

UNIVERSITY COLLEGE LONDON

DOCTORAL THESIS

**Song plays: Using machine learning to
model the temporal patterns of music
listeners**

Author:

Badrul ALOM

Supervisor:

Dr. Mark HERBSTER

Advised by:

Pedro Mediano (Emotech
Ltd.)

*A thesis submitted in fulfillment of the requirements
for the degree of Master of Science*

in the

Department of Computer Science

August 22, 2017

Declaration of Authorship

I, Badrul ALOM, declare that this thesis titled, “Song plays: Using machine learning to model the temporal patterns of music listeners” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

University College London

Abstract

Faculty Name

Department of Computer Science

Master of Science

Song plays: Using machine learning to model the temporal patterns of music listeners

by Badrul ALOM

While areas of customer recommendation have received a lot of attention in recent years, predicting users behaviour, particularly the timing of their actions, has not had as much focus. The modeling of temporal processes representing human behaviour can be applied across many industries. In this research we examine different methods for predicting a user generated event based on analyzing the timing of their past interactions. Our context is that of music listening and we examine a Bayesian inference model, logistic regression, a linear and RBF SVM, and a RNN-LSTM model. We find that precision and recall pose a trade-off, with a linear SVM being better at the former and an RBF model being better at the latter. We also find that an RNN model achieves relatively good result with a low amount of training data and expect that as computational power improves it will lead to top in class results in this area.

Chapter 1

Introduction

The modeling of event sequences is useful across industries. For instance the periods in which a customer makes an online purchase can help determine the optimal periods for target marketing. The times at which public transport users tend to travel can help better manage resources to meet demand. The times at which a medical illness re-occurs can help predict future episodes.

In all these cases modeling the temporal behaviour of the system is important in predicting the next event. While the area of product recommendation has received extensive attention in recent years, the area of recommendation timing less so. This research looks at how we can model the temporal behaviour of individuals, in order to predice the probability of an event happening at time t .

1.1 Context

We take as our context for this research, the goal of estimating the probability that a user of a home-audio device would like to listen to music at a time period t , given their play history h . One application of this research would be to allow home audio devices to recommend music to a user at an ooportune time. It could then also be extended for other activities.

The goal will be evaluate the effectiveness of several different machine learning methods. The research was guided by Emotech Ltd., a home audio hardware and software company and the creators of Olly [9].

1.2 Data

The dataset being used in this analysis is the LastFM1k dataset, which is freely available online and contains the listening history of a thousand LastFM listeners. It consists of a series of timestamps denoting when user started played a song. We wish to learn the temporal patterns of a user's behaviour in order to predict the next item in the sequence - a play or non-play event.

The dataset contains the timestamp, user ID, and track ID of users listening habits over a number of years (2005-2009).

1.3 Structure of the report

We first perform a literature review of

Chapter 2

Literature Review

Within literature the problem of predicting the next occurrence of an event is referred to as event prediction, sequence prediction, or point process modeling. Analyzing data of this sort presents its own unique challenges and questions. What are the ways to represent the data? What types of stochastic models are appropriate for explaining the structure in the data? How can we measure how well the data is described by a particular model?

Regarding the representation of the event data, there are three ways of doing this, event count, event times, and inter-event intervals (see figure 2.1).

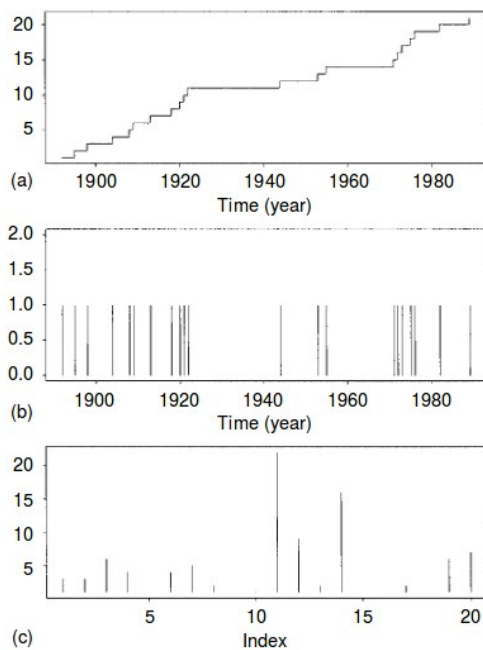


FIGURE 2.1: Three different representations of the same point-process
a) cumulative count b) date of occurrence c) interval time between floods

Inter-event has traditionally been one of the most common ways of representing the data, for example in the times between financial transactions [4].

2.1 Poisson Point Process

Poisson processes [5] are characterized by the fact that the probability of an event in a small time interval is independent of the event at all other times.

By making use of this independence assumption one can make sure of models that construct a poisson distribution to describe the data. Due to their ease of use Poisson processes are often the first-order models used to characterize spiking systems. However given the intercorrelation between time intervals they are not so useful for temporal point processes. However in some cases it is possible to aggregate temporal data into a form that removes the inter-dependence between occurrences. We see this in used in the Bayesian Inference model described in the next chapter.

