CS 229 Project Report: San Francisco Crime Classification

Charles Hale, Feng Liu, {cphalepk, lufeng}@stanford.edu

Abstract

Different machine learning approaches were conceptualized and implemented for predicting the probabilities of crime categories for crimes reported in San Francisco. The crimes records used in the research are downloaded from a competition on Kaggle. A Bayesian model, a mixture of Guassians model (stratified and unstratified), and logistic regression are implemented. A satisfactory result was achieved with Bayesian model, corresponding to a Kaggle leaderboard of 852th out of 2335 teams.

1 Introduction

There is a huge amount of crimes reported every day in the city of San Francisco. The San Francisco Police Department (SFPD) has captured all of these reports within their systems and has made the data publicly available in a Kaggle competition. The challenge is to "predict the category of crimes that occurred" based on information the information in their reports.

2 The Dataset

The data used in this classification problem are records of crimes that happened in San Francisco between the dates of 1/1/2003 and 5/13/2015, as derived from SFPD Crime Incident Reporting system. The training set and test set rotate every week, meaning week 1,3,5,7... belong to test set, and weeks 2,4,6,8... belong to training set. In the each set, we are given the following information:

- Dates - timestamp of the crime incident, in the format of YYYY-MM-DD HH:MM:SS.
- Category - category of the crime incident (only in training set). This is the target variable we are going to predict.
- Descript - detailed description of the crime incident (only in training set).
- DayOfWeek - the day of the week.
- PdDistrict - name of the Police Department District.
- Resolution - how the crime incident was resolved (only in training set).
- Address - the approximate street address of the crime incident;
- X - Longitude;
- Y - Latitude.

3 Objective

Our goal is to predict the probability that a crime belongs to certain category based on its time and location. Predictions are evaluated using the multi-class logarithmic loss:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

where N is the number of cases in the test set, M is the number of class labels, log is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j. In order to avoid the extremes of the log function, predicted probabilities are replaced with $\max(\min(p, 1-10^{-15}, 10^{-15})$. This is the objective given by the Kaggle competition guidelines. And, after running SVM classifiers on the data and achieving no more than 26 percent accuracy on any of the classes, we decided that the multivariate log-loss is a reasonable and more appropriate loss function for this use case.

4 Preliminary Findings

As a start, we explored the data to figure out some of the peculiarities of this problem. And our preliminary findings then shaped the approaches we decided to take.

1. After plotting the data based on their location coordinates, we realized that about 67 points had latitudes of 90 degrees–meaning they took place at the North Pole. These points were heavy outliers and were excluded them from our training set.

2. The data was heavily skewed in terms of the frequency of each category. Only 19 of the 39 classes had proportions of the training set in amounts above .5 percent. Since we were seeking to minimize the log-loss, we decided our efforts focus overwhelmingly on doing well on distinguishing between these major classes, since errors on the remaining classes would contribute little to the error.

3. Many of our intuitive guesses were correct. For instance, certain classes of crime were much more frequent on weekends (and vice versa). Many crimes had different frequencies depending on the time of day. As such, we found it appropriate, when designing our models, to use feature mappings that take these intricacies into account. Namely, we mapped the time stamps into a much higher dimensional feature space, i.e.–month, weekday, time of day, etc.

4. We did a lit review of findings based on this data set. Of the thousands of posts about it, most involved finding interesting ways of visualizing the data, and divulged little information about approaches to the actual classification problem.

5. The training set contained over 8 million reports, but each report had a relatively small amount of information. We decided to disregard most of the textual columns such as PdDistrict and Address (as location information is already captured in numerical form by the coordinates), and Descript and Resolution (as they are not given in the test set, and their information is simply a supplement to Category). Thus we were left with Dates, the coordinates, and the labels. This problem then became an issue of predicting between a large amount of classes on a large set of data points, based on a relatively small amount of features.

6. Given that we had plenty of data with a small amount of features, we figured the most difficult part of this problem would, in fact, be in finding the most appropriate model for this application.

5 Approaches

So far, we have implemented two methods.

5.1 Bayesian Model

We can rewrite the probability we aim to predict as $P(z|x_1, x_2, ..., x_p)$, where z is the category of crime, x_1 to x_p are the features we use in prediction. This method is inspired by the Naive Bayes model, where we create a model for each of the features individually, under the assumption that they are independent. Yet each of our feature models are build using locally-weighted regression methods. The assumption that we can treat each of the features independently is proposed based on the fact that the correlations between features are generally not significant. For instance, the correlation between longitude and latitude is among the highest, on the order of 10^{-1}, while other correlations are on the order of or less than 10^{-2}.

