Sparse travel time estimation from streaming data

Saif Eddin Jabari1, Nikolaos M. Freris, Deepthi Mary Dilip
Division of Engineering, New York University Abu Dhabi, Saadiyat Island, P.O. Box 129188, Abu Dhabi, U.A.E.

Abstract
We address two shortcomings in present travel time distribution estimation methods, which specifically apply to the case of congested stop-and-go traffic. The first shortcoming is related to the determination of the number of modes, which can change from one location in a traffic network to another as well as by time of day. The second shortcoming is the wide-spread use of mixtures of Gaussian probability densities that can assign positive probabilities to negative travel times and offer too little flexibility because of their symmetric shape. To address these issues, this paper develops a sparse kernel density estimation (KDE) technique using asymmetric kernels. Sparse estimation is applied to avoid the need for a predefined number of mixture components. The use of asymmetric Gamma kernels ensures nonnegative supports while also providing increased flexibility in the shapes of the distributions. Most notably, we derive a novel kernel based on Mittag-Leffler functions: the kernel generalizes the Gamma kernel and provides enhanced fitting flexibility. We present an adaptive approach to simultaneously infer the locations and scale parameters of the kernels. In order to accommodate within-day variability of travel time distributions, and further allow for a real-time implementation of the proposed methodology (i.e., fast computations on streaming travel time data), we introduce a recursive algorithm which efficiently updates the fitted distribution whenever new data become available. Experimental results using real-world travel time data illustrate the efficacy of the proposed methods. Our experiments attest to the superiority of Mittag-Leffler kernels over standard Gamma kernels and Gaussian kernels in terms of both estimation accuracy and parsimony. Additionally, the use of recursive algorithms leads to substantial computational savings when operating on streaming datasets.

Keywords: travel time reliability, sparse modeling, kernel density estimation, adaptive estimation, Gamma kernel, recursive estimation, streaming data, Mittag-Leffler function.

1. Introduction

Travel times are among the prime measures of traffic and travel performance in congested road networks. They can vary dramatically from one location to another and by time of day. This variability, when taken into account by travelers, can play a vital role in deciding which route to choose. It is also a key factor in assessing the resilience and reliability of network road segments.

Quantifying variability involves the accurate estimation of travel time probability distributions. In signalized urban network settings, variability in travel times can be traced to uncertainty about network supplies and demands (Du et al., 2012), the effect of traffic signals, and the characteristics of route choice (Ramezani and Geroliminis, 2012). The interrupted nature of traffic contributes significantly to the variability in urban travel times and is the key factor contributing to the multi-modality of their distributions (Hunter et al., 2013b; Li et al., 2014).

Travel time distributions have been extensively studied. Along expressways, travel time distributions are typically well captured by unimodal functions such as the lognormal distribution (Richardson and Taylor, 1978; Rakha et al., 2006; Pu, 2011; Arezoumandi, 2011), the Gamma distribution (Polus, 1979) and the Weibull distribution (Al-Deek and Emam, 2006). However, in urban settings

---

1Corresponding author. E-mail: sej7@nyu.edu. URL: http://engineering.nyu.edu/people/saif-eddin-jabari.
with traffic signals, multi-modality tends to be the case (Guo et al., 2010). Along high-speed arterials, travel times are well represented by bi-modal distributions: one mode for vehicles that arrive during a green wave and another for those that encounter a red signal. In congested networks with spillover dynamics, one tends to observe more than two modes (Hofleitner et al., 2012b; Yang et al., 2014; Rakha et al., 2011). To account for this multi-modality of travel time distributions, researchers have commonly resorted to the use of Gaussian mixtures (Guo et al., 2010; Feng et al., 2014). To accommodate asymmetrically distributed travel times (most commonly observed in congested urban traffic), skewed component distributions, e.g., the Gamma and lognormal distributions have also been adopted (Kazagli and Koutsopoulos, 2013). A non-parametric approach utilizing negative exponential kernels was presented in Rahmani et al. (2015). Ji and Zhang (2013) developed a hierarchical Bayesian mixture travel time model to model the interrupted nature of traffic flow in urban roads.

The Expectation Maximization (EM) algorithm (Redner and Walker, 1984) and Bayesian techniques have been widely used learning approaches. The Bayesian approach applies Markov chain Monte Carlo (MCMC) procedures to estimate the values of selected parameters, and is known to be computationally demanding (Chen et al., 2014). As a result, the majority of prevalent methods utilize the EM algorithm which is, nonetheless, primarily suitable for Gaussian mixtures. The main shortcoming of Gaussian mixtures stems from the symmetry of the Gaussian density: when the underlying distributions are skewed (as is commonly the case with travel times), typically a large number of mixture components is needed for accurate approximation by a mixture of Gaussian (more generally, symmetric) densities. Another disadvantage that comes with adopting Gaussian mixture components is that the resulting probability distribution has negative travel times in its support.

A major limitation of traditional mixture modeling is that it requires a priori knowledge of the number of mixture components. Since the number of components is usually unknown in real-world problems, the issue of determining the right number while simultaneously estimating the associated model parameters may become quite challenging. In situations where the number of components reflects the number of clusters in the data, nonparametric methods have been proposed as an alternative for selecting the number of clusters. In the case of travel times, these clusters represent the vehicle groups that encounter similar stop-and-go conditions; these can vary with location, from day-to-day, and with time of day, as a result of varying traffic conditions. The problem of determining the optimal number of components has been addressed by researchers in various fields through sparse kernel density estimators using support vector machines (Mukherjee and Vapnik, 1999), penalized histogram difference criteria (Lin et al., 2013) and orthogonal forward regression (Chen et al., 2004, 2008).

The variability of travel times throughout the day presents a need for real-time estimation techniques from streaming measurements. Hofleitner et al. (2012a) fuse streaming GPS probe data with model parameters learned from historical data using the EM algorithm. To tackle the issue of large amounts of data (and avoid the need for re-estimating the bulk of parameters), the EM algorithm is run offline with the model parameters updated periodically, for instance every week or every month. Wan et al. (2014a), on the other hand, adopted an online EM algorithm. Notwithstanding, these two methods may not capture within-day variations since they both use all the data that is available (including data for different periods and different traffic characteristics, e.g., they may use PM peak period data to estimate AM peak travel times). While the use of a time window in conjunction with the online EM algorithm (Hunter et al., 2013a) may tackle this problem, the issue of predefining the number of components still remains. Since the number of components can change with every successive estimation, the ability to automatically infer the number of mixture components—whilst fitting the mixture model parameters—from streaming measurements becomes critical in order to maintain consistency with traffic conditions on the ground.

This paper proposes a kernel density estimation approach for real-time estimation of travel time distributions. Our analysis substantially extends and expands our previous work (Dilip et al., 2017). We overcome the issues mentioned above pertaining to Gaussian kernels by replacing them with Gamma kernels, which have positive support and can be asymmetric (i.e., they are more flexible). Going a step further, we devise a richer set of kernels (with variable location and scale parameters) that can be combined to capture a wide variety of travel time distributions. Subsequently, the prob-
lem becomes one of choosing those kernels that most closely capture the empirical distributions. To promote parsimony (i.e., fit with as few kernels as possible), we propose the use of an $\ell_1$-regularizer, which is known to promote sparsity (Tibshirani, 1996). We proceed to demonstrate how to apply this methodology on streaming data by a) updating the inputs whenever a new travel time sample or batch of travel times arrives and b) warm-starting the numerical optimization; this allows for a very fast update of the fitted distribution and renders the proposed approach amenable to an online implementation subject to stringent real-time constraints.

The use of Gamma kernels presents specific analytical challenges: the first one is known as boundary bias, which is a known issue when dealing with most (if not all) non-Gaussian kernels. The issue has to do with the dependence of the shape of the distribution on its location. We overcome this difficulty by reversing the roles of location parameter and scale in the kernel. However, this in turn makes it difficult to ensure that the resulting kernel probability densities integrate to unity. We resolve this issue analytically: the resulting remedy needs to be carried out only once, as a processing step off-line, thereby it does not affect the applicability of our proposed methods in real-time.

Furthermore, we present an adaptive version of our methodology. This allows for heterogeneous sets of kernels (in terms of scale). This is done so as to further enhance the parsimony while at the same time boost the fitting accuracy of the resulting distributions. In order to accomplish this, the Gamma kernel function must be generalized to ensure that the densities integrate to unity, when using variable scale parameters. We accomplish this by devising a novel kernel which generalizes Gamma kernels by means of using the Mittag-Leffler function (which can be seen as a generalization of the exponential function); correspondingly, we call this new kernel as the Mittag-Leffler kernel.

In brief, the main contributions of the paper are synopsized as follows:

- We invoke sparse density estimation for learning travel time distributions. Our method adopts a mixture of kernels and uses numerical optimization for optimally selecting the weights based on a histogram of travel times. The proposed approach is very general, in that any kernel function can be used (Gaussian, Gamma, Epanechnikov, bi-square etc.).

- We rigorously analyze the problem of discretizing the Gamma kernel so that the corresponding mixture reflects a valid probability distribution, for any given shape of the Gamma distribution.