2.2 Temporal Point Processes

A temporal point process [2] is a way of modeling event data as a sequence of a fixed period intervals. Temporal point processe models seek to estimate the probability of an event happening at time t , based on an event history upto, but not including, time t . At its simplest it can be represented as:

$$\xi = \sum_{i=1}^n \delta_{X_i},$$

where δ denotes the Dirac measure, a probability measure of whether a set contains point X or not.

In the case of music listening we can model a sequence of events and non-events, where t_i can either be 0 (did not play music) or 1 (played music). This is the form we use in all but the Bayesian model as described in the next chapter.

2.2.1 Conditional Intensity Function in Point Processes

The conditional intensity function is a key part of traditional point processes modeling [3] where the probability of an event $\lambda(t)$ is derived from a stochastic model of the inter-event time, for instance the Poisson process at time t .

Conditional intensity functions can be inhomogenous such as with a Gaussian Kernel $\lambda(t) = \sum_{i=1}^k \alpha_i (2\pi\sigma_i^2)^{-1/2} \exp(-(t - c_i)^2/\sigma_i^2)$, for $t \in [0, T]$ where c_i and σ are fixed center and standard deviations, respectively, and α_i is the weight for kernel i .

Or they can vary in intensity, such as with the self-exciting (Hawkes) process where the intensity is determined by previous events through the parametric form $\lambda(t) = \mu + \beta \sum_{t_i < t} g(t - t_i)$ and where g is some non-negative kernel function.

As noted by Wass et. al [16], conventional point process models often make unrealistic assumptions about the generative processes of the event sequences. The conditional intensity function make various parametric assumptions about the latent dynamics governing the generation of the observed point patterns. As a consequence, model misspecification can cause significantly degraded performance using point process models.

2.3 Deep RNN Point Process Models

In recent years deep learning has demonstrated the power to learn hierarchical non-linear patterns on large-scale datasets [7] through multiple layers of abstraction (e.g. multi-layer feedforward neural networks). It has achieved state-of-the-art performances on a wide range of applications, such as computer vision [6], natural language processing [14], and protein structure prediction [8].

However it has not been applied to temporal point processes until recently with Xiao et. al [16] applying Generative Adversarial Networks (GANs) to the problem. GANs consist of two neural network models - a generator tasked with generating (i.e. predicting) a future sequence of events based on the history, and a discriminator tasked with detecting the true (ground truth) sequence amongst the generated ones.

For measuring the loss between a generated and true sequence, the authors found the Wasserstein-Distance [10] performed better than Maximum Likelihood Estimate (MLE) which they remarked "may suffer from mode dropping or get stuck in an inferior local minimum".

Their findings showed that where as parametric point process models work better with problems where a parametric form exists, with real world data a GAN model with Wasserstein-Distance outperformed all other models (including an RNN model using MLE). This signals a promising new direction for temporal point process research.

Chapter 3

Experimental Design

In this chapter we describe the different methods that were assessed for the task of predicting whether a user is interested in listening to music at time t given their play history h . The methods, in order of gradually increasing sophistication, are as follows:

1. Baseline model
2. Bayesian Inference
3. Binary Logistic Regression
4. Linear SVM Classifier
5. Non-Linear SVM Classifier
6. Recurrent Neural Networks

All methods, except for the Bayesian Inference method, require the data to be structured as a time-series. The Bayesian method adopts a different approach that requires the data to be aggregated into half-hourly buckets of a week.

The data preparation that was performed is described first in the next section, followed by an explanation each method, and the evaluation criteria for assessing the methods.

3.1 Data preparation

3.1.1 Data transformation

The analysis was carried out in Python (via Jupyter notebooks) running on Ubuntu. The raw data consisted of timestamps of when a song was played and a user ID. These were loaded as-is into a SQLite3 database in order to reduce the need to repeat data preparation steps. The methods themselves utilized Scikit-learn for all models, save for the RNN model which used Tensorflow.

UserIDs were converted to integer (e.g. 'User0005' became '5') and a period lookup table was created at n minute intervals, to which all timestamps in our main dataset could be mapped to. n was chosen to be 30 although it is possible to re-run the analysis for other levels of granularity.

More significantly the data, which contained entries for the times at which each user listened to music, was supplemented with all the times they did *not* listen to music, between their date of their first and last play. This was required in order to generate a sequence of play and non-play events. It was noted however that in cases where a user stops listening to music for a long period, then listens to it sporadically, we would get an even greater imbalance of data (see later).

3.1.2 Time-series fetures

The approach to feature selection for this research was to devise an initial of features based on the preliminary analysis, and use that list across all models.

The time series features that were derived from the raw data were time-lags: $t, t-1, t-2, t-3, t-4, t-5, t-12hrs, t-23.5hrs, t-24hrs, t-24.5hrs, t-1wk, t-2wks, t-3wks, t-4wks$ where t is the event being predicted.

$t-1$ to $t-5$ represent user activity in the previous 2.5 hours. The remaining time-lags were chosen to represent half-day, daily, and weekly cycles, with additional empahsis around the -24 hour mark due to the daily patterns observed in the preliminary analysis (see next chapter).

