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 5, 2017

Declaration of Authorship

I, Badrul ALOM, declare that this thesis titled, "Modeling Temporal point processes using Recurrent Neural Nets" 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

Modeling Temporal point processes using Recurrent Neural Nets

by Badrul ALOM

tbc

Chapter 1

Introduction

Event sequences are common in areas such as retail, finance, and utilities. For example, the times at which a customer makes a purchase from an online retailer, the time between financial transactions in the stock market, and the times at which utility service customers make high use of gas and electricity. These are a particular category of event sequences in which the temporal factor could be said to be a key dimension in predicting the next occurence. In such cases, understanding and predicting user behaviors, based on purely the temporal aspect of their occurence, are of great practical, economic, and societal interest.

1.1 Motivation

The impetus for this research was to estimate the liklihood of a person being interested in listening to music in the current time-period based only on their listening history. The raw data is a series of timestamps denoting when songs were played. The implicit assumption is that a persons playlist history contains a temporal pattern such as a combination of daily and weekly schedule, that can be modelled. This concept can also be applied to other areas where a temporal pattern is thought to exist at an individual user level, such as the repeat purchase of household products, or the sleeping and eating habits of a person.

The objective of the research is to evaluate the effectiveness of several different techniques for determining the probability of a user listening to music in a given period. One such application of this would be home audio devices which could anticipate when a user would like to listen to music and play without user intervention.

The research was guided by Emotech Ltd. the creators of Olly [7].

1.2 Point Processes

One way of modeling the problem is as series of events and non-events known as a temporal point process. This has a rich history of methods as outlined in the literature review.

1.3 The dataset

The dataset being used in this analysis is the LastFM1k dataset containing the listening history of a thousand LastFM listeners.

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

1.4 Structure of the report

tbc

Chapter 2

Literature Review

Within literature the most common way of tackling the problem is through Point Processes. More recently both Gaussian Processes and Deep Learning have been applied.

2.1 Point Processes

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 Conditional Intensity Function

A conditional intensity function is the most popular method for 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.

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 $\lambda(t) = \mu + \beta \sum_{t_i<t} g(t-t_i) where i s s o m e n o n - n e g a t i v e k e r n e l f u n c t i o n.$

However, 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

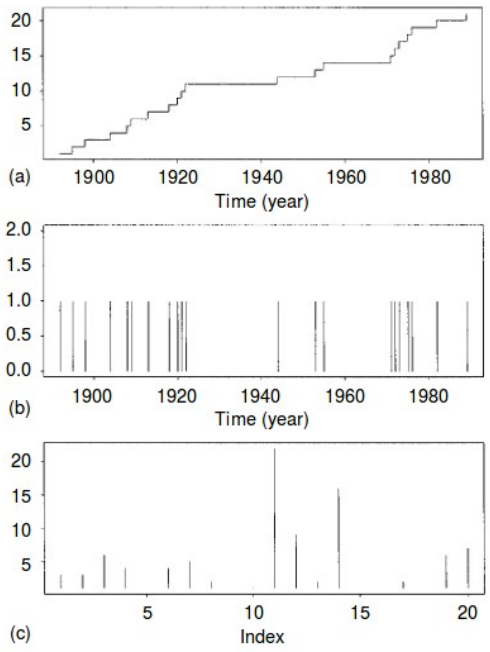

FIGURE 2.1: Three different representations of the same point-process a) cumulative count b) date of occurence c) interval time between floods

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 non-linear 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

Data Exploration

Before we discuss methodology it is useful to first ground ourselves in what the data looks like. Here we provide some preliminary analysis that was performed that helped shape the design decisions in the next chapter.

3.0.1 Basic Statistics

Due to technical and time constraints the analysis was carried out on apprixmately 10 percent of the full LastFM 1k dataset.

3.0.2 Daily play patterns

When we count the track plays by hour of the day we observe 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

While the peak of 5pm is easily explained by 9-5 workers finishing their jobs, it is strange that the pre-work hours of 6-8am has the lowest listening time out of the entire 24hr day.