- We develop a new kernel, which we call the Mittag-Leffler (M-L) kernel, that generalizes the Gamma kernel. The M-L kernel allows for an adaptive selection, in the sense that any given histogram may be fitted by a mixture of kernels of variable shapes, placed at variable positions.

- We propose methods for online learning from streaming observations. We tackle two real-life scenarios namely: (a) sequential estimation, i.e., enhancing the learning fidelity when new data become available, without the need to re-process the entire dataset and (b) rolling-horizon learning, i.e., a method for tracking time-varying travel time distributions (e.g., for different time frames of a given day, or from day-to-day). Our methods are directly amenable to an online implementation subject to stringent real-time processing requirements, as evidenced by our experiments.

- We perform extensive numerical experiments that validate the efficacy of our approach. Specifically, our experiments showcase favorable performance over current practice in three aspects: (1) superior fitting fidelity, (2) higher parsimony and (3) efficient online tracking with substantive speed-up of computations.

The remainder of this paper is organized as follows: first, we describe the model for empirical distributions (from travel time data) in Section 2. The proposed estimation method is presented in Section 3, first for continuous distributions and subsequently for their discretized counterparts. The use of Gamma kernels, boundary bias, and the corresponding discretization are illustrated in Section 4. We derive the Mittag-Leffler kernel (a generalization of Gamma kernels) to allow for variable scale parameters in Section 5; this approach is referred to as adaptive (scale-wise) estimation. The numerical solution for optimally selecting the weights, alongside a post-processing mechanism are presented in Section 6, while recursive estimation from streaming time series is discussed in Subsection 8.5. Section 8 is devoted to testing of the proposed approach using both synthetic data (for
validation) and real-world data (for demonstrating the applicability in real-life settings). Our findings firmly support the efficacy of the proposed sparse kernel density estimation with asymmetric Mittag-Leffler kernels for learning travel time distributions, in terms of improved fitting accuracy at a simultaneous benefit of substantially ameliorated parsimony in the selected kernels. Finally, Section 9 concludes the paper.

2. Parzen density estimator

Given \( S \) samples \( T_1, \ldots, T_S \) drawn from a population with (unknown) probability density function \( p(\cdot) \), the Parzen density, also known as Parzen window (PW) estimator (Parzen, 1962; Cacoullos, 1966; Raudys, 1991; Silverman, 1986) at \( t \) is given by:

\[
\hat{p}(t) = \frac{1}{S} \sum_{j=1}^{S} \kappa_h(t - T_j),
\]

where \( \kappa_h(\cdot) \) is a window (or kernel) of width \( h \), and \( h \) is called the smoothing parameter. The Parzen density can equivalently be interpreted as a modified histogram, allowing for kernels to be non-rectangular. As an example, choosing \( \kappa_h(\cdot) \equiv \delta(\cdot) \), where \( \delta(\cdot) \) is the Dirac delta function, we get the standard empirical distribution \( \frac{1}{S} \sum_{j=1}^{S} \delta(t - T_j) \), which uses kernels with zero bandwidth, \( h = 0 \). Typically \( \kappa_h(\cdot) \) is a probability density function (PDF); for example, we use the Gaussian density with variance \( h^2 \) in our experiments. Several methods have been proposed to determine \( h \) based either on minimizing the mean square error or based on cross-validation techniques (see (Lacour et al., 2016) and references therein for a contemporary treatment of the bandwidth selection problem).

Parzen window (PW) estimators can also be regarded as a special type of finite mixture models, where the mixture components are assigned equal weights and are located exactly at the training data. The PW estimator generally requires as many kernels as the number of training samples. As a result, it may incur substantial storage requirements, and is also prone to overfitting. In this paper, the PW estimators serve as empirical distribution functions (or generalized histograms), and the goal is to develop and fit parsimonious models that may as well achieve higher (out-of-sample) prediction accuracy.

3. Sparse density estimation

3.1. Sparse kernel density estimation

Consider the kernel density

\[
p(t) = \sum_{m=0}^{M-1} \theta_m \phi_m(t),
\]

where \( \{\phi_m(\cdot)\}_{m=0}^{M-1} \) are the kernel functions (\( M \) in total) and \( \{\theta_m\}_{m=0}^{M-1} \) are the kernel weights. We will allow \( M \) to be large so that (2) is rich enough to fit a broad class of distributions. Our aim is to achieve a sparse representation of \( \hat{p}(\cdot) \) (a parsimonious fit), i.e., one with most of the elements of the vector \( \theta = [\theta_0 \ldots \theta_{M-1}]^T \) being zero while maintaining test performance or generalization capability comparable to that of the PW estimate obtained with an optimized bandwidth \( h \). We thus seek to solve:

\[
\min_{\theta \in \Omega} \frac{1}{2} \left\| \hat{p}(\cdot) - \sum_{m=0}^{M-1} \theta_m \phi_m(\cdot) \right\|_2^2 + w \|\theta\|_1,
\]

where \( \Omega \subseteq \mathbb{R}^M \) is a set of \( M \)-dimensional vectors that we consider for the optimization problem, and \( w \geq 0 \) is the regularizing parameter. The \( L_2 \)-norm is over a suitably chosen (infinite-dimensional) functional space\(^2\) and the \( \ell_1 \)-norm is the usual (finite dimensional) vector norm, i.e., the sum of

\[^{2}\text{Typically,}\]

\[
\frac{1}{2} \left\| \hat{p}(\cdot) - \sum_{m=0}^{M-1} \theta_m \phi_m(\cdot) \right\|_2^2 = \frac{1}{2} \int_{\mathbb{R}} \left( \hat{p}(t) - \sum_{m=0}^{M-1} \theta_m \phi_m(t) \right)^2 dt.
\]
absolute values of the vector entries. The first term in the objective function, \( \| \hat{p}(\cdot) - \sum_{m=0}^{M-1} \theta_m \phi_m(\cdot) \|_2 \) is a measure of goodness-of-fit: it is the (squared) \( L_2 \)-distance between the empirical distribution (of the data) \( \hat{p}(\cdot) \) and the fitted distribution \( \bar{p}(\cdot) \). The second term is an \( \ell_1 \)-regularizer: \( \| \cdot \|_1 \) is known to promote sparsity (Tibshirani, 1996) in the vector of weights \( \theta \), i.e., a parsimonious solution. Finally, note that a higher value for \( w \) yields higher sparsity of the optimal solution vector \( \theta \) of the optimization problem (3).

3.2. Support discretization

To solve the estimation problem (3) via numerical optimization, we discretize the travel times. This is done by defining disjoint intervals (of the same or variable lengths) in the support of the distribution and associate with each interval a representative value (denoted by \( t_n \) for the \( n \)-th interval, e.g., its midpoint). Subsequently, each data point is assigned the representative value of the interval it lies in: let \( \tau(\cdot) \) be a surjective mapping from the continuous interval \([0, T_{\text{max}}]\) into the discrete set \( \{t_n\}_{n=0}^{N-1} \), i.e., \( \tau(\cdot) \) performs the operation \( t \mapsto t_n \). In effect, the function \( \tau(\cdot) \) takes a continuous travel time \( t \) and returns its representative \( t_n \) in the discrete set. Consequently, the distributions \( \hat{p}(\cdot) \) and \( \bar{p}(\cdot) \) are approximated by vectors of size \( N \), denoted, respectively, by \( \hat{p} \) and \( \bar{p} \). We thus have for any \( t \geq 0 \) that

\[
\bar{p}(t) \approx \bar{p}_\tau(t) = \sum_{m=0}^{M-1} \theta_m \phi_m(\tau(t)).
\]

(4)

The locations of the \( M \) kernels simply constitute a set of travel times, which we denote by \( \{t_m\}_{m=0}^{M-1} \); note that these do not necessarily coincide with the discrete support of the distribution. Besides, we will consider kernels with variable width, so that \( \{t_m\}_{m=0}^{M-1} \) may not have \( M \) distinct values, i.e., some values coincide (this corresponds to the case of placing multiple kernels of different width at the same location), and \( M > N \) is possible. The distinct values in \( \{t_m\}_{m=0}^{M-1} \) are taken to be a subset of the discrete support of the distribution \( \bigcup_{m=0}^{M-1} \{t_m\} \subseteq \{t_n\}_{n=0}^{N-1} \); denoting the number of distinct values in the set \( \{t_m\}_{m=0}^{M-1} \) by \( M' \), we have necessarily \( M' \leq N \). For the case of single scale parameter used (see Section 4) it holds that \( M = M' \leq N \).

The \( M \) kernels are further quantized in accordance with the discretization of the support as follows: we define \( \phi_{n,m} \equiv c_{n,m} \phi_m(t_n) \), where \( c_{n,m} \) is a constant that depends on the discretization method, and the \( n \)-th kernel function. Similarly, we may quantize the PW by setting \( \hat{p}_n = \alpha_n \hat{p}(t_n) \). In essence, \( c_{n,m}, \alpha_n \) are a measure of the width of the \( n \)-th interval; for example \( \alpha_n \equiv \Delta \) for a uniform discretization with step-size \( \Delta \) (e.g., we use a second-by-second uniform discretization in our experiments and set \( \alpha_n \equiv 1 \)). We discuss the issue of kernel discretization (the selection of \( \{c_{n,m}\} \)) in detail in Subsection 4.2 and Section 5. Finally, we set \( \bar{p}_n \equiv \bar{p}(t_n) \).

