2016 15th IEEE International Conference on Machine Learning and Applications

Automatic Optimization of Localized Kernel Density Estimation For Hotspot Policing

Mohammad Al Boni and Matthew S. Gerber
Department of Systems and Information Engineering, University of Virginia
Charlottesville, Virginia, USA
Email: {ma2sm, msg8u}@virginia.edu

***Abstract*—Kernel density estimation is a popular method for identifying crime hotspots for the purpose of data-driven policing. However, computing a kernel density estimate is computationally intensive for large crime datasets, and the quality of the resulting estimate depends heavily on parameters that are difficult to set manually. Inspired by methods from image processing, we propose a novel way for performing hotspot analysis using localized kernel density estimation optimized with an evolutionary algorithm. The proposed method uses local learning to address three challenges associated with traditional kernel density estimation: computational complexity, bandwidth selection, and kernel function selection. We evaluate our localized kernel model on 17 crime types from Chicago, Illinois, USA. Preliminary results indicate significant improvement in prediction performance over the traditional approach. We also examine the effect of data sparseness on the performance of both models.**

I. Introduction

Hotspot mapping is tool used by police departments to analyze historical crime records and identify future areas of high risk. According to a report by the Bureau of Justice Statistics in 2007, 92% of police departments serving more than 1 million citizen in the United States use hotspot mapping [19]. Hotspot mapping assumes that criminal activities are spatially stable over time. To the extent that this is true, hotspot methods can use historical crime incidents to generate predictions for future areas of high risk. There are many ways to generate hotspot maps [3, 4, 8, 10, 16]. A recent study has shown that kernel density estimation (KDE) outperforms other hotspot methods in terms of prediction accuracy [9]. KDE requires the analyst to define a number of parameters, and researchers have proposed various ways of setting these parameters [2, 5, 8, 11]. However, the optimal KDE parameters will vary depending on the study region, the crime type, and the data characteristics.

In this work, we propose a novel method for generating hotspot maps using localized kernel density estimation (LKDE), where the LKDE parameters are automatically optimized using a genetic algorithm. The proposed method is motivated by ideas from image processing, specifically convolution filtering. We have evaluated the LKDE on historical crime records of 17 types from Chicago, Illinois, USA. The proposed LKDE significantly outperforms a standard KDE on 12 of 17 crime types. Furthermore, the peak performance gains occur within a small percentage of surveillance, making LKDE practically applicable given limited police resources.

The contributions of this work include (1) a computationally efficient method of estimating a local kernel density, (2) a method of learning a dynamic bandwidth and exponentially decaying interpolation kernel, and (3) a robustness analysis of both KDE and LKDE in the face of increasing data sparseness.

II. Related Work

The KDE method has been widely adopted by police analysts for hotspot mapping. It is also included in many commercial and non-commercial spatial analysis packages such as ArcGIS, MapInfo, R, and CrimeStat III. Furthermore, KDEs have been integrated with spatial, temporal, and social media data to improve the performance of crime prediction models [1, 6, 12]. Researchers have studied KDEs and compared them to other hotspot mapping methods [9, 15, 18]. Chainey et al. argue that KDEs offer the best prediction performance across all hotspot mapping methods [9]. On the other hand, Pezzuchi and Levine argue that KDEs are not universally the best considering different crime types and environment characteristics [15, 18]. One of the reasons for this ongoing debate is that KDE performance can vary depending on how parameters are set within the estimator. There are three user-defined parameters that can affect KDE performance: grid cell size, interpolation method (kernel function), and search radius (bandwidth). Chainey has examined the effect of different grid cell sizes and bandwidths on KDE performance [7], and he concluded that varying the grid cell size has little to no impact on performance. More important is the choice of bandwidth. These findings were later confirmed by Hart and Zandbergen [13]. They also studied the effect of various interpolation methods on performance and found that among the three parameters, interpolation methods have the biggest performance impact. Researchers have offered guidance on setting the grid cell size and bandwidth [2, 5, 8, 11]. However, to date, there has been no guidance about which interpolation method should be used with respect to various scenarios. To fill this gap, we propose a new method for kernel density estimation in which we learn the optimal bandwidth and interpolation method from the data. Moreover, we examine the effect of data sparseness on the performance of the proposed method and the traditional KDE method. This has not been studied in previous research.

