UNIVERSITY OF CALIFORNIA,
IRVINE

Demand Forecasting with Time-Varying Arrival Rates: Modeling and Evaluation with Hotel Data

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Management

by

Alexander C. Robinson

Dissertation Committee:
Professor John Turner, Chair
Professor L. Robin Keller
Professor Amelia Regan

2022

TABLE OF CONTENTS

Page

LIST OF FIGURES **iv**

LIST OF TABLES **v**

ACKNOWLEDGMENTS **vi**

VITA **vii**

ABSTRACT OF THE DISSERTATION **viii**

1 Introduction **1**

2 Background and Literature **6**

2.1 The Expectation Maximization Algorithm 6

2.1.1 Background and History 6

2.1.2 General Formulation 8

2.1.3 Variants and Generalizations 14

2.2 Modeling Censored Demand for Perishable Products 19

2.3 Contributions 24

3 The Censored Demand Problem with Hotel Data **28**

3.1 Model and Data 29

3.1.1 Purchase Probability 29

3.1.2 Arrival Rate Function 30

3.1.3 Data Description 33

3.2 Specification of $\lambda(\tau)$ 34

3.2.1 Convergence and Optimality of SVMH 39
3.2.2 What if your Step Configuration is Wrong? 42
3.3 Estimation Using Expectation Maximization 43
3.3.1 Expectation Maximization 45
3.4 Expectation Maximization Results 47
3.4.1 Train and Test Procedure 48
3.4.2 Results and Discussion 48

4 Regularized Expectation Maximization **51**
4.1 Intuition and Theoretical Properties 51
4.2 REM Convergence and Comparison to EM: Numerical Results 56

5 Conclusions **64**

Bibliography **67**

Appendices **71**

LIST OF FIGURES

Page

1.1 Simple Example Displaying the Perils of Assuming a Constant Arrival Rate. 4

3.1 Example Step Function for a Time-Varying Arrival Rate 32

3.2 The Head of the Real Hotel Data 33

3.3 The Two Types of Moves Possible in SVMH. 37

3.4 Example Run of the SVMH Algorithm 38

3.5 The Expectation Maximization Process 47

3.6 Percentage Difference in MSE between Variable Arrival Rate Models and Constant Arrival Model.
SVMH achieves an 8%-10% improvement over the constant (TvR) arrival rate in-sample, and 6%-8% out-of-sample. 50

4.1 Example Likelihood Plateaus for Each Data Type. x-axis: affine weight (EM: x = 1, in red, REM: x = 0, in green), y-axis: likelihood
EM and REM achieve almost identical likelihoods and lie on a plateau in the incomplete likelihood function. 57

4.2 Iterations to Convergence for Sim3 Dataset over Various REM Weights
Iterations follow an elbow plot with min at $w = 1$ and flat "forearm" 59

4.3 In-Sample and Out-Of-Sample MSE for a Sim3 Dataset over Various REM Weights
Variation in MSE is very slight, though MSE increases at higher weights. . . 60

4.4 Predicted Revenue Curves for Example EM and REM models. REM and EM models show very similar predicted revenue curves. 63

LIST OF TABLES

Page

3.1 Key Variables, Sets, and Parameters Descriptions 30

3.2 Optimality of Results for SVMH, Based on 2500 Random Trials for Each Dataset
SVMH achieves the optimal solution in 4 out of 8 cases, other cases differences are both expected and not significant in estimation. 40

3.3 In-Sample Mean Squared Error for All Datasets by Step Selection Method.
SVMH achieves the lowest MSE in every case. 49

3.4 Out-of-Sample Mean Squared Error for All Datasets by Step Selection Method
SVMH achieves lowest MSE in all cases except Sim1, which has a constant arrival rate. 49

4.1 Average Percentage Decrease in Iterations for REM Models Compared to EM on the Same Data.
REM decreases iterations required for convergence by 81% on average. . . . 58

4.2 Parameter Values for a Sim3 Dataset for Various REM Weights
Parameters become inelastic to weight choice at higher weights after initial change. 60

4.3 Average Percentage Difference in Out-of-Sample MSE between EM and REM Solutions
(positive percentage indicates REM MSE > EM MSE)
REM and EM solutions show negligible performance differences out-of-sample. 62

E.1 Average Percentage Decrease in Convergence Time for REM Models Compared to EM on the Same Data 79

E.2 True Parameters Used to Generate Simulated Data 79

ACKNOWLEDGMENTS

I would like to thank Professors John Turner and Robin Keller for all their guidance and advice throughout my time at UCI. I would also like to thank my parents, whose support and parenting got me to where I am today.

Finally, and most importantly, to Emma, I could not have gotten through this without your endless love, patience, and support. And to Walter, the best dog in the world. I love you both!

VITA

Alexander C. Robinson

EDUCATION

Doctor of Philosophy in Management **2022**
University of California, Irvine *Irvine, California*

Bachelor of Science in Mathematics **2017**
Bucknell University *Lewisburg, Pennsylvania*

RESEARCH EXPERIENCE

Graduate Research Assistant **2017–2022**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2018–2022**
University of California, Irvine *Irvine, California*

WORK EXPERIENCE

Operations Research Scientist **2022–present**
CSX Technology *Jacksonville, Florida*

AWARDS

Outstanding Undergraduate Teaching Assistant **2020-2021 AY**
Paul Merage School of Business, University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATION

Building Insights on True Positives vs. False Positives: Bayes' Rule **2022**
Decision Sciences Journal of Innovative Education, Special Issue: Teaching Data Analytics and Statistics, Volume 20, Issue 4

ABSTRACT OF THE DISSERTATION

Demand Forecasting with Time-Varying Arrival Rates:
Modeling and Evaluation with Hotel Data

By

Alexander C. Robinson

Doctor of Philosophy in Management

University of California, Irvine, 2022

Professor John Turner, Chair

The censored demand estimation problem has been well studied over the past 20 years. The seminal paper is Talluri and Van Ryzin (2004), which models demand as the product of an unobserved arrival rate and purchase probability, and uses Expectation Maximization (EM) for estimation. In this work, I apply their model to the case of a small hotel with a single product, and adapt the model to use a time-varying arrival rate. Modeling the arrival rate as a step function, the choice of where the steps fall is a crucial hyperparameter, so I propose a heuristic algorithm, the Step Variance Minimization Heuristic (SVMH), to specify the size and location of the steps in the arrival rate function. On both real and simulated hotel data, SVMH is superior in terms of both in-sample and out-of-sample mean squared error when compared to (a) a constant arrival rate and (b) evenly-spaced steps of the same number. To improve the convergence time of EM, I introduce a variant algorithm, Regularized Expectation Maximization (REM), that uses 81% fewer iterations than EM, on average. I show that REM is a Generalized Expectation Maximization (GEM) algorithm, and thus has similar theoretical properties to EM. I also conducted multiple numerical studies that show that models estimated using REM perform similarly in-sample and out-of-sample to their EM counterparts, and produce nearly identical revenue curves. Thus, the convergence improvements offered by REM do not come at a cost to good parameter estimation.

Chapter 1

Introduction

Accurate demand estimation is crucial to effectively employing revenue management and price optimization. Finding a revenue-maximizing price requires knowledge about the relationship between price and demand. Estimating the Price Response Function (PRF) that defines this relationship can be a difficult task in certain settings. One particularly difficult setting is when demand is censored in some way, such as when demand is proxied by sales data. Sales data offers an incomplete picture of demand since the lack of a sale does not necessarily indicate a lack of demand.

To illustrate the problem with sales data, consider a brick and mortar store that only sells one product. If a customer enters the store and makes a purchase, we can infer that the price was below the customer's willingness-to-pay. Conversely, if a customer enters the store and does not purchase, we can infer that the price was above their willingness-to-pay (WTP). However, if we watch a customer walk past the store on the street without entering, we can make no inferences about how the price compares to their WTP. The key issue is that sales data only records the presence or absence of a sale, making an arrival without a non-purchase indistinguishable from a non-arrival. This problem is particularly acute in the hospitality

and travel industry, where pricing is done dynamically and the products are perishable. Dynamic pricing means that prices must be set and changed more frequently, making the pricing problem harder. Perishable inventory increases the pressure to price optimally - since the product will expire - and increases the likelihood of demand fluctuations as interest in the product changes over the course of the sales horizon (further literature on dynamic pricing includes Gallego and Van Ryzin (1994), Gallego and Van Ryzin (1997), and Bitran and Caldentey (2003)). The obvious solution to censored sales data, keeping track of arrivals as well as sales, is also much harder in the hospitality industry, where prices are viewable on a variety of different platforms and websites. Web traffic is also made up of a mix of serious customers, people idly browsing, and a significant proportion of bots, which make page-views a poor proxy for arrivals like those in our brick and mortar store.

The problem of estimating demand that has been censored by sales data has been well-studied over the past 15 years. The seminal paper on this topic is Talluri and Van Ryzin (2004) (henceforth TvR). TvR approach the problem by separating demand into an arrival probability and a choice model representing purchase probability. Their model makes several key assumptions, notably that the sales horizon should be broken into small discrete time-steps such that only one arrival can occur in each. TvR primarily use a MultiNomial Logit choice model (MNL), and thus obtain a log-likelihood function which they note is non-concave and difficult to maximize directly. To get around this obstacle, they use the Expectation Maximization (EM) algorithm to estimate the arrival rate and MNL parameters, by iteratively guessing the probability of an arrival for each time-step, and then maximizing the likelihood function with respect to said guesses.

Splitting demand into an arrival rate and and a purchase probability addresses the problems posed by censored demand data, as one has information on the arrival rate, often interpreted as market size, that is separated from price effects. Thus, one can accurately estimate the effect of price on demand for revenue management purposes. The TvR model makes the

assumption that arrival rate is constant throughout the sales horizon. I argue that in many practical settings, the arrival rate in fact varies over time. Note that I am not referring to how arrival rates vary for different stay dates, e.g., due to peak travel seasons or high demand periods such as holidays when many people travel. Rather, my focus is on modeling how the arrival rate varies over time for a *fixed stay date*, as time passes and we accept bookings closer and closer to the stay date. For example, if most customers purchase hotel rooms 1-2 weeks prior to travel, we would have arrival rates peak 1-2 weeks prior to the stay date. Modeling these changes in arrival rate over the sales horizon for hotel rooms with a *fixed stay date* is crucial to accurate price estimation and optimization. Indeed, if one assumes a constant arrival rate when in fact the arrival rate is time-varying, this can lead to model mis-specification and biased parameter estimation.

My focus in this thesis is the single product case. The single-product setting is less well studied, and no less difficult to solve than the more common multi-product setting. Though there are no substitution effects that might censor sales, there is also less relative pricing information (there is no second product to compare attribute or price effects against). The data we have for this paper is from a small hotel with only a single room type. Although my model can be used in other settings with time-varying demand, for ease of reading I will anchor our context to hotel forecasting and note that adaptations to other use cases are natural and immediate.

To illustrate the danger of incorrectly assuming a constant arrival rate in the single product setting, consider Figure 1.1. The points represent daily sales figures over 3 weeks. The only data available on these sales is the price set for that day. The purchase probability used to generate this data was $P_1(u(p))$, a logistic function of utility $u(p) = \beta_0 + \beta_1 p$ for price p. The true beta parameters are $\beta_0 = 2$ and $\beta_1 = -0.22$; note the negative relationship between purchase probability and price. However, Figure 1.1 seems to show a positive relationship between price and sales, which is confirmed by the black dotted line of best fit. The omitted

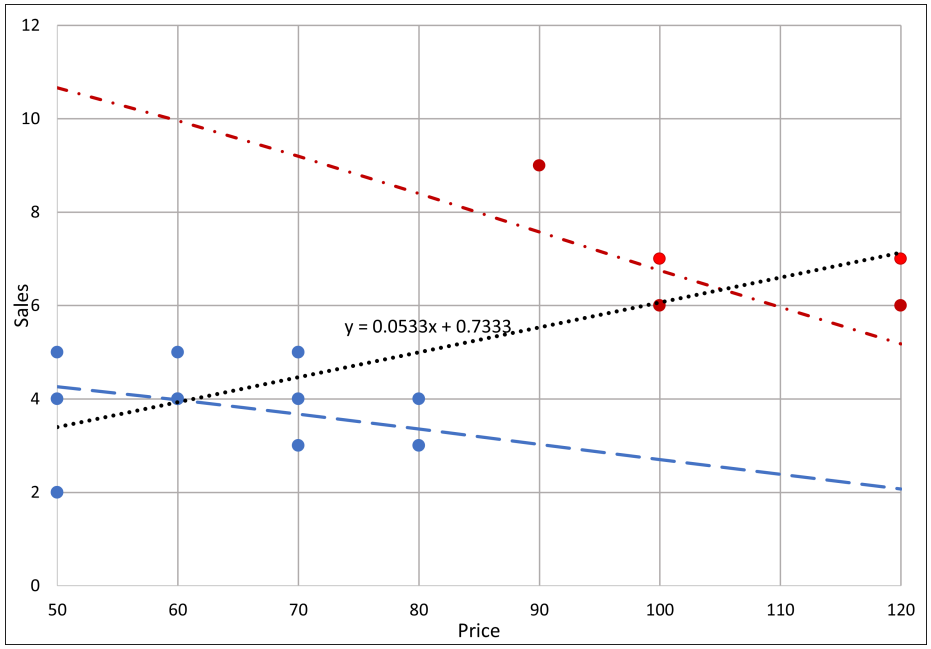

Figure 1.1: Simple Example Displaying the Perils of Assuming a Constant Arrival Rate.

variable here is of course the change in arrival rate - the hourly probability that a customer arrived was 0.25 for the blue points and 0.6 for the red points. Thus, failing to account for the variable arrival rate results in a wild mis-specification of the crucial parameter, β_1. However, if we condition on the underlying arrival rate and estimate the high arrival rate data (red dash-dotted line) separately from the low arrival rate data (blue dashed line), we would establish the true negative relationship between price and sales.

The scenario in Figure 1.1 frequently plays out in the real world. For example, a hotel manager might have had a hunch that demand would be high during the red days, and set high prices accordingly. Proven right, sales were high despite, not because of, the high prices. A similar effect would also occur with a revenue management system, if historical data predicts the red data points would have high demand, causing the system to set high prices. Situations such as these represent a kind of endogeneity, where the manager or system uses assumed knowledge about demand to set the price, thus correlating the error in the response variable with price.

This work adapts the TvR framework with a focus on a time-varying arrival rate in the single-product setting, including their use of EM for estimation. Though modifying the TvR model to accommodate a time-varying arrival rate is relatively straightforward, to my knowledge there has not been a systematic numerical study to explore the practical aspects of implementing such a model. These practical concerns include specification of the variable arrival rate as a function of time, the comparative out-of-sample performance of variable and constant arrival rate models, and the impact of incorporating the variable arrival rate on parameter estimation and convergence. I use both real and simulated hotel data in my analysis, though my model may be applied to any perishable product.

In the course of my work, I frequently encountered the well known problems of slow and inconsistent convergence with the EM algorithm. In addition to my work on the single-product censored demand problem, I also present a variant to EM, called Regularized Expectation Maximization (REM). REM drastically decreases convergence time compared to EM, both in terms of iterations required and clock time. In this work, I provide numerical evidence for reduced convergence times, while establishing that REM possesses similar theoretical properties to EM, and behaves similarly in terms of likelihood, out-of-sample performance, and revenue predictions.

This thesis will proceed as follows. Chapter 2 is a brief overview and literature review of the censored demand literature, and on EM in general. Chapter 3 covers the censored demand problem, including specification of the model, discussion of the estimation procedure, discussion on specifying the arrival rate step function, and numerical results. Chapter 4 details REM, starting with theoretical considerations and intuition, and then continuing with several sets of numerical results. Finally, I conclude the thesis in Chapter 5, and offer suggestions for further work.

Chapter 2

Background and Literature

In this Chapter, we will review two distinct streams of literature. First, in Section 2.1, I give an overview of the EM algorithm and its many variants. Section 2.2 reviews the literature surrounding the censored demand problem. Finally, in Section 2.3, I will outline the contributions of my work.

2.1 The Expectation Maximization Algorithm

2.1.1 Background and History

Expectation Maximization (henceforth EM) is an iterative algorithm originally designed to solve maximum likelihood estimation problems that are difficult to solve using direct maximization techniques. In most cases, maximum likelihood problems can be solved using numerical optimization techniques such as the Newton-Raphson method or gradient descent (see Nocedal and Wright (2006)). However, in cases with missing data or latent parameters, direct maximization can prove impractical. In the language of EM, the original likelihood

function with missing data or parameters is called "incomplete". In EM, rather than maximizing the incomplete likelihood function, we form a so-called "complete" likelihood function that is easier to solve, conditioned on guesses for the complete data, than the incomplete likelihood function.

The foundational ideas of EM have existed for over half a century. In the 1950's and 60's, several special cases of EM were developed for specific applications, notably the Baum-Welch algorithm for learning Hidden Markov Models (Baum et al. (1970), Welch (2003)). Dempster et al. (1977) (DLR) provided the first general formulation and discussion of the algorithm's properties (and coined the term "Expectation Maximization") and potential applications. After the DLR paper, use of EM grew rapidly, with a wide variety of applications and myriad variants being developed over the subsequent decades.

In general terms, EM consists of an expectation step (E), where the complete likelihood function is formed, conditioned on the incomplete data and the current parameter guesses, and a maximization step (M), where the conditional expectation of the complete likelihood function is maximized to produce a new set of parameter estimates. The algorithm alternates between the E-step and the M-step until convergence is achieved - typically when either the parameter estimates or the log-likelihood stop changing.

Due to its relatively reliable convergence and ease of implementation, EM has been used in myriad applications over the years. In their original paper, DLR give 7 different examples of how EM might be used. They provide a nice overview of the various ways EM can be used. The 7 examples are: missing data, truncated or censored data, finite mixture models, estimating the variance components of mixed model ANOVAs, hyper-parameter estimation, iteratively re-weighted least squares, and factor analysis. As previously noted, EM has also been used to estimate Hidden Markov Models (Baum et al. (1970), Welch (2003)), as well as to perform clustering (Redner and Walker (1984)), and in fields such as genetics (Diffey et al. (2017)), image analysis (Qian and Titterington (1992)), and structural engineering

(Matarazzo and Pakzad (2016)).

