Title

Hierarchical Modeling and Shrinkage for User Session LengthPrediction in Media Streaming

Authors

Dedieu, Antoine

Mazumder, Rahul

Zhu, Zhen

et al.

Publication Date

2018-10-17

DOI

10.1145/3269206.3271700

Hierarchical Modeling and Shrinkage for User Session Length Prediction in Media Streaming

Article · March 2018

4 authors, including:

Antoine Dedieu

23 PUBLICATIONS **137** CITATIONS

SEE PROFILE

Hierarchical Modeling and Shrinkage for User Session Length Prediction in Media Streaming

Antoine Dedieu*[1], Rahul Mazumder†[1], Zhen Zhu‡[2,3], and Hossein Vahabi§[2]

[1]Massachusetts Institute of Technology
[2]Pandora Media, Inc.
[3]Stanford

June 26, 2018

Abstract

An important metric of users' satisfaction and engagement within on-line streaming services is the *user session length*, i.e. the amount of time they spend on a service continuously without interruption. Being able to predict this value directly benefits the recommendation and ad pacing contexts in music and video streaming services. Recent research has shown that predicting the exact amount of time spent is highly nontrivial due to many external factors for which a user can end a session, and the lack of predictive covariates. Most of the other related literature on duration based user engagement has focused on dwell time for websites, for search and display ads, mainly for post-click satisfaction prediction or ad ranking.

In this work we present a novel framework inspired by hierarchical Bayesian modeling to predict, at the moment of login, the amount of time a user will spend in the streaming service. The time spent by a user on a platform depends upon user-specific latent variables which are learned via hierarchical shrinkage. Our framework enjoys theoretical guarantees and naturally incorporates flexible parametric/nonparametric models on the covariates, including models robust to outliers. Our proposal is found to outperform state-of- the-art estimators in terms of efficiency and predictive performance on real world public and private datasets.

1 Introduction

On-line streaming services such as Pandora, Netflix, and Youtube constantly seek to increase their share of on-line attention by keeping their users as engaged as possible with their own services [13]. A well known challenge is how to measure the users' engagement, and what are the key components that create an engaging streaming service. One important engagement metric is the amount of time spent by users within the service. When users access streaming services, they usually watch videos, movies, on-line TV or listen to music, and after a while they leave the service. We refer to the user interaction from the moment they start the service to the moment they leave as a *user session*, and the time spent during one session as *user session length* [20].

*adedieu@mit.edu
†rahulmaz@mit.edu
‡zzhu@pandora.com
§puya@pandora.com

In this paper, we aim to predict the user session length using real world datasets from two music streaming platforms, i.e. predicting, at the beginning of the session, the amount of time they will spend listening music. Understanding and modeling the factors that can affect the session length is of great use for various downstream tasks. In fact it allows recommender systems to tune the explore vs exploit parameters for each user. In addition, having an accurate estimate of the users' session lengths allows the streaming service to adjust ad pacing per user. Ads can be rescheduled in a way to keep the revenue target (i.e. total number of ads presented) as well as improve user experience.

Predicting the length of user sessions is very challenging as the authors reported in [20]. First, user sessions can end for many different external reasons that have nothing to do with quality of the streaming services, such as moving to subway, reaching home, and most of these contextual covariates are not easily accessible because of technological or privacy reasons [20]. Second, a certain amount of users are casual users in the sense that they only use the streaming services a few times per month, which makes the problem of estimating session lengths for those users hard. Furthermore, a lack of predictive covariates makes it harder to correctly predict what is going to be the next session length for a user.

A first approach toward session length analysis and prediction [20] is based on a Boosting algorithm. Most of the other related research were focused on modeling the time spent after clicking a search result [3, 12] or advertisement [1]. The best models are based on survival analysis, mainly because data are censored in these applications. In fact, after clicking of a search result or an ad, users can either turn back to the search page or can abandon the final page, instead of turning back to the main search page. This is not the case of session length data that is not censored[1] – this opens the door to using a suite of methods based on regression modeling, shrinkage and relevant generalizations. Furthermore, in the case of web search or ad click, the user enters a query or click and checks the results, therefore the intersection between search or ad title and the landing page give highly predictive features [12]. In the case of a streaming service, the user interaction can be very low, but still they can have very long sessions ("lean-back" behavior) [20].

In this paper, we propose a novel framework inspired by hierarchical Bayesian modeling and shrinkage principles that allows us to express session lengths in terms of user-specific latent variables. These variables are estimated via a joint learning framework which is rather broad in scope – we use Bayes, Empirical Bayes and MAP estimation techniques–particular choices are based on computational tractability considerations. Informally speaking, our models learn by borrowing strengths across all users and making use of rich covariate information. We also propose models that can incorporate outliers in data. A salient aspect of our framework is its modularity – it includes state-of-the-art models as special cases, and naturally allows for a hierarchy of flexible generalizations. Hence, it allows the practitioner to glean insights about the problem, by assessing incremental gains in predictive accuracy associated with different generalizations and the incorporation of covariate-information. We present tailored algorithmic approaches based on modern large convex optimization techniques to address the computational challenges. We summarize some of our key findings in this paper:

- We outperform a baseline estimator by a margin of 13% to 19% and the state-of-the art in session length prediction [20] up to 4.3% in terms of *Mean Absolute Error* measured in seconds on 2 different real world datasets.
- We show that some of our proposed prediction models can be (a) more accurate than state of the art (with 1-2% relative improvement in prediction performance), (b) between 22 to

[1]Observations are called censored when the information about their survival time is incomplete.

43 times faster in training time; and (c) 30-500 times faster in prediction time, reaching around 1ms.

- We provide a modular framework specifically for this problem that allow more flexible generalizations,

2 Key idea

The main focus of this paper is the prediction of user session length at the moment of login into a streaming service. For this purpose, we exploit the users past interaction with streaming services, previous sessions lengths, build a set of features (as we will describe in Section 4.1), and we set a learning framework for prediction purposes. We proceed by using a Gaussian approximation for the distribution of the log of a users' session length (Section 4.1). As we will see, this will help us in creating a well-grounded and tractable inferential framework. Since we want a prediction framework that is performing well for very active users (with many sessions) and less-frequent users (with only a few sessions per month), we propose a formal framework inspired by hierarchical Bayesian shrinkage ideas that allows us to model the session-length of a user in terms of user-specific latent variables. Fundamental principles of Bayesian shrinkage and estimation encourage users to borrow strengths across each other, including (but not limited to) covariate information. This (i) ameliorates the high variance (and hence low predictive accuracy) of user-specific maximum likelihood estimates for less-frequent users; and (ii) leads to an overall boost in prediction accuracy for more frequent users as well. Bayesian decision theory and empirical Bayes methodology [6] provides a formal justification of our framework. The notion of shrinkage that we undertake here is quite broad: It applies to cases with or without covariate information; by a flexible choice of priors on the latent variables, we can also build models robust to heavy tailed errors. We show that the state-of-the-art model [20] on this particular problem is a special case of our framework. For flexible models and/or priors when Bayes estimators become computationally demanding, we recommend MAP estimation for computational efficiency, this includes the case for robust modeling with a Huber loss [9]. We resort to techniques in modern large scale convex optimization to achieve computational scalability and efficiency.

3 Mathematical Framework

We formally develop the inferential framework to address the session length prediction problem. We denote the total number of users by N. For every $i \in [N] := \{1, \ldots, N\}$, let n_i be the number of past sessions of user i, and $\tilde{y}_{ij}$ be the time spent by user i in the jth session. We will work with log of session length $y_{ij} = \log(\tilde{y}_{ij})$ as our response as this gives a better approximation to the Gaussian distribution (See Table 1). We note that similar (variance stabilizing) transformations[2] are often used in the empirical Bayes literature [6] so that one can take recourse to the rich literature in Gaussian estimation theory. We denote by $\boldsymbol{y}_i = (y_{ij})_{j \in [n_i]} \in \mathbb{R}^{n_i}$ a vector of log-session-lengths of user i; $N_0 = \sum_{i=1}^{N} n_i$ the total number of sessions across all users; and $\boldsymbol{y} = (\boldsymbol{y}_i)_{i \in [N]} \in \mathbb{R}^{N_0}$ is a vector of log-session-lengths of all users across all sessions. In addition, covariate information

[2]We note that it is also possible to fine-tune the transformation by considering a family of the form: $\log(\cdot)^\tau$ for $\tau > 0$; and optimizing over τ to obtain the closest approximation to a Gaussian distribution in terms of the Kolmogorov-Smirnov goodness of fit measure (for example). However, we do not pursue this approach here; as it leads to marginal gains in eventual prediction performance.

per-session is available (See Section 4.1). For a given user, some of these features are fixed across sessions (age, gender...) and others depend upon the session (network, device...). Let $\boldsymbol{x}_{ij} \in \mathbb{R}^d$ denote covariates corresponding to the jth session of user i. $\boldsymbol{X} \in \mathbb{R}^{N_0 \times d}$ denotes a matrix with rows $\boldsymbol{x}_{ij}$ that are stacked on top of each other. The kth row of $\boldsymbol{X}$ corresponds to kth entry of the response vector $\boldsymbol{y}$. The columns of $\boldsymbol{X}$ were all mean-centered and standardized to have unit ℓ_2 norm.