In addition a few non-time lag features were selected. One of which was the number of hours away from 5pm in either direction (so a timestamp at 4pm and 6pm would both equal 1), based on the observations in the preliminary analysis of 5pm being a peak hour. The others were binary features representing the day of the week (isMon, isTue etc.). Again these were based on the patterns observed in the preliminary analysis.

To improve the speed and quality of our models, rows where all time-lags indicated a non-play were removed from the dataset. Only 0.62% of such cases were found to be a play-event at time t .

3.2 Validation and Test dataset selection

Our working dataset was a subset of the full 1000 user dataset, and comprised of 4,217,228 rows of training data across 97 users. Of this a random sample of 100,000 rows was taken on which 5-fold cross-validation.

Due to the time it takes for training cross validation was not performed on the RNN model, instead 10 randomly selected users were held back from the main training dataset and a random sample of 10,000 rows were taken from each user.

Data Imbalance

Data imbalance is when the training data is when one or more of the classes is under-represented in the dataset. Of the 4,217,228 rows in our data set, 361,081 (8.6%) were play events and 91.4% were non-play events.

This could lead to models that achieve a high accuracy score by following either or both of the following two heuristics: * Predict non-event for everything * Predict t will be the same as $t-1$

There are several methods for dealing with data imbalance [1] including restricting input data, having a weighted loss function, and using recall as an evaluation measure.

In this research class weights were employed to automatically adjust weights inversely proportional to class frequencies in the input data $n_{samples}/(n_{classes} * np.bincount(y))$.

This would have the impact of encourgaing the model to predict play events more frequently. However our goal is to encourage the model to predict play events, while minimizing the number of false postivies on the predictions (i.e. we attach less cost to false-positives for non-event prediction). Encouraging this sort of behaviour requires not only class weighting, but also sample weighting to give a higher penalty to false positives for play-predictions. This strategyh was also employed in the models as shown in the following Python snippet (1 represents a Play event):

$$sampleWeights = 1 + (y[:,] == 1) * (x[:, 1] == 0)$$

3.3 Methods

3.3.1 Baseline Model

Our baseline model will be to assume $t = t - 1$, that is to say a person will listen to music at period t if, and only if, they listened to music in the period immediately prior. As music listening events tend to be clustered (people listen to music in batches) the accuracy of the baseline model is expected to be fairly high. However it will not be able to predict the first play of a listening session, or where the listening session duration lasts no longer than a single period, which in the experiments is defined as a 30 minute period.

3.3.2 Bayesian Inference

We employ a simple Bayesian Inference approach by utilizing the Beta-Binomial model. The Beta-Binomial model is built upon the weekly patterns observed in the preliminary analysis. Conceptually it seeks to build up a users personalized weekly listening profile as observations come in, using a Beta-Binomial probability distribution, with the priors based on the population as a whole. We then assess how effective this profile is at predicting listening events for that user.

Prior probabilities were calculated for each half-hourly period in a week ($24*2*7=336$ timeslots). Fig 3.1 shows the calculations for the first 2.5 hours of a Sunday (d-hour-hh format).

The likelihood function which represents the probability of a user listening to music was defined as a binomial distribution, where k is the number of plays in a given period, n is the sum of plays and non-plays, and θ is the unknown probability parameter for the binomial distribution.

$$\binom{n}{k} p^k (1 - p)^{n-k}$$

As the Beta distribution is a conjugate prior to the Binomial the model can be reduced to: $Beta(\alpha + P, \beta + Q)$ where P is the count of plays and Q is the count of non-plays.

Our parameters for the beta distribution, α and β , are derived from the training set, with an estimate of the mean for each half-hourly time period as shown in fig 3.1.

Timeslot	mean	var	a	b
1-00-1	0.098577	0.010807	0.711953	6.510370
1-00-2	0.092327	0.011242	0.595911	5.858451
1-01-1	0.090256	0.011784	0.538632	5.429205
1-01-2	0.089523	0.011741	0.531937	5.409996
1-02-1	0.087637	0.011772	0.507577	5.284259

FIGURE 3.1

We first calculate the probability of a play (total plays in period / count of plays and non-plays) per user, then take the mean and variance across users. a and b are then determined as: $a = (\frac{1-\mu}{\sigma} - \frac{1}{\mu})\mu^2$ and $\beta = \alpha (\frac{1}{\mu} - 1)$.

Finally we convert the probabilities into a binary outcome by optimizing for a threshold λ at which we predict a play event.

3.3.3 Binary Logistic Regression

This method (and the subsequent methods) adopts a more classical time-series approach to the prediction problem by constructing a dataset as a sequence of play events at time t $Y_t = 1$ and non-play events $T_t = 0$ with t representing a time period of a fixed interval, which for our research will be 30 minute chunks. We seek to calculate the probability of an event in the current time period t , given the history of events: $p(Y_t = 1 | Y_h)$, for each individual user together with additional non-time-series features (see .

