Time-Varying Mixtures of Markov Chains: An Application to Road Traffic Modeling

Sean Lawlor, *Student Member, IEEE*, Michael G. Rabbat, *Senior Member, IEEE*

***Abstract*—Time-varying mixture models are useful for representing complex, dynamic distributions. Components in the mixture model can appear and disappear, and persisting components can evolve. This allows great flexibility in streaming data applications where the model can be adjusted as new data arrives. Fitting a mixture model with computational guarantees which can meet real-time requirements is challenging with existing algorithms, especially when the model order can vary with time. Existing approximate inference methods may require multiple restarts to search for a good local solution. Monte-Carlo methods can be used to jointly estimate the model order and model parameters, but when the distribution of each mixand has a high-dimensional parameter space, they suffer from the curse of dimensionality and and from slow convergence. This paper proposes a generative model for time-varying mixture models, tailored for mixtures of discrete-time Markov chains. A novel, deterministic inference procedure is introduced and is shown to be suitable for applications requiring real-time estimation, and the method is guaranteed to converge at each time step. As a motivating application, we model and predict traffic patterns in a transportation network. Experiments illustrate the performance of the scheme and offer insights regarding tuning of the algorithm parameters. The experiments also investigate the predictive power of the proposed model compared to less complex models and demonstrate the superiority of the mixture model approach for prediction of traffic routes in real data.**

I. Introduction

Mixture models are a useful tool for modelling complex distributions. Allowing a mixture model to be dynamic, with a varying number of components as well as time-varying mixand parameters, allows for a very flexible model which can be applied to many forms of streaming data. In this paper we develop a procedure for fitting time-varying mixtures of discrete-time finite-state Markov chains.

We are motivated by the application of traffic modelling using streams of automatic vehicle identification data [1]. A model of traffic patterns can be used for a variety of traffic analysis applications [2]. Potential applications include vehicle path prediction, visualization of the difference in traffic flows between varying time-spans, and other forms of statistical detection and estimation such as convoy detection [3]. Models for traffic routes, and traffic flow, are commonly used in law enforcement, city planning, private for-profit parking industry, and other applications [4].

S. Lawlor and M.G. Rabbat are with the department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, Canada.
E-mail: sean.lawlor@mail.mcgill.ca, michael.rabbat@mcgill.ca

This work was funded in part by Genetec Inc., and by the Natural Sciences and Engineering Research Council of Canada.

A. Automatic Vehicle Identification Data

Traditionally, road traffic is measured and monitored using sensors such as inductive loops embedded in the roadway that provide counts of the number of vehicles passing by a particular location. Increasingly, traffic can also be measured and estimated using data from *automatic vehicle identification* (AVI) sensors, such as bluetooth sensors or cameras performing license plate recognition. Bluetooth sensors record the MAC address of the in-vehicle entertainment/communications system. License plate recognition cameras recognize and record the license plate of vehicles which pass in their field of view. Sensors that read the highway toll-pass ID of an equipped vehicle when it passes nearby (such as E-ZPass in northeastern U.S.A. [5]) provide similar information. While each of these three types of sensors has different strengths and weaknesses (e.g., reliability, detection rate), in all cases the resulting data can be leveraged in the same fashion for traffic modelling. Each AVI observation includes the ID of the vehicle observed, the time, and the location of the observation.

In typical monitoring systems, AVI sensors report their data in a streaming manner to a fusion center. Since observations are made in an event-driven manner (with events corresponding to a vehicle passing near a sensor), the sequence of observations of a particular vehicle at different sensors can be viewed as an irregularly sampled time series. Since vehicles may come and go freely within the coverage area of the sensor network, the observation sequence of a vehicle can start or stop at any point and the number of observations (the length of the observation sequence) of a vehicle is variable.

This paper considers the case where a region has been instrumented with a collection of static AVI sensors. The sensors are treated as the states of a Markov process, and the trajectories of individual vehicles are modeled as realizations of this process. The observation times are only used to order the sequence of locations where a vehicle is observed, leading to a discrete-time model for the sequence of AVI sensors.

B. Previous Work

1) Time-Varying Mixture Models: In this paper we propose to model a network of AVI sensors as a time-varying mixture of discrete-time Markov chains. Previous work on parameter estimation in time-varying mixture models typically adopts a Bayesian non-parametric perspective and focuses on mixtures of Dirichlet processes [6]. However these works perform inference using Monte-Carlo based methods, where computational performance guarantees are difficult to achieve. Stephens [7] proposes a Bayesian representation of a mixture model using a marked point-process. However MCMC-based inference methods have challenges in high-dimensional parameter spaces (e.g, the entries of a transition matrix), especially in the streaming setting. Work on deterministic inference methods of these types of models, where computational timing guarantees are more feasible, is limited. Also much of the previous work on time-varying mixture models focuses on mixtures of Gaussian distributions [8] which do not directly apply to AVI data because of the discrete nature of the observations.

The algorithm proposed in this work includes a novel automatic selection of the mixture model order. The early literature on model order selection includes approaches using model selection criteria such as the *Akaike information criterion* (AIC) [9], *Bayes information criterion* (BIC) [10], and *model description length* (MDL) criterion [11]. These criteria are used to score potential candidate models and require computing all of the candidate models before selecting one, which can be computationally intensive. Models which require only computing a single or small set of candidate models are better suited to this class of problems, in which we would like to track traffic patterns as they evolve.

More recent work on automatic model order selection is performed by Corduneanu and Bishop [12] where a Bayesian specification of the mixture model is made. Variational methods are used to anneal the number of components from some maximum number down to a number which specifies the model most accurately. Another important work in annealing a maximum number of components is the CEM^2 algorithm [13]. This class of work continues with Chen et al. [14] which replaces the variational estimation schemes of [12] with an EM approach using unique penalty terms in the likelihood specification of the mixture model. These penalty terms allow for components to disappear if they are deemed unnecessary.

We opt for an alternate approach which does not require pre-specification of the maximum number of components. Verbeek et al. [15] explore this by taking a greedy approach of adding clusters as deemed necessary. At each iteration their algorithm fits a model with k components using the EM algorithm and then searches over a set of possible candidate models for a suitable new component to insert (if necessary), then fitting the new model with $k + 1$ components using the EM algorithm. This repeats until no new components are inserted.

2) Traffic Modelling: Previous approaches to modelling road network traffic typically use other sensors such as road counters, induction loops, cell phone data, and manual counting to estimate traffic patterns [16], [17].

Typically in traffic estimation research, estimation of traffic flow is done through the estimation of an *origin-destination* (OD) matrix. Peterson [18] provides an overview of many methods and estimation techniques typically applied to the OD-matrix estimation problem. Most of the methods are initialized using manually-collected traffic survey data, which is time-consuming and labor-intensive to gather, and then they use data from count sensors such as inductive loops to update or refine the OD matrix. An OD matrix is a helpful tool in city planning, but it only quantifies the volume of vehicles traveling between each origin and destination. Estimation of the OD matrix using AVI data is also addressed in [19]. The *path* which traffic takes, however, is not estimated in traditional OD matrix estimation schemes. The model proposed in this paper goes one step further and estimates the paths that vehicles take to move from one location to another.

A preliminary version of this work appears in [20]. There we introduced the idea of modelling traffic using a mixture of Markov chains. This paper expands on that work in a number of ways: 1) the inference algorithm described here differs from the one in [20] and admits a simpler analysis of convergence and computational complexity; 2) the performance evaluation is greatly expanded; 3) the issue of tuning algorithm parameters is investigated and a procedure for determining suitable choices is described.

C. *Contribution*

This work proposes a new method for the estimation of a time-varying mixture of Markov chains. The proposed algorithm extends the Classification EM algorithm [21] so components of the mixture can be greedily added as necessary. It also proposes novel penalization methods on model complexity to automatically choose a more appropriate model. A discussion of how the choice of the threshold on the proposed penalization method based on the Kullback-Leibler (KL) divergence influences is also provided. The accuracy of the parameter estimates and estimated model order generated by this algorithm are analyzed and it is shown that the selection of the correct model order and estimation of the component parameters is well solved by this method. Experiments illustrate that the proposed time-varying mixture model representation of traffic is more accurate than an approach using a single Markov chain (i.e., a model of order one) for predicting vehicle routes when predictions must be made from a small number of initial observations.

D. *Paper Organization*

The rest of the paper is organized as follows. Section II gives a detailed description of the problem setup and assumptions. Section III proposes a generative time-varying mixture model. Section IV discusses *maximum a posteriori* (MAP) inference for the proposed model and motivates the need to use approximate inference schemes for this class of models. Section V describes a novel approximate algorithm for parameter inference of time-varying mixture models. The proposed method involves two tuning parameters, and Section VI investigates the proper choice of these parameters. Section VII presents experimental results, including a comparison to an alternative MCMC-based approach, and we conclude in Section VIII.

II. Problem Description

This section gives a more detailed description of the problem of modelling traffic using AVI data. We describe characteristics of the measurement system that make the problem challenging. Then we discuss the assumptions made and describe performance metrics which will be used to evaluate the proposed model and estimation algorithm.

A. Vehicle Identification Data

Consider a system of urban roads instrumented with AVI sensors. When a vehicle passes within range of a sensor, the sensor records the vehicle's unique identifier as well as the time and location. AVI sensors typically have a short range of detection (e.g., 10 meters). The measurements from many of these sensors, at different locations in the road network, are transmitted to a fusion center whose goal is to estimate the traffic patterns, or flows, through the network of sensors.

More formally we consider a collection of C sensors. Let the set of sensor indices be $\Omega = \{1, ..., C\}$. The sequence of observations of vehicle i produced by the network is $x^i = \{x^i_0, x^i_1, x^i_2, ..., x^i_{n_i}\}$ where $x^i_j \in \Omega$ is the sensor that captured the j^{th} observation of vehicle i. In this paper we focus on modelling the routes present in the transition sequences between sensors and therefore ignore the transition times it takes to move from one sensor to another.

B. Measurement system characteristics and assumptions

We make the following assumptions about the measurements made by the system. First the sensors are synchronized so that the timestamps from different sensors are directly comparable. This is justified since existing AVI sensors are typically equipped with GPS receivers that provide reliable and accurate synchronization. This allows us to properly time-order observation sequences of vehicles so there are no errors in the order of vehicle observations.

Second, we assume that a vehicle cannot be observed by two sensors at *exactly* the same time. This ensures that the order of observations of a vehicle is well-defined, and it is justified when the timestamps at each sensor are of a sufficiently high resolution and sensors have non-overlapping fields of view.

Third, we assume that the vehicle identifiers (e.g. license plate of the vehicle) recorded by the sensors do not contain errors and that there are no missed detections (i.e., a vehicle is always detected when it passes within range of a sensor). This is reasonable for license plate recognition cameras, which have a very low error rate and high detection probability, but it is less reasonable for other AVI sensors (e.g., Bluetooth-based). Extending the model and inference method to accommodate errors and missed detections is a subject of future work.

Fourth, we assume that the sensors are static and their locations are known to the fusion center. Thus, the structure of the network of AVI sensors does not change over time.

