Addressing Redistribution in Bikeshare Using Cox Processes and Online Supervised Learning

Walter Dempsey
University of Chicago
5734 S. University Avenue
Chicago, IL
wdempsey@uchicago.edu

Jette Henderson
The University of Texas at Austin
Austin, TX 78712
jette@ices.utexas.edu

ABSTRACT

A bicycle sharing system (bikeshare) is a service in which individuals can check-out bicycles from stations for a short period of time and drop them off at any of the stations in the system. The main purpose of bikeshare is to provide transportation for short trips, and it has become common in many cities including New York, Paris, and Beijing. One of the issues with bikeshare is the possible imbalance of bikes across stations, specifically that some stations may have no bikes to check-out while other stations have no slots for arriving bikes. Many cities employ redistribution techniques on an ad hoc basis. In order to improve the redistribution process, we must estimate the expected number of bikes at a station at some point in the future, as well as the chance that the station becomes empty or full in that time window. To do this, we model the arrival and departure of bikes as a log Gaussian Cox Process. We build a block approximation of the underlying Gaussian processes by clustering neighborhoods, which will allow for parallel estimation. Moreover, we build online methods for updating the parameters from streaming station data. In order to assess performance, we apply these techniques to the Washington, D.C., bikeshare data. We end with a brief discussion of how our approach can be used in analyzing new stations without sufficient data and as well as new networks/cities.

General Terms

Log Gaussian Cox Processes, Block-Approximate Gaussian Processes, Parallel Algorithms, Online Learning

1. INTRODUCTION

Bicycle Sharing Systems have become ubiquitous over the past several years, providing an alternative mode of transportation and improving the connectivity of their respective cities. These networks, which are typically referred to as "bikeshare," consist of a number of stations distributed throughout the city with individuals able to check-out bicycles from stations for a short period of time before dropping them off at another station in the system. A common issue with bikeshare is that traffic patterns result in bike imbalance across stations, with some stations being either empty or full. We call these **extreme** stations. These imbalances can lead to delays as bikeshare users must find alternate stations, while also lowering overall ridership due to reliability concerns. Bikeshare systems tackle this issue by some form of redistribution; however, the methods for redistribution of bikes are ad hoc and may be sub-optimal. We propose a method that will give bikeshare companies a tool with which they can proactively approach the problem of redistribution. Use of the tool will help bikeshare companies anticipate extreme stations before they arise, which may ultimately help users view bikeshare as a reliable and consistent form of public transportation.

For a given time in the future and a station, we aim to predict the number of bikes at the station as well as the probability that the station will become extreme at some point in this time window given its current state as well as auxiliary variables. This will allow us to predict extreme stations as well as provide an estimated optimal number of bikes that should be at that station to avoid extreme events in the next time window. This foresight will allow bikeshare companies to anticipate extreme stations and keep them from occurring.

We model arrival and departure of bikes at each station as a pair of correlated Poisson Processes. We propose then modeling the entire bikeshare system as a set of correlated Poisson Processes, where two pairs are correlated based on their spatial proximity. This set is therefore modeled as a log-Gaussian Cox Process in which we capture station-level, temporal, and spatial random effects.

This approach is computationally intensive, and therefore we propose approximation methods for estimation, which allow us to run estimation in parallel and provide necessary speedups at the cost of some inexactness. The bikeshare data analyzed is also streaming, and therefore we provide an online algorithm that updates parameter estimates given the new observations. This approach allows for updating models given new data without re-fitting the entire model every time. We end with an application of our techniques to Washington, D.C., bikeshare data. We analyze the performance of our method and provide methods for prediction of new stations given estimation from surrounding stations.

2. ARRIVAL AND DEPARTURE PROCESSES AT A SINGLE STATION

We start by considering the temporal process at a given station. Let $Y_i(t)$ denote the number of bikes at the station at time t. The bike count is an integer-valued process with jumps. We use $Y(t)$ to define a pair of counting processes that we then model.

The **arrival counting process**, $N_i^a(t)$ can be defined as the number of bikes arriving at the station up to time t. Assuming a finite number of bike arrival times in any interval, we can then define these counting processes as:

$$N^a(t) = \sum_{s<t} |Y_i(s) - Y_i(s-)| \, \mathbf{1}\left[Y_i(s) - Y_i(s-) > 0\right] \quad (1)$$

The **departure counting process**, $N_i^d(t)$ can be defined likewise for bikes leaving the station. A consideration is that a station which is empty cannot have departures. Therefore, $Y_i(t) = 0$ in some interval $[t, t+\delta]$ implies $N_i^d(t+\delta) - N_i^d(t) = 0$. Let A_d be the set of all such windows. Then, we assume the departure counting process to be an in-homogenous Poisson point process on the complement of this set, $\mathbb{R}^+ - A_d$. We consider the same to be true of the arrival process and the window of all times for which the station is full, A_f.

For now, we ignore these constraints on the process. We assume that at a station, the arrival and departure counting processes are correlated. This implies that we can model the pair as a Cox process where the rate parameters includes a station block effect. That is, $N_i^a(t)$ ($N_i^d(t)$ respectively) are poisson processes with rate parameter $\lambda_i^a(t)$ ($\lambda_i^d(t)$ respectively) where

$$\log\left(\lambda_i^a(t)\right) = \mu_i^a(t) + b_i \quad (2)$$

where b_i is a random station effect. The rate parameter for the departure process is defined similarly.

3. BIKESHARE SYSTEM

We now consider the complete bikeshare system. Stations in close geographic proximity provide auxiliary information which should be leveraged. For example, if a station has many arrivals in a given morning, this is a strong indicator that the number of arrivals at a neighboring station will also be high. Figure 1 illustrates this behavior of stations that are in close proximity to one another. These two stations are .05 miles apart, and on average, we can see that they have similar behavior. We see the same general trends when looking at a randomly chosen subset of days for one of the stations in Figure 2.