A natural point estimate of the time spent by the ith user may be taken to be the average time spent by the user in past sessions – however, we see that this does not lead to good predictions due to the high variance associated with users with few sessions. This behavior is not surprising and occurs even in the simple Gaussian sequence model as explained in Section 3.1.

3.1 Review of Bayes, Empirical Bayes and MAP

This section provides a brief review of Bayes, Empirical Bayes (EB) and Maximum A posteriori (MAP) estimation in the context of the Gaussian sequence model [6]. The exposition in this section lays the foundation for generalizations to more flexible structural models that we present subsequently.

The Bayes Estimator: We consider a latent Gaussian vector $\boldsymbol{\mu}_{n\times 1} = (\mu_1, \ldots, \mu_n)$ where, $\mu_i \overset{\text{iid}}{\sim} \mathcal{N}(0, A^2)$; that gives rise to an observable Gaussian vector $\boldsymbol{z} = (z_1, \ldots, z_n)$ such that $z_i|\mu_i \sim \mathcal{N}(\mu_i, 1)$ for all i. Note that the posterior distribution of $\boldsymbol{\mu}|\boldsymbol{z}$ is given by a n-dimensional multivariate Gaussian with mean $B^2\boldsymbol{z}$ and covariance $B^2\boldsymbol{I}$, i.e., $\boldsymbol{\mu}|\boldsymbol{z} \sim \mathcal{N}_n(B^2\boldsymbol{z}, B^2\boldsymbol{I})$ where, $B^2 = \frac{A^2}{1+A^2}$. Recall that the Bayes estimator is the mean of the posterior $\boldsymbol{\mu}|\boldsymbol{z}$ and given by:

$$\hat{\boldsymbol{\mu}}^{\text{Bayes}} = \mathbb{E}(\boldsymbol{\mu}|\boldsymbol{z}) = \left(1 - 1/(1+A^2)\right)\boldsymbol{z}. \tag{1}$$

We remind the reader that the Bayes estimator shrinks each observation towards the mean 0 of the prior distribution – this is to be contrasted with the usual maximum likelihood (ML) estimator, $\hat{\boldsymbol{\mu}}^{\text{ML}} = \boldsymbol{z}$ that does not shrink μ_i's. The Bayes estimator has smaller risk than the ML estimator (Theorem 1).

Empirical Bayes (EB) estimator: The Bayes estimator depends upon the unknown hyper-parameter A^2; which needs to be estimated from data. The "Empirical Bayes" (EB) framework [6] achieves this goal by using a data-driven plug-in estimator for A^2 in (1) – this leads to an EB estimator for $\boldsymbol{\mu}$.

The basic EB framework obtains an unbiased estimator for A^2 based on the marginal distribution of $\boldsymbol{z} \sim \mathcal{N}_n\left(\boldsymbol{0}, \left(1+A^2\right)\boldsymbol{I}\right)$. Using standard properties of Gamma and inverse-Gamma distributions, it follows that $(n-2)/S$ (where, $S = \sum_{i=1}^n z_i^2$) is an unbiased estimator of $\frac{1}{A^2+1}$. This leads to an EB estimate $\hat{\boldsymbol{\mu}}^{\text{EB}} = \left(1 - \frac{n-2}{S}\right)\boldsymbol{z}$, which has smaller risk (Theorem 1) than the ML estimator.

Theorem 1 *[6] For $n \geq 3$, the EB estimator $\hat{\boldsymbol{\mu}}^{EB}$ has smaller risk (defined as $R(\hat{\boldsymbol{\mu}}) := \mathbb{E}(\|\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\|_2^2)$) than the ML estimator $\hat{\boldsymbol{\mu}}^{ML}$, i.e., $R(\hat{\mu}^{EB}) < R(\hat{\boldsymbol{\mu}}^{ML})$ for any $\boldsymbol{\mu}$. The risks of the EB and Bayes estimators are comparable, with a relative ratio: $\frac{R(\hat{\boldsymbol{\mu}}^{EB}) - R(\hat{\boldsymbol{\mu}}^{Bayes})}{R(\hat{\boldsymbol{\mu}}^{Bayes})} = \frac{2}{nA^2}$.*

We make the following remarks: (i) Theorem 1 states that the price to pay for not knowing A is rather small, and as n becomes large the Bayes and EB estimators are similar. (ii) Instead of taking

an unbiased estimator for $1/(A^2+1)$ as above, one can also take a consistent estimator which might be easier to obtain for more general models (see Section 3.2). For more general models, a good plug-in estimate for A may be obtained based on validation tuning. The framework above provides important guidance regarding a range of good choices of A thereby reducing the computational cost associated with the search for tuning parameters.

MAP estimation: As an alternative to the Bayes/EB estimator, we can also consider the MAP estimator, a mode of the posterior likelihood $\boldsymbol{\mu}|\boldsymbol{z}$. Here the MAP estimate ($\hat{\boldsymbol{\mu}}^{\text{MAP}}$) coincides with the Bayes estimator. The MAP and Bayes estimators are not the same in general. For flexible priors/models, computing Bayes estimators can become computationally challenging and one may need to resort to (intractable) high dimensional MCMC computations. In these situations (see Section 3.3), the MAP estimator may be easier to compute from a practical viewpoint. For all models used in this paper, we observe that MAP computation can be tractably performed via convex optimization.

3.2 Model 1: Modeling user effects

We present a hierarchical shrinkage framework for predicting user-specific session lengths, generalizing the framework in Section 3.1.

Suppose that the log-session lengths of the ith user are normally distributed with mean μ_i; and these latent variables $\{\mu_i\}_1^N$ are generated from a centered Gaussian distribution. The (random) μ_i is the ith user effect. This leads to the following hierarchical model:

$$y_{ij}|\mu_i \overset{\text{iid}}{\sim} \mathcal{N}\left(\mu_i, \sigma_1^2\right), i \in [N],\ j \in [n_i]; \quad \mu_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_0^2) \tag{2}$$

generalizing the model in Section 3.1 to the case with multiple replications per user. The posterior distribution is given by:

$$\mu_i|\boldsymbol{y}_i \sim \mathcal{N}\left(\frac{\bar{y}_i}{1+\lambda/n_i}, \frac{\sigma_1^2}{\lambda+n_i}\right)\ \forall i,$$

where, $\lambda = \sigma_1^2/\sigma_0^2$ and $\bar{y}_i = \sum_{j=1}^{n_i} y_{ij}/n_i$ is the mean of the vector $\boldsymbol{y}_i$. The Bayes estimator of $\boldsymbol{\mu}$ is given by the posterior mean $\hat{\mu}_i^{\text{Bayes}} = \frac{\bar{y}_i}{1+\lambda/n_i}$. Here, the MAP estimator of $\boldsymbol{\mu}$ coincides with the Bayes estimator as well. We note that the Bayes/MAP estimators in this example bear similarities with the model in Section 3.1 – we shrink the observed mean of each user towards the global mean of the prior distribution: this lowers the variance of the estimator at the cost of (marginally) increasing the bias. The amount of shrinkage depends upon the number of sessions of the ith user via the factor $1+\lambda/n_i$. In particular, the shrinkage effect will be larger for users with a small number of sessions.

