UNIVERSITY OF CALIFORNIA
RIVERSIDE

Modeling Crime Data Using Point Processes

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Gaurav Jhaveri

June 2017

Thesis Committee:

Dr. Christian Shelton, Chairperson
Dr. Eamonn Keogh
Dr. Vagelis Papalexakis

The Thesis of Gaurav Jhaveri is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, without whose help, I would not have been here. I would like to thank my parents, for making me who I am today. Lastly, I would like to thank my friends, I would not have made it through without them.

To my parents for all the support.

ABSTRACT OF THE THESIS

Modeling Crime Data Using Point Processes

by

Gaurav Jhaveri

Master of Science, Graduate Program in Computer Science
University of California, Riverside, June 2017
Dr. Christian Shelton, Chairperson

Analyzing criminal activity is an extremely challenging task. The sheer volume of incidents make it a time-consuming and man-power intensive task for police departments to detect any patterns of crime manually.

In this thesis, we compare two point-process based models — the Piecewise-Constant Conditional Intensity Model and Multiplicative Forests — in their ability to analyze crime. Specifically, we compare the performance of both models in learning temporal dependencies in crime incidents. To accomplish this, we make use of 15 years of crime data provided by the City of Chicago. Additionally, we also look at the seasonality of crime, the types of crime most indicative of other crimes, and other consistently-recurring themes. We see promising results that could help authorities with predictive policing and better tackling criminal activity.

Contents

List of Figures

Chapter 1

Introduction

An undeniable fact of human society is the omnipresence of crime in every settlement. The sheer volume of criminal data can make its analysis seem like an insurmountable task. After all, every incident comes with its own bag of goodies — Date, time, location, category, offender, victim, to name a few. These incidents have complex underlying dynamics that can be difficult, if not impossible, to learn manually. The goal of this thesis is to use models based on marked point processes to identify temporal dependencies in crime data and, consequently, give some insight into crime patterns. The hope is that the information obtained from these models will assist in predictive policing and better allocation of police resources.

In Chapter 2 we look at the previous work done on this subject in the fields of machine learning, statistics and criminology. In Chapter 3 we explain the models that are being compared, including providing notation and visual aids to better understand the models. In Chapter 4 we go over the model-specific implementation details and go over

areas with potential for performance improvements. Chapter 5 covers the experimental setup, including pre-processing the dataset all the way up to the results. We use the log likelihood scores to compare both models. Finally, in chapter 6 and 7 we go over the potential for future work and draw some conclusions based on our results.

Chapter 2

Previous Work

Every facet of this problem has been tackled by researchers from a diverse range of disciplines. Indeed, if we are to tackle the problem of crime analysis and prediction then it must involve experts from criminology, computer science, statistics and several other fields. Thus, before moving on to the original work of this thesis, a multi-disciplinary literature review will help us get a better sense of what parts of this issue have already been addressed.

Point processes are extremely well-studied. While we will be reviewing point processes to provide context to our models, there are several papers that more rigorously cover the theory behind them. There are several applications of point processes that look at the spread of activities both in time and space that are applicable to the task at hand. Farajtabar et al. (2014) [4] study information diffusion in the context of social networks by developing a temporal point process model. However, this makes the assumption that the underlying network is known, which is an unrealistic scenario in crime data. Lian et. al (2015) [5] develop a multitask point process model. However, they do not explore the

correlation between event types. Instead, they focus on the variability between subjects within the same category. The diversity of the population is an important consideration in healthcare analysis but does not help us achieve our goal of a big-picture analysis of criminal activity.