We wish to introduce the spatial dependency into our current model because of this correlation

A log-Gaussian Cox Process captures this spatial dependence through random effects in the stochastic rate parameters. Let $N_i(t) = (N_i^a(t), N_i^d(t))$ be the pair of counting processes for station i. Then we can assume that the arrival rate parameters are associated as are the departure rate parameters such that

$$\log\left(\lambda_i^a(t)\right) = \mu_i^a(t) + b_i + Z(t; i) \quad (3)$$

Figure 1: Average number of bikes at two Capital Bikeshare stations closely located to one another
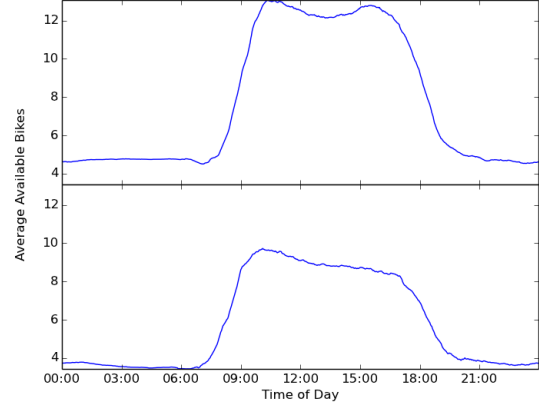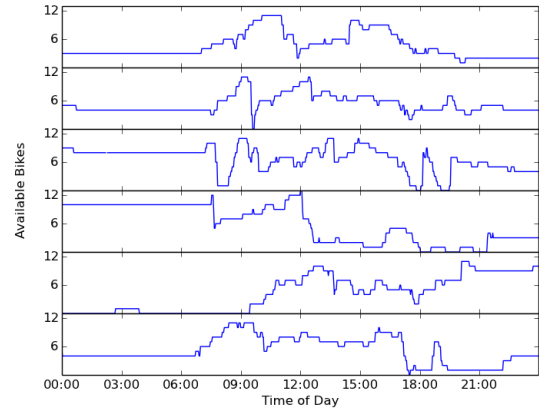

Figure 2: Number of bikes at one Capital Bikeshare station over the course of six randomly chosen days

where $Z(t; i)$ is a mean-zero gaussian process with covariance that may depend on both time t and space i. We can incorporate the station effects into the gaussian process model. We then consider a simple separable variance components model :

$$\mathrm{cov}(Z(t; i), Z(t'; j)) = \sigma_0^2 \delta_{tt'} \delta_{ij} + \sigma_1^2 \delta_{ij}$$
$$+ \sigma_2^2 \exp\left(-\frac{d(i,t),(j,t))}{\lambda}\right)$$

where $d((i,t),(j,t))$ is a specified distance metric. In our work we consider the l_1 distance. The first term is measurement error at time t at station i, the second is the random station effect, and the third is a general mean associated with all the arrival parameters. See Moller et al. [7] for additional details on the Log Gaussian Cox Process.

Covariates can be chosen to account for temporal trends. We can choose a factor model for $\mu_i^a(t)$ that models fixed effects attributable to time of day, month, and year, or a basis expansion in terms of low-order Fourier terms. In this case, the distance metric will only depend on the location of

the stations and the third term will only be included when t and t' are equal.

We assume a similar model for the departure rates. The random block effects link the arrival and departure processes at a given station, and the third term in the gaussian process links the rate parameters for the arrival processes across stations. The departure processes across stations are linked via a similar term, albeit different parameter values, (σ_3^2, λ').

4. MAXIMUM LIKELIHOOD ESTIMATION

Fast gaussian process methods for point process intensity estimation have been studied (see Cunningham et al. [3]). We outline a different approach based on combining approximation likelihood with block-approximation of the gaussian process covariance function, and optimization via stochastic gradient descent methods.

Suppose that there are n total stations. Now take the observations at a particular time window, $(t, t + \delta)$ (we model half-hour windows). The general form of the Cox Process Likelihood associated with the data,
$Y(t) = \left((Y_1^a(t), Y_1^d(t)), \ldots, (Y_n^a(t), Y_n^d(t)) \right)$ is:

$$l_t(\theta; Y(t)) = \int_\Lambda P(Y, \Lambda | \theta) d\Lambda$$

$$= \int \prod_{i=1}^n \frac{\lambda_i^a \lambda_i^d}{Y_i^a Y_i^d} \frac{e^{-\lambda_i^d - \lambda_i^a}}{(2\pi)^n |\Sigma|^{1/2}} \exp\left((\lambda - \mu(\theta))^T \Sigma(\theta)^{-1} (\lambda - \mu(\theta)) \right) d\lambda$$

$$= E_{\lambda|\theta}[l^\star(\lambda; Y(t))]$$

where $l^\star(\lambda; Y)$ is the likelihood for the inhomogeneous Poisson process, and λ is the vector of rate parameters for the dual processes at each station. As we incorporate temporal trends through the the model for the mean, μ, the complete likelihood is the sum over all time windows. Supposing we have T total time windows, $(t_1, \ldots, t_T)$, then the complete likelihood is:

$$l(\theta; Y) = \sum_{i=1}^T E_{\lambda|\theta}[l^\star(\lambda; Y(t_i)]$$

In our setting, the likelihood for each time window consists of estimating an expectation over the finite-dimensional distribution of λ. The number of stations is of the order of several hundred. Washington, D.C. ,has roughly 250 stations, and therefore the dimension of our expectation is approximately 500. The high dimensionality of the integration appears formidable.

