Modeling temporal patterns of user behaviour using machine learning

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 MSc. Data Science

in the

Department of Computer Science
University College London

This report is submitted as part requirement for the MSc Data Science at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

September 2017

University College London

Abstract

Faculty Name
Department of Computer Science

MSc. Data Science

Modeling temporal patterns of user behaviour using machine learning

by Badrul ALOM

While the area of customer recommendations has received a lot of attention in recent years, predicting user behaviour, particularly the timing of their actions, has not had as much focus. The modelling of temporal processes representing human behaviour can be applied across many industries. In this research we examine different methods for predicting whether a user is interested in listening to music based on past listening history. We examine several models: Beta-Binomial, logistic regression, SVM with both a linear and RBF kernel, and a deep-learning RNN-LSTM model. In addition we have a baseline model in which the prediction at time t was assumed to be the same as the previous period, $t-1$. We found that the Linear SVM model achieves the highest precision, while the logistic regression and baseline models achieved the highest recall. Across both metrics, our baseline model performs the best overall. This may be an issue with the framing of the question and/or the nature of the dataset. Some additional analysis performed towards the end in which we sought to predict the start of a play sequence instead, showed that all models performed poorly, indicating a need for more research in this space. We also found that precision and recall posed a trade-off, with only the RBF-SVM model found to give a good balance on both.

We conclude our research with things to consider when modelling temporal point processes and future directions for this research.

Chapter 1

Introduction

Within the field of recommender systems, the area of predicting *what* a user would be interested in has received extensive attention in recent years, but *when* they would be interested in it, less so. The modelling of temporal patterns is useful across industries. For instance the times at 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 modelling the temporal behaviour of the system is important in predicting the occurrence of the next event.

In this research we look at how we can model the temporal behaviour of users, in order to help gauge their interest in an event at current time t, conditional on their history, h. More formally this is known as a temporal point process and we examine the existing methods for modelling such problems as well as recent experimentations with applying deep learning to the problem.

We compare various classical linear and non-linear techniques, as well as a baseline model in which we assume the event at time t matches the event in the prior period, time $t-1$. Our methods are as follows:

- Beta-Binomial - a simple application of Bayesian Inference
- Logistic Regression
- SVM with a Linear Kernel
- SVM with an RBF Kernel
- Recurrent Neural Network with LSTM

One challenge in modelling events is the sparsity of the data. Depending on your interval range, events can occur 1 in every 10 intervals, 100 intervals, or 1000 intervals. This leads to an imbalanced data set, which has to be dealt with in order to ensure effective modelling.

Another challenge is the scale of the data. Temporal data is often collected these days through computer log files. Such automatic collection means the amount of data available in a temporal point process problem is likely to be high. The scalability of any method to large amounts of data is therefore also of importance.

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 opportune time. It could then also be extended for other activities.

The goal was to 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 [17].

1.2 Structure of the report

In the next chapter we perform a brief review of existing methods of modelling temporal point processes. In Chapter 3 we detail the design of the experiments performed. Chapter 4 then presents the results of the preliminary analysis that helped shape the experimental design, before presenting the main results themselves and discussing each model in turn. Chapter 5 then concludes the report with a summary of insights and suggestions for how the research could be progressed.

1.3 Code

The full set of code used in this research can be found at:
https://github.com/BadrulAlom/EventPrediction

Chapter 2

Literature Review

Analysing event data 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? Within literature the problem is referred to as event prediction, sequence prediction, or temporal point process modelling.

2.1 Modelling temporal point processes

When modelled as a temporal point process, the data can be represented as a sequence of fixed period intervals in one of three ways: as an ordered list of event times $T_i, ...T_n$, as inter-event times $U_i, ...U_n$ where $U_i = T_i - T_{i-1}$, or as a counting process where $N(t)$ is a count of the number of events occurring before time t [2].

Of these, inter-event time is the most commonly used, for example in the times between financial transactions [9]. An example of the counting process is the scoring of goals in soccer [11]. 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.

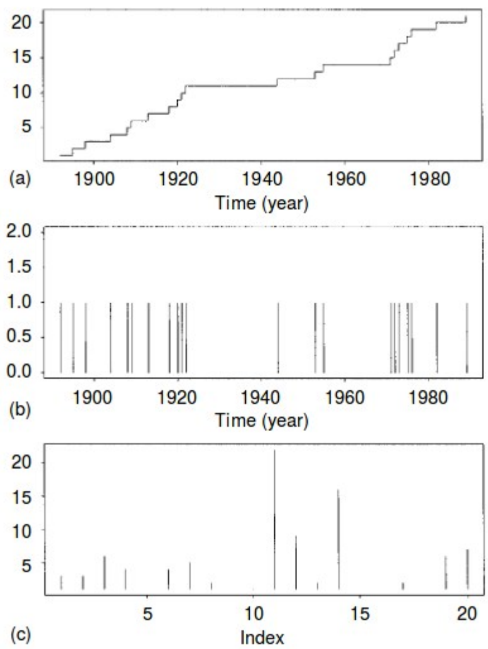

FIGURE 2.1: Three different representations of the same point-process a) event count b) date of occurence c) inter-event time

2.1.1 Poisson Process

The Poisson process [12] is one of the most widely used counting processes. It is a simple point process in which the probability of an event at time t is assumed to be independent from all other times, and where the probability function follows a Poisson distribution with parameter λ.

