscript{1} Tel.: +30 2311 257716; fax: +30 2310 474128.
E-mail address: diok@iti.gr.
1. 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 is to provide a clear formulation of the traffic jams prediction problem, in order to suggest new ways of improving the relevant algorithms found in the current literature, as well as to provide an efficient means of visualizing congestion in a meaningful and comprehensive way. Congestion prediction is a challenging task that is broadly studied by several researchers (Yang, 2013; Huisken, 2000; Huisken and van Maarseveen, 2000). Congestion prediction techniques face the problems of congestion modelling and congestion prediction. The former concerns defining the jam and non-jam states of a road and the latter concerns predicting jams using classification algorithms. Apart from these problems, in this paper the problem of visualizing congestion is taken into account.

Our contributions are numerous. Upon presenting a heuristic framework for modelling congestion, we explore the nature of four classifiers that are used by current literature for the task of traffic congestion prediction. Furthermore, we present a new algorithm, indicating the importance of feature selection, and exploring the effect of using correlation metrics on network data to predict congestion. Finally, we present a traffic jam visualisation approach, based on cluster analysis that illustrates dense congestion areas in a comprehensive manner.

Section 2 of this paper summarizes state-of-the-art techniques of current literature. The dataset used and the congestion modelling approach are described in sections 3 and 4 respectively. In Section 5, the application of current approaches as well as a new approach for congestion prediction are presented. Section 6 presents an algorithm for visualizing jams. Our evaluation is presented in Section 7 and Section 8 concludes this paper.

2. State-of-the-art on Congestion Prediction

Several approaches, such as the one by Yang (2013) formulate the problem as a binary classification task. Data is drawn from loop detectors that provide 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, that are used to train a classifier. Upon identifying the features for each road, classification is performed assuming Gaussian distributions over the datasets, so that the final probability for the state of a road is given by the quotient of the distribution. The author uses mean precision to evaluate the results and performs analysis to determine the optimal number of features.

An approach by Huisken and Maarseveen (2000), upon prior research in (Huisken, 2000), involves collecting data using induction loops on the motorway A10 of Amsterdam. The metrics used include traffic volume and occupancy, i.e. the percentage that the detector is “on”. In addition, the average and standard deviation of speed in a road segment is calculated using series of loop detectors. The metrics are given as features to classifiers and the road state is determined by an oracle. The authors test different classifiers, including multi-linear regression, an ARMA model, a heuristic Fuzzy Logic classifier, and three neural network classifiers.

Few researchers have attempted to adapt the congestion prediction techniques to speed probe (rather than loop detector) scenarios. W. Labeeuw et al. (2009) apply different classifiers using a speed probe dataset on a ring road. However, a ring road scenario is actually limited and their approach suffers from scalability issues. Other lines of work, such as the one by G. Marfia et al. (2011), are mainly directed towards distributed systems, thus deviate from the scope of this paper.

In contrast with the previous approaches, the 2nd task of the IEEE ICDM Contest: TomTom Traffic Prediction for Intelligent GPS Navigation (Wojnarski et al., 2010) that took place in 2010 provided with interesting insight concerning the raw form of the data and the noise they may have. The dataset consisted of sequences of road segments. Jammed road segments were given during a 20-minute interval and the goal was to identify the jams in the next 40-minute interval, and the length of the output road sequence (number of jams). The winning algorithm compared sequences of the training set with the respective ones in the test set and classified sequences to jams by finding whether the most similar training sequences are also jammed.


3. Data Acquisition

Although the approaches of Section 2 are interesting, their scope is limited since they are dependent on specific scenarios. The algorithms are adapted to loop detector or other specific scenarios (Yang, 2013; Huisken and van Maarseveen, 2000; Wojnarski et al., 2010) and are mostly unable to scale in a speed probe scenario (Huisken and van Maarseveen, 2000). In any case, their applicability in speed probe datasets is not thoroughly tested.

Our dataset was given by TomTom1, in the context of the European-funded research project eCOMPASS2. The data consists of two week speed probes, from Mar 18, 2012, to Mar 31, 2012, for the city of Berlin. The road network initially contains nodes and links, which are defined as straight lines. The dataset is provided as a set of instantaneous speeds that correspond to links, including also the direction of the vehicle in the link. The application of any algorithm on the data is prone to both scalability and noise issues because the data are sparse since a link may have minimal or no data for different time moments. Thus, the links are combined to form roads, i.e. segments between two intersections. In addition, 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. Noticeably, 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 (Welford, 1962).