4.1 Monte Carlo Estimation

Standard Monte Carlo methods would employ the estimate:

$$l_{MC}(\theta) = s^{-1} \sum_{j=1}^s l\left(\theta | Y, \lambda^{(j)} \right) \tag{4}$$

where $\lambda^{(j)}$ are simulated realizations of λ. This is highly inefficient. Based on the work of Geyer (1999) [5] and Diggle et al. (2013) [4], we use the following to provide robust, efficient estimation of the log-likelihood function. Given un-

normalized joint density of X and λ, $f(X, \lambda|\theta)$, then

$$\hat{L}(\theta) = \log \left\{ s^{-1} \sum_{j=1}^s r\left(Y, \lambda^{(j)}, \theta, \theta_0 \right) \right\} \tag{5}$$

$$- \log \left\{ s^{-1} \sum_{j=1}^s r\left(X^{(j)}, \lambda^{(j)}, \theta, \theta_0 \right) \right\}$$

where $r(X, \lambda, \theta, \theta_0) = f(Y, \lambda|\theta)/f(Y, \lambda|\theta_0)$. The result provides a Monte Carlo approximation to the log-likelihood function, and consequently the maximum likelihood estimate, $\hat{\theta}$, by simulating the process at a single value, θ_0. The accuracy is a function of the number of simulations, s, and the proximity of θ_0 to θ. Assuming n stations, the density is given by:

$$\left[\prod_{i=1}^n \frac{(\lambda_i^a)^{Y_i^a}(\lambda_i^d)^{Y_i^d}}{Y_i^a! Y_i^d!} e^{-\lambda_i^a - \lambda_i^d} \right]$$
$$\times \frac{1}{\sqrt{2\pi}^n |\Sigma|^{1/2}} \exp\left((\lambda - \mu(\theta))^T \Sigma(\theta)^{-1} (\lambda - \mu(\theta)) \right)$$

Therefore, the ratio simplifies to:

$$r(Y, \lambda, \theta, \theta_0) = \left(\frac{|\Sigma(\theta_0)|}{|\Sigma(\theta)|} \right)^{1/2}$$
$$\times \frac{\exp\left((\lambda - \mu(\theta))^T \Sigma(\theta)^{-1} (\lambda - \mu(\theta)) \right)}{\exp\left((\lambda - \mu(\theta_0))^T \Sigma(\theta_0)^{-1} (\lambda - \mu(\theta_0)) \right)}$$

The pair $(Y^{(j)}, \lambda^{(j)})$ are simulated joint realizations of X and λ at $\theta = \theta_0$. In the first term, X is held fixed and $\lambda^{(j)}$ are conditional on X. Conditional simulation of λ requires Markov Chain Monte Carlo (MCMC) methods. Design, computational issues, and design are detailed by Diggle et al. (2013). Given a current estimate, $\lambda^\star$, we use a proposal distribution which is a random-walk on the $\log(\lambda^\star)$.

$$\log(\lambda) \sim N\left(\log(\lambda^\star), c \cdot \Xi \right)$$

where $\Xi = -E[I(\lambda^\star)]^{-1}$. This is given by $\left[\text{diag}(\lambda) + \Sigma^{-1} \right]^{-1}$. The term c controls the acceptance rate for a random walk proposal so that it is tuned to around 0.234 percent.

Algorithm 2 outlines pseudocode for finding the maximum likelihood estimate, $\hat{\theta}$.

Algorithm 1 Approximate Cox Process Estimation

$\mathbf{ApproxCP}(Y, s, \theta_0)$
for j in $1 : s$ **do**
    Generate $(\lambda^{(j)}, Y^{(j)})$ at $\theta = \theta_0$
    Generate $\lambda_2^{(j)}$ given Y at $\theta = \theta_0$ via MCMC
end for
Estimate $\hat{L}(\theta)$ by equation (5)
$\hat{\theta} \leftarrow \arg\max \hat{L}(\theta)$ given θ_0

It only remains to choose the estimate $\theta_0 = (\mu_0, \sigma_0^2)$. We estimate the parameters for each Poisson process independently to get initial estimates for the mean parameters, μ_0. We take the simple average of the estimated mean squared errors for each station, $\bar{\sigma}_i^2 = (\bar{\sigma}_{i,a}^2, \bar{\sigma}_{i,d}^2)$. This provides an estimate for $\text{var}(Y)$ across stations. We use the relationship between the variance of the observed counts, and the

variance of the stochastic rate parameters. Since we have assumed the variance to be stationary, we use the conditional variance formula to show that:

$$\text{var}(Y) = \text{E}\left[\text{var}(Y|\lambda)\right] + \text{var}(\text{E}[Y|\lambda])$$
$$= \text{E}\left[\text{diag}(\lambda)\right] + \text{var}(\lambda)$$

Then we can use the variability of the count data, and subtract the rate parameters from the diagonal to approximate $\text{var}(\lambda)$. This implies that if we build a semi-variogram, we can use these to estimate σ_1^2 and λ. Moreover, we can estimate the station-level block effects by matching the arrival and departure average mean square errors at the station level and measure the correlation, $\hat{\sigma}_1^2 = \frac{1}{n}\sum_i(\bar{\sigma}_{i,a}^2 \cdot \bar{\sigma}_{i,a}^2)^2$. We can then create the initial estimate for the measurement error by

$$\sigma_0^2 \approx \frac{1}{n}\sum_i\left[\sigma_{i,a}^2 - \lambda_{i,a} + \sigma_{i,d}^2 - \lambda_{i,d} - 2\cdot\left(\sigma_1^2 - \sigma_2^2\right)\right]$$

5. A PARALLEL APPROXIMATE ESTIMATION ALGORITHM

The assumed dependence among all counting processes makes maximum likelihood estimation computationally intensive. This is due to the covariance being positive between any two stations. Therefore, we must model the entire system of counting processes simultaneously and the dimension of the gaussian process becomes prohibitively large. For example, there are over 100 loading stations in Chicago for which we have almost a year of data. If we disaggregate the data to half-hour windows, the complete set of observations is over one-million. Given the computational cost of the high-dimensional integral, we cannot hope to estimate our model in a practical amount of time.

We start by assuming we capture the temporal dependence through a factor model. However, we still must worry about the spatial dependence as the dimension would still exceed $10,000$. However, the dependency, while positive, is weak for stations that are well-separated. Therefore, we can approximate the true underlying gaussian process by a block-approximate gaussian process, which exploits the weak dependence between stations that are far apart. Alternative methods exist such as the low-rank covariance matrix approximations (see Chen et al. [2]). However, Stein [8] shows that block-approximation often provides much better approximation to the likelihood than a low rank approximation.