978-1-5090-6167-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICMLA.2016.50

III. Mathematical Approach

A. Kernel Density Estimation

KDEs are models that, when applied to crime, estimate higher risk in areas of higher historical density. The earliest KDE formulation was put forth by Rosenblatt in 1956 [20]. It was later extended by Parzen in 1962 [17]. The first step in performing KDE is to build a grid with equally sized squares. One then calculates density estimates for the center points of the squares such that cells with more surrounding incidents have higher density estimates and therefore higher predicted risk. The distances between incident points and the center points are scaled using a bandwidth parameter, and the results are fed to an interpolation function. The KDE is formally defined as

$$k(p,h) = \frac{1}{Ph}\sum_{j=1}^{P} K\left(\frac{||p-p_j||}{h}\right), \quad (1)$$

where P is the total number of crime incidents, h is a smoothing parameter (bandwidth), p is the point at which a density estimate is calculated, $||\cdot||$ is the L-2 norm, and K is an interpolation (kernel) function. Researchers have suggested different values for the bandwidth. Bailey and Gatrell claimed that the bandwidth should be set to 0.68 times the total number of incidents raised to the power of -0.2 [2]. Felson argues that the bandwidth should be set to cover a few blocks surrounding each point [11]. Most statistical packages offer default values for the bandwidth based on heuristics. For example, ArcGIS provides five different algorithms for setting the bandwidth.[1] One approach is to set the bandwidth to the average distance between the center points and all incident points. R software, on the other hand, provides a function[2] to estimate the bandwidth using a multivariate selection algorithm [22]. As for the interpolation method, there are a number of functions that can be used such as normal, triangular, quadratic, uniform, and bi-weight density functions. However, there has been no study of which function should be used in different situations, and often this is determined empirically. Another parameter, which is not formally part of the KDE formula, is the grid cell size for the hotspot map. Chainey and Ratcliffe have suggested that the grid cell size should be set to the shorter of the width and height of the region's bounding box divided by 150 [8]. Caplan and Kennedy suggest that the optimal grid cell should be within 1/2 and 1/3 the mean blockface length [5].

Another important factor is the computational complexity of KDEs. The summation in Equation 1 is over all P incident points, and this summation is calculated for all G grid points. Thus, the L-2 norm, the smoothing calculation, and the kernel function will be evaluated $O(G*P)$ times to produce a hotspot map. In a city like Chicago, Illinois, it is not uncommon for G to be on the order of 10^4 and P to be on the order of 10^3 for just one month of crime within the city boundary. The resulting computational burden can be high, with $O(10^7)$ evaluations of the kernel function. For example, in the experiments described later in this paper, we build a grid of 15,574 cell points, and for one month of training theft incidents, we would have around 4,500 incident points. Therefore, assuming that the bandwidth is given, a KDE operation would require around 140 million non-basic operations (i.e., L-2 norm and kernel function). In practice, the incident data are downsampled to reduce P and produce a feasible execution time at the expense of losing data.

B. Localized Kernel Density Estimation

Inspired by concepts from image processing, specifically convolution filtering, we propose an alternative method for performing KDE that significantly reduces computational complexity, facilitates learning of a bandwidth and a non-standard kernel function, and improves the resulting estimation performance for crime prediction. Convolution filtering can be used to perform a number of basic operations such as edge detection, corner detection, and sharpening [21], and the same concepts have been used as part of convolutional neural networks [14]. In convolution filtering, we define a kernel (also referred to as convolution matrix or mask), and we apply this kernel at each pixel of the image such that the new pixel value equals a weighted average of surrounding pixel values. In the context of our problem, one can think of the study region as an image with each grid cell being a pixel and each pixel value being the number of crime incidents in each cell. Thus, a blurring convolutional operation would result in a localized density estimate from the surrounding cells.