Finally, we were also given the free flow speed of each road, which is defined as the average speed that the vehicles appear to have when traffic on the road is minimum. It is usually measured during night hours. The free flow speed has proven to be quite useful for identifying the notion of congestion, as it is explained in Section 4.

4. Modelling Traffic Congestion

According to the current state-of-the-art (see Section 2), modelling traffic congestion usually implies the definition of appropriate heuristics with respect to the metrics provided. 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 $\mu_T$ and $\sigma_T$ respectively. The presence or not of a jam in road $r$ at time $t$ is determined using the following equation:

$$S_{r,t}^\tau = \begin{cases} 
    \text{Jam}, & \text{if } 100 \cdot \frac{\mu_r(t)}{FF_r} \leq \mu_T \text{ and } \sigma_r(t) \leq \sigma_T \\
    \text{NonJam}, & \text{otherwise}
\end{cases}$$

(2)

where $FF_r$ is the free flow speed of road $r$. Thus, the mean speed threshold $\mu_T$ is defined as a percentage of the free flow speed, while the standard deviation threshold $\sigma_T$ is defined such that it ensures capturing anomalies. Intuitively, equation (2) provides quite realistic distinction of jams. Assuming values 60 and 40 for $\mu_T$ and $\sigma_T$ 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.

5. Traffic Congestion Prediction

Upon reducing congestion prediction to a binary classification problem, subsections 5.1 and 5.2 discuss existing approaches to the problem and subsection 5.3 presents a new algorithm using improved feature selection.

5.1. Feature Selection

Each road is represented as a time series containing its average speed for each time interval. Existing approaches in traffic congestion prediction are either univariate or multivariate. Univariate approaches take into account only the time series of the road of which the state is predicted, whereas multivariate ones use also data from other roads. Univariate analysis fails to capture the nature of congestion, since traffic in a specific area is highly dependent on the local neighbourhood. Intuitively, the congested state of road depends not only on the previous speed values of the road itself, but also on the values of roads in the local area of the road. Thus, feature selection should take into account both the road of which the state is predicted and its local area, as in Figure 1.

---

1 http://www.tomtom.com/
2 http://www.ecompass-project.eu/
The features consist of the arithmetic means of the road $r$ as well as its neighbouring roads $n_1, n_2, ..., n_s$, where $s$ is the number of neighbouring roads. Three time intervals are taken into account, corresponding to present time $t$, to one ($t-1$) and to two ($t-2$) intervals before present time. The class feature is nominal ($Jam$, $NonJam$) and corresponds to one interval ahead of present time ($t+1$). It is also possible to set the class feature to two or three intervals ahead, thus predicting jams earlier. The scenario is defined as binary classification with scalar features. Training data is in the form depicted in Figure 1, while testing data is identical, excluding the class feature value.

5.2. Existing Approaches

Classifiers can be divided into two categories: generative and discriminative classifiers. Generative classifiers attempt to model the probability distribution of the data in order to classify a new observation, whereas discriminative ones classify observations without creating such a model (Ng and Jordan, 2002; Bouchard and Triggs, 2004). Generative classifiers are known to perform better on small datasets since they approach their asymptotic error much faster, whereas discriminative classifiers are better suited towards large datasets where they exhibit lower error. Thus, in a large real traffic scenario, discriminative classifiers seem preferable for their effectiveness. However, generative classifiers may also be a good fit since their models exhibit better noise tolerance which could prove useful in a sparse data scenario. Concerning the problem at hand, 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 more 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. Hence, we implemented four techniques in order to investigate their effectiveness on the problem.

