UNIVERSITY COLLEGE LONDON

DOCTORAL THESIS

Modeling Temporal point processes using Recurrent Neural Nets

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 9, 2017

Declaration of Authorship

University College London

Abstract

Faculty Name
Department of Computer Science

Master of Science

Modeling Temporal point processes using Recurrent Neural Nets

by Badrul ALOM

tbc

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. At an aggregate level, behaviour may appear to be deterministic, such as the times at which peak rush-hour occurs, but such behaviour is often composed of thousands or millions of individual stochastic processes such as the decisions made by individuals as to whether to leave work at 5pm or continue working a little longer.

While the modeling of aggregate patterns is well understood, these models ofen breakdown when applied to customizing results for individual users. At this level the temporal patterns of an individual combined with the behaviour of the population may be a better predictor of event timing. For instance, sticking with the example above, the times at which a person has lunch during the day may help predict that they will finish work a little later.

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, and whether deep learning is better able to learn these patterns than other methods.

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 right now, based on their listening-event history. One such application of this research would be to allow home audio devices to suggest music toi a user at an ooportune time.

The goal will be evaluate the effectiveness of several different methods, including an LSTM-RNN. The research was guided by Emotech Ltd., a home audio hardware and software company and the creators of Olly [7].

1.2 Data

The dataset being used in this analysis is the LastFM1k dataset, which is freely available oneline and containings 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 users behaviour in order to predict the next item in 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

tbc

Chapter 2

Literature Review

Event prediction is alternatively referred to in literature as sequence predictions and temporal point processes. The problem can be defined mathematically as determining:

$$p(x_t|x_h)$$

where h is the event history sequence, $t - n...t - 1$.

2.1 Point Processes

This is one of the most widely used method for modeling event sequences. A temporal point process [1] is a way of modeling events data with t being a sequence of a fixed period interval with $t_i \epsilon R + and i \epsilon Z +$.

It can be modelled as a series of inter-event times (time until next event) or the number of events occurring in the interval. Examples of point processes are the times between financial transactions [3], and the times at which a customer makes a purchase from an online retailer *** insert ref***.

At its simplest a point process 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.

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.

There are different ways of representing point process data as shown in figure 2.1, with the inter-event time being the most common.

2.1.1 Time-series analysis

2.1.2 Conditional Intensity Function

A conditional intensity function is the key part of modeling point processes [2]. In this method the probability of an event $\lambda(t)$ is derived from a stochastic model such as the Poisson process.

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

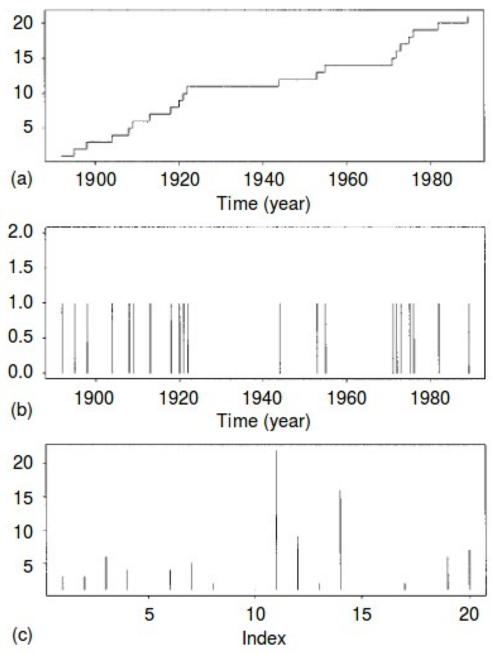

FIGURE 2.1: Three different representations of the same point-process a) cumulative count b) date of occurence c) interval time between floods

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 $\lambda(t) = \mu + \beta \sum_{t_i<t} g(t - t_i) where g is some non-negative kernel function.$

As noted by Wass et. al [13], 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.

2.2 Deep RNN Point Process Models

In recent years deep learning has demonstrated the power to learn hierarchical nonlinear patterns on large-scale datasets [5] 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 [4], natural language processing [11], and protein structure prediction [6].

However it has not been applied to temporal point processes until recently with Xiao et. al [13] 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 (fround truth) sequence amongst the generated ones.