More specifically to our dataset, the spread of crime and other properties exhibited by criminal activity have been explored from several different angles. There's plenty of evidence that suggests that for certain crime-types, such as burglaries, a single incident can trigger several other crimes of a similar nature. Intuitively, this makes sense because the occurrence of a crime indicates vulnerabilities at that location. Burglars are likely to return as this is the location they've done their homework on, making an encore easier, now equipped with knowledge and the experience of a successful burglary. Mohler et al. (2011)[6] use residential burglary data provided by the Los Angeles Police Department to implement a self-exciting point process based on this information of repeat-victimization. This is explained in more detail by Johnson (2008)[7] where he explores two major theories behind the clustering of crime events in space and time. This behavior is not limited to burglaries. More serious types of crime also exhibit these dynamics[8][9]. Linderman & Adams [10] explore latent network structures of gang-related homicides in Chicago. A number of articles by the National Institute of Justice have also tackled crime in Chicago. Block and Block analyzed street gang crime in Chicago [11]. Block also did extensive research [12] in to the seasonality of different types of crime.

Chapter 3

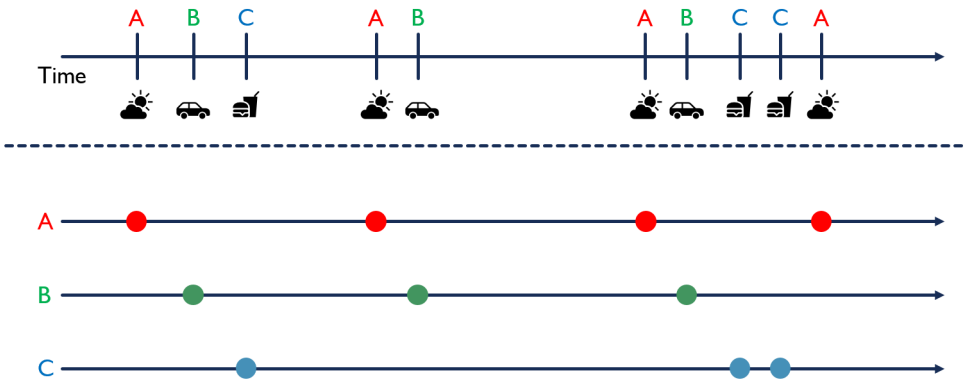
Background

3.1 Point Processes

We can represent events of a certain domain on a timeline. These events can be marked with a time as well a type. That's all a temporal point process is, a list of times of events. The only caveat is that we do not know when and how many of these events will occur. These events have an underlying process linking them together that, if captured, can provide us with valuable insight in to our domain. Let's consider seismic activities as an example. Earthquakes are not isolated incidents and usually a major earthquake leads to several minor earthquakes and aftershocks. If we're able to learn a model of seismic activity then we can better prepare ourselves in the future. Events in a temporal point process may also have additional information attached to them, such as the intensity of an earthquake, the blood pressure of a patient, etc.

Point processes consider each event type individually, specifying that they occur according to a rate function $\lambda(t|h)$ where t is the time period over which the rate is active

Figure 3.1: A timeline (top) split-up into point processes (bottom)



and h is the event history. Figure 3.1 shows a timeline broken up into its constituent point processes.

3.2 Notation

An event sequence is represented as $y = \{(t_i, l_i)\}_{i=1}^{n}$ where t_i is the time when the i^{th} event occurred and l_i is the event type (or label), drawn from a finite set $\mathcal{L}$. The history h at time t is the subset of y with times less than t. Formally, $h(t, y) = \{(t_i, l_i) \mid (t_i, l_i) \in y, t_i \leq t\}$

The likelihood is given by

$$p(x \mid \theta) = \prod_{l \in \mathcal{L}} \prod_{i=1}^{n} \lambda_l(t_i \mid h_i, \theta)^{\mathbb{1}(l_i)} e^{-\Lambda_l(t_i \mid h_i; \theta)} \tag{3.1}$$

where $\lambda_l(t_i \mid h_i, \theta)$ is the conditional intensity function for type l and $\Lambda_l(t \mid h; \theta) = \int_{t(h)}^{t} \lambda_l(\tau \mid h; \theta)\, d\tau$. $\mathbb{1}$ is the indicator function. $\lambda_l(t \mid h; \theta)$ is the expected rate of events of type l at time t given its history h.