We estimate the conditional probability as

$$\begin{aligned} P(z|x_1, x_2, ..., x_p) &= \frac{P(x_1, x_2, ..., x_p|z)P(z)}{P(x_1, x_2, ..., x_p)} \\ &= \frac{P(z)\prod_{i=1}^{p} P(x_i|z)}{\prod_{i=1}^{p} P(x_i)} \\ &= \frac{P(z)}{\prod_{i=1}^{p} P(x_i)} \prod_{i=1}^{p} \frac{P(z|x_i)P(x_i)}{P(z)} \\ &= \frac{\prod_{i=1}^{p} P(z|x_i)}{P(z)^{p-1}} \end{aligned}$$

In this way, the problem transforms to estimating $P(z|x_i)$ for all features (Naive Bayes).

The potential features are the serial number of time interval of 14 days, the day of week, the hour of day, the longitude, and the latitude. The choice of 14 days is basically because the entries in training data were recorded in alternative weeks.

Along each feature, for example, a time interval, we calculate the proportion of the number of a certain category of crimes out of the total number of all crimes that happened in the same interval. Preliminary analysis showed that the proportion of different categories of crimes have different shapes. Despite the general trend we could see directly from the plot, there also exists too much noise in the proportion calculated by counting. Therefore, curve fitting is necessary.

We prefer a non-parametric method in this step. Locally weighted linear regression may be a plausible method to perform curve fitting. The optimal parameter for local weighted regression is determined by cross-validation.

After have acquired the conditional probability on each feature, we combined them using the formula given above to calculate the predicted probability matrix.

5.2 Supervised Clustering

With this approach, we take on the world view that crime stems from a set of trends. A trend in crime is a set of similar crimes on the basis of their category, place, and time. So, in order to predict the probability of a crime belonging to a certain category,

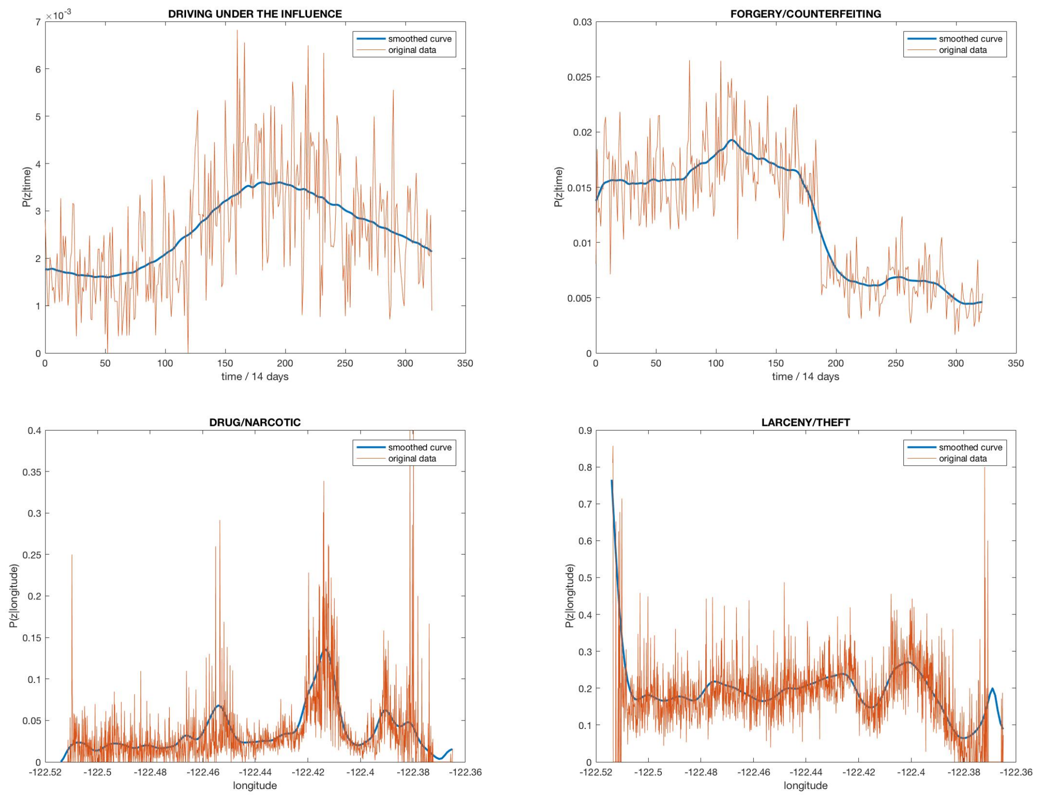

Figure 1: Locally weighted regression results with best span selected with CV in the Bayesian model

we first build a model of different crime trends in the city of SF, and then subsequently estimate its probable category based on which trend it is likely to belong to.

Initially as an assumption, we stratified our trends based on category, i.e.– we assume that crimes with different categories must be separate trends. In this case, we have to cluster the data into trends without referring to the category or label, because all the data in a cluster must be the same category. Clusters are decided based on some similarity measure, which we take as the L2 norm under a particular feature mapping. We also, for the purposes of producing a final probability, consider the distance to be related to the conditional probability of a point given a trend. So, in order to find a probability, we first use clustering to find a "trend".