Figure 1 illustrates our localized kernel density estimation (LKDE) approach. In comparison with the traditional KDE, the size of the convolution kernel is similar to the concept of bandwidth while the weighted average is conceptually similar to the interpolation method. However, in the LKDE, the interpolation can vary with respect to each cell depending on the corresponding kernel weight, which produces a more flexible and non-standard interpolation. Moreover, the proposed approach significantly reduces the computational cost since its complexity depends on the kernel size rather than the number of incidents as in the KDE. Returning to our example grid and crime data above, given an 11-by-11 kernel, an LKDE would require approximately 1.9 million basic operation versus the 140 million required for a standard KDE. Figure 2 shows two sample hotspot maps produced using KDE and LKDE given the same training and testing data (the latter indicated with dark, filled circles).

The approach described above requires decisions regarding kernel size and convolution values. In the following section, we present an approach for learning a dynamic-sized and dynamic-valued kernel via genetic algorithms.

C. Learning Convolutional Kernels for LKDE

The success of LKDE depends on the quality of the convolutional kernels. Since kernels can be of different sizes and shapes, and their values can span a wide range, we require

[1] Default search radius (bandwidth) algorithm. Retrieved July 18, 2016, from http://pro.arcgis.com/en/pro-app/tool-reference/spatial-analyst/how-kernel-density-works.htm.

[2] Hpi function, ks package, p10. Retrieved July 18, 2016, from https://cran.r-project.org/web/packages/ks/ks.pdf.

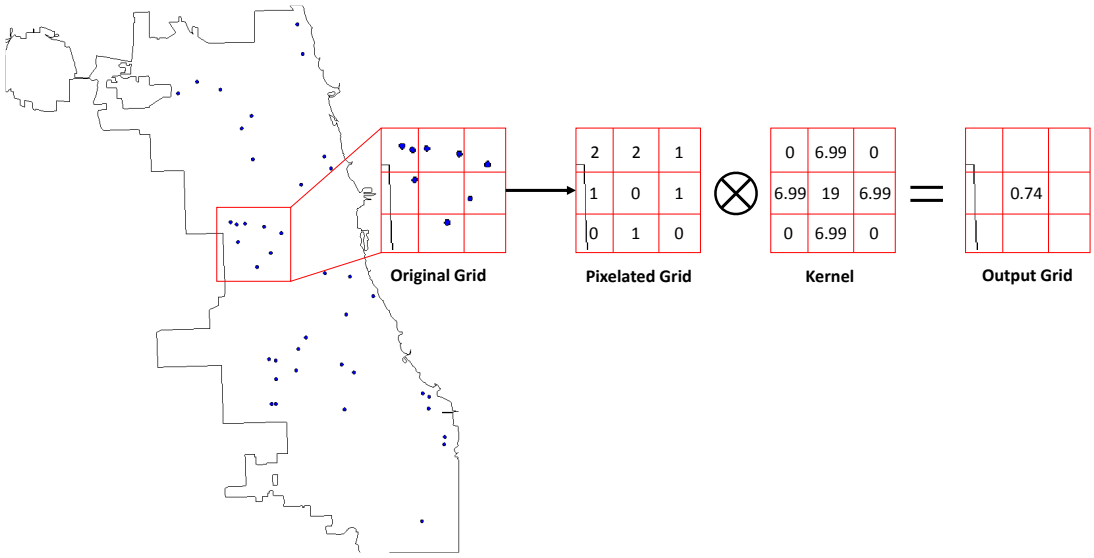


Fig. 1: An illustration of our localized kernel density estimation approach using convolution filtering. In this illustration, we show the map of Chicago, Illinois with assault incidents shown as blue points. The LKDE process involves (1) building an overlay grid, (2) counting the frequency of incidents within each grid cell, (3) fixing the center of a convolution kernel at each cell, and (4) performing a convolutional operation. The final output represents the density estimate and will be later used to build hotspot maps. Note that zero-padding is required for grid cells on the boundary.