2.1.2 General Formulation

In the formal setting for EM, we begin with some observed data y with an associated density $p(y|\theta)$ and a description of the complete data x with an associated density $p(x|\theta)$. The parameters θ exist in a set Ω. We assume x is a realization of a continuous random vector X, and y is a realization of a continuous random vector Y. There must also exist a mapping $x \to y(x)$ from X to Y. One common interpretation of X with respect to Y is $X = (Y, Z)$ where Z is some missing data, however there are a variety of other interpretations. Gupta and Chen (2011) list some simple examples, such as X being a random vector and Y being the mean of its components, or X being a complex number with Y as its magnitude. Indeed, the flexibility with which one can define the mapping $x \to y(x)$ is one of the reasons that EM has such widespread applications.

Given this setup, the natural problem to solve is to find the maximum likelihood estimate (MLE) of θ:

$$\hat{\theta} = \arg\max_{\theta \in \Omega} p(y|\theta).$$

Or equivalently, finding the θ that maximizes the log of $p(y|\theta)$, the log-likelihood of y. However, as I noted previously, there are some situations in which the MLE is difficult to obtain by direct maximization. Thus, we turn to EM.

Rather than trying to maximize $L(\theta) = \log p(y|\theta)$ directly, EM maximizes it indirectly by successively maximizing $\log p(x|\theta)$ conditioned on y. The two steps of EM are:

- The E step. Compute the "Q function":

$$Q(\theta|\theta^{(m)}) = E_X\{\log p(x|\theta)|y, \theta^{(m)}\}.$$

  That is, the Q function is the conditional expectation of $p(x|\theta)$, given the observable data y and the current parameter estimates $\theta^{(m)}$.

- The M step. Maximize $Q(\theta|\theta^{(m)})$:

$$\theta^{(m+1)} = \arg\max_{\theta\in\Omega} Q(\theta|\theta^{(m)})$$

The algorithm commences with some initial parameter guess $\theta^{(0)}$, and then the E and M steps are performed alternately until the θ parameters converge. The only theoretical guarantee offered by EM, absent any additional conditions, is that the log-likelihood will never get worse. That is:

$$\log p(y|\theta^{(m+1)}) \geq \log p(y|\theta^{(m)})$$

This relies upon the fact that $\log p(y|\theta^{(m)})$ is a lower-bound for $\log p(y|\theta)$, so long as $Q(\theta|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)})$ (for a nice proof, see Gupta and Chen (2011)). Since $\theta^{m+1} = \arg\max_{\theta\in\Omega} Q(\theta|\theta^{(m)})$, then it must be that $Q(\theta^{m+1}|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)})$, and hence $\log p(y|\theta^{(m+1)}) \geq \log p(y|\theta^{(m)})$.

Since the log-likelihood $L(\theta)$ improves at each iteration, a bounded sequence $\{L(\theta^{(m)})\}_{m=1}^{\infty}$ of likelihood values will converge monotonically to some L^*. In general, L^* is not guaranteed to be a global or even local maximum of L. Wu (1983) provides an in-depth discussion of EM convergence, including when we can be confident that L^* is a local or global maximum. As long as $Q(\theta|\theta^{(m)})$ is continuous in both θ and $\theta^{(m)}$, L^* will be a stationary value, and the final parameter estimate θ^* will be a stationary point as long as $\frac{dQ(\theta|\theta^{(m)})}{d\theta}$ is continuous in θ and $\theta^{(m)}$. Though Wu does provide a condition under which L^* will be a local maximum,

he also notes that the condition is often difficult to verify. In practice, one should vary the initial condition $\theta^{(0)}$ to ensure that EM does not get stuck at a saddle point.For L^* to be a global maximum, and θ^* a global maximizer, Wu requires that $L(\theta)$ be unimodal in Ω.

The convergence speed of EM is a well studied issue. Famously, it can be rather slow compared to other optimization methods, and speed can depend on the choice of $\theta^{(0)}$. DLR show that convergence of EM is linear, and depends on the amount of information loss due to the incomplete data (eg., convergence depends linearly on information loss). McLachlan et al. (2004) give a measure for the observed global convergence rate for θ:

$$r = \lim_{k\to\infty} ||\theta^{(k+1)} - \theta||/||\theta^{(k)} - \theta||$$

where a high r implies slow convergence.

A Short Example

The following example was originally given by DLR in their seminal paper, and has since been a popular simple example of EM in action (for example, in Gupta and Chen (2011)). Suppose we have observed data consisting of observations of 4 types of bird, (y_1, y_2, y_3, y_4) where y_i is the number of times each type of bird was observed. We can model this random histogram Y as a multinomial distribution with two parameters: the number of birds observed $n \in \mathbb{N}$, and the probability that a given bird will be of each type p_i, $i \in \{1, 2, 3, 4\}$. Then, the probability of seeing some random histogram y is:

$$P(y|p) = \frac{n!}{y_1!y_2!y_3!y_4!} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4}$$

Now, suppose due to competition between the species of birds, their populations are held in ratio to each other. Thus, the probability p of seeing a given bird can be parameterized by

some unknown value $\theta \in (0, 1)$ as follows:

$$p_\theta = \begin{bmatrix} \frac{1}{2} + \frac{1}{4}\theta & \frac{1}{4}(1-\theta) & \frac{1}{4}(1-\theta) & \frac{1}{4}\theta \end{bmatrix}^T$$

Ecologists would like to find the true θ, based on the observed data y. Thus, we must find the θ that maximizes the probability of the observed histogram y. Plugging in p_θ to $P(y|p)$, we obtain the incomplete likelihood function $P(y|\theta)$:

$$P(y|\theta) = \frac{n!}{y_1!y_2!y_3!y_4!} \left(\frac{1}{2} + \frac{1}{4}\theta\right)^{y_1} \left(\frac{1}{4}(1-\theta)\right)^{y_2} \left(\frac{1}{4}(1-\theta)\right)^{y_3} \left(\frac{1}{4}\theta\right)^{y_4}$$

In such a small example, it would be fairly trivial to directly maximize $\log(P(y|\theta))$ in terms of θ. However, for the purposes of illustrating EM, suppose that the first category of bird (counted by y_1) is actually made up of two different species that are difficult to distinguish at a distance, $1A$ and $1B$. Species $1A$ is the dominant species, and comprises fully half of the birds in the area. However, species $1B$ must compete with the other 3 species for food, and their population is controlled by θ. Thus, the histogram Y is incomplete data from a 5 category multinomial population with densities (note that $1A$ and $1B$ are now separated):

$$q_\theta = \begin{bmatrix} \frac{1}{2} & \frac{1}{4}\theta & \frac{1}{4}(1-\theta) & \frac{1}{4}(1-\theta) & \frac{1}{4}\theta \end{bmatrix}^T.$$

Therefore the complete data is $X = (x_{1A}, x_{1B}, x_2, x_3, x_4)$, where $y_1 = x_{1A} + x_{1B}$, $y_2 = x_2$, $y_3 = x_3$, and $y_4 = x_4$. Plugging all this in, the complete likelihood function is:

$$P(x|\theta) = \frac{n!}{x_{1A}!x_{1B}!x_2!x_3!x_4!} \left(\frac{1}{2}\right)^{x_{1A}} \left(\frac{1}{4}\theta\right)^{x_{1B}} \left(\frac{1}{4}(1-\theta)\right)^{x_2} \left(\frac{1}{4}(1-\theta)\right)^{x_3} \left(\frac{1}{4}\theta\right)^{x_4}. \quad (2.1)$$

To perform the E-step, we must find $Q(\theta|\theta^{(m)}) = E_{X|y,\theta^{(m)}}[\log p(X|\theta)]$. To do so, we take the log of (2.1) and ignore all terms that do not involve θ, and then take the expectation of

the result. Doing so, we obtain:

$$Q(\theta|\theta^{(m)}) = E_{X|y,\theta^{(m)}}[(X_{1B} + X_4)\log(\theta) + (X_2 + X_3)\log(1-\theta)]$$

It remains to find $E_{X|y,\theta^{(m)}}[X]$. By leveraging the binomial distribution, and specifically the fact that $y_1 = x_{1A} + x_{1B}$, we can write the probability that exactly x_{1A} birds from species $1A$ were observed in the y_1 observations as follows:

$$P(x_{1A} \text{ out of } y_1 \text{ observations }) = \frac{y_1!}{x_{1A}!x_{1B}!}\left(\frac{1/2}{1/2+\theta^{(m)}/4}\right)^{x_{1A}}\left(\frac{\theta^{(m)}/4}{1/2+\theta^{(m)}/4}\right)^{x_{1B}}$$

Making further use of the binomial distribution, we can write the conditional complete log-likelihood function as:

$$\begin{aligned}P(x|y,\theta^{(m)}) =& \frac{y_1!}{x_{1A}!x_{1B}!}\left(\frac{1/2}{1/2+\theta^{(m)}/4}\right)^{x_{1A}}\left(\frac{\theta^{(m)}/4}{1/2+\theta^{(m)}/4}\right)^{x_{1B}} \times B(y_1|x_{1A}+x_{1B},p_{\theta,1}) \\ & \times B(y_2|x_2,p_{\theta,2})B(y_3|x_3,p_{\theta,3})B(y_4|x_4,p_{\theta,4}).\end{aligned}$$

where $B(n|k,p)$ is the probability of k successes in n trials with probability p, and $p_{\theta,i}$ refers to the ith element of p_θ, defined above.

Given that we know that $y_1 = x_1 + x_2$, $y_2 = x_3$, $y_3 = x_4$, and $y_4 = x_5$, we obtain (after some simplification):

$$E_{X|y,\theta^{(m)}}[X] = \left[\frac{2}{2+\theta^{(m)}}y_1 \quad \frac{\theta^{(m)}}{2+\theta^{(m)}}y_1 \quad y_2 \quad y_3 \quad y_4\right]^T.$$

Plugging these in to our Q function, the M-step is therefore:

$$\begin{aligned}\theta^{(m+1)} &= \arg\max_{\theta\in(0,1)} \left(\left(\frac{\theta^{(m)}}{2+\theta^{(m)}} y_1 + y_4 \right) \log(\theta) + (y_2 + y_3) \log(1-\theta) \right) \\ &= \frac{\frac{\theta^{(m)}}{2+\theta^{(m)}} y_1 + y_4}{\frac{\theta^{(m)}}{2+\theta^{(m)}} y_1 + y_2 + y_3 + y_4}\end{aligned}$$

EM as Lower Bound Maximization

An interesting interpretation of EM is as a maximization-maximization procedure, in which the algorithm iteratively maximizes the lower bound on L. In particular, we guess that X has distribution $\tilde{P}$ with density $\tilde{p}(x)$. Let P_θ be the conditional distribution with density $p(x|y,\theta)$. Then we maximize the following objective alternately in terms of $\tilde{P}$ and θ:

$$F(\tilde{P}, \theta) = L(\theta) - D_{KL}(\tilde{P}||P_\theta)$$

where $D_{KL}(\tilde{P}||P_\theta)$ is the Kullback-Leibler divergence between the current guess $\tilde{P}$ of the complete data distribution, and the likelihood P_θ conditioned on θ (Gupta and Chen (2011)). The Kullback-Leibler divergence is defined as follows:

$$D_{KL}(\tilde{P}||P_\theta) = \int_{-\infty}^{\infty} \tilde{p}(x) \log \frac{\tilde{p}(x)}{p(x|y,\theta)} dx.$$

Maximizing $F(\tilde{P}, \theta)$ with respect to θ produces a lower-bound on $L(\theta)$ since D_{KL} is always non-negative. Then maximizing $F(\tilde{P}, \theta)$ with respect to $\tilde{P}$ tightens the lower bound, given the current estimate for θ. Gupta and Chen (2011), among others, show that iteratively carrying out these maximizations is in fact equivalent to EM.

2.1.3 Variants and Generalizations

The most basic generalization of EM is appropriately named Generalized Expectation Maximization (GEM), first proposed in the original DLR paper. In GEM, rather than finding $\theta^{(m+1)} = \arg\max_{\theta\in\Omega} Q(\theta|\theta^m)$, we instead require

$$\theta^{(m+1)} \text{ such that } Q(\theta^{m+1}|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)}).$$

That is, we choose $\theta^{(m+1)}$ so that it only produces an increase in the Q function at $\theta^{(m)}$, rather than maximizing over Ω. Wu (1983) shows that GEM procedures have many of the same convergence properties as EM, and so a common result for many EM variants is to show that they are GEM procedures.

In the decades since EM was formalized, there have been a multitude of variants proposed for various purposes. According to McLachlan et al. (2004), the extensions and variants of EM typically aim to achieve one of the following 4 goals:

1. Produce standard errors of the MLE with EM
2. Combat difficulties in either the E-step or M-step computations
3. Improve convergence speed
4. Adapt EM to Bayesian maximum likelihood estimations

We will address the first 3 variant types. An overview of EM in a Bayesian context can be found in McLachlan et al. (2004).

Standard Errors with EM

Several methods have been proposed to add to the EM calculation to obtain an estimate of the covariance matrix for the MLE. These methods tend to fall into one of two categories:

standard error estimates are obtained either by estimating an information matrix (the negative Hessian of the log-likelihood), or by using re-sampling methods such as bootstrapping. A prominent method in the former category, Supplemented EM (SEM) was first proposed by Meng and Rubin (1991). Like other information matrix methods (a review can be found in Jamshidian and Jennrich (2000)), SEM leverages the relationship:

$$H = Q''(I - M'),$$

shown by DLR, where H is the Hessian of the log-likelihood function, Q'' is the Hessian of the Q function, I is an $n \times n$ identity matrix (assuming $\theta \in \mathbb{R}^n$) and M' is the gradient of the so-called "EM operator", defined by

$$\theta^{(m+1)} = M(\theta^{(m)}).$$

That is, EM implicitly defines a mapping from Ω to itself such that $\theta^{(m+1)}$ is related to $\theta^{(m)}$ as above. In SEM, we numerically differentiate M at the θ to which EM converged, θ^*, and then use the above formula for H to obtain an estimate. The first step in SEM is to run EM to obtain a sequence of parameter estimates $\{\theta^{(m)}\}$ until $||\theta^{m+1} - \theta^{(m)}|| < \varepsilon$, and set $\theta^* = \theta^{(m+1)}$. Then, we obtain sequences r_{ij} as follows:

$$r_{ij}^{(m)} = \frac{M_i(\theta^* + (\theta_j^{(m)} - \theta_j^*)\boldsymbol{u}^j) - \theta_i^*}{\theta_j^{(m)} - \theta_i^*}$$

where M_i is the ith component of M, θ_j is the jth component of θ, and $\boldsymbol{u}^j$ is a vector with the jth component equal to 1 and all other components equal to 0. The sequences $\{r_{ij}^{(m)}\}$ are stopped as soon as $||r_{ij}^{(m+1)} - r_{ij}^{(m)}|| < \sqrt{\varepsilon}$, and $r_{ij}^* = r_{ij}^{(m+1)}$. The estimate of M' is the matrix of the r_{ij}^*'s. Jamshidian and Jennrich (2000) note that SEM can break down if some components of $\theta^{(m)} - \theta^*$ are zero, and that the fix proposed by Meng and Rubin (1991) is not always straightforward. One further drawback common to all information matrix methods,

is that the estimated information matrix is only valid asymptotically. Thus, in some contexts the sample size has to be very large to be able to draw inferences from the information matrix (McLachlan et al. (2004)).

Bootstrapping in general refers to re-sampling a dataset to calculate sample statistics or perform statistical tests without needing to satisfy the requirements that said statistics or tests may require for a single dataset (Efron (1992)). The so-called bootstrap samples are randomly drawn with equal weight and with replacement from the original dataset. Formally, if our dataset consists of N samples $y_1, \ldots, y_N$ with distribution $\hat{F}$ formed from the observed data, a bootstrap sample is:

$$\tilde{y}_1, \ldots, \tilde{y}_N \overset{\text{i.i.d.}}{\sim} \hat{F}.$$

That is, to form a bootstrap sample we randomly sample N points with replacement from our observed data. To apply bootstrapping to EM, we re-sample the observed data y to obtain the boostrap sample $\tilde{y}$. Then, EM is applied to $\tilde{y}$ to obtain the MLE $\tilde{\theta}^*$. By re-sampling B times, one can estimate the covariance matrix as follows:

$$\text{Cov}(\tilde{\theta}^*) \approx \sum_{b=1}^{B} = \left(\tilde{\theta}_b^* - \bar{\tilde{\theta}}^*\right)\left(\tilde{\theta}_b^* - \bar{\tilde{\theta}}^*\right)/(B-1)$$

where $\bar{\tilde{\theta}}^* = \sum_{b=1}^{B} \tilde{\theta}_b^*/B$. The standard error of the ith element of θ^* is the positive square root of the ith diagonal element of $\text{Cov}(\tilde{\theta}^*)$ (McLachlan et al. (2004)). Tibshirani and Efron (1993) show that 50-100 bootstrap samples (eg. $50 < B < 100$) are generally sufficient for standard error estimation.

Difficult E or M steps.

In some contexts, either the E or M steps can be difficult to compute. In the former case, one solution is to execute the E-step via Monte Carlo simulation. In the mth iteration of EM, one makes N independent draws of the missing data $Z = (z^{(1m)}, \ldots, z^{(Nm)})$ from the

conditional distribution $p(z|y, \theta^{(m)})$. The Q function can then be approximated by:

$$Q(\theta|\theta^{(m)}) \approx Q_N(\theta|\theta^{(m)}) = \frac{1}{N}\sum_{n=1}^{N} \log p(\theta|z^{nm}, y)$$

This method is referred to as Monte Carlo EM (MCEM) and was first proposed by Wei and Tanner (1990). A drawback of this approach is that the Monte Carlo sampling invalidates the monotonicity property of EM. However, Booth and Hobert (1999) note that MCEM does converge with high probability in certain circumstances. One must also specify a choice of N for each iteration. Wei and Tanner (1990) recommend starting with small values of N in early iterations, and increasing N as the algorithm goes on.

A common problem with implementing EM is that the M-step can be complicated or difficult to maximize. McLachlan et al. (2004) note that in such cases, the M-step can be made easier if it is undertaken conditional on some functions of the parameters being estimated. Meng and Rubin (1993) introduced a class of GEM algorithms they call the Expectation-Conditional Maximization (ECM) algorithm. In the ECM algorithm, a complicated M-step is replaced by several computationally simpler conditional maximization (CM) steps.