3.3 Piecewise-Constant Conditional Intensity Models (PCIMs)

Gunawardana et al. (2011)[1] developed a model for learning temporal dependencies in event streams known as the Piecewise-Constant Conditional Intensity Model (PCIM). In PCIMs the rate functions are constant over intervals. That is, $\lambda_l(t \mid y)$ is piecewise constant in t for all $t > t(y)$. λ takes on values $\{\lambda_{ls}\}$ for $s \in \Sigma_l$, where Σ_l is a discrete set of label-dependent states. The value λ_{ls} is specified by a piecewise constant state function $\sigma_l(t, y)$, so that $\lambda_l(t \mid y) = \lambda_{l\sigma_l(t,y)}$. The state s has all the information about t and y required to calculate $\lambda_l(t \mid y)$.

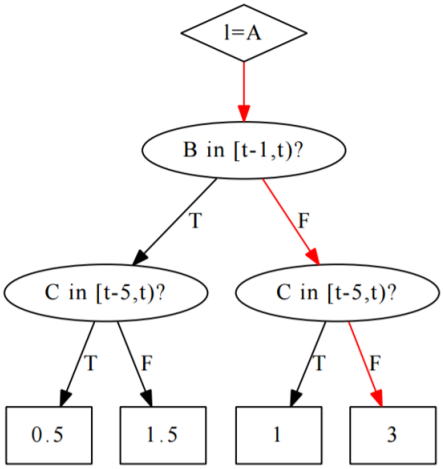
Every state function σ_l can be represented by a decision tree where the leaves represent the states s. The internal nodes are decision nodes that map t and y to one of the child nodes. These mapping functions are piecewise constant in time, so the state function $\sigma_l(t, y)$ represented by a decision tree is also piecewise constant.

We start with the trivial decision tree for every event type l. These trees are then grown in a greedy fashion based on a scoring function which can either be a maximum likelihood score or a maximum a posteriori score. Figure 3.2 shows the decision tree representation of a PCIM.

3.4 Conjoint Piecewise-Constant Conditional Intensity Models (C-PCIMs)

C-PCIMs[2] are an extension of PCIMs. A key difference is that the conditional intensity functions in C-PCIMs take on values from a single set of values shared across

Figure 3.2: A decision tree representing a PCIM. Reprinted from Gunawardana et al. (2011)[1].



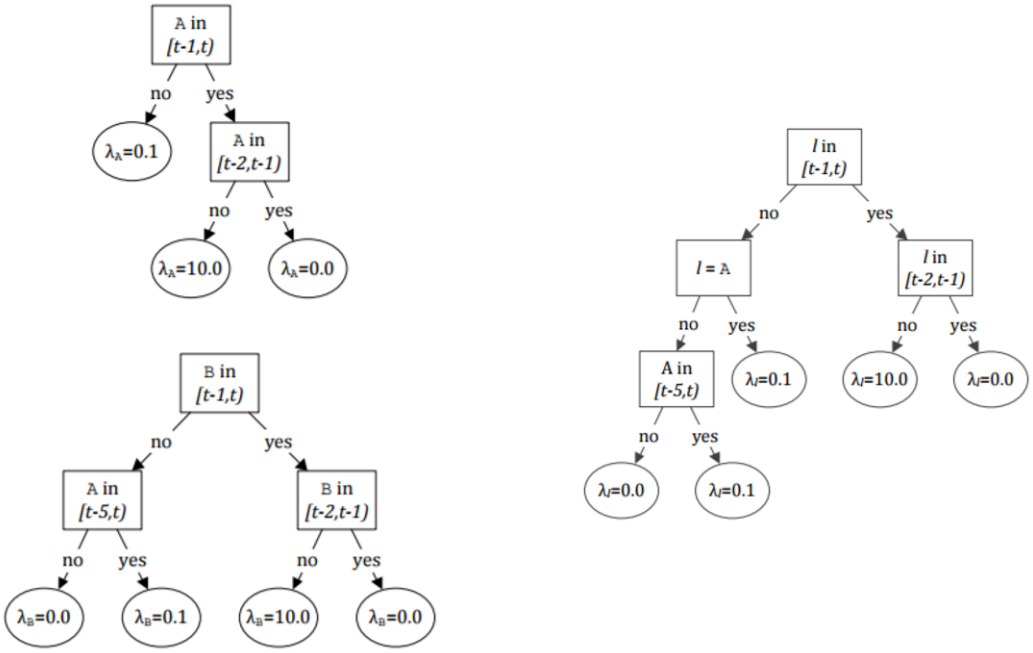
all event types $l \in \mathcal{L}$. In other words, λ now takes on values from $\{\lambda_s\}$ for $s \in \Sigma$ which are shared for all l. Additionally, C-PCIM state functions are also a function of the event type l whose conditional intensity is being evaluated. They are represented as $\sigma(l, t, y)$. So, $\lambda_l(t \mid y) = \lambda_{\sigma(l,t,y)}$. This allows an intensity value λ_s to be shared across conditional intensity functions for different event types at different times and with different histories.