For measuring the loss between a generated and true sequence, the authors found the Wassertein-Distance [8] 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 where as parametric point process models work better with problems where a parametetric form exists, with real world data a GAN model with Wasserterin-Distance outperformed all other models (including an RNN model using MLE). This signals a promising new direction for temporal point process research.

Chapter 3

Preliminary analysis

As a precursor to the methodology discussion in the next chapter, we first present a preliminary analysis of the dataset.

3.0.1 Daily play patterns

Track plays, grouped into 30 min intervals, demonstrate a clear daily pattern with usage hitting the peak at around 5pm and a trough at around 6am.

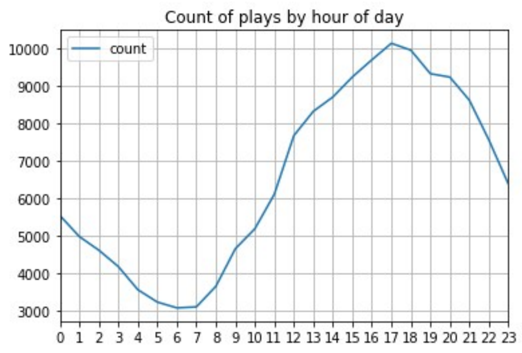

FIGURE 3.1: 5-5.30pm is peak listening time

The peak of 5pm is likely explained by people finishing work at 5pm, however the bothdecline during pre-work hours of 6-8am was unexpected and may be a product of how the data was gathered.

Zooming out to view the pattern across an entire week in figure 3.2, we see that the daily pattern occurs across every day of the week with weekends having a lower total number of plays.

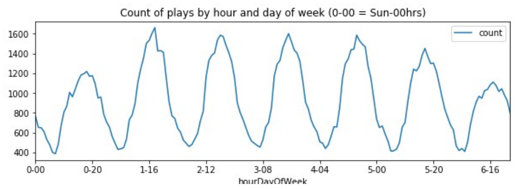

FIGURE 3.2: Most popular times to listen to music across all users

If we then select two random at users we can see how strong these patterns are over a period of week. Figure 3.3 shows that the daily patters are still discernable at the individual user level although they are not as clear cut. This demonstrates why models developed using aggregate level data may not translate well to individual user prediction.

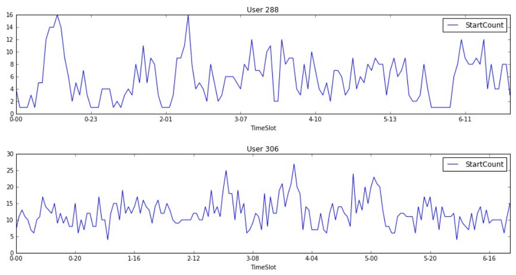

FIGURE 3.3: Most popular times to listen to music by individual user

3.0.2 Unbalanced data

The dataset is highly imbalanced with approximately 8.4

3.0.3 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 tracs was a few seconds suggesting the user skipped tracks.

Figure 3.4 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.

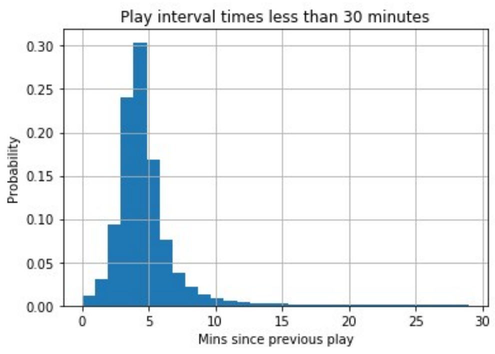

FIGURE 3.4

For our purposes these are include as evidence that the user was interested in playing music at time t.

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.

3.0.4 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.

[***TODO: Replace with scatter plot***]

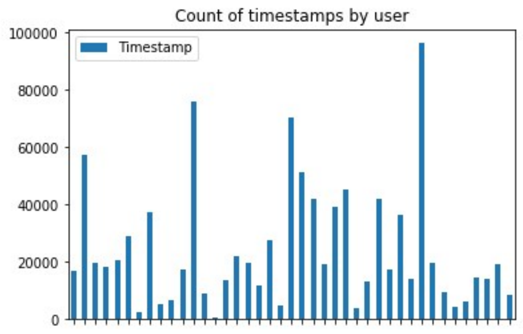

FIGURE 3.5: Total play count by user

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.