Our binary logistic model is therefore defined as:

$$p(Y_t = 1 | Y_h) = \sigma(w^T x + b)$$

$$p(Y_t = 1 | Y_h) = 1 - p(Y_t = 0 | Y_h)$$

with σ being the sigmoid function defined as:

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

Determining the optimal weights and constant can be determined by maximization of the log-likelihood or the minimization of the negative log-likelihood [**NegLog**]. The latter is given as:

$$C \sum_{i=1}^n \log(\exp(-y_i(X_i^T w + b)) + 1)$$

for class $C = 1$.

In addition we will be comparing results with and without the L2 regularizer term: $+1/2w^T w$

Regularization is technique for encouraging the model to converge on a smaller set of strong features rather than a wide set of weak features and is important in helping us determine which time-lags and additional features are important and which are not.

3.3.4 Linear SVM Classifier

SVM models seek to determine a separation plane between classes based on the support vectors, the data points closest to the decision boundary.

A linear SVM regression model performs this through the Epsilon Intensive loss function [15]. The objective becomes to *minimize*:

$$\max(0, \|(y_i - w_i x_i - b) - \epsilon\|)$$

In other words we ignore cost functions that are within a certain margin ϵ . In our case this may be of importance in cases where the probability of user listening to music is close to the decision boundary, which may be the case for the very first song played at the start of a session.

3.3.5 Non-Linear SVM Classifier

Here we extend a regression model to include a Gaussian RBF kernel. A Gaussian kernel is a popular method for modeling non-linear decision boundaries, so for example if the decision to play music is based on some non-linear combination of day of the week, time of the day, and time since last play. Our model becomes:

$$p(E = 1) = b + \sum_{i=1}^N w_i RBF(x, x_i)$$

where the RBF kernel is defined as:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

Note that our actual implementation in Tensorflow makes use of the more computationally efficient algorithm as defined by McClure [11]. This restated method has a parameter γ that takes the place of $2\sigma^2$ and which requires tuning.

Mini-batching was also performed with a batch size of 2000 and 5 iterations of the dataset, as it was found that rows of more than 5000 caused memory errors, and iterations beyond 5 showed minimal or no decrease in loss).

3.3.6 Recurrent Neural Networks

Recurrent nets are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies. Whilst a traditional Feed-Forward network [13] has input nodes, hidden layers, and an output layer, with data flowing in one direction only, RNNs allow for the hidden state from one timestep of the neural net to be an input into the next (see fig. 3.2).

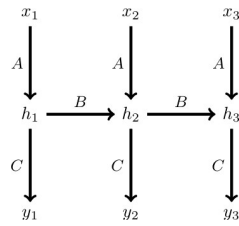


FIGURE 3.2: RNN with 3 timesteps

RNN models can employ different methods the propagation of the hidden state over time. One such well known method is Long Short-Term memory (LSTM) [12]. Here the hidden state is the product of a further four layers that interact in a way as to learn what information to retain and what information to throw away.

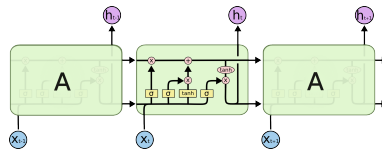


FIGURE 3.3: LSTM

We could therefore provide the model with a sequence of historical data at times $t - 1..t - n$ where $n > 1$) which would be fed into the RNN as separate time steps in forward temporal order. The hidden state would propagate information from t_n forward at each time step, until it was used to predict the outcome at a time t .

Practically speaking this means feeding in the data with input shape (batch rows, time-steps, features). Some points of interest to fellow researchers are:

1. The features dimension here does not need to contain all time-lags as we do in our other time-series models. However it must contain $t-1$. In this research we experiment with both $t-1$ only and all our timelag features in this research.
2. The time-steps dimension is where the features list in previous time steps will be stored. These are then 'unrolled' within Tensorflow and fed into the LSTM. In this research we experiment with different lengths of time-steps.
3. When the data is unrolled, time step t for all rows in the batches are processed together as one block, before moving onto the next time-step.
4. Constructing the 3-d shape often requires building them up in slices. A significant speed up was observed in Python when using a pre-allocated array vs. appending to it.

3.4 Evaluation criteria

Deciding on an appropriate evaluation measure requires careful consideration to the costs attached to different predictions. In our case we assign a lower cost to predicting a non-play event, and being wrong, than predicting a play-event and being right (as the cost of suggesting music when the user is likely not interested is higher than not playing music when they likely not interested).

We also wish to rate models higher if they manage to predict the first play of a listening session, as this is seen to be harder than simply predicting a play event, when the last n events were also a play event.

With these in mind a number of possible evaluation criteria were looked at. The most-straight forward of which was *accuracy*. This computes the count of correct predictions as a fraction of the total number of predictions. While this is an intuitive measure, it would not distinguish between models that had a high accuracy on the play-events vs. a high accuracy on non-play events.

To do this we look at precision. Precision (P) is defined as the number of true positives over the number of true positives plus the number of false positives. A positive in this case is a play-event so our precision equation becomes:

$$Precision = \frac{CorrectPlayPredictions}{TotalPlayPredictions}$$

This will be measuring our models on how many of their play event guesses were correct which takes care of one of our considerations outlined above. However it does not distinguish between models that predict a play event only when the previous event was a play (a 'safe bet' and so a high precision score) vs. models that predict correctly in cases where the previous event was not a play and are therefore harder to get right. We turn to another measure to aid with this: recall.