5.2.1. Gaussian Bayes

The Gaussian Bayes classifier is a generalisation of the Bayes theorem for modelling beliefs and it is called naïve since conditional independence between the features is assumed. Let the features be $x_1, x_2, ..., x_n$, a probability is derived for each possible value $y_i$ of the class feature $y$ according to the following equation:

$$P(y_i | x_1, x_2, ..., x_n) = \frac{\prod_{i} P(x_i | y_i) \cdot P(y_i)}{\sum_{i} \prod_{i} P(x_i | y_i) \cdot P(y_i)}$$

(3)

where $P(x_i | y_i)$ is the probability of a feature having value $x_i$ given that the class is $y_i$. Thus, the nominator of equation (3) is defined as the product of those probabilities for all features, multiplied by $P(y_i)$ which is the prior probability of the class being $y_i$. The denominator serves only as a normaliser so that the probabilities of all class feature values sum up to 1. Using the classifier in a scalar-features scenario, such as the congestion prediction one, requires making an assumption about the distribution of the data, i.e. approximating the $P(x_i | y_i)$ term. Assuming the distribution is Gaussian, with mean $\mu_{yi}$ and standard deviation $\sigma_{yi}$, the probability is computed as:

$$P(x_i | y_i) = \frac{1}{\sqrt{2\pi\sigma_{yi}^2}} \cdot \exp\left(-\frac{(x_i - \mu_{yi})^2}{2\sigma_{yi}^2}\right)$$

(4)

Finally, using equation (3) the algorithm provides a probability for each class value.

5.2.2. 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. Given $d$-dimensional distributions of the form:

$$P_\theta(x) = \frac{1}{(2\pi)^{\frac{d}{2}} \cdot \sqrt{\det{\sigma}}} \cdot \exp\left(-\frac{(x - \mu)^T \cdot \sigma^{-1} \cdot (x - \mu)}{2}\right)$$

(5)
where $\mu$ is the mean and $\Sigma$ is the covariance matrix of the distribution, the probability of a mixture of $K$ Gaussians, i.e. a Gaussian Mixture Model (GMM) is:

$$P(x) = \sum_{j=1}^{K} w_j \cdot P_{x_j} (x)$$

where $w_j$ is the prior probability of the $j$-th Gaussian. Given the traffic congestion prediction scenario, the problem is reduced to creating the models for the two classes (Jam, NonJam). Upon training the models, any new sample is assigned to a model according to the probability given by equation (6). In fact, the class of a new instance may be selected using the probabilities derived from the two models. Finally, training the algorithm comes down to defining an optimisation problem given equations (5) and (6). The problem involves finding the parameters, i.e. the means and the deviations of each distribution, that maximise the probability of the model. The problem is approached using the two-step Expectation-Maximisation (EM) algorithm. The expectation step calculates the probability of sample $x_i$ to belong to mixture $k$ and the maximisation step estimates the parameters of mixture $k$ using the probabilities. Continuously iterating over the two steps results in constructing the model.

5.2.3. k-Nearest Neighbors

The $k$-Nearest Neighbors (kNN) classifier is given a training set in the form of features $x_{t_1}, x_{t_2}, ..., x_{t_n}$ and the (known) class feature $y_t$ for all different time moments $t$ of the training set. Given a sample $x_{t_1}, x_{t_2}, ..., x_{t_n}$ to classify, the algorithm initially finds the Euclidian distances between the sample and each of the known training samples. 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.

5.2.4. Support Vector Machines

The main idea of Support Vector Machines (SVMs) is to construct a hyperplane that sets apart the classes of a sample. Although there may be several such hyperplanes, SVMs attempt to find the optimal hyperplane, i.e. the hyperplane that has maximum distance from instances on both sides. In classifying jams, SVMs are given training data in the form of features $x_{t_1}, x_{t_2}, ..., x_{t_n}$ and the class feature $y_t$, in order to construct the optimal hyperplane in an $n$-dimensional space. Given a new instance $x_{t_1}, x_{t_2}, ..., x_{t_n}$, the classifier returns whether the instance is on the Jam or on the NonJam side of the hyperplane, and possibly its distance from the hyperplane.

We used a Radial basis function (RBF) kernel with $\gamma$ equal to 0.1, and the penalty multiplier $C$ equal to 10.