Correspondingly, we obtain a vector \( \hat{p} \in \mathbb{R}^N \) and a matrix \( \Phi \equiv [\phi_{n,m}] \in \mathbb{R}^{N \times M} \). We have

\[
\bar{p} = \Phi \theta,
\]

(5)

and we consider, in the following, the (discrete) estimation problem:

\[
\min_{\theta \in \Omega} \frac{1}{2} \| \hat{p} - \Phi \theta \|_2^2 + w \| \theta \|_1 = \frac{1}{2} \sum_{n=0}^{N-1} \left( \hat{p}_n - \sum_{m=0}^{M-1} \phi_{n,m} \theta_m \right)^2 + w \sum_{m=0}^{M-1} |\theta_m|,
\]

(6)

which is known as the (constrained) Least Absolute Shrinkage and Selection Operator (LASSO) in the statistics and machine learning literature (Tibshirani, 1996).

4. Gamma kernels

The most commonly used kernels (e.g., normalized Gaussian kernels, biweight kernels and Epanechnikov kernels (Bishop, 2006)) all suffer from assigning non-zero probability to negative travel times. In addition, the aforementioned kernels are all symmetric, whereas travel time distributions tend to exhibit asymmetric shapes, thereby a large number of symmetric kernels are needed to obtain
an accurate (but non-parsimonious) fit. To overcome these issues, we propose the use of Gamma kernels.

Like most asymmetric kernels, the shape of the Gamma kernel (specifically, its width) depends on both the scale parameter as well as the location parameter (e.g., its mean). This change in shape results in what is referred to as boundary bias in the statistics literature (cf. Chen (2000)) and is addressed by changing the roles of parameter and argument when placing the kernels, as explicated next. In contrast, the shape of a Gaussian kernel is only dictated by its scale parameter.

4.1. Boundary bias

The probability density function (PDF) of the Gamma distribution is given by:

\[ p_G(t; \beta, \sigma) = \frac{1}{\sigma \Gamma(\beta)} \left( \frac{t}{\sigma} \right)^{\beta - 1} e^{-\frac{t}{\sigma}}, \tag{7} \]

where \( \beta > 0 \) is the shape parameter, \( \sigma > 0 \) is the scale parameter, and where \( \Gamma(\cdot) \) is the Gamma function. We extend the Gamma kernel mixture model proposed by Chen (2000) to allow for variable (sparse) weights: to evaluate the probability density at \( t \), the model a) uses a single Gamma PDF with its mode (equal to \( (\beta - 1)\sigma \)) coinciding with \( t \), b) it evaluates the densities of the sample points using this single function, and c) calculates a weighted sum of these densities. Effectively, the roles of parameter and argument are reversed. This prevents bias from kernels located near the boundaries.

Specifically, the placement of the kernels is done as follows: in order to locate the (mode of the) kernel at the argument \( t \), the location parameter is set so that \( (\beta - 1)\sigma = t \), i.e., we set \( \beta = 1 + (t/\sigma) \); hence, for a given scale parameter \( \sigma > 0 \), the \( m \)-th Gamma kernel is given by:

\[ \phi_m(t) = p_G(t_m; 1 + \frac{t}{\sigma}, \sigma) \tag{8} \]

The estimated probability (before discretization) is then given by:

\[ \bar{p}(t) = \sum_{m=0}^{M-1} \theta_m p_G(t_m; 1 + \frac{t}{\sigma}, \sigma) = \sum_{m=0}^{M-1} \frac{\theta_m}{\sigma \Gamma(1 + \frac{t}{\sigma})} \left( \frac{t_m}{\sigma} \right)^{\frac{1}{2}} e^{-\frac{t_m}{\sigma}}. \tag{9} \]

This mechanism is illustrated in Fig. 1.

![Fig. 1: Mixture of Gamma kernels: Gamma PDF centered at \( t \) and evaluated at times \( \{t_m\}_{m=1}^5 \) with weights \( \{\theta_m\}_{m=1}^5 \).](image)

4.2. Discretization of the Gamma kernel

In this section, we elaborate on the appropriate normalization, post-discretization, i.e., we show how to select the constants \( c_n \equiv c_{n,m} \) (i.e., independent of \( m \)) for \( n = 0, \ldots, N - 1 \) when building the kernel matrix \( \Phi \) so that the discretized density \( \bar{p} \) indeed represents a valid probability mass function.
(PMF), i.e., its entries sum up to 1. It is worthy noting that Gamma kernel density functions in the statistics literature do not, in general, integrate to unity. In other words, \( \int_0^\infty p_G(t_m, 1 + \sigma^{-1} t, \sigma) dt \neq 1 \): this is a consequence of reversing the roles of parameter and argument. In this paper, we give close attention to this issue and ensure that our approach guarantees that, post-discretization, \( \bar{\Phi} \) is a valid PMF.

It is also important to point out that the standard approach of normalizing \( \bar{\Phi} \) as a post-processing step is not applicable in our case. In standard practice, one only considers goodness-of-fit: given that each kernel vector substantially affect the goodness-of-fit for a given sparsity level.

Therefore, we carefully design the discretization in such a way that the resulting kernel probability densities sum to unity (approximately). Mathematically, we require that \( \sum_{n=0}^{N-1} \bar{\Phi}_n \approx 1 \), where the approximation error is kept below a predefined threshold. Since each kernel vector \( \{\phi_{n,m}\}_{n=0}^{N-1} \) is interpreted as a probability distribution, we will first require that \( \sum_{n=0}^{N-1} \phi_{n,m} \approx 1 \) for all \( m \in \{0, \ldots, M-1\} \) (equivalently, we require that \( \Phi \) is approximately column-stochastic). We propose a choice for the discretization constants \( \{c_n\}_{n=0}^{N-1} \) so that this is indeed the case.

For simplicity, we assume a uniform discretization of the support \( \{t_n\}_{n=0}^{N-1} \), that is, we set a constant \( \Delta \) such that \( t_n = n\Delta \) for \( n = 0, \ldots, N-1 \) and further \( t_m = (m+1)\Delta \) for \( m = 0, \ldots, M-1 \). Define \( \tilde{\Delta} \equiv \Delta/\sigma \). Then,

\[
\sum_{n=0}^{N-1} \phi_{n,m} = \sum_{n=0}^{N-1} c_n p_G(t_m, 1 + n\Delta, \sigma) = \sum_{n=0}^{N-1} c_n \frac{1}{\sigma \Gamma(1 + n\Delta)} \left( \frac{t_m}{\sigma} \right)^{n\Delta} e^{-t_m/\sigma}.
\]

Selecting the scale parameter so that \( \tilde{\Delta} = 1 \), i.e., \( \sigma \equiv \Delta \), the above sum satisfies the desired property (summation to unity) for any \( m \) when \( c_n = \sigma \) for all \( n \) and \( N \to \infty \). This holds since the terms inside the sum on the right hand side of (10) take the form of the probability mass function of a Poisson distributed random variable with rate parameter \( t_m/\sigma \). As such, we choose \( N \) so that the sum is approximately unity. This can be accomplished as follows: let \( X \) be a Poisson random variable with rate parameter

\[
\max_{\{t_n\}_{n=0}^{M-1}} \frac{t_m}{\sigma} = \frac{(M-1)\Delta}{\sigma} = M - 1.
\]

For an acceptable error threshold \( \varepsilon > 0 \), we choose \( N \) so that \( \mathbb{P}(X \geq N) \leq \varepsilon \): this is simply the \((1-\varepsilon)\)-percentile point of \( X \). Analytically, \( N \) is chosen as:

\[
N \equiv \min \{ n : \mathbb{P}(X \geq n) \leq \varepsilon \}.
\]

Note that (i) choosing the rate parameter as \( \max_{m=0, \ldots, M-1} \frac{t_m}{\sigma} \) renders our choice of \( N \) independent of \( t_m \) and ensures that the threshold error is not exceeded for any \( t_m \); (ii) this is only achievable when \( N > M \); and (iii) ensuring that exactly \( \sum_{n=0}^{N-1} \phi_{n,m} = 1 \) for all \( j \in \{0, \ldots, M-1\} \) (as opposed to it being approximately equal to unity) can be achieved by re-defining \( \phi_{0,m} \) (a constant which depends on \( m \)) as

\[
\phi_{0,m} \equiv (\varepsilon_m + 1) e^{-t_m/\sigma},
\]

where

\[
\varepsilon_m = \sum_{n=N}^{\infty} \frac{1}{n!} \left( \frac{t_m}{\sigma} \right)^n.
\]

\footnote{We do not place a kernel at value zero, as it would be a trivial one (identical to 0) cf. (8).