Finally, we assume that the sensors transmit their observations to the fusion center over a transmission channel with negligible delays and errors so that ordering errors due to delayed data is impossible. This assumption can be justified since the number of bits required to encode an individual measurement is very small, so transmission delay should be significantly smaller than the time between successive observations of a vehicle.

C. Sequential modeling

In this work we consider a sequential setting where the observations arrive in time windows $t \in \{1, 2, ...\}$. To simplify the presentation, we take all time windows to be the same length. For example, in a typical urban deployment, a time window may be one hour long. A typical vehicle trip may last 20–30 minutes, and the number of observations of the vehicle (which depends on the number and density of cameras deployed) may be between five and ten. We assume that the distribution governing vehicle routes is stationary within a time window, and it may vary from one time window to the next. The number of vehicles observed in time window t is $N(t)$. The collection of all AVI observations in time window t are organized into the set of per-vehicle observation sequences, $X(t) = \{x^1, ..., x^{N(t)}\}$. In each time window a model is created to describe the data observed within that time window. This model uses the model estimated in the previous time window as a prior on the current model. This enforces the notion of the model evolving. The details of the model used to estimate traffic patterns are described in Section III.

Our goal is to model vehicle routes, as given by the sequence of locations where the vehicle is observed. One possible application is to predict vehicle trajectories through the network. In Section VII we report the results of experiments where, after fitting the model with training data, we test the accuracy of predicting the next location where a vehicle will be observed, x^i_n, given the history $x^i_0, \dots, x^i_{n-1}$.

Note that we do not model the observation times (or inter-observation times) in this work, since the focus is on modelling vehicle routes. Such information could be incorporated, if it is of interest in a given application, following the approach described in [22]. There a probability density function $f(t|x, x')$ is associated with each pair of sensors $(x, x') \in \Omega \times \Omega$, and it is used to express the probability that a vehicle is observed by sensor x' no more than t seconds after having been observed at sensor x. These densities can be estimated while also fitting other route-related model parameters; see [22] for details.

III. The Model

Observations from automatic vehicle identification (AVI) sensors result in timestamped and location-tagged observations of vehicles identified by their unique identifier. If one groups the observations by vehicle, then each vehicle has a location (state) where it was initially observed at and then a sequence of following observations. Each following observation can be viewed as a transition from the previously observed state. We can then model these observations as transitions between the states of a Markov chain where the states of the chain are the AVI sensors. In order to capture more complex traffic patterns we propose to use a mixture of Markov chains, since a single Markov chain may be unable to properly summarize the multiple traffic patterns present.

We model the sequence of sensors observing a particular vehicle i as following a first-order discrete-time Markov chain. Under the first-order Markov assumption, sufficient statistics for the sequence $x^i = (x^i_0, x^i_1, \dots, x^i_{n_i})$ of observations of vehicle i are the initial state x^i_0 and a matrix[1] $X^i \in \mathbb{Z}^{|\Omega| \times |\Omega|}_{\geq 0}$ of transition counts, where $X^i_{j,k}$ is the number of times the

[1]We denote the set of non-negative integers by $\mathbb{Z}_{\geq 0}$.

vehicle i was observed at sensor k immediately after having been observed at sensor j.

If one considers a single Markov chain at time window t with initial state distribution $\pi(t)$ and transition matrix $P(t)$, which we group together and write as $\phi(t) = (\pi(t), P(t))$ for convenience, the likelihood of the observations of vehicle i is

$$p(x^i|\phi(t)) = \pi_{x_0^i}(t) \prod_{j=1}^{|\Omega|} \prod_{k=1}^{|\Omega|} (P_{j,k}(t))^{X^i_{j,k}}. \qquad (1)$$

Using only one Markov chain to describe all of the traffic observations in time-window t provides limited predictive capacity. For example, envision a large volume of traffic moving from the suburbs into the downtown core of a city during the morning rush-hour. At the same time, traffic may be moving from the downtown core into the suburbs at a much smaller volume. If this behavior was modelled using a single Markov chain, the small traffic volume moving from downtown to the suburbs may get lost. Modelling this as a mixture model may allow us to identify these inter-twined traffic flows occurring within the same time-window (e.g., with one component capturing each "direction" or "flow"). Therefore, in order to capture more traffic patterns present in the data, we propose to use a mixture of Markov chains. This mixture model contains $M(t)$ mixture components at time-window t, and the mth mixture component has parameters $\phi^{(m)}(t) = \{\pi^{(m)}(t), P^{(m)}(t)\}$, $m = 1, \dots, M(t)$. The likelihood for a vehicle x^i under each individual component is then $p(x^i|\phi^{(m)}(t))$ as in (1).

Each vehicle's path is assumed to be generated by one of the components in the mixture model. In order to model this, we follow the standard approach of using the binary random variables $z_i^{(m)}$ for each vehicle i and mixture component m, with $z_i^{(m)}$ equal to one if and only if the movement of vehicle i is governed by mixture component m, and $\sum_{m=1}^{M(t)} z_i^{(m)} = 1$. Further let $\boldsymbol{\alpha}(t) = (\alpha^{(1)}(t), ..., \alpha^{(M(t))})$ denote the distribution of $z_i^{(m)}$ with $\Pr(z_i^{(m)} = 1) = \alpha^{(m)}(t)$ for a vehicle i observed during time-window t.

Since the assignment variable $\boldsymbol{z}_i$ is not observed, one typically works with the marginalized complete-data likelihood,

$$p(x^i|\boldsymbol{\alpha}(t), \boldsymbol{\phi}(t)) = \sum_{m=1}^{M(t)} \alpha^{(m)}(t) p\left(x^i|\phi^{(m)}(t)\right) \qquad (2)$$

where $\boldsymbol{\phi}(t)$ is the collection of all mixture component parameters $\phi^{(m)}(t)$ in time-window t. Finally, assuming that the paths of different vehicles are i.i.d., the likelihood of $N(t)$ vehicles generating the observations $X(t)$ in time-window t is

$$p(X(t)|\boldsymbol{\alpha}(t), \boldsymbol{\phi}(t)) = \prod_{i=1}^{N(t)} \sum_{m=1}^{M(t)} \alpha^{(m)}(t) p\left(x^i|\phi^{(m)}(t)\right). \quad (3)$$

In our previous work [22] we considered the problem of detecting when two or more vehicles were traveling along correlated routes (i.e., detecting convoys), and non-convoy traffic was modelled as independent samples from a (static) mixture of Markov chains.

The remainder of this section defines the model dynamics.

A. Mixture Component Death

Consider the mixture model in time-window $t-1$ with $M(t-1)$ components. This mixture model can be described by the vector of mixture weights $\boldsymbol{\alpha}(t-1)$ and parameters for each component $\phi^{(m)}(t-1), m = 1, ..., M(t-1)$. Let $\boldsymbol{\Theta}(t) = (\boldsymbol{\alpha}(t), \boldsymbol{\phi}(t))$ denote the complete set of model parameters at time t.

Existing components either persist or die between time-windows, and they die with probability

$$p_d(\phi^{(m)}(t), \alpha^{(m)}(t)). \qquad (4)$$

This probability can be constant or could be related to the rate at which $\alpha^{(m)}(t)$ is decreasing signifying that the component is "dying". It could also be related to a rate of decrease in the *Kullback-Leibler* (KL) divergence between a pair of components signifying they may be "merging". The notions of a component dying on its own or merging with another are handled by the algorithm in a following section.

For each component death, the mixture weight assigned to that component is distributed proportionally among the remaining components which is similar to the generative model of Stephens [7].

B. Persisting Component Evolution

If a component persists from time-window $t-1$ to t its parameters evolve according to the dynamics

$$\phi^{(m)}(t) \sim H(\phi^{(m)}(t-1)), \qquad (5)$$

where $H(\cdot)$ is a distribution parametrized by the previous iteration's component's parameters with pdf $h(\phi^{(m)}(t)|\phi^{(m)}(t-1))$. In order to easily compute the MAP estimate, this distribution would ideally be conjugate to the distribution parametrized by $\phi^{(m)}(t)$. In the traffic model considered here, with a *time-varying mixture model* (TVMM) of *discrete-time Markov chains* (DTMC), we adopt the model that

$$P^{(m)}(t) \sim \begin{bmatrix} \mathrm{Dir}(P^{(m)}_{(1,:)}(t-1)) \\ \mathrm{Dir}(P^{(m)}_{(2,:)}(t-1)) \\ \vdots \\ \mathrm{Dir}(P^{(m)}_{(|\Omega|,:)}(t-1)) \end{bmatrix} := \mathbf{Dir}\left(\boldsymbol{P}^{(m)}(t-1)\right) \qquad (6)$$

and

$$\pi^{(m)}(t) \sim \mathrm{Dir}(\pi^{(m)}(t-1)) \qquad (7)$$

where $\mathrm{Dir}(\cdot)$ is the Dirichlet distribution.

C. Mixture Component Birth

The number of mixture components born at each time-window follows a Poisson distribution with parameter $\lambda(t)$ as $n_b \sim \mathrm{Pois}(\lambda(t))$. This allows for a countably infinite number of components to be born in each time-window. Each new component which is born in time-window t has parameters distributed according to some distribution, $\phi^{(m)}(t) \sim H(\mathbb{G})$, which has a density $h(\phi^{(m)}(t)|\mathbb{G})$. Here $\mathbb{G}$ are the parameters, or set of parameters, of the base distribution of the model. In our TVMM DTMC these distributions are

$$P^{(m)}(t) \sim \mathbf{Dir}(G_p) \text{ and } \pi^{(m)}(t) \sim \mathrm{Dir}(G_\pi) \tag{8}$$

where $\mathbb{G} = \{G_p, G_\pi\}$ are the base distribution parameters.

Each new component which is born has a mixture weight distributed according to a Beta distribution so that

$$\alpha^{(M(t)+1)}(t) \sim \mathrm{Beta}(1, M(t)). \tag{9}$$

Following [7], for each new component added, the weights of previously existing components are scaled down by $\left(1 - \alpha^{(M(t)+1)}\right)$. This model for component birth can be related to the notion that components are "born from the prior" so they have some shared knowledge between all components but are independent from all other existing components. In this model new components also receive progressively smaller weights which helps in that no new component can appear and take the majority of the weight in one iteration. It will be born small and, if necessary, will grow through its dynamics. This notion of a curbed mixture weight with additional births is also adopted from [7] and differs from the typical non-parametric Dirichlet process mixture models in that the parameter of the Beta distribution varies, depending on the number of existing components, over time.