5.1 Hierarchical Clustering

In order to approximate the gaussian process, we must determine the independent station blocks. This requires partitioning the entire set of stations, $\Omega = \{\mathbf{z}_i = (x_i, y_i), i = 1,\dots S\}$, where (x_i, y_i) are the latitude and longitude for station i. The resulting partition must capture high in-block and low between-block covariance.

We assumed that the spatial covariance between stations is a decreasing function of the Euclidean distance between the two points. In particular, we describe the relationship via a squared exponential covariance function, $K(\|\mathbf{z}_i - \mathbf{z}_j\|) = \sigma_3^2 \exp\left(-\|\mathbf{z}_i - \mathbf{z}_j\|^2/\lambda\right)$, where λ is a range parameter. This

is the limit of the Matern covariance function, a set of stationary covariance functions typically used in spatial statistics. If we knew the range parameter, λ, a priori then the partitioning of Ω can reflect the range of the spatial process. However, the range is typically unknown before fitting the model, and therefore cannot be used to build the partition.

We therefore perform k-means clustering using l_1 distance. The choice of k clusters reflect true neighborhood clustering; moreover, a larger k implies weaker approximations, but faster estimation, while a smaller k implies the opposite. We therefore choose k to balance the computational-approximation tradeoff.

Figure 3 shows the hierarchical clustering applied to the Washington D.C. bikeshare stations. The clusters reflect the well-defined neighborhoods for the most part. We focus our efforts on the clusters that most closely match well-defined neighborhoods.

Figure 3: Hierarchical K-means Clustering of Bikeshare Stations in Washington, D.C.
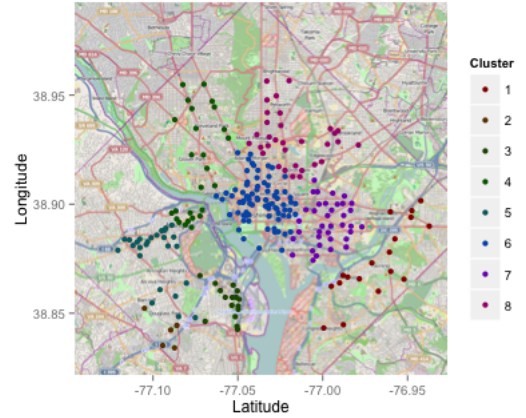
We employ the clustering without knowledge of the range parameter; however, once we have the maximum likelihood estimate, $\hat{\theta}$, we will have an estimate of the range parameter, $\hat{\lambda}$. The k-means clustering algorithm does not consider linear scaling of the distance function. Therefore, if we simply re-estimate the clusters using the modified Euclidean distance, $\|\mathbf{z}_i - \mathbf{z}_j\|/\hat{\lambda}$, we will achieve the same partition. However, we can now use the range estimate, to choose k more.

5.2 Parallel Estimation

The block-diagonal approximation lightens the computational burden of the Monte-Carlo Simulations, as the conditional simulation of λ conditional on Y can be done in parallel. However, the likelihood does not factorize by cluster under the current assumptions. The approximate likelihood, $\hat{L}(\theta)$, does become a sum over independent clusters, but the assumption of common variance components means that estimation cannot be done in parallel across clusters.

If we assume that the variance components are distinct for

each cluster, then we can estimate parameters separately for each cluster by exploiting variational independence. This leads to additional computational savings as we can then parallelize the entire estimation procedure. If the true variance components are identical for each cluster, then the estimated variance components for each cluster will be unbiased estimators. Therefore, we only sacrifice slight losses in efficiency for gross gains in speed.

Algorithm 2 Approximate Likelihood Algorithm

---

 for i in stations **do**
  $\mu_i \leftarrow$ Fit marginal model i
 end for
 Obtain initial estimate for $\sigma_0^2 = \text{mean}\{\sigma_{0,j}^2\}$
 Obtain initial estimate for $\sigma_1^2 = \sigma_2^2 = \sigma_0^2/2$
 $U_1, U_2, \ldots U_k \leftarrow \text{cluster}(k, \text{stations})$
 for j in $1 : k$ **do**
  $\theta_j \leftarrow \text{ApproxCP}(Y_j, s, \theta_0)$ for Cluster U_j
 end for

Given the approximate maximum likelihood estimate, θ, we now turn our attention to estimating the desired quantities.

6. PREDICTION

For a particular station i at a particular time t, we know the number of bikes at the station, $Y_i(t)$. We are interested in estimation of the expected number of bikes at some future time point, $t + \delta$. To answer the question of redistribution, we also need to know the probability that the number of bikes reaches an extreme value in the time window $(t, t + \delta)$. Therefore, assuming a maximum number of spots τ, we want to estimate $\text{E}\left[Y(t + \delta)|Y(t), \hat{\theta}\right]$ and

$$P[Y(s) = 0 \text{ or } Y(s) = \tau \text{ for some } s \in (t, t + \delta)|Y(t), \hat{\theta}].$$

6.1 Expected Number of Bikes

Our block approximation from section 5 allows estimation of both quantities to be done separately for each cluster. Suppose station i sits inside cluster $U \subset \Omega$. Let the expected rate parameter at time s given $(Y, \hat{\theta})$ to be given by $\hat{\lambda}(s)$. Then the expected number of arrivals in the window is given by $\int_t^{t+\delta} \hat{\lambda}_a(s)ds$. The expected number of departures is the same formula with the integrand replaced by $\hat{\lambda}_d$. Then we can estimate the expected number of bikes at the station by:

$$\text{E}_{\hat{\theta}}\left[Y(t + \delta) \mid Y(t)\right] = Y(t) + \int_t^{t+\delta} \hat{\lambda}_a(s)ds - \int_t^{t+\delta} \hat{\lambda}_d(s)ds$$

To calculate the expected rate parameter at time s, we need to calculate the probability of the rate parameter given the maximum likelihood estimate and the observed counts. The predicted rate parameter is given by:

$$P[\lambda_i(s)|Y, \hat{\theta}] \propto \int P[Y|\lambda(s), \hat{\theta}]P[\lambda_i(s)|\hat{\theta}]\prod_{j \neq i} P[\lambda_j(s)|\hat{\theta}]d\lambda_j(s)$$

We assume that the rate parameter is constant in each half-hour time window. Let $N_i^a(t, t+\delta)$ be the number of arrivals in the time window starting at time t at station i. Then let $N^a(t, t + \delta)$ be the vector of arrivals for stations in cluster U. Then we can estimate the rate parameter for window

$(t, t + \delta)$, λ by

$$\int_0^\infty \lambda P[\lambda|N^a, \hat{\theta}]d\lambda$$

We calculate these expectations by Monte Carlo estimation. The expected number of bikes is then a simple difference of the Monte Carlo estimated expected arrival and departure rate parameters added to the total number of current bikes. Because the intervals are independent, the expected number of bikes in the future is just the sum of independently estimated rate parameters for each disjoint window.