3.0.5 Summary

The data follows a strong daily pattern at the aggregate level as well as the individual level, albeit to a lesser extent. Therefore the time of the day or most recent 24 hours is likely to be a strong predictor. Weekends have been sene to exhibit differences to weekdays so the day of the week is also a good feature. The data is high unbalanced and consists of a low amount of plays to non-plays, which are on averages around 5-6 minutes in duration.

Chapter 4

Methodology

From our preliminary analysis we have seen that listening habits, at an aggregate level, follow a strong daily usage pattern. However at an individual user level these patterns are not as strong. In this section we discuss different methods of increasing complexity, that we wish to asses.

The methods are:

- Beta-Binomial model
- Epsilon intensive loss function
- RBF Regression
- Recurrent Neural Networks

We start with discussing the data preparation that was performed in order to perform the analysis as well as the evaluation criteria. We then discuss each of the methods.

4.1 Data Preparation

The analysis was carried out in Python (via Jupyter notebooks) running on Ubuntu. The raw data consisted of timestamp of when a song was played and a user ID. These were loaded as-is into a SqlLite3 database. The methods themselves utilized a mixture of scikit learn, Tensorflow, and GPFlow.

UserIDs were then 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.

The data was then transformed into time series data with the following features generated: PeriodID, UserID, HrsFrom5pm, isSun, isMon, isTue, isWed, isThu, isFri, isSat, t, t1, t2, t3, t4, t5, t10, t12hrs, t24hrs, t1wk, t2wks, t3wks, t4wks.

HrsFrom5pm was chosen as based on the preliminary analysis 5pm appeared to be the peak listening time for the population as whole (see next chapter) and was calcuated as the absolute number of hours, in either direction, from 6pm. Days of the week were codified into a one-hot vector notation, and t ... t4wks represent whether or not a user listened to music in the current period, in t minus 1 period, ... t minus 12 hours ago, all the way through to t minus 4 weeks ago.

In this way, for each play or non-play we capture a snapshot of the history up to that point. 4 weeks was chosen as the history length as it was felt to be an adequate amount of time to capture the signals that would help predict a play event.

4.2 Test dataset

Two types of test data were selected. The first was a hidden periods test set, formed through random selection of random months from the training dataset and holding them back.

For simplicity future periods after this month were kept in the training dataset, although it is acknowleged that this does not reflect the real world in which future periods would not be available.

The second method was to hold back the entire history of 10 randomly selected users. This would allow us to assess how well the model performs with new users. Note however that even here data earlier than 1 months history was excluded in order for the time-series based methods to have enough history for the time series base methods to work. Dealing with a lack of information on new users is known as the cold-start problem and in this case the prior probabilities as gathered in the BFreq model would be one way of estimating the liklihood of listening to music. We will touch upon this briefly in the next chapter.

The output field for all but the RNN model was a one dimensional vector consisting of $y = \epsilon(0, 1)$.

In the case of the RNN model one-hot encoding was used such that $y\epsilon([0, 1], [1, 0])$. This was purely a Tensorflow design choice (as it allows the model to be easily extended to a multi-class use case should the need arise).

4.3 Evaluation critera

The main evaluation criteria across all models was precision, recall, and f1-score. These are common metrics used in information retrieval.

Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p). A true positive in this case is predicting that user plays or does not play music and being correct.

$$P = \frac{T_p}{T_p + F_p}$$

Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n). For example if we predicted 100 plays correctly but there were in fact 110 plays in the dataset, then recall would be 100/110 = 91

$$R = \frac{T_p}{T_p + F_n}$$

A high precision score does not necessarily mean a high recall score, and often an improved precision can mean a lower recall (making fewer guesses but with a higher degree of accuracy). We place this in the real world context and ask what we value. For a home audio device, suggesting music when the user does not want to listen to music would carry a higher cost than not suggesting music when a user does not wants to listen to music. Therefore precision is more important than recall. Of course a high score in both is ideal, and this is captured by the F1 score.

F1 is defined as the harmonic mean of precision and recall.

$$F1 = 2\frac{P \times R}{P + R}$$

Several models were evaluated. As the goal of the research was to evaluate the effectiveness of different methods, extensive feature engineering beyond what has been described was not performed.

Random salection of test periods was perormed each time any of the results were run, while test users were selected at the outset and kept fixed throughout.