Estimating the hyper-parameters: The estimators above depend upon hyper-parameters σ_0, σ_1 via $\lambda = \sigma_1^2/\sigma_0^2$, which is unknown and needs to be estimated from data. In the spirit of an EB estimator we obtain a plug-in estimator for λ. To this end we use the marginal distribution of $\boldsymbol{y}_i$, which follows $\mathcal{N}_{n_i}(\boldsymbol{0}, \boldsymbol{\Sigma}_{n_i})$ where, $\boldsymbol{\Sigma}_{n_i}$ has diagonal entries equal to $\sigma_0^2+\sigma_1^2$ and off-diagonal entries equal to σ_0^2. Consequently, $\boldsymbol{y}_i\boldsymbol{y}_i^T$ is an unbiased estimator for the covariance matrix $\boldsymbol{\Sigma}_{n_i}$. In particular, if $T_i = \|\boldsymbol{y}_i\|_2^2$ then the estimators

$$\hat{\sigma}_0^2(i) = \frac{(n_i\bar{\boldsymbol{y}}_i)^2 - T_i}{n_i(n_i-1)} \quad \text{and} \quad \hat{\sigma}_0^2(i) + \hat{\sigma}_1^2(i) = \frac{T_i}{n_i} \tag{3}$$

are unbiased estimators of σ_0^2 and $\sigma_0^2 + \sigma_1^2$ (respectively). To see this, note that $\hat{\sigma}_0^2(i)$ is obtained by taking the average of all the $n_i(n_i - 1)$ off-diagonal entries of the matrix $\boldsymbol{y}_i \boldsymbol{y}_i^T$. Similarly, $\hat{\sigma}_0^2(i) + \hat{\sigma}_1^2(i)$ corresponds to the average of the diagonal entries. Estimators in (3) are based solely on observations from the ith user; and can have high variance if n_i is small (which is the case for less heavy users). Hence, we aggregate the estimators across all N users to obtain improved estimators of σ_0^2, σ_1^2 given by:

$$\begin{aligned} \hat{\sigma}_0^2 &= \tfrac{1}{N} \sum_{i=1}^{N} \hat{\sigma}_0^2(i) = \tfrac{1}{N} \sum_{i=1}^{N} \tfrac{(\mathbf{1}_{n_i}^T \boldsymbol{y}_i)^2 - T_i}{n_i(n_i - 1)} \\ \hat{\sigma}_1^2 &= \tfrac{1}{N} \sum_{i=1}^{N} \hat{\sigma}_1^2(i) = \tfrac{1}{N} \sum_{i=1}^{N} \tfrac{T_i}{n_i} - \hat{\sigma}_0^2. \end{aligned} \tag{4}$$

Using laws of large numbers, one can verify that $\hat{\sigma}_0^2$ (and $\hat{\sigma}_1^2$) are consistent estimators for σ_0^2 (and σ_1^2). Interestingly, this holds under an asymptotic regime where, $N \to \infty$ but $\min_i n_i$ remains bounded – this regime is relevant for our problem since there are many users with few/moderate number of sessions. We emphasize that even if N is large but the n_i's are small, shrinkage plays an important role and leads to estimators with smaller risk than the usual maximum likelihood estimator $\mu_i^{\mathrm{ML}} = \bar{\boldsymbol{y}}_i$ for $i \in [N]$. The plug-in estimators suggested above lead to consistent estimators for the Bayes and EB estimators. This framework provides guidance regarding the choice of the tuning parameters in practice (and reduces the search-space associated with hyperparameter tuning).

3.3 Model 2: Modeling with covariates

We describe a generalization of Model 1 that incorporates user and device-specific covariates (See Section 4.1 for details). Our hierarchical model is now given by:

$$\begin{aligned} y_{ij} | \boldsymbol{\beta}, \mu_i &\overset{\text{iid}}{\sim} \mathcal{N}\left(\boldsymbol{x}_{ij}^T \boldsymbol{\beta} + \mu_i, \sigma_1^2\right), i \in [N], j \in [n_i] \\ \text{where, } \ \boldsymbol{\beta} &\sim \mathcal{N}_d(\mathbf{0}, \sigma_2^2 \boldsymbol{I}), \quad \mu_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_0^2), i \in [N]. \end{aligned} \tag{5}$$

The above represents a generative model with latent variables $(\boldsymbol{\mu}, \boldsymbol{\beta})$ – where, $\boldsymbol{\beta} \in \mathbb{R}^d$ denotes a vector of regression coefficients corresponding to the covariates $\boldsymbol{X}$; and μ_i explains the residual user-specific effect of user i. Both the latent variables are normally distributed with mean zero; and given these parameters, y_{ij}'s are normally distributed with mean $\boldsymbol{x}_{ij}^T \boldsymbol{\beta} + \mu_i$. This model is more structured and also more flexible than Model 1 in that the ith user effect has two components: a global regression-based response $\boldsymbol{x}_{ij}^T \boldsymbol{\beta}$ (this depends upon both the user and the session); and a residual component μ_i. We now derive the EB and MAP estimators.

Let us define $\tilde{\boldsymbol{\mu}} = \frac{\sigma_2}{\sigma_0} \boldsymbol{\mu}$ and the latent vector $\boldsymbol{\gamma} = (\boldsymbol{\beta}, \tilde{\boldsymbol{\mu}})$. Model (5) can be reformulated as:

$$y_{ij} | \boldsymbol{\gamma} \overset{\text{iid}}{\sim} \mathcal{N}\left(\tilde{\boldsymbol{x}}_{ij}^T \boldsymbol{\gamma}, \sigma_1^2\right), \forall i, j; \qquad \boldsymbol{\gamma} \sim \mathcal{N}_{N+d}\left(\mathbf{0}, \sigma_2^2 \boldsymbol{I}\right),$$

where, $\tilde{\boldsymbol{x}}_{ij} \in \mathbb{R}^{d+N}$ is such that its first d entries correspond to $\boldsymbol{x}_{ij}$, its $(d+i)$th entry is σ_0/σ_2; and all remaining entries are 0. If $\tilde{\boldsymbol{X}}_{N_0 \times d+N}$ be the matrix obtained by row concatenation of the $\tilde{\boldsymbol{x}}_{ij}$'s; then the posterior distribution of $\boldsymbol{\gamma} | \boldsymbol{y}$ is given by

$$\boldsymbol{\gamma} | \boldsymbol{y} \sim \mathcal{N}_{N+d}\left(\boldsymbol{H}^{-1} \tilde{\boldsymbol{X}}^T \boldsymbol{y}, \sigma_2^2 \boldsymbol{H}^{-1}\right),$$

where, the matrix $\boldsymbol{H} = \tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}} + \alpha\boldsymbol{I}$ and the regularization parameter $\alpha = \sigma_1^2/\sigma_2^2$. The Bayes estimate of $\boldsymbol{\gamma}$ is given as:

$$\hat{\boldsymbol{\gamma}}^{\text{Bayes}} = \mathbb{E}(\boldsymbol{\gamma}|\boldsymbol{y}) = \left(\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}} + \alpha\boldsymbol{I}_{d+N}\right)^{-1}\tilde{\boldsymbol{X}}^T\boldsymbol{y} \in \mathbb{R}^{d+N}.$$

$\hat{\boldsymbol{\beta}}^{\text{Bayes}}$ and $\hat{\boldsymbol{\mu}}^{\text{Bayes}}$ can be derived from the components of $\hat{\boldsymbol{\gamma}}^{\text{Bayes}} = \left(\hat{\boldsymbol{\beta}}^{\text{Bayes}}, \frac{\sigma_2}{\sigma_0}\hat{\boldsymbol{\mu}}^{\text{Bayes}}\right)$. In this model, the MAP estimator coincides with the Bayes estimator, and can be computed as $(\hat{\boldsymbol{\beta}}^{\text{MAP}}, \hat{\boldsymbol{\mu}}^{\text{MAP}}) \in \operatorname{argmin}\mathcal{L}_2(\boldsymbol{\beta}, \boldsymbol{\mu})$, where, $\mathcal{L}_2(\boldsymbol{\beta}, \boldsymbol{\mu})$ is the convex function:

$$\mathcal{L}_2(\boldsymbol{\beta}, \boldsymbol{\mu}) := \sum_{i=1}^{N}\left\{\sum_{j=1}^{n_i}(y_{ij} - \boldsymbol{x_{ij}}^T\boldsymbol{\beta} - \mu_i)^2 + \lambda\mu_i^2\right\} + \alpha\|\boldsymbol{\beta}\|_2^2, \tag{6}$$

and $\lambda = \sigma_1^2/\sigma_0^2$, $\alpha = \sigma_1^2/\sigma_2^2$ are hyper-parameters. In Section 3.4, we propose Algorithm 1 to minimize Problem (6).

An empirical Bayes estimator of $(\boldsymbol{\beta}, \boldsymbol{\mu})$ can be computed by using data-driven estimators for the hyper-parameters. We can obtain consistent estimators of the hyper-parameters following the derivation in Section 3.2. Since this derivation [3] is quite tedious, we do not report it here. In practice, we recommend tuning (λ, α) on a validation set (where, λ is taken to be in the neighborhood of the values suggested by Section 3.2 pertaining to Model 1). As we discuss in Section 3.5, this does not add significantly to the overall computational cost, as our algorithm effectively uses warm-start continuation [8] across different tuning parameter choices.