The Poisson process has been used to model:

- the number of car accidents at a site or in an area [27];
- the location of users in a wireless network [4]
- customer purchases [8]
- jumps in asset prices [13]

2.1.2 Hawkes Process

In the Poisson process the probability at time t is dependent on time t only. In other types of point process models the probability is dependent on periods prior to t as well, such as 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 a non-negative kernel function.

The Hawkes process is particularly useful in systems with rapid changes to rate of events and is used a lot in finance to model the intensity of the arrival of trades [10]

Conditional Intensity Function

The probability function used in a temporal point process is more formally known as a conditional intensity function. It represents the infinitesimal rate at which events are expected to occur around a particular time t, conditional on the prior history of the point process prior to time t.

If the conditional intensity remains constant over time it is referred to as a homogeneous or stationary point process [7]. If however it can vary with time it is inhomogeneous.

Note that the logit function used in generalized linear model, can also be thought of a conditional intensity function and has been used to develop sophisticated models such as [3] and [22].

However, as noted by Wass et. al [28], point process models using a conditional intensity function often 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 in point process models.

2.2 Deep Learning

In recent years deep learning has demonstrated the power to learn hierarchical non-linear patterns on large-scale datasets [15] 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 [14], natural language processing [24], and protein structure prediction [16].

However it has not been applied to temporal point processes until recently with Xiao et. al [28] 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 Wassertein-Distance [19] performed better than Maximum

Likihood Estimate (MLE) which they remarked "may suffer from mode dropping or get stuck in an inferior local minimum".

Their findings showed that while parametric point process models work better with problems where a parametetric form exists, with real world data a GAN model with Wasserterin-Distance outperforms all other models.

2.2.1 Recurrent Neural Networks (RNN)

RNNs 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 [23] 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. 2.2).

$$
\begin{tikzcd}
x_1 \arrow[d, "A"'] & x_2 \arrow[d, "A"'] & x_3 \arrow[d, "A"'] \\
h_1 \arrow[r, "B"] \arrow[d, "C"'] & h_2 \arrow[r, "B"] \arrow[d, "C"'] & h_3 \arrow[d, "C"'] \\
y_1 & y_2 & y_3
\end{tikzcd}
$$

FIGURE 2.2: RNN with 3 timesteps

LSTM

RNN models can employ different methods for the propogration of the hidden state over time. One such well known method is Long Short-Term memory (LSTM) [21]. 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.

Malhotra [18] demonstrates that LSTMs with 2 layers can learn higher level temporal patterns without prior knowledge of the pattern duration, while more recently [29] attempts have been made to combine concepts of temporal point processes with that of an RNN. Here the authors viewed the conditional intensity function of a point process as a non-linear mapping between the predicted transient occurrence intensity of events with different

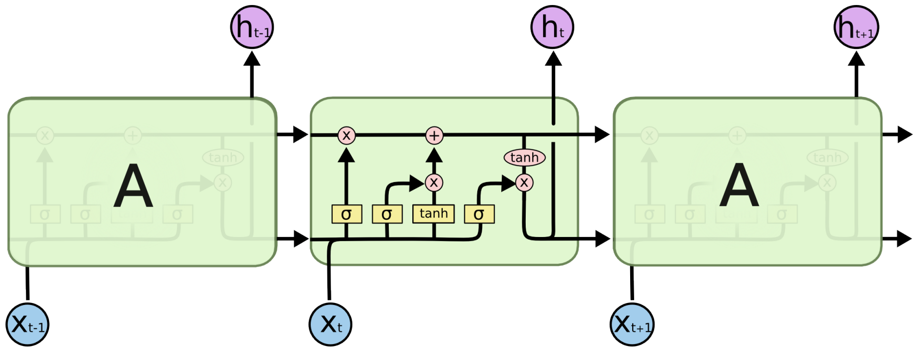

FIGURE 2.3: LSTM

types, and the model input information of event participators, event profile and the system history.

They model this non-linear mapping by utilizing two RNNs: one for modelling the time-series data and associated features and a second to model long-range dependency over history with arbitrary time intervals; specifically a sequence of event type and the time since the previous event. In this way they can capture both the temporal based patterns, as well as the non-temporal, event-correlations which approximates to a conditional intensity function based on inter-event time.

The favoured evaluation metrics in their research were precision, recall, f1 score, and a confusion matrix, with a logistic regression model used as a baseline to compare against.

2.3 Summary

While not an exhaustive review of techniques we have seen some of the ways in which point processes can be modelled using traditional conditional intensity based models, as well as more recent areas of experimentation using deep learning. In the next chapter we layout the approach and methods we wish to explore in the context of this research.

Chapter 3

Experimental Design

In this chapter we describe the different methods that were assessed. The methods, in order of gradually increasing sophistication, are as follows:

1. Baseline model
2. Beta-Binomial model
3. Binary Logistic Regression
4. Linear SVM Classifier
5. RBF 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 in the next section, followed by an explanation each method, and the evaluation criteria for assessing the methods in the sections that follow.

3.1 Data preparation

3.1.1 Data origination

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 a user started playing 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).

3.1.2 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 SqlLite3 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, against which all timestamps in our main dataset were mapped to. n was chosen to be a period of 30 minutes.

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.

3.1.3 Feature selection

The features that were chosen were based on the preliminary analysis (see next chapter) and consisted of time-series and non-time series features. The time-series features were binary, representing play (1) or non-play (0) events at $t, t-1, t-2, t-3, t-4, t-5, t-12hrs, t-23.5hrs, t-24hrs, t-24.5hrs, t-1wk, t-2wks, t-3wks$, and $t-4wks$.

