Developer Documentation

Version: 1.0
Effective Date: 16-Feb-2015

Application Design Background

Abstract

[ ]

Application Design Initiation

Initiation Date 13-Feb-15

Coordinating Author ..... Matt Houseman

[ ]

Key Contributors .....George Mohler

..... Matt Houseman

[ ]

Approver.....[ ]

Scope / Applicability

[ ]

Revision History

Version	Date	Updated By	Comments
1.0	13-Feb-15	Matt Houseman	Initial draft

Table of Contents

BACKGROUND

This isn't meant to be an academic publication, therefore I will 'leveraging' from previous publications from George Mohler, Jeff Brangtingham and others without footnotes or any other attributions other than the following:

1) http://paleo.sscnet.ucla.edu/MohlerEtAl-JASA-2011.pdf
2) http://www.researchgate.net/publication/261564053_Marked_point_process_hotspot_maps_for_homicide_and_gun_crime_prediction_in_Chicago
3) The following PowerPoint presentations that I've included on the Wiki as background documents:
    a. ESADE_Sept2013.pptx
    b. UCSC_2_12.ppt
    c. USF_Sept2013.pptx

There is a need to have one document that provides an overview of the current predictive algorithm from a BIZ and technical point of view. That's what I intend to provide in this document.

SEISMOLOGY AND CRIME

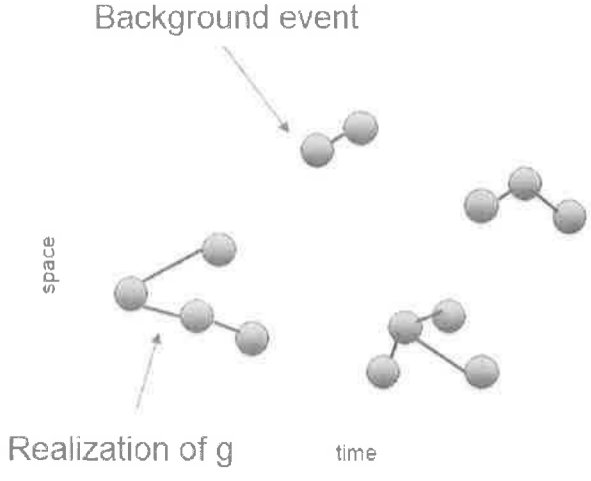
Like aftershocks following an earthquake, the theory with our predictive algorithm is that other crimes will follow in the geographic location of the initial crime. Earthquakes are a natural phenomenon; it makes sense that if a person is successful in accomplishing a task, that person would attempt to replicate the conditions for success.

The relevant seismology model that is referenced is Epidemic Type Aftershock Sequence (ETAS) Model. Refer to http://en.wikipedia.org/wiki/Aftershock for additional background information.

This is true of the assumptions of our predictive algorithm. For example, a criminal may have been successful in carrying out a burglary on a certain day of the week (Friday), at a certain time (10pm) and in a particular neighborhood. Based on a number of burglaries similar to the original burglary, our algorithm would then predict that future crimes would occur in 'hotspot'.