5.3. Using Global Features – A New Approach Towards Traffic Congestion Prediction

The feature selection procedure presented in subsection 5.1 is certainly based on a fair intuition. The assumption that congestion is a local effect is strong. However, as proven also in (Diamantopoulos et al., 2013), the condition of a road may depend on several other roads all over the road network. Intuitively, this is expected to occur since e.g. a ring road could affect major roads of the city even though they may not be neighbouring. Given the feature set of Figure 1, the problem of predicting congestion for a road $r$ is effectively reduced to determining the roads that affect the state of the road at hand, i.e. roads $n_1, n_2, ..., n_r$. Generally, real networks, such as the one described in Section 3, may contain thousands of roads, and furthermore some roads may have sparse or noisy data, or simply data that have no impact on traffic. Concerning congestion on road $r$, it is necessary to determine the roads that are strongly “connected” to road $r$. Current trends suggest using correlation metrics, such as the Cross Correlation Function (CCF) (Cheng, Haworth and Wang, 2012, Kamarianakis and Prastacos, 2005), or the Coefficient of Determination (CoD) (Cheng, Haworth and Wang, 2012, Diamantopoulos et al., 2013), or a transformation, such as Principal Component Analysis (PCA) (Hammer, 2010).

In this work, the use of correlation metrics is preferred since they have been proven to be a strong fit for real-life traffic scenarios (Diamantopoulos et al., 2013). The most well known correlation metric is the Pearson Product-Moment Correlation Coefficient (PPMCC). Given two time series $x$ and $y$, the PPMCC is defined as:

$$PPMCC_{xy} = \frac{E[(x - \mu_x)(y - \mu_y)]}{\sigma_x \sigma_y}$$

where $\mu_x$ is the mean and $\sigma_x$ is the standard deviation of the time series $x$ (Haworth and Wang, 2012). In the case of traffic however, the time series of two roads may not be correlated for the same time intervals. For example, the speed of road $r$ at time $t$ may not be affected by the speed of road $n_1$ at time $t$. However, it may be affected by
the speed of $n_j$ at time $t-1$. This implies that there is a need for a metric that takes temporal lag into account. A widely used lag-aware metric is the CCF between two time series $x$ and $y$, defined as:

$$CCF_{xy} = \frac{E[(x_i - \mu_x)(y_{i+k} - \mu_y)]}{\sigma_x \sigma_y}, \quad k = 0, \pm 1, \pm 2, \ldots$$

(9)

where $k$ is the lag between the two time series. Similarly to (Cheng, Haworth and Wang, 2012, Diamantopoulos et al., 2013), we take the squared value of CCF and multiply by 100 to obtain the CoD of two time series $x$ and $y$:

$$CoD_{xy}(k) = 100 \cdot \left[\frac{E[(x_i - \mu_x)(y_{i+k} - \mu_y)]}{\sigma_x \sigma_y}\right]^2, \quad k = 0, \pm 1, \pm 2, \ldots$$

(10)

Thus, the algorithm requires two main steps to train a model for predicting the state of a road $r$ in the next time interval. The first step involves finding the $s$ most well-correlated roads with the road $r$ (in our scenario $s=10$). The CoD value between road $r$ and each network road is calculated and the $s$ roads for which their CoD with $r$ is the largest are selected. After that, these roads form the feature set that is shown in Figure 1. Hence, the second step includes executing the SVM algorithm (that proved to be the most efficient – see subsection 7.1).

6. Traffic Congestion Visualisation

In this section, we present a traffic jam visualisation 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 slowdown. However, jam indications in sparse areas may indicate noise in data or insignificant micro-jams. Based on the aforementioned remarks, we perform cluster analysis using a modification of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996) algorithm in order to identify dense congestion areas. Our algorithm is shown in Figure 2.

**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 \cup NeighRoads'
            if R' not in any cluster
                C.add(R')

Fig. 2. The density-based clustering algorithm that accepts as input the minimum number of roads that form a cluster (MinRoads) and the order of neighbouring roads to be considered (o) and creates the clusters.

As shown in Figure 2, 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 neighbouring roads that are congested. If the number of congested roads in the nearby area is higher than MinRoads, then a new cluster is initialised and expanded using the function ExpandC. ExpandC initially adds the road at hand (R) to the cluster and then iterates over each of the neighbouring roads (NeighRoads) and adds both the neighbouring road (R’) and the latter’s new neighbours (NeighRoads’) as long as they form a congested area with more than MinRoads.
Fig. 3. Visualisation of traffic congestion that (a) are real or (b) are predicted using convex hulls that encircle clusters of jammed roads. The gradient level of each convex is proportional to the expected jam strength at the corresponding area.

Upon identifying congestion areas, the problem lies in visualizing them. The clusters are 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. \((x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3), \ldots\). The problem of finding the smallest convex shape that contains these points is known in the literature as finding the convex hull of the points. We used Graham Scan, a simple algorithm named after Graham (1972), 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 finalises when it returns to the initial point.