Zooming out to view the pattern across an entire week we see that the daily pattern is occurs across every day of the week

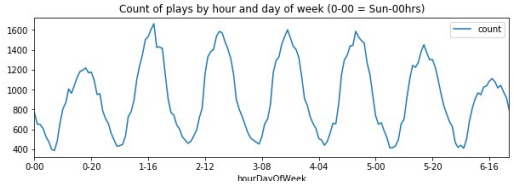

FIGURE 3.2: Most popular times to listen to music across all users

A sample of users also showed

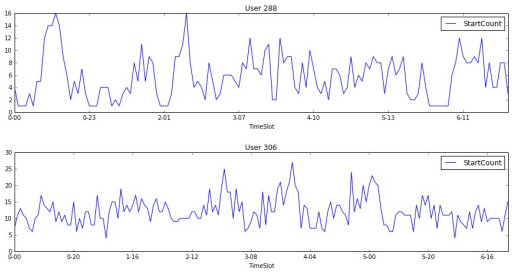

FIGURE 3.3: Most popular times to listen to music by individual user

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.

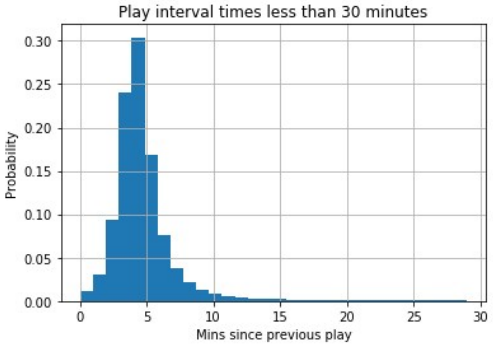

FIGURE 3.4

However for our purposes we can include these as evidence that the user was interested in playing music at time t.

We can also assume that the song plays are not i.i.d, 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 Outlier analysis

An analysis of plays by user reveals a high amount of variance between users on how many tracks are played.

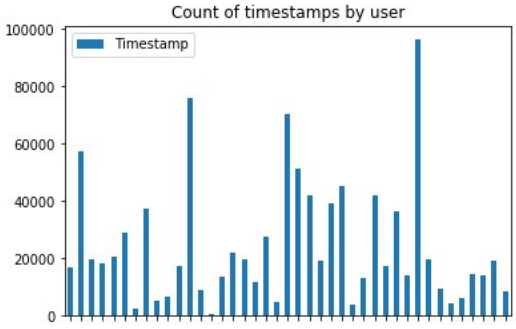

FIGURE 3.5

3.0.5 Conclusion

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 the different methods , of increasing complexity, that were evaluated.

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 table was defined of n minute intervals to which all timestamps 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. As can be imagiend this increased the size of the dataset signifciantly from <> rows to <> rows.

This was necessary in order to evaluate the success of the models in predicting when users would like to listen to music vs. when they would not. From here the data was modelled in two different ways.

THe second method of modeling the data retains the temporal aspect and is inline with times-series approaches. The data was structure into the following features: PeriodID, UserID, HrsFrom5pm, isSun, isMon, isTue, isWed, isThu, isFri, isSat, t, t1, t2, t3, t4, t5, t10, t12hrs, t24hrs, t1wk, t2wks, t3wks, t4wks.

The first two are self-explanatory. HrsFrom65m was chosen as based on the preliminary analysis 6pm 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.

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

Chapter 5

Methodology

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.

5.1 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.

5.1.1 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.

5.1.2 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:

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

5.1.3 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)

5.1.4 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.

5.1.5 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. 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.

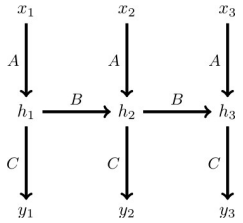

FIGURE 5.1: RNN with 3 timesteps

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].

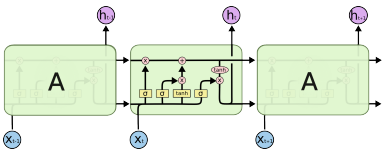

FIGURE 5.2: LSTM

Chapter 6

Evaluation

In this chapter we present the results of the evaluation starting with the best result obtained from each model from different hyper-parameter testing.