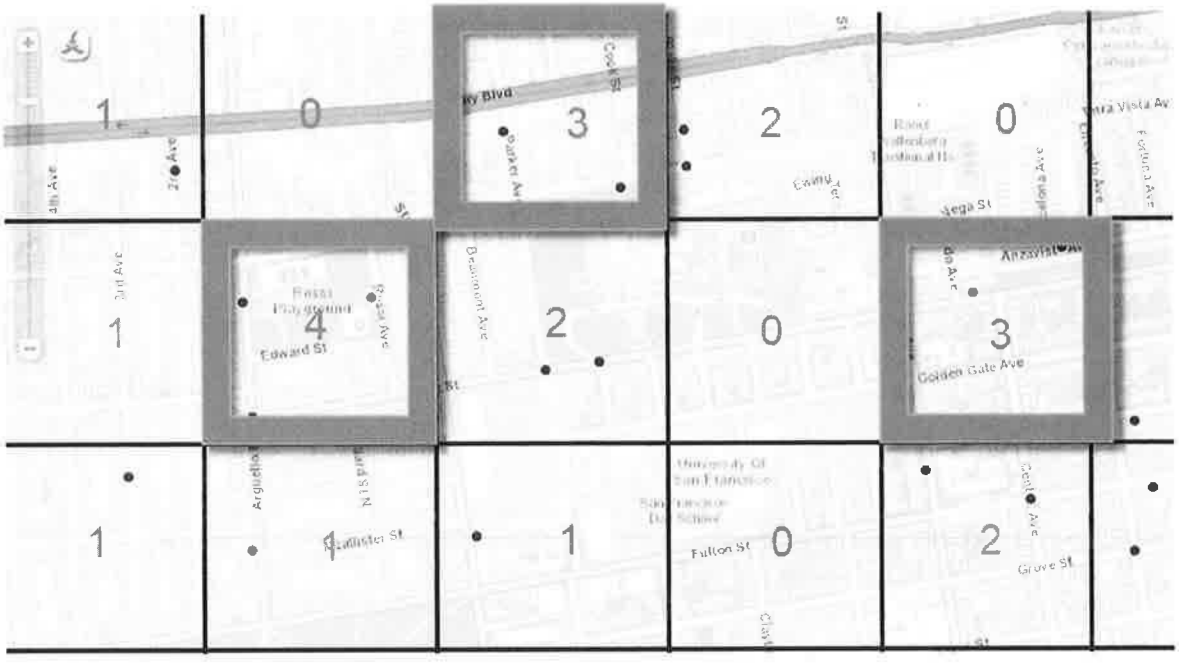
[redacted]

- Lay down "background" events according to Poisson process μ
- Iterate the following: given locations of previous generation, lay down next generation of events according to Poisson process g centered at parent events
- Stop when a generation has no events or is outside the space-time window



'HOTSPOT' POLICING

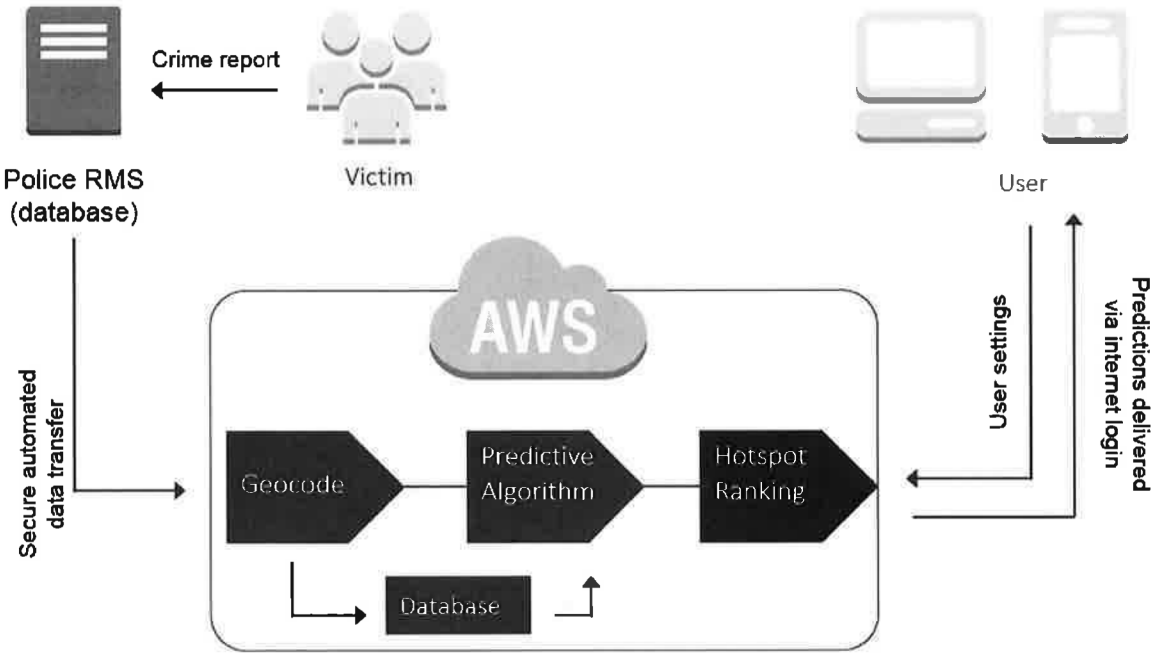
The following is how we render these predictions to our end users. What is displayed here is a grid placed over a map with a count of crimes (red number in each cell) and the location of each crime (blue dots in each cell). The prediction algorithm will identify where it's suggested that law enforcement spend some of their patrol time. This is 'hotspot' policing. We identify these 'boxes' by rendering them as bold red grid cells as shown below:



SYSTEM FLOW

So far, I've described at a high level the ideas behind the predictive algorithm and shown an example of hotspot policing.