For each category i, we find $\{trend^{(ij)}\}_{j=1}^{k^{(i)}}$ via Mixture of Gaussian clustering.

$$
\begin{aligned}
&P(i|x_1, x_2, \ldots, x_n) \\
&= \frac{P(x_1, x_2, \ldots, x_n|i)P(i)}{P(x_1, x_2, ..., x_n)} \\
&\propto P(x_1, x_2, \ldots, x_n|i)P(i)
\end{aligned}
$$

where $P(x_1, x_2, \ldots, x_n|i)$ is simply the pdf of our gaussian mixture, the set of $trends^{(i*)}$.

Explicitly,

$$
\begin{aligned}
&P(x_1, x_2, \ldots, x_n|i) \\
&= \sum_{j=1}^{k^{(i)}} P(x_1, x_2, ..., x_n, trend^{(ij)}|i) \\
&= \sum_{j=1}^{k^{(i)}} P(x_1, x_2, ..., x_n|trend^{(ij)})P(trend^{(ij)}|i)
\end{aligned}
$$

An important part of this is our assumption that we can calculate $P(x_1, x_2, \ldots, x_n|trend^{(ij)})$. We take this to be proportional to $exp(-||\phi(x_1, x_2, ..., x_n) - z_j^{(i)}||_2^2)$ (or that this conditional distribution is Gaussian, with mean equal to the $z_j^{(i)}$, the centroid of $trend^{(ij)}$). This is also the distance metric by which we decide if a label belongs to a cluster.

In this case, it is natural to use the EM algorithm to find each trend. However, as a minimum viable product, we implemented this with k-means, and then assumed all covariance matrices to be the identity.

The results of this were the same as that of the trivial estimation. We found this result to be quite natural given that the centroids would simply be representatives of points in the dataset, and by computing distance from them, we more or less end up with the average, giving us approximately the trivial estimation.

Moving to the Mixture of Gaussians EM-Algorithm, we had $trend^{(ij)} \sim \mathcal{N}(\mu^{(ij)}, \Sigma^{(ij)})$.

We found that the computation of each of these models would often result in numerical errors. Some classes had less points than clusters or features. We added regularization to make sure that covariance matrices would stay full-rank. However, some of them were still poorly conditioned, so we instead took the approach of ignoring small classes, and simply predicting their probability to be zero.

The results of this method were inconsistent, but mostly tended to do worse than trivial prediction. To ensure that we were not over-fitting, we ran this method on the full data set, and only produced worse results. We also tried eliminating some of the features, and found that the method tended to do better on the training set with less features. As a result, we concluded that this model was fundamentally unsuitable or unstable for our data.

5.3 Unsupervised Clustering

We pivoted to this method after having poor results with supervised clustering. Here we calculate a set $\{trend^{(j)}\}_{j=1}^{N}$ using our entire dataset. We do not stratify between classes. We use the EM-algorithm and have each $trend^{(j)} \sim \mathcal{N}(\mu^{(j)}, \Sigma^{(j)})$. Then, we calculate

$$P(i|x_1, x_2, \ldots, x_n)$$
$$= \sum_{j=1}^{N} P(i, trend^{(j)}|x_1, x_2, ..., x_n)$$
$$= \sum_{j=1}^{N} P(i|trend^{(j)}, x_1, x_2, ..., x_n) P(trend^{(j)}|x_1, x_2, ..., x_n)$$

$P(trend^{(j)}|x_1, x_2, ..., x_n)$ we just get by evaluating the posterior distribution of our mixture of gaussians.

$$= \sum_{j=1}^{N} P(i|trend^{(j)}) P(trend^{(j)}|x_1, x_2, ..., x_n)$$

Which we get via the assumption that the distribution of crimes is determined by the trend alone. And further, we look at:

$$P(i|trend^{(j)}) = \sum_{l=1}^{m} P(i, x^{(l)}|trend^{(j)})$$
$$= \sum_{l=1}^{m} \{y^{(l)} = i\} P(x^{(l)}|trend^{(j)})$$
$$\propto \sum_{l=1}^{m} \{y^{(l)} = i\} P(trend^{(j)}|x^{(l)}) P(x^{(l)})$$

Here again, we get $P(trend^{(j)}|x^{(l)})$ via the posterior distribution of our mixture of gaussians, and we get $P(x^{(l)})$ via the pdf of the entire gaussian mixture. Algorithmically, we can think of $P(i|trend^{(j)})$ as an $N \times k$ matrix, P_{locals}, where each row of this matrix is the distribution of each category within a trend. As given by the formula above, we have a training step

$$P_{locals} = P_{posterior,X}^{T} * diag(p_X) * Y$$

(and then we divide each row by the sum), where

$$Y \in \mathbb{R}^{m \times k}, Y_{ij} = 1\{y^{(i)} = j\}$$

and

$$P_{posterior,X} \in \mathbb{R}^{m \times N}$$

,

$$(P_{posterior,X})_{ij} = P(trend^{(j)}|x^{(i)})$$

Then, using that matrix, when we want to predict on a test set determined by the matrix $\tilde{X}$, we have $\hat{P} = P_{posterior,\tilde{X}} * P_{locals}$.