D. Mixture Weight Dynamics

The fully dynamic model is now defined, save for the evolution of the vector of mixture weights, $\boldsymbol{\alpha}$, between time windows $t-1$ and t. The dimension of this vector can change between time-windows and it is modelled as following a multinomial distribution. Consequently, it is natural to model these weights as evolving according to a Dirichlet distribution because the Dirichlet distribution is conjugate to the multinomial distribution [23]. However the parameter to the Dirichlet must be the same dimension as the variable output from the distribution. Therefore we propose that one considers that the model first determines the component births and deaths, where the mixture weights are adjusted for each birth and death, and then the dimension of the mixture weight vector is known. After the births and deaths are established, we have a resized and rescaled weight vector for time $t-1$ called $\boldsymbol{\alpha}'(t-1)$. With this rescaled mixture weight vector the next mixture weight vector is distributed according to

$$\boldsymbol{\alpha}(t) \sim \mathrm{Dir}(\boldsymbol{\alpha}'(t-1)). \tag{10}$$

IV. MAP Inference for a Mixture of Markov chains

In the ideal case we would estimate the parameters of a time-varying mixture model at time t via Bayes' Rule and get a maximum a posteriori (MAP) estimate of the parameters. Given the data from time-windows $1, ..., t$, denoted $\mathbf{X}(1:t) = \{X(1), ..., X(t)\}$, we want to estimate the parameters that maximize the posterior $p(\boldsymbol{\Theta}(t)|\mathbf{X}(1:t))$ which can be rewritten as

$$\begin{aligned} p(\boldsymbol{\Theta}(t)|\mathbf{X}(1:t)) =& \frac{p\left(X(t), \boldsymbol{\Theta}(t)|\mathbf{X}(1:t-1)\right)}{p(X(t))} \\ &\propto p\left(X(t)|\,\boldsymbol{\Theta}(t)\right) p\left(\boldsymbol{\Theta}(t)|\mathbf{X}(1:t-1)\right) \qquad (11) \\ = p\left(X(t)|\,\boldsymbol{\Theta}(t)\right) &\int p\left(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1)\right) \\ &\times p\left(\boldsymbol{\Theta}(t-1)|\mathbf{X}(1:t-1)\right) d\,\boldsymbol{\Theta}(t-1) \end{aligned}$$

by application of Bayes' theorem where the denominator in the first equation is simply a scaling factor so it does not affect the maximization. The map estimate is then given by

$$\begin{aligned} \hat{\boldsymbol{\Theta}}_{MAP}(t) = \underset{\boldsymbol{\Theta}(t)}{\arg\max}\, p\left(X(t)|\,\boldsymbol{\Theta}(t)\right) \qquad (12) \\ \times \int p\left(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1)\right) p\left(\boldsymbol{\Theta}(t-1)|\mathbf{X}(1:t-1)\right) d\,\boldsymbol{\Theta}(t-1). \end{aligned}$$

As the dimensionality of the model grows, computing the integral over the entire parameter space $\boldsymbol{\Theta}(t-1)$ quickly becomes intractable. We can however further develop the terms appearing in the MAP objective.

In order to solve this maximization, we need expressions for $p(X(t)|\,\boldsymbol{\Theta}(t))$ and $p(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1))$ in (12). The likelihood $p(X(t)|\,\boldsymbol{\Theta}(t))$ is given by (3). The transition density $p(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1))$ is governed by the model dynamics described in Sec. III. The term $p(\boldsymbol{\Theta}(t-1)|\mathbf{X}(1:t-1))$ is the recursive posterior from time window $t-1$, acting as the current time-window's prior.

In order to express the parameter transition density we need to match the persisting mixture components in time window $t-1$ with those in time window t. If n_d components die from $t-1$ to t then $M(t-1) - n_d$ components persist.

Consider the set of choices of size $M(t-1) - n_d$ from the set $\{1, ..., M(t-1)\}$. Furthermore consider all permutations of each resulting choice in this set of choices. The concatenation of all permutations of all possible choices from this set is denoted by $\mathcal{A}(t)$. This is the set of all possible mappings of the components at time-window $t-1$ to those which persisted to time-window t. In the definition of $p(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1))$ we need to marginalize over all the possible mappings in $\mathcal{A}(t)$. We can note that incorrect mappings will likely have very small probability, and there will likely only be one mapping which results in a reasonably large probability. Now if one further assumes that the probability of death for any component is p_d

for all t, then we can define $p(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1))$ to be[2]

$$\begin{aligned} & p(\,\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1)) \qquad (13)\\ &= \sum_{n_d(t)=\max(0,M(t-1)-M(t))}^{M(t-1)} \Bigg\{ f_d(n_d(t)|p_d, M(t-1))\times \\ &\quad f_b(M(t)-M(t-1)+n_d(t)|\lambda(t))\times \\ &\quad \left(\sum_{A\in\mathcal{A}(t)} \prod_{m=1}^{M(t-1)-n_d(t)} g(\phi^{(m)}(t)|\phi^{(A_m)}(t-1))\right)\times \\ &\quad \left(\prod_{n_b(t)=M(t-1)-n_d(t)+1}^{M(t)} g(\phi^{(m)}(t)|\mathbb{G}) f_\alpha(\alpha^{(m)}(t)|M(t-1))\right) \\ &\quad \times f_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}(t)|\boldsymbol{\alpha}'(t-1)) \Bigg\} \end{aligned}$$

where

$$f_d(x|p_d, M(t-1)) = p_d^x\,(1-p_d)^{M(t-1)-x} \qquad (14)$$

$$f_b(k|\lambda(t)) = \frac{\lambda(t)^k e^{-\lambda(t)}}{k!} \qquad (15)$$

$$f_\alpha(x|M(t)) = \frac{(1-x)^{M(t)-1}}{\mathrm{B}(1, M(t))} \qquad (16)$$

$$f_{\boldsymbol{\alpha}}(\mathbf{x}|\boldsymbol{\alpha}'(t-1)) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha}'(t-1))} \prod_{m=1}^{|\mathbf{x}|} x_m^{\alpha'_m(t-1)} \qquad (17)$$

and where $\mathrm{B}(\cdot)$ is the Beta function and $f_{\boldsymbol{\alpha}}()$ controls the evolution of the *rescaled* mixture component weights ($\boldsymbol{\alpha}'$) defined in (10).

In order to do inference on this model, one could resort to a Monte-Carlo based method, such as Gibbs Sampling. However, due to the high-dimension of the model parameter space the number of burn-in and sampling iterations of the MCMC method quickly goes to infinity in order to sample reasonable candidates. As a simple example, consider the TVMM DTMC model again with a state space of $|\Omega| = 25$ representing a network of 25 nodes. If the current model in time-window t is comprised of 3 DTMCs, one needs to estimate the mixture weights ($\boldsymbol{\alpha}(t)$), the initial state distributions ($\pi(t)$), and transition matrices ($P(t)$) for each component resulting in

$$\begin{aligned} &(|\alpha|-1) + |\alpha| \times [(|\Omega|-1) + |\Omega|(|\Omega|-1)] \\ &\quad = 2 + 3(24 + 25\times 24) = 1874 \end{aligned}$$

free parameters which need to be estimated. In addition to the number of free parameters which need to be estimated, there is the issue that random samples from the posterior distribution in a MCMC method will be candidate mixture models which have a varying number of components. Therefore determining a method of averaging the posterior samples or choosing the best candidate model in order to obtain the best parameter estimate is not a straightforward task as well.

Now since MCMC-based methods result in unreasonable sampling times, the question becomes how does one estimate the parameters of this type of model? An alternate might be to try and derive an exact Maximum a Posteriori (MAP) estimate based on the probability definitions in eqs. (12) to (17). If one excludes the definition of $p(\boldsymbol{\Theta}(t)|\,\boldsymbol{\Theta}(t-1))$ for a moment and only focuses on $p(X(t)|\,\boldsymbol{\Theta}(t))$ this would result in a maximum likelihood (ML) estimate of the mixture model which we can show is still intractable to compute.

To demonstrate the infeasibility of a ML solution in the mixture model, consider the hidden assignment variable $z_i^{(m)}$ for vehicle i outlined in Section III again. This variable $z_i^{(m)} = 1$ when vehicle i's path is distributed according to mixture component m and 0 otherwise. Now the updated likelihood function using this hidden variable definition is

$$f(X(t)|\,\boldsymbol{\Theta}(t)) = \prod_{i=1}^{N(t)} \sum_{z_i\in\mathcal{Z}} \prod_{m=1}^{M(t)} f(x^i, z_i^{(m)}|\phi^{(m)}(t)) \qquad (18)$$

$$= \prod_{i=1}^{N(t)} \sum_{z_i\in\mathcal{Z}} \prod_{m=1}^{M(t)} \left(\pi_{x_0^i}^{(m)}(t) \prod_{j=1}^{|\Omega|} \prod_{k=1}^{|\Omega|} \left(P_{j,k}^{(m)}(t)\right)^{X_{j,k}^i} \right)^{z_i^{(m)}} \qquad (19)$$

where the marginalization over all possible z_i's make computing the ML estimate quickly go to infinity. The actual number of iterations to marginalize over $\mathcal{Z}$ is $M(t)^{|X(t)|}$ which one can see is dependent on the mixture-model order in time-window t and the number of vehicles observed in time-window t. This marginalization requirement is true of all types of mixture models, not only a TVMM of DTMCs. These types of models are traditionally estimated using the Expectation-Maximization algorithm [24] which replaces the hidden assignments z_i from binary random variables with weighted probabilities by stating that vehicle i has probability $\alpha_i^{(m)}$ of being distributed according to component m. This greatly simplifies the computation of the marginal likelihood and it then becomes possible to maximize the *expected complete log-likelihood* via a simple recursive algorithm. It proceeds by choosing the component mixture probabilities based on the ratio of the likelihood of the data being distributed according to each individual component. This is performing a form of *clustering*, which can motivate the need to first *cluster* the data into groups and then updating the individual component parameters based on the data membership to each component. We propose the automatic hard EM algorithm defined in the following section to address the clustering and estimation problem in a time-varying mixture model.

V. Automatic Hard EM Estimation for Time-Varying Mixture Models

When the true number of underlying components in a mixture model is unknown a priori, then a mixture model estimation scheme needs to estimate the number of mixture components as well as the parameters of the individual mixture model components and the mixing weights for each component. Traditional *Expectation-Maximization* (EM), *Classification EM* (CEM), and k-Means algorithms require that the number of mixture components be known beforehand [24],

[2]We adopt the convention that $\prod_{i=1}^{0} x = 1$.

[21], [25]. This means that a modification to these types of estimation schemes is necessary.

We propose an approach to model order estimation similar to that described in [15], which progressively adds components. However, the proposed approach eliminates the search over candidate models by simply adding the same base candidate model $\mathbb{G}$ at each iteration. This reflects the generative model where new components born at each iteration are generated according to this global prior.

A. Modified Hard EM Clustering

The EM algorithm [24] can be viewed as performing a form of clustering where the cluster assignments are soft: for each datapoint we get a likelihood that the datapoint came from each of the possible clusters. Then the cluster parameter estimates are updated to the weighted average of the data assigned to the cluster. Once the weighted cluster assignments and component parameters stop changing, to within some tolerance, the algorithm is considered to have converged. By modifying the EM algorithm to make hard assignments instead of weighted assignments, one obtains an alternate method commonly referred to as Hard EM. This is a reasonable approximation when the mixture components are well-separated, and it is generally much faster to compute. Hard EM algorithms can be considered to have converged once the component assignments stop changing since, once this occurs, the component parameter estimates will be exactly the same in subsequent iterations.