$t-1$ 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 emphasis around the -24 hour mark due to the daily patterns observed in the preliminary analysis.

The non-time lag features were binary features representing the day of the week (isMon, isTue etc.) and the number of hours away from 5pm in either direction - so a timestamp at 4pm and 6pm would both be 1. Again this was based on the observations in the preliminary analysis of 5pm being a peak listening hour.

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 and 5-fold cross-validation employed.

Data Imbalance

A 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 working data set, 361,081 (8.6%) were play events and 91.4% were non-play events. This can result in models that achieve a high accuracy score by following either 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 [5] including restricting input data, having a weighted loss function, and using recall as an evaluation measure. We used recall as well as weighting. Both class weights and sample weights were employed in the Scikit-learn models. The class weights were inversely proportional to play & non-play frequencies in the input data, while the sample weights were a hyper-parameter. In the RNN model only sample weighting was used.

The effect of class weighting is to encourage the model to predict play events. The effect of the sample weighting is to encourage the model to predict play events while minimizing the number of false-positives.

3.3 Methods

3.3.1 Baseline Model

Our baseline model will be to assume that the prediction at time t is the same as the outcome in the prior period, $t-1$. As music listening events tend to be clustered (people listen to music in batches) the accuracy of the baseline model will 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 Beta-Binomial Model

The Beta-Binomial model is a simple form of Bayesian Inference. Conceptually it assumes listening habits follow a weekly pattern and can therefore be summarized as the probability of a play event, at any time period within a week.

It therefore divides a week into 336 half-hourly timeslots (30 min timeslots x 336 = 1 week) and builds up a probability table for each timeslot based on the frequency of plays of the entire population. This forms the prior probability of the Beta-Binomial distribution. This prior probability is then adapted for each user based on their play history at time t. If the user a is a new user, then their history would be empty and as such their probability of listening to music would simply be based on the prior probability derived from the overall population. Fig 3.1 shows the calculation of the prior for the first 2.5 hours of a Sunday (represented as day-hour-halfhour format).

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

The mean and var come from the population data, and a and b are derived from these as is usual with a Beta distribution.

If a user does have a play history, given time t, then this history can be used in a likelihood function. The likelihood function represents the probability of a user listening to music, and we define this as a binomial distribution, where k is the number of plays they have had 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. For which the parameters 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. To do this 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 $\beta = \alpha\left(\frac{1}{\mu} - 1\right)$.

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

An important point to note in all this is how to determine the start and end of a period in which we count up the users listening habits, significantly impacts the model. For example, it would be natural to take the very first song play of a user in our dataset as the date they started listening to music, and start counting the timeslots they listened in from that point forward. But where do we stop counting? The simple approach is to say that the very last record we have of a song play by that user in our dataset represents the end of the counting period. But as fig. 3.2 demonstrates, a users listening profile may not remain constant over time – they may for instance stop using the music service for many years and then listen again one more time before they stop altogether.

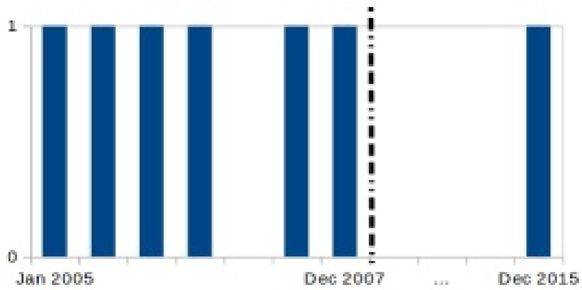

FIGURE 3.2

The consequence of this is that the probability of listening at a given timeslot in week, decreases substantially if we are counting the large gaps as well. The alternative would be to somehow determine for each user, what counts as a 'typical' listening period and use that to form our prior. This was not considered practical however within this research and therefore the simplistic rule for deriving our prior had to be used. We see in the results chapter that this results in very low probabilities. Despite this, the impact ought to be offset by using a lower probability threshold at which we determine something to be a play event, as we do using a ROC curve in the results chapter.

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 $T_t = 1$ and non-play events $T_t = 0$, with t representing a time period of a fixed interval of 30 minute chunks.

t	t1	t2	t3	t4	t5	t10	t12hrs	t23_5hrs	t24hrs	t24_5hrs	t1wk	t2wks	t
1	1	0	0	0	0	0	1	0	0	1	0	0	0
1	1	1	0	0	0	0	1	0	0	0	1	0	0

FIGURE 3.3: Example of times-series data

We seek to model the probability of an event in the current time period t, given the history of events: $p(Y_t = 1 \mid Y_h)$. Our binary logistic model is therefore defined as:

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

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

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.