6.2 Probability of Station Being Extreme

To calculate the probability of station j being empty in the time window (denoted $P_j(t, t + \delta)$) requires a simulation-based approach. Suppose we are interested in the probability over the smallest window, $(t, t + \delta)$. We start by generating the constant arrival and departure rate parameters in the window for station j in U, λ_j^a and λ_j^d respectively.

Then for a particular station, we can wish to simulate the arrivals and departure of bikes. Given the rate parameters, the arrival and departure processes are independent, and, therefore, the superposition principle implies that the first event has an exponential waiting time, T_1, with rate parameter $\lambda_j = \lambda_j^a + \lambda_j^d$. The probability that the first event is an arrival is λ_j^a/λ_j. If $T_1 < \delta$, then we have an event in the time window. If that event was an arrival, then we set $Y(t + T_1) = Y(t) + 1$. If the event was a departure, then we subtract one from the current number. We then continue this process until $\sum_k T_k > \delta$. We then have a random instance of $\{Y_j(s) \, s \in (t, t + \delta)|Y_j(t)\}$. We do this for each station j in U.

The number of bikes is bounded to be in $[0, \tau_j]$, where τ_j is the maximum number of slots at station j. Therefore, if the number of bikes is zero in the time window, we cannot generate a departure. The same is true for a full station and generating arrivals. We can handle these constraints in two ways. We can continue to generate these departures and arrivals but ignore those that push the process outside of its boundaries. By the thinning principle, this is equivalent to only generating arrivals at the rate λ_j^a when the station is empty or departures at the rate λ_j^d when the station is full. On the other hand, instead of ignoring the arrivals, we can send these arrivals to nearby stations. This will be more accurate; however, we would then need to estimate the travel time to the nearest station that is not full. We do not currently employ such methods, as they add a serious additional computational obstacle. However, our current framework is highly adaptable, and we could incorporate this approach.

Let $\Delta_j^{(i)}$ be an indicator that the number of bikes at station j reaches an extreme position in the time window for the ith instance from our simulation. Then we estimate the probability by the simple average:

$$\hat{P}_j = \bar{\Delta}_j = s^{-1} \sum_{i=1}^s \Delta_j^{(i)}$$

These are simultaneously estimated for each station in the cluster. We can therefore estimate the probabilities in parallel for each cluster, U_1 through U_k.

6.3 Proper Distribution of Bikes

The above provides detailed information that will allow bikeshare companies to redistribute bikes proactively before stations become empty or full. We now address the question of appropriate distribution of bikes. That is, before peak commuting times, what is the correct distribution of bikes across stations in a city that will minimize the chance of a station becoming extreme?

Let Y be a vector containing the number of bikes at each station in the city. Then we can calculate the probability of each station becoming extreme in the desired time window, P. For example, we may be interested in the time window that lasts from one hour before rush hour to several hours after rush hour (e.g., 6A.M to 12PM). We then assume some objective function of the probabilities, $f : P \to \mathbb{R}$. For instance, we may be interested in minimizing the maximum, $f(P) = \max_j P_j$, or in minimizing the sum of squares, $f(P) = \sum_j P_j^2$.

The goal is then to find the best allocation of bikes for the city with the constraint that the total number of bikes in the system is fixed, $\|Y\|_1 = C$. In particular,

$$\max f(P)$$
$$\text{s.t. } P = \{P[Y_j(s) \text{ extreme for } s \in (t, t+\delta)|Y_j(t)\}$$
$$\text{and } \|Y\|_1 = C$$

We can add additional constraints that may be necessary in practice. For example, we may require a certain number of bikes per neighborhood. The search space is quite large. Given C total bikes and N total stations, the search space is of size N^C. Therefore, given f and Y, we propose a simple simulated annealing approach to finding the global optimum. Details are given by Bertsimas and Tsitsikis [?].

The annealing approach requires an initial state. We assume that the system currently is working at least approximately close to an optimum. Therefore, we take Y_0 to be approximately the average number of bikes at each station at time t. Simulated annealing produces $\hat{Y}$. As we only care about distribution at a few distinct times, we only need to calculate optimum allocation several times.

Slight changes to the objective function may result in drastically different distributions of bikes across the city. It is necessary to investigate this behavior. For example, we can have the objective function take into account the rate parameters, penalizing empty and busy stations more than empty and idle ones. This does not increase the search space as the rate parameters are not functions of Y; however, it does add additional computational burdens.

The optimum allocation minimizes the redistribution though out the day, saving resources and time.

7. ESTIMATION AND PREDICTION FOR A NEW STATION

Many bicycle sharing systems grow over time. In Chicago, over 50 new stations have been added in the past few months. Even in cities with long-standing bikeshare systems, new stations are added as citizens of neighborhoods continue to request additional capacity. It is therefore necessary to be able to estimate parameters for new stations even before we have much data on the new station.

Let $z = (x, y)$ denote the latitude and longitude of the new station. Then we calculate the distance from the new station to the centroids of the various clusters discussed in Section 5.1. We then assume that the new station will be added to the cluster such that the centroid of the cluster is closest in Euclidean distance.

Within this cluster, we calculate a weighted average of the parameters at each station, θ_j, where the weights are proportional to the exponential squared covariance function. We use this as an initial estimate

$$\hat{\theta} = \frac{\sum_j \exp\left(-\frac{|z - z_j|}{\lambda}\right) \theta_j}{\sum_j \exp\left(-\frac{|z - z_j|}{\lambda}\right)} \tag{6}$$

where the sum is over all stations in the assigned cluster. Prediction then occurs in the same manner, where we employ Kriging methods for interpolation of the spatial process to the new station location.