C-PCIMs use a single decision tree for all event types l. The decision nodes contain functions that can depend on t and y, as in PCIMs, but also on the event type l. Figure 3.3 shows an example of a set of PCIM decision trees and an equivalent C-PCIM representation.

3.5 Multiplicative Forest Point Processes (MFPPs)

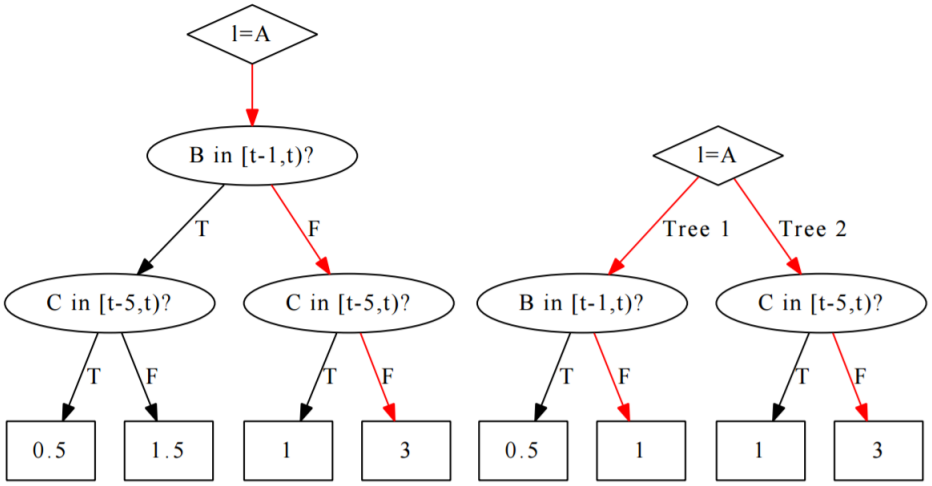
Multiplicative Forests replace decision trees with random forests[3]. Consider a path ρ through a decision tree for every label l. This corresponds to a recursive subpartition resulting in a set Σ_ρ where every $(l, s) \in \Sigma_\rho$ corresponds a leaf intensity $\lambda_{l\rho}$. That is,

Figure 3.3: Decision trees representing a PCIM (left) and its C-PCIM equivalent (right). Reprinted from Parikh et al. (2012)[2].



$\lambda_{ls} = \lambda_{l\rho}$. The partition corresponding to a single intensity is given by the intersection $\sum_\rho = \cap_{j=1}^{k} \sum_{\rho,j}$ of sets corresponding to the active paths through trees $1 \ldots k$. The intensity $\lambda_{l\rho}$ is given by the product of leaf intensities. Figure 3.4 shows an example of a PCIM decision tree and its equivalent Multiplicative Forests representation.

Figure 3.4: Decision tree representing a PCIM (left) and its Multiplicative Forests equivalent (right). The active paths are shown in red. Reprinted from Weiss et al. (2013)[3].