a learning algorithm to identify optimal kernels. However, the solution space is large and non-convex. Therefore, we developed an evolutionary algorithm, specifically a genetic algorithm (GA), to learn the convolutional kernels. GAs require the following elements.

1) Encoding: Kernels can be encoded a various ways. One way is to encode each kernel value separately. For example, the 3-by-3 kernel in Figure 1 would be encoded as a 9-numbered chromosome. However, given the sparseness of the training data and the fact that kernel size can vary from one situation to another, we should favor smaller chromosomes. Additionally, the roughly stable spatial distribution of crime suggests that the historical crime frequency of the current grid cell (i.e., the one to which we are applying the convolutional operation) should have the highest impact on the predicted risk of crime within that cell. As we move away from the current cell, historical crime frequencies should have less impact on the convolutional operation. Formally, we build exponentially decaying kernels using the following decay process:

$$N_d = N_0 e^{-\lambda d} \tag{2}$$

where N_0 is the value of the central grid cell (19 in the kernel of Figure 1), d is Manhattan distance from the central cell to the current cell, and λ is the decay rate. This setup permits the construction of dynamic-sized convolutional kernels using only three parameters: initial value, largest considered distance ($\widetilde{d}$), and the decay rate. Each chromosome encodes these three parameters, and the optimization task is to identify the best parameterization for the observed crime distribution. Figure 3 shows examples of a 3-by-3 and a 5-by-5 kernel such that $N_0 = 19$, $\lambda = 1$, and $\widetilde{d}$ equals 1 and 2, respectively. Finally, we encode each chromosome as a binary vector such that each parameters is encoded using a fixed number of bits. For example, if we represent the initial value, N_0, using five bits, then the initial value can take a value between 0 and 31.

2) Fitness Function: The fitness of each chromosome is calculated as the area under the curve (AUC) of the crime surveillance plot for the prediction [12].

3) Selection Scheme: We adopted a traditional roulette wheel selection scheme in which the probability of selecting a chromosome for the next iteration depends on its fitness value:

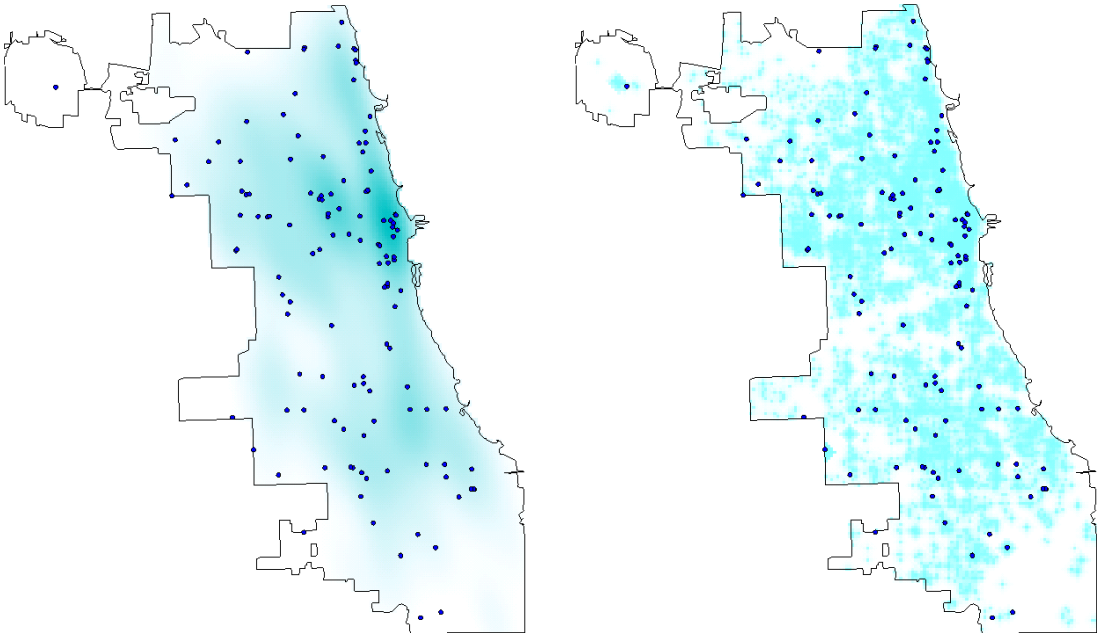
$$p_i = \frac{f_i}{\sum_{j=1}^{M} f_j} \tag{3}$$