There's an entire system flow that comprises the PredPol landscape which is pictured below:



1) The Police Management System (RMS) contains crime data
2) The secure automated data transfer is created by IT personnel at the Police Department running our User Setup Form and creating the data pipe connection
3) Geocoded crime data is pulled into the PredPol database from the Police RMS via the data pipe connection
4) A scheduled job is run to calculate a Police Departments hotspot rankings
5) Users can filter on crime types and are able to view the hotspot rankings on a grid in a Web User Interface

PREDICTIVE ALGORITHM DATA SOURCES

The models (tables) that the predictive algorithm uses are listed below:

1) districts
    a. Predictions are run per district using the district id.
2) settings
    a. Attribute/values pairs such as longmax, longmin, etc.
3) predictions
    a. Probabilities per crime type (category) written to this table. Only the most recent predictions are kept. Previous predictions are removed prior to running the prediction generation job.
4) crime_types
    a. List of crimes per districts. Burglary, gang activity, battery, etc.
5) crimes
    a. This data is imported from the data pipe. It includes geocoded and time stamped crime data. Each row also contains crime type, docket number, violation code and other information.
6) shifts
    a. Rows in this table contain the name of the shift, hour of when the shift starts, hour of when the shift ends and the number of boxes covered by the shift
7) prediction_vectors
    a. Contains every combination of crime types for every district for every day for which predictions were generated
8) maps
    a. Defines latitude min/max, longitude min/max, district, prediction vector and shift
9) boxes
    a. Reference a map row, nearby address, grid coordinates (i, j), latitude min/max, longitude min/max and district

OPERATION OF THE PREDICTIVE ALGORITHM

The creation of the daily prediction data is scheduled using a scheduler named Resque. The Resque scheduler is very similar to Cron. For those who would like more information on Resque then click the resque-scheduler link. If you would like more information on Cron then click http://en.wikipedia.org/wiki/Cron.

Every day, the Resque scheduler runs the PredPol/lib/work/run_predictions.rb script. The RunPredictionsJob::perform() method does the following:

1) Get a list of districts
2) Loop through the list of districts, create an OracleJob for each district and pass each district id into the created OracleJob's perform method

The PredPol/lib/work/oracle.rb script contains the core code of the Predictive Policing algorithm. The main entry point is the OracleJob::perform() method. The perform method does some setup work then in turn calls the OracleJob::prediction_main() method. The prediction_main method does the following:

1) Get a list of active crimes
    a. Get a list of crimes by crime type (category)
    b. Create predictions for this list of crimes by crime type
2) Create maps

The OracleJob::create_predictions() loops through each grid cell where each grid cell has a geolocation identified with latitude and longitude values. As the method loops over each grid cell, the random rate of crime is set proportional to the counts of crimes in each grid cell. The model initially assumes that half of the crimes are random. Recall that the count was populated by getting a list of active crimes. As we loop through the crimes, calculate the probability that a crime triggered a subsequent crime. The closer in time between the two events, the higher the probability. If the crimes were at the exact same location, force the probability to be 100%. If the time difference was less than an hour or greater than 50 days then force the probability to 0%. Filter out any blacklisted geolocations then persist the results to the predictions table. There are 24 probability columns in the predictions table: probability_00, ..., probability_23 that correspond to hours in the day.

ADDITIONAL INFORMATION

With any software company, the source of truth is the source code. George and I have spent many hours documenting the oracle.rb file with inline comments. I highly encourage you to read through the source code and trace through the various code paths. For example, at the end of the prediction_main method in the OracleJob class, the create_maps method is dispatched. As one traces through the create_maps method, you will see that classes such as PredictionVector, Shift and Map are referenced. These are of course several of our models which can be found in PredPol/app/models as prediction_vector.rb, shift.rb and map.rb respectively. Finally, these models map one to one with tables of the same name, with the exception that the tables are plural: prediction_vectors, shifts and maps.