Chapter 4

Implementation

4.1 Learning

For each of the models, we start out with a stump with rate $\lambda = 1.0$. That is, all event sequences and times are mapped to the root node. To perform a split, we score every leaf node against the entire pool of tests to find the best split to make according to a maximum likelihood score. We select the leaf-test combination that gives the largest gain in score. This process is repeated until our stopping criteria is fulfilled.

In the case of Multiplicative Forests, we allow for the possibility of adding a new tree rather than making a split on an existing one.

4.2 PCIMs

We maintain a priority queue consisting of all leaf nodes and the best (according to the previously mentioned scoring function) candidate test to replace each leaf. To make

a split, we simply have to split the node present at the front of the priority queue. Each time a leaf node is split, its children are scored and added to the priority queue.

4.3 MFPPs

The forest maintains a vector of priority queues, one for each tree. However, there were some additional challenges to be dealt with. First, every time we make a split in a tree, every leaf not in the same tree must be rescored. This is because the change in score by a split is determined by how a leaf intersects with leaves from other trees. If the leaf nodes in any of the other trees change, then the previously calculated change in score is no longer valid. This detail alone causes the MFPP code to run considerably slower than the PCIM code. While the PCIM has rescore only 2 leaf nodes after each split, the MFPP must rescore anywhere between 3 leaf nodes to an entire tree consisting of several leaf nodes. There are a few ways to reduce this increase in time, but each of them results in a sub-optimal forest. First, instead of rescoring all the non-belonging nodes at each pass, we can choose to rescore them every p passes. Thus we're making the assumption that if, previously, a test was the best candidate for a leaf, then it must at least be good enough (if not the best) after a new split. Second, we can restrict the number of trees and the number of leaves per tree. Once a tree reaches the maximum number of leaves, it is no longer considered for splitting and as such, does not need to be rescored. Third, we can choose to grow trees in iteratively. This means that once we move away from growing one tree, we cannot add any leaves to it at a later point.

In the original Multiplicative Forests for Point Processes paper by Jeremy Weiss

and David Page[3], modification 2 is their preferred choice of building a multiplicative forest. They compare a PCIM with 100 nodes to a MFPP with 10 trees and 10 leaves in each of those trees.

4.4 Parallelizability

The most time-consuming part of the code in both cases is (re-)scoring the leaf nodes. Depending on the dataset we are dealing with, there could be hundreds of potential tests. However, this also presents an obvious parallelization opportunity. We can divide up the pool of tests in to smaller subsets, one for each core. For a given leaf, each of the cores reports the best test from its assigned subset of tests. We can then compare these tests to get the overall most viable test for a given leaf.

The performance improvement we get from parallelizing our code is tremendous. The speed-up is especially useful for MFPPs, where we might need to (re)score tens of leaves at each iteration. Here, a few minutes saved at each iteration could result in a few hours being saved over the course of a 100+ iterations. The difference is noticeable even in the PCIM code, albeit the nature of the model itself helps in that only 2 leaf nodes need to be scored at each iteration.

Chapter 5

Experiments

In this section we will look at the performance of PCIMs and Multiplicative Forests in modeling crime data.

5.1 Dataset

5.1.1 About

The dataset contains reported incidents of crime that occurred in the City of Chicago from 2001 to 2015. There are over 6 million rows in this dataset, where each row is a single incident consisting of the date and time of the incident, the type of crime, the location (down to the block level) along with additional information. This project only analyzes temporal dependencies and as such, ignores all the other columns except the date-time stamp and the type of crime. In the future work section we will talk about some improvements that can be made, including adding a spatial component.