where f_i is the fitness of chromosome i, M is the population size, and p_i is the selection probability of chromosome i.

4) Crossover Strategy: We randomly select two chromosomes and an index. Then, the offspring are generated from the two chromosomes with values swapped at the selected index. Each chromosome will fix a subset of the bits, p, and swap the remaining subset $(1-p)$ from the other chromosome.

5) Mutation Strategy: We randomly choose a chromosome and an index. Then, we generate a random value between 0 and 1. Finally, in order to produce the mutated offspring, we set the bit at the chosen index to 1 if the generated number is greater than 0.5; and 0 otherwise.

6) Stopping Condition: We used a maximum number of iterations strategy as the stopping condition. After reaching that number, we stop the GA and return the best chromosome.



(a) Hotspot map produced by KDE.

(b) Hotspot map produced by LKDE.

Fig. 2: Two hotspot maps generated by KDE and LKDE. Darker shading represents higher density estimates and predicted risk. LKDE produces a less smooth risk surface because only training points in the immediate vicinity of each cell are used for the estimation. Note that blue points are the true incident points from the testing window.

0	6.99	0
6.99	19	6.99
0	6.99	0

(a) 3x3 kernel

0	0	2.57	0	0
0	2.57	6.99	2.57	0
2.57	6.99	19	6.99	2.57
0	2.57	6.99	2.57	0
0	0	2.57	0	0

(b) 5x5 kernel

Fig. 3: Example exponentially decaying kernels with $N_0 = 19$, $\lambda = 1$, and $\widetilde{d}$ set to 1 and 2 for Sub-Figure a and b, respectively.

IV. Experiments and Evaluation

A. Data Sources

We evaluated the KDE and LKDE methods on historical crime data obtained from the City of Chicago data portal.[3] These data are extracted from the Chicago Police Department's Citizen Law Enforcement Analysis and Reporting (CLEAR) system. This system includes up-to-date criminal records representing the current judgement of the criminal justice system. For example, if a crime is recorded within CLEAR, but subsequent investigation determines that no crime actually occurred, then the record will be removed. The data included 196,347 incidents occurring between 2013-07-28 and 2014-04-14. We extracted the latitude-longitude location at block-level resolution, the timestamp, and the type of each crime incident. The data covered 17 crime types, which are shown in Table I along with per-type frequencies.

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

Crime Type	Frequency(%)
THEFT	46,767 (23.82%)
BATTERY	34,509 (17.58%)
NARCOTICS	22,209 (11.31%)
CRIMINAL DAMAGE	19,588 (9.98%)
OTHER OFFENSE	11,731 (5.97%)
BURGLARY	11,642 (5.93%)
ASSAULT	11,357 (5.78%)
DECEPTIVE PRACTICE	9,278 (4.73%)
MOTOR VEHICLE THEFT	7,643 (3.89%)
ROBBERY	7,602 (3.87%)
CRIMINAL TRESPASS	5,441 (2.77%)
WEAPONS VIOLATION	1,987 (1.01%)
PUBLIC PEACE VIOLATION	1,886 (0.96%)
OFFENSE INVOLVING CHILDREN	1,555 (0.79%)
SEX OFFENSE	1,379 (0.70%)
PROSTITUTION	937 (0.48%)
INTERFERENCE WITH PUBLIC OFFICER	836 (0.43%)
Total	196,347

TABLE I: Frequency of historical crime types in Chicago, covering 2013-07-28 through 2014-04-14.