Since we assume that vehicles follow a single DTMC in the mixture model, we use Hard EM to estimate the DTMC that generated the observed vehicle path. We then use the paths of the group of vehicles assigned to a component to compute a simple MAP estimate of the component's parameters.

B. The Proposed Algorithm

For each time-window t, the proposed inference algorithm is divided into three phases. The first phase is a modification of the Hard EM algorithm to jointly estimate model parameters and the model order. The phase is initialized with the mixture model parameters estimated in the previous time-window, and it allows for an arbitrary number of new components to be created. In the second phase, components that have fewer than L_α data points assigned to them are trimmed from the model. In the last phase, pairs of the remaining components are merged if the KL divergence between them falls below a threshold L_{KL}. These last two phases simplify the resulting model, reducing the storage requirements and simplifying future evaluations of the model, while also acting as a sort-of regularizer to avoid overfitting. Pseudocode is shown in Alg. 1 and each phase is described in more detail next.

1) Modified Hard EM: The modification that we propose to the standard hard EM algorithm allows the algorithm to greedily add new components as necessary, so the number of components does not need to be specified in advance. The first phase begins by adding a new component with parameters $\mathbb{G}$ (line 3). It then finds the component which achieves the maximum likelihood for each observed path (line

Algorithm 1 Automatic Hard EM algorithm for T time-windows

```
1:  $\mathbf{\Theta}(0) = \{\}$
Require: Data $\mathbf{X}(1:T) = \{\mathbf{X}(1), ..., \mathbf{X}(T)\}$
2:  for $t \in 1..T$ do
3:      $\hat{\phi}_0 = \{\phi^{(m)}(t-1)\}^{\forall m} \bigcup \{\mathbb{G}\}$
4:      while $|S_{M(t)}| > 0$ do
5:          $i = 1$
6:          repeat
7:              for $m \in \{1, ..., M(t)\}$ do
8:                  $S_m = \left\{x^j : p(x^j|\hat{\phi}_{i-1}^{(m)}) > p(x^j|\hat{\phi}_{i-1}^{(m')}) \forall m' \neq m\right\}$
9:                  $\hat{\phi}_i^{(m)} = \arg\max_{\hat{\phi}_i^{(m)}} f(S_m|\hat{\phi}_i^{(m)}) h(\hat{\phi}_i^{(m)}|\hat{\phi}_0^{(m)})$
10:             end for
11:             $i = i + 1$
12:         until $|\hat{\phi}_i^{(m)} - \hat{\phi}_{i-1}^{(m)}| = 0, \forall m$
13:         $\alpha^{(m)}(t) \leftarrow \frac{|S_m|}{|\mathbf{X}(t)|}, \forall m$
14:         if $|S_{M(t)}| > 0$ then
15:             $\hat{\phi}_i(t) = \{\hat{\phi}_i^{(m)}(t)\}^{\forall m} \bigcup \{\mathbb{G}\}$
16:             $\hat{\phi}_0(t) = \{\hat{\phi}_0^{(m)}(t)\}^{\forall m} \bigcup \{\mathbb{G}\}$
17:         end if
18:     end while
19:     $\mathbf{\Theta}(t) \leftarrow \left\{(\alpha^{(1)}(t), \hat{\phi}_i^{(1)}(t)), ..., (\alpha^{(M(t))}, \hat{\phi}_i^{(M(t))}(t))\right\}$
20:     Trim components in $\mathbf{\Theta}(t)$ with $\alpha^{(m)} < L_\alpha$
21:     Merge components in $\mathbf{\Theta}(t)$ with
22:         $D_{KL}(\phi^{(m)}(t)||\phi^{(m')}(t)) < L_{KL}$
23: end for
```

8). From the vehicle paths assigned to each component, the MAP estimate of the parameters is computed (line 9). In the MAP estimate computation, $f(S_m|\hat{\phi}_i^{(m)})$, is the likelihood of the data assigned to component m, and $h(\hat{\phi}_i^{(m)}|\hat{\phi}_0^{(m)})$ is the same as in (5). By adding components with initial parameters of $\mathbb{G}$, this is equivalent to setting the parameters of the prior over the component parameter distribution to the same values. This means that the distribution over the components, $h(\cdot|\cdot)$, is the same for all newly added components. These steps are repeated until the assignments no longer change from one iteration to then next (line 12). Then (line 13), the algorithm updates the mixture weights $\alpha^{(m)}(t)$ based on the number of observations assigned to each component.

Next (line 14), the algorithm checks if any data has been assigned to the most-recently appended component. Since this component was initialized as the base distribution $\mathbb{G}$, it is more vague (i.e. it has higher variance or spread) than the components propagated from the previous time-window. If paths are assigned to this component, then at least one new cluster is deemed to have formed, and another component is added (lines 14–17). Then the Hard EM algorithm is executed again with this new, larger, model order. This process repeats until another instance of the base component $\mathbb{G}$ is added and no data points are assigned to it. In the worst case scenario where all observed paths in a time-window are well-separated, one could end up with a number of components equal to the number of observed paths. However in the empirical studies discussed in Section VII, we did not observe this extreme, and typically $M(t) \ll |\mathbf{X}(t)|$. This estimation technique allows new component creation and existing component movement.

If this were the only phase of the algorithm, the number of components would continuously grow without limit since we only add new components in this phase of each time-window. Therefore we next need a method of annealing the number of components to those which are necessary, both for computational considerations and to avoid over-fitting.

2) Trimming Weak Components: The most basic way of trimming components that do not describe the data is to remove components that are associated with a small proportion of data; these are precisely the components with small weights $\alpha^{(m)}(t)$. This is implemented in line 20. The threshold, L_α, could be constant, a time-window dependent value, or a data size-dependent value. One might wish to keep all components which are sufficiently well-separated from the rest, in which case L_α would be set to zero.

3) Merging Similar Components: A more complex notion is examining when two components become very close to each other (meaning they are essentially the same distribution). After the clustering has converged and weak components are deleted, we compute the KL divergence between all pairs of components and merge pairs if the KL divergence from one to the other is below a threshold, L_{KL}, starting from the smallest difference and proceeding sequentially. To merge components m and m' we take the weighted average of their parameters,

$$\alpha^{\text{merged}}(t) = \alpha^{(m)}(t) + \alpha^{(m')}(t)$$
$$\phi^{\text{merged}}(t) = \frac{\alpha^{(m)}(t)\phi^{(m)}(t)}{\alpha^{(m)}(t)+\alpha^{(m')}(t)} + \frac{\alpha^{(m')}(t)\phi^{(m')}(t)}{\alpha^{(m)}(t)+\alpha^{(m')}(t)}.$$

This is implemented in line 21 of Alg. 1. The choice of an appropriate threshold L_{KL} is discussed in Section VI.

C. Intuitive Explanation of the Automatic Hard EM Algorithm

Next we provide an intuitive explanation to help understand the function of the Automatic Hard EM algorithm. Suppose that the mixture model estimated at time-window $t-1$ has two components, and 500 data points are observed in time-window t. In the first pass through the CEM algorithm (lines 6–12), each data point will be assigned to one of three components: the two from the previous time-window, and the additional one added in line 3.

After this first pass of the CEM has converged, if no data points were assigned to the third mixture component then the algorithm determines that the original two components were sufficient to model the data, and the while loop exists. Otherwise, if at least one data point was assigned to the third mixture component, then the algorithm allows for the possibility of increasing the model order further. Another generic component is added in lines 15–16, and another pass of the CEM algorithm is performed. This repeats until no data points are assigned to the most recently added component, at which point it is determined there is no need to further increase the model order. Thus this first phase involves greedily increasing the model order as long as the CEM continues to associate data points to new mixture components.

To be concrete, suppose that this phase finishes with a model of order $M(t) = 4$, and the number of data points associated to each component are as follows:

- Component 1 has 200 data points;
- Component 2 has 0 data points;
- Component 3 has 300 data points;
- Component 4 has 0 data points.

Recall that the first two components were propagated forward from the previous time step, and components 3 and 4 were added in this time-window.

At this stage the last two algorithm steps (trimming and merging) are executed. The trimming phase deletes components to which no (or only a few) data points have been assigned; in our example, component 2 from the previous time-window has died, since it no longer explains any of the observations, and component 4 is eliminated because it was added unnecessarily.

The last step is the merging step. Following our example, if the KL divergence between the remaining two components, 1 and 3, is smaller than the threshold L_{KL}, then the initial state distributions and transition matrices of these two components are deemed too similar, and so they will be merged, resulting in a final model with a single component. If the KL divergence between the two components is larger than L_{KL} then the two components are deemed to be sufficiently different that they both remain and propagate forward to the next time-window.

D. On the Convergence of the Algorithm per Time-Window

For a fixed model order, Wu [26] shows that convergence of the EM algorithm to a local maximum is guaranteed by demonstrating that the EM iterations monotonically increase the likelihood. Convergence of the CEM algorithm to a local maximum, in the case of a fixed model order, is shown by Celeux and Goavert [21]. They accomplish this by introducing the *classification maximum likelihood* (CML) criterion,

$$C_2(Z(t), \mathbf{\Theta}(t)) = \sum_{m=1}^{M(t)} \sum_{x_i \in Z^{(m)}(t)} \log\left(\alpha^{(m)}(t) p(x_i|\phi^{(m)}(t))\right) \tag{20}$$

where $Z^{(m)}(t) \subset \mathbf{X}(t)$ is the subset of data in $\mathbf{X}(t)$ which is associated with component m. Then they show that each CEM iteration increases this criterion. Since it is bounded above, it follows that the algorithm converges to a local maximum.

Now we discuss why the Automatic Hard EM algorithm is guaranteed to converge within each time window. Specifically, we explain why the while loop spanning lines 4–18 in Algorithm 1 is guaranteed to converge. First, observe that the inner loop spanning lines 6–12 correspond precisely to the CEM algorithm with fixed model order $M(t)$. Convergence of this inner loop is guaranteed by the results of [21], and the value to which it converges is a local maximum of the CML criterion.

It remains to be shown that the outer loop terminates—specifically, that the conditional in line 4 eventually evaluates to false. Recall that $M(t)$ is the number of components in the mixture model. Within the loop spanning lines 4–18, $M(t)$ is strictly increasing. Specifically, if the condition in line 14 is true then $M(t)$ is incremented in lines 15 and 16, and the condition in line 14 is true if the most recently added component has at least one observation sequence assigned to it. Under the assumption that there are a finite number of observations in each time window, the maximum number of components is also finite; it is at most equal to the number of observations $N(t)$, in which case every component has exactly one datapoint assigned to it. Since $M(t)$ is monotonic increasing and bounded above during the execution of lines 4–18, this portion of the algorithm is guaranteed to converge. The last two steps evaluated in each time window (lines 20 and 21) both strictly decrease the number of components in the mixture model, and so they also are guaranteed to converge. Hence, the Automatic Hard EM algorithm is guaranteed to converge at each time step.