5.4 Other Approaches

We considered briefly using Support Vector Machines for this classification problem, as we would easily be able to distinguish between kernels and enhance our feature mappings. However, Support Vector Machines do not have a natural analog to the probability we are trying to estimate. Beyond just getting good classifications, we would need to find some way to ascribe meaning to the margins, which seemed to be an unsure path.

We also briefly tried multinomial logistic regression. The hypothesis used in multinomial regression here is a log-odd of polynomial form, with binary variables indicating categorical features of the hour and the day of week. We submitted our prediction to Kaggle and the test loss was 3.88358, worse than the trivial prediction of 2.68016 (this result will be analyzed and discussed further in the discussion section). Multinomial logistic regression also seemed to be computationally expensive. Since we had so many classes (even excluding ones with near zero probability), the cost would expand quickly in the number of features. With limited computational resources, it would be difficult to experiment with model/feature selection.

We also considered K-nearest neighbors for this problem, as it would seem natural. We could assume from experience that, similar crimes are more likely to happen in the "neighborhood", both spatially or temporally. Simply speaking, we would like to estimate the probability of a crime by what happened nearby.

In the temporal dimension, the frequency crimes of is supposed to show periodicity. But K-nearest

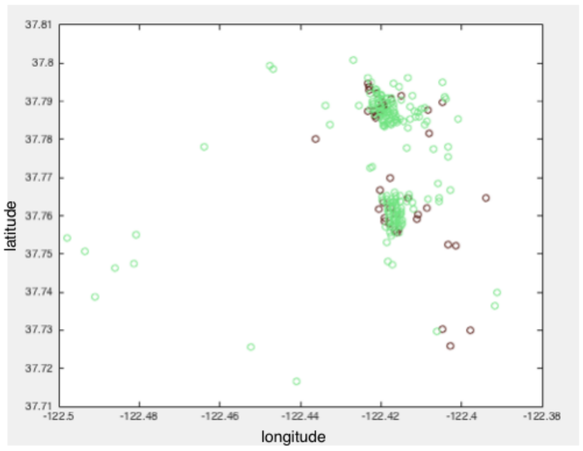

Figure 2: The reports of prostitution in our training subset, with each cluster shown in different colors.

neighbors seemed to be a more limited version of our clustering approaches. Additionally, this method could be too expensive in computationally. In its most naive implementation, in order to make an estimation on a new data point, we would have to calculate the distance from every point in the training set. This would be approximately $O(nm^2)$, which would not be tractable for one sweep, much less in choosing K or our feature mapping. Every method we implemented maintained linear time in the size of the training set and linear time in the predictions.

6 Results

6.1 The trivial submission

The trivial submission mentioned above refers to, regardless of any differences among features, simply "predicting" the probability of any case belonging to each category to be the category's frequency in the training data. The test loss was 2.68016 and it ranked 1553th on the Kaggle leaderboard.

6.2 Bayesian model

The training loss is 2.5345. The test loss is 2.53674, computed by Kaggle.

This result ranks 852th out of 2335 teams on the Kaggle leaderboard, and moves 701 positions forward compared to the trivial submission which predict probability without any feature.

6.3 Supervised Clustering

This method had poor results. We had a training loss of 2.8602 and a test loss of 2.90456.

6.4 Unsupervised Clustering

For our training set, we achieved a loss of 2.65053 and in our test submission to Kaggle we had a loss of 2.65053 and ranking of 1476.

7 Discussion and Future Work

Logistic regression may still has room for improvement in this problem. One obstacle by now is the time complexity of model selection. One possible solution is to add one assumption of additivity, such that a generalized additive model can be built for logistic regression. Techniques like backfitting can be utilized to regress on each feature separately, thus lower the time complexity.

In the Bayesian model proposed above, further improvements may be achieved by splitting the features more finely. For example, the empirical probability given the hour of the day can be split one more step depending on the weekdays or weekends. And we also notice that the longitude and the latitude are not completely independent as other features do. To modify this we can pair the longitude and the latitude, and perform locally weighted regression in two dimension of location. These may be the directions in which we could try more in the future.

Additionally, our unsupervised clustering methods may benefit from better feature mappings. Figure 2 shows the clusters found for prostitution. It would appear there are clearly two different clusters by location. However our algorithm was unable to distinguish them.