Recall is defined as the number of true positives over the number of true positives plus the number of false negatives. A false negative is where a model predicts a

non-event, when in fact there was an event. For example if we predicted 100 plays correctly but there were in fact 110 plays in the dataset, then recall would be $100/(100 + 10) = 91$

$$Recall = \frac{CorrectPlayPredictions}{TotalPlayPredictionsInDataset}$$

Precision and recall in combination was therefore our chosen evaluation method. In addition tests were carried out examine how well the models performed on predicting the first play event in a series.

3.5 Summary

We have described how our data was transformed to make it useful for our experiments. In particular how we had to convert our list of play events into a list of play and non-play events for every period. By doing this we are faced with an imbalance of data as non-play events make up 91% of the data and so we employ a weighting strategy to counteract this.

We then described the methods we will be utilizing in our experiments, consisting of a Baseline model which is simply to assume $t = t - 1$, a Bayesian model which build up a weekly profile of listening habits for each user, Logistic and SVM models that apply classical machine learning time-series prediction techniques to our problem, and finally a deep learning model in the form of an RNN-LSTM where we seek to utilize the capabilities for it to learn temporal patterns through a hidden memory state.

Finally we looked at different ways for measuring the success of our problem, with precision and recall being the preferred choice.

In the next chapter we shall present the results of our experiments and a discussion of what this tells us about learning user-level temporal patterns and performing event prediction.

Chapter 4

Results

We begin our discussion of the results with preliminary analysis of the data that helped shape the experimental design. Here we seek to understand what are some of the overarching patterns in music listening habits and how these look at an individual level.

After this we present a summary of our results followed by a discussion of the performance of each individual method.

4.1 Preliminary analysis

4.1.1 Daily play patterns

By grouping track plays into 30 min intervals and aggregating by periods within a day, we see a clear daily pattern with music listening hitting a peak at around 5pm and a trough at around 6am.

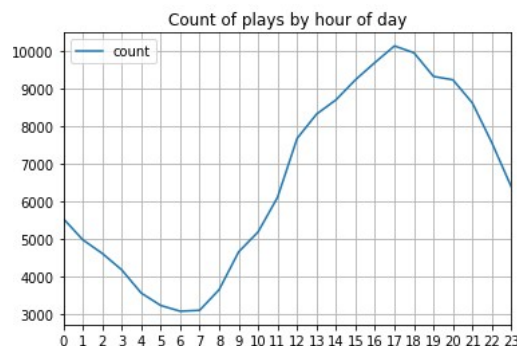


FIGURE 4.1: 5-5.30pm is peak listening time

Zooming out to view the pattern across an entire week in figure 4.2, we see that the daily pattern occurs across every day of the week with weekends having a lower total number of plays.

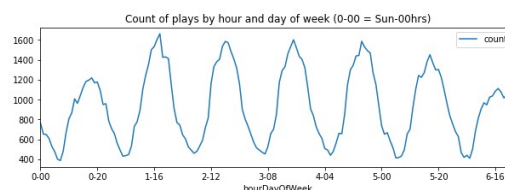


FIGURE 4.2: Most popular times to listen to music across all users

At a high level therefore one can get good accuracy by simply anticipating music demand to peak at 5pm. However if we select two users at random, we see (see

fig. 4.3) that these daily patters are not as strongly discernable. This demonstrates why models modeling the high levels patterns is not enough for individual user prediction.

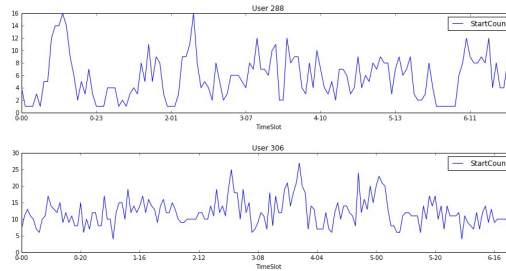


FIGURE 4.3: Most popular times to listen to music by individual user

4.1.2 Inter-event times

The dataset contains a timestamp associated with each user. This does not necessarily mean the user played a song in its entirety. Analysis shows plenty of cases where the interval time between tracks was a few seconds suggesting the user skipped tracks.

Figure 4.6 shows a frequency plot of intervals. Intervals beyond 30 minutes continue the exponential decrease and are not shown. We see that while the mode is on par with a typical song length, there is a significant number of plays that lasted under 5 minutes.

4.1.3 Time-series analysis

Here we examine our data once it has been transformed in a binary sequence of events (1) and non-events(0). We seek to understand better how an optimizer may perform based on traits of the data.

We begin with assessing how well our baseline model may perform based on assuming $t = t - 1$. Fig 4.4 shows that the 76% of Plays, also had a play in t-1. However simply using this as a rule would also capture 2.2% of non-plays.