We now move beyond the Gaussian prior setup considered so far and consider a Laplace prior on $\boldsymbol{\beta}$. In this case, and the models we consider subsequently, Bayes estimators are difficult to compute due to high-dimensional integration that require MCMC computations. With computational tractability in mind, we will resort to MAP estimation for these models.

Laplace prior on $\boldsymbol{\beta}$: Motivated by ℓ_1 regularization techniques [19] popularly used in sparse modeling, we propose a Laplace prior on $\boldsymbol{\beta}$ – the corresponding MAP estimators lead to sparse, interpretable models [8, 19]. Here, computing the Bayes estimator becomes challenging and requires MCMC computation. However, the MAP estimator is particularly appealing from a statistical and computational viewpoint; and given by $\left(\hat{\boldsymbol{\beta}}^{\text{MAP}}, \hat{\boldsymbol{\mu}}^{\text{MAP}}\right) \in \operatorname{argmin}\mathcal{L}_1(\boldsymbol{\beta}, \boldsymbol{\mu})$, where,

$$\mathcal{L}_1(\boldsymbol{\beta}, \boldsymbol{\mu}) := \sum_{i=1}^{N}\left\{\sum_{j=1}^{n_i}(y_{ij} - \boldsymbol{x_{ij}}^T\boldsymbol{\beta} - \mu_i)^2 + \lambda\mu_i^2\right\} + \alpha\|\boldsymbol{\beta}\|_1 \tag{7}$$

is a convex function with hyper-parameters λ, α. The tuning parameters are chosen based on a validation set. Section 3.4 presents an algorithmic framework based on first order convex optimization methods [16, 21] for optimizing Problem (7) – our proposed algorithm leads to significant computational gains compared to off-the-shelf implementations.

[3] Note that for Model (5) the marginal distribution of $\boldsymbol{y}$ is a multivariate Gaussian with mean zero and covariance matrix $\boldsymbol{\Sigma}$, which is a function of $\{\sigma_i\}_0^2$ and $\boldsymbol{X}\boldsymbol{X}^T$. Following Section 3.2, we have $\mathbb{E}(\boldsymbol{y}\boldsymbol{y}^T) = \boldsymbol{\Sigma}$. We can then derive consistent estimators of $\{\sigma_i\}_0^2$ based on functionals of $\boldsymbol{y}\boldsymbol{y}^T$ and entries of $\boldsymbol{X}\boldsymbol{X}^T$.

3.3.1 Nonparametric modeling with covariates

The framework presented above is quite modular–it allows for flexible generalizations, allowing a practitioner to experiment with several modeling ramifications, and understand their incremental value (prediction accuracy vis-a-vis computation time) in the context of the particular application/dataset.

Recall that the basic model put forth by Model 2 is $y_{ij}|\theta_{ij} \overset{\text{iid}}{\sim} \mathcal{N}\left(\theta_{ij}, \sigma_1^2\right)$ where, $\theta_{ij} = \boldsymbol{x}_{ij}^T\boldsymbol{\beta} + \mu_i$. We propose to generalize this linear 'link' by incorporating flexible nonparametric models for the covariates, as follows:

$$y_{ij}|\theta_{ij} \overset{\text{iid}}{\sim} \mathcal{N}\left(\theta_{ij}, \sigma_1^2\right), \text{ and } \theta_{ij} = f(\boldsymbol{x}_{ij}) + \mu_i, \tag{8}$$

where, $f(\cdot)$ is a flexible nonparametric function of the covariates. For example, we can train f via Gradient Boosting Trees (GBT) [8] as our non-parametric model[4]. Trees introduce nonlinearity and higher order interactions among features and can fit complex models. By adjusting tuning parameters like learning rate, maximal tree-depth, number of boosting iterations, etc, they control the bias-variance trade-off and hence the generalization ability of a model. Given a continuous response $\boldsymbol{z}_{n\times 1}$ and covariates $\boldsymbol{U}_{n\times d}$; GBT creates an additive function of the form $f(\boldsymbol{U}) = \sum_k \eta h_k(U)$ where, $h_k(\cdot)$'s are trees of a certain depth and η is the learning rate – the components $\{h_k\}$ are learned incrementally via steepest descent on the least squares loss $\|\boldsymbol{z} - f(\boldsymbol{U})\|_2^2$ with possible early stopping. This imparts regularization and improves prediction accuracy.

Summarizing the general framework: In summary, our framework assumes that we have access to an oracle that solves the following optimization problem

$$\hat{f} \in \text{argmin}_f \left\{\|\boldsymbol{z} - f(\boldsymbol{U})\|_2^2 + \Omega(f)\right\}, \tag{9}$$

with a regularizer $\Omega(\cdot)$ that restricts the family f. Problem (9) encompasses the different models that we have discussed thus far: e.g., model (7) (here, $f(U) = U\boldsymbol{\beta}$ and $\Omega(\boldsymbol{\beta}) = \alpha\|\boldsymbol{\beta}\|_1$), model (6) (here, $f(U) = U\boldsymbol{\beta}$ and $\Omega(\boldsymbol{\beta}) = \alpha\|\boldsymbol{\beta}\|_2^2$); and GBT.

For flexible nonparametric models, a MAP estimator can be obtained by minimizing the negative log-likelihood of the posterior distribution jointly w.r.t $\boldsymbol{\mu}$ and f. This entails minimizing the negative log-likelihood of the posterior distribution – this is given by the function $\mathcal{L}(f, \boldsymbol{\mu})$ (up to constants) as follows:

$$\mathcal{L}(f, \boldsymbol{\mu}) := \sum_{i=1}^{N} \left\{ \sum_{j=1}^{n_i} (y_{ij} - f\left(\boldsymbol{x}_{ij}\right) - \mu_i)^2 + \lambda\mu_i^2 \right\} + \Omega(f), \tag{10}$$

where, $\lambda = \sigma_1^2/\sigma_0^2$. Section 3.4 presents an algorithmic framework for minimizing (10) to obtain estimates $\hat{f}, \hat{\boldsymbol{\mu}}$.

We note that for the class of the models in Section 3.3.1, it is not clear how to tractably construct and compute Bayes/EB estimators–we thus focus on MAP estimation; and note that the associated tasks can be cast as tractable convex optimization problems.

3.3.2 Some Special Cases

As we have noted before, an important contribution of this paper is to propose a general modeling framework – so that a practitioner can glean insights from data analyzing the incremental gains

[4]We also experimented with classical Classification and Regression trees as well as random forests, but the best predictive models were obtained via GBT.

available from different modeling components. To this end, we note that if all the residual user effects are set to zero (i.e, $\mu_i = 0$ for all i), then we can use covariates alone to model the user effects. In the case of model (6) with $\boldsymbol{\mu} = \mathbf{0}$ this is referred to as Ridge in Section 4.3. Furthermore, if we learn $f(\cdot)$ via boosting (with $\boldsymbol{\mu} = \mathbf{0}$) then we recover the model proposed in [20] (denoted as SIGIR2017 in Section 4) as a special case. Predictive performances of these models are presented in Section 4.

3.3.3 Robustifying against outliers

The models described above assume a normal distribution (see (8)) — in reality, to be more resistant to outliers in the data it is useful to relax this assumption to account for heavier tails [9] in the error distribution. To this end, with computational tractability in mind, we propose a scheme which is a simple and elegant modification to our framework, by assuming a stylized decomposition of the link $\theta_{ij} = f(\boldsymbol{x}_{ij}) + \mu_i$ in (8). To this end, we write

$$\theta_{ij} = f(\boldsymbol{x}_{ij}) + \mu_i + s_{ij} \tag{11}$$

and place an additional prior on s_{ij}'s – they are all drawn from π_δ, where, $\pi_\delta(u) = \delta \exp(-2\delta|u|)$ is the Laplace density. The MAP estimator for this joint model requires minimizing the following convex function

$$\begin{aligned}\mathcal{L}(f, \boldsymbol{\mu}, \boldsymbol{s}) := &\sum_{i=1}^{N} \left\{ \sum_{j=1}^{n_i} (y_{ij} - f(\boldsymbol{x}_{ij}) - \mu_i - s_{ij})^2 + \lambda \mu_i^2 \right\} \\ &+ \Omega(f) + 2\delta \sum\nolimits_{ij} |s_{ij}|.\end{aligned} \tag{12}$$

w.r.t. the variables $f, \boldsymbol{\mu}, \boldsymbol{s}$. It is not immediately clear why an estimator available from Problem (12) has robustness properties. To this end, Theorem 2 establishes a crisp characterization of Problem (12) in terms of minimizing a Huber loss [9] on the residuals $y_{ij} - f(\boldsymbol{x}_{ij}) - \mu_i$, where, the Huber-loss is given by

$$H_\delta(a) = \begin{cases} a^2 & \text{if} \quad |a| \le \delta \\ \delta(2|a| - \delta) & \text{otherwise.} \end{cases}$$

The Huber loss is quadratic for small values of a (controlled by the parameter δ) and linear for larger values – thereby making it more resistant to outliers in the y space. The Huber loss remains relatively agnostic to the size of the residuals, therefore offering a robust approach to regression [9]. If δ is small, $H_\delta(a)$ resembles the least absolute deviation loss function—this makes it more suitable for the MAE metric used for evaluation in Section 4.

