Analyzing Crime in Chicago Through Machine Learning

Nathan Holt

Rochester Institute of Technology
nxh7119@rit.edu

May 14, 2017

Abstract

In this paper, we attempt to apply techniques from machine learning to data from a dataset released by the government on crime in the city of Chicago. We apply K-means clustering as a way of compartmentalizing the records of crime. We then propose methods of "predicting" hate crimes based upon the clustered data.

I. Introduction

The city of Chicago publishes an up-to-date list of all reported crimes. The records span from 2002 to the modern date, allowing a few days of delay to catalog the crimes and publish them. According to official FBI data, Chicago is one of the leading cities in homicides, having more than quadruple the amount of crimes in New York City, and more than double the amount of crimes in Los Angeles [3]. Being able to parse the data presented by these huge data sets is a problem central to understanding the crimes in the city of Chicago.

By analyzing the data in a mathematically rigorous way, researchers may be able to glean insight into the underlying causes of crimes, and also may be able to figure out indicators of future crimes to occur. All of this categorization and analysis falls under the umbrella of Data Science, a field which analyzes large sets of data using probability and statistics, and makes useful conclusions from the analysis.

In this paper, we will attempt to parse the city of Chicago's up-to-date dataset, and try to perform some crime "prediction." We will do this by utilizing techniques from machine learning: specifically, K-means clustering.

II. Methods

In this section, I will discuss the methods used to actually obtain the results.

i. The Data Set

The data set is publicly available through the city of Chicago's website [2].

The information presented in this data set is quite comprehensive, including information about the date and time of the crime, location of the crime, type of crime, etc. For the purposes of this paper, we will focus on the time of the crime and the type of crime.

The type of crime is given a standardized set of codes called the Illinois Uniform Crime Reporting (IUCR) codes. Thus, each IUCR corresponds to a specific type of crime. The list of crime codes and corresponding crimes can also be found through the city of Chicago's website [1].

ii. K-means Clustering

K-means clustering provides a way to group data points together in a way that minimizes differences between the data points in the same group. By applying these methods, we can take n data points and partition them into k clusters. The algorithm seeks to minimize the following function:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

In this equation, x is a vector that corresponds to a specific crime instance in the data set.

μ_i is the "centroid." It is is the average point (also a vector) in a given cluster. The entire algorithm works by minimizing the distance between all data points (crimes), and the corresponding centroid of the cluster that they're in. We square the Euclidean distance between these two points to make all distances positive, a common technique in statistics.

S is the set containing all cluster assignments. Thus, S_i contains all the points in the ith cluster.

We sum over all the elements in a given cluster, S_i, and then we sum over all the different clusters, to obtain a total "distance" of all points from the centers of their corresponding clusters.

We then attempt to minimize this by finding the optimal assignment of clusters, S.

iii. Serialization of Data

In order to make K-means clustering viable to our data set, we must serialize the data to convert all crime information into a corresponding point on the graph. For simplicity, we are focusing on three specifications of a crime: the date the crime occurred, the time the crime occurred, and the type of crime that occurred.

To serialize our data, we must convert these three fields to points in 3 dimensional space, with each dimension corresponding to either: date, time, or IUCR code. To do this, I looped through the fields in the following way:

- Dates are serialized starting with January 1st corresponding to 1, January 2nd corresponding to 2, and so on.
- Times are serialized starting with midnight corresponding to 1, 12 : 01 corresponding to 2, and so on.
- IUCR Codes are serialized starting with the lowest values and moving on to the higher values.

Since we are going to take the Euclidean distance in each dimension, we want closer values of the serialized data to correspond to more similar crimes. This is accomplished as a slight change in the value of the date or time serialization means a slight change in the date or time. And, based on the way that the state of Illinois classifies crimes based on codes, close IUCR code numbers correspond to similar crimes. For example, first degree murder and second degree murder are neighboring IUCR codes, meaning that they take on values that are next to each other when we serialize the data. Thus, we can use this way of serialization to preserve distance, so related things are closer valued.

As a side note, during the research and implementation of K-means clustering, it was discovered that the city of Chicago had missing entries in the IUCR code data set. Several IUCR codes used in the data set of crimes in Chicago had no corresponding entries in their data set. I chose to pretend that these are valid IUCR codes that are just not cataloged in their database of IUCR codes. Thus, I hard-coded a few entries of IUCR for crime codes that would fit appropriately for the Euclidean distance. For example, if IUCR codes of 120 and 122 are valid and serialized as 3 and 4 respectively, then I would serialize 121 as 4, and shift 122 to 5 accordingly. I only shifted values over when there was an entry with "valid" IUCR code according to their database. Everything else was left unchanged in terms of serialization. Code is available for viewing on my website [1].

iv. Lloyd's Algorithm

Thus far, we have described the method of K-means clustering, and how to parse and serialize the data for usage in K-means clustering, but we haven't discussed the algorithm to actually minimize the distances between the points and the centroids. There are several approaches of how to do this, but for the purposes of this paper, we will use Lloyd's Algorithm.