We may have a point on or near the boundary. In that case, we can then assign the new station to each cluster with probability proportional to the exponential squared distance from the centroids. Then the above estimate is denoted $\hat{\theta}_i$ for cluster U_i and the estimated parameters become the weighted average of the above estimates. The predictions are handles in a similar way, and prediction for each cluster is done separately and then compiled by weighted averaging.

Once we have collected enough data for the station, we can then revert to standard estimation procedures. Section 8 outlines a different approach to the above, and provides additional tools for estimation of parameters given new data is continually streaming.

8. ONLINE LEARNING: ALGORITHM AND PREDICTION

Unlike other spatial-temporal datasets, the bikeshare datasets are continually updated each day. Each day generates an observation of the underlying stochastic process. Unfortunately, the methods outlined above are computationally expensive and too slow to be re-fit on a daily basis. Moreover, if we assume that the system is in equilibrium (meaning parameter values are constant and not changing) then once we have enough data, the additional observations will only be marginally useful and continuing to fit the model will be unnecessary.

However, the bikeshare systems do change over time as does their use. Additionally, for new stations, we would like a more natural manner for switching from the weighted average of neighbor estimates to standard estimation of the parameters. We outline such an approach below.

Our assumption that the temporal trends are captured in the mean model implied that the stochastic process over disjoint time windows is independent. The likelihood is then factorized into a sum over these disjoint time windows. When the likelihood factorizes into the sum of differentiable functions,

stochastic gradient descent (SGD) methods can be used for finding the maximum likelihood estimate (see Bottou [1] for further details of stochastic gradient descent in large-scale machine learning).

In stochastic gradient descent, the true gradient of the function $\frac{\partial}{\partial \theta} L(\theta)$ is approximated by the gradient at a single time window, $(t, t + \delta)$. Updates are then given by:

$$\theta_{t+1} \leftarrow \theta_t + \gamma_t \nabla L_t(\theta) \tag{7}$$

where γ_t is the step size, and L_t is the likelihood given the observations at time t. Also, θ_t is the estimate of the parameter by employing SGD using data in sequence up to time t. We start with the same initial estimates as described in Algorithm 2. We hope that this algorithm behaves like its expectation, despite the noise introduced by using only one time window at each step. We cycle through the time windows until convergence is reached. Convergence results typically require the step size to be a decreasing sequence

$$\sum_t \epsilon_t^2 < \infty \quad \text{and} \quad \sum_t \epsilon_t = \infty \tag{8}$$

We start with the first time window and work forward, shrinking the step size as a rate of $O(n^\alpha)$ for $\alpha < \frac{1}{2}$ to ensure convergence. We approximate the derivative of the likelihood based on the Monte Carlo approach outlined in Section 4.1. Given the current estimate, $\hat{\theta}$ we can then update the parameters for each day of data without having to refit the entire model.

This requires calculating the derivative of the likelihood. We instead calculate the derivative of the approximate log-likelihood.

$$\frac{\partial}{\partial \theta} \hat{L}(\theta)) = \frac{\sum_{j=1}^s \frac{\partial}{\partial \theta} r\left(Y, \lambda^{(j)}, \theta, \theta_0\right)}{\sum_{j=1}^s r\left(Y, \lambda^{(j)}, \theta, \theta_0\right)} - \frac{\sum_{j=1}^s \frac{\partial}{\partial \theta} r\left(Y^{(j)}, \lambda^{(j)}, \theta, \theta_0\right)}{\sum_{j=1}^s r\left(Y^{(j)}, \lambda^{(j)}, \theta, \theta_0\right)}$$

The last step is to find the derivative with respect to θ of $r(Y, \lambda, \theta, \theta_0)$. We assume that the we have a linear mean model, $\mu(\theta) = X\beta$, and a variance components model, $\Sigma = \sum_k \sigma_k^2 K_k$. The derivative with respect to β and σ_j^2:

$$\frac{\partial}{\partial \beta} r(X, \lambda, \theta, \theta_0) = \frac{X^T \Sigma^{-1}(\lambda - X\beta)}{\exp\left((\lambda - \mu(\theta_0))^T \Sigma(\theta_0)^{-1}(\lambda - \mu(\theta_0))\right)}$$

$$\frac{\partial}{\partial \sigma_j^2} r(X, \lambda, \theta, \theta_0) = \frac{(\lambda - X\beta)^T \Sigma^{-1} K_j \Sigma^{-1}(\lambda - X\beta)}{\exp\left((\lambda - \mu(\theta_0))^T \Sigma(\theta_0)^{-1}(\lambda - \mu(\theta_0))\right)}$$

The numerator in both cases is the typical derivative one would see for normal distributions. The denominator is constant, and it implies we can further simplify the derivative

of the approximate likelihood:

$$\frac{\partial}{\partial \beta} \hat{L}(\theta)) = \frac{\sum_{j=1}^s X^T \Sigma^{-1}\left(\lambda^{(j)} - X\beta\right)}{\sum_{j=1}^s \exp\left((\lambda^{(j)} - X\beta)^T \Sigma^{-1}(\lambda^{(j)} - X\beta)\right)}$$
$$- \frac{\sum_{j=1}^s X^T \Sigma^{-1}\left(\lambda^{(j)} - X\beta\right)}{\sum_{j=1}^s \exp\left((\lambda^{(j)} - X\beta)^T \Sigma^{-1}(\lambda^{(j)} - X\beta)\right)}$$

$$\frac{\partial}{\partial \sigma_j^2} \hat{L}(\theta)) = \frac{\sum_{j=1}^s (\lambda^{(j)} - X\beta)^T \Sigma^{-1} K_j \Sigma^{-1}\left(\lambda^{(j)} - X\beta\right)}{\sum_{j=1}^s \exp\left((\lambda^{(j)} - X\beta)^T \Sigma^{-1}(\lambda^{(j)} - X\beta)\right)}$$
$$- \frac{\sum_{j=1}^s (\lambda^{(j)} - X\beta)^T \Sigma^{-1} K_j \Sigma^{-1}\left(\lambda^{(j)} - X\beta\right)}{\sum_{j=1}^s \exp\left((\lambda^{(j)} - X\beta)^T \Sigma^{-1}(\lambda^{(j)} - X\beta)\right)}$$

again where the first term has the $\lambda^{(j)}$ from simulations of the joint distribution given $\theta = \theta_0$ and the latter term has the $\lambda^{(j)}$ from simulations given the data, Y, and $\theta = \theta_0$.