5.1.2 Processing

The dataset needed some modifications to adapt it to our models. First, the crime type for each incident was converted to an integer for simplicity. There were 28 different types of crimes that needed to be mapped. This allowed us to create a key-value mapping where each key was the crime type and the values were the times at which the incidents had occurred. Second, the date-time stamp for each incident was converted to number of hours elapsed since January 1st, 2001. If there are multiple incidents of the same type occurring at the same time, the model only considers 1 of them and ignores the others. This resulted in nearly 30% of the data being lost in some cases. To fix this, we added some noise between 0 to 59 seconds drawn uniformly at random to each incident. While we ended up with a richer dataset, there was an unexpected downside to how the models were learning with this new data. The models created new states to account for this additional data, states that lasted only for a few seconds. In reality, the incidents were independent from each other, having occurred at the same time, but the models now created several states with gargantuan rates to model dependencies that did not exist. To fix this, we shifted all the tests 10 minutes to the "left," making them ignore any other incidents that have occurred in the immediate past.

5.2 Setup

For both PCIMs and MFPPs, we use a sliding window over the dataset to train the models on 5 successive years and test it against the 6th year.

5.2.1 Hyper-Parameters

Before starting with the experiments, it is important to first ensure that we're performing an apples-to-apples comparison. It is a fairly effortless task to make one model look worse than the other by simply modifying its hyper-parameters in a way that the model is no longer suited to the task.

For each of the models, we use number of splits as the stopping criteria. Initially, we train a PCIM on the first 4 years and cross-validate it with the log-likelihood of the 5th year. We use this to find the optimal number of splits, which is 80. This is the number of splits to use in all the experiments for both, PCIMs and Multiplicative Forests.

Jeremy Weiss and David Page[3] use number of trees and number of splits per tree as hyper-parameters. In our experiments, we will not place any restrictions on the number of trees or the number of leaves in each tree. Instead, the model is free to add a tree to the forest whenever it sees fit. Furthermore, it is also allowed to come back to a previously grown tree if it improves the score.

5.2.2 Test Pool

The learning power of these models is determined by the pool of tests made available to them and is arguably the most important part of the experimental setup. There are 3 types of tests that we have used:

- VarTest — Test if an event is of a certain type. There are a total of 28 types.
- TimeTest — Divide the timeline into ranges, then test if the time is within a certain interval of that range. Repeat for all the ranges. All tests of this category divided the

time up in to 1 year ranges. The intervals used were

  - Test if the event time was in quarter 1 of the year.
  - Test if the event time was in quarter 2 of the year.
  - Test if the event time was in quarter 3 of the year.
  - Test if the event time was in quarter 4 of the year.

- CountTest — Test if more than n events of type l have occurred in $[t_{\text{start}}, t_{\text{end}})$. The time ranges $[t_{\text{start}}, t_{\text{end}})$ used were

  - 1 day
  - 1 week
  - 2 weeks
  - 1 month

  We approximate the average number of events that have occurred in each of these ranges. Let this be denoted by $n_{(\text{avg, t})}$ where t is the range under consideration. Then, the values of n used were

  - $\frac{n_{(\text{avg, t})}}{4}$
  - $\frac{n_{(\text{avg, t})}}{2}$
  - $n_{(\text{avg, t})}$
  - ${n_{(\text{avg, t})}}^2$
  - ${n_{(\text{avg, t})}}^4$

Table 5.1: PCIMs vs. MFPPs

Training Years	**Testing Years**	**PCIM**	**MFPP**
2001-2005	2006	**239917**	238272
2002-2006	2007	225855	**225882**
2003-2007	2008	209534	**210327**
2004-2008	2009	156860	**160126**
2005-2009	2010	**129504**	127871
2006-2010	2011	**103123**	103026
2007-2011	2012	86226.5	**86469.1**
2008-2012	2013	**54203.9**	54164.6
2009-2013	2014	**12636.8**	12345
2010-2014	2015	**362.252**	308.416

Applying these tests across all the crime types and for a variety of time-ranges we get a pool of nearly 200 tests.

5.3 Results

5.3.1 Log Likelihood Comparisons

Table 5.1 shows the log likelihood scores for each of the models. PCIMs outperform MFPPs in 6 out of the 10 experiments.

5.3.2 Observed Trends

There are a few trends that stand out as we observe the trees and forests produced by the models. The most prominent one is that crime is highly seasonal in nature. The spring-summer months have far higher criminal activity than the winter months.