	Count	%
Plays	361,081	
of which t-1 is a play	274,985	76.2%
of which t-1 is a non-play	86,096	23.8%
Non-Plays	3,856,147	
of which t-1 is a play	86,088	2.2%
of which t-1 is a non-play	3,770,059	97.8%
Total	4,217,228	

FIGURE 4.4

Furthermore the 23.8% of Plays that did not have a Play in the prior period are harder to predict yet of more interest as they represent the beginning of the listening period and therefore more useful to a music recommender system.

Given the daily patterns we have seen, it might be reasonable to assume that looking at the same period 24 hours prior may be a good indicator of whether t is a play event. However as we see from fig. 4.5 this is not a reliable indicator either.

	Count	%
Plays	361,081	
of which t-1 is a play	274,985	
of which t-24hrs is a play	102,522	37.3%
of which t-24hrs is a non-play	172,463	62.7%
of which t-1 is a non-play	86,096	
of which t-24hrs is a play	23,478	27.3%
of which t-24hrs is a non-play	62,618	72.7%

FIGURE 4.5

What both of these results tell us is that fairly high precision score of around 76% ought to be possible purely based on t-1 but going above this while having a good precision score on the non-play events will be harder.

Finally it was found that of rows where all timelags had a non-play event, 0.62% of rows had a play event. As this was a low number it was deemed safe to remove all such rows from the dataset in order to improve the speed and quality of analysis.

4.1.4 Outliers

The data was checked for any unusual outliers that may impinge upon the goal of developing a model to predict user behaviour. An analysis of plays by user reveals a high amount of variance between users on how many tracks are played.

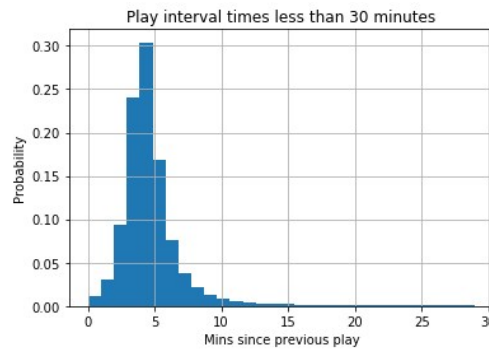


FIGURE 4.6

For our purposes these are include as evidence that the user was interested in playing music at time t and therefore treated as a Play event.

We can also assume that the song plays are not independent of one another, in that the probability of a play event at time $t+1$ is significantly higher if there was an event at time t .

Further analysis showed one user in particular with very high amount of plays, with very low durations, suggesting it was likely to have been generated by a bot, possibly a LastFM test. This was excluded from the dataset.

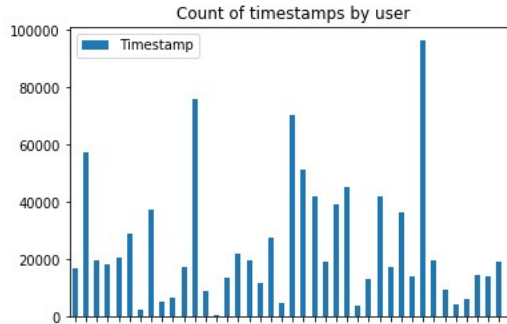


FIGURE 4.7: Total play count by user

4.2 Main results

4.2.1 Summary

Fig. ?? shows the results from our experiments. We see how that the baseline model scores highly on precision but low on recall. In other words given that a sequence of plays often appear together, assuming $t = t - 1$ is often quite often correct, but it fails to pick up the start and end points of a sequence of song plays.

Model	Train rows	Precision	(+/-)	Recall	(+/-)
Baseline	500k	79%	0.02%	13%	0.03%
Bayesian Inference	500k	41%	0.5%	13%	0.5%
Logistic Regression	500k	67%	0.5%	73%	0.1%
Linear SVM Classification	500k	81%	0.04%	68%	0.1%
SVM RBF Classification	100k	77%	0.01%	76%	0.1%
RNN-LSTM (T-1)	50k	70%	0.1%	69%	0.1%
RNN (T1 .. T5weeks)	50k	70%	0.8%	64.0%	0.7%

FIGURE 4.8: Summary of results. Note that both RNN models included non-time lag features (e.g. IsMon)

The Bayesian Inference model performs poorly on both measures. This would suggest that the difference between the population patterns and what is observed at an individual level is too large.

Our remaining models exhibit the tension between achieving a good accuracy and achieving a good recall. The Logistic regression model performs below average on precision but is second highest on recall. The Linear SVM and RMF regression models are on par with one another with the former achieving the highest score on prevision, and the latter on recall.

The Linear SVM model ignores probabilities that fall within a margin of the decision boundary during its optimization process. That this translates into performing well on precision may be due to placing lesser emphasis on ad-hoc play events that aren't indicative of a users general behaviour, as these probablities would likely fall near the decision boundary. Alternatively it could be better at dealing with the first play of a sequence whereby if often, but not always occurs at the same time each day; again leading to probabilities nearer the decision boundary.

The RBF model generally performs well on both metrics. Note that the RBF classification was restricted to 100k rows of training data as it was found the computational cost of 500k rows was too high so it had less training data to work with than

the Linear SVM. The high recall rate of 68% indicates that inputs have a non-linear relationship to the output and that this is especially helpful to predicting the start and end of a play sequence as seen by the high recall score.