\footnote{A very similar analysis can be carried out for discretizing the PW \( \hat{\pi}(. \cdot) \), i.e., we may choose \( N' \) large enough based on the density \( \kappa_N(. \cdot) \) and select \( N \) as the maximum of \( N' \) and the value in (12). Note that the resulting value \( n\Delta \) may as well exceed the value \( T_{max} \) in which case we simply consider 'large' travel times in the support: this is not an critical issue since these are assigned zero (or very low) probabilities by the PW estimator.}}

\[

\]
Note that \( \varepsilon_m \) is increasing in \( m \), therefore is upper-bounded by \( \varepsilon_M \). This concludes the descriptive analysis of discretizing Gamma kernels.

5. Our proposal: the Mittag-Leffler kernel

One drawback of the approach outlined above is the necessity for a single scale parameter \( \sigma \) (as it becomes evident from our derivation of the normalizing constant). To tackle this issue, we proceed to devise adaptive kernels, i.e., generalized Gamma kernels using Mittag-Leffler functions, where the scale parameters belong in a discrete set. Before we can do so, we need to address the issue of the units of measurement: that is, the assumption that \( \Delta = 1 \) is no longer feasible since \( \sigma \) is allowed to vary from one kernel to another. To ensure summability to unity, we generalize the Gamma kernel to one which uses a generalized form of the exponential function. This can be achieved by replacing the exponential in \( \exp(-t/\sigma) \) in (7) with the reciprocal of the (scaled) Mittag-Leffler function (Haubold et al., 2011):

\[
E_p(t) \equiv \sum_{n=0}^{\infty} \frac{t^n}{\Gamma(1 + nb)}
\]

(15)

Note that the exponential function is a special case of the Mittag-Leffler function obtained for \( b = 1 \); i.e., \( E_1(t) \equiv \exp(t) \). The elements of the generalized kernel matrix, which we name the Mittag-Leffler (M-L) kernel are then given by (we first consider variable position \( i.e., t_m \) but single scale parameter \( \sigma \) for the time being; we will relax this assumption next, cf. (21).)

\[
\phi_{n,m} = c_{n,m} p_{M-L}(t_m; 1 + n\Delta, \sigma) = c_{n,m} \frac{1}{\sigma \Gamma(1 + n\Delta)} \left( \frac{t_m}{\sigma} \right)^{n\Delta} \left[ E_\Delta \left( \frac{t_m}{\sigma} \Delta \right) \right]^{-1}.
\]

(16)

The right-hand side of (16) is the probability mass function of a generalized hyper-Poisson random variable, for a particular selection of parameters. Indeed, let \( X \) be a hyper-Poisson random variable and let \( p_{\tilde{X}}(\cdot; a, b) \) denote its probability mass function, with \( a, b \) being the parameters of the distribution. It holds that

\[
P(\tilde{X} = n) \equiv p_{\tilde{X}}(n; a, b) = a^n \frac{1}{\Gamma(1 + nb)} E_b(a).
\]

(17)

Setting \( a \equiv (t_m / \sigma)\bar{\Delta} \), \( b \equiv \bar{\Delta} = \bar{\Delta} / \sigma \), and \( c_{n,m} \equiv \sigma \) for all \( n, m \), we have that, for each \( m \), \( \{\phi_{n,m}\} \) in (16) is a probability mass function of some hyper-Poisson random variable. Since this random variable depends on \( m \), we denote it by \( \tilde{X}_m \). In a similar manner to the Gamma kernel, set

\[
N \equiv \min \{ n : P(\tilde{X}_m \geq n) \leq \varepsilon, \text{ for all } m \in \{0, \ldots, M - 1\} \} = \min \{ n : P(\tilde{X}_{M-1} \geq n) \leq \varepsilon \}
\]

(18)

for some given tolerance level \( \varepsilon > 0 \). Consequently, we have that \( \sum_{m=0}^{N-1} \phi_{n,m} \approx 1 \) and, once again, we may assure that \( \sum_{m=0}^{N-1} \phi_{n,m} = 1 \) exactly by re-defining

\[
\phi_{0,m} \equiv (\bar{\varepsilon}_m + 1) \left[ E_\bar{\Delta} \left( \frac{t_m}{\bar{\Delta}} \right) \right]^{-1},
\]

(19)

where

\[
\bar{\varepsilon}_m = \sum_{n=N}^{\infty} \frac{1}{\Gamma(1 + n\Delta)} \left( \frac{t_m}{\sigma} \right)^{n\Delta}.
\]

(20)

We next utilize the results above to develop an adaptive kernel mixture, that is, we let the scale parameter \( \sigma_m \) vary with \( m \). Correspondingly, we define \( \Delta_m \equiv \Delta / \sigma_m \), whence the adaptive kernels are given by:

\[
\phi_{n,m} = \sigma_m p_{M-L}(t_m; 1 + n\Delta_m, \sigma_m).
\]

(21)

The summability requirement \((\sum_{m=0}^{N-1} \phi_{n,m} = 1, \text{ for each } m)\) is ensured by following the recipe in (16) - (20) (replacing \( \sigma \) with \( \sigma_m \)).
Observe that in an M-L kernel mixture multiple kernels (of variable scale \(\sigma_m\)) may be placed at the same travel time \(t_m\); this implies that \(M > M'\), and \(M > N\) is possible. Similarly, the set \(\{\sigma_m\}_{m=0}^{M-1}\) need not have distinct values (i.e., kernels with the same scale parameter are placed over variable times). Nonetheless, the above analysis shows that necessarily \(N > M'\), where \(M'\) denotes the number of distinct values of the times \(\{t_m\}_{m=0}^{M-1}\) where a kernel is placed.

6. Numerical optimization

In what follows, the (constrained) LASSO problem (6) is considered by taking \(\Omega = \mathbb{R}_+^M\) (the positive orthant) as opposed to \(\Omega = \{\theta \in \mathbb{R}_+^M \mid \sum_{m=0}^{M-1} \theta_m = 1\}\) (the probability simplex). This is done purposefully for two reasons. First, this choice yields more efficient numerical optimization methods, which is especially important for real-time learning (effectively, the projection to the positive orthant is much simpler than the projection to the simplex, which requires sorting). Second, and more importantly, setting \(\Omega = \{\theta \in \mathbb{R}_+^M \mid \sum_{m=0}^{M-1} \theta_m = 1\}\) would result in the optimization problem:

\[
\min_{\theta \geq 0} \frac{1}{2} \| \hat{p} - \Phi \theta \|_2^2 + w^\top \theta
\]

subject to \(1^\top \theta = 1\),

which is further equivalent to (since the second term in the objective is determined by the equality constraint)

\[
\min_{\theta \geq 0} \frac{1}{2} \| \hat{p} - \Phi \theta \|_2^2
\]

subject to \(1^\top \theta = 1\).

This leaves no control over sparsity, since the objective no longer depends on the control parameter \(w\). This is clearly an undesirable feature when aiming for parsimonious solutions in a controllable fashion, and justifies our choice of selecting \(\Omega = \mathbb{R}_+^M\) in what follows. Ensuring that \(\sum_{n=0}^{N-1} p_n = 1\) (exactly) is discussed in Appendix A: this can be ensured by a zero-overhead post-processing mechanism.

LASSO (6) is a convex problem (Boyd and Vandenberghe, 2004) and there exist a multitude of schemes for solving it numerically. Aside from generic convex solvers such as CVX (Grant and Boyd, 2014), many numerical optimization methods have been developed: these include applications of the fast proximal gradient method of Nesterov (2013) such as (Beck and Teboulle, 2009; Wright et al., 2009), of the Alternating Direction Method of Multipliers (ADMM) (Parikh and Boyd, 2014) such as (Afonso et al., 2010), and of interior point methods (Kim et al., 2007). Recently, a quasi-Newton solver featuring substantial acceleration for high-accuracy solutions was devised by (Sopasakis et al., 2016).

In this paper, we consider \(\Omega = \mathbb{R}_+^M\), a constrained LASSO problem (non-negative weights):

\[
\min_{\theta \geq 0} \frac{1}{2} \| \hat{p} - \Phi \theta \|_2^2 + w^\top \theta
\]

which has a differentiable objective and very simple constraint set. Aside of using CVX (which may be relatively slow as a generic solver) we have implemented a fast projected gradient method for this problem as well as use the log-barrier interior-point method (l1_ls) based on the analysis in (Kim et al., 2007). For the latter, we set a logarithmic barrier for the non-negative constraints as \(-\sum_{m=0}^{M-1} \log(\theta_m)\) and augment the objective function to obtain the associated centering problem

\[
\min_{\theta \in \mathbb{R}_+^{M+1}} \frac{z}{2} \| \hat{p} - \Phi \theta \|_2^2 + zw\sum_{m=0}^{M} \theta_m - \sum_{m=0}^{M} \log(\theta_i),
\]

where an additional kernel may be added to ensure summability to unity (see Appendix A). For each instance of constrained LASSO, we solve a sequence of these problems for increasing \(z\) (note that centering problem becomes equivalent to the original as \(z \to +\infty\)).
6.1. Post-processing