In addition we will be using an L2 regularizer term, $+1/2w^T w$, to prevent over-fitting:

3.3.4 Linear SVM Classifier

SVM models work by determining a separation plane between classes based on the support vectors - the data points closes to the decision boundary. A linear SVM regression model performs this through the Epsilon Intensive loss function [26]. 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 RBF SVM Classifier

Here we use an RBF kernel in our SVM model. An RBF kernel is a popular method for modelling non-linear decision boundaries. The model is stated as:

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

where x, x' are any two feature vectors (i.e. rows of input) and

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

i.e. we compare each row of x, with every other row of x as demonstrated by the code shown in fig. 3.4 [25].

```
function K = kernel (X,Z,gamma),
% RBF kernel for data in X and Z. X may be training data and Z
                                                the test data
% For training we need kernel between X and X
% data points are row vectors in X and Z matrices
[m1 n] = size(X)
[m2 n] = size(Z)
for i = 1:m1,
  for j = 1:m2,
    K(i,j) = exp(-gamma*norm(X(i,:)-Z(j,:))^2);
  end
end
```

FIGURE 3.4

Note that actual implementations of this make use of more computationally efficient method [20]. The restated method has a hyper-parameter γ that takes the place of $2\sigma^2$. Nevertheless the Scikit-Learn implementation of the RBF kernel has a fit time complexity of more than quadratic to the number of rows of data, making it hard to scale to datasets with more than 10,000 rows. [6].

RBF Hyper-Parameters

The RBF-SVM model requires two parameters to be defined, γ and C. As explained by [1], gamma is a parameter of the RBF kernel itself and can be thought of as the 'spread' of the kernel and therefore the decision region. When γ is low, the 'curve' of the decision boundary is very low and thus the decision region is very broad. When γ is high, the 'curve' of the decision boundary is high, which creates islands of decision-boundaries around data points. C is a parameter of the SVM optimization and is the penalty for misclassifying a data point. When C is small, the classifier is okay with misclassified data points (high bias, low variance). When C is large, the classifier is heavily penalized for misclassified data and therefore bends over backwards avoid any misclassified data points (low bias, high variance).

3.3.6 RNN-LSTM model

The construction of the RNN model requires a number of hyper-parameters. These are as follows:

- Time-steps: How many time steps to use (= batch depth)
  - Learning rate: Learning rate of backprop algorithm
- Hidden units: Number of units per hidden layer
- Layers: Number of hidden layers
- SampleWeighting: Weighting to apply in the cost function for labels that are a play event
- User iteration: How many random users to select for training)
- SamplesPerUser: How many mini-batches to select for each user
- BatchRows: How many random periods to select in each mini-batch (=batch height)
- Batch iterations: How many iterations to perform on one batch

Batch shape

We provide the RNN model with a sequence of historical data at times $t - 1..t - n$ with $n > 1$). This is fed into the RNN as separate time steps in forward temporal order (earliest time-steps are processed first). The hidden state would propagate information forward at each time step, until it was used to predict the outcome at time t.

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

1. The features dimension here only contains $t - 1$ and no other feature.
2. The time-steps are 'unrolled' within Tensorflow and fed into the LSTM. In this research we experiment with different lengths of time-steps, with 48 (half-hour periods) representing a day, 336 for a week, and 672 for 2 weeks.

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.

Samples and iterations

A challenge in testing the RNN model was to balance the number of users being samples from, with the number of samples to take from each user, and the number iterations to then do on each sample. A distinction between batch rows and samples per user was required for computational reasons to reduce the demand on RAM per mini-batch, vs. feeding in one large mini-batch per user.

Cost function and weighting

Both the logistic loss and a weighted softmax cross entropy loss function were tried in the model (the latter requires converting the output label into one-hot encoding format). Logistic loss was found to be the more effective of the two cost function and hence used for our results.

Secondly due to the imbalanced data, adding weighting to the costs of a Play event was found to be crucial in getting good results. The key snippet of code for the cost function is given below:

```
_logits = RNN(x, weights, biases,n_steps)
_prob = tf.sigmoid(_logits)
_weightstf.add(1,tf.multiply( _
tf.cast(tf.equal(y,1),'int32'),n_weighting))

_logloss =
tf.losses.log_loss(predictions=_prob, _ labels=y,epsilon=0.00001, _
weights=_weights)

_cost = tf.reduce_mean(_logloss)
```

3.4 Evaluation criteria

Deciding on an appropriate evaluation measure requires careful consideration to the costs attached to different predictions. In our case the cost of suggesting music when the user is likely not interested is higher than not playing music when they are likely to be interested.

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 the 'Play' events predicted were correct. However it does not distinguish between models that predict a play event only on 'safe bets', such as when t-1 was also a play event, vs. those that are attempting to guess the start of a play sequence. For this we turn to recall.

Recall is defined as:

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

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

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 simply assumes $t = t - 1$, a Bayesian model which builds up a weekly profile of listening habits for each user, Logistic and SVM models that apply classical machine learning techniques to our time-series problem, and finally a deep learning model in the form of an RNN-LSTM where we seek to utilize its ability 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 together with insights on each model.

Chapter 4

Results

We begin our discussion of the results with preliminary analysis of the data. Here we seek to understand some of the overarching patterns in 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 (fig. 4.1).

FIGURE 4.1: 5-5.30pm is peak listening time

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

At an aggregate level therefore one can get good accuracy by simply anticipating music demand to peak at 5pm. However if we select two users

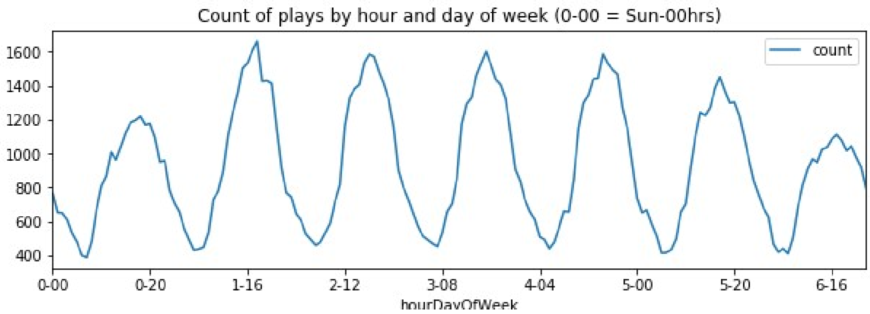

FIGURE 4.2: Most popular times to listen to music across all users

at random (fig. 4.3), we see that these daily patters are not as strongly discernible. As we see later this is likely impacting the Bayesian Inference model as it begins with using the population distribution as the prior.