The RNN models were trained on 50k rows of data due to the computational cost of evaluating a large number of hyperparameters. Two types of models were tested - one with $t - 1$ as the only time-lag information and one with all other time-lags included. Given the much smaller training set the results we observe are better than expected. The final parameters for the models were a 250 units, 4 LSTM layers, and a class-weighting of 2.0 for play events.

4.3 Beta-Binomial analysis

We can plot the prior probability for any given time period as shown in fig 4.9.

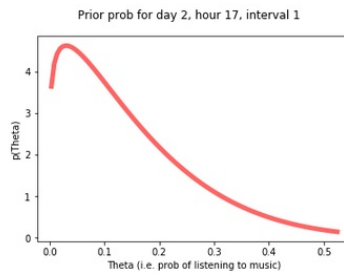


FIGURE 4.9

Here we see that the θ , which represents the prior probability of listening to music in the timeslot, is most likely to be less than 0.1 for this timeslot. Note that this timeslot is for hour 17 (i.e. 5pm) which from our preliminary analysis we know is the peak listening time. The fact that the probability is so low suggests that users may often have weeks in which no music was played, and hence the prior probability for a timeslot is dragged down.

For our posterior function, the threshold at which we determine a Play event was determined by comparing the false positive rates with the true positive rates using a ROC curve (4.10). From this, 0.4 was selected as the optimal threshold at which to determine a Play event across all users.

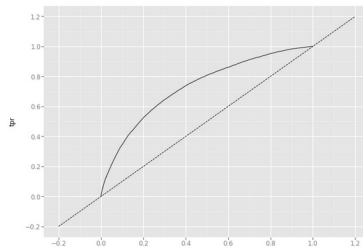


FIGURE 4.10: ROC curve showing 0.4 as the optimal threshold

The results from the model are shown in figure 4.11. We see that the recall and precision of play events (as denoted by 1) is very low suggesting that relying on a Bayesian approach centered around a weekly profile of each users habits is not an effective method for predicting a play event for a new time period.

	precision	recall	f1-score	support
0	0.93	0.98	0.96	227823
1	0.40	0.13	0.20	19763
avg / total	0.89	0.91	0.89	247586

FIGURE 4.11: Beta-Bionmial Model Results

4.4 Logistic regression analysis

An advantage of logistic regression over non-linear models is that the coefficients are easily interpretable. While feature engineering was not the focus of this research an examination of these can yield insight into which features may be unnecessary.

Field	Log Reg.
t-1	3.99677
t-2	0.66650
t-3	0.42272
t-4	0.40080
t-5	0.49478
t-23.5 hrs	1.12593
t-24hrs	0.55006
t-24.5 hrs	0.53480
HrsFrom5pm	-0.05715
isSun	-0.23220
isMon	-0.10579
isTue	-0.14689
isWed	-0.12977
isThu	-0.14111
isFri	-0.19522
isSat	-0.27923

FIGURE 4.12

Fig 4.12 shows that $t - 1$ was by far the most important feature. This is expcted and it's importance crowds out the other features.

Interestingly $t-23.5$ is the second strongest time-lag and more significant than $t-24$ hrs. This likely reflects the fact that having a consistent $t - 1$ is more important than simply knowing what t was 24 hours prior.

The non-time lag features seem to have very little effect, which was also see in fig 4.13 where we do a forward stepwise regression are barely impacted by these features.

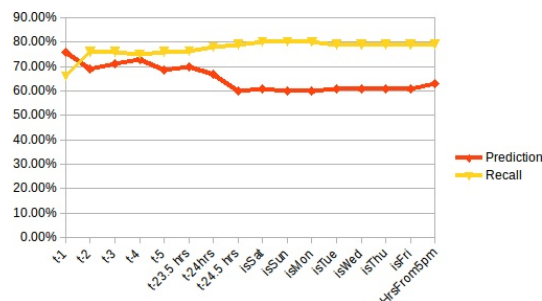


FIGURE 4.13: Stepwise regression, adding in fields from left to right

4.5 RNN analysis

For our evaluation of the RNN-LSTM model the dataset was reduced in order to perform the desired number of hyperparameter tests. Typical run times were several hours long. The number of hidden layers, units, timeteps, samples, and iterations all played a significant part in impacting the time it took to train the model.

Fig 4.14 shows the results of some of the other tests that were performed. The table has been shaded to aid understanding with darker shading considered 'better' whether it be more samples of the data were used, or a bigger graph was constructed.