Theorem 2 *Minimizing Problem* (12) *w.r.t* $(f, \boldsymbol{\mu}, \boldsymbol{s})$ *is equivalent to minimizing* $\mathcal{L}_\delta(f, \boldsymbol{\mu})$ *(below) w.r.t.* $(f, \boldsymbol{\mu})$*:*

$$\mathcal{L}_\delta(f, \boldsymbol{\mu}) := \sum_{i=1}^{N} \left\{ \sum_{j=1}^{n_i} H_\delta(y_{ij} - f(\boldsymbol{x}_{ij}) - \mu_i) + \lambda \mu_i^2 \right\} + \Omega(f) \tag{13}$$

Proof 1 *We use a variational representation of the Huber loss*

$$H_\delta(a) = \min_{s \in \mathbb{R}} \psi(s, a) := \left\{ (s - a)^2 + 2\delta|s| \right\}. \tag{14}$$

To derive identity (14)*, we compute* $\hat{s}$ *a minimizer of* $s \mapsto \psi(s, a)$ *in* (14) *via soft-thresholding:* $\hat{s} = \text{sign}(a)(|a| - \delta)_+$ *(where,* $(\cdot)_+ := \max\{\cdot, 0\}$*). We plug-in the value of* $\hat{s}$ *into* $\psi(s, a)$ *and upon some simplification obtain* (14)*. The proof of the theorem follows by applying* (14) *to Problem* (12)*, where, we minimize* $\mathcal{L}(f, \boldsymbol{\mu}, \boldsymbol{s})$ *wrt* $\boldsymbol{s}$ *to obtain criterion* (13) *involving* $f, \boldsymbol{\mu}$ *(and not* $\boldsymbol{s}$*).*

The above development: decomposition (11) and hence criterion (12) nicely falls within the general hierarchical framework discussed in this paper. In fact all models described before this section can be cast as special instances of Problem (12) by setting $\delta = \infty$ and stylized choices of $f(\cdot), \Omega(\cdot)$. Our numerical experiments suggest that a finite nonzero choice of δ leads to the best out-of-sample prediction performance, thereby suggesting the importance of doing robust modeling in this application. In addition, our model is nicely amenable to the computational methods discussed in Section 3.4. This further underlines the flexibility of our overall framework – even if our basic assumption relies on Gaussian errors at the core, simple hierarchical modeling decompositions of the latent variables make it flexible enough to accommodate adversarial corruptions in the data.

3.4 Computation via Convex Optimization

All the estimation problems alluded to above can be cast as convex optimization problems; for which we resort to modern computational methods [16, 21]. To compute the estimators mentioned in Section 3.3, we need to minimize Problem (12). To this end, we use a block-coordinate descent scheme [21]: at iteration t, we minimize (12) w.r.t. f, followed by a minimization w.r.t the latent vectors $\boldsymbol{\mu}, \boldsymbol{s}$. The algorithm is summarized below.

Algorithm 1: <u>Block-Coordinate-Descent for MAP estimation</u>
Input: $\boldsymbol{X}, \boldsymbol{y}$, tuning parameters, tolerance ϵ; initialization $f^0, \boldsymbol{\mu}^0, \boldsymbol{s}^0$.
Output: An estimate $(\hat{f}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{s}})$, minimizing Problem (10)

(**1**) Repeat Steps 2 to 5 until $|L_t - L_{t-1}|/L_{t-1} > \epsilon$ for $t \geq 1$.

(**2**) Let $\hat{f}^{(t)} \in \text{argmin}_f \, \mathcal{L}\left(f, \hat{\boldsymbol{\mu}}^{(t-1)}, \hat{\boldsymbol{s}}^{(t-1)}\right)$ be a solution of the optimization problem (10) with $\boldsymbol{\mu}, \boldsymbol{s}$ held fixed at $\hat{\boldsymbol{\mu}}^{(t-1)}, \hat{\boldsymbol{s}}^{(t-1)}$ respectively – this is equivalent to solving (9) with $\boldsymbol{z} = \boldsymbol{z}^{(t)}$ where $z_{ij}^{(t)} = y_{ij} - \hat{\mu}_i^{(t-1)} - \hat{s}_{ij}^{(t-1)} \; \forall i, j$.

(**3**) Update the residuals $r_{ij}^{(t)} = y_{ij} - \hat{f}^{(t)}(\boldsymbol{x}_{ij}), \; \forall i, j$. Estimate user-specific effects via: $\hat{\boldsymbol{\mu}}^{(t)} \in \text{argmin}_{\boldsymbol{\mu}} \, \mathcal{L}\left(\hat{f}^{(t)}, \boldsymbol{\mu}, \hat{\boldsymbol{s}}^{(t-1)}\right)$ (with $f, \boldsymbol{s}$ respectively set to $\hat{f}^{(t)}, \hat{\boldsymbol{s}}^{(t-1)}$). This is a closed-form update: $\hat{\mu}_i^{(t)} = \frac{1}{n_i + \lambda} \sum_{j=1}^{n_i} \left(r_{ij}^{(t)} - s_{ij}^{(t-1)}\right), \; \forall i$.

(**4**) Update the vector: $\hat{\boldsymbol{s}}^{(t)} \in \text{argmin}_{\boldsymbol{s}} \, \mathcal{L}\left(\hat{f}^{(t)}, \hat{\boldsymbol{\mu}}^{(t)}, \boldsymbol{s}\right)$, corresponding to the sparse corruptions. The closed form update is $\hat{s}_{ij}^{(t)} = \text{sign}(\eta_{ij}^{(t)})(|\eta_{ij}^{(t)}| - \delta)_+$; where, $\eta_{ij}^{(t)} = r_{ij}^{(t)} - \mu_i^{(t)} \; \forall i, j$.

(**5**) Set the value of $L_{t+1} = \mathcal{L}\left(\hat{f}^{(t)}, \hat{\boldsymbol{\mu}}^{(t)}, \hat{\boldsymbol{s}}^{(t)}\right)$.

Algorithm 1 applies to a fixed choice of the hyper-parameters. We need to consider a sequence of hyper-parameters to obtain the best model based on the minimization of prediction error (see Section 4) on a validation set. In the case of models (6) and (7) estimates of $\boldsymbol{\beta}$ can be computed over a grid of parameters by using warm-starts across different tuning parameters–to this end, the EB estimators provide a good ballpark estimate of relevant tuning parameters. Section 3.5 describes specialized algorithms that are found to speed up the computations pertaining to models (6) and (7) when compared to off-the-shelf implementations of these algorithms. We note that

GBT does not benefit from warm-start continuation across hyper-parameters. For Algorithm 1, we use a stopping criterion of $\epsilon = 0.01$ (Step 1) in the experiments.

3.5 Computational Considerations

We consider certain algorithmic enhancements for Algorithm 1 that lead to important savings when the number of sessions become large (of the order of millions). We focus on the critical Step 2 of Algorithm 1 when $f(U) = U\boldsymbol{\beta}$ and $\Omega(\boldsymbol{\beta})$ corresponds to the ridge or ℓ_1 regularization – this leads to a problem of the form:

$$\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{z}^{(t)} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \Omega(\boldsymbol{\beta}) \right\}. \tag{15}$$

where $\Omega(\boldsymbol{\beta}) \in \left\{\alpha\|\boldsymbol{\beta}\|_2^2,\ \alpha\|\boldsymbol{\beta}\|_1\right\}$. Indeed, in these instances, we found out that the default implementation of Python's `scikit-learn` package [17] was prohibitively slow for our purpose, and hence careful attention to algorithmic details seemed necessary (details below). We derived new algorithms for (15) with an eye towards caching numerical linear algebraic factorizations, exploiting warm-starts, etc; as we describe below.