In Section 7 we introduced the estimation procedure for parameters associated with a new station. The stochastic gradient descent algorithm provides a natural mechanism for moving from this initial estimate toward the maximum likelihood estimate. Furthermore, introduction of a new station may change the rate parameters associated with the neighboring stations. In that sense, SGD is a consistent and straightforward approach when we wish to re-estimate parameters after incorporating new stations. When we introduce a new station, the step size is increased in order to reflect the potential changes to the parameters. However, if there are no further changes to the system, the step size decreases again in the same manner such that conditions (8) are met.

9. EXPERIMENTAL RESULTS

9.1 Washington D.C. Bikeshare

We will test our method using data from the Washington, D.C., bikeshare system. This data set consists of station-level reports of the number of bikes and empty slots every minute. We augmented this data with weather data recorded on an hourly basis.

In addition to the station-level data, we also use data that deals with the redistribution of bicycles by bikeshare employees. Each time a bikeshare employee picks up or drops off bikes at a station, the bikeshare company records the trip. We refer to this redistribution of bicycles as rebalancing. Capital Bikeshare, the bikeshare system in Washington, D.C., provided us with rebalancing data covering a year of rebalancing trips. This data contains the starting and ending stations for a bicycle and the times the bicycle was picked up and dropped off at another station. We used this data to remove rebalancing events from the station-level data. Specifically, if a rebalancing event occurred during a timeframe of interest, we subtracted the number of bikes arriving to the station via rebalancing from the total number of bikes that arrived during the timeframe. We performed a similar procedure, but instead added to the bikes at a station, if bikeshare employees removed bikes from a station.

In our analysis of the Washington, D.C., bikeshare data, a few patterns emerged around high activity times. The figure below anecdotally illustrates these common usage patterns. The purple line is the average number of bicycles at a bikeshare station located in downtown D.C., and the green line is the average number of bicycles at a bikeshare station located in an area of D.C. that is more residential. The averages were taken for every Monday over a year-long period. In the morning, the residential station empties out as people head to work and the downtown station fills up as people arrive at work. This figure demonstrates two potential problems in a bikeshare system, specifically riders can arrive at bike stations to check-in a bike but find no slots to dock their bike or potential riders arrive at a station to check out bikes but there are no bikes available. For this predictive model to be useful, it must perform well during these "high activity" times at the stations. Specifically our goal is to mitigate these situations in high activity times.

Figure 4: Average number of bikes at two Capital Bikeshare Stations taken over every Monday in one year
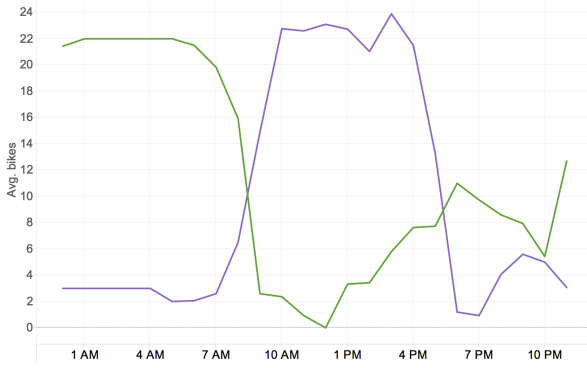
Different usage patterns may manifest in other bikeshare systems. For example, a drastic elevation change in a city could lead to riders using bikeshare on the way downhill but other public transit options on the way uphill (see [6]).

9.2 Performance Indices

We wish to measure the success of our current modeling approach. We were interested in estimating two specific quantities for a given station: (1) the expected number of bikes at some point in the future given the current number of bikes, $E_{j,\theta}(t + \delta|t) = E[Y_j(t + \delta)|Y_j(t), \theta]$, and (2) the probability of the station becoming extreme during this time window, $P_{j,\theta}(t, t + \delta)$.

For station j, at time t with window length δ, we use the mean-squared error of prediction in both cases to measure success of the algorithm:

$$F_1(j, t, \delta) = \sum_{t'=t} \|Y_j(t + \delta) - E_{j,\hat{\theta}}[t + \delta|t]\| \quad (9)$$

$$F_2(j, t, \delta) = \sum_{t'=t} \|\Delta_j(t, t + \delta) - P_{j,\hat{\theta}}(t, t + \delta)\| \quad (10)$$

To assess how well our model predicts the number of bikes

at a station, we used out-of-sample validation with a rolling time window. We used the first year of data to train an initial model. We then used the model fit to predict the next four time steps, where each time step is a 15-minute increment. We also predicted the number of bikes at the station two hours out. Then, we added one day and one hour to our training set, retrained our model on the first year plus that additional day and hour, predicted the number of bikes 15, 30, 45, 60, and 120 minutes out, and compared these predictions to the actual numbers in our dataset.

In Figures 5 and 6, we see that, as expected, the farther out the prediction goes, the more the prediction degrades.