In ECM, the parameter vector θ is divided up into S sub-vectors:

$$\theta = \left(\theta_1^T, \ldots, \theta_S^T\right)^T.$$

For the sth sub-vector, the CM-step is performed by maximizing only over θ_s with all $\theta_{i\neq s}$ held fixed. The key property that Meng and Rubin (1993) show is that ECM is a GEM algorithm - that is, the likelihood improves after each CM-step. Formally,

$$Q(\theta^{(m+1)}|\theta^{(m)}) \geq Q(\theta^{(m+(S-1)/S)}|\theta^{(m)}) \geq \cdots \geq Q(\theta^{(m)}|\theta^{(m)})$$

that is, each successive CM-step improves the value of the Q function, which implies (as

shown earlier) that the log-likelihood improves with each CM-step. This result confers the same convergence properties of GEM onto ECM, as discussed in Wu (1983). Meng and Rubin (1993) also discuss the so-called multi-cycle ECM algorithm. A cycle is defined as one E-step followed by one CM-step. Thus, in the multi-cycle ECM algorithm, one performs an E-step before each CM-step rather than performing one E-step for all the CMs for a given iteration. This has the potential benefit of larger likelihood increases at each iteration, though computation times may increase. McLachlan et al. (2004) note that the multi-cycle variant may not be a GEM algorithm, however it is typically easy to show that the log-likelihood still improves at every step.

Convergence Speed

One of the most commonly cited drawbacks of EM is that relative to other maximum likelihood methods, it typically converges quite slowly. As such, there have been numerous attempts to speed up EM convergence. As DLR note, EM convergence speed depends on the amount of information missing from the incomplete data y. Thus, a major stream of research has been focused on speeding EM up by augmenting the observed data y.

One such approach builds off of the ECM algorithm. In the Expectation-Conditional Maximization Either (ECME) algorithm (Liu and Rubin (1994)), we add on to the ECM framework by replacing some of the CM-steps and instead maximizing a constrained version of the actual likelihood function. Recall that in ECM, we partition the parameter vector θ into subsets θ_s, $s \in \{1, \ldots, S\}$ and then solve smaller conditional M-steps for each subset, holding the rest fixed. In ECME, we initialize disjoint sets $\mathcal{S}_Q$ and $\mathcal{S}_L$ such that $\mathcal{S}_Q \cup \mathcal{S}_L = \{1, \ldots, S\}$. If $s \in \mathcal{S}_Q$ then θ_s is solved for using a CM-step, and if $s \in \mathcal{S}_L$ then we solve for θ_s by maximizing the actual likelihood function $p(x|\theta)$, constrained so that $\theta_{i \neq s}$ are held fixed (eg. the analog of the CM-step). Liu and Rubin (1994) claim that the ECME algorithm converges faster than both EM and ECM, and show that ECME is a GEM algorithm.

The Alternating Expectation-Conditional Maximization algorithm (AECM) (Meng and Van Dyk

(1997)) further generalizes ECME by allowing for the data augmentation procedure to go beyond a simple partition of θ, and also allowing for the augmentation scheme to vary within iterations. AECM uses a set of constraining functions $G^{[c]} = \{g_s^{[c]}(\theta), s = 1, \ldots, S^{[c]}\}$ for the cth cycle (defined in the same way as for multi-cycle ECM). A cycle then consists of an E-step, and $S^{[c]}$ CM-steps, and a full iteration of AECM consists of sum number of cycles. Meng and Van Dyk (1997) show that under certain technical conditions on the g functions, AECM converges and is a GEM.

A more recent approach to speeding up EM is EM with Frequent Updates (FreEM) (Yin et al. (2012)). FreEM proposes to increase convergence speed by updating the parameter estimates more frequently, by performing the E and M steps on a subset of the data, and then using the updated parameters to calculate the the E and M steps for the next subset. Yin et al. (2012) present two variants of this method, firstly where the updates are performed on a "block" of data $B_i \subseteq X$, and secondly where the Q function is re-computed using a sub-range of missing data Z. The authors further show that FreEM is able to be parallelized, to further reduce computation time.

2.2 Modeling Censored Demand for Perishable Products

Hotel Forecasting.

There are a variety of papers concerned with forecasting demand for hotels. Pekgün et al. (2013) is perhaps the most prominent such paper. This paper uses proprietary data from the Carlson Rezidor hotel group to develop a price optimization model which relies on demand forecasts from a linear regression. Another common approach is to use traditional time-series forecasting techniques on a hotel booking curve. Lee (2018) models arrivals as

a non-homogenous Poisson process, which he uses to forecast a booking curve over time. Zakhary et al. (2011) use a Monte Carlo approach to simulate the booking curve and obtain forecast densities. One feature common to much of the hotel demand forecasting literature is a reliance on classical forecasting techniques, such as time series methods and modeling seasonality. Booking curves are typically modeled independent of any price changes, which means that such methods are ill suited to providing input for a dynamic pricing policy.

Censored Demand Estimation.

As discussed in Chapter 1, Talluri and Van Ryzin (2004) is the seminal paper on the study of censored demand. They approach the problem by separating demand into a constant arrival probability, and a purchase probability modeled as a multinomial logit function of purchase utility. They estimate their model using the Expectation Maximization (EM) algorithm. TvR's framework has been a guide for much of the subsequent literature on the censored demand problem – in particular, their use of an arrival probability and the MNL choice model. Vulcano et al. (2010) provide a numerical study of the TvR model and EM estimation on an airline dataset, and report that choice modeling leads to average revenue improvements of 1-5% over methods used by the airline. Most of the work that builds on the TvR paper focuses on one or more of the following features of the paper: the use of discrete timesteps, the use of EM as the estimation method, the assumption that the non-purchase option has utility zero, and their suggestion that the utility functions for each option need not include constants.

On the first topic, Talluri (2009) notes that breaking the sales horizon into discrete time-steps may pose estimation problems, such as obtaining inconsistent estimates across different numbers of time-steps. TvR's assumption that their "outside option" - the no-purchase option - has utility zero has also proved controversial. As Ferguson et al. (2012) note, this assumption may be invalid, and result in biased parameter estimates. Most, if not all, subsequent papers on the censored demand problem attempt to estimate the non-purchase

utility rather than fixing it to zero. In the same paper, Ferguson et al. (2012) also take issue with the lack of constant utility terms in TvR's model, again arguing that this can bias the estimates. MNL choice models are often specified with constant terms specific to each choice alternative, called Alternative Specific Constants (ASC's). These constant terms represent the marginal utility of each alternative not captured by the model's attributes. However, as with any discrete choice model, we can only uniquely estimate differences in utility. Thus, if we estimate J such terms for J products, they will not be unique – as one can add a constant to each utility function and estimate identical choice probabilities. To remedy this, one utility value must be fixed as a reference to produce unique parameter estimates – usually the utility associated with the outside option. This is the reason that TvR fix the utility of their outside option, and thus the ASC, to zero. As noted above, many authors now argue that the outside option should not have a fixed utility of zero.

The two main streams of work that follow TvR are differentiated by their approach on how they handle the non-purchase utility, and thus fix the lack of uniqueness of the ASC's. One stream, including Vulcano et al. (2012), and Abdallah and Vulcano (2021) use outside market share information to specify the non-purchase utility. The other stream uses a two-step approach in which the parameters for the purchase options are estimated first, and the arrival rate and non-purchase option are estimated second, conditioned on the results of the first step. This stream includes Newman et al. (2014), Li and Talluri (2020), and Cho et al. (2021). In both streams of literature, it is generally accepted that the arrival rate and non-purchase utility (provided it is non-zero) are related, since the non-purchase utility can be used to find the expected number of customers who did not purchase. Thus the arrival rate is typically calculated as a function of the non-purchase utility.

Specifying Non-Purchase Utility from Known Market Share.

Vulcano et al. (2012) extend the original TvR framework by modeling arrivals as a non-homogenous Poisson process, with arrival rate λ_t for each period $t \in 1, \ldots, T$. To ensure

unique identification of their parameters, they assume knowledge of some market share s, which is used to fix the utility of the outside alternative. Their key contribution is to reframe the problem in terms of primary demand, that is, the demand for each alternative that would be present if every alternative was available in every time period. This transformation vastly simplifies their EM procedure, providing significant performance improvements over the TvR EM method. However, as Newman et al. (2014) note, the simplified EM procedure relies on the fact that Vulcano et al. (2012) use a "constants-only" choice model without slope parameters, and the performance improvement disappears when utility is extended to be a linear function of the product attributes (which includes price).

Abdallah and Vulcano (2021) build on the model of Vulcano et al. (2012), and provide two new key contributions. First, they establish conditions under which the likelihood function is guaranteed to have a unique maximum. Second, instead of transforming the problem in terms of primary demand, they estimate their model using a minorization-maximization (MM) algorithm. In their MM method, the authors derive a minorizor of the likelihood function, and then iteratively maximize the minorizor until the parameter estimates converge. Vulcano and Abdullah note that a key advantage of their MM approach is that unlike EM, it is guaranteed to converge to the global optimum of their log-likelihood function. Like the 2012 paper, the Vulcano and Abdullah approach incorporates market share information to estimate the arrival rates and non-purchase utility.

Two-Step Estimation Methods.

The first significant paper in the two-step stream of literature is Newman et al. (2014). Here, the authors change the TvR discrete time-steps to continuous time-windows, within which the arrival rate is modeled as a Poisson process. They then maximize their parameters in two steps: first solving for the choice parameters conditioned on a purchase (i.e., excluding the non-purchase utility) and plugging the estimated choice parameters into their likelihood function to compute the arrival rate and non-purchase utility. The first step is

provably concave, and a result from McFadden (1977) ensures the estimates will be consistent. The second step can be transformed into a one-dimensional non-convex problem, by writing arrival rate as a function of the non-purchase utility. They show their method to be computationally faster and more efficient than EM methods.

Li and Talluri (2020) also use a two-step method similar to that in Newman et al. (2014). While the first step is the same in both papers, Li and Talluri posit that the Newman method relies on the assumption that all sales are independent across time, which is only the case if the product assortments offered are exogenously chosen. However, if the assortments are chosen based on past sales (e.g., using a revenue management system), the independence assumption is invalid. To rectify this problem, Li and Talluri develop a "policy robust" method for the second step, using a Generalized Method of Moments to solve for the non-purchase utility and arrival rate.

Other Recent Works.

Cho et al. (2021) seeks to unify the the Two-Step and Market Share streams to some degree. After first solving for their choice parameters, again conditioned on a sale occurring, the authors then offer two ways to find the non-purchase utility: using market share information or using a reference product. The reference product approach requires additional assumptions on consumer behavior, that imply that at least one choice has the same or similar ASC as the outside alternative. They provide a search algorithm to find which product to use as the reference product, and also provide an algorithm to choose between non-purchase utilities produced by each method.

A final contribution to the literature on the censored demand problem does not fit neatly into either stream of literature. Subramanian and Harsha (2021) solve for the choice parameters, market size parameters, and the lost sales probability for each time-period by solving an Mixed Integer Program. While other approaches use maximum likelihood to estimate the parameters, Subramanian and Harsha extend Berkson's method, which minimizes error

rather than maximizing likelihood. Using this framework, they construct a loss function for the objective of their MIP. One of the key innovations in their approach is correctly estimating the lost sales probability for each time period – essentially analogous to non-purchase utility. They do this by randomly sampling several candidate probabilities and then using SOS2 variables in their MIP to choose the interval in which the arrival rate providing the best fit falls.

2.3 Contributions

The literature on censored demand has grown significantly in recent years, and presents a number of different approaches to the original problem outlined by TvR. However there are several areas that the literature does not yet adequately cover. Notably, with the exception of the original TvR model and Subramanian and Harsha (2021), all the approaches reviewed above require two or more observed products (Cho et al. (2021), Subramanian and Harsha (2021)). In many applications, including the hotel industry, multiple products, such as room classes, are available and thus requiring two or more products is not restrictive. However, there do exist cases where the data is highly censored - thus requiring specialized techniques - and where only a single product is available. For example, many budget chain hotels only have one, or two very similar types of room - say double queen or single king bed in the same room layout. In such cases, the room types are essentially identical in features and are often offered at the same or very similar price points. This is the case for the hotel that we obtained data from, and the two kinds of rooms are not differentiated in the data. It is worth noting that, like the original TvR model, our model can be extended to the multi-product space. However, in the presence of multiple products, there are several other models (as discussed in Section 2.2) that may be more appropriate.

The approach used by Subramanian and Harsha (2021) (S&H) is distinct from the TvR

stream of work, and requires several assumptions that our work does not require. Firstly, they assume that a positive number of sales for each product occurs for all time periods. They note that this is reasonable due to their use of aggregated data. However, when only a single product is present, or when the volume of sales is low, this assumption becomes less reasonable. Indeed, our real world data contains many daily observations with no recorded sales. Thus, in order to satisfy the assumption of positive sales for each time period, one would have to aggregate at beyond the daily level. Doing so would cause information about daily pricing changes to be lost. This problem is particularly acute in the single product case, as in the multiple product case one can glean information about the relative utility of different products - impossible with just one product. S&H do provide a heuristic adjustment for zero sales observations, in the case that aggregation is not feasible. However, this adjustment again relies on the presence of multiple products to preserve relative utilities - in the single product case this is not possible.

Secondly, extra modeling effort and data is required for the S&H approach, as one must explicitly model the relationship between market size (or arrival rate) and additional independent variables. To do so, the modeler must know the functional form of the arrival rate with respect to time, or assume that the arrival rate is constant. Our approach allows for a time-varying arrival rate without specifying a functional form.

Inherent to the modeling approach of TvR as well as the subsequent literature is an identification problem that makes it difficult to unambigously separate purchase probability from arrival rate. Since demand is modeled as the product of these two quantities, and arrival rate is unobserved, one could have a low arrival rate and high purchase probability produce the same demand prediction as a high arrival rate and low purchase probability. TvR themselves note the existence of this problem, and literature such as Li and Talluri (2020) have pointed out that such a property is problematic for practitioners attempting to optimize their prices, as it matters whether sales were due to a high demand rate or good pricing. The literature

at large tends to address this problem by using a non-zero no-purchase utility that is tied to arrival rate, which is either determined from outside information (Market Share approach) or solved for directly (Two-Step approach), while also leveraging the existence of multiple product options. the single product case, and without suitable external sources to estimate the market size, these other approaches are not viable options.

I do not claim to solve the identification problem in my work. In fact, in the Regularized Expectation Maximization algorithm, I leverage the fact that multiple optimal solutions exist on the likelihood surface to provide a route to faster convergence. However, our use of a time-dependent step function to model arrival rate does help to lessen the impact of the identification problem. By estimating multiple arrival rates across the sales horizon, while we may not be certain that the individual magnitudes of the arrival rates are correct, we can be confident that their ratio to one another is well-identified. To reiterate, the crux of the identification problem is that two sets of parameters may yield the same approximate likelihood and demand predictions, e.g., one set of parameters with high arrival rate and low purchase probability, and another with the reverse pattern. With multiple arrival rates, this can still happen, but since the purchase probability parameters are the same for the entire horizon, the ratio between the arrival rates should be more or less the same for both sets of parameters. This property provides crucial information as to the relative magnitudes of the arrival rate. For example, with 2 arrival rates (far away and close to the stay date) we may not know for sure whether arrival rate 1 is 0.4, but we will know how it compares to arrival rate 2. The practitioner is thus still able to see where the arrival rate peaked or fell, and scale prices up or down accordingly.

To summarize my contributions once again, in this paper I extend the TvR framework and apply it to a single product case in the hotel industry. I adapt the TvR model to our context and model the arrival rate as a step function over time, and conduct an extensive investigation as to the practicalities of doing so. In particular, I define a heuristic algorithm to

specify the arrival rate step function for a given dataset, and examine its properties. I have also conducted numerical experiments on the effect of mis-specifying the arrival function. In terms of prediction, I compare my variable-arrival rate model to the constant arrival rate model from TvR on both simulated and real hotel data. My variable arrival model outperforms the constant arrival model by up to 10%, out-of-sample. I also validate my arrival rate step-selection algorithm by showing that it consistently out-performs a naive, evenly-spaced arrival rate step configuration.

In the course of my work, I encountered the well-documented convergence issues of the EM algorithm. To combat these issues, I develop a new variant of EM called Regularized Expectation Maximization (REM). By adding a penalty term to the maximization step, REM converges 81% faster than EM in terms of both iterations and clock time. I have found that EM and REM tend to converge to slightly different solutions. However, I have performed extensive numerical tests to show that the EM and REM solutions are functionally equivalent. I show that both the EM and REM solutions sit on a plateau in the incomplete likelihood surface, and thus achieve nearly identical likelihoods. Furthermore, I provide a proof that REM meets the criteria for a Generalized Expectation Maximization method, and as a result shares many of the convergence properties of EM. Finally, I show numerically that REM and EM solutions produce near identical revenue curves, and have negligible differences in both in- and out-of-sample performence.

Chapter 3

The Censored Demand Problem with Hotel Data

This chapter covers my treatment of the censored demand problem, with evaluation on real and simulated hotel data. In Section 3.1, I will discuss the model and the data used for evaluation. Section 3.2 covers my proposed Step Variance Minimization Heuristic algorithm for selecting the steps of the arrival-rate step-function, a crucial choice that must be made prior to estimation. A poor choice of step configuration can result in biased parameters and inaccurate arrival rates. In Section 3.3, I detail the estimation procedure with EM, and how EM is adapted to the problem at hand. Finally, in Section 3.4.2 I present the numerical results from my EM trials, which show that using a variable arrival rate provides significant performance improvements over the original TvR model, and that my SVMH step-selection method is superior to a naive method of step selection.

3.1 Model and Data

In this section, I describe the demand model, including the time-varying arrival function. I also describe the data I use for my results, both simulated and real.

Our demand model takes the same form as that of Talluri and Van Ryzin (2004). We will assume that the sales horizon is broken into timesteps t small enough that $P(2 \text{ or more Arrivals in } t) = 0$. The demand function is thus defined in terms of the probability of making a sale in a given timestep:

$$\text{P[1 Sale in timestep t]} = \lambda(\tau)P_1(p, \tau).$$

That is, the probability of making a sale is defined as the product of arrival function $\lambda(\tau)$ and purchase probability function $P_1(p, \tau)$ for a given price p and Days Before Arrival (DBA) $\tau \in \{0, 1, \ldots, T\}$. In the hotel context, days before arrival refers to the number of days remaining before the stay date for a given room - essentially τ counts down towards the end of the sales horizon with length T days.

3.1.1 Purchase Probability

I model purchase probability as a logistic function of purchase utility. Purchase utility is assumed to be linear in price p and DBA τ:

$$\text{U(p,}\tau; \boldsymbol{\beta}) = \beta_0 + \beta_1 p + \beta_2 \tau.$$

However, so long as the utility remains linear in its parameters $\boldsymbol{\beta}$, one could include a wide variety of covariates, interactions, and polynomial terms. The utility for not purchasing is assumed to be zero, so β_0 represents the utility of purchasing when price and DBA are 0.