3.5.1 ℓ_2 regression subproblem

When $\Omega(\boldsymbol{\beta}) = \alpha\|\boldsymbol{\beta}\|_2^2$, the ridge estimator has an analytical expression:

$$\hat{\boldsymbol{\beta}}^{\mathrm{R}} = \left(\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I}_d\right)^{-1}\boldsymbol{X}^T\boldsymbol{z}^{(t)}, \tag{16}$$

which needs to be computed for several tuning parameters, and iterations. To reduce the computational cost, we obtain an equivalent expression for $\hat{\boldsymbol{\beta}}^{\mathrm{R}}$ via the eigendecomposion of $\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{d\times d}$ given by, $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Gamma}\boldsymbol{V}^T$, with $\boldsymbol{\Gamma}$ being a diagonal matrix with eigenvalues $\{\gamma_i\}_1^d$ – this has a cost of $O(d^3)$ in addition to the $O(N_0d^2)$ cost of computing $\boldsymbol{X}^T\boldsymbol{X}$ (and they can both be done once, off-line). This leads to $\hat{\boldsymbol{\beta}}^{\mathrm{R}} = \boldsymbol{V}\boldsymbol{\Gamma}\boldsymbol{X}^T\boldsymbol{z}^{(t)}$ which can be computed with cost $O(N_0d + d^2)$. In our experiments (Section 4) d is small, which leads to a cost that is linear in N_0. The predicted values $\boldsymbol{X}\hat{\boldsymbol{\beta}}^{\mathrm{R}}$ can be computed with an additional cost of $O(N_0d)$. Note that computing estimator (16) for different values of the tuning parameter α does not require additional eigendecompositions – this is critical in making the overall algorithm efficient, especially when training across multiple values of the hyper-parameter.

3.5.2 ℓ_1 regression subproblem

When $\Omega(\boldsymbol{\beta}) = \alpha\|\boldsymbol{\beta}\|_1$, Problem (15) becomes equivalent to a Lasso estimator – we emphasize that `scikit-learn`'s implementation of Lasso became rather expensive for our purposes since it could not effectively exploit warm-starts and cached matrix computations. This motivated us to consider our own implementation, based on proximal gradient descent [2, 16]. To this end, since $N_0 \gg d$, we precomputed[5] $Q := \boldsymbol{X}^T\boldsymbol{X}$ and considered a d-dimensional quadratic optimization problem of the form:

$$\min_{\boldsymbol{\beta}} \quad F(\boldsymbol{\beta}) := \boldsymbol{\beta}^T Q\boldsymbol{\beta} - 2\langle \boldsymbol{X}^T\boldsymbol{z}^t, \boldsymbol{\beta}\rangle + \alpha\|\boldsymbol{\beta}\|_1, \tag{17}$$

[5]Note that this computation is also required for the ridge regression model.

where, the smooth part of $F(\boldsymbol{\beta})$ has C-Lipschitz-continuous gradient – that is, it satisfies $\|\nabla F(\boldsymbol{\gamma}) - \nabla F(\boldsymbol{\beta})\|_2 \leq C\|\boldsymbol{\gamma} - \boldsymbol{\beta}\|_2, \forall \boldsymbol{\gamma}, \boldsymbol{\beta}$ for $C = 2\max_i \gamma_i$ (recall, γ_i's are eigenvalues of Q). A proximal gradient algorithm for (17) performs the following updates:

$$\boldsymbol{\beta}_{k+1} \in \text{argmin}_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \left\| \boldsymbol{\beta} - \left(\boldsymbol{\beta}_k - \frac{1}{L} \nabla F(\boldsymbol{\beta}_k) \right) \right\|_2^2 + \frac{\alpha}{L} \|\boldsymbol{\beta}\|_1 \right\} \tag{18}$$

till convergence. Note that $\boldsymbol{\beta}_{k+1}$ can be computed via soft-thresholding, i.e., $\boldsymbol{\beta}_{k+1} = \mathcal{S}_{\alpha/L}(\boldsymbol{\beta}_k - \frac{1}{L}\nabla F(\boldsymbol{\beta}_k))$ where, for a vector $\boldsymbol{a} \in \mathbb{R}^d$ the ith coordinate of the soft-thresholding operator $\mathcal{S}_\tau(\boldsymbol{a})$ is given by $\text{sgn}(a_i)\max\{|a_i| - \tau, 0\}$. Note that the objective function $F(\boldsymbol{\beta})$ is strongly convex[6]; and hence sequence $\boldsymbol{\beta}_k$ converges to an κ-suboptimal solution to Problem (17) in $O(\log(\frac{1}{\kappa}))$ iterations [16] – i.e., it enjoys a linear convergence rate. Every iteration of (18) has a cost of $O(d^2)$ (arising from the computation of $\nabla F(\boldsymbol{\beta})$ and the soft-thresholding operation). In addition, computing $\boldsymbol{X}^T \boldsymbol{z}^{(t)}$ costs $O(N_0 d)$ (this is computed once at Step 2 of Algorithm 1). Problem (17) needs to be computed for several tuning parameters and iterations (of Algorithm 1) – this does not add much to the overall run-time as the proximal gradient algorithm can be effectively warm-started – this is found to speed-up convergence in practice.

3.5.3 GBT subproblem

When the optimization in Step 2 involves performing GBT, the runtimes increase substantially (See Section 4.5). Unlike the models in Sections 3.5.1,3.5.2; GBT is computationally intensive and needs to be done for every iteration of Algorithm 1. Unlike the optimization based algorithms for ℓ_1/ℓ_2 regression as described above, GBT does not naturally accommodate warm-starts across iterations, and/or tuning parameters.

4 Experiments

Here, we evaluate the effectiveness of our prediction model with respect to several baselines and state of the art session length prediction solutions. We proceed by describing our datasets, our evaluation framework, the comparisons, and then present the results.

4.1 Datasets

We used two different real world datasets of users listening to music, namely, PMusic and lastfm. PMusic is a sample of user interaction data from a major music streaming service in United States, and lastfm is a publicly available dataset from last.fm [4]. We defined the user sessions as periods of continuous listening, interrupted if the user stop or pause the music for more than 30 minutes [20]. For PMusic we gathered data from a small subset of PMusic users for a period of 3 months (February-May 2016) resulting in $3,976,561$ sessions [7]. lastfm public dataset was gathered between 2004 to 2009 and it contains $911,770$ sessions for $1,000$ different users. Table 1 reports some statistics about the user session length in the two datasets. For the log values, we first take the log transform of the raw data, as mentioned in the modeling part, and then normalize. An interesting finding is that mean and median are quite different for the raw data in both datasets.

[6]Note that $F(\boldsymbol{\beta}) - \rho/2\|\boldsymbol{\beta}\|_2^2$ is convex for $\rho = 2\min_i \gamma_i$, i.e., the minimum eigenvalue of Q – this means that $F(\boldsymbol{\beta})$ is strongly convex with strong convexity parameter ρ.

[7]Due to confidentiality we can not report the number of users for this dataset.

Table 1: Summary statistics of normalized user session lengths in the two datasets. The upper half are on the normalized raw session lengths. The bottom half are on the normalized log session lengths.

Stats	PMusic	lastfm
25th quantile(raw)	0.008	0.009
median(raw)	0.021	0.029
mean(raw)	0.044	0.060
75th quantile(raw)	0.049	0.069
25th quantile(log)	0.57	0.59
median(log)	0.66	0.69
mean(log)	0.65	0.62
75th quantile(log)	0.74	0.76

Table 2: Example of user-based and contextual features used in the models.

Feature	Description
gender	gender of the user
age	age of the user
subscription_status	whether the user is ad-supported
device	device used for the session
network	type of network used for the session
absence_time	time elapsed since the previous session
previous_duration	duration of the previous session
avg_user_duration	average user session length (training)
session_time	session started in morning or afternoon

In fact, as reported in [20], Weibull distributions give a better fit to user session lengths, while after a log-transformation, the data can be reasonably modeled via normal distributions, which is what our modeling framework requires for tractable inference.

Feature Engineering. For all the sessions in PMusic we create two kinds of features, namely, user-based and contextual as in [20]. Table 4 reports some of the co-variates used in our models. As user-based features we consider *"gender (the gender of the user), age (the age of the user), subscription_status (whether the user is ad-supported)"*, these features are fixed for a given user. As contextual features we consider *"device (the device used for the session), network (the type of network used for the session), absence_time (time elapsed since the user's previous session), previous_duration (the duration of the user's previous session)"*.

We refine this set of features to include additional contextual features to [20], this is mainly to lower the variance of the past sessions, and introduce non-linearity. We consider as additional features *"avg_user_duration(average user session length in training set), log_avg_user_duration (logarithm of avg_user_duration), log_absence_time (logarithm of absence_time), log_previous_duration (logarithm of previous_duration), session_time (whether the user session started in morning or afternoon)"*. For lastfm dataset the *"age, subscription_status, device, network* are missed.