B. Evaluation Metrics

Researchers have proposed a number of evaluation metrics [9, 12, 15] for crime prediction. Chainey et al. proposed the Predictive Accuracy Index (PAI), which is calculated as the hit rate (the ratio of incidents occurring within hotspots to the

total number of incidents) divided by the percentage of all area covered by the hotspots [9]. For example, if the selected hotspots covered 20% of the study region and 30% of future crime fell into these hotspots, the PAI would be $\frac{30\%}{20\%} = 1.5$. PAI is thus higher when more crime is present within a smaller set of predicted hotspots. Although intuitive, PAI relies on the selection of an area-coverage parameter for the hotspots. This selection can be difficult to justify, and it is not straightforward to visualize and relate PAI scores for multiple area-coverage selections. More practically, if it turns out that the police have resources sufficient to patrol an area that is smaller or larger than the selected area-coverage size, then the PAI that was calculated will not be an accurate indication of performance when guided by the hotspot map.

Surveillance plots generalize the PAI to include performance at all area-coverage values, which are depicted in a series (see Figure 4 for examples). The x-values in this series are the area-coverage percentages, and the y-values in this series are the hit rates associated with the x% "hottest" area as predicted by the model. For a given x-y coordinate in a surveillance plot, the PAI is thus $\frac{y}{x}$. Surveillance plots support a natural resource allocation process: If public safety officials determine that they have patrol resources sufficient to cover x% of their area, the surveillance plot presents the anticipated crime hit-rate (y) at this coverage as well as potential hit-rate gains and losses resulting from more and less investment in the patrol activity (moving right and left along the surveillance series). We calculate the area under the curve (AUC) as a scalar summary of performance in order to rank multiple surveillance plots. Formally, the AUC is defined as

$$AUC = \sum_{i=1}^{N} \left((\widetilde{x}_i - \widetilde{x}_{i-1}) * \frac{\widetilde{y}_i + \widetilde{y}_{i-1}}{2} \right), \qquad (4)$$

where $\widetilde{x}$ is the cumulative percentage of most threatened area predicted by the model, $\widetilde{y}$ is the accumulative percentage of testing incidents captured within the surveilled area such that $\widetilde{x}_0 = \widetilde{y}_0 = 0$, and N is the total number of performance points included in the surveillance plot. For example, each surveillance plot in Figure 4 include $N = 100$.

C. Experimental Setup

Our experimental setup for comparing KDE and LKDE models was structured as follows: (1) We created a grid of 200-meter squares covering the official city boundary of Chicago; (2) we enumerated 70 random testing days between 2013-08-28 and 2014-04-14 for each crime type; (3) for each testing day and crime type, we fitted KDE and LKDE models using crime data prior to the testing day; (4) we used each fitted model to predict hotspots for the associated testing day, and we aggregated all testing days for a model and crime type following the process described by Gerber [12].

For example, to evaluate the burglary KDE model on the testing day of 2013-09-29, we used all burglary incidents between 2013-8-29 and 2013-09-28. We used the `kde` function from the `ks` package in R, optimizing the bandwidth with the `Hpi` heuristic. We then computed the surveillance plot for the fitted KDE using burglaries occurring on 2013-09-29. After repeating this process for the 70 testing days we aggregated the surveillance plots to arrive at a final evaluation of the KDE model on the burglary crime type. We then repeated this process for each crime type.