Purchase probability is therefore

$$P_1(p, \tau; \boldsymbol{\beta}) = \frac{e^{U(p,\tau;\boldsymbol{\beta})}}{1 + e^{U(p,\tau;\boldsymbol{\beta})}},$$

where the bold symbol $\boldsymbol{\beta}$ represents the vector $[\beta_0, \beta_1, \beta_2]$. The non-purchase probability is defined as $P_0 = 1 - P_1$, or

$$P_0(p, \tau; \boldsymbol{\beta}) = \frac{1}{1 + e^{U(p,\tau;\boldsymbol{\beta})}}.$$

Since we are only considering the single product case, we need only account for the probability that said product is purchased or not. There is nothing in our model preventing the use of P_1 as an MNL for multiple products. As I mentioned in Section 2.3 however, there may be other models that are more appropriate in the multi-product case. Table 3.1 gives a summary of our notation.

Variable	**Description**
$S_j, \bar{S}_j$	The set of all purchases/non-purchases contained in arrival rate step j
$P_1(p, \tau), P_0(p, \tau)$	Probability of making a sale/no sale given price p and DBA τ, logistic function
$\boldsymbol{\beta}$	Parameter vector for P_1, P_0, consisting of an intercept, and slopes for price and DBA
$\lambda(\tau)$	Arrival Rate function, a step function of DBA
$\boldsymbol{\lambda}, \lambda_j$	Vector of arrival rate parameters, arrival rate for step j
$\hat{a}(k)$	Expected probability of an arrival at observation k
$\boldsymbol{\beta}^{(m)}, \lambda_j^{(m)}$, etc.	Indicating a parameter estimated at the m^{th} iteration

Table 3.1: Key Variables, Sets, and Parameters Descriptions

3.1.2 Arrival Rate Function

We will define the arrival probability as $P(1 \text{ Customer arrives during } t)$, where again, the timesteps t are small enough that the arrival rate is always < 1. Note that due to this

assumption, the arrival rate for a single timestep is the same as the arrival probability, and so these terms may be used interchangeably.

Though Talluri and Van Ryzin (2004) define a constant arrival rate λ, I define $\lambda(\tau)$, a function of DBA. Specifically, $\lambda(\tau) : \mathbb{N} \to \mathbb{R}$ as a step function over the range of τ. Though $\lambda(\tau)$ depends on τ, a parameter which changes every day, keep in mind that the arrival rate is defined per timestep (eg. 1 hour), and the dependency on τ means that the arrival rate is the same for all timesteps corresponding to DBA τ. Formally, $\lambda(\tau)$ is defined as:

$$\lambda(\tau) = \begin{cases} \lambda_1 & \tau \in T_1 \\ \lambda_2 & \tau \in T_2 \\ \vdots & \\ \lambda_n & \tau \in T_N \end{cases}$$

where $\bigcup_{j=1}^{N} T_j = T$ are a partition of the sales horizon that make up the steps of our step function, and $\min\{\tau : \tau \in T_j\} > \max\{\tau : \tau \in T_{j+1}\}$ (this second condition ensures that only consecutive DBAs are contained in a step). The composition of this partition (i.e. the size and location of the steps) is a key set of hyper-parameters of our model, which I will address in Section 3.2.

Another way to parameterize $\lambda(\tau)$ is by defining the "boundaries" for each step, that is: the DBAs where each step ends. For a step T_j, let $\tilde{\tau}_j = \min\{\tau : \tau \in T_j\}$. Using this definition, we can define a step configuration with $N - 1$ boundaries, since $\tilde{\tau_N} = 0$ (or 1 if the horizon ends at DBA=1). In this thesis, when I want to describe a step configuration, I will do so by listing the $\tilde{\tau}_j$. In general, I will include the boundary $\tilde{\tau}_N = 0$, so that the number of stated boundaries equals the number of steps. Modeling arrival rate as a step function allows the arrival rate to vary with time without a priori knowledge of functional form. Since customer arrivals are unobservable, it can be difficult to determine how arrivals might

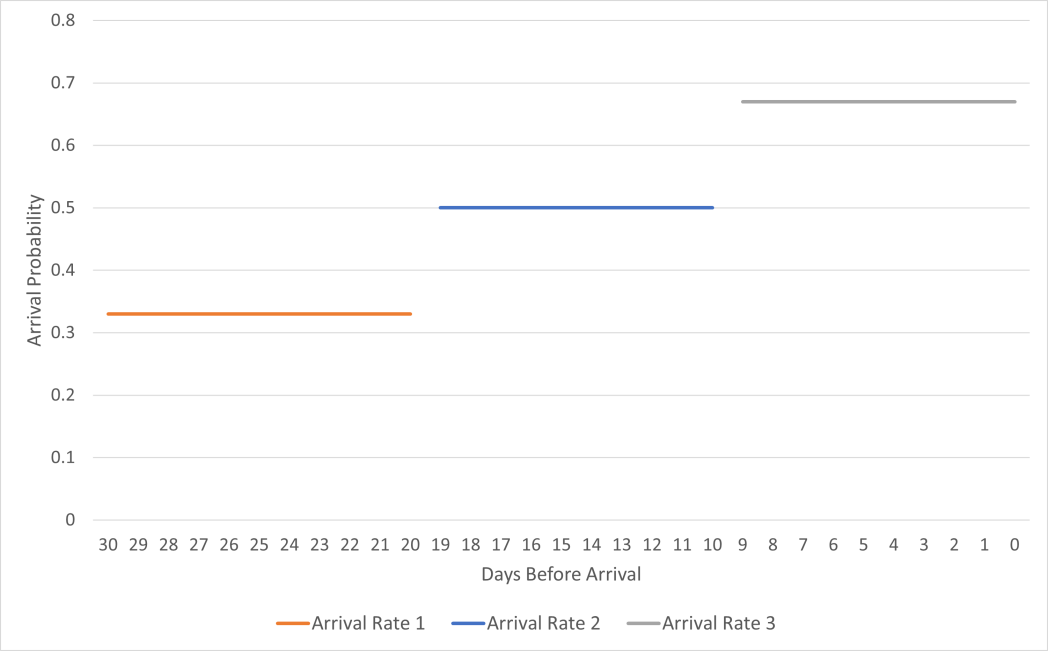

Figure 3.1: Example Step Function for a Time-Varying Arrival Rate

fluctuate over time. While several reasonable proxies do exist, such as sales rate and market size, data on market size may not be readily available, and sales rate can be misleading (see Figure 1.1). The step function does not presuppose a shape for the arrival function, and in fact does not force the arrival rate to change at all (we could have $\lambda_1 = \lambda_2 = \cdots = \lambda_n$). Figure 3.1 illustrates $\lambda(\tau)$ defined over a 30 day sales horizon, with three separate arrival rates. We have λ_1 for the first 10 days, λ_2 for the second 10 days, and λ_3 for the third, i.e. $T_1 = \{30, \ldots, 21\}, T_2 = \{20, \ldots, 11\}, T_3 = \{10, \ldots, 1\}$. The arrival rates for the step function will be referred to as λ_j's or as the vector $\boldsymbol{\lambda}$.

Specifying the contents of the steps $T_1, \ldots, T_N$ - or equivalently, the hyperparameters $\tilde{\tau}_1, \ldots \tilde{\tau}_N$ - is an important choice, as mis-specification can result in biased parameter estimates. In Section 3.2 I provide a heuristic algorithm for determining the steps, and in Section 3.2.2 we study the impact of mis-specification.

3.1.3 Data Description

For the numerical studies, I used two sources of data - real world data and simulated data.

The real-world data is sales data from a small budget hotel. It comprises 80,000 rows, covering transactions recorded from 6/1/2017 to 9/23/2018 with corresponding stay dates between 6/1/2017 and 1/31/2019. Each row of data records all bookings made on a given transaction date for a given stay date, the price those transactions were made at, and the month, day of the week, and demand season. Figure 3.2 shows the head of the data.

Season	Weekday	Month	Stay Date	Transaction Date	DBA	Bookings	Price
MID	Thursday	June	6/1/2017	6/1/2017	0	6	60
MID	Friday	June	6/2/2017	6/1/2017	1	15	75
MID	Friday	June	6/2/2017	6/2/2017	0	2	75
MID	Saturday	June	6/3/2017	6/1/2017	2	5	100

Figure 3.2: The Head of the Real Hotel Data

The simulated data was generated to match our model and mimic hotel data, with Poisson arrivals and a logistic purchase probability function. The user inputs were horizon length, number of stay dates, and true parameters for $\lambda(\tau)$ and P_1. For each stay date, a price was randomly determined for each DBA. Then, for each DBA τ in a horizon for a given stay date s, 24 (one an hour) Bernoulli trials using success probability $\lambda(\tau)$ determined how many arrivals occurred on that day. For each such arrival, a further Bernoulli trial with success probability $P_1(\tau, p_{\tau s})$ where $p_{\tau s}$ is the price assigned for DBA τ of stay date s determined whether or not a purchase occurred. This procedure gives a binary indicator for purchase or no purchase every hour for every day in the sales horizon, for each stay date. All the simulated data in this paper used a 30 day horizon, with 10 stay dates - unless otherwise stated.

To evaluate different methods of step selection, I used three simulated datasets: with one, two, and three steps in their arrival functions. The "Sim1" data has a constant arrival rate,

and indeed displays relatively constant sales throughout the horizon - there is no discernible trend. The "Sim2" data has two arrival rates, split in the middle of the 30 day horizon - sales are relatively low and stable for the first 15 days, and then jump to a higher level and increase from there for the second 15 days. The "Sim3" data has 3 distinct arrival rates, a low arrival rate for the first 10 days, a moderate arrival rate for the next 15 days, and a high arrival rate for the final 5 days of the horizon. The sales slowly increase throughout, before a sharp peak at the end of the horizon - this shape mimics what I have observed in my real data. From the real data, I used data from the high demand season for our results. The periods when demand intensity is highest are those where revenue management is most useful. It is also the period when knowing the arrival rate is most useful, since when demand intensity is high, the arrival rate is more likely to change throughout the sale horizon.

3.2 Specification of $\lambda(\tau)$

In this section, I discuss how to specify the steps of the step-function arrival rate described in 3.1. Given a desired number of steps, I propose both a math program and a heuristic algorithm to select steps, and analyze the performance of our heuristic relative to the math program. I also briefly discuss the implications of mis-specifying the step-function for parameter estimation.

Before estimation via EM can be carried out, we must first define both the number and location (the T_j from Section 3.1.2) of the steps in the step function $\lambda(\tau)$. The number and location of the steps is a crucial choice that is highly dependent on the data. Poorly-chosen steps could obscure the true shape of the arrival rate as a function of time, and result in mis-specified beta parameters.

If one possesses prior information about how the arrival rate changes over time, this informa-

tion can and should be used to specify $\lambda(\tau)$. For example, if there is reason to believe that arrivals are particularly high for the final 3 days of the sales horizon, but relatively constant elsewhere, then a logical choice would be to use 2 steps with a break at DBA = 3.

However, such information may not be available or difficult to obtain. Further, using intuition to specify the steps may miss out on unexpected nuances in the data, or become tedious for long selling horizons.

If we were able to observe the number of arrivals at each day throughout the sales horizon, we would be able to directly calculate the arrival rate for each day. If we then wanted to fit the calculated arrival rates to a step function, we would group consecutive DBAs with similar arrival rates together into steps. In reality, we can't observe arrivals, but we can use similar logic to determine how to break the horizon into steps. Instead of arrivals, we can use the total sales made on a given DBA, and then form steps from groups of consecutive DBA's in the sales horizon with similar sales numbers. The underlying assumption here is that arrivals are proportional to sales.

In fact, one can solve this problem exactly with a math program. Given D DBAs, each with total sales y_d, we want to group the DBAs d into N steps such that a) the DBAs in each step are consecutive, and b) some measure of deviation in the y_d's assigned to the same step is minimized. A formulation for this program is found in Appendix B, where the objective is to minimize the sum of the squared deviations between y_d and the average sales in the step that contains d. Though this program is solvable for small instances (e.g. few DBAs and few steps), it is an MIP and thus for larger instances (e.g. 7 steps, 30 day horizon) can be very slow to solve.

To produce high-quality choices for the number and location of the steps, we developed the Step Variance Minimization Heuristic (SVMH). As the name suggests, this heuristic uses the sum of the variances of the sales in each step as its objective function. The heuristic functions

by executing a series of "moves" from a given initial configuration. These moves are chosen based on criteria intended to ensure that at each iteration, the total step variance is reduced. When all candidate moves fail to reduce the total variance further, the algorithm ends. The moves that SVMH considers are best described in terms of the boundaries between steps, rather than the steps themselves. For example, in the example in Figure 3.3, the boundary between the first two steps is between DBA 20 and DBA 19. Since steps must contain consecutive DBAs, the step selection problem is equivalent to the problem of placing $N-1$ boundaries (the $\tilde{\tau}_j$ from Section 3.1.2) in the sales horizon to create N steps.

SVMH is initialized by providing as input the desired number of steps N, and then finding the sum of all sales recorded for each DBA (e.g. DBA d has y_d sales). Finally, place $N-1$ boundaries arbitrarily in the sales horizon, to form the N steps. This will give the initial configuration for the algorithm to work from.

The first type of move considered by SVMH is called a translation move. A translation move is when the inner boundary between two steps is translated left or right, up to the outer boundary of either step. SVMH evaluates two translation moves per pair of consecutive steps at each iteration. The procedure is as follows: for each of the two steps in the pair, find the pair of DBAs in each for which the difference between their sales (the y_d) is largest. The boundary is moved to be between these DBAs, so that everything to the right (for the first step in the pair, with DBAs strictly greater than those in the second step) or left (for the second step) of the new boundary becomes part of the other step. The two new configurations are then scored according to the objective function and that score is recorded.

The second move type is a "jump" move, so-called because we jump a boundary over another boundary. This has the effect of fusing two steps entirely together, while splitting another step in two. The jump move was developed because the translation moves can sometimes get stuck in a loop, particularly with small steps. Being able to combine two small steps together is a useful way to escape local optima. SVMH considers just one jump move per

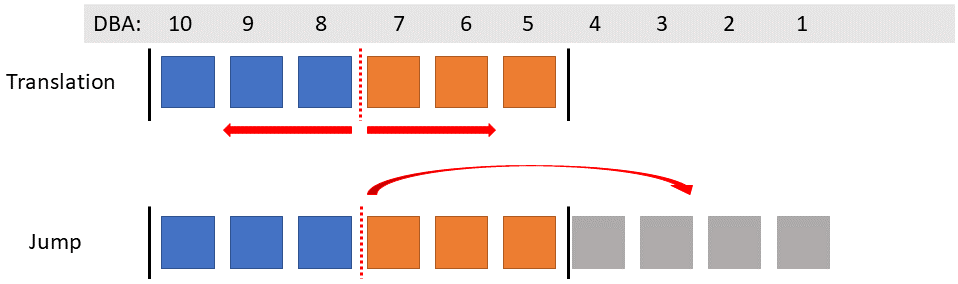

Figure 3.3: The Two Types of Moves Possible in SVMH.

iteration. The two steps that fuse together are the consecutive pair for which the last DBA in the first step in the pair and the first DBA in the second step have the smallest difference in sales. The step chosen to be split in two is the step not included in the pair that will be fused that has the biggest difference in sales between two consecutive DBAs (the same criterion as the translation move, applied across all remaining steps).

This figure shows the two kinds of moves visually. Each color represents a different step. The translation move shifts the boundary between blue and orange to the left or right, but not exceeding the black bars. The jump move removes the boundary between blue and orange, so that they combine into a single step, and places it inside grey, which splits into two new steps.

The selection criteria mean that at each iteration, SVMH considers $2(N-1)+1$ step configurations ($N-1$ consecutive pairs of steps, x2 translation moves +1 jump move). Given an initial configuration, SVMH evaluates the new configurations produced by each of the moves. The move that produces the step configuration with the lowest variance is adopted as the new configuration if the variance is improved over the incumbent solution. Otherwise, the algorithm ends and outputs the current configuration. If the algorithm continues, the moves are recalculated using the new current configuration.

Figure 3.4 shows an example run of SVMH. The red lines represent the places in each step with the biggest pairwise difference in sales between two DBAs. At Iteration 1, in the blue

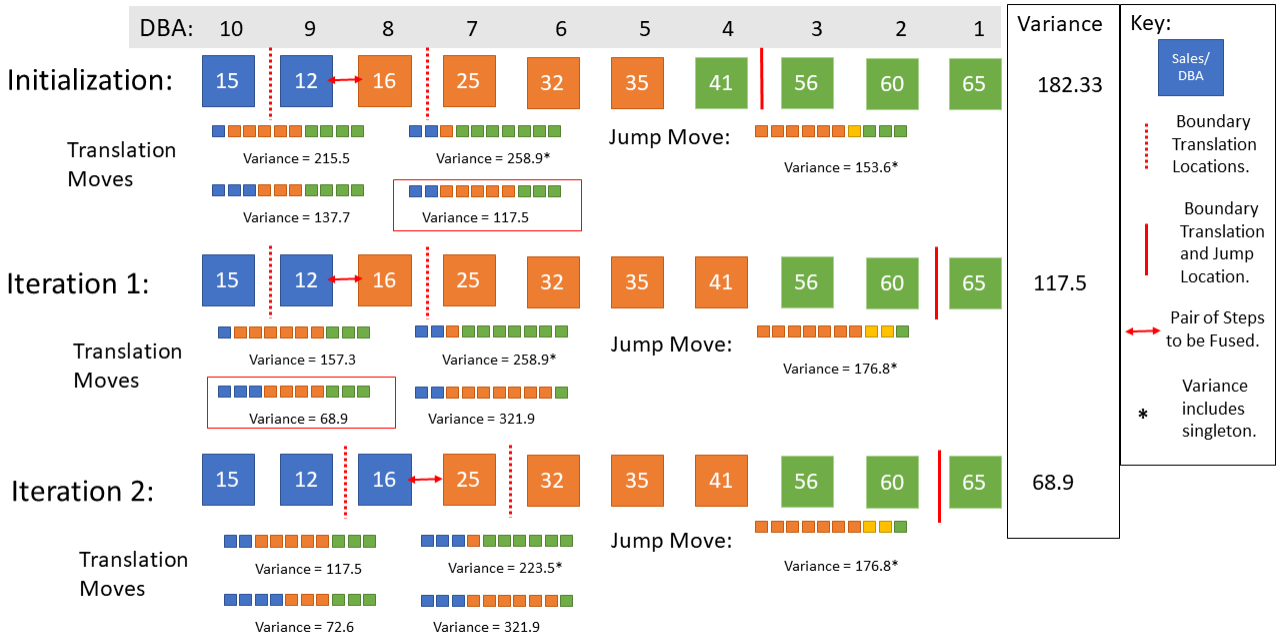

Figure 3.4: Example Run of the SVMH Algorithm

step there is only one choice, between DBA 10 and 9. For the orange step, DBAs 7 and 8 have a difference of 9 (25-16) and for green DBAs 4 and 3 have a difference of 15 (56-41). The translation moves between orange and green are therefore: all orange DBAs ≥ 7 join green or all green DBAs ≤ 4 join orange (in this case just 4). Blue is unchanged for either possibility. For the fuse move, the two closest steps are blue and orange with a distance of 4 (16-12). The only remaining step is green, where the largest pairwise difference is again between 4 and 3. Thus the move puts blue and orange together and splits green into two new steps. Out of all these moves, plus the translation move for blue and orange, the move that adds DBA 4 to the orange step has the lowest variance and is adopted as the new configuration. In Iteration 2 we calculate new moves and choose to add DBA 8 to blue. Finally in Iteration 3 none of the moves improve on our current variance score and the algorithm terminates.