Once a numerical solution of LASSO (22) is obtained, it is important to numerically post-process it. For example, we aim for ‘zero’ values in the solution vector $\theta$, but this practically corresponds to very small entries. One simple yet effective way to define the zero entries of $\theta$ is by thresholding, e.g., setting all entries $|\theta_i| < \epsilon \|\theta\|_\infty$ to zero, for some small value of $\epsilon$, e.g., $\epsilon = 10^{-3}$ was used in our experiments. After thresholding, the support of the solution $\text{supp}(\hat{\theta})$ (the set of non-zero entries) and the corresponding number of non-zero entries $s_w \equiv |\text{supp}(\theta)|$ are determined. An additional way to improve sparsity is by combining nearby kernels that appear in the (thresholded) solution, i.e., kernels whose locations lie within a predetermined distance. Finally, we may improve the reconstruction fidelity by performing constrained least-squares on the resulting support: i.e., we obtain a new matrix $\Phi_s \in \mathbb{R}^{N \times s_w}$ by selecting the set of columns of $\Phi$ corresponding to the support, and perform constrained least-squares to update the entries $\hat{\theta}_s$:

$$
\minimize_{\theta \in \Omega_s} \|\hat{\theta} - \Phi_s \theta\|_2^2.
$$

This is usually referred to as a de-biasing step, where $\Omega_s = \mathbb{R}^{s_w}$ is selected in light of the discussion in Appendix A. We denote the resulting final solution obtained after post-processing (thresholding for support detection, combining neighboring kernels, de-biasing, and stacking back zero entries for values not belonging in the support so as to obtain an $M$—dimensional vector) by $\theta$, with slight abuse of notation, in the sequel.

6.2. Increasing the sparsity

We may further increase the sparsity by scaling the weights $\theta$ in a way that favors kernels with larger scale parameters, as a type of preconditioning. Formally, let $\Sigma \in \mathbb{R}^{M \times M}$ be a diagonal matrix with elements $\Sigma_{m,m} = \sigma_m$ and consider the following re-scaled version of the estimation problem (note that this is only meaningful in the adaptive kernel case, i.e., for M-L kernels as well as Gaussian kernels, whereas the Gamma kernels need to have the same scale parameter):

$$
\minimize_{\theta \in \Omega} \frac{1}{2} \|\hat{\theta} - \Phi \theta\|_2^2 + w \|\Sigma^{-1} \theta\|_1.
$$

By modifying the weight vector in this way, we penalize the basis kernels in proportion to the inverse of the scale parameter. This is done to encourage the sparse density algorithm to choose basis kernels with larger scale parameters (and hence fewer kernels) to capture the distribution; informally, when two or more kernels yield a fitting accuracy comparable with one wider kernel, the latter will be selected.

A second way of increasing the sparsity is by combining kernels that are very closely spaced, as discussed in the aforementioned post-processing procedure. We have adopted these mechanisms in our experiments on real-data and observed substantive performance amelioration.

6.3. Choice of regularization parameter

The regularization parameter $w$ controls the trade-off between sparsity and reconstruction error. If the regularization parameter $w$ is sufficiently large most of the coefficients are driven to zero, thus leading to a sparse model with only a few (relevant) kernel functions. However, this typically leads to poor fitting accuracy (low goodness-of-fit). On the other hand, when $w$ is sufficiently small one retrieves the best possible fit (non-negative least-squares), which is (in general) not sparse: most (typically, all) coefficients are non-zero. In selecting $w$, the aim is to balance the trade-off between goodness-of-fit and sparsity. The problem of choosing the appropriate regularization parameter is crucial as it governs the selection of the sparsest model that can faithfully reconstruct the underlying distribution of the data. One approach to select a suitable $w$, which makes good use of the available dataset (Efron and Gong, 1983; Turney, 1994), is $k$-fold cross-validation. Notwithstanding, cross-validation techniques do not promote sparsity in general, but are rather geared towards avoiding overfitting. Moreover, an issue with cross-validation is that it does not lead to consistent model selection for LASSO.
We propose a simple scheme for tuning the parameter \( w \) to balance the trade-off between goodness-of-fit and sparsity. For this purpose, we use a metric inspired by the analysis in (Reid et al., 2013; Sun and Zhang, 2012) on scaled-LASSO, namely

\[
S^2_w = \frac{\| \hat{p} - \Phi \hat{\theta}(w) \|^2}{M - s_w},
\]

where \( s_w = |\text{supp}(\theta(w))| \) is the cardinality of the support set (as determined via the post-processing mechanism of Subsection 6.1), i.e., the number of non-zero entries of the solution vector. We use \( \theta(w) \) to emphasize the dependence of the (constrained) LASSO solution on the regularizing parameter \( w \).

The metric \( S^2_w \) in (26) captures the trade-off between a) goodness-of-fit, as measured by the squared \( \ell_2 \)-error \( \| \hat{p} - \Phi \hat{\theta}(w) \|^2 \) and b) sparsity \( (M - s_w) \) (the number of zeros in the solution \( \theta \)); it is proportional to the former and inversely proportional to the latter. Note, therefore, that seeking to minimize this metric leads to aiming for simultaneously maximizing the goodness-of-fit and parsimony, and this is exactly the approach that we adopt in this paper. Last, \( S^2_w \) is well-defined for \( s_w < M \), i.e., it is not defined for values of \( w \) close to \( 0^5 \) where typically \( s_w = M \); we may extend it to take the value infinity in such case (since a sparse solution is desirable).

For \( w = 0 \), one retrieves the constrained least-squares solution:

\[
\min_{\theta \geq 0} \| \hat{p} - \Phi \theta \|^2_{\ell^2},
\]

which serves as a lower bound for the squared \( \ell_2 \)-error (best possible goodness-of-fit) but is known to be non-sparse \((s_w = M \) in most cases). For \( w > w_0 \) where

\[
w_0 \equiv \| \Phi^\top \hat{p} \|_{\ell^\infty},
\]

the all-zero solution is retrieved \((s_w = 0)\); this maximizes sparsity but yields a squared \( \ell_2 \)-error equal to \( \| \hat{p} \|^2 \). One may then search over variable values of \( w \) and select the one that minimizes \( S^2_w \).

For example, we may consider values of \( w \) in a logarithmic scale: starting from \( w_0 = \eta^k w_0 \) for values \( w_k = \eta^k w_0 \) for some \( \eta \in (0, 1) \), e.g., \( \eta = 0.95 \) was chosen in our experiments, where \( k \) is successively increased until a termination criterion is met (in our experiments we have considered:

\[
\frac{\| \hat{p} - \Phi \hat{\theta}(w_{k+1}) \|^2 - \| \hat{p} - \Phi \hat{\theta}(w_k) \|^2}{\| \hat{p} - \Phi \hat{\theta}(w_k) \|^2} < \epsilon', \text{ with } \epsilon' = 10^{-3}.
\]

An alternative is to achieve a desirable sparsity level exactly by means of the search mechanism above in conjunction with bisection. This can be applied to all the sparse estimation problems that we consider in this paper (cf. Fig. 4 for illustration).

7. Recursive estimation

The sparse density estimation methods that we have presented thus far implicitly assume that the travel times are all available for density estimation purposes. This is an inherent issue with traditional data analysis methods that naturally amount to offline data processing. In order to capture real-time variation in travel time (for instance due to recurrent or non-recurrent events), this section presents an efficient online algorithm that operates directly on streaming measurements. Our approach is inspired from and extends the work of Freris et al. (2013a,b) and Sopasakis et al. (2016) on recursive compressed sensing, which applies LASSO to successive overlapping windows of the data stream; the reader is also referred to (Hofleitner et al., 2013, 2014) for a closely related very efficient homotopy method that sequentially solves LASSO on streaming measurements.

The key observation is that the dimensionality of our problem \( M \) (size of \( \theta \)) does not depend on the size of the dataset \( S \), but depends solely upon the granularity of time discretization (as well as the choice of scale parameters for M-L kernels). For efficient sparse kernel density estimation using streaming data it is important to devise a method that (a) efficiently updates the Parzen density based on new measurements and (b) provides fast numerical solutions to the LASSO problem (22).
Satisfying these two requirements ascertains that the resulting method is suitable for an online implementation subject to high frequency streaming measurements and stringent real-time constraints in estimating variable densities. To accomplish the second requirement, we propose using warm-starting in solving (22), i.e., we use the previously obtained estimate $\hat{\theta}$ as a starting point to an iterative LASSO solver (while properly updating the Parzen vector $\hat{p}$). This is advantageous and leads to a substantial acceleration; we demonstrate this experimentally in Subsection 8.5. We showcase how the first requirement can be satisfied by considering two scenarios: (i) sequential processing of travel times, i.e., more data become available from the ‘same’ underlying distribution, whence the changes in estimated parameter $\theta$ reflect enhancing the learning outcome based on new data, and (ii) a rolling-horizon setup, in which data are processed via windowing (Freris et al., 2013a,b) so as to track online variabilities in the travel time densities (e.g., due to building congestion and/or online traffic control) in real-time; in such case, the underlying distribution is considered to vary with time and the goal is to ‘track’ its parameters. This second scenario provides a handy data analytics tool to assay changes in experienced travel times during a given day, or from day-to-day within a given week or month. It is notable that parameter tracking can also be used for anomaly detection, for instance, to identify accidents based on abrupt changes in the travel time distribution. We briefly discuss the two scenarios below. We assume, without any loss in generality, that the online algorithm accepts streaming travel time data and processes the data one observation at a time.