Lloyd's Algorithm is a method of finding evenly spaced subsets of a larger set given a specific number of partitions (clusters).

[1] To view the code, please go to https://nathanwayneholt.com/ChicagoCrimes.py

The first step in the algorithm is to assign initial clusters. There is much in the literature about various ways to assign initial clusters. However, I just chose the naive approach, and took the first k points in the data set. I consider these data points as the k centroids of the clusters. One we have the initial centroids, we assign each data point to one of the clusters, based upon minimization of the Euclidean distance between the centroid and the data point. This corresponds to the equation:

$$S_i = \{x : \|x - \mu_i\|^2 \leq \|x - \mu_j\|^2 \ \forall j, 1 \leq j \leq k\}$$

As before, the μ_i is the centroid point of the ith cluster. Thus, the initial clusters are just all the points whose Euclidean distance is minimized.

Then, the iteration begins. In each step of the iteration, we calculate new centroids based on the assignment of points in the current iteration. We take the distances summed up together (the equation on the second page), and divide by the number of points in the cluster. That is, we take the average of the points, and declare this as the new centroid of the cluster. This corresponds to the equation:

$$\mu_i^{t+1} = \frac{1}{|S_i^t|} \sum_{x_j \in S_i^t} x_j$$

Three things are important to note here. First, the summation of the points is done component-wise, since they're three dimensional data points. Secondly, since we're taking the average of all the components to create a new centroid, the new centroid we create may not be a point in our original data set. This is okay, since we don't care if the centroid of a given cluster is in our data set. We just care that the distance between the points and the centroid is minimized. We only naively set the original centroids to be points in our data set to make initialization of the algorithm easy. Thirdly, it's important to note that now that we're iterating, I introduce the superscript notation to denote the given iteration. Thus, μ_i^{t+1} is the centroid of the ith cluster, on the $t+1$ iteration of the algorithm.

The iteration continues until the centroids don't change when updated. At this point, the centroids are said to "converge," and we take that as our final centroids.

It's also important to note that since we have a discrete set, we must partition the values in each axis as values between 0 and 1. Otherwise, the weighting calculated by the Euclidean Norm will not be the same in every dimension. As a visual proof of this, I have included the following figure in which I removed normalization of all the values in all dimensions. As visually apparent, it simply partitions the set based upon the time dimension. This is because there is a much larger span of possible values in the time dimension (orders of magnitude larger), so the algorithm would minimize the distance between points based solely on this dimension; the other dimensions would have little impact.

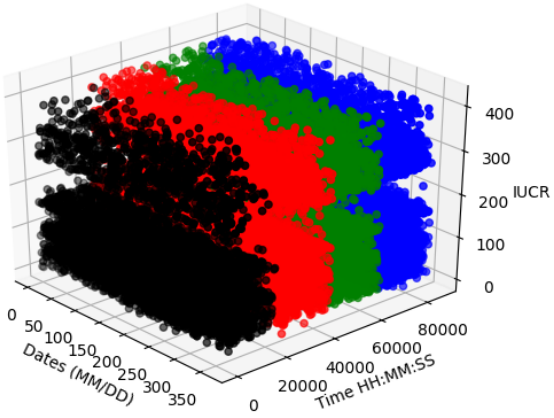
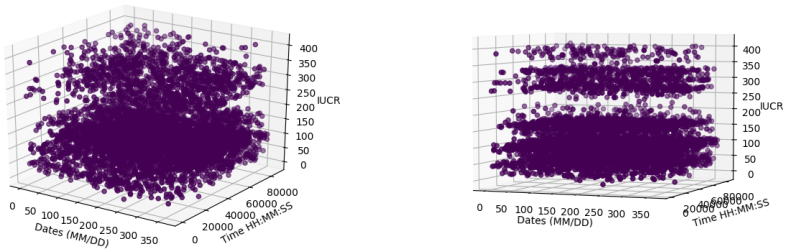


Figure 1: *A plot showing the dominance of the time dimension over others without normalization*

III. Results

First, we'll plot visualizations of the points in 3 dimensional space. Since the data set we serialize is over 6.5 million entries long, Python will crash if attempting to plot all the points in a single display. Instead, we sample points.