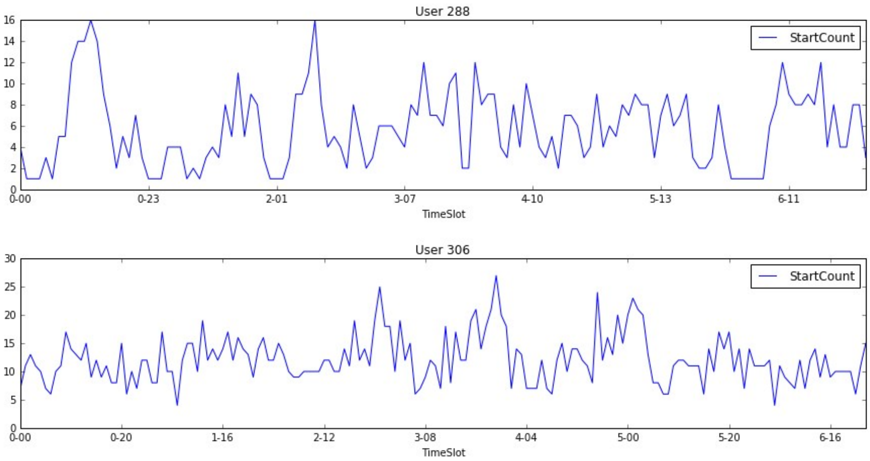

FIGURE 4.3: Most popular times to listen to music by individual user

4.1.2 Outlier analysis

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.

Fig. 4.4 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. 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.

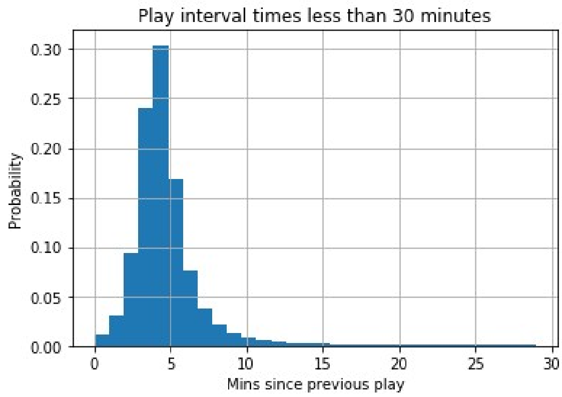

FIGURE 4.4

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.

Fig. 4.5 shows the histogram for plays greater than a day.

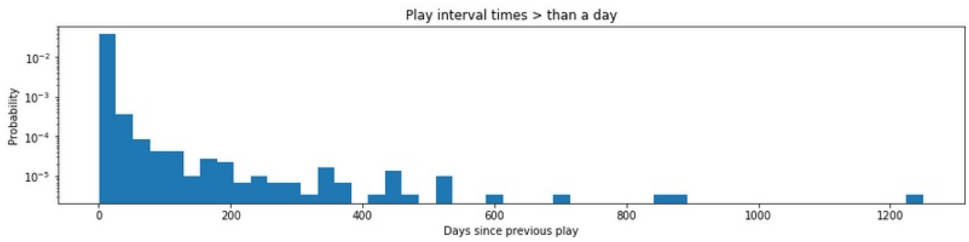

FIGURE 4.5

As one would expect the probability of someone playing music decreases the longer in time they go without playing music. For any music recommender system being able to predict whether a user is likely to listen to music in 1 day, 200 days, or 1200 days, would allow for different strategies for user retention.

4.1.3 Time-series analysis

Here we examine our data once it has been transformed into a binary sequence of play events (1) and non-play 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. Fig. 4.6 shows that the 76% of Plays, also had a play in $t-1$. However this rule also captures 2.2% of non-plays events.

	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 4.6

The 23.8% of Plays that did not have a play in the prior period are harder to predict, yet are of more interest as they represent the beginning of a listening period.

Given the daily patterns we have seen, it might be reasonable to assume that $t - 24hrs$ may help us to determine the start of an session. However analysis shows that in only 34% of play events was there also a play event in 24 hours prior.

What both of these results tell us is that fairly high precision score of around 76% ought ot be possible purely based on $t - 1$ but going above this will be a lot harder.

4.2 Main results

4.2.1 Summary

Fig. 4.7 shows the results from across all experiments after 5-fold cross validation.

Model	Train rows	Precision	Recall
Baseline	500k	76%	80%
Logistic Regression (t-1 only)	500k	76%	80%
Bayesian Inference	500k	40%	13%
Logistic Regression	500k	66%	78%
Linear SVM Classification	500k	80%	73%
RBF SVM Classification	100k	77%	76%
RBF SVM Classification (t-1 only)	100k	76%	76%
RNN-LSTM (t-1 only)	50k	60%	75%

FIGURE 4.7: Summary of results

We see that none of the models score better than the Baseline model. The logistic regression model is able to match the Baseline model once we restrict the input to $t - 1$.