7.1. Sequential data processing

We consider an ‘infinite’ stream of travel time data $\{T_1, T_2, \ldots\}$ (without loss of generality, they are taken discretized as elaborated in Section 3 and Section 4, so that they belong in the set $\{t_n\}_{n=0}^{N-1}$). Sequential processing amounts to learning the underlying kernels corresponding to using the first $K + 1$ data points based on the estimated kernels using the first $K$ ones, for $K \in \mathbb{Z}_+$. The Kth LASSO problem is

$$\hat{\theta}^{(K)} \in \text{argmin}_{\hat{\theta} \geq 0} \frac{1}{2} \left\| \hat{p}^{(K)} - \Phi \hat{\theta} \right\|^2_2 + w \hat{\theta}^\top \theta. \quad (29)$$

Observe that the kernel matrix $\Phi$ does not depend on $K$, but depends on our choice of time discretization (as well as the scale parameters for M-L kernels). As explained above, we use warm-starting to obtain the solution $\hat{\theta}^{(K+1)}$ while using as starting point to our numerical solver the previous solution $\hat{\theta}^{(K)}$. The Parzen density (1) is recursively updated as follows:

$$\hat{p}^{(K+1)}(t) = \frac{K}{K+1} \hat{p}^{(K)}(t) + \frac{1}{K+1} \kappa_h(t - T_{K+1}). \quad (30)$$

Since we consider discretized data, the values $\kappa_h(t - t_j)$ can be precomputed for $t \in \{t_n\}_{n=0}^{N-1}$ and $j = 0, 1, \ldots, N - 1$. Let us define the matrix $P \in \mathbb{R}^{N \times N}$ (depending exclusively on time discretization, where the $i$th column of $P$, denoted by $P_i$, is given by:

$$P_i = \{\kappa_h(t_n - t_i)\}_{n=0}^{N-1}. \quad (31)$$

Therefore, the vector $\hat{p}^{(K+1)}$ can be obtained from $\hat{p}^{(K)}$ along with the new data point $T_{K+1}$ using $O(N)$ operations as follows:

$$\hat{p}^{(K+1)} = \frac{K}{K+1} \hat{p}^{(K)} + \frac{1}{K+1} P_{T_{K+1}}, \quad (32)$$

where $P_{T_{K+1}} \in \mathbb{R}^{N}$ is the column of $P$ corresponding to the (discretized) travel time $T_{K+1}$.

7.2. Rolling-horizon data processing

This recursive scheme allows real-time streaming data to be incorporated into the model as they arrive, and gradually removes old data that becomes irrelevant. This is achieved by sampling the input stream recursively via overlapping windowing similar to (Freris et al., 2013a,b), rather than using all historical data available to learn the model parameters. This enables the sparse density
model to adapt to changes in the underlying data distribution (due, for example, to within-day and
day-to-day variability in traffic conditions).

We define $T^{(i)}$ to be the $i$th window taken from the streaming travel time data. Without loss of
generality, we will assume a fixed window of length $W$, and for a travel time data stream $\{T_1, T_2, \ldots\}$,
we define $T^{(i)} \equiv \{T_i, \ldots, T_{i+W-1}\}$ and, similarly, $T^{(i+1)} \equiv \{T_{i+1}, \ldots, T_{i+W}\}$ to be two consecutive
windows. Denoting the Parzen density corresponding to travel times in $T^{(i)}$ by $\hat{p}^{(i)}$, learning from the
$i$th window amounts to solving
\[
\hat{\theta}^{(i)} \in \arg\min_{\theta \geq 0} \frac{1}{2} \|\hat{p}^{(i)} - \Phi \theta\|_2^2 + w \mathbf{1}^\top \theta.
\] (33)

Noting the overlap between two consecutive windows, the $\{\hat{\theta}^{(i)}\}$ sequence of parameters can be
estimated recursively: this can be achieved by leveraging the solution obtained from the data in the
$i$th window to warm-start the iterative solver for LASSO in window $i+1$.

The Parzen density (1) at any time $t$ corresponding to the $i$th window is given by
\[
\hat{p}^{(i)}(t) = \frac{1}{W} \sum_{j=1}^{i+W-1} \kappa_h(t - t_j).
\] (34)

Thus, the empirical PW estimator can be viewed as a sliding kernel estimator with a shifted ker-
nel $\kappa_h(t - T_{i+W})$ being added for every successive window, while the outdated kernel $\kappa_h(t - T_i)$ is
removed, i.e.,
\[
\hat{p}^{(i+1)}(t) = \hat{p}^{(i)}(t) + \frac{1}{W} [\kappa_h(t - T_{i+W}) - \kappa_h(t - T_i)].
\] (35)

Again, the vector $\hat{p}^{(i+1)}$ can be obtained from $\hat{p}^{(i)}$ very efficiently using $O(N)$ operations, as follows:
\[
\hat{p}^{(i+1)} = \hat{p}^{(i)}(t) + \frac{1}{W} (P_{T_{i+W}} - P_{T_i}),
\] (36)

where again $P_{T_{i+W}}, P_{T_i} \in \mathbb{R}^N$ are the columns of $P$ corresponding to the (discretized) travel times
$T_{i+W}, T_i$, respectively.

Owing to the substantial overlap between consecutive data windows, the optimal solution to the
$(i + 1)$th problem is expected to be close to that of the previous problem. This leads to substantial
acceleration in solving successive LASSO problems as demonstrated in Sec. Subsection 8.5.

8. Experimental validation

In this section, we present numerical experiments that demonstrate the merits of our methods on
real-life datasets.

8.1. Numerical testing

We first tested the performance of the proposed approach on a synthetic dataset, using a known bi-
modal probability density. The example we consider compares the performance of Gaussian kernels
and the M-L kernels. For this example, a data set of $S$ randomly drawn samples was used to construct
the kernel density estimate and a separate (out of sample) test data set of size $S_{\text{test}}$ was used to
calculate the root-mean square error (RMSE) defined by:
\[
\text{RMSE} = \sqrt{\frac{1}{S_{\text{test}}} \sum_{j=1}^{S_{\text{test}}} (\hat{p}(t_j) - p(t_j))^2}.
\] (37)

The (true) density to be estimated is given by a mixture of two densities: a Gaussian and a Lapla-
cian with equal weights (0.5):
\[
p(t) = 0.5 \frac{1}{\sqrt{200\pi}} e^{-\frac{(t-60)^2}{200}} + 0.5 \frac{0.2}{2} e^{-0.2|t-30|}.
\] (38)
The density estimation was carried out using a data sample of size $S = 2000$, while the error is reported for an out-of-sample dataset with $S_{\text{test}} = 10,000$. For travel times, we considered uniform per-second discretization of the interval $[1,300]$s, i.e., $M' = 300$. The scale parameter $\sigma_m$ was allowed ten values $\{1,2,\ldots,10\}$ (therefore $M = 10M' = 3000$) for both Gaussian and M-L kernels. For both kernels, we set $N = 2M' = 600$, corresponding to uniform per-second discretization of the interval $[1,600]$s.

![Fig. 2: True PDF vs. PW PDF vs. sparse kernel PDF](image1)

![Fig. 2: True PDF vs. PW PDF vs. sparse kernel PDF](image2)

For this example, all computations were performed using Matlab and CVX (Grant and Boyd, 2014) for numerical optimization. The test was performed ten times and average values are reported. A representative comparison between the density obtained using our proposed approach (for both
Gaussian and M-L kernels), the PW density (using Gaussian kernel with variance $h = 1.5$), and the true density is presented in Fig. 2.

From this figure, it is evident that the sparse estimators provide a very good fit to the true distribution, as is also reported in Table 1 (where RMSE is reported within $\pm 1$ standard deviation). The achieved sparsity was less than 5% and 0.4% of the sample sizes in the case of the Gaussian and M-L kernel, respectively. This indicates that the M-L kernel promotes higher sparsity than the Gaussian kernel, i.e., higher compression rate, while at the same time achieving an order of magnitude improvement in goodness-of-fit, cf. Table 1. In fact, the proposed sparse M-L estimator even outperformed PW in terms of accuracy, which perfectly demonstrates the superior fitting capabilities of our kernel over the Gaussian.

Table 1: Performance comparison on synthetic data.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>Number of kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>PW estimator</td>
<td>5.50e-04 ± 9.96e-05</td>
<td>2000</td>
</tr>
<tr>
<td>Sparse Gaussian estimator</td>
<td>3.3e-03 ± 1.92e-05</td>
<td>95</td>
</tr>
<tr>
<td>Sparse M-L Estimator</td>
<td>4.49e-04 ± 1.16e-04</td>
<td>7</td>
</tr>
</tbody>
</table>

8.2. Experiments on real datasets

In this section, we test the merits of our approach on real-life datasets.