SVMH only considers a small subset of the possible configurations for a given sales horizon (if we have D DBAs and N steps, there are $\binom{D}{N-1}$ possible configurations), which means it

almost always converges rapidly (on a mid-range laptop, essentially instantaneously). The algorithm itself is deterministic, but dependent on the initial configuration. However, testing reveals that given a random initial configuration, it is nevertheless quite stable, and from most randomly chosen initial step configurations will converge to the same solution, which is often optimal. In the next subsection, we discuss the convergence and optimality (with regards to the math program formulation) of SVMH.

3.2.1 Convergence and Optimality of SVMH

There are two key questions to consider about any step selection heuristic. Firstly, how close does the heuristic come to solving the problem it is approximating (our math program)? Secondly, since we cannot observe arrivals, and must use sales as a proxy, how well does SVMH retrieve the true step configuration? In this section I will interrogate both questions. Since the configuration reached through SVMH depends on the initial configuration given to the algorithm, I performed randomized testing to determine its general behavior. For each of our datasets (described in more detail in Section 3.1.3), and number of steps (2 and 3), 2500 random configurations were fed into SVMH, and the output recorded. Alongside this, I also solved the math program for each dataset and number of steps to find the optimal solution.

It is worth noting that the objective function for the math program and SVMH are different (squared deviation from the mean vs. variance). Thus, there were instances where the minimum variance SVMH solution did not match the solution produced by the math program. In such cases, the average optimality gap for SVMH was calculated across the minimum variance solutions achieved in each of the 2500 trials. I defined the optimality gap as (Math Program Sq Error - Avg SVMH Sq Error)/Math Program Sq Error. The procedure for SVMH can be used with other objective functions, and testing was also performed using the

SVMH procedure with the same objective as the math program. However, this squared error version was significantly more sensitive to the initial configuration and produced a larger optimality gap than the variance version. Table 3.2 records the optimality and convergence results for SVMH. SVMH-n will refer to the SVMH solution using n steps.

Dataset	**# Steps**	**Optimal Solution**	**Best SVMH Solution (% of Trials)**	**Avg. Optimality Gap**
Sim3	**2**	5, 0	5, 0 (100%)	0%
Sim3*	**3**	20, 5, 0	20, 5, 0 (100%)	0%
Sim2*	**2**	15, 0	15, 0 (100%)	0%
Sim2	**3**	15, 10, 0	15, 12, 0 (100%)	30.5%
Sim1	**2**	13, 0	10, 0 (37.5%)	26.3%
Sim1	**3**	26, 13, 0	5, 3, 0 (45.9%)	45.9%
High Season	**2**	3, 0	3, 0 (100%)	0%
High Season	**3**	14, 3, 0	3, 1, 0 (74.6%)	19.7%

Table 3.2: Optimality of Results for SVMH, Based on 2500 Random Trials for Each Dataset SVMH achieves the optimal solution in 4 out of 8 cases, other cases differences are both expected and not significant in estimation.

As Table 3.2 shows, SVMH converges to the true optimum in half the instances tested. In the fourth column, the percentage in parentheses shows the frequency with which the best solution in terms of squared error was achieved. SVMH is highly robust to initial start position, and achieved a single solution in 5 out of 8 cases. The two worst cases in terms of consistency of solution were on the Sim1 dataset, which had a constant arrival rate - meaning the algorithm was searching for a solution not present in the data. In every case, the solution that SVMH arrived at most frequently out of the 2500 random trials was also the best solution in terms of total squared error. In 4 out of 8 cases, SVMH converged to the true optimum from every one of the trials. Thus, to our first question, we can be confident that SVMH does solve the math program it approximates quite well, and often to optimality.

There were four cases where SVMH had a non-zero average optimality gap. In each, the best SVMH solution differed from the optimal solution, and in all but one, SVMH arrived at

multiple solutions across the 2500 trials. That SVMH struggled on the Sim1 dataset should not be surprising. The Sim1 dataset has a constant arrival rate, and thus the algorithm was searching for a solution not present in the data - which shows sales remaining fairly constant across the horizon. The two cases with optimality gaps were SVMH-3 on Sim2 and SVMH-3 on High Season. The Sim2 dataset has two arrival rates, and thus again, SVMH was searching for a third step not reflected by the data. Given that SVMH-2 on High Season had an optimality gap of 0%, we may infer that the High Season data also has just two arrival rates. It is also worth noting that for these two cases, SVMH had a significantly smaller optimality gap than a random solution. The SVMH-3 optimality gap for High Season was 19.7%, and for random solutions the gap was 116%. For SVMH-3 on Sim2, the gap was 30.5%, compared to a massive 1065% for random solutions.

For all the cases where the SVMH solution differed from the optimal solution (e.g. A non-zero optimality gap), I also ran EM (see section 3.3 and estimated parameters for both the SVMH and optimal solutions and compared their goodness-of-fit and out-of-sample error. On average across all cases, the optimal solutions had an in-sample mean squared error 0.44% better than the SVMH solution, and 0.21% worse out of sample. In other words, when the math program and SVMH reached different solutions, those solutions performed nearly identically in and out-of-sample after EM estimation. It is also worth noting that for the Sim2 SVMH-3 and High SVMH-3 cases, the SVMH and optimal solutions only differed by one boundary, and the solutions matched for the SVMH-2 cases.

Table 3.2 also speaks to our second question. The true underlying step configuration for the simulated datasets are: constant for Sim1, two steps, the first step ending at DBA 15 for Sim2, and three steps, ending at DBAs 20, 5, and 0 for Sim3. When given the correct number of steps, SVMH correctly identified the configurations for both Sim2 and Sim3. We may further note that the SVMH-2 run for Sim3 and the SVMH-3 run for Sim2 were partially correct - with true boundaries still present in the solution. Though we cannot know the true

arrival rate for the High Season real data, this pattern is repeated: the SVMH-3 solution builds off of the SVMH-2 solution.

3.2.2 What if your Step Configuration is Wrong?

Though our SVMH heuristic is generally effective at providing step configurations, no method is perfect and mis-specification is always possible when dealing with unobserved arrivals. There are two possible ways to mis-specify the step configuration: number and location. If steps are placed in the wrong location, there is a risk of biased parameters. Of course, the arrival rate parameters will be biased, since they directly depend on the step locations. However, the beta parameters may also be biased. If DBA d is placed in Step A, the assumption is that DBA d shares an arrival rate with all other DBAs in Step A. Thus, the wrong arrival rate is likely to be estimated for DBA d. This means that DBA d, and all prices that were recorded for DBA d, might be accorded a different effect on sales than reality, resulting in biased $\boldsymbol{\beta}$ parameters.

In terms of the number of steps, one can specify too many or too few. Specifying too many steps is a less impactful error. While adding additional steps will always improve in-sample fit, the over fitting will hurt out-of-sample performance. However, because the step function structure of $\lambda(\tau)$ does not force any arrival rate to a particular value, extraneous steps often end with arrival rates that are similar to one another. For example, if one was to estimate a two-step configuration on a dataset that in reality had a constant arrival rate, it is likely that the two estimated arrival rates would be almost equal. Furthermore, if the arrival rates of several consecutive steps are estimated to have similar values, this is a likely indication that those steps can be combined into one.

If one specifies too few steps, clearly at least one arrival rate will be incorrect - as in reality the corresponding step should have been split in two. This can also result in biased $\boldsymbol{\beta}$

parameters, in the same way as if the steps were in the wrong place. In general, it is safer to specify more steps and reduce that number if the estimation or SVMH results indicate so (e.g. the SVMH results are unstable over random trials).

3.3 Estimation Using Expectation Maximization

In this section I describe the estimation procedure using the Expectation Maximization algorithm, and detail how we adapt EM to the model.

Our model requires that we estimate values for $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$. Following Talluri and Van Ryzin (2004), I use the Expectation Maximization procedure to estimate these parameters. The data, with K observations, is partitioned into two sets: S, containing all observations k where a purchase (sale) occurred, and $\bar{S}$ containing all non-purchase observations. Then, we further partition both S and $\bar{S}$ into subsets S_j ($\bar{S}_j$), $j \in \{1, ..., N\}$ (e.g., one for each step) which contain only those purchases that occurred in step j of $\lambda(\tau)$. For example, if we refer back to Figure 3.1, S_1 would contain all purchase observations from days 30-21.

With the data thus partitioned, we may construct the following likelihood function:

$$\text{Incomplete Likelihood: } L(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \prod_{j=1}^{N} \prod_{k \in S_j} \lambda_j P_1(p_k, \tau_k; \boldsymbol{\beta}) \prod_{k \in \bar{S}_j} (1 - \lambda_j P_1(p_k, \tau_k; \boldsymbol{\beta})). \quad (3.1)$$

I will refer to this likelihood function as *incomplete* for reasons that will be explained shortly. As Talluri and Van Ryzin (2004) and Ferguson (2020) note, likelihood functions of this form are likely to be difficult to optimize directly. Equation (3.1) is non-concave, as is its logarithm, and as such we risk converging to a local maximum if we attempt direct maximization.

To estimate $\boldsymbol{\beta}$ and $\boldsymbol{\lambda}$ more effectively, I follow TvR and modify our likelihood function. First,

consider Equation (3.2).

$$a(k) = \begin{cases} 1 \text{ arrival at observation } k \\ 0 \text{ otherwise.} \end{cases} \tag{3.2}$$

That is, $a(k) = 1$ if an arrival occurs at observation k, and 0 if not. Of course, $a(k) = 1$ for every $k \in P$ - the crux of this problem will be estimating when $a(k) = 1$ for $k \in \bar{P}$.

Next, we rearrange Equation (3.1) as follows:

$$\prod_{j=1}^{N} \prod_{k \in S_j} \lambda_j P_1 \prod_{k \in \bar{S}_j} (1 - \lambda_j P_1) = \prod_{j=1}^{N} \prod_{k \in S_j} (\lambda_j P_1) \prod_{k \in \bar{S}_j} (\lambda_j P_0)^{a(k)} \prod_{k \in \bar{S}_j} (1 - \lambda_j)^{1-a(k)} \tag{3.3}$$

I will refer to (3.3) as the *complete* likelihood function. The complete likelihood function separates the observations in which a purchase did not occur into arrivals and non-arrivals, with the associated probabilities. Note I have suppressed dependence of P_1 and P_0 on $\boldsymbol{\beta}$, p_k, and τ_k for brevity. Taking the natural logarithm, we obtain Equation (3.4), our complete log-likelihood function.

$$\text{Log Likelihood: } LL(p, \tau; \boldsymbol{\beta}, \boldsymbol{\lambda}) = \sum_{j=1}^{N} \left(\sum_{k \in S_j} \log(\lambda_j P_1) + \sum_{k \in \bar{S}_j} a(k) \log(\lambda_j P_0) + (1 - a(k)) \log(1 - \lambda_j) \right) \tag{3.4}$$

In order to maximize LL, we need to know the value of $a(k)$ for all observations k. While the value for $a(k)$ is clear when $k \in S_j$ for all j (clearly $a(k) = 1$), we have no way of knowing which non-purchase observations had arrivals and which did not. In other words, the value of $a(k)$ is unknown when $k \in \bar{S}_j$, and therefore to maximize LL we also need to estimate $a(k)$ values for all $k \in \bar{S}_j$.

3.3.1 Expectation Maximization

To estimate (3.4), we turn to the Expectation Maximization (EM) algorithm. EM is an iterative method used to estimate parameters from a maximum likelihood function when the parameters depend on latent or unobservable variables. The idea behind EM is that one starts with an *incomplete* likelihood function, which is missing some crucial information and is thus difficult to estimate. In our case, the missing information is whether an arrival occurred at each observation. The incomplete likelihood function is then *completed* by adding in the missing information, as in (3.3). Since the missing information is of course, missing, the expectation step of EM finds its conditional expectation - conditioned on the current parameter guess. Then, the expected values are plugged into the complete likelihood function, which is then maximized - this is the maximization step. The process then repeats, using the new parameter estimates to re-calculate the expectation step, until the parameters converge to within some tolerance $\varepsilon > 0$.

Talluri and Van Ryzin (2004) give several formulas and derivations for applying EM to the censored demand context. In the expectation step, we need to find $E(a(k))$ for all $k \in \bar{P}$. Applying Bayes' Rule, TvR provide an expression for $E(a(k))$, which I have adapted to accommodate our variable arrival rate. Denote $\hat{a}(k)^{(m)}$ as the estimate for $a(k)$ computed at the m^{th} iteration of EM, where

$$\hat{a}(k)^{(m)} = E[a(k)^{(m)}] = \frac{\lambda_{j(k)}^{m-1} P_0(p_k, \tau_k; \boldsymbol{\beta}^{(m-1)})}{\lambda_{j(k)}^{(m-1)} P_0(p_k, \tau_k; \boldsymbol{\beta}^{(m-1)}) + (1 - \lambda_{j(k)}^{(m-1)})}. \tag{3.5}$$

The function $j(k)$ maps observation k to the step j (and corresponding arrival rate λ_j) of $\lambda(\tau)$ in which it occurs. From (3.5), we can see the interpretation of $\hat{a}(k)$ as the probability that a customer arrived for a non-purchase observation.

Next, if we take the derivative of LL (i.e., Equation (3.4)) with respect to λ_j, we obtain a

closed-form solution for each λ_j:

$$\lambda_j^{(m)} = \frac{|S_j| + \sum_{k \in \bar{S}_j} \hat{a}(k)^{(m)}}{|S_j| + |\bar{S}_j|} \tag{3.6}$$

The formula for λ_j in Equation (3.6) is very intuitive. The estimate for λ_j at iteration m is equal to the total number of expected arrivals divided by the total number of observations for step j. The number of expected arrivals is the total number of purchases (which were arrivals for sure) plus the expected number of arrivals from the non-purchase observations.

In iteration m of the EM method the maximization step solves for $\boldsymbol{\beta}$ by maximizing Equation (3.7) with the $\hat{a}(k)^m$ estimates in place of the $a(k)$'s. Note that Equation (3.7) is simply (3.4) with the mth estimates of $\hat{a}(k)$ plugged in and constant terms with respect to $\boldsymbol{\beta}$ omitted (hence, the λ_j's are not present).

$$\boldsymbol{\beta} \text{ M-Step: } \sum_{k \in S_j} \log(P_1(\boldsymbol{\beta})) + \sum_{k \in \bar{S}_j} \hat{a}(k)^m \log(P_0(\boldsymbol{\beta})) \tag{3.7}$$

Since P_1 is a logistic function of $U(\boldsymbol{\beta})$, and $U(\boldsymbol{\beta})$ is linear, LL is concave in $\boldsymbol{\beta}$ and thus its maximum can be easily obtained for any fixed set of $\hat{a}(k)$ values (proof in Appendix A). More generally, any link function $P_1(\boldsymbol{\beta})$ that is concave in $\boldsymbol{\beta}$ will also suffice, as concavity of LL is preserved.

The EM algorithm is initialized by choosing initial values for all λ_j and the components of the $\boldsymbol{\beta}$ vector. As Talluri (2009) and Ferguson (2020) note, an unfortunate choice of initial parameters for EM can result in poor convergence or unstable estimates. However, in testing I have found that choosing "reasonable" initial parameters tends to give good results. All of our numerical results were produced using initial parameters of $\lambda_j = 0.5 \ \forall j$, $\beta_0 = 1$, and $\beta_i = 0.001$ for all $i \neq 0$. To ensure the robustness of our estimates, I further carried out tests with randomly-selected initial parameters and found that the final parameter estimates

remained consistent despite changing the initial values.

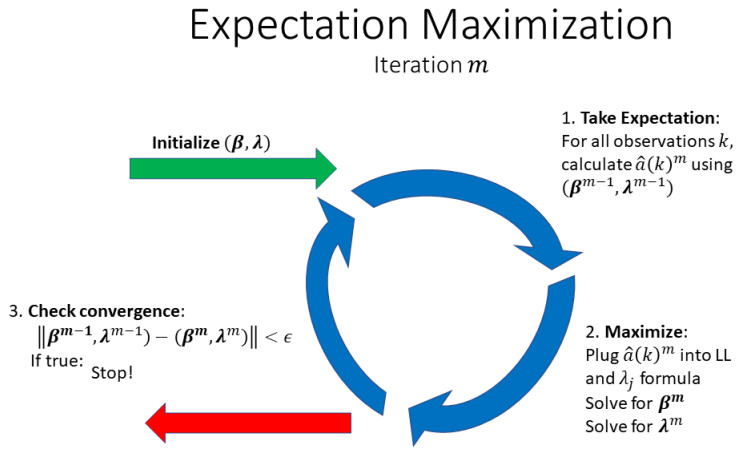

Figure 3.5: The Expectation Maximization Process

Convergence of EM is achieved when $||(\boldsymbol{\lambda}^{(m-1)}, \boldsymbol{\beta}^{(m-1)}) - (\boldsymbol{\lambda}^{(m)}, \boldsymbol{\beta}^{(m)})|| < \varepsilon$ for a suitably-chosen small $\varepsilon > 0$. In this paper, I used the L2 norm and ε =1E-05. Figure 3.5 summarizes how the EM algorithm is applied to our demand estimation context.