Estimating the LKDE proceeded similarly to the example above, except that prior to computing the density estimate we optimized an exponentially decaying convolutional kernel using a GA. We used the final three days of the training data for evaluating the fitness of chromosomes (kernel size, decay rate, and interpolation values), and we used one month prior to these days for training during GA optimization. For example, for the testing day 2013-09-29, we optimized the convolution kernel using a GA trained on incidents between 2013-8-26 and 2013-09-25 and evaluated the chromosomes on burglaries between 2013-09-26 and 2013-09-28. We set the population size to 50, we used 50% crossover rate, 5% mutation rate, and we set the maximum number of iterations to 30. The optimal chromosome (convolution kernel) was then used to predict burglaries for the testing day 2013-09-29. Lastly, we aggregated the resulting surveillance plots as described above to get a final surveillance plot for burglary, which was compared with the aggregated plot for KDE. Note that in order to match the experimental settings for both KDE and LKDE, we evaluated the `Hpi` function on the same data points used in the GA: one month prior to three days from the testing day.

D. Experimental Results

Table II shows the aggregated surveillance plot AUC over 70 prediction days by crime type and model, sorted by decreasing crime frequency. LKDE outperformed KDE for 12 out of 17 crime types, often achieving substantial gains. For example, the LKDE model captures 25% more criminal incidents (absolute) than the KDE at the same 0.01 area-coverage rate. The KDE often outperformed LKDE on infrequent crime types, which constitute less than 4% of all crime.

Given the fact that public safety officials can only cover a small percentage of the study region, we analyzed the performance gain over various area-coverage rates. For example, Table II shows that by using LKDE, we gain 4% AUC over the KDE model for burglary and 1% AUC over the KDE model for prostitution. These results might suggest that LKDE achieves greater gains for burglary than for prostitution. However, these gains are computed at 100% area-coverage, which is certainly not a feasible coverage rate in practice. If instead we examine the x-value location of peak gain achieved by the LKDE model over the KDE model (Figure 4), we find that, for burglary, peak gain of 9% is achieved at 25% area coverage, versus a peak gain of 25% being achieved at 1% area coverage for prostitution. These results indicate that the practically achievable gain for LKDE is much greater for prostitution than for burglary. Table II shows the peak gain for the LKDE model over the KDE model in the fourth and fifth columns. These results indicate not only the effectiveness of LKDE in

Crime Type	Overall Performance KDE	LKDE	Δ-Peak X	Y	Average Kernel Size
THEFT	0.7156	**0.7860**	0.19	14.20%	17x17
BATTERY	0.7269	**0.7923**	0.18	13.30%	16x16
NARCOTICS	0.8331	**0.8749**	0.08	12.27%	16x16
CRIMINAL DAMAGE	0.6725	**0.7275**	0.22	10.34%	17x17
OTHER OFFENSE	0.6937	**0.7298**	0.13	7.72%	16x16
BURGLARY	0.6998	**0.7403**	0.25	8.92%	18x18
ASSAULT	0.7204	**0.7559**	0.13	8.65%	19x19
DECEPTIVE PRACTICE	0.7433	**0.7657**	0.02	8.01%	19x19
MOTOR VEHICLE THEFT	0.6987	**0.7125**	0.13	3.77%	20x20
ROBBERY	0.7463	**0.7768**	0.22	7.73%	21x21
CRIMINAL TRESPASS	0.7592	**0.7991**	0.05	15.37%	20x20
WEAPONS VIOLATION	**0.7826**	0.7764	0.11	4.98%	24x24
PUBLIC PEACE VIOLATION	**0.7401**	0.7342	0.06	6.88%	22x22
OFFENSE INVOLVING CHILDREN	**0.6837**	0.6722	0.04	4.07%	23x23
SEX OFFENSE	**0.6745**	0.6627	0.18	4.04%	35x35
PROSTITUTION	0.8458	**0.8552**	0.01	25.22%	17x17
INTERFERENCE WITH PUBLIC OFFICER	**0.7889**	0.7810	0.09	4.70%	22x22

TABLE II: Performance on 70 random prediction days with respect to the 17 crime types along with the peak gains of using the LKDE instead of the KDE. The x-values in column four indicate area coverage and the y-values indicate gains achieved by surveilling $x\%$ according to the LKDE model instead of the KDE model.