It may be that $t = t - 1$ is the only pattern that is consistently important across users, with all other time-lags having too much variance across users for them to be useful.

In a similar vein, the Bayesian Inference model performed poorly on both measures, suggesting that that the difference between the population and what is observed at an individual level differs too much for the prior probability to be accurate for any one individual user, something we saw some indications for in our preliminary analysis.

Our remaining models exhibit the the tension between good accuracy and good recall, with only the RBF SVM model demonstrating the ability to reach a balance score across both. The Logistic regression and RNN models perform below average on precision but well on recall, while the Linear SVM model scores better on precision. We will discuss each one model in turn next.

4.3 Beta-Binomial Model

We outline here how the model was derived in order to aid understanding of its performance.

We can plot the prior probability for any given time period, as shown in fig. 4.8.

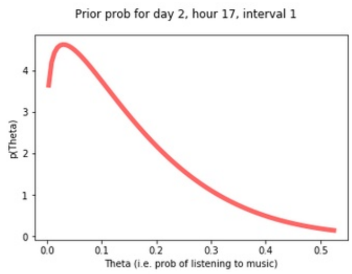

FIGURE 4.8

Here we see that θ, which represents the prior probability of listening to music in the specified timeslot, is likely to be less than 0.1. Notice that this timeslot is for hour 17 (i.e. 5pm) which, from our preliminary analysis, we know is the peak listening time at an aggregate level. The fact that the probability is so low for this timeslot would likely be due to the presence of lots of 5pm periods in which music was not listened to at the individual user level. For instance - if our dataset contains large gaps between weeks in which music was listened to.

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 (fig. 4.9). From this 0.4 was selected as the optimal threshold at which to determine a Play event across all users.

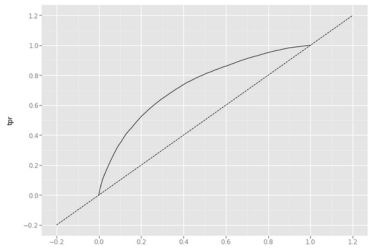

FIGURE 4.9: ROC curve showing 0.4 as the optimal threshold

The results from the model are shown in table 4.1. We see that the recall and precision of play events (as denoted by 1) is very low. Clearly the approach of building up a weekly profile based on the population then updating it with individual observations is not very effective. This does not however rule some other form of Bayesian modelling such as Bayesian Logistic Regression.

	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

TABLE 4.1: Beta-Binomial Model Results

4.4 Logistic regression analysis

For logistic regression we are able to examine the coefficients to understand how the model is making use of the input data.

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

TABLE 4.2

Table 4.2 shows that $t - 1$ was by far the most important feature and its importance crowds out the other features. Interestingly $t - 23.5$ is the second strongest time-lag and more significant than t-24hrs. As t-23.5 would be 24 hours prior to $t-1$, it suggests that having a consistent $t-1$ is more important than simply knowing what t was 24 hours prior; which we know from our preliminary analysis is not very effective anyway.

The non-time lag features seem to have very little effect. We see this in the forward stepwise regression chart in fig. 4.10 which shows that the inclusion of additional features barely impacts the results.

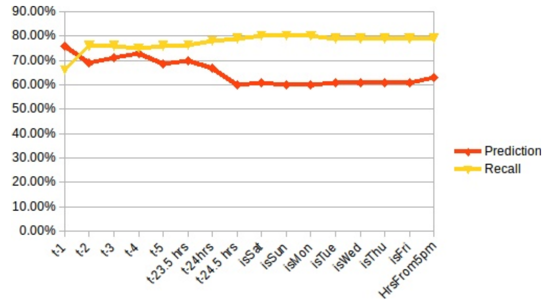

FIGURE 4.10: Stepwise regression, adding in fields from left to right

4.5 SVM analysis

That the Linear SVM performs well on precision but not recall may be due to it ignoring probabilities that fall within the margin of the decision boundary during optimization, such as the start of a play sequence or ad-hoc plays by a user that do not fit their regular listening pattern. Hence the model is more likely to only predict a play event when the probability is sufficiently high.

The RBF model on the other hand 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.

In order to determine optimal $gamma$ and C values, a grid search was carried out with precision and recall of the test dataset plotted on a heat map. In order to get through the many iterations of the data, the input data was reduced to 30,000 training samples. Fig 4.11 shows the final heat map, after multiple iterations to home in on the optimal values, with precision on the left and recall on the right. The optimal values are deemed to be 0.6 for C and 0.76 for Gamma, although as the charts show there is some scope for movement around those values.

The final results for the RBF model (precision of 77%, recall of 76%) was about the same when only $t-1$ was provided as an input, suggesting the

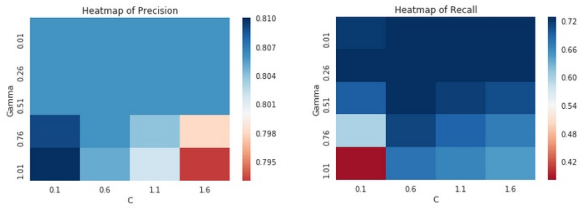

FIGURE 4.11: Heatmap of hyper-parameter search. Dark blue shade = better precision and recall

RBF model is likely approximating to the $t = t - 1$ rule used in the baseline model.

Finally, while we managed to get as high as 100,000 rows of our training in our training, the computational complexity of the RBF kernel meant it was not practical to go any higher thereby limiting the scalability of this approach and our ability to potentially obtain higher scores than what we already had.