8.2.1. Dataset

The sparse kernel density estimation was applied to travel times extracted from vehicle trajectories made available by the Next Generation SIMulation (NGSIM) program\(^6\) Peachtree Street dataset. The arterial section is approximately 640 meters (2100 feet) in length, with five intersections and two or three through lanes in each direction. The section is divided into six intersection-to-intersection segments which are numbered from one to six, running from south to north. Of the five intersections, four are signalized while intersection 4 is un-signalized. The Peachtree Street data consists of two 15-minute time periods: 12:45 p.m. to 1:00 p.m. (noon dataset) and 4:00 p.m. to 4:15 p.m. (PM dataset). The dataset includes detailed individual vehicle trajectories with time and location stamps, from which the travel times of individual vehicles on each link were extracted. In this study, the link travel time is the time a vehicle spends from the instant it enters the arterial link until the instant it passes the stop-bar at the end of the link (i.e., the time spent at intersections is excluded).

The second dataset we used contains vehicle trajectory data collected under the NGSIM program on eastbound I-80 in the San Francisco Bay area in April 2005. The study area is approximately 500 meters in length and consists of six freeway lanes, including a high-occupancy vehicle (HOV) lane and an on-ramp (see Punzo et al. (2011) for details). Using seven cameras mounted on top of a 30-story building adjacent to the freeway, a total of 5648 vehicle trajectories were captured on this road section in three 15-minute intervals: 4.00 p.m. to 4.15 p.m.; 5.00 p.m. to 5:15 p.m.; and 5:15 p.m. to 5.30 pm. These periods represent the build-up of congestion, the transition between uncongested and congested conditions, and full congestion during the peak period, respectively.

8.2.2. Fitting results and comparisons

In order to demonstrate the effectiveness of the proposed sparse kernel density estimation, we have chosen to model the travel time distributions of southbound traffic on the signalized arterial links along Peachtree Street for the two time periods. We used Gaussian kernels for the estimation of the PW density (the empirical distribution $\hat{p}$), where the bandwidth $h$ was calculated according to the approximation proposed by Silverman (1986): $h = 1.06\varsigma S^{-1/5}$ is picked so as to minimize the integral mean-square error (where $\varsigma$ is the sample variance and $S$ is the sample size). For the M-L kernel mixture, we used $M' = 300$ kernels with the scale parameter set $\sigma_m \in \{1,2,3,4,5\}$ (i.e., $M = 1500$ kernels were used in the estimation procedure).

Fig. 3 (a) shows the PW PDF and the estimated sparse kernel PDF (using M-L functions) for the travel times of the southbound vehicles during the noon period. The fitted distribution is clearly bi-modal and closely follows the PW PDF. The bi-modality of the travel time distribution can be attributed to the presence of two traffic states: non-stopped vehicles along the entire corridor in the southbound direction and stopped vehicles experiencing delay at one or more of the signals. Observe that while the number of kernels required to calculate the PW density is equal to the number of data samples (58 for this case), the sparse algorithm achieves a similar accuracy with a much sparser representation: only 4 M-L kernels were needed; i.e., a compression rate of about 15:1.

(a)  

(b)  

Fig. 3: Travel time densities of Peachtree Street (southbound, noon); (a) M-L kernel, (b) Gaussian mixture with two modes, (c) Gaussian mixture with three modes, (d) Gaussian mixture with four modes.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>Number of kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse M-L Estimator</td>
<td>0.0012</td>
<td>4</td>
</tr>
<tr>
<td>EM</td>
<td>0.0029</td>
<td>2</td>
</tr>
<tr>
<td>EM</td>
<td>0.0055</td>
<td>3</td>
</tr>
<tr>
<td>EM</td>
<td>0.006</td>
<td>4</td>
</tr>
</tbody>
</table>

We compared our approach against the Expectation Maximization (EM) algorithm (Bishop, 2006), the prevalent method for estimation of Gaussian mixture models (Wan et al., 2014a,b). The EM algorithm (using Gaussian mixtures) has been widely used for the estimation of travel time densities, despite its slow rate of convergence (Wu, 1983; Archambeau et al., 2003), and the dependence of the parameter estimates on the choice of the initial values (Biernacki et al., 2003). The commonly adopted method to prevent the EM algorithm from getting trapped in local minima is to start the algorithm with different initial random guesses (Wan et al., 2014b). The importance of properly defining the stopping criterion to ensure that the parameters converge to the global maximum of the likelihood function has been highlighted in (Karlis and Xekalaki, 2003; Abbi et al., 2008). In all our experiments, we used ten randomly selected initial estimates; for termination criterion, we used a lower bound (selected as $10^{-5}$) on the sup-norm of the difference between two successive parameter estimates.
One issue with the EM algorithm is that it requires predetermining the number of kernels. This is in contradistinction with our method, which optimally determines the number of kernels concurrently with the fitting procedure. Given the number of kernels, the EM algorithm is an iterative method to estimate the mean and variance of each Gaussian kernel, along with the weight vector $\theta$. Note, however, that the EM algorithm can be considered an ‘educated heuristic,’ in that it is not guaranteed to converge to an optimal (i.e., maximum-likelihood) solution, not to mention it does not minimize the fitting RMSE.

The results are summarized in Fig. 3 and Table 2: the optimal sparse fitting contains 4 M-L kernels, and we also tested the EM algorithm with 2, 3 and 4 Gaussian kernels. It is evident that using M-L kernels substantially outperforms the estimates obtained by using the EM algorithm in terms of goodness-of-fit.

Furthermore, note that that our model has the favorable property that the goodness-of-fit typically increases in the number of kernels used. This is not the case with the EM algorithm, where using a larger number of kernels may result in an increase of RMSE in many instances. Fig. 4 (see also Table 1) illustrates this: we evaluated the RMSE for our sparse density estimator vs. the EM algorithm for variable number of used kernels (for M-L kernels, we varied the regularizing parameter $w$ so as to achieve different sparsity levels).

![Fig. 4: Fitting accuracy of Sparse Density Estimator vs EM for variable number of components; dataset: I-80.](image)

8.3. Inference with parsimonious models

In order to highlight the predictive capabilities and interpretability of parsimonious models, we have tested our method on hold-out real data from the I-80 dataset: we divided the bulk of the I-80 data in two parts (corresponding to different timestamps): a) a training dataset and b) a hold-out test dataset (where we selected a ratio of 4 : 1 for training vs. test data). We then fit our model using the training data and tested its performance (measured via goodness-of-fit) on the hold-out test data. It is worth noting that this scenario is a challenging one due to the heterogeneity of the travel times recorded over intervals of variable traffic conditions. The results are reported in Fig. 5: Fig. 5 (a) plots the PW on the training and hold-out data, along with the sparse density obtained using M-L kernels (12 kernels were used by our sparse density estimator in this case); Fig. 5 (b) plots the fitting error (RMSE) for both our method and the EM algorithm using variable number of kernels (1-12). It is evident from this experiment that our method clearly outperformed the EM algorithm in terms of higher fitting accuracy on hold-out data.

We note, in passing, that a commonly used way to avoid overfitting is by means of an $\ell_2$-norm
regularizer (also known as Tichonov regularization (Bishop, 2006)) as follows:

$$\minimize_{\theta \in \Omega} \frac{1}{2} \| \tilde{p} - \Phi \theta \|_2^2 + \tilde{w} \frac{1}{2} \| \theta \|_2^2,$$

where the notation $\tilde{w}$ is used to emphasize the fact that this non-negative weight parameter differs from $w$. The latter is calculated using the methods in Subsection 6.3, while the former can be calculated using cross-validation.

We tested our method vs. $\ell_2$—regularization on the Peachtree (northbound, noon) dataset. For both methods, we chose $M = 1500$ M-L kernels for model selection ($M' = 300$ and a scale parameter set $\sigma_m \in \{0.2, 0.3, 0.5, 1, 1.5\}$). For $\ell_2$-regularization, the value $\tilde{w}$ was selected from the set $\{5 \cdot 10^{-5}, 5 \cdot 10^{-4}, 5 \cdot 10^{-3}, 5 \cdot 10^{-2}, 5 \cdot 10^{-1}\}$ by 5 : 1 cross-validation. Fig. 6 illustrates the results. Both methods achieved an RMSE of about 0.008. Nonetheless, the number of basis kernels (and corresponding
weights) that need to be stored to re-create and predict the travel time density was substantially reduced to 5 M-L kernels using sparse density estimation (from 84 needed for $\ell_2$- regularization). In addition to the advantage of reduced storage requirements, the sparse density estimate allows to make inferences about the underlying data through the selected basis kernels. For instance, the selected M-L kernels indicate that the underlying travel time data can be approximated well by two peaks located at around $t = 11$ and $t = 45$. On the other hand, the basis kernels selected by the $\ell_2$-norm regularization are much less informative. This parsimony is further illustrated in Fig. 7 where the experiment was conducted on the I-80 dataset.

![Weight vector $\theta$](a) using $\ell_2$-regularization and (b) using sparse density estimation ($\ell_1$-regularization); dataset: Peachtree (northbound, noon).