An example of the visualisation is shown in Figure 3, which depicts a subset of the map with real and predicted jams for 10 minutes ahead of present time. The congested areas are shown with red colour, while the gradience of the colour is a metric of the concentration of congestion in the specific area. Intuitively, the strongly-coloured 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 hull.

7. Evaluation

This section presents the results of our evaluation for traffic congestion prediction and congestion visualisation.

7.1. Prediction Evaluation

Concerning congestion prediction, the binary classifiers analyzed in Section 5 are evaluated against data for the city of Berlin (see Section 3). The data are split into two weeks, thus data for one day of the first week (Thursday of week 1) were used to train the algorithms and data from the subsequent day on the next week (Thursday of week 2) were used to test them. An 8-hour interval from 10:00 to 18:00 was used since it was interesting enough to contain the number of jams required for a satisfactory evaluation. This interval appeared to have the most jams throughout the day. The parameters \(\mu_T\) and \(\sigma_T\) were set to 60 and 40 respectively for both training and testing the algorithms. The classifiers had to predict the condition of each road for 1 to 6 intervals ahead, i.e. 5 to 30 minutes ahead of present time. Congestion is expected to be rarer than free flow conditions. Thus, the classifiers require appropriate adaptation to the nature of the data, which is normal since all classifiers can actually produce scalar outputs in a pre-specified interval. For example, Gaussian Bayes provides a probability of a road being congested, and SVMs provide not only the subspace (divided by the hyperplane) to which the sample belongs but also its distance from the hyperplane. Converting this output to a nominal \((Jam, NonJam)\) value involves creating a threshold \(\theta\), that receives values from 0 to 1 with step 0.05.
Given a value of $\theta$, a classifier provides nominal values (Jam, NonJam) for all the roads of the dataset. For each run, True Positives (TP) are defined as the number of roads that are predicted to be congested and actually are congested, False Positives (FP) as the ones predicted to be congested yet traffic flows freely, True Negatives (TN) are defined as the ones predicted to be on free flow and traffic flows freely, and False Negatives (FN) as the ones predicted to be on free flow but actually are congested. Given these metrics, one may calculate the precision, $P = TP/(TP+FP)$, the recall, $R = TP/(TP+FN)$, the accuracy, $A = (TP+TN)/(N)$, and the F-measure, $F = (2PR)/(P+R)$, of an algorithm. Precision denotes the percentage of jams that were correctly classified as jams, and recall denotes the percentage of jams that were successfully predicted. In a skewed distribution scenario, such as the congestion prediction one, the metrics may be deceiving. For example, 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 minimised. In fact, classifying all roads as free flow achieves quite high accuracy since congested roads are much fewer. Thus, F-measure is actually the most appropriate metric.

The bar chart of Figure 4a illustrates their values when approximately 50% of jams are successfully predicted ($\theta$’s were selected so that recall is almost equal for all algorithms). As shown in that figure, discriminative classifiers seem to perform quite better than generative ones. Both SVMs and the kNN classifier achieve higher precision than Gaussian Bayes and GMMs, and considering recall is fixed, this indicates that the first ones produce much less false positives than the last ones. The low accuracy values of Gaussian Bayes and GMMs suggest the classifiers may also provide numerous false negatives. Finally, the F-measure indicates once again that the performance of both SVMs and kNN is indeed superior. Concerning the effect of our proposed feature selection procedure, the SVM with CoD seems quite strong, achieving best results in all aforementioned metrics.

Since comparing the algorithms over predicting a percentage of jams is hasty, the effect of $\theta$ on the performance of the algorithms is analyzed. This is accomplished using a Receiver Operating Characteristic (ROC) curve. Drawing the curve involves calculating sensitivity and specificity for different values of $\theta$. Sensitivity is defined equally to recall ($TP/(TP+FN)$) and specificity is defined as $FN/(FP+TN)$. The x-axis of the ROC curve is 1-Specificity and the y-axis is Sensitivity. As shown in the curve of Figure 4b, the SVMs using CoD outperform all other algorithms, regardless of the value of $\theta$. In addition, “plain” SVMs perform satisfactorily and kNN is stable, clearly outperforming both Gaussian Bayes and GMMs. In fact, the performance of the last two algorithms seems to be rather inadequate when they capture from 20% to 70% of jams (i.e. Sensitivity from 0.2 to 0.7). This can be interpreted as a problem with the modelling nature of the algorithms. Generative models are subject to overgeneralisation, thus they achieve low Sensitivity for low values of 1-Specificity. These conclusions are also indicated by the Area Under the Curve (AUC) metric, which is shown in Figure 4c. Both SVMs and kNN outperform the generative classifiers, and the curve of SVM with CoD covers almost 75% of the total area.