4.4 Beta-Binomial model (BetaBin)

The Beta-Binomial model is one of the most simple Bayesian inference models, and has a tractable solution.

We define our liklihood, the probability of a user listening to music, 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..

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

Here our period, or rather timeslot, is the set of half hourly intervals within an entire week (so 24*2*7=336 timeslots). This is different to the time-series approach we adopt for the latter methods. Here the frequency of plays and non-plays are dervied at the userID, timeslot level, where timeslot is a string concatenation of weekday, hour of day, and start minute of period.

Our prior is Beta distribution of θ:

$$p(\theta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$

where

$$\mathrm{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

A the Beta distribution is a conjugate prior to the Binomial the formula 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 prior parameters, α and β, are dervied from the training set, with an estimate for each half-hourly time period.

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

The remaining models adopted a time-series approach. Let $E = 1$ represent a Play event, and $E = 0$ a non-play event. t is the half-hourly time period we are predicting for, and h represent the half-hourly history. We therefore seek to calculate the probability of an event in the currenty time period, given the history of events, $p(E_t \mid E_h)$, for each individual user.

4.5 Binary Logistic Regression (LogReg)

Here our model is: $p(E_t = 1|E_h) = \sigma(w'x + b)$

where w' is a weight matrix transpose, and x is our input features, and b is a constant.

4.6 Linear SVM Regression

A linear SVM regression model is characterized by the Epsilon Intesive loss function [12]. By modifying the loss function to ignore errors that are within a certain margin, ϵm helps prevent overfitting.

Our objective becomes to minimize:

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

4.7 Non-Linear RBF Regression

Here we extend a regression model to include a Gaussian RBF kernel:

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

where RBF is:

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

Note that the actual implementation in Tesnorflow made use of the fast computation of the RBF kernel as defined by McClure [9]. The restated method has a hyperparameter γ that takes the place of $2\sigma^2$ which requires calibrating.

The Kernel allows us to solve for non-linear problems by transforming our x values to a high dimensional space. Note that performing such calculations is computationally intensive and as such mini-batch training will be performed with a batch size of 2000 and 5 iterations of the dataset. (Rows of more than 5000 tended to cause memory errors, and iterations beyond 5 showed minimal or no decrease in loss)

4.8 Feature engineering and Regularization

Feature engineering is the process of constructing and selecting what one considers to be useful predictors in from a dataset. A high level of feature engineering can lead to muchh faster model traning times as uncessary features have been man manually pruned ahead of time. Regularization is the idea of letting the modle learn the which features are important through addin a penalty term to the cosrt function which coerces the optimziation process to favour fewer stronger weights than many weak ones. In this dataset we have discussed some feature engineering performed in the data preparation stage in addition to employing regularization. In the logistic regression model, L2 regulzarization is used.

4.9 Recurrent Neural Networks

Setting up an RNN model in Tensorflow requires careful attention to how shapes are fed i

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. Whereas a Feed-Forward network has input nodes, hidden layers, and an output layer, with data flowing in one direction only; RNNs allow for the hidden state from timestep of the neural net to be an input into the next.

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

FIGURE 4.1: RNN with 3 timesteps

An LSTM is an extension of an RNN model in which the hidden layer which is transmitted across time steps is further divided into four layers that interact in a way as to learn what information to retain, and what information can be thrown away [10].

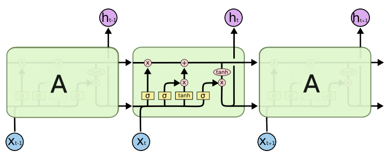

FIGURE 4.2: LSTM

Setting up an RNN model in Tensorflow requires careful attention to how shapes are fed i

Chapter 5

Experiments

In this chapter we present the results of the experiments conducted and discus the refinements made to the models as a result of the experiments. We start with a summary of the best results obtained by each method before going into a more in-depth evaluation in the subsequent sections.

5.1 Results summary

Looking at the F1-score we see that the RNN model performed the best overall with the logistic regression model close behind. Looking solely at new users we can say that the RNN model makes correct guesses 72% of the time, and being well balanced between being right its guesses (precision), and guessing all of the time right events (recall).

We see a somewhat stronger performance on recall, relative to logistic regression on the hidden periods assessment. This can be considered the performance under a 'busuines-as-usual' scenario beyond the cold-start phase.