Run number	1	2	3	4	5	6	7	8	9	10	11
Features list	t-1	t-1	t-1	All	All	All	All	All	All	t-1	All
User iteration	2	5	5	20	3	10	20	100	20	20	20
Samples per user	1000	1000	1000	500	2	10	20	100	10	50	50
Total samples	2000	5000	5000	10000	6	100	400	10000	200	1000	1000
Batch Iterations	20	50	10	10	5	5	5	20	10	10	10
Batch size	1000	1000	1000	20	100	50	50	50	10	50	50
Time steps	336	672	672	672	672	672	5	5	48	672	672
Hidden Units	150	250	250	250	250	100	100	150	150	250	250
Layers	1	1	1	1	2	3	4	4	4	4	4
Imbalance data weighting	none	none	none	none	none	none	none	none	2	2	2
Precision	57.00%	50.00%	71.00%	66.00%	57.00%	22.00%	77.00%	48.00%	36.00%	70.00%	70.00%
Recall	14.0%	23.0%	62.0%	46.0%	33.00%	14.00%	42.00%	43.00%	31.00%	69.00%	64.00%

FIGURE 4.14: RNN results over multiple runs. Darker shading means better

We see that the improvement in precision comes about most significantly through the use of 4 layers (see run no. 7) although a higher number of iterations of a mini-batch also seems to help improve precision (see run 2 vs. 3).

Yet too many iterations of a mini-batch adversely impacts recall. Good recall appears to be a product of a higher number of hidden units in a layer, a higher number of time steps being fed into the LSTM, and a higher number of layers. All these supports the idea that the ability to detect the start and end of a play session, requires a highly non-linear solution.

Chapter 5

Summary

5.1 Conclusion

Predicting the propensity of a user to listen to music at a certain time, based on their recent listening history can be applied to a range of other areas such as the propensity to purchase, electricity usage, or the demands on a public transport system. In all these a good modeling of the temporal patterns helps with prediction.

Within the literature modeling the problem as a temporal point process is one of the most common methods and this research did likewise in several of the machine learning algorithms that were evaluated.

As music listening is typically long periods of non-play events, followed by sequential periods of play events, the problem becomes one of a balance between precision and recall. Improving recall requires predicting the start and end of a sequence while precision favours restricting predictions to when $t - 1 = 1$. This effect is seen in our baseline model which scored 79% on precision and 13% on recall after 5-fold cross validation.

It is also seen in the linear SVM models which is less impacted by cases that fall close to the decision boundary as the start of and end of a sequence are likely to be.

From a practical perspective, predicting the start and end of series of events is what is of most interest. In this respect it seems non-linear models may perform better than linear ones with the RBF model achieving a 76% recall score (while maintaining a high 77% precision score), and the the RNN-LSTM model achieving 69% and 70% respectively.

5.2 Future research

Future research into the matter temporal point processes should investigate whether non-linear models are indeed better at detecting both the start of an event process as well as noiser data in which the patterns may not be as obvious.

If non-linear models perform better at these problems, then, given the performance of RNN-LSTM models on a small set of data, they may be prove to perform exceptionally well in learning to model such temporal patterns with larger amounts of data and training. Currently the training time on an average laptop proved to be a bottleneck but with processing speeds constantly improving we may be entering an age where such models become practical in the commercial data science field.

Finally the research touched upon (see Appndices) but not fully investigated the time it takes for models to adapt to new users. Hybrid models or advanced Bayesian models, that balance the observations coming from the user with wider prior knowledge, may be the key to learning the behaviour of individuals. After all the best predictor for a users behaviour, is observing the user themselves.

Appendix A

Appendices

A.1 A. Further detail of experiments

Bibliography

- [1] Jason Brownlee. *8 tactics to combat imbalanced classes in your machine learning dataset*. 2015. URL: <http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>.
- [2] DJ Daley and D Vere-Jones. *An introduction to the theory of point processes*. 2003.
- [3] Nan Du et al. “Time-sensitive recommendations from recurrent user activities”. In: *Advances in Neural Information Processing*. NIPS. 2015.
- [4] Robert Engle and Jeffrey R. Russell. “Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data”. PhD thesis. University of California, 1996.
- [5] J. F. C. Kingman. *Poisson Processes*. 1992.
- [6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing*. NIPS. 2012.
- [7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521 (May 2015).
- [8] Pietro Di Lena, Ken Nagata, and Pierre Baldi. “Deep architectures for protein contact map prediction”. In: *Bioinformatics* 28 (19 Oct. 2012).
- [9] Emotech Ltd. *Your robot with personality*. URL: <https://www.heyolly.com/>.
- [10] Léon Bottou Martin Arjovsky Soumith Chintala. “Wasserstein GAN”. PhD thesis. Facebook AI Research, 2017.
- [11] Nick Mclure. *Tensorflow Cookbook*. 2016. URL: https://github.com/nfmcclure/tensorflow_cookbook/blob/master/04_Support_Vector_Machines/04_Working_with_Kernels/04_svm_kernels.ipynb.
- [12] Christopher Olah. *Understanding LSTMs*. 2017. URL: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning internal representations by error propagation”. In: *Parallel distributed processing: explorations in the microstructure of cognition*. Cambridge, MA: Association for Computational Linguistics, 1986, pp. 1631–1642.
- [14] Richard Socher et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, WA: Association for Computational Linguistics, 2013, pp. 1631–1642.
- [15] Vladimir N. Vapnik. *The nature of statistical learning theory*. 1995.
- [16] Shuai Xiao, Mehrdad Farajtabar, and Xiaojing Ye. “Wasserstein Learning of Deep Generative Point Process Models”. PhD thesis. Georgia Institute of Technology, 2017.