4.6 RNN-LSTM analysis

The RNN model had a large number of hyper-parameters to search through resulting in significantly more time required to train than the other models. Fig 4.12 shows details of a few of these runs, including the model that gave the best set of results (run num. 4). Key differences between the runs are shaded in grey.

Run number	**3**	**4**	**5**	**6**	**7**	**8**
Features list	t-1	t-1	t-1	t-1	t-1	t-1
Cost function	logloss	logloss	logloss	logloss	logloss	logloss
User iteration	30	60	60	60	60	60
Samples per user	5	5	5	5	3	3
Batch size	50	50	50	50	50	50
Total input rows	7,500	15,000	15,000	15,000	9,000	9,000
Batch Iterations	100	50	50	50	50	100
Time steps	48	48	48	48	48	336
Hidden Units	1	1	1	2	2	1
Layers	1	1	2	1	2	1
Learning rate	0.05	0.05	0.05	0.05	0.05	0.05
Imbalance data weighting	8	8	8	8	8	8
Test Results:						
Test Precision (sample = 50)	73.00%	63.00%	68.00%	64.00%	55.00%	70.00%
Test Recall (sample = 50)	73.0%	75.0%	72.0%	71.0%	77.0%	72.0%
Test Precision (sample = 50)	63.00%	59.80%	67.10%	70.00%	62.40%	70.00%
Test Recall (sample = 50)	73.0%	77.0%	68.8%	69.8%	72.5%	72.5%
Test Precision (sample = 50)	70.00%	65.00%	51.00%	62.50%	62.90%	54.80%
Test Recall (sample = 50)	59.6%	74.0%	77.0%	74.4%	75.3%	74.1%
Test Precision (sample = 50)	67.90%	63.00%	70.50%	61.10%	68.00%	30.00%
Test Recall (sample = 50)	72.9%	76.6%	71.7%	72.6%	73.8%	82.0%
Test Precision (sample = 50)	65.00%	61.50%	56.60%	71.00%	69.80%	70.00%
Test Recall (sample = 50)	74.0%	75.5%	77.0%	70.9%	71.6%	71.0%
Test Precision (sample = 50)	**67.78%**	**62.46%**	**62.64%**	**65.72%**	**63.62%**	**58.96%**
Test Recall (sample = 50)	**70.50%**	**75.62%**	**73.30%**	**71.74%**	**74.04%**	**74.32%**

FIGURE 4.12: Example of hyper-parameter search. Key settings shaded.

In total 50+ experiments were performed with different hyper-parameter settings. The following observations were made based on the results of all experiments.

Early stopping

Continued training of models eventually led to performance decrease, with predictions becoming either all play events or all non-play events depending on the weighting used. Possible measures to counteract this may be to use drop-outs, a form of regularization in neural networks. However this was not investigated. Instead models were trained up-to around 15,000 rows of data.

More layers or units did not improve performance

Moving up from 1 hidden unit and 1 hidden layer did not result in performance improvement, and performance started to drop if it increased too high (10 units or 3+ layers). It appears that this was due to the model over-fitting the data.

The sample weighting was critical

The amount of weighting required to deal with the data imbalance varies depending on how many rows of training data there are, and the level of imbalance. In our case 8 was found to be the level at which precision and recall remained high across a training set of 15,000 rows. Lower than this and all predictions soon converge to non-play events; higher than this they all become play events (and hence low precision, high recall).

Going beyond a 24hour timestep did not improve performance

Moving beyond a time-step of 48 (i.e. 24 hours) to 336 (1 week) or more led to a decrease in performance. This matches what we saw in the logistic regression model where periods beyond the 24 hour period had little impact in predicting outcome.

4.7 Summary

We found that the Linear SVM model achieves the highest precision, while the logistic regression and baseline models achieved the highest recall. Overall across both metrics, our baseline model performs the best. Furthermore from our examination of the coefficients and experiments with the RNN model, it did not appear that time-lags beyond 24 hours were a useful predictor. Indeed logistic regression performed best, and on-par with the baseline, when $t - 1$ was the only input feature. These results suggests that additional features make it harder for the models to attribute the right level of importance to $t - 1$.

This may be an issue with the framing of the question and/or the nature of the dataset. As an addendum to our initial research, a quick analysis was performed on first play event data only. This dataset was constructed by removing all rows where there was a play event in periods $t - 1$ to $t - 5$, leaving behind rows where the user had not listened to music in the past 2.5

hours. We were interested in seeing how are our models performed without any tuning.

We found that all models performed poorly including the baseline (as would be expected), with precision and recall both at 0% for all but the logistic regression model which had a precision of 0.7% and a recall of 0.3%.

From a scalability perspective both the logistic model and linear SVM performed well on 500k rows of training data, RBF SVM was capped at 100k rows, and the RNN around 15-20k rows due to much more hyper-parameter testing. Using the logistic model as a guideline was also found to be helpful in determining parameters for the RNN.

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 products, electricity usage, or the demands on a public transport system. The ability to accurately model these patterns is therefore of great significance to industry. Traditional temporal point processes modelling requires making assumptions about the data, such as the feature engineering done in this research, in order to build a prediction model and struggle with capturing highly non-linear patterns in a scalable manner.

We applied a range of techniques to the task of modelling temporal point processes and used precision and recall as our evaluation measure. Our Baseline model which assumed event at time t = event at time $t-1$ performed the best overall, with a precision of 76% and a recall of 80%. As music listening is typically long periods of non-play events, followed by consecutive periods of play events, it is easy to achieve high scores using this heuristic. Notable results were the Linear SVM achieving a higher precision score of 80% making it suitable for cases where precision is of importance, and the RBF model which demonstrated an ability to evenly balance precision and recall.