Finally, we used the McNemar’s test (McNemar 1947) in order to test the significance of the results. The test checks the performance of two classifiers $C_1$, $C_2$ for a specific $\theta$ value by initially counting 4 variables $A$, $B$, $C$, and $D$. $A$ is the number of instances that were correctly classified by both classifiers, $B$ is the number of instances correctly classified by $C_1$ but incorrectly classified by $C_2$, $C$ is the number of instances that is correctly classified
by $C_2$ but incorrectly classified by $C_1$, and $D$ is the number of instances that is incorrectly classified by both classifiers. The test statistic is $\chi^2=(B–C)^2/(B+C)$. Using $\chi^2$ one can derive the degrees of freedom, and finally the two-sided $p$ value, denoting the level at which the difference in the results of the two classifiers is significant. Due to space limitations, we refrain from presenting a full test for all different algorithm pairs for all $\theta$ values. In any case, the test showed that classifier performance is clearly distinguishable for values 1-Specificity between 0.2 and 0.8. Indicatively, the results of the SVMs with CoD classifier are significant at 0.1% ($p<0.001$) comparing with either Gaussian Bayes, GMMs, or $k$NN, while comparing with the “plain” SVMs, the difference was significant at 0.5% ($p<0.005$) near 1-Specificity value 0.2 and significant at 0.1% ($p<0.001$) near value 0.8. Additionally, the differences of all other classifier pairs in this interval are significant at 0.1% ($p<0.001$).

7.2. Visualisation Evaluation

Evaluating cluster analysis is a task well-known in the relevant literature. In this work, we evaluate the clustering algorithm internally and externally. Internal evaluation provides an indication of properly selected clusters in terms of density. A metric that determines whether the clusters are dense enough and properly separated from each other, is the Davies Bouldin (DB) index, named after Davies and Bouldin (1979), which is defined as:

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

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 $c_x$ and $c_y$. When the nominator of equation (16) is small, the clusters can be considered densely connected since the intra-cluster distances are small, so the points are 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. Thus, a low DB index is considered satisfactory. Although the value of the DB index may depend on data quality, our use of it is rather indicative in order only to support our proof of concept without claiming full assessment of the algorithm. In our case, its value is 0.37. This is rather low (most algorithms expect values higher than 0.5), indicating our algorithm is effective. Having low DB index is expected, since congestion areas are dense and our algorithm manages to identify whether a new cluster should be formed in each case.

Although internal evaluation provides a general measure of the performance of clustering algorithms, since clustering is an unsupervised task, there is no safe way to determine the effectiveness of an algorithm without human interference. Thus, we may evaluate our algorithm externally in a qualitative manner. An example of the areas found by the algorithm is shown in Figure 3. As shown in this figure, our algorithm constructs clearly distinguished clusters, and the visualisation of each convex indeed covers the cluster and indicates its centroid.

8. Conclusion

Although the problems of congestion modelling and congestion prediction are well known in current literature, the contribution of this paper lies in applying the algorithms in a real-life probe dataset scenario. Apart from applying different algorithms to the problem, successfully achieving scalability and noise tolerance, we also made a comparison between generative and discriminative classifiers, which resulted in the observation that the latter outperforming the former ones. Furthermore, a new feature selection technique, based on using CoD, was introduced and applied to the problem. With support from subsection 7.1, our technique outperforms all other techniques, indicating that using correlation metrics on global data achieves very satisfactory results. Concerning congestion visualisation, our algorithm yields satisfactory results and the visualisation seems comprehensive.

Future work in traffic congestion prediction is related to determining whether to use discriminative over generative classifiers. Our conclusions can be extended by being tested on different datasets. Finally, the development of new algorithms can certainly provide a solid base of future research, by experimenting with the selection of new features that yield to better performance and prediction accuracy.

Acknowledgments

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