4.2 Evaluation

We sort our dataset by chronological order, use the first 80% for the training set, 10% for the validation set, and the rest 10% for the test set. Additionally we require each user in the validation or test set to appear at least once in the training set. The final datasets for PMusic and lastfm have respectively in total $3,949,137$, and $713,089$ sessions. For the models that need parameter tuning, we first train the models on the training set for each set of the parameters. Then we use the validation set to pick the best set of parameters. Finally, we use that set of parameters for training on the combined set of training and validation, and predict on the test set. For the evaluation metric of our session length prediction model, we use *Normalized Mean Absolute Error* measured in seconds, averaged over all the test sessions and normalized by the Baseline model which by our definition has $\mathsf{MAE} = 1$. MAE is a good metric due the possibility of important errors resulting from very large session length. More formally, let $|S_{test}|$ be the number of sessions in the test set and $\tilde{y}_{ij}$ be the time spent by user i on his jth session, where j is a test session of user i, and $\tilde{y}^p_{ij}$ be the predicted value then:

$$\mathsf{MAE} = \frac{1}{|S_{test}|} \sum_{(i,j)\in S_{test}} |\tilde{y}^p_{ij} - \tilde{y}_{ij}|$$

4.3 Comparisons

We compare our model with several baselines and state of the art methods. In particular we have considered the following:

Baseline. The baseline model is the per-user mean session length, i.e., we compute for each user the mean session length in the training set and use the value as a prediction value for all the test sessions of the same user.

XGBoost. This corresponds to a Gradient Boosting Model [5] run on basic features to predict session-length. We do not consider the log-transformation.

SIGIR2017. This is method in [20] that is using a modified version of boosting algorithm. Our tuned models have a number of trees in $\{10, 15, 50, 100\}$, with depth $\{6, 10\}$ and use a learning rate in $\{0.1, 0.05\}$.

Baseline can be interpreted as a natural baseline and SIGIR2017 is the state of the art in this particular application. Among the models proposed in this paper (cf Section 3), we consider the following in the experiments:

Model1. This is the model described in Section 3.2, where we don't use any covariates. All the parameters of this model were derived using parameter estimation described in 3.2.

Ridge. This is the Ridge estimator defined in Section 3.3.2, i.e., we perform a ridge regression only on covariates. We take 50 values of the tuning parameter (as per Section 3.3).

Model2-L2. This is the Bayes (which is also the MAP) estimator for the model presented in Section 3.3 with an ℓ_2^2 regularization on $\boldsymbol{\beta}$. We run Algorithm 1 (Section 3.5.1) on a 2D grid of tuning parameters (α, λ) with 500 different values (Section 3.3).

Model2-L1. This is MAP estimator model presented in Section 3.3 with the use of an ℓ_1 regularization on $\boldsymbol{\beta}$. We use Algorithm 1 (Section 3.5.2) for computation, and take 500 values of the 2D grid of tuning parameters (Section 3.3).

Model2-GBT. This model uses Gradient Boosting Trees (GBT) to compute the MAP estimator (10) via Algorithm 1 (Section 3.5.3). We use the same sequence of tuning parameters as in SIGIR2017 and a sequence of 10 λ values in $[1, 10]$.

Model3-L2. This is the extension of Model2-L2 to criterion (12) (or equivalently (13) with the Huber loss)[8].

Model3-GBT. This is the extension of Model2-GBT to criterion (12) (or equivalently (13) with the Huber loss).

Similar to the Baseline model, Model1 does not consider covariates. Model1 however, performs shrinkage on the user-specific effects, and thus any gain in predictive accuracy (as evidenced in Table 3) is due to shrinkage. Ridge considers only covariates and does not include the residual user-specific effects. The rest of the models use features regarding the context and user, and additional user-specific effects. All versions of Model3 allow us to build models that are less sensitive to outliers, hence any performance boost in MAE over its Model2-counterpart can be attributed to robustness. As described in Section 5 we do not have censored data, and we are interested in making point predictions on user session-lengths, therefore survival analysis models are not suitable for our scenario — hence they are not included in our comparisons.

4.4 Effectiveness

We report the results regarding the effectiveness of our model. Table 3 reports the results of the Normalized MAE on all the models in Section 4.3. By borrowing strength across users, Model1 improves over the Baseline even without using any covariate-information. SIGIR2017, the model presented in [20] is benefiting from the usage of the covariates, and it is clearly better than Model1. XGBoost which relies (solely on) covariates, has poor performance on both the datasets. Model2-GBT by combining hierarchical shrinkage with flexible modeling of covariates, reaches a significantly lower MAE than SIGIR2017. This observation shows the importance of the user effect in our hierarchical modeling framework. Model2-L2 is performing quite well in all the datasets and only considers 2 hyper-parameters. We did not observe any gain in MAE by using an ℓ_1 penalization, though the models were sparse (in $\boldsymbol{\beta}$) when compared to ℓ_2 regularization. Model3-GBT has the lowest MAE for all the datasets — thereby suggesting the usefulness of using a robust model for training purposes. For the PMusic dataset, the robustification strategy leads to good improvements: Model3-L2 has MAE 0.891 whereas, its non-robust counterpart Model2-L2 has MAE 0.911. Further improvements are possible by using nonparametric modeling of the covariates. Overall, our Model3-GBT seems to be the best in terms of prediction in both datasets. Model2-L1 or Model2-L2 are close to SIGIR2017 for PMusic and better for lastfm and as we see in Table 6 they are actually much faster in training time.

Feature Importance. By centering and normalizing the columns of the matrix of covariates $\boldsymbol{X}$, the absolute values of the coefficients of $\hat{\boldsymbol{\beta}}$ for Ridge or Model3-L2 suggest the relative importances of the features. Table 4 reports the highest absolute values of the coefficients for the Ridge estimator and for Model3-L2 estimator for PMusic dataset. Device and time-related features appear as the most relevant features. The two most important features for Ridge correspond to logarithm of the average user session length in training set and the absence time since last session. In addition, considering user effect on Model3-L2 lowers the magnitude of time-related features, even though they still appear in the top ones.

[8]For Model3-L2 and Model3-GBT, we take 7 values of $\delta \in [0.1, 10]$.

Table 3: Normalized MAE on test set for our model compared to the baselines and state of the art.

Models	MAE PMusic	MAE lastfm
Baseline- no covariates	1.0	1.0
XGBoost	1.005	0.862
SIGIR2017	0.910	0.826
Model1- no covariates	0.936	0.830
Ridge	0.921	0.828
Model2-L1	0.911	0.824
Model2-L2	0.911	0.824
Model3-L2	0.891	0.822
Model2-GBT	0.878	0.812
Model3-GBT	**0.871**	**0.811**

Table 4: Feature importance for PMusic dataset considering highest absolute value for Ridge and Model3-L2 (we centered and normalized all the features first).

Ridge	*log_avg_user_duration*	0.516
	absence_time	0.430
	avg_user_duration	0.064
	device=smartphone	0.045
	device=Web	0.042
Model3-L2	*device*=smartphone	0.097
	device=Web	0.088
	avg_user_duration	0.085
	absence_time	0.069
	log_avg_user_duration	0.068

Performance Breakdown by Sessions per User. We perform a break down of users into three different types by quantiles of number of sessions per user in the training set. Table 5 reports the normalized MAE for these three groups. The results show a monotonic decrease in terms of gain in performance (with respect to the Baseline) for all the models with number of sessions per user. This serves as a validation of an important message presented through our modeling framework – shrinkage is more critical for less active users when compared to the Baseline. In fact, even for the more frequent users, we observe that shrinkage helps when compared to the Baseline (but the gains are less pronounced). Both Model2-GBT/Model3-GBT outperform the state of art SIGIR2017 for all three cutoffs chosen. The gains obtained by Model3-GBT over Model2-GBT for PMusic dataset can be primarily attributed to the robustness to outliers. Model2-GBT/Model3-GBT perform better than SIGIR2017 by modeling the user-specific effects – this gain seems to be most prominent for heavier users for both PMusic and lastfm datasets.

Table 5: Normalized MAE, restricted to people in the first decile, the first two deciles or the last 8th deciles of the training set. (Hierarchical) shrinkage has a more prominent effect over the Baseline model (with MAE=1) for users with fewer sessions.