3.4 Expectation Maximization Results

In this section, I evaluate the performance of the SVMH arrival rate step selection method against two baselines - a constant arrival rate, meaning the step function had only one step - or in other words the classical TvR method, and using evenly-spaced steps as a naive method, namely splitting the selling horizon in half (steps ending at DBA 15 and DBA 0) or thirds (steps ending at DBAs 20, 10, and 0). All three methods will be evaluated on each of the three simulated data types described in Section 3.1.3. I will show that using a variable arrival rate produces significant in sample and out of sample performance improvements over a constant arrival rate, and that when the data indicates a variable arrival rate, the SVMH

procedure is the clear winner in- and out-of-sample.

3.4.1 Train and Test Procedure

The train and test procedure used for all numerical results in this paper differs slightly for simulated and real data. For simulated data, 5 training datasets were generated for each data type according to the procedure described in Section 3.1.3. A separate EM run estimated the parameters for each of the 5 training sets, and the resulting parameters were tested on the other 4 training sets. The in- and out-of-sample results presented in Tables 3.3 and 3.4 are averages of the 5 in-sample Mean Squared Error (MSE) statistics, and the 5×4 out-of-sample MSE statistics.

For the real data, 3 training sets and 3 test sets were randomly sampled from a data pool. An EM run estimated the parameters for each training set, which were tested on each testing set. The in-sample MSE numbers presented are averages of the in-sample fit for each training set, and the out-of-sample MSE numbers are averages of the 3×3 out-of-sample readings.

To ensure consistent results for each data type from SVMH, the input to the algorithm - the total sales for each DBA - was the average total sales for each DBA across the 5 (or 3 for the real data) training sets, rounded to the nearest integer.

3.4.2 Results and Discussion

Tables 3.3 and 3.4 show the in-sample and out-of-sample results for each of our datasets and step configurations. Mean squared error (MSE) was calculated by comparing predicted sales per day with the actual sales per day. The top row of results in the data, in grey, is the constant step configuration. Note that when comparing results to one another, one should look down the columns and not across the rows. The Even step configurations, splitting

Method	# Steps	Simulated Data			Real Data
		Sim3	Sim2	Sim1	High Season
Constant	**1**	2.07	2.98	3.75	5.36
Even	**2**	2.02	2.70	3.74	5.35
Even	**3**	2.06	2.95	3.73	5.31
SVMH	**2**	1.87	2.70	3.74	5.03
SVMH	**3**	1.83	2.68	3.73	5.00

Table 3.3: In-Sample Mean Squared Error for All Datasets by Step Selection Method. SVMH achieves the lowest MSE in every case.

the sales horizon into two or three even steps are next, followed by the two and three step solutions provided by SVMH, the details for which can be found in Table 3.2. The cells highlighted in yellow are the best performing (lowest MSE) configuration for each dataset. The 3 step SVMH configurations had the best in-sample fit for all 4 datasets. Out of sample, SVMH configurations performed the best in 3 of the 4 datasets, except the Sim1 dataset.

The in-sample results in Table 3.3 support the ability of SVMH to correctly determine the arrival rate function. The SVMH step configurations consistently had the best fit, even when the number of steps was incorrect. For example, the SVMH-2 model had better in-sample fit than the Even-3 model on the Sim3 data, even though the Sim3 dataset has 3 arrival rates. As I discussed in 3.2.2, correct placement of steps is typically more important than having the correct number. Table 3.3 provides strong evidence that SVMH performs well in this respect.

Method	# Steps	Simulated Data			Real Data
		Sim3	Sim2	Sim1	High Season
Constant	**1**	2.14	3.12	3.87	6.50
Even	**2**	2.11	2.87	3.88	6.50
Even	**3**	2.15	3.14	3.88	6.44
SVMH	**2**	1.98	2.87	3.91	6.14
SVMH	**3**	1.95	2.87	3.93	6.12

Table 3.4: Out-of-Sample Mean Squared Error for All Datasets by Step Selection Method SVMH achieves lowest MSE in all cases except Sim1, which has a constant arrival rate.

A constant arrival rate was the best choice only for the Sim1 dataset out-of-sample, which one

might expect since that dataset in fact has a constant arrival rate. However, as a testament to the robustness of using a step function arrival rate, the step functions for the variable configurations were all more or less flat for the Sim1 dataset. In the remaining three sets, both SVMH configurations performed significantly better out-of-sample than both the naive even method and in particular the constant arrival rate.

Figure 3.6 shows the percentage difference between the MSE results of the variable arrival models (SVMH and Even) and the constant arrival model for each dataset. A positive percentage represents an improvement over the constant model, or a smaller MSE. These plots allow us to view the performance of SVMH by the same metric across all datasets. We can see clearly from these plots that a variable arrival rate not only provides superior in-sample fit to a constant arrival rate, but also improves out-of-sample prediction significantly, by between 6% and 10%. Furthermore, the SVMH step configurations are considerably better than both a constant arrival rate and the naive even method. On the datasets where a variable arrival rate seemed to be appropriate (Sim3, Sim2, High), SVMH outperforms the naive method by a significant margin.

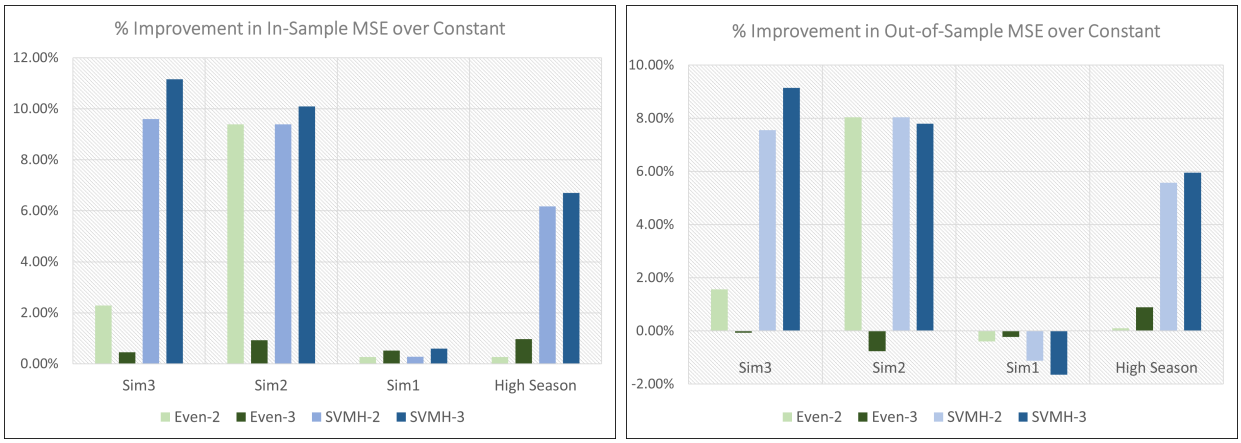

Figure 3.6: Percentage Difference in MSE between Variable Arrival Rate Models and Constant Arrival Model.
SVMH achieves an 8%-10% improvement over the constant (TvR) arrival rate in-sample, and 6%-8% out-of-sample.

Chapter 4

Regularized Expectation Maximization

In this Chapter, I describe my variant of EM, Regularized Expectation Maximization (REM). REM was developed to improve upon the notoriously slow and unpredictable convergence times of EM estimation. In Section 4.1 we will define REM with respect to the censored-demand problem from Chapter 3, and establish some general theoretical properties. As well, I give some intuition as to why REM is able to reduce convergence time. Then, Section 4.2 gives numerical results for the improved convergence, and conduct a detailed numerical study of the predictive properties of REM models compared to EM.

4.1 Intuition and Theoretical Properties

In this section, I define the REM algorithm, discuss the intuition behind REM, and then establish some theoretical properties.

Expectation Maximization is well established in the literature as a method for estimating

latent or unknown parameters. A common criticism of EM is its relatively slow rate of convergence. Convergence speed can vary widely between datasets, initial parameters, and in our case, different arrival rate step configurations. Take for example, the constant arrival estimation for two of the Sim1 datasets. Both datasets used identical data generating processes, and the same initial parameters. The EM estimation process converged in 1052 iterations, or about 7 minutes for one set, while the other took 18,683 iterations and 129 minutes. Convergence speed can also vary widely for different step configurations. In the same set of results, one dataset converged in 1012 iterations (3.4 minutes) when using the 2-step SVMH configuration, but took 13,986 iterations (130 minutes) for a 2-step evenly split configuration. Recall that I consider the EM algorithm to have converged when the parameters (in our case λ_j's and $\boldsymbol{\beta}$'s) stop changing between iterations; i.e., when the parameters change by less than 1E-05.

As these anecdotes suggest, discovering a general rule for when a particular model might be slow to converge has proved to be difficult. Two datasets that appear functionally identical can converge at entirely different rates. Convergence is typically fairly slow, both in terms of iterations and time. For the results for the classical EM algorithm (3.3 and 3.4) , the simulated datasets converged on average in 9,138 iterations (63 minutes) with a minimum of 942 iterations (6.4 minutes) and a maximum of 42,784 iterations (290 minutes). The real data converged in an average of 11,872 iterations (128 minutes), with a minimum of 47 iterations (30 seconds, and a clear outlier), and a maximum of 30,330 iterations (338 minutes). Though the shorter convergence times are certainly manageable, an average convergence time of over an hour for both real and simulated datasets is less than ideal.

This variation in convergence time can sometimes be solved by choosing different initial parameters for EM. However, this is not an exact science (the typical procedure is to randomize initial parameters), and I have found in my trials that the wrong choice for initial parameters can lead to divergence or a non-optimal solution. To reiterate, all runs used the same set of

"reasonable" initial parameters (based on intuition on the approximate magnitudes of the parameters): $\boldsymbol{\lambda} = \{0.5, \ldots, 0.5\}$, and $\boldsymbol{\beta} = (1, 0.001, 0.001)$. I found that this initial set of parameters always converged, and always to a reasonable solution. From there, I sought to improve convergence.

Our Regularized Expectation Maximization (REM) algorithm significantly reduces convergence time, while achieving similar out-of-sample performance to, and preserving many of the desirable properties of, classical EM. In general, regularization refers to adding a penalty term to the objective function of an optimization problem. Typically, it is used to shrink the parameter space, such as in LASSO regression, where unimportant parameters are forced to zero (e.g., Tibshirani (1996), Hoerl and Kennard (1970)). In the Regularized Expectation Maximization (REM) algorithm, we add a penalty term to the objective of the $\boldsymbol{\beta}$-maximization step, but for a different purpose. The goal of our penalty term is to restrict the difference between $\boldsymbol{\beta}^{(m)}$ and $\hat{\boldsymbol{\beta}}^{(m-1)}$, the $\boldsymbol{\beta}$ values from the current and previous iterations.

The expectation step of REM is unchanged from that of EM. The objective function for the part of the maximization step, which optimizes over the $\boldsymbol{\beta}$'s, becomes:

$$\max_{\boldsymbol{\beta}^{(m)}} \quad (-w) \left|\left| \boldsymbol{\beta}^{(m)} - \hat{\boldsymbol{\beta}}^{(m-1)} \right|\right| + \sum_{k \in S_j} \log(P_1(\boldsymbol{\beta}^{(m)})) + \sum_{k \in \bar{P}} \hat{a}(k)^{(m)} \log(P_0(\boldsymbol{\beta}^{(m)}))$$

where m is the iteration number, w is a positive penalty parameter, and $|| \cdot ||$ refers to the L2 norm.

A key property of the EM algorithm is that at every iteration, the value of the incomplete likelihood function is guaranteed to increase. Variations of EM that share this property are called Generalized Expectation Maximization (GEM) algorithms. The GEM property allows GEM algorithms to share several useful convergence properties with EM, depending on the structure of the completed likelihood function (see Wu (1983) for details).

Theorem 1. *Regularized Expectation Maximization satisfies the GEM Property. That is, at*

an arbitrary iteration m, $L(\theta^{(m+1)} \geq L(\theta^{(m)})$, where L is the incomplete likelihood function, and $\theta^{(m)}$ is the set of parameters estimated in iteration m. For proof see Appendix D.

As our results below in Section 4.2 show, REM can drastically reduce convergence time compared to EM on the same data. Though this is a highly desirable result, we should first consider what REM is doing during estimation, and how the estimated parameters are affected, compared to EM.

To gain some intuition of why REM works, recall that EM is an iterative method for solving the maximum likelihood estimation problem (3.3). At each iteration, the sub-problem (3.4) is solved with fixed estimates for $a(k)$'s to produce an updated estimate for $\boldsymbol{\beta}$. While the sub-problem (3.4) is a convex optimization problem in the $\boldsymbol{\beta}$'s and so itself has no convergence issues, the overarching problem (3.3) that EM is solving is a non-convex optimization problem. As a consequence, we must be careful to shape how the iterates of $\boldsymbol{\beta}^{(m)}$ behave from one iteration to the next in the overarching EM method, to encourage convergence in the overarching problem (3.3).

In my experiments, I observed that when using standard non-regularized EM, the $\boldsymbol{\beta}$'s would often rapidly increase (or decrease) away from their initial values in the early iterations, before having to come back down (or respectively, up) to converge. This behavior is particularly prevalent in the intercept, β_0, which in some cases doubled between one iteration and the next. A possible cause of this behavior is that the EM algorithm is somewhat myopic. The goal of each iteration's sub-problem is only to maximize the likelihood given the parameter values calculated in the previous iteration.

As we iterate through the EM algorithm, Equation (3.7) is solved at each M-step, which generates a sequence of $\boldsymbol{\beta}^{(m)}$'s. If the $\boldsymbol{\beta}^{(m)}$ sequence is volatile, i.e., the values change materially from one iteration to the next, this in turn causes the $\boldsymbol{\lambda}^m$ sequence to vary, as well as the $\hat{a}(k)^m$ sequence, which in turn causes the objective function (3.7) to change

materially in each iteration, which results in changes to $\boldsymbol{\beta}^{(m)}$. As you can see, this argument is circular, and constitutes a form of "feedback loop," where changes in the values in the $\boldsymbol{\beta}^{(m)}$ sequence reverberate and cause additional changes to the $\boldsymbol{\beta}^{(m)}$ sequence. As one might imagine, this problem is particularly acute when β_0 is doubling in between iterations. The idea with REM is to dampen the effect of this feedback, by limiting the extent to which $\boldsymbol{\beta}^{(m)}$ changes at any given iteration. This is similar to reducing the step sizes made at each iteration, which improves convergence as we are less likely to overstep and course-correct, and so are able to converge with fewer steps.

A further analogy for how REM works is that of a trust-region method (see for example, Nocedal and Wright (2006)). In numerical optimization, a trust-region method finds a region in which one can accurately locally approximate the objective (usually as a quadratic), and then takes a step in the direction indicated by the approximation. In a way, REM's penalty term creates a soft trust-region centered on the previous parameter values. The further away a solution is from the center of the trust-region, the more heavily it is penalized. Thus, we can be sure that the sequence of $\boldsymbol{\beta}$ solutions will not head off in an unproductive direction. I refer to this trust region as "soft" since there are no firm boundaries, just a continuously increasing penalty. The so-called size of the trust region – how far solutions can deviate – is determined by the penalty parameter w. If w is large, the trust-region will be small and vice-versa.

In general, running EM and REM on the same data may produce two different sets of parameter estimates. I have observed that typically, as we increase the penalty weight w from zero (EM case) to a higher value (more conservative REM case), the β slope terms for price and DBA change only modestly, while the most significant salient changes in the parameters are that the intercept (β_0) tends to grow smaller while the arrival rates (λ's) tend to grow larger. However, REM parameter estimates consistently perform similarly to those of EM in terms of in-sample fit and out-of-sample performance, and have similar likelihood

values.

The reason REM can arrive at different parameters than EM while still achieving similar performance and likelihood value is the presence of a "likelihood plateau". Suppose one runs EM ($w = 0$) and REM for some weight $w > 0$ on the same data, and obtains parameter estimates $(\boldsymbol{\lambda}_{EM}, \boldsymbol{\beta}_{EM})$ and $(\boldsymbol{\lambda}_{REM}, \boldsymbol{\beta}_{REM})$. By taking a series of affine combinations of these two solution sets:

$$(\boldsymbol{\lambda}_{New}, \boldsymbol{\beta}_{New}) = x(\boldsymbol{\lambda}_{EM}, \boldsymbol{\beta}_{EM}) + (1 - x)(\boldsymbol{\lambda}_{REM}, \boldsymbol{\beta}_{REM}),$$

for $x > 0$ we obtain a set of other possible parameter values. If we then plug each set of parameters into our incomplete likelihood function (3.1) and plot the resulting likelihood we see a cross section of the likelihood surface. The shape of this surface is a relatively flat plateau, with steep drops on either side. It is this structure that REM exploits, by converging more quickly to a solution with a slightly lower likelihood that lies on this plateau.

Figure 4.1 shows examples of these plateaus (using $w = 2$) from each of our types of dataset. For each of them, I observe steep slopes on the sides for extreme weights, and a central plateau. Though the EM solution (at x = 0) will always have the highest likelihood value, it always sits in the center of a flat plateau. These flat areas represent different combinations of parameters that all produce similar likelihoods to that produced by the EM solution.

4.2 REM Convergence and Comparison to EM: Numerical Results

In this section, I will demonstrate the substantially faster convergence of REM over EM. I compare the convergence times for the EM runs in Section 3.4.2 with REM runs on the

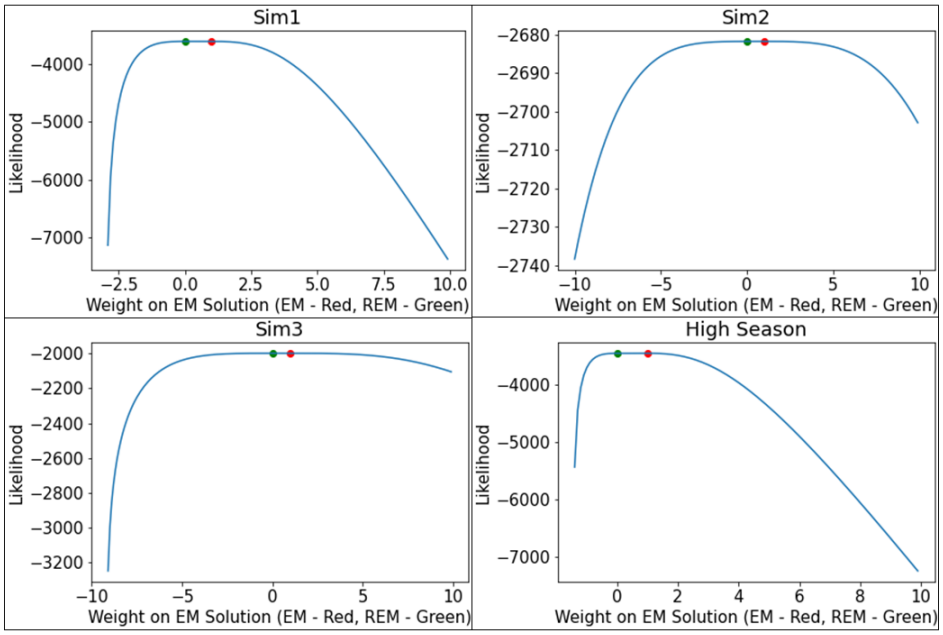

Figure 4.1: Example Likelihood Plateaus for Each Data Type. x-axis: affine weight (EM: x = 1, in red, REM: x = 0, in green), y-axis: likelihood
EM and REM achieve almost identical likelihoods and lie on a plateau in the incomplete likelihood function.

same data. I will also show that in addition to having similar likelihood values to EM, REM solutions also perform similarly out of sample, and produce almost identical revenue curves.