Figure 5: Square root of mean-squared error between predicted and actual number of bikes over time
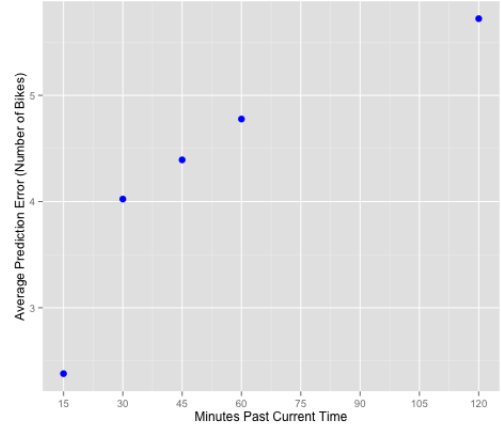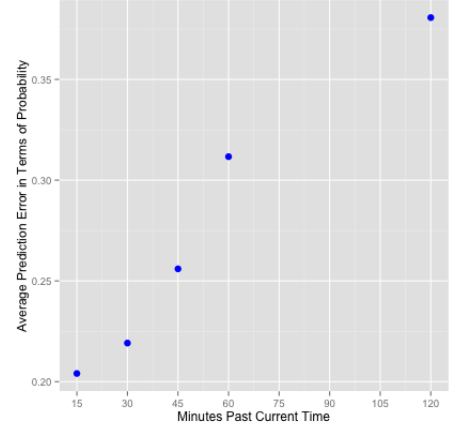

Figure 6: Square root of mean-squared error of probability that station is empty in the next t minutes


We then visualize the distribution of prediction errors generated during the validation. Each graph displays the difference between our prediction and the actual number of bikes for all training and test sets described above. There are 90 such test sets, so each histogram displays 90 error terms.

When predicting 15 minutes in advance, 54% of our predictions are within 1 bike of the actual bikes at the sta-

tion, 82% are within 3 bikes, and 93% are within 5 bikes. These results demonstrate that our method gives a profi-

Figure 7: Error for 15 minute prediction
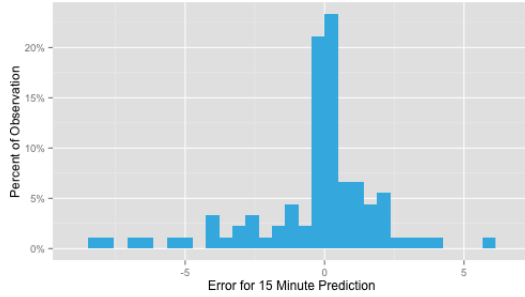
cient on-the-ground-understanding of what will happen at a bikeshare station in the near future.

Another facet of our analysis involves giving bikeshare employees guidance on how many bikes should be at a station at the beginning of a time window. Specifically, we simulate the conditional distribution that the distribution of the number of bikes we'll end up with after two hours given an initial number of bikes at this time of the day. Additionally, given any initial value of bikes, we calculate the probability of a bike station emptying out or filling at any point during the time interval. Then, for any tolerance level, our method suggests the appropriate number of bikes to put at the station at the beginning of the timeframe.

Figure 8: Distribution of bikes at a station given different initial number of bikes at a start time
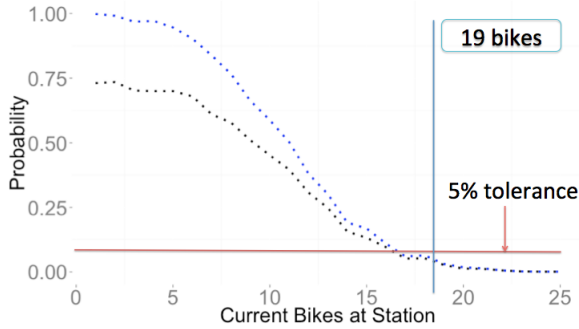

Figure 8 demonstrates this idea. The x-axis is the number of bikes at a station at beginning of the time interval - 7 AM, in this case. The y-axis is the probability that the station will be empty. The blue line is the probability that the station will be empty at any time between 7 AM and 9 AM, as a function of the number of bikes initially at the station. For example, if the station has 10 bikes at 7 AM, the probability that the station will be empty at some point between 7 AM and 9 AM is about 60%. The black line is the probability that the station will be empty right at 9 AM. Using the same example as before, if the station starts with 10 bikes, the probability that it will be empty at 9 AM is about 50%.

Now we can pick a tolerance level and figure out how many bikes we need to put at the station to achieve this tolerance

level. Say that we want to be 95% sure (or have a tolerance of 5%) that the station does not empty out in the next two hours. This plot shows us that we need to initially stock the station with 19 bikes to be 95% sure.

10. SUMMARY
Moving people around efficiently in cities a fundamental problem in urban planning as city populations continue to grow. Bikeshare widens the transportation options for a city's citizens. Bikeshare also has the capacity to make other public transportation options more feasible by connecting people to bus stops and train stations that are too far to get to on foot. It is a creative approach to a modern transportation problem, and using predictive modeling, we show a method to ensure bikeshare companies have a proactive tool to deal with redistribution issues in their network of stations. Currently, we model bike arrivals and departures at a given station and show how correlation with other stations can be incorporated to gain a better understanding of the state of the entire bikeshare system. We presented an online method...

11. ACKNOWLEDGEMENTS

12. REFERENCES
[1] Leon Bottou. Large-scale machine learning with stochastic gradient descent. *COMPSTAT*, pages 177–187, 2010.
[2] Jie Chen, Nannan Cao, Kian Hsiang Low, Ruofei Ouyang, Colin Keng-Yan Tan, and Patrick Jaillet. Parallel gaussian process regression with low-rank covariance matrix approximations. *Uncertainty in Artificial Intelligence*, 2013.
[3] John P. Cunningham, Krishna V. Shenoy, and Maneesh Sahani. Fast gaussian process methods for point process intensity estimation. *International Conference on Machine Learning*, 2008.
[4] Peter J. Diggle, Paula Moraga, Barry Rowlingson, and Benjamin M. Taylor. Spatial and spatio-temporal log-gaussian cox processes: Extending the geostatistical paradigm. *Statistical Science*, 28(4):465–660, 2013.
[5] Charles Geyer. Likelihood inference for spatial point processes: Likelihood and computation. *Stochastic Gradient Geometry*, 1999.
[6] J. H. Lin and T. C. Chou. A geo-aware and vrp-based public bicycle redistribution system. *International Journal of Vehicular Technology*, 2012.
[7] Jesper Moller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998.
[8] Michael L. Stein. Limitations on low rank

approximations for covariance matrices of spatial data.
Spatial Statistics, 2013.