Although the algorithm is deterministic and is guaranteed to converge, we note that there are no guarantees about the quality of the estimate to which the algorithm converges. Recall that the CEM iterations monotonically increase the CML criterion given in (20). The loop spanning lines 4–18 involves repeatedly executing CEM iterations while progressively increasing the model order each time CEM converges. Consequently, lines 4–18 indeed monotonically increase the CML criterion, and so when this loop converges the algorithm is at a local maximum of the CML criterion. However, the trimming and merging steps in lines 20 and 21 will generally result in a decrease of the CML.

VI. Threshold for Merging Similar Components

Before proceeding to numerical experiments, we discuss how to choose L_{KL}, the limit at which two estimated components are merged. Since the parameters of the DTMCs in the mixture model are random variables, the KL divergence between pairs of estimated DTMCs can also be considered a random variable parametrized by the hyper-parameters $\phi^{(m)}(t)$ and $\phi^{(m')}(t)$. The KL divergence in time-window t between two DTMCs is non-symmetric and is defined as

$$D_{KL}(m||m';\boldsymbol{\Theta}(t)) = \sum_{i=1}^{|\Omega|} \pi_i^{(m)}(t) \sum_{j=1}^{|\Omega|} P_{i,j}^{(m)}(t) \log \frac{P_{i,j}^{(m)}(t)}{P_{i,j}^{(m')}(t)}.$$

The KL divergence test in Alg. 1 aims to identify when two estimated component distributions are sufficiently close to be merged into one. This means that the hyper-parameters which govern $\phi^{(m)}(t)$ and $\phi^{(m')}(t)$ should be the same hyper-parameters. In this TVMM setting, this set of *shared* hyper-parameters is either $\{G_\pi, G_P\}$ or $\{\pi^{(m)}(t-1), P^{(m)}(t-1)\}$ for some m.

Due to the complexity of this expression, a closed-form expression for the distribution of the KL divergence function between two random DTMCs is not available. Therefore we propose an experimental evaluation which investigates the distribution of sampled KL divergences from mixture models with a mis-specified model order [27].

A. Simulated dataset 1

In order to perform an experimental evaluation of the distribution of the KL divergence, we first introduce a simulated dataset which will be used in the evaluation. The simulated dataset is created using a random network of 25 nodes generated as a random Delaunay graph: nodes are placed uniformly and independently in the unit square, and edges are obtained from the Delaunay triangulation of these points. The resulting graph is planar, similar to many road networks. An example random Delaunay graph is shown in Fig. 1. This dataset contains 100 time-windows where in each time-window a mixture model (as defined in the generative model of Sec. III) governs the sampled data. We refer to this as Simulated Dataset 1 (SDS1), and it is generated from a mixture-model with a model order that varies between one and three components during the 100 time-windows. The probability of component death in the model is $p_d = 0.1$ and the parameter of the Poisson distribution controlling component birth is $\lambda(t) = 1$ for all t. The mixture model order over the 100 time-windows is shown in Fig. 3(a). Vehicle observations in SDS1 have a path length that is sampled uniformly between 13 and 23.

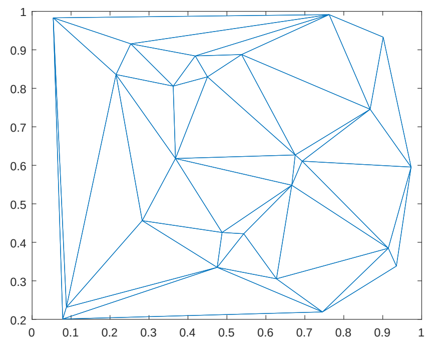


Fig. 1. An example random Delaunay graph with 25 nodes

B. Fitting a mis-specified model

Using the TVMM in SDS1, we sample 50 datasets, each containing observations of 5000 vehicles per time-window. For each of these datasets, we fit a model with exactly $M(t)+1$ components at every time-window by running the EM algorithm [24] with 10 random restarts. We keep the model with the largest likelihood which, for the same model-complexity in each fit, is equivalent to using BIC or AIC as a model selection criterion.

Then, for each best fit model, we compute the KL divergence between all distinct pairs of components in each time-window. The distribution of the KL divergence values is shown in Fig. 2(a), where it can be clearly seen that there are two peaks present. We postulate that the first peak, near zero, corresponds to the over-fit models which should be merged. In other words, one would like to see the peak near zero disappear and only the remaining peak be present, corresponding to estimated components that are well-separated.

We continue to investigate this hypothesis by examining when there is a mis-specified model of order $M(t)+2$ in each time-window. Using the same method as for when $M(t)+1$, the distribution of the KL divergence values is shown in Fig. 2(b). Here we can see the mass of the distribution near

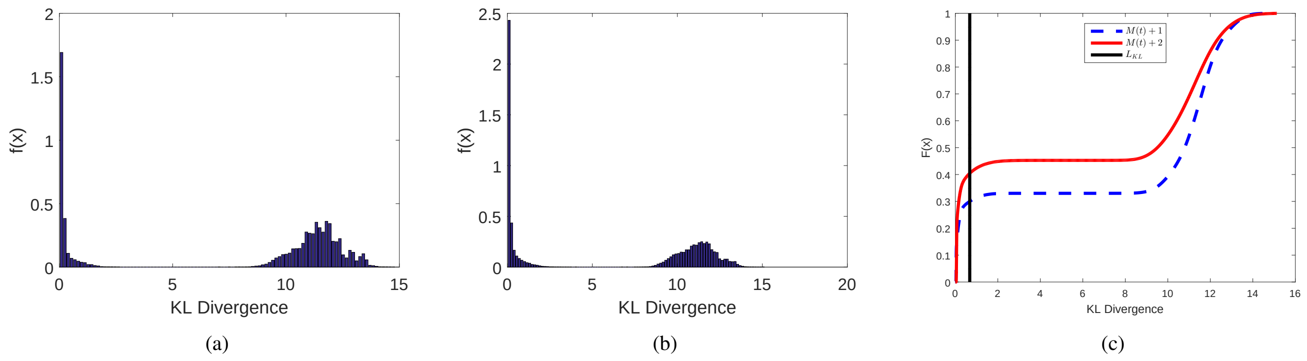


Fig. 2. (a) A histogram density estimate over the sampled KL divergence values for the mis-specified $M(t)+1$ model. (b) A histogram density estimate over the sampled KL divergence values for the mis-specified $M(t)+2$ model.(c) The CDF of the fitted functions in (a) and (b) with the proposed KL divergence threshold $L_{KL} = 0.678$ superimposed at the vertical black line.

zero has indeed grown. To further see this consider Fig. 2(c) which shows the CDF of the distributions in Fig. 2(a) and Fig. 2(b), obtained using kernel density estimates. We can clearly see that the mass in the near zero models has indeed grown under the *more mis-specified* model ($M(t)+2$), which further supports setting L_{KL} to trim this lower peak.

We propose to set the threshold L_{KL} by computing the numerical derivative of the CDF and searching for the first value after the left-most peak where the slope approaches zero (e.g., a slope less than 0.05). For the simulated network this results in a value of $L_{KL} = 0.678$, shown as the vertical, black line in Fig. 2(c). The reason for choosing the threshold as close to 0 as possible is we want to be as sensitive as possible to components which will naturally appear close together and will randomly have a small divergence while removing components which are over-fit and have an unnaturally small KL divergence.

VII. Numerical Experiments

We now continue with a performance evaluation of the automatic hard EM algorithm using both simulated and real data. The first simulated dataset is SDS1 which was described in Sec. VI-A. Two other simulated datasets were also created with a random network of 25 nodes simulated as another realization of a random Delaunay graph. These simulated datasets each also contain 100 time windows, and in each time window observations are sampled from a mixture model. The plot of this second dataset's mixture model order over time is shown in Fig. 3(b). For this network and mixture model, two datasets (SDS2-a and SDS2-b) were generated. The first dataset, SDS2-a, contains sets of vehicle observations with path length distributed uniformly between 5 and 15, and SDS2-b contains sets of vehicle observations with path length distributed uniformly between 15 and 25. The intention is to investigate how the number of observations per-vehicle affects the accuracy. The probability of death and birth are set to the same as SDS1, described in Sec. VI-A. A summary of the simulated dataset parameters is given in Table I.

Comparing the time-varying model orders in Fig. 3, note that SDS1 corresponds to a simpler test case and SDS2, with overall larger model orders, is more challenging. For SDS2-a and SDS2-b we generate 10 independent realizations for each of the eight possible values of $N(t)$ outlined in Table I in order to report average results.

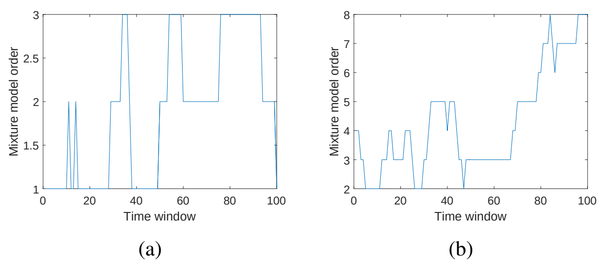


Fig. 3. (a) The mixture model order (number of mixture components) for the first dataset (SDS1) over the 100 time-windows. (b) The mixture model order for the second and third datasets (SDS2-a and SDS2-b) over the 100 time-windows.

We set $L_\alpha = \frac{2}{N(t)}$ so that we only delete components where two or less vehicles gets assigned after the hard EM phase. For these simulations we want to see all possibly relevant components which have been estimated. For SDS2-a and SDS2-b, the same analysis as SDS1 was performed in order to choose the L_{KL} threshold (see Sec. VI), and we conclude that a threshold of $L_{KL} = 0.4$ is appropriate for this network.

A. Evaluation of the Algorithm Performance

Evaluation of Alg. 1 was performed by examining both the estimated number of components and the ℓ_1 error of the model component parameters.

Dataset SDS1 had a perfect model-order fit for all trials, so we focus on SDS2 where we have non-perfect fits. The estimate for the model order of SDS2-a is shown in Fig. 4 for 1000 vehicles per time-window in one of the 10 random trials. The other fitted models of SDS2-a and SDS2-b also showed that the number of estimated components is not a perfect fit and varies from what is shown in Fig. 4 (i.e. the error in the estimated number of components is not the same through all the different dataset sizes). We attribute this increase in the error of the number of mixture components to two sources. First, the model order complexity can be quite large (up

TABLE I
SUMMARY OF THE PARAMETERS OF THE THREE SIMULATED DATASETS SDS1, SDS2-A, AND SDS2-B. THE SET $\mathcal{N} = \{100, 200, 500, 1000, 2000, 5000, 10000, 20000\}$.