The performance of each model is then discussed further in the sections that follow.

6.1 Results summary

Within the context of a home audio decide, the cost of mistakenly suggesting a user wants to listen to music is higher than the cost of not suggesting at all. We therefore focus our results on the predictions where a 'play' event was predicted by the model and examine both the precision and recall scores for both the hidden periods for existing users as well as new users.

All models required a number of runs to try and determine the best hyper-parameters, as dicusssed in the next few chapters, and it may be the case that hyper-parameter settings exist for each model that would provide a boost in performance.

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.

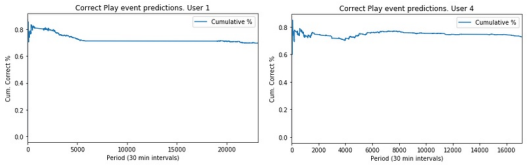

FIGURE 6.1

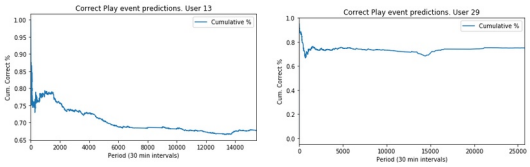

FIGURE 6.2

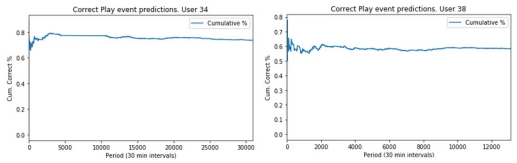

FIGURE 6.3

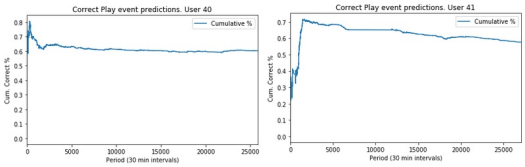

FIGURE 6.4

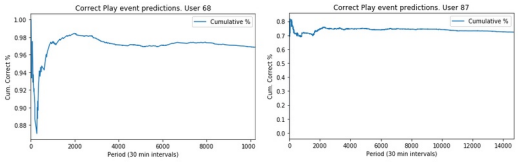

FIGURE 6.5

6.2 Adaptability to new users

6.3 Beta-Binomial model

6.4 Logistic Regression

6.5 RNN1

Chapter 7

Chapter Title Here

7.1 Main Section 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam ultricies lacinia euismod. Nam tempus risus in dolor rhoncus in interdum enim tincidunt. Donec vel nunc neque. In condimentum ullamcorper quam non consequat. Fusce sagittis tempor feugiat. Fusce magna erat, molestie eu convallis ut, tempus sed arcu. Quisque molestie, ante a tincidunt ullamcorper, sapien enim dignissim lacus, in semper nibh erat lobortis purus. Integer dapibus ligula ac risus convallis pellentesque.

7.1.1 Subsection 1

Nunc posuere quam at lectus tristique eu ultrices augue venenatis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam erat volutpat. Vivamus sodales tortor eget quam adipiscing in vulputate ante ullamcorper. Sed eros ante, lacinia et sollicitudin et, aliquam sit amet augue. In hac habitasse platea dictumst.

7.1.2 Subsection 2

Morbi rutrum odio eget arcu adipiscing sodales. Aenean et purus a est pulvinar pellentesque. Cras in elit neque, quis varius elit. Phasellus fringilla, nibh eu tempus venenatis, dolor elit posuere quam, quis adipiscing urna leo nec orci. Sed nec nulla auctor odio aliquet consequat. Ut nec nulla in ante ullamcorper aliquam at sed dolor. Phasellus fermentum magna in augue gravida cursus. Cras sed pretium lorem. Pellentesque eget ornare odio. Proin accumsan, massa viverra cursus pharetra, ipsum nisi lobortis velit, a malesuada dolor lorem eu neque.

7.2 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

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.