Model	$< q_{10}$	$< q_{20}$	$> q_{20}$
PMusic			
Model1	0.860	0.876	0.938
Ridge	0.790	0.809	0.925
SIGIR2017	0.780	0.806	0.904
Model2-GBT	0.778	0.804	0.881
Model3-GBT	**0.767**	**0.792**	**0.875**
lastfm			
Model1	0.610	0.798	0.834
Ridge	**0.606**	0.789	0.833
SIGIR2017	0.622	0.779	0.829
Model2-GBT	**0.606**	**0.775**	0.817
Model3-GBT	0.609	**0.775**	**0.816**

4.5 Efficiency

We further investigate the efficiency of our model by looking at the training time and time of prediction for our best models and state of art solution. We implemented everything using PYTHON [9]. We run the experiments in a MacBook Pro with 2.7GHz Intel Core i5 with 8 GB of RAM. For each model we fixed a set of parameters to tune. We then run each model sequentially, each time with different fixed parameters. Table 6 reports the average running time (across the number of runs done for tuning purposes) and the time to predict all the test instances for our best models and SIGIR2017. The most important thing to note here is that Model3-L2 can be trained between 20 to 30 times faster than any tree based method such as SIGIR2017, and it has an MAE that is 1% to 2% better than SIGIR2017. This is mainly because the Model3-L2 (and Model2-L2) computations benefit from warm-starts (see Section 3.4), so the increase in number of tuning parameters do not increase the run time as in the case of SIGIR2017 and Model3-GBT that are based on Gradient Boosting Trees. They are also at least 30 to 500 times faster in prediction time, and therefore they can be used for real-time prediction with time constraints. While Model3-GBT is slow in training time, it shows better MAE performance compared to other methods. However, we note that these models are meant for off-line training in the context of the application herein–while it is important to have algorithms that are fast, they are not of critical importance as in tasks where real-time learning is of foremost importance.

5 Related work

Session length is an important metric serving as a proxy for user engagement. Therefore the solutions and evaluation are tailored to similar duration based engagement metric such as dwell-time prediction. **Dwell-time.** Liu *et al.* in [15] presented one of the first studies on dwell-time for web search. Kim *et al.* in [12] proposed a dwell-time based user satisfaction prediction model in the web search context. Lalmas *et al.* in [14] proposed new way to improve ad ranking. Barbieri *et al.* in [1] propose to use

[9] Our code will be released, we are removing the link to github due to double blind process.

Table 6: Average training time, time of prediction for our best models and the state of the art. We report for each model the effectiveness (in terms of MAE), training time (in seconds), and average prediction time (in seconds), averaged across the tuning parameters. The GBT-based models have longer training and prediction times, when compared to models that are not based on GBT.

Models	MAE	Train. Time	Pred. Time
PMusic			
SIGIR2017	0.910	731	7.70
Model2-L2	0.911	17	0.24
Model3-L2	0.891	36	0.24
Model3-GBT	0.871	1885	9.71
lastfm			
SIGIR2017	0.826	22	0.53
Model2-L2	0.819	1	0.001
Model3-L2	0.819	5	0.001
Model3-GBT	0.812	66	0.64

survival forest [10] using landing page and user feature in the ad context to estimate the dwell-time and incorporate it in the ad ranking system as a quality score. Vasiloudis *et al.* in [20] presented recently a first session length prediction model in the music streaming context using survival analysis and gradient boosting [5]. In that work, the authors show that in the case of media streaming services the probability of a session to end evolves differently for different users. In particular 44% of the users exhibit "negative-aging" length distributions, i.e. sessions that become less likely to end as they grow longer. Although not directly comparable, we report that this percentage is going to 98.5% for dwell-time on a web page after a search, i.e., after clicking on a search result the more you stay the more you are likely to stay on the clicked page. Finally, Jing *et al.* in [11] presented a neural network based model combined with survival analysis for recommendation purpose and absence time prediction at the same time. In our work we compared against the most recent related work done by Vasiloudis *et al.* [20]. It is worth mentioning that while dwell-time can benefit from survival analysis, because a user can click on a search engine result and never turn back, in our case we don't have censored data, therefore our is a regression problem.

Empirical Bayes and MAP estimation: The statistical models proposed in this paper are inspired by Empirical Bayes methods that are well-known in statistics community, dating back to [18, 7]. In theory, when the model is true, these estimators are known to lead to estimators with better prediction accuracy when compared to (unregularized) maximum likelihood estimators. Empirical Bayes estimators offer an appealing trade-off between frequentist and Bayesian modeling [6] — in that expensive MCMC computations may be avoided by a clever combination of guided hyperparameter tuning and numerical optimization (to obtain MAP estimates for example). We also consider more flexible models wherein MAP (maximum a posteriori) estimation becomes a pragmatic choice from a computational viewpoint. To our knowledge, this is the first time such a methodology is used in the context of user session length prediction.

6 Conclusions and future work

In this paper, we presented a new hierarchical modeling framework, inspired by core Bayesian modeling principles to predict the amount of time people will spent in a streaming service, and in particular listening to streaming music. We also propose modern convex optimization algorithms for enhanced computational efficiency and tractable inference. Our family of flexible models is meant to provide a practitioner insights regarding the incremental gains in predictive accuracy with enhanced modeling components. We focused on predicting the amount of time a user might spend on a platform, at the beginning of the user session.

In our experimental section, we have shown that our method is performing better than the state of the art in this context. Furthermore, we have shown that our model is better for heavy users as well as for users with few sessions. Due to the flexibility of our models, we can achieve lower prediction error for users with many sessions. We can also flexibly incorporate the choice of loss functions that are more robust to outliers in the data. Our results show that our models can lead to an improvement of up to 4.3% in MAE on real-life data. Some of our models can be 22 to 43 times faster (with a 1 to 2% improvement in MAE) in training time; and 30 to 500 times faster in prediction time.

So far we have always investigated a scenario where the model is learned off-line, and it tries to predict the user session length with only few static and off-line extractable features. In the future we aim to extend our model to an on-line version. Furthermore we want to investigate the session length prediction utility within advertising or recommender systems context.

References

[1] Nicola Barbieri, Fabrizio Silvestri, and Mounia Lalmas. 2016. Improving Post-Click User Engagement on Native Ads via Survival Analysis. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 761–770. https://doi.org/10.1145/2872427.2883092

[2] Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2, 1 (2009), 183–202.

[3] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A Context-aware Time Model for Web Search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 205–214. https://doi.org/10.1145/2911451.2911504

[4] O. Celma. 2010. *Music Recommendation and Discovery in the Long Tail*. Springer.

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[6] Bradley Efron. 2012. *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*. Vol. 1. Cambridge University Press.

[7] B Efron and C Morris. 1972. Limiting the risk of Bayes and empirical Bayes estimators. II. The empirial Bayes case. *J. Amer. Statist. Assoc.* 67 (1972), 130–139.

[8] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.

[9] Peter J Huber. 2011. Robust statistics. In *International Encyclopedia of Statistical Science*. Springer, 1248–1251.

[10] Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. 2008. Random Survival Forests. *The Annals of Applied Statistics* 2, 3 (2008), 841–860. http://www.jstor.org/stable/30245111

[11] How Jing and Alexander J. Smola. 2017. Neural Survival Recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 515–524. https://doi.org/10.1145/3018661.3018719

[12] Youngho Kim, Ahmed Hassan, Ryen W. White, and Imed Zitouni. 2014. Modeling Dwell Time to Predict Click-level Satisfaction. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14)*. ACM, New York, NY, USA, 193–202. https://doi.org/10.1145/2556195.2556220

[13] Dmitry Lagun and Mounia Lalmas. 2016. Understanding User Attention and Engagement in Online News Reading. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16)*. ACM, New York, NY, USA, 113–122. https://doi.org/10.1145/2835776.2835833

[14] Mounia Lalmas, Janette Lehmann, Guy Shaked, Fabrizio Silvestri, and Gabriele Tolomei. 2015. Promoting Positive Post-Click Experience for In-Stream Yahoo Gemini Users. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1929–1938. https://doi.org/10.1145/2783258.2788581

[15] Chao Liu, Ryen W. White, and Susan Dumais. 2010. Understanding Web Browsing Behaviors Through Weibull Analysis of Dwell Time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, New York, NY, USA, 379–386. https://doi.org/10.1145/1835449.1835513

[16] Yu Nesterov. 2013. Gradient methods for minimizing composite functions. *Mathematical Programming* 140, 1 (2013), 125–161.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[18] Herbert Robbins. 1954-1955. An empirical Bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, Vol. I. UC Press, 157–163.

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

[20] Theodore Vasiloudis, Hossein Vahabi, Ross Kravitz, and Valery Rashkov. 2017. Predicting Session Length in Media Streaming. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 977–980. https://doi.org/10.1145/3077136.3080695

[21] Stephen J Wright. 2015. Coordinate descent algorithms. *Mathematical Programming* 151, 1 (2015), 3–34.