Dataset	SDS1	SDS2-a	SDS2-b
Model complexity	Fig. 3(a)	Fig. 3(b)	Fig. 3(b)
Network size ($\lvert\Omega\rvert$)	25	25	25
Number of observations in each time window ($N(t)$)	$N(t) \in \mathcal{N}$	$N(t) \in \mathcal{N}$	$N(t) \in \mathcal{N}$
Number of realizations	1	10	10
Observation sequence path length	$\sim$ Uniform$(13, 23)$	$\sim$ Uniform$(5, 15)$	$\sim$ Uniform$(15, 25)$
L_{KL}	0.678	0.4	0.4
L_{α}	$2/N(t)$	$2/N(t)$	$2/N(t)$

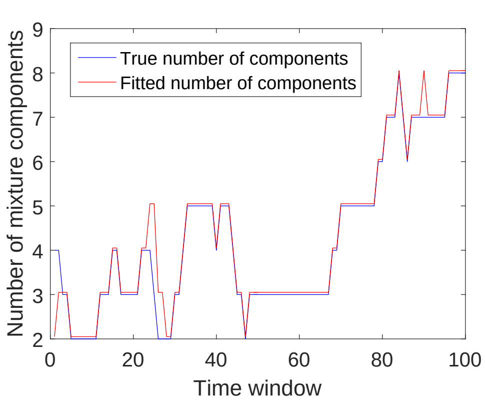


Fig. 4. An example estimated number of components versus the true number of components over 100 time windows for the dataset in SDS2-a with 1000 vehicles per time-window.

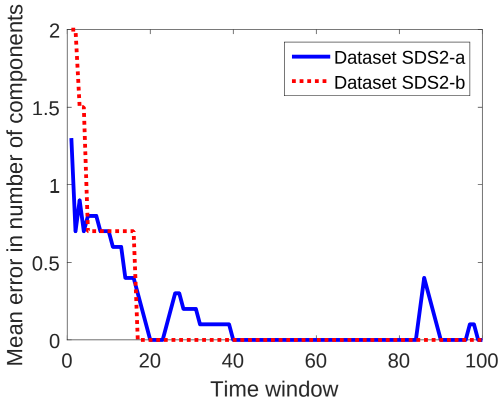


Fig. 5. The average absolute error in the number of estimated components for SDS2-a and SDS2-b using 5000 vehicles per time-window and averaged over 10 trials

to 8 mixture components). Also, since the average vehicle observation length is smaller than in the first dataset, misclassification of vehicles to the wrong component (or creation of false components) is more likely.

We also investigate how the algorithm performs when selecting the number of mixture model components for varying lengths of the vehicle observation sequences. In Fig. 5 we see that, after an initial transient period, the algorithm maintains the estimate of the mixture model order more easily when there are longer observation sequences (as in SDS2-b). Selecting the correct number of components is an important step in the estimation of the overall mixture model, but we want to further investigate the error in the model's parameter estimates as well.

Next we examine the fidelity of the estimated model parameters. We quantify the accuracy using two approaches discussed below: the marginal ℓ_1 error, and the component-wise ℓ_1 error.

1) Marginal ℓ_1 error: The first approach to error quantification is based on the true and estimated marginal models, computed from the mixture model as

$$P^{(\text{marg})}(t) = \sum_{m=1}^{M(t)} \alpha^{(m)}(t) P^{(m)}(t) \tag{21}$$

$$\text{and } \pi^{(\text{marg})}(t) = \sum_{m=1}^{M(t)} \alpha^{(m)}(t) \pi^{(m)}(t). \tag{22}$$

As a benchmark, we estimate a one-component model using the maximum a posteriori (1-MAP) estimate of the marginal distribution which is

$$\hat{p}_{j,k}^{(\text{1-MAP})}(t) = \frac{n_{j,k} + \hat{p}_{j,k}^{(\text{1-MAP})}(t-1)}{1 + \sum_{k=1}^{|\Omega|} n_{j,k}} \tag{23}$$

$$\text{and } \hat{\pi}_j^{(\text{1-MAP})}(t) = \frac{n_j + \hat{\pi}_j^{(\text{1-MAP})}(t-1)}{1 + \sum_{j=1}^{|\Omega|} n_j}\ , \tag{24}$$

where $n_{j,k}$ is the total number of observed transitions from j to k and n_j is the number of observations of starting state j. For the first time-window ($t = 1$), we use the same global prior as with our proposed model.

We measure accuracy in terms of the ℓ_1 error,

$$\ell_1(\hat{p}, \hat{\pi}, p, \pi) = \sum_{j=1}^{|\Omega|} \sum_{j=1}^{|\Omega|} |\hat{p}_{j,k} - p_{j,k}| + \sum_{j=1}^{|\Omega|} |\hat{\pi}_j - \pi_j|, \tag{25}$$

where $\hat{p}$ and $\hat{\pi}$ are the estimated marginal transition matrix and initial state distribution, and p and π are the true marginal transition matrix and initial state distribution. This reduces the error to a single scalar value and has the benefit that it is straightforward to calculate even when the model order is not estimated correctly.

Recall that the total variation distance between two probability densities is at most two. The metric defined in equation (25) is the sum of $|\Omega|+1$ total variation distances (one for the initial state distribution, and one for each row of the transition matrix). In all of the simulated datasets there are $|\Omega| = 25$ states (i.e., sensors), so $\ell_1(\hat{p}, \hat{\pi}, p, \pi) \leq 52$.

Fig. 6 shows the ℓ_1 errors among the marginal distributions on all three simulated data sets, for estimates generated using the proposed mixture model as well as using a single-component Markov chain estimated using the MAP approach described above. In general, the mixture model gives a significantly lower error than using a simple 1-component Markov model. Note that the time windows where the errors of the two models coincide in SDS1 occur when the true model order is $N(t) = 1$. The ℓ_1 marginal error of the mixture estimate also appears to be correlated with the model order. In these simulations, the total number of observations (vehicles) per time-window is constant, so when the model order is higher there are fewer observations per component, resulting in a higher error. Also observe that the steady-state marginal error of the mixture model is slightly higher for SDS2-a than it is for SDS2-b. This is expected, since there are fewer observations per vehicle, on average, in SDS2-a.

We next examine how the error in the marginal distribution changes as a function of the number of observed vehicles, $N(t)$. Fig. 7 shows that the error of the mixture model goes to zero significantly faster than that of the 1-MAP as $N(t)$ increases on the SDS2-b dataset. Datasets SDS1 and SDS2-a exhibit the same behaviour, only differing in the scale of the plots, and they are therefore not shown here.

2) Component-wise ℓ_1 errors: When the model order is estimated correctly, we also examine the average component-wise errors. In this case, the labels (indices) of components in the true and estimated models may not coincide, so to assess the component-wise error we first perform a matching as follows. Let $(\pi^{(m)}, P^{(m)})$ denote the model parameters associated with component m of the true mixture model, and let $(\hat{\pi}^{(m')}, \hat{P}^{(m')})$ denote the parameters of component m' of the estimated mixture model. Note that we have dropped the time index t to simplify the notation. Suppose that there are M mixture components overall, and let $\mathbb{S}_M$ denote the set of permutations of M elements. We first find the permutation $\sigma^\star$ that minimizes the total ℓ_1 error,

$$\sigma^\star = \arg\min_{\sigma \in \mathbb{S}_M} \sum_{m=1}^{M} \ell_1 \left(\hat{P}^{(\sigma(m))}, \hat{\pi}^{(\sigma(m))}, P^{(m)}, \pi^{(m)} \right).$$

The solution $\sigma^\star$ is computed using the Hungarian algorithm. Then we compute the average component-wise ℓ_1 errors for the initial state distributions, $\pi^{(m)}$, the transition matrices, $P^{(m)}$, and mixture weights, $\alpha^{(m)}$, separately. For example, for the initial state distribution, the average component-wise ℓ_1 error is

$$\frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{|\Omega|} \left| \pi_i^{(m)} - \hat{\pi}_i^{(\sigma^\star(m))} \right|.$$

Note that it is not straight-forward to assess component-wise errors when the model order has not been accurately estimated. For example, if the estimated model has more components than the true model, it may be that two or more components in the estimate should be associated with a single component in the true model, or alternatively, it may be that one component in the estimated model should be partially associated with two or more components in the true model. Hence, we only report component-wise errors for time-windows where the model order has been correctly estimated.

Fig. 8 shows the average component-wise errors for the SDS2-a dataset with $N(t) = 5000$ observations. The same trends were observed for the other two datasets so they are not shown here. In general, we observe that the average component-wise errors also correlate well with the model order. As the model becomes large (e.g., time-windows 80–100), the average component-wise ℓ_1 also becomes large, suggesting that the estimated components do not precisely match those of the true model. This may again be attributed to the fact that, in the simulation, when the model order is higher there are fewer observations per component. However, as we will see in the experiments on real data reported below, this error does not necessarily imply that predictions made using the estimated mixture are inaccurate. We also computed the standard deviation of component-wise errors, but these were all orders of magnitude smaller than the mean errors so they are not shown in the figure.

B. Computation Time

The real benefit of Algorithm 1 compared to some of the non-deterministic approaches, such as Gibbs sampling in [7], is the computation time and complexity. Even with 20,000 observations in each time-window, the computation time never exceeds 30 seconds per iteration on a laptop with a quad-core 2 GHz Intel Core i7 processor and 8 GB of RAM. This makes this algorithm practical for real-time applications where estimation of a complex mixture model is necessary without having unlimited time or computational resources available. In addition to computation time, the algorithm is deterministic and requires no random restarts or bootstrapping.

C. Comparison to Gibbs Sampling

The MCMC method described in Stephens [7] was also applied to the generative model in Section III. This algorithm was run on a single time-window ($t = 35$) of SDS1 where the true mixture has 3 components and 10,000 vehicles are observed. The method was run for 5,000 burnin iterations and 50,000 sampling iterations. The best candidate model, with the correct number of components, chosen in the sampling had a ℓ_1 marginal error of 20.68, which is significantly larger than the ℓ_1 error of 0.8444 obtained using Alg. 1. Upon further inspection, we observe that the Gibbs sampler appears to sample poor candidate models due to the high-dimension of the parameter space (we conclude this because most sampled models are immediately rejected using the approach described in [7]). Consequently the estimated model order alternates between two and three specific mixture components and does not explore the space of possible candidate components well. Based on these observations we conclude that the class of MCMC models will fail to estimate traffic models with such a high-dimensional and time-varying parameter space. Therefore we do not consider this approach further.

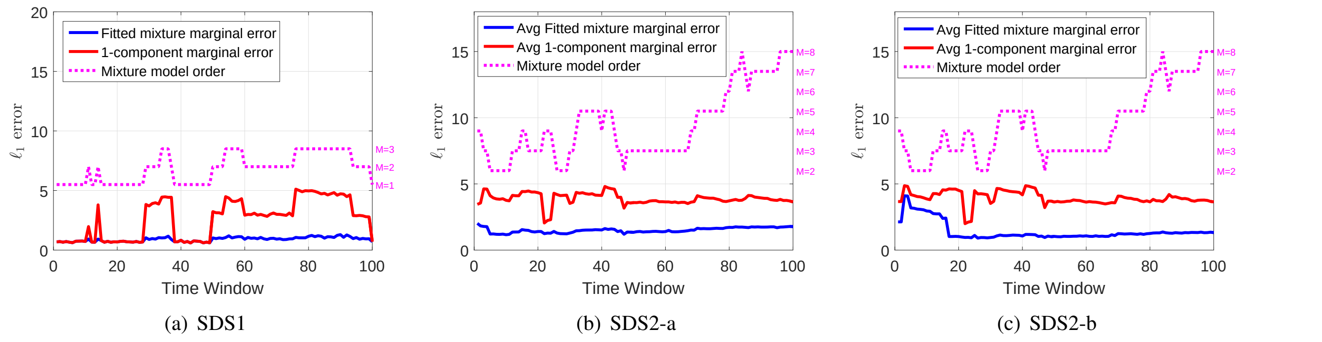


Fig. 6. (a) The ℓ_1 error of the estimated marginal Markov chain distribution versus the true underlying distribution for SDS1 with 5000 vehicles per time-window. Also shown is a MAP estimated marginal distribution and the mixture model superimposed. (b) The ℓ_1 error of the estimated marginal Markov chain versus the truth with SDS2-a and 5000 vehicles per time-window. (c) The same plot as (b) for dataset SDS2-b.