However as discussed in the RNN results section, the results are not as clear cut when we try to separate out the feature engineering element of our model from the LSTM specific element. Recall that our dataset consisted of rows of data, each one containing information from time-lags t-1, t-2 etc.

In theory an LSTM ought to be able to learn such features by itself based on its architecture. As part of the testing it was found that RNN recall reduces from 72% to around 3% when time lags 1-5 are not directly encoded. This indicates a clear failure to pick up the most important of time-lags t-1. Speculation as to why is discussed in more detail in the RNN results section.

5.2 Adaptability to new users

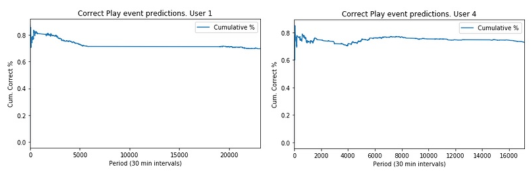

FIGURE 5.1

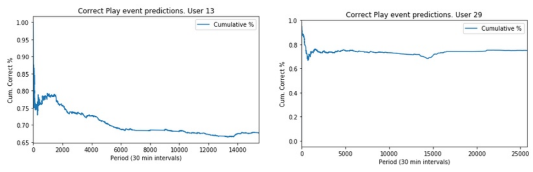

FIGURE 5.2

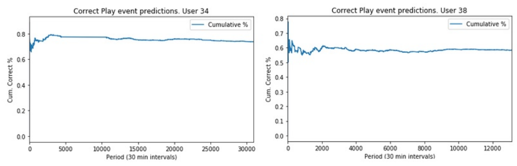

FIGURE 5.3

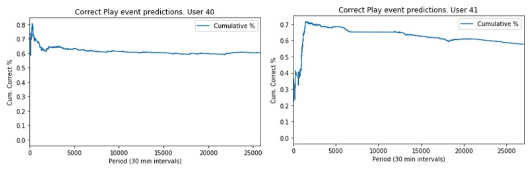

FIGURE 5.4

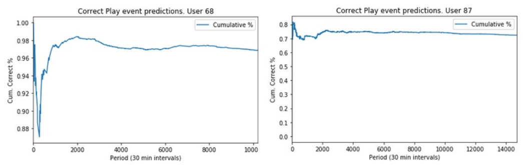

FIGURE 5.5

5.3 Beta-Binomial model

5.4 Logistic Regression

5.5 RNN-LSTM

Of all the models, the RNN required the most effort to set up correctly. We provide details here to help others implementing a similar model.

5.5.1 Input Shape

In order to utilize the learning capabilities of an LSTM in Tensorflow, the input, x, must be of shape (rows,depth, cols) where rows are separate rows of training data, the depth is the time-steps that will be unrolled by the RNN, and cols are the number of features. It is important to note that when the data is unrolled, time step t for all batches are processed in one block, before t-1 , t-2 etc.

Constructung the 3-d shape often requires building them up in slices. It is highly advisable to pre-allocate the 3d array when doing this rather than extending the array on each iteration.

5.5.2 Imbalanced data

The experiments looked at the difference between feeding the RNN data rows with time lag features (t-1 .. t-5, and t-24) explictly provided, vs. excluding them and expecting the model to determine the optimal time lags for itself. A significant decrease in performance was found whehn they were excluded with recall for play events being extremely low. Further analysis indicated the recall decreases the more it is trained suggesting that the model was learnig to priotize the non-play events at the expense of play-events due to the imbalance in the data (non-play events make up 90

A weighted cost function was used to emphase play events as seen below, which led to some improvement.

Chapter 6

Summary

6.1 Conclusion

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:
`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.
If you want to completely hide the links, you can use:
`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.
If you want to have obvious links in the PDF but not the printed text, use:
`\hypersetup{colorlinks=false}`.

Bibliography

[1] DJ Daley and D Vere-Jones. *An introduction to the theory of point processes*. 2003.

[2] Nan Du et al. “Time-sensitive recommendations from recurrent user activities”. In: *Advances in Neural Information Processing*. NIPS. 2015.

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

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

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

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

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

[8] Léon Bottou Martin Arjovsky Soumith Chintala. “Wasserstein GAN”. PhD thesis. Facebook AI Research, 2017.

[9] 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.

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

[11] 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.

[12] Vladimir N. Vapnik. *The nature of statistical learning theory*. 1995.

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