The results could raise the questions of whether rather than trying predict all play events, it is better to try and predict the probability n steps into the future, or the inter-event time, the start of a play sequence instead. This last idea is something which was briefly tried at the end of our experiments. All of the models performed poorly in this case.

Finally we experimented with an RNN-LSTM model. As our literature review showed, the application of deep learning to temporal point processes is still in its infancy but shows promising signs. Our own research is inconclusive though positive enough to warrant further research. As mentioned, careful thought needs to be given to the problem definition in order to perform a reasonable assessment.

Our take-away for the modelling of temporal processes in general, is to consider the following:

- How much does the data differ at the macro vs. micro level?
- Are events singular events or occur in clumps?
- How balanced is the dataset?
- How well do simple heuristics explain the temporal patterns?

5.2 Future research

There are several directions one could follow to take this research forward.

1. Attempt to predict the start of a play sequence
2. Attempt to predict events n steps into the future, where $t-1$ to $t-5$ are not available
3. Investigate the variance between the temporal patterns of individual users and that of the population and models that can better deal with these
4. Evaluation of more sophisticated RNN structures such as that described in our literature review [29].
5. Evaluation of advanced Bayesian models such as Bayesian Logistic regression
6. Application of Gaussian Point processes as way of getting around RBF scalability limitations
7. Investigation into how quickly models can learn to model the behaviour of new users

Bibliography

[1] Chris Albon. *SVC Parameters When Using RBF Kernel*. 2016. URL: https://chrisalbon.com/machine-learning/svc_parameters_using_rbf_kernel.html (visited on 08/29/2017).

[2] P. Andersen and O Børgan. *Statistical Models Based on Counting Processes*. Springer, 1993.

[3] Adrian Baddeley et al. "Logistic regression for spatial Gibbs point processes". In: *Biometrika* 101.2 (2014), pp. 377–392.

[4] Bartłomiej Błaszczyszyn, Mohamed Kadhem Karray, and H Paul Keeler. "Wireless networks appear Poissonian due to strong shadowing". In: *IEEE Transactions on Wireless Communications* 14.8 (2015), pp. 4379–4390.

[5] 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/.

[6] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), p. 27.

[7] Sung Nok Chiu, Dietrich Stoyan, and Wilfrid S. Kendall. *Stochastic Geometry and Its Applications*. John Wiley and Sons, 2013.

[8] A.S.C. Ehrenberg. *The pattern of consumer purchases*. London: Charles Griffiin and Company, Ltd., 1959.

[9] Robert Engle and Jeffrey R. Russell. "Autoregressive Conditional Duration: A New Model for Irregularly Spaced Transaction Data". PhD thesis. University of California, 1996.

[10] Stephen Hardiman, Nicolas Bercot, and Jean-Philippe Bouchaud. "Critical reflexivity in financial markets: a Hawkes process analysis". In: (2013).

[11] A. Heuer, C. Müller, and O. Rubner. "Soccer: Is scoring goals a predictable Poissonian process?" In: *EPL (Europhysics Letters)* 89 (Feb. 2010), p. 38007. DOI: 10.1209/0295-5075/89/38007. arXiv: 1002.0797 [physics.data-an].

[12] J. F. C. Kingman. *Poisson Processes*. Clarendon Press, 1992.

[13] Michael Kirch and Wolfgang J Runggaldier. "Efficient hedging when asset prices follow a geometric Poisson process with unknown intensities". In: *SIAM Journal on Control and Optimization* 43.4 (2004), pp. 1174–1195.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing*. NIPS. 2012.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (May 2015).

[16] Pietro Di Lena, Ken Nagata, and Pierre Baldi. "Deep architectures for protein contact map prediction". In: *Bioinformatics* 28 (19 Oct. 2012).

[17] Emotech Ltd. *Your robot with personality*. URL: https://www.heyolly.com/.

[18] Pankaj Malhotra et al. "Long short term memory networks for anomaly detection in time series". In: *Proceedings*. Presses universitaires de Louvain. 2015, p. 89.

[19] Léon Bottou Martin Arjovsky Soumith Chintala. "Wasserstein GAN". PhD thesis. Facebook AI Research, 2017.

[20] 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.

[21] Christopher Olah. *Understanding LSTMs*. 2017. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[22] Tuomas Rajala. "A note on Bayesian logistic regression for spatial exponential family Gibbs point processes". In: *arXiv preprint arXiv:1411.0539* (2014).

[23] 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.

[24] 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.

[25] KP Soman, R Loganathan, and V Ajay. *Machine learning with SVM and other kernel methods*. PHI Learning Pvt. Ltd., 2009.

[26] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, 1995.

[27] Emilo C Venezian. "The distribution of automobile accidents—Are relativities stable over time?" In: *Insurance: Mathematics and Economics* 12.1 (1993), pp. 68–69.

[28] Shuai Xiao, Mehrdad Farajtabar, and Xiaojing Ye. "Wasserstein Learning of Deep Generative Point Process Models". PhD thesis. Georgia Institute of Technology, 2017.

[29] Shuai Xiao et al. "Modeling the Intensity Function of Point Process Via Recurrent Neural Networks." In: *AAAI*. 2017, pp. 1597–1603.