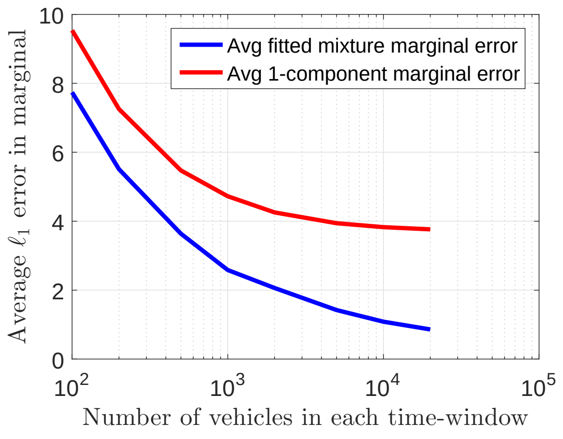


Fig. 7. The average ℓ_1 error in the estimate of the marginal distribution estimated using the proposed method and a MAP estimate for SDS2-b with a varying number of vehicles in each time-window. The number of vehicles is on a log-scale.

D. Real Data Analysis

Next we run Alg. 1 on a real LPR dataset provided by a corporate partner. The dataset is a collection of LPR reads from a network of 20 LPR cameras over a period of 31 days and 2 hours. We take each time-window to be an hour long so there are a total of 746 time-windows; the effect of time-window length is discussed further in Section VII-F. In every time-window we create observation sequences for the vehicles which appear in that window. From the data in each time-window, we estimate the mixture model using Alg. 1. The threshold parameters L_α and L_{KL} are set to

$$L_\alpha = \frac{2}{|X(t)|} \text{ and } L_{KL} = 0.12 \tag{26}$$

so that the trimming of weak components deletes any component with less than two vehicles assigned to it. We estimate the KL divergence threshold, $L_{KL} = 0.12$, by running Alg. 1 once with $L_{KL} = 0$ so that we essentially over-fit as much as possible. We then compute the KL divergence between all pairs of components and show the density of the KL divergences in Fig. 9(a). Here we can clearly see another peak near 0, which we hypothesize is appearing due to the same reason as in the simulated case. Those components with very small divergence are being over-fit and should be merged into other existing components. In Fig. 9(a) we draw the red line to demonstrate where we place the L_{KL} threshold to trim off that lower peak, which is at 0.12. We then plot the CDF of the resulting KL divergences when Alg. 1 is run with $L_{KL} = 0.12$ which demonstrates that the lower peak is no longer present. There is no ground-truth distribution for this dataset, so validating the KL divergence threshold beyond this is not feasible.

The estimated number of components is shown in Fig. 10(a) where we see that the estimated mixture model is quite complex, having up to 18 components. However upon closer inspection, we observe a cyclic behavior, as shown in Fig. 10(b), where the model is most complex at rush-hour in the morning and afternoon and least-complex during the late evening and early morning. This agrees with one's intuition about traffic simply due to the fact that at rush-hours, the volume of traffic traversing the roads will be greatest therefore the largest number of paths are likely to be observed.

E. Mixture Model Predictive Ability

There is no ground-truth model available for the real dataset. To assess the quality of the estimated mixture model, we consider the following prediction task. In each time-window, we split the observations (i.e., per-vehicle sequences) into a training set and a test set. Specifically, 20% of the observations are randomly (uniformly) sampled to form a test set for the time-window, and the remaining 80% of the samples form the training set. For each time-window, we use the training data to update the estimated model parameters. Then, for each test point (the sequence of observations of a single vehicle), the task is to predict the last state where the vehicle will be observed given all observations but the last one; i.e., for a test sequence $(x_1, x_2, \ldots, x_n)$, we must predict the value of x_n given $x_1, \ldots, x_{n-1}$. We allow the prediction to be in the form of a probability density over the set Ω, which can also be viewed as the parameters of a multinomial distribution.

We again compare the performance of the proposed mixture model, fit using the automatic hard EM algorithm, with that of a single-component (1-MAP) model. For the 1-MAP model, the prediction is simply given by row x_{n-1} of the estimated

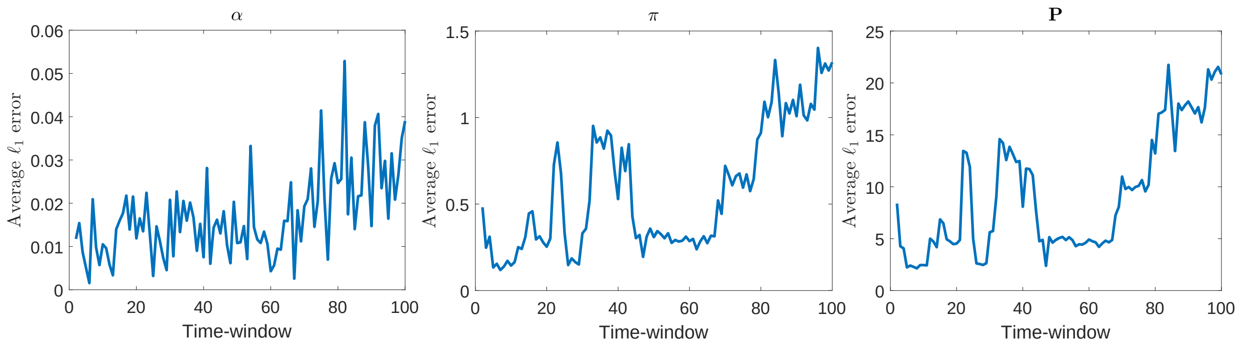
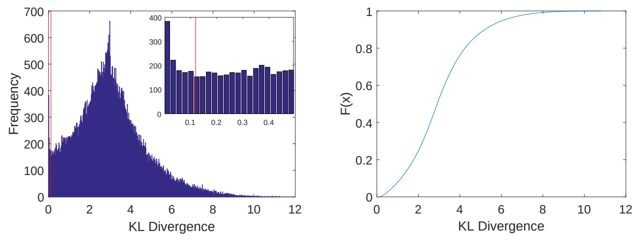
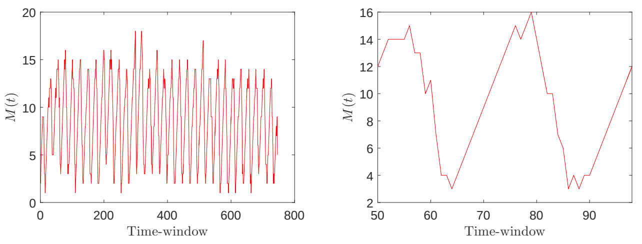


Fig. 8. Average component-wise ℓ_1 errors of the mixture weights $\alpha^{(m)}$, initial state distributions $\pi^{(m)}$, and transition matrices $P^{(m)}$, on SDS2-a for $N(t) = 5000$ observations per time-window.



(a) (b)

Fig. 9. (a) A histogram of the distribution of the KL divergence values when setting the KL divergence threshold to 0 ($L_{KL} = 0$). The red line ($y = 0.12$) denotes the proposed threshold value, L_{KL}, to remove the lower-component, over-fit estimated DTMCs. (b) The CDF of the distribution of the KL divergences in Fig. 9(a) once we impose our threshold of $L_{KL} = 0.12$. Note that there is no “hump” near zero meaning we have removed unnecessary, over-fit components.



(a) (b)

Fig. 10. (a) The estimated number of components for the real LPR dataset for the 746 time-windows. (b) A portion of the estimated number of components on the real dataset in Fig. 10(a) zoomed in to view the cyclic behavior.

transition matrix. For the mixture model, we use the first $n-1$ observations of each test vehicle to determine which of the estimated mixture components best explains the observations, and then the prediction is given by row x_{n-1} of the transition matrix of that specific component. As a baseline, we also compare with a naive, uniform prediction that places mass $1/|\Omega|$ on every sensor (i.e., every state in Ω).

The ideal predicted distribution would have unit mass on position x_n, and zero everywhere else. To measure the quality of the predictions given by the mixture model and 1-MAP, we compute the multinomial log-loss

$$LL_{MN}(p^{(pred)}) = -\frac{1}{N_{\text{test}}(t)} \sum_{i=1}^{N_{\text{test}}(t)} \sum_{j=1}^{|\Omega|} y_{i,j} \log\left(p_{i,j}^{(pred)}\right),$$

where $N_{\text{test}}(t)$ is the number of testing samples in time-window t, $y_{i,j}$ is the indicator that the final observation of the ith test vehicle was made at sensor j, and $p^{(pred)}$ is the predicted probability that the ith test vehicle was last observed at sensor j, given the initial observations of that vehicle. The multinomial log-loss of a particular training instance is equal to zero if the prediction puts all of its mass on the correct sensor, and it increases as the predicted probability of the correct state decreases.

Fig. 11 shows the multinomial log-losses for each time window. The boxplots show that the proposed model (“Mixture fit”) has a lower median and much lower spread of LL_{MN}, and thus its predictions are closer to the truth for a larger volume of the datapoints than predictions made by the single-component MAP Markov model (“1-component fit”). This is likely due to the proposed model providing more accurate estimates for vehicles on less-frequently observed routes; these can be represented as low-weight mixture components in the mixture model, while the corresponding data gets washed out in the 1-component model. The vast majority of predictions made by the mixture model have lower loss than the naive prediction, whereas the upper whisker of the 1-component fit lies above the naive loss line.

Fig. 12 shows performance as a function of the number of vehicle observations ($n-1$ in the discussion above) available before making a prediction. The mixture model clearly shows a benefit when making predictions from a few initial observations. When the number of observations of a vehicle gets large, both the mixture and 1-component fits are comparable.

F. Time-Window Length Selection

The selection of an appropriate time-window length when estimating the time-varying mixture model is critical to the algorithm’s performance. If the time-window length is set too large, then the true underlying distribution of traffic may change significantly within one time-window. On the other hand, if the time-window is too short than there may not be a

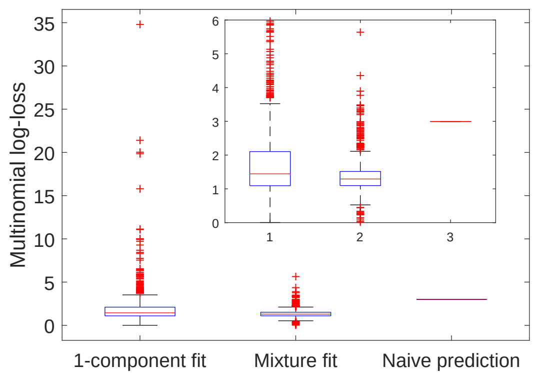


Fig. 11. The Multinomial log-loss using three different estimates: (1) A standard 1-component MAP estimate, (2) Our proposed mixture model and Automatic Hard EM algorithm, and (3) Simply assigning $\frac{1}{n}$ to all trnasition matrix weights (i.e. naive prediction). This plot includes a zoomed-in region in the top right to better see the boxes comparison.