Table 4.1 shows the average percentage decrease in iterations between the original EM runs shown in Tables 3.3 and 3.4 and the regularized versions, using $w = 2$. I report the reduction in iterations here, as the clock time can depend on the machine used for the EM runs. However, the reduction in clock time is nearly identical to the reduction in iterations (an equivalent table to Table 4.1 for time can be found in Appendix E, Table E.1). The regularized runs consistently converged in fewer iterations than the non-regularized algorithm. In each case, the regularized algorithm solved the exact same problem as the non-regularized problem - the same data and the same initial parameters. Across all the data, and all step configurations, the REM runs needed 81% fewer iterations than the EM runs.

As Table 4.1 shows, the impact of REM on convergence varies by dataset and by step

Method	# Steps	Simulated Data			Real Data	Average
		Sim3	Sim2	Sim1	High Season	
Constant	**1**	81%	85%	97%	62%	81%
Even	**2**	65%	97%	96%	45%	76%
Even	**3**	40%	68%	97%	33%	60%
SVMH	**2**	80%	97%	98%	97%	93%
SVMH	**3**	90%	97%	98%	97%	96%
Average		71%	89%	97%	67%	**81%**

Table 4.1: Average Percentage Decrease in Iterations for REM Models Compared to EM on the Same Data.
REM decreases iterations required for convergence by 81% on average.

configuration. The Sim1 dataset saw the greatest improvement in the number of iterations (97% on average) compared to the other data types, while the convergence for the SVMH configurations improved the most across all datasets, compared to the constant or evenly-spaced step configurations (93% and 96% for SVMH-2 and SVMH-3). Interestingly, the Sim1 data was neither fastest nor slowest in terms of EM convergence (coming middle with an average convergence of 8119 iterations). The same is true for the SVMH runs, which converged in the most (2 step) and second least (3 step) iterations for their EM runs (11436 and 10460 iterations on average respectively).

Out of the 105 EM runs used for the results in this paper, 93 saw decreases in convergence time when the same data and step configuration was used in an REM run. The same weight ($w = 2$) was used in all instances for consistency, even though there were clearly cases where alternative values for w could yield additional benefits. I have observed that, generally speaking, there is an "elbow shaped" relationship between w and convergence time. That is, as w increases, convergence time drops until a minimum is reached, at which point convergence time begins to increase again. Increasing w also worsens the in-sample fit for the model. The key to choosing a good weight then, is to choose a weight high enough to reduce convergence time, without compromising the goodness of fit.

Choosing the perfect weight w is highly-dependent on the training data used, and from our

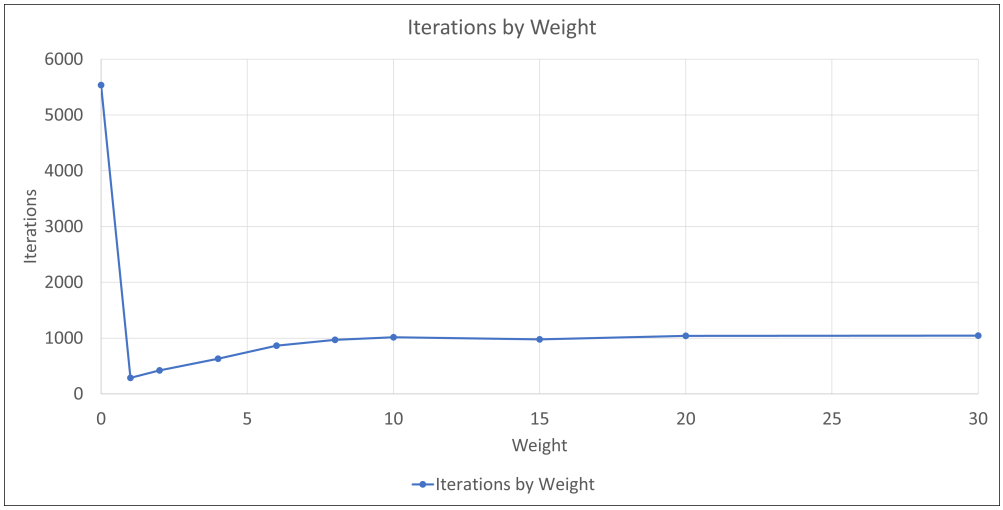

Figure 4.2: Iterations to Convergence for Sim3 Dataset over Various REM Weights
Iterations follow an elbow plot with min at $w = 1$ and flat "forearm"

experience it is difficult to generalize a rule to select it. How a given penalty weight affects a given EM run depends on the size of the training data, the step configuration, the initial parameter values for EM, and characteristics of the data itself. Even two training sets drawn from the same data source may require different weights to produce a similar reduction in convergence time.

Unfortunately, I have been unable to determine a general relationship between choice of weight w, convergence time, and parameter estimation. As noted previously, convergence time, and thus weight choice, depends on start position, the size of the data, the complexity of the data, and the step-configuration. With so many competing factors, it is very possible that if a closed-form relationship between w and convergence does exist, it is so complex as to be effectively useless. However, we can observe how various choices for w impact REM estimation on the same dataset. Using 3 individual training sets, one each from the Sim1, Sim2, and Sim3 data I ran REM using $w = 4, 6, 8, 10, 15, 20, 30$ (in addition to the EM and $w = 2$ runs already discussed). We observed how convergence time, in- and out-of-sample error, and estimated parameter values changed as w was increased. Figure 4.2 shows the number of iterations needed for convergence for each value of w for the Sim3 data. There

Figure 4.3: In-Sample and Out-Of-Sample MSE for a Sim3 Dataset over Various REM Weights
Variation in MSE is very slight, though MSE increases at higher weights.

is a sharp drop in iterations between the EM run ($w = 0$) and $w = 1$, with a minimum at $w = 1$. This is the "elbow" shape we discussed earlier, with a relatively flat "arm" as w increased - the minimum iterations was 289, and the "arm" flattens out at around 1000.

Weight	Arrival Rates	Intercept	DBA	Price
TRUE	20: 0.100, 5: 0.400, 0: 0.800	2.000	-0.090	-0.022
0	20: 0.224, 5: 0.356, 0: 0.515	1.707	-0.118	-0.016
1	20: 0.198, 5: 0.301, 0: 0.447	2.187	-0.127	-0.018
2	20: 0.207, 5: 0.321, 0: 0.472	1.988	-0.123	-0.017
4	20: 0.229, 5: 0.368, 0: 0.532	1.607	-0.116	-0.016
6	20: 0.256, 5: 0.426, 0: 0.606	1.259	-0.110	-0.015
8	20: 0.267, 5: 0.450, 0: 0.637	1.137	-0.108	-0.014
10	20: 0.271, 5: 0.457, 0: 0.646	1.101	-0.107	-0.014
15	20: 0.275, 5: 0.464, 0: 0.654	1.063	-0.107	-0.014
20	20: 0.278, 5: 0.467, 0: 0.656	1.046	-0.107	-0.014
30	20: 0.283, 5: 0.469, 0: 0.655	1.030	-0.108	-0.014

Table 4.2: Parameter Values for a Sim3 Dataset for Various REM Weights
Parameters become inelastic to weight choice at higher weights after initial change.

Figure 4.3 shows the in-sample and out-of-sample MSE for the same Sim3 dataset over increasing values of w. Note that the worst in-sample and out-of-sample fit is for $w = 1$, the weight that minimized convergence time. It is fairly common, that the value of w that

minimizes convergence time has the worst in-sample fit (e.g. the same was true for the Sim1 and Sim2 data I tested). However, note that at higher weights, both in- and out-of-sample fit improve, and even improve on the MSE values for the EM run. The weight that minimizes in-sample and out-of-sample fit is $w = 8$, which converged in 967 iterations - still an 81% improvement over EM. Out-of-sample fit did begin to degrade at higher weights, though it is worth drawing attention to the scales on both plots - the difference between the best and worst MSE for both was around 0.01, or 0.5%. Though fit does change both in and out of sample as the penalty weight is increased, the change is very small.

Table 4.2 shows the estimated parameter values for the Sim3 dataset across each REM run. In general, after an initial increase at $w = 1$, as the weight is increased, β_0 gets smaller in magnitude, while the $\boldsymbol{\lambda}$'s increase in magnitude (again after an initial decrease). The slope β's change in proportion to the changes in β_0. An interesting observation is that the changes in the parameters become less and less elastic to changes in weight. The change in parameters between $w = 2$ and $w = 4$ are significantly larger than the changes between $w = 20$ and $w = 30$. Note also that the changes in parameters are in proportion - though the λ_j's increase steadily, they do so more or less in proportion to one another. If we recall the likelihood plateau from earlier, this implies that at higher weights the REM solutions are not falling off the plateau so much as moving around it. This property is useful, especially when allied to our observations about convergence time and fit, since one can set a high weight to start with and be confident of fast convergence, similar fit to EM, and that the parameter estimates will not be substantially affected by the weight. The weight can then be reduced, either to improve convergence time, fit, or both. The trends shown in Figure 4.3 and Table 4.2 were replicated in the other two datasets I studied. Though applying REM does give different parameters to EM, the effects of choosing different weights are predictable and relatively small - and the greatest effect is a dramatic reduction in convergence time. When selecting a penalty weight for a given dataset, it is a good idea to perform some degree of search. Starting the search at higher values for w (e.g., $w > 5$), running REM and then

decreasing the weight if desired is a good approach. A high weight to start will increase the likelihood of fast convergence, and the weight can then be decreased to improve fit until convergence starts to worsen significantly.

Method	# Steps	Simulated Data			Real Data	Average
		Sim3	Sim2	Sim1	High Season	
Constant	**1**	2.8%	-0.5%	0.8%	0.1%	0.8%
Even	**2**	1.4%	0.3%	0.9%	0.2%	0.7%
Even	**3**	1.8%	-0.5%	1.0%	-0.1%	0.5%
SVMH	**2**	-0.3%	0.3%	-0.1%	0.1%	0.0%
SVMH	**3**	-0.1%	0.1%	-0.2%	0.1%	0.0%
Average		1.1%	0.0%	0.5%	0.1%	**0.4%**

Table 4.3: Average Percentage Difference in Out-of-Sample MSE between EM and REM Solutions
(positive percentage indicates REM MSE > EM MSE)
REM and EM solutions show negligible performance differences out-of-sample.

Though REM can substantially decrease computation time, an important consideration is how it impacts out-of-sample performance. A decrease in convergence time is of little use if the estimation is of noticeably lower quality. Though we know the EM and REM solutions have similar likelihoods, this is not necessarily a guarantee of out-of-sample performance. Table 4.3 shows the average percentage difference between the out-of-sample performance of the non-regularized estimation and the regularized estimation. A positive percentage indicates that the regularized model had a worse (higher) MSE than the EM model. On average, the EM and REM solutions differed by just 0.4% out-of-sample. Interestingly, the constant arrival rate models differed the most, by 1%, while the SVMH REM solutions were actually exactly the same, on average, as their EM counterparts. Overall, the out-of-sample performance difference between EM and REM solutions is negligible.

I also plotted the revenue curves for both the EM and REM solutions, and compared them. The revenue curves were plotted for the appropriate price or DBA range, with the other parameter held fixed, and show how the model predicts revenue, calculated as predicted sales $\times$ price over the range of the dependent variables. These plots show the actual predictive

behavior of a model. Figure 4.4 shows these revenue curves over the range of prices used, with DBA held fixed at 15, for examples of each dataset. The REM and EM predicted revenue curves are almost identical, and their optimal prices are very close, at most differing by a few dollars in price, and a few cents in revenue, in all 4 cases. Therefore the two models are effectively interchangeable for price optimization purposes.

The comparison between EM and REM for the remaining datasets demonstrates that the two solutions are functionally the same. Though parameter estimates do vary between EM and REM, the two solutions have a negligible difference in out-of-sample mean squared error, very similar likelihood values, and finally near-identical predicted revenue curves. All the while, the REM solutions converged in 81% fewer iterations, on average.

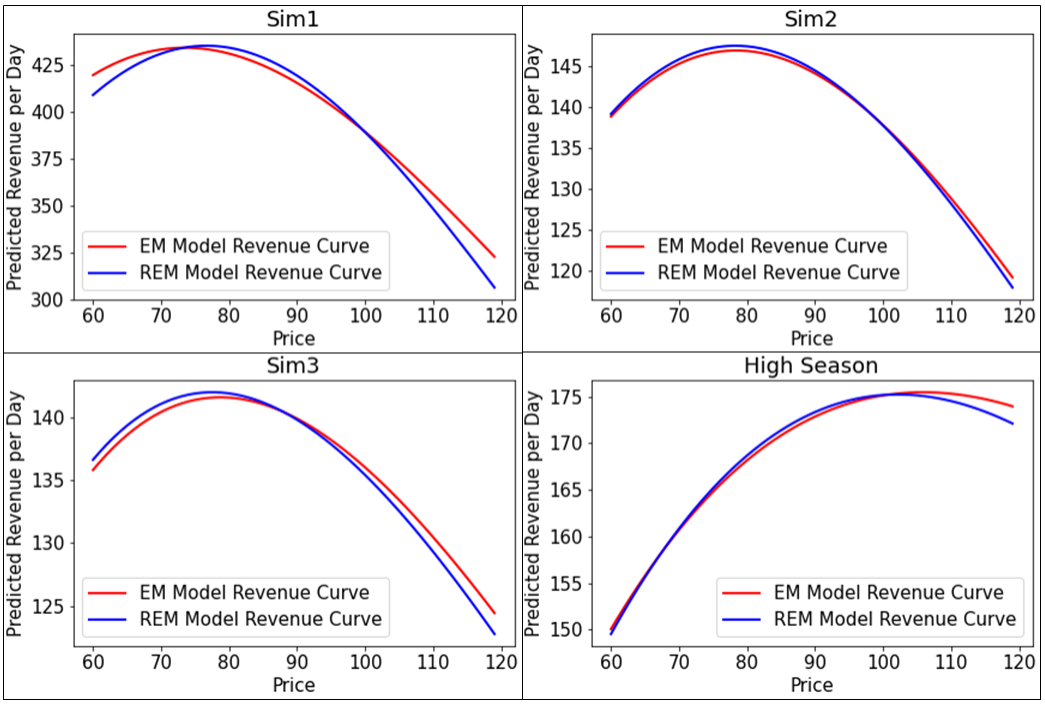

Figure 4.4: Predicted Revenue Curves for Example EM and REM models. REM and EM models show very similar predicted revenue curves.

Chapter 5

Conclusions

A key component of any dynamic pricing policy is a demand function that depends in some way on price. When demand is presented via sales data, this is not a straightforward task, since demand is only observed when a sale is made. This problem is well studied, particularly in the multi-product case, and a common solution is to estimate an arrival rate alongside parameters for purchase probability.

The various multi-product methods in this field do not work in the single-product case. Many smaller or budget hotels only have a single room class, or their rooms are differentiated by very little (eg. two queen beds vs. one king, with no price difference). These hotels also may not have access to detailed information that might allow them to estimate an arrival rate (eg. online click rates).

Being able to estimate the arrival rate is still key to accurate demand estimation in the single-product case. As illustrated in Figure 1.1, if the arrival rate changes during the sales horizon, one is at risk of badly mis-specifying key parameters.

In this work, I based my model on the work of Talluri and Van Ryzin (2004), modified with

a time-varying arrival rate. The arrival function is a step function, which provides both flexibility and simplicity. Key to using a step-function is determining where in the sales horizon to place the steps. I present both an exact math program and the SVMH heuristic algorithm to choose steps such that each step is internally similar in terms of sales. I show that SVMH not only provides a good approximation of the math program, but also achieves high quality solutions.

Next, I compared the in- and out-of-sample mean squared error of our method to both a naive selection method and a constant arrival on 4 different types of data. The data includes simulated data using 1, 2, and 3 arrival rates, and a set drawn from a real hotel. The SVMH solutions are consistently superior both in and out-of-sample to both the constant arrival rate and the naive method. Out-of-sample testing shows improvements of up to 10% in demand prediction error over a constant arrival model. According to Lee (1990), a 10% improvement in prediction accuracy can translate to up to a 3% increase in revenue in the airline industry. Though the correspondence to the hotel industry is likely not one-to-one, the increases in forecast accuracy shown by our model could help a small hotel such as the one we acquired our data from accrue tens of thousands of additional dollars in revenue annually.

In the course of my work, I repeatedly ran into problems with the notoriously slow convergence of Expectation Maximization. EM by nature has slow and unpredictable convergence, which is only exacerbated by the step-function arrival rate. My Regularized Expectation Maximization (REM) algorithm significantly speeds up convergence by adding a penalty term to each iteration's maximization problem over the $\boldsymbol{\beta}$'s. REM shares key properties with EM, and numerical evidence shows that EM and REM solutions perform similarly out-of-sample, have near identical likelihoods, and predict similar revenues. In our tests, REM decreased the iterations required for convergence by 80% on average, with some models improving by over 95%.

Further work could focus on further exploration of the step-function arrival rate, such as

investigating the relationship between optimality in terms of step variance or squared error, and in-sample or out-of-sample performance. Alternatively, one could move away from the step function entirely, and examine other functional forms, such as piecewise functions. REM has many potential future directions, including establishing theoretically the relationship between penalty weight and convergence, and testing the algorithm in a more general setting.

This work provides insight into a little studied piece of the censored demand estimation field. I examine the practical considerations involved in implementing a variable arrival rate in the TvR model, and provide a method for specifying the variable arrival function. Those models that use my SVMH method to select the arrival rate step function show significantly improved out-of-sample demand prediction over both the original TvR model and a naive variable arrival model. Allied with REM, this work provides a method to estimate a demand function using censored sales data that is both fast and accurate.

Bibliography

Tarek Abdallah and Gustavo Vulcano. Demand estimation under the multinomial logit model from sales transaction data. *Manufacturing & Service Operations Management*, 23 (5):1196–1216, 2021.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

Gabriel Bitran and René Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5(3):203–229, 2003.