(a) *The default view* **(b)** *A rotated view*

Figure 2: *A plot of a sampling of every* 1000 *points in the dataset*

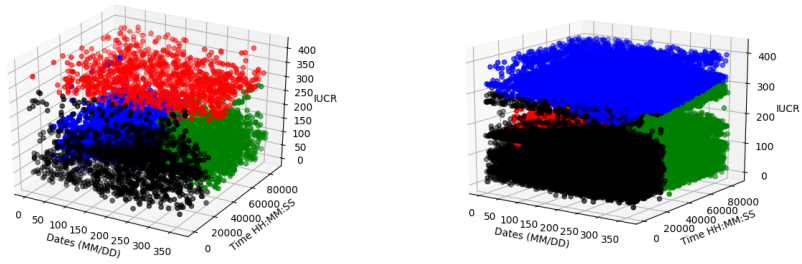
In figure 2, we see a visualization of the serialized data, sampled every 1000 points. Since there are approximately 6.5 million data points total, there are approximately $6,5000$ points on these visualizations.

Immediately, by plotting these points we can garner insight into the underlying structure of the data that would not otherwise be visible in just a spreadsheet. We notice that crimes corresponding to IUCR codes between 200 to 250 are very spare in contrast to other crimes. When cross-referencing this with the database of IUCR codes provided by the state of Illinois, we obtain that these values of IUCR codes correspond to sexual crimes. Immediately, we see that sexual crimes such as rape or sexual assault are far less frequent than other crimes.

We also gather that gambling is far less frequent than other crimes by the same method. It's worth noting that the database includes all reported crimes, but unreported crimes are not listed in the database (of course, that's because no one has reported them!), so it may not be entirely accurate of what actually happens in the city of Chicago.

Now that data is serialized, we can run our Python implementation of Lloyd's Algorithm and K-means Clustering. We have to make a choice of value of K (decide how many clusters to partition into). We partition into 4 clusters first.

When run, we obtain the following output:



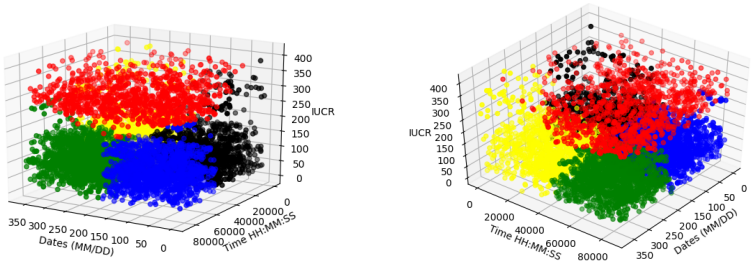
(a) *Sampling every 1000points* **(b)** *Sampling every 100 points*

Figure 3: *Plots showing the result of K-means clusters for* $K = 4$

Here I have color-coded the points to make it visually easy to identify which belongs to which cluster.

In figure 3, we show the output generated by the algorithm when partitioning the set into 4 clusters.

We also chose to run to algorithm for $K = 5$, in order to see the potential differences in the way points cluster based upon a small change in the number of clusters.



(a) *The default view*

(b) *A rotated view*

Figure 4: *A plot of a sampling every* 1000 *points, when* $K = 5$

In figure 4, we can see the effects of partitioning into 5 clusters instead of 4. We see that, in general, the highest value of IUCR codes are in their own cluster, whereas the lower IUCR codes are split into 4 clusters. This is largely due to the large gap 50 between the 200 and 250 IUCR codes, as mentioned earlier.

IV. Discussion

This project has largely been a proof of concept to show that clustering data sets about crime is entirely possible, and allows us to gain insight into the underlying trends of crime.

i. Future Work

Clustering data and analyzing it is an important step to many other machine-learning algorithms. In particular, prediction of crime points can be achieved by multiple methods:

- Minimizing the distance between a crime occurrence and the centroid of a cluster
- Performing regression analysis on the identified clusters and fitting crimes to the best fit line

In the current implementation, we can take a partially filled out crime report and match it to the most likely cluster based on the minimization of the dimensions provided. This leads to the possibility of studying crimes on specific dates, times of the day, or specific types of crime. For example, perhaps law enforcement would be interested in knowing the most statistically likely type of crime happening on January 1 at midnight. If that was the case,

the crime output would take the form of $(1, 1, z)$, where z corresponds to an IUCR code. By minimizing the distance to centroids, we could calculate the most likely cluster it's in. We can also form a regression line for serialized points we have that match coordinates via least squares or similar methods. We could also implement the K-nearest neighbors algorithm (which finds the most similar points given a specific points), with the condition that they be in the same cluster as we predict our data point will be in. This would provide a list of k possible crime types that are most likely to occur in Chicago on January 1 at midnight.

References

[1] *Chicago Police Department - Illinois Uniform Crime Reporting (IUCR) Codes | City of Chicago | Data Portal*. URL: https://data.cityofchicago.org/Public-Safety/Chicago-Police-Department-Illinois-Uniform-Crime-R/c7ck-438e/data.

[2] *Crimes - 2001 to present | City of Chicago | Data Portal*. URL: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2.

[3] Eliott C. McLaughlin. *With Chicago, it's all murder, murder, murder ... but why?* Mar. 2017. URL: http://www.cnn.com/2017/03/06/us/chicago-murder-rate-not-highest/.