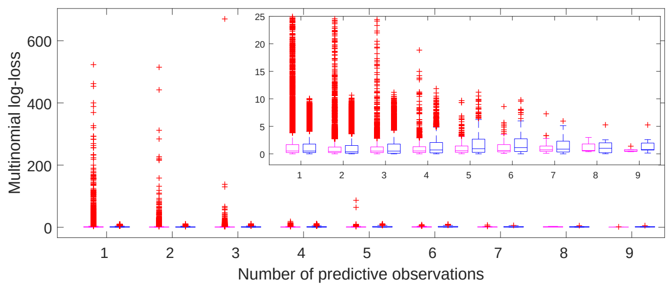


Fig. 12. The individual Multinomial log-losses per-vehicle versus the number of observations available to compute the predictive multinomial density. In each group, the left box plot (cyan) is the prediction of the 1-component model and the right (blue) is our proposed mixture model. The upper corner shows a zoomed-in portion of the boxplots focusing on the range of multinomial log-loss between 0 and 25.

sufficient number of observations available to estimate model parameters within each window.

Towards selecting a reasonable time-window length for the real data experiments, we explore some related characteristics of this dataset. We begin by organizing the dataset by vehicle, and segment the per-vehicle observations into individual trips. Recall that the entire dataset spans a period of one month, and many of the vehicles appearing in the dataset are observed multiple times during that period. The maximum distance between two sensors is roughly 40 km. Assuming a minimum speed of 10 km/hr, the maximum time it would take observe a vehicle at one of these sensors after it was observed at the other is four hours. Hence, whenever the consecutive observations of a vehicle are more than four hours apart we consider it the beginning of a new trip.

The number of observations per trip and the duration of each trip are shown in Fig. 13. In total there are 27,7798 observed vehicle trips. The median number of observations per trip is 3.5183, and the median trip duration is 36.0585 minutes. The red line in Fig. 13(b) indicates a duration of 60 minutes, which is the value we use for the trip-window duration. In this case, 86.08% of the observed trips have a duration of 60 minutes or less.

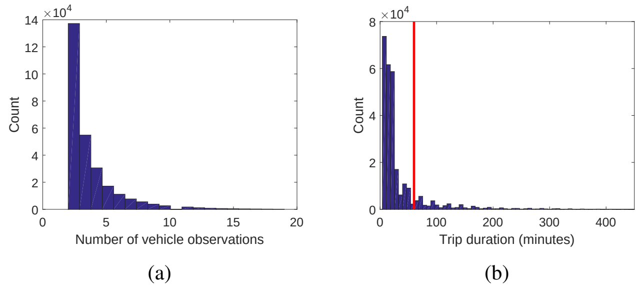


Fig. 13. (a) Histogram of the distribution of number of vehicle observations per trip in the real dataset. Note that vehicles that are only observed once are excluded from the histogram for clarity. (b) Histogram of trip durations (the time from the initial observation of a vehicle to the final observation in a single trip). The superimposed red line denotes the 60-minute (1 hour) mark which we have used as the window-size for the results reported in the paper.

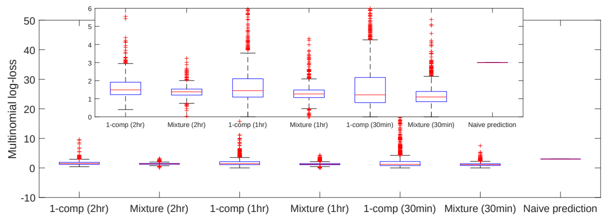


Fig. 14. Boxplots of the multinomial log-loss for three different time-windows (30 minutes, 1 hour, 2 hours). The first, third, and fifth plots are using the single-component MAP estimated DTMC models while the second, fourth, and sixth plots are using the time-varying mixture of DTMCs proposed in this paper. The last plot is a “naive” fit using a single DTMC with $1/|\Omega|$ in all cells of the transition matrix. For clarity, a zoomed-in portion of the boxplots is provided.

We also repeat the prediction experiment described in Sec. VII-E with models trained using both 30 minute and 2-hour time windows. For each scenario, we estimate a time-varying mixture of Markov chains using the proposed approach, as well as a time-varying single-component Markov chain estimated as the MAP with the model from the previous time window as the prior. The results of these trials are shown in Fig. 14. Observe that there is a slight increase in the variability of the performance (multinomial log-loss) of the proposed mixture model as the time-window length decreases. However there is a much greater increase in the variability (as evidenced by the increase in the whisker locations) in the single-component estimated model. This suggests that the mixture model is less sensitive to the choice of time-window length than a single-component model.

VIII. Conclusion

This article proposes a novel approach to fitting time-varying mixture models, with an specific application to discrete-time Markov chains for the problem of traffic estimation. The Markov model specified in this paper enables complex, time-varying, network models to be modelled using AVI data. We then go on to define an algorithm to estimate the parameters of this time-varying mixture model and specifically apply it to AVI data. The performance of the algorithm is assessed using simulated data and data from a real deployment. The results indicate that the proposed model provides more accurate predictions of vehicle trajectories, especially when the prediction is made using fewer initial observations.

A. Future Work

1) Errors in Data Source: A future avenue of research stemming from this work is the modelling of errors in the incoming data stream. We currently assume that the observed vehicle IDs are error free. For license plates, the errors which can occur are insertions, deletions, and replacements of characters in the license plate. A probability distribution over the possible errors could be postulated and then transition count matrices for each vehicle could be updated based on the likelihood of the recorded license plate. This would allow for a natural extension to model the input errors in the recording of license plates.

2) Adapting the Time-Window Length: Another extension would be to dynamically adapt the time window length. To motivate why, consider normal traffic in an urban environment. Traffic patterns typically have two very high volume periods, during the morning and afternoon rush-hours. At these times, the largest variation in traffic patterns are observed (see, e.g., the real-data example in Section VII-D). On the other hand, traffic patterns observed in the middle of the night are likely remain unchanged for many hours since few vehicles are travelling at that time.

Therefore the mixture models which are estimated to be governing the current traffic patterns may be valid for longer or shorter times based on the observed traffic patterns and volumes at different times of the day. One approach could be to run a sequential hypothesis test in parallel to the estimation scheme, to detect when the current model no longer adequately fits or explains incoming observations. When the fit to current data becomes sufficiently poor then an adaptive model update would be triggered.

References

[1] R. Smeed, "Road pricing: the economic and technical possibilities," British Transport and Road Research Laboratory, Tech. Rep., 1964.

[2] J. Ren, X. Ou, Y. Zhang, and D. Hu, "Research on network-level traffic pattern recognition," in *Proc. on Intelligent Trans. Sys.* IEEE, 2002, pp. 500–504.

[3] S. Lawlor, T. Sider, N. Eluru, M. Hatzopoulou, and M. G. Rabbat, "Detecting convoys using license plate recognition data," 2016, to appear in *IEEE Trans. on Signal and Information Process. over Networks*.

[4] M. Nanni, R. Trasarti, B. Furletti, L. Gabrielli, P. V. D. Mede, J. D. Bruijn, E. D. Romph, and G. Bruil, *Transportation Planning Based on GSM Traces: A Case Study on Ivory Coast*. Springer International Publishing, 2014, pp. 15–25.

[5] Maine Turnpike Authority. (2016, July) Welcome to the Maine turnpike's E-ZPass program. [Online]. Available: https://ezpassmaineturnpike.com/EZPass/

[6] F. Caron, M. Davy, and A. Doucet, "Generalized Polya urn for time-varying Dirichlet process mixtures," in *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, UAI*, July 2007.

[7] M. Stephens, "Bayesian analysis of mixture models with an unknown number of components- an alternative to reversible jump methods," *The Annals of Statistics*, vol. 28, pp. 40–74, 2000.

[8] M. Abhijith, P. Ghosh, and K. Rajgopal, "Multi-pitch tracking using Gaussian mixture model with time varying parameters and grating compression transform," in *2014 IEEE Inter. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, May 2014, pp. 1473–1477.

[9] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. on Automatic Control*, vol. 19, no. 6, pp. 716 – 723, 1974.

[10] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[11] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[12] A. Corduneanu and C. M. Bishop, "Variational Bayesian model selection for mixture distributions," in *Intl. Conf. on Artificial Intelligence and Statistics*. Morgan Kaufmann, Jan. 2001, pp. 27–34.

[13] M. A. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, March 2002.

[14] J. Chen and A. Khalili, "Order selection in finite mixture models with a nonsmooth penalty," *J of the American Statistical Association*, vol. 103, no. 484, pp. 1674–1683, 2008.

[15] J. Verbeek, N. Vlassis, and B. Krose, "Efficient greedy learning of Gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.

[16] M. G. Singh and H. Tamura, "Modelling and hierarchical optimization for oversaturated urban road traffic networks," *International Journal of Control*, vol. 20, pp. 913–934, 1974.

[17] B. S. Kerner, S. L. Klenov, and D. E. Wolf, "Cellular automata approach to three-phase traffic theory," *Journal of Physics A: Mathematical and General*, vol. 35, no. 47, pp. 9971–10 013, November 2002.

[18] A. Peterson, "The origin-destinaton matrix estimation problem - analysis and computations," Ph.D. dissertation, Linköping University, Linköpings universitet, SE-601 74 Norrköping, Sweden, 2007.

[19] N. V. D. Zijpp, "Dynamic OD-matrix estimation from traffic counts and automated vehicle identification data," *Tranprn Res. Record: J. of the Transprn Res. Board*, vol. 1607, pp. 87–94, 1997.

[20] S. Lawlor and M. G. Rabbat, "Estimation of time-varying mixture models: An application to traffic estimation," in *2016 IEEE Wrksp. on Stat. Signal Process. (SSP)*, June 2016.

[21] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Computational Statistics & Data Analysis*, vol. 14, no. 3, pp. 315–332, 1992.

[22] S. Lawlor, T. Sider, N. Eluru, M. Hatzopoulou, and M. Rabbat, "Detecting convoys using license plate recognition sensors," *IEEE Trans. Signal and Information Processing over Networks*, vol. 2, no. 3, pp. 391–405, Sep. 2016.

[23] S. Kotz, N. Balakrishnan, and N. Johnson, *Continuous Multivariate Distributions. Volume 1: Models and Applications*. Wiley, 2000, vol. 1.

[24] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. of the Royal Statatistics Society Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.

[25] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297.

[26] C. J. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.

[27] H. White, "Maximum likelihood estimation of misspecified models," *Econometrica*, vol. 50, no. 1, pp. 1–25, 1982.