Next, battery, robbery and theft are the most significant indicators of overall rise in crime. This is almost a tautology considering the fact that these were the most frequently

occurring types of crime.

More surprisingly, deceptive practice — which includes fraud, embezzlement, identity theft, identity impersonation — in the past 24–48 hours almost always results in an increase in crime of all types.

There are a few other correlations observed. Motor vehicle thefts and robberies are often linked, an increase in one indicates an increase in the other. Similarly, gambling and narcotics crimes also show a correlation. It is important to note that the model simply shows correlation, and does not imply causation.

Finally, arson and gambling are most strongly affected by seasonal trends, with up to a 6x increase in rate in the spring-summer months.

Chapter 6

Future Work

This thesis only analyzes the temporal dependencies amongst crime types in Chicago. There are several improvements and additions that can be made to make this a more thorough analysis. For starters, including a spatial component would be the most obvious way forward. Crime rates vary significantly by area of the city, with more economically developed areas having low rates, but other sections having much higher rates of crime. For example, in 2013, high crime districts saw 38.9 murders as compared to low crime districts which only saw 2.5 murders per 100,000 people. The models as they currently stand are unable to capture this stark contrast.

The next step would be to see if a model of crime developed in one city performs well when tested in another city. There are a few challenges with this task. First, crime datasets are hard to come by. Only a handful of major cities such as New York, Los Angeles, Austin and a few others keep up-to-date and publicly accessible records. Second, each of these cities' police departments report and classify crimes in slightly different ways. So, it

is important to make sure we have a unified classification of crimes for all cities to ensure that we're testing for the same type of crime across cities.

Chapter 7

Conclusions

This was just a first step in to the massive world of crime analysis. With so many factors in play, there are always some tweaks and changes that can be made to improve our models. We've suggested some good starting points for improvements in our future work section.

As for selection of models, crime data does not seem to exhibit the kind of independencies that are captured by MFPPs. The insignificant training time combined with the better performance make PCIMs the better choice of model for future crime data analysis.

Bibliography

[1] Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems*, pages 1962–1970, 2011.

[2] Ankur P Parikh, Asela Gunawardana, and Chris Meek. Conjoint modeling of temporal dependencies in event streams. 2012.

[3] Jeremy C Weiss and David Page. Forest-based point process for event prediction from electronic health records. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer, 2013.

[4] Mehrdad Farajtabar, Yichen Wang, Manuel Gomez Rodriguez, Shuang Li, Hongyuan Zha, and Le Song. Coevolve: A joint point process model for information diffusion and network co-evolution. In *Advances in Neural Information Processing Systems*, pages 1954–1962, 2015.

[5] Wenzhao Lian, Ricardo Henao, Vinayak Rao, Joseph E Lucas, and Lawrence Carin. A multitask point process predictive model. In *International Conference on Machine Learning*, pages 2030–2038, 2015.

[6] George O Mohler, Martin B Short, P Jeffrey Brantingham, Frederic Paik Schoenberg, and George E Tita. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493):100–108, 2011.

[7] Shane D Johnson. Repeat burglary victimisation: a tale of two theories. *Journal of Experimental Criminology*, 4(3):215–240, 2008.

[8] Michael D Porter, Gentry White, et al. Self-exciting hurdle models for terrorist activity. *The Annals of Applied Statistics*, 6(1):106–124, 2012.

[9] Erik Lewis, George Mohler, P Jeffrey Brantingham, and Andrea L Bertozzi. Self-exciting point process models of civilian deaths in iraq. *Security Journal*, 25(3):244–264, 2012.

[10] Scott Linderman and Ryan Adams. Discovering latent network structure in point process data. In *International Conference on Machine Learning*, pages 1413–1421, 2014.

[11] Carolyn R Block and Richard Block. *Street gang crime in Chicago*. Citeseer, 1993.

[12] Carolyn R Block. *Is crime seasonal?* Illinois Criminal Justice Information Authority Chicago, 1984.