James G Booth and James P Hobert. Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(1):265–285, 1999.

Sanghoon Cho, Mark Ferguson, Pelin Pekgun, and Jongho Im. Robust demand estimation with customer choice-based models for sales transaction data. *Available at SSRN 3598259*, 2021.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Simon M Diffey, Alison B Smith, AH Welsh, and Brian R Cullis. A new reml (parameter expanded) em algorithm for linear mixed models. *Australian & New Zealand Journal of Statistics*, 59(4):433–448, 2017.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics*, pages 569–593. Springer, 1992.

Mark E Ferguson. Estimating demand with constrained data and product substitutions. In *Channel Strategies and Marketing Mix in a Connected World*, pages 1–27. Springer, 2020.

Mark E Ferguson, Laurie A Garrow, and Jeffrey P Newman. Application of discrete choice models to choice-based revenue management problems: A cautionary note. *Journal of Revenue and Pricing Management*, 11(5):536–547, 2012.

Guillermo Gallego and Garrett Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40(8):999–1020, 1994.

Guillermo Gallego and Garrett Van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, 45(1):24–41, 1997.

Maya R Gupta and Yihua Chen. *Theory and use of the EM algorithm*. Now Publishers Inc, 2011.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Mortaza Jamshidian and Robert I Jennrich. Standard errors for em estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(2):257–270, 2000.

Anthony Owen Lee. Airline reservations forecasting: Probabilistic and statistical models of the booking process. Technical report, Cambridge, Mass.: Flight Transportation Laboratory, Dept. of Aeronautics and Astronautics, 1990.

Misuk Lee. Modeling and forecasting hotel room demand based on advance booking information. *Tourism Management*, 66:62–71, 2018.

Anran Li and Kalyan Talluri. Estimating demand with unobserved no-purchases on revenue-managed data. *Available at SSRN 3525773*, 2020.

Chuanhai Liu and Donald B Rubin. The ecme algorithm: a simple extension of em and ecm with faster monotone convergence. *Biometrika*, 81(4):633–648, 1994.

Thomas J Matarazzo and Shamim N Pakzad. Stride for structural identification using expectation maximization: iterative output-only method for modal identification. *Journal of Engineering Mechanics*, 142(4):04015109, 2016.

Daniel McFadden. Modelling the choice of residential location. *Cowles Foundation Discussion Papers*, 1977.

Geoffrey J McLachlan, Thriyambakam Krishnan, and See Ket Ng. The em algorithm. Technical report, Papers, 2004.

Xiao-Li Meng and Donald B Rubin. Using em to obtain asymptotic variance-covariance matrices: The sem algorithm. *Journal of the American Statistical Association*, 86(416): 899–909, 1991.

Xiao-Li Meng and Donald B Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

Xiao-Li Meng and David Van Dyk. The em algorithm—an old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(3): 511–567, 1997.

Jeffrey P Newman, Mark E Ferguson, Laurie A Garrow, and Timothy L Jacobs. Estimation of choice-based models using sales data from a single firm. *Manufacturing & Service Operations Management*, 16(2):184–197, 2014.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Pelin Pekgün, Ronald P Menich, Suresh Acharya, Phillip G Finch, Frederic Deschamps, Kathleen Mallery, Jim Van Sistine, Kyle Christianson, and James Fuller. Carlson rezidor hotel group maximizes revenue through improved demand management and price optimization. *Interfaces*, 43(1):21–36, 2013.

W Qian and DM Titterington. Stochastic relaxations and em algorithms for markov random fields. *Journal of Statistical Computation and Simulation*, 40(1-2):55–69, 1992.

Richard A Redner and Homer F Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.

Shivaram Subramanian and Pavithra Harsha. Demand modeling in the presence of unobserved lost sales. *Management Science*, 2021.

Kalyan Talluri. A finite-population revenue management model and a risk-ratio procedure for the joint estimation of population size and parameters. *Available at SSRN 1374853*, 2009.

Kalyan Talluri and Garrett Van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Robert J Tibshirani and Bradley Efron. An introduction to the bootstrap. *Monographs on Statistics and Applied Probability*, 57:1–436, 1993.

Gustavo Vulcano, Garrett Van Ryzin, and Wassim Chaar. Om practice—choice-based revenue management: An empirical study of estimation and optimization. *Manufacturing & Service Operations Management*, 12(3):371–392, 2010.

Gustavo Vulcano, Garrett Van Ryzin, and Richard Ratliff. Estimating primary demand for substitutable products from sales transaction data. *Operations Research*, 60(2):313–334, 2012.

Greg CG Wei and Martin A Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

Lloyd R Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):10–13, 2003.

CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of Statistics*, pages 95–103, 1983.

Jiangtao Yin, Yanfeng Zhang, and Lixin Gao. Accelerating expectation-maximization algorithms with frequent updates. In *2012 IEEE International Conference on Cluster Computing*, pages 275–283. IEEE, 2012.

Athanasius Zakhary, Amir F Atiya, Hisham El-Shishiny, and Neamat E Gayar. Forecasting hotel arrivals and occupancy using monte carlo simulation. *Journal of Revenue and Pricing Management*, 10(4):344–366, 2011.

Appendices

A Proof of Concavity of LL

Theorem 2. $LL(p, \tau; \boldsymbol{\beta}) = \sum_{j=1}^{N} \left(\sum_{k \in S_j} \log(\lambda_j P_1) + \sum_{k \in \bar{S}_j} a(k) \log(\lambda_j P_0) + (1 - a(k)) \log(1 - \lambda_j) \right)$ *is concave in* $\boldsymbol{\beta}$.

Proof. First, let's rewrite LL to split up the logarithms and rearrange the sums slightly:

$$\mathrm{LL}(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \sum_{j=1}^{N} \left(\sum_{k \in S_j} \log(\lambda_j) + \log(P_1) + \sum_{k \in \bar{S}_j} a(k) \log(\lambda_j) + \log(P_0) + (1 - a(k)) \log(1 - \lambda_j) \right)$$

Since we are only concerned with concavity in terms of $\boldsymbol{\beta}$, we can ignore all terms that do not involve Q, giving the following expression:

$$\sum_{i=1}^{N} \sum_{k \in S} \log(P_1(p_k, \tau_k; \boldsymbol{\beta})) + \sum_{k \in \bar{S}} \hat{a}(k) \log(P_0(p_k, \tau_k; \boldsymbol{\beta}))$$

Since sums of concave functions are also concave, we simply need to verify the concavity of

$\log(P_1)$ and $\log(P_0)$.

$$\begin{aligned}\log(P_1) &= \log\left(\frac{e^{\boldsymbol{y}^T\boldsymbol{\beta}}}{1+e^{\boldsymbol{y}^T\boldsymbol{\beta}}}\right) \\ &= \log(e^{\boldsymbol{y}^T\boldsymbol{\beta}}) - \log(1+e^{\boldsymbol{y}^T\boldsymbol{\beta}}) \\ &= (\boldsymbol{y}^T\boldsymbol{\beta}) - \log(1+e^{\boldsymbol{y}^T\boldsymbol{\beta}})\end{aligned}$$

Clearly the first term is concave in $\boldsymbol{\beta}$, since it is linear. For the second term, $1 + e^{\boldsymbol{y}^T\boldsymbol{\beta}}$ is convex in $\boldsymbol{\beta}$, since e^x is a convex function. The composition of $\log(x)$ with $1 + e^{\boldsymbol{y}^T\boldsymbol{\beta}}$ is convex as well. This is because $\log(x)$ is non-decreasing concave, so when it is composed with a convex function the result is convex. Since the second term is negative, it is therefore concave. Thus, $\log(P_1)$ is concave in β_i. A very similar argument shows that $\log(P_0)$ is concave. We can conclude that LL is concave in $\boldsymbol{\beta}$. □

B Math Program for Step Selection

Given DBAs $1, \ldots, D$ *and a desired number of steps* N*, let* y_d *be the total number of sales recorded on DBA* d*. Let* x_s *be the average number of purchases for step* s*, and let* z_{ds} *be a binary variable that indicates whether DBA* d *is in step* s*. Finally, let* M *be some sufficiently large integer.*

The first two constraints ensure that $v_d = x_s$ *only when* d *is a member of step* s*. The next three constraints ensure that each DBA is a member of exactly one step, the first and last DBAs are members of the first and last steps, and finally that steps must contain consecutive DBAs.*

$$\underset{z}{\text{minimize}} \ \sum_{d=1}^{D}(y_d - v_d)^2$$

subject to

$$x_s - M(1 - z_{ds}) \leq v_d \ \ \forall d = 1 \ldots D, \ \forall s = 1 \ldots N$$

$$x_s + M(1 - z_{ds}) \geq v_d \ \ \forall d = 1 \ldots D, \ \forall s = 1 \ldots N$$

$$\sum_{s=1}^{m} z_{ds} = 1 \ \ \forall d = 1 \ldots D$$

$$z_{11} = 1, \ z_{DS} = 1$$

$$z_{ds} + z_{et} \leq 1 \ \ \forall s > t, \ \forall e > d$$

$$x_s \geq 0 \ \ \forall s, \ z_{ds} \in \{0, 1\} \ \ \forall d, \ \forall s, \ v_d \in \mathbb{R} \ \ \forall d$$

C SVMH Pseudocode

Input: *Set D of DBAs in descending order; set $\{y_d : d \in D\}$: total number of purchases made on DBA d across all observations k in the dataset; desired number of steps N; initial step configuration $S^0 = \{s_1, s_2, \ldots, s_N\}$, parameterized by boundaries b_i such that for all $d \in s_i$, $b_{i-1} > d \geq b_i$*

Output: *N steps, where a step is a set of DBAs such that the DBAs are consecutive.*

Let $V(S) : S \to \mathbb{R}$ be the scoring function, where $V(S) = \sum_{i=1}^{N}$ Variance$\{y_d : d \in s_i\}$

Let C^m represent the list of all candidate step configuration from iteration m

Let $P(N)$ be all consecutive pairs of integers, $(1,2),(2,3),\ldots,(N-1,N)$

Set $S_{min} = S^0$, the current configuration.

Set counter $m = 0$

while *True* **do**

  for (i,j) *in* $P(N)$ **do**

    F_{ij} *= ForwardTranslate(S_{min}, i, j)*

    B_{ij} *= BackwardTranslate(S_{min}, i, j)*

    C^m *+=*F_{ij}, B_{ij}

  end

  J_{ij} *= Jump(S_{min})*

  C^m *+=* J_{ij}

  Let C_{min} be the configuration in C^m such that $V(C_{min}) < V(C_i) \forall C_i \in C^m$

  if $V(C_{min}) < V(S_{min})$ **then**

    $S_{min} = C_{min}$

    m+=1

  end

  else

    Return S_{min}

  end

end

Algorithm 1: Step Variance Minimization Heuristic

The subordinate routines, Translate and Jump are described in Section 3.2. Given a pair of consecutive steps s_i, s_j, a forward translation refers to moving the boundary between the steps, b_i forward into s_j (so part of s_j is merged with s_i. A backward move is the reverse, moving b_i backwards into s_i.

D Proof that REM is a Generalized Expectation Maximization Algorithm

To place REM in a more general context, we now formulate REM in more general notation. In a general EM problem, we have observed data y, complete data x, and parameters $\theta \in \Omega$. Both x and y are drawn from random variables X and Y, which have densities $p(x|\theta)$ and $p(y|\theta)$ respectively. The goal is to find θ^* such that:

$$\theta^* = \arg\max_{\theta \in \Omega} p(y|\theta).$$

At the m^{th} iteration of EM, we first find $Q(\theta|\theta^{(m)}) = E\{\log p(x|\theta)|y, \theta^{(m)}\}$, and then maximize Q to obtain

$$\theta^{(m+1)} = \arg\max_{\theta \in \Omega} Q(\theta|\theta^{(m)}).$$

In REM, we alter this procedure by first separating θ into two subsets, θ_c and θ_i, where θ_c consists of those parameters that can be solved via a closed form solution, and θ_i consists of those parameters that must be optimized iteratively in the M-step. Clearly, $\theta = \theta_c \cup \theta_i$, and it may be the case that one or other may be empty. Further, it must be the case that $Q(\theta|\theta^{(m)}) = Q_c(\theta_c|\theta^{(m)}) + Q_i(\theta_i|\theta^{(m)})$ where Q_c (Q_i) is the restriction of Q to θ_c (θ_i). In other words, Q is separable in θ_c and θ_i so that the two parameter sets can be solved for

separately.

Given these requirements, the M-step can now be written as:

$$\theta^{(m+1)} = \theta_c^{(m+1)} \cup \theta_i^{(m+1)} = \arg\max_{\theta_c \in \Omega} Q_c(\theta_c|\theta^{(m)}) + \arg\max_{\theta_i \in \Omega} Q_i(\theta_i|\theta^{(m)}).$$

In REM, θ_c is solved for normally. However, for θ_i, $Q_i(\theta_i|\theta^{(m)})$ is replaced by

$$\widetilde{Q_i}(\theta_i|\theta^{(m)}, w) = -w||\theta_i - \theta_i^{(m)}|| + Q_i(\theta_i|\theta^{(m)})$$

for $w > 0$ and an appropriate norm $||\cdot||$. REM is thus defined as the following steps at the m^{th} iteration:

- *E-step: Calculate $Q(\theta|\theta^{(m)}, y)$ for observed data y and parameter guess $\theta^{(m)}$*

$$Q(\theta|\theta^{(m)}, y) = Q_c(\theta_c|\theta^{(m)}, y) + Q_i(\theta_i|\theta^{(m)}, y)$$

- *M-step: Maximize $Q(\theta|\theta^{(m)}, y)$ with respect to θ, given weight w.*

$$\theta^{(m+1)} = \theta_c^{(m+1)} \cup \theta_i^{(m+1)} = \arg\max_{\theta_c \in \Omega} Q_c(\theta_c|\theta^{(m)}, y) + \arg\max_{\theta_i \in \Omega} \widetilde{Q_i}(\theta_i|\theta^{(m)}, y, w)$$

Proof that REM is a GEM *The only theoretical guarantee about EM convergence is that the value of the incomplete log-likelihood function will increase with every iteration. Formally:*

$$L(\theta^{(m+1)}) \geq L(\theta^{(m)}). \tag{D.1}$$

Dempster et al. (1977) show that (D.1) *holds so long as both*

$$Q(\theta^{(m+1)}|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)}) \tag{D.2}$$

and

$$H(\theta^{(m+1)}|\theta^{(m)}) \leq H(\theta^{(m)}|\theta^{(m)}) \tag{D.3}$$

both hold. In EM, (D.3) *holds by definition of H, and since REM does not change the construction of either Q or H (just the optimization of Q), it will still hold for REM. Thus, it remains to show* (D.2)*. DLR define an algorithm such that* (D.2) *holds in the M-step as a Generalized Expectation Maximization(GEM) algorithm. A first step for many EM variants is to show that the GEM property holds. We now show that REM satisfies* (D.2)*, and thus* (D.1) *holds.*

Theorem. *In an arbitrary iteration m of the REM algorithm, for the mth parameter set $\theta^{(m)}$ and $m+1$th parameter set $\theta^{(m+1)}$:*

$$Q(\theta^{(m+1)}|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)})$$

Proof. In REM, we obtain

$$Q(\theta|\theta^{(m)}) = Q_c(\theta_c|\theta^{(m)}) + Q_i(\theta_i|\theta^{(m)})$$

in the E-step. Thus, it is sufficient to show that

1. $Q_c(\theta_c^{(m+1)}|\theta^{(m)}) \geq Q_c(\theta_c^{(m)}|\theta^{(m)})$
2. $Q_i(\theta_i^{(m+1)}|\theta^{(m)}) \geq Q_i(\theta_i^{(m)}|\theta^{(m)})$

It is clear that (i) is true, since $\theta_c^{(m+1)} = \arg\max_{\theta_c \in \Omega} Q(\theta_c|\theta^{(m)})$.

For (ii), certainly $\widetilde{Q_i}(\theta_i^{(m+1)}|\theta^{(m)}) \geq \widetilde{Q_i}(\theta_i^{(m)}|\theta^{(m)})$, again by the definition of $\theta^{(m+1)}$. Then, we have

$$\begin{aligned}
\widetilde{Q_i}(\theta_i^{(m+1)}|\theta^{(m)}) &\geq \widetilde{Q_i}(\theta_i^{(m)}|\theta^{(m)}) \\
&= -w||\theta_i^{(m)} - \theta_i^{(m)}|| + Q_i(\theta_i^{(m)}|\theta^{(m)}) \\
&= Q_i(\theta_i^{(m)}|\theta^{(m)}).
\end{aligned}$$

Thus, $\widetilde{Q_i}(\theta_i^{(m+1)}|\theta^{(m)}) \geq Q_i(\theta^{(m)}|\theta^{(m)})$. Finally, since $w > 0$ and $||\cdot|| \geq 0$, since it is a norm, $-w||\theta - \theta^{(m)}|| < 0$ for all $\theta \in \Omega$. Therefore,

$$\begin{aligned}
Q_i(\theta^{(m+1)}|\theta^{(m)}) &\geq -w||\theta_i^{(m+1)} - \theta_i^{(m)}|| + Q_i(\theta_i^{(m+1)}|\theta^{(m)}) \\
&= \widetilde{Q_i}(\theta_i^{(m+1)}|\theta^{(m)}) \\
&\geq Q_i(\theta_i^{(m)}|\theta^{(m)})
\end{aligned}$$

Thus, we conclude that (ii) holds, and as a result $Q(\theta^{(m+1)}|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)})$ □

E Additional Results and Tables

Method	# Steps	Simulated Data			Real Data	Average
		Realistic	Linear	NoTrend	High Season	
Const	**1**	82%	85%	97%	52%	79%
Even	**2**	67%	97%	96%	41%	75%
Even	**3**	51%	68%	97%	27%	61%
SVMH	**2**	80%	97%	98%	97%	93%
SVMH	**3**	90%	98%	98%	97%	96%
Average		72%	90%	97%	66%	**81%**

Table E.1: Average Percentage Decrease in Convergence Time for REM Models Compared to EM on the Same Data

Dataset	Parameter Name			
	Arrival Rate (DBAs >= #)	β **Intercept**	β **Price**	β **DBA**
Sim1	0: 0.5	2	-0.022	-0.01
Sim2	15: 0.25, 0: 0.75	2	-0.022	-0.01
Sim3	20: 0.1, 5: 0.4, 0: 0.8	2	-0.022	-0.09

Table E.2: True Parameters Used to Generate Simulated Data