8.4. Merits of the M-L kernel

In this section, we demonstrate the superiority of the adaptive approach with M-L kernels over the non-adaptive (Gamma kernels with single-scale parameter $\sigma$) in terms of parsimony of the attained estimate. For this case study, we considered the travel time distribution of the northbound traffic along Peachtree street in the noon time period. The sparse density estimation was first carried out using the M-L kernels with $\sigma_m \in \{1,2,3,4,5\}$ and then using Gamma kernels with single
parameter $\sigma = 1$. The solutions are depicted in Fig. 8(a) and Fig. 8(b) respectively, where we have used $M = 1500$ ($M' = 300$ uniform per-second discretization) for the M-L kernels and $M = 300$ for the Gamma kernels.

The figures indicate that the travel time density can be efficiently represented using two dominant modes (with different scale parameters). However, in the case of the Gamma kernels worse accuracy was achieved by using a much higher number of kernels. Although using a $\sigma = 5$ reduces the number of Gamma kernels required to 2, the sparse Gamma estimate cannot accurately capture the shape of the distribution, as shown in Fig. 9(a); in contrast, the adaptive kernel approach gives an M-L estimate that is indistinguishable from the PW density, as depicted in Fig. 9(b). This implies that fine-tuning the single scale parameter is needed, depending on the sample data, whereas in the case of the adaptive estimation the selection of the (optimal) location/scale parameters is performed automatically by the sparse density estimator.

**Interpreting the results.** From the weight vector of the M-L kernels in Fig. 8(a), it is clear that the predominant basis kernels associated with the highest weights are the M-L kernels with $\sigma = 5$ located at $t = 97$ seconds, and $\sigma = 3$ located at $t = 158$. From this alone, we can infer the most likely travel times of the northbound (noon) traffic along Peachtree street, whereas the weight vector associated with the Gamma kernels is not quite as informative.

**8.5. Real-world testing of recursive algorithm**

The recursive algorithm on streaming data was tested using the I-80 dataset. We track the changes in the travel time density on I-80 using the recursive algorithm, by taking a fixed window size of $W = 100$ travel time samples for each instance of sparse density estimation (along with parameters $M' = 300, N = 600$ corresponding to per-second uniform discretization and scale parameters $\sigma_m \in \{1,2,3,4,5\}$, whence $M = 1500$ M-L kernels are considered). By processing the newly arriving samples one at a time (and simultaneously discarding the oldest ones), the density is constantly updated with time following the mechanism explicated in Subsection 7.2. The travel time densities for the PM peak period predicted by the recursive algorithm are depicted in Fig. 10, where we can observe that the number of modes, as well as their locations, vary significantly over time. For the first time period under consideration, the travel time density at (a representative) time of 4:04 p.m. is plotted; clearly, the density can be captured by a bi-modal distribution. This corresponds to the
uncongested period where the travel times of nearly all the vehicles are below 80 seconds. However, at about 5:08 p.m. (which represents the time when congestion begins to build up), the number of modes increases to 3, introducing a new cluster of vehicles with travel times between 70 and 120 seconds. After congestion has set, the number of modes again reduces to 2 in the third time-period, and the locations of these modes indicate that the travel times of all vehicles have increased. In brief, these results highlight the capability of the recursive algorithm to track the varying travel time density in real-time, in a means that is also robust to the variations encountered by individual vehicles. The model parameters estimated by the recursive algorithm reflect the underlying traffic conditions, and can capture the multi-modality in these distributions very efficiently.

The run-time was reported to be just over 2.5 minutes for recursive estimation vs. about 2.5 hours using the standard method (non-recursive one). This experiment solidifies our claim for the feasibility of a truly real-time implementation of our methods (note that a run-time of 2.5 minutes was needed to track the variability over an interval of 45 minutes). A series of snapshots illustrating the dynamic variation of densities is given in Fig. 11.
9. Conclusions

We have introduced an efficient model-based approach for estimating travel time distributions in urban networks. Our methods employ sparse model selection on a parametric mixture of kernels for obtaining parsimonious estimates that accurately characterize measured histograms of travel times. The numerical examples employed in the paper demonstrate that the proposed approach is a viable alternative to existing kernel density estimators and yields estimates with (a) higher goodness-of-fit, (b) substantial compression compared to the Parzen estimates, and (c) robustness to over-fitting.

In this sparsity-seeking framework, ensuring integrability of the kernel estimates (i.e., ensuring that the resulting function is a PDF) cannot be achieved by normalization as is traditionally done. For this purpose, we have developed a new kernel using Mittag-Leffler functions which was shown to
outperform Gaussian kernels in terms of both accuracy and parsimony. We have also demonstrated how an adaptive approach, that is to say using kernels with variable placements as well as shapes, can further enhance sparsity accuracy.

Most learning algorithms, including sparse model selection, are naturally offline in the sense that they operate on the entirety of a given dataset. To address the crucial problem of online travel time estimation, we have proposed algorithms that directly operate on streaming data measurements in two settings: (a) successively improving the fitting fidelity when new data become available and (b) tracking the variability of the travel times in real-time. Our experiments report a speed-up of several orders of magnitude over offline data analysis.

The approach developed in this paper can be used to recursively compute the distributions of any data stream with positive support, and may therefore find applications in other traffic data, as well (e.g., speeds, occupancies, traffic densities, time headways, spacings, etc.). Additionally, the studied problem is very general one, far beyond the purview of transportation and traffic engineering and
one may view our methods as novel machine learning primitives. To the best of our knowledge, the summability to unity issue with asymmetric kernels was not previously addressed in the statistics literature in the context of regularized problems.

Acknowledgments

This research was funded in part by the NYU Global Seed Grant for Collaborative Research and the National Science Foundation (NSF) under grant CCF-1717207.

Appendix A Ensuring summability to unity

Let $[\theta_0^* \ldots \theta_{M-1}^*]^T$ solve (6), where (for the sake of generality) the kernel elements $\phi_{n,m}$ are given by (21). Suppose $\sum_{m=0}^{M-1} \theta_m^* < 1^\prime$. To address the summability to unity issue, we may append a kernel to the solution with negligible impact on the outcome. Consider the kernel vector $\psi(t',\sigma') \in \mathbb{R}^N$, the elements of which are given by

$$\psi_n(t',\sigma') \equiv \sigma' p_{M-L} \left( t' + \frac{t_n}{\sigma'}, \sigma' \right),$$  

(Appendix A.1)

for $n = 0,1,\ldots N - 1$. The parameters $t'$ and $\sigma'$ are chosen so that

$$\max_{n \in \{0,\ldots,N-1\}} \psi_n(t',\sigma') \leq \varepsilon,$$  

(Appendix A.2)

for some predefined tolerance threshold $\varepsilon > 0$. Define $\Delta' \equiv (\sigma')^{-1} \Delta$ so that $\psi_n(t',\sigma') = \sigma' p_{M-L} (t' + n\Delta', \sigma')$. Consider a choice of $t'$ and $\sigma'$ so that $t'/\sigma' = 1$, then

$$\max_{n \in \{0,\ldots,N-1\}} \sigma' p_{M-L} (t' + n\Delta', \sigma') = \max_{n \in \{0,\ldots,N-1\}} \frac{1}{\Gamma(1 + n\Delta') E_{\Delta'}(1)}. $$  

(Appendix A.3)

A well-known property of the Gamma function is that it achieves a global minimum in $\mathbb{R}_+$, which is $\Gamma(x_{\text{min}}) = 0.885603$ (for $x_{\text{min}} = 1.461632$). Consequently,

$$\max_{n \in \{0,\ldots,N-1\}} \psi_n(t',\sigma') \leq \frac{1}{0.88E_{\Delta'}(1)}$$  

(Appendix A.4)

so that $\sigma'$ is chosen to ensure that

$$E_{\Delta'}(1) \geq \frac{1}{0.88\varepsilon}.$$  

(Appendix A.5)

We now append $\psi(t',\sigma')$ to $\Phi$ (as a column to the right) and set $\theta = [\theta^*^\top \ 1 - \sum_{m=0}^{M-1} \theta_m^*]^\top$. First, notice that the choice of $\sigma'$ above does not depend on $\theta^*$ (but depends exclusively on the discretization interval $\Delta$). Therefore, this calculation can be performed offline. Since $\theta_M = 1 - \sum_{n=0}^{M-1} \theta_m^* < 1$, by design, we know that the contribution of $\psi(t',\sigma')$ to $\overline{p}$ is smaller than $\varepsilon$ (since its contribution to all support values $\{t_n\}_{n=0}^{N-1}$ is smaller than $\varepsilon$). This motivates restricting attention to LASSO constrained to the positive orthant (vs. the probability simplex).

References


Ji, Y., Zhang, H., 2013. Travel time distributions on urban streets: Estimation with hierarchical Bayesian mixture model and application to traffic analysis with high-resolution bus probe data, in: 92nd Annual Meeting of the Transportation Research Board.


Pu, W., 2011. Analytic relationships between travel time reliability measures. Transportation Research Record: Journal of the Transportation Research Board 2254, 122–130.


Yang, F., Yun, M., Yang, X., 2014. Travel time distribution under interrupted flow and application to travel time reliability. Transportation Research Record: Journal of the Transportation Research Board 2466, 114–124.