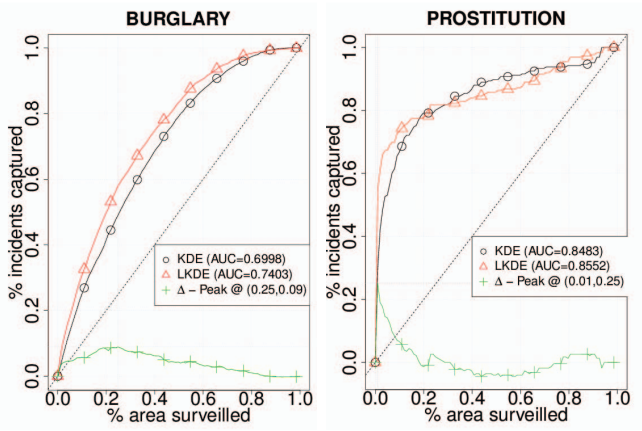


Fig. 4: Aggregated surveillance plots of burglary and prostitution crimes. The gain series, shown in green, is computed as LKDE-KDE. The highest peak gain occures at the location specified by $\Delta - Peak$.

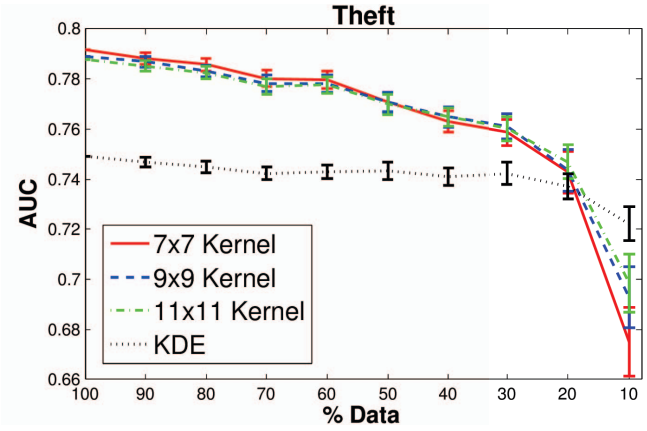


Fig. 5: Performance of KDE and three fixed-size LKDEs on randomly choosen subsets of theft incidents.

outperforming the KDE but also that these gains are often achieved for low area-coverage values (i.e., at small x-values), increasing the likelihood of practical realization.

Finally, we calculated the average kernel size learned by the GA across the 70 days (final column in Table II). We observed that with sparser data (bottom rows), the optimized kernels become larger, taking more of the surrounding area into account when estimating the risk at the interpolation point. We believe that this flexibility in kernel size is a primary advantage of the LKDE method. It allows the LKDE to take advantage of dense data when possible, without affecting prediction in areas of sparse data.

E. Robustness Analysis

To better understand how data sparseness affects KDE and LKDE performance, we performed a robustness analysis on theft, which is the most common crime type in the dataset. The analysis proceeded as follows: (1) We chose four random testing days; (2) for each testing day, we fixed the size of the convolutional kernel in the GA and gradually removed a percentage of random observations from the training data; (3) we learned three kernels of sizes 7x7, 9x9, and 11x11. We also performed KDE on the same data, estimating the bandwidth using the `Hpi` procedure; (4) we evaluated the KDE and LKDE models on the testing days, computing AUC scores from the aggregated surveillance plots. We then replicated these steps 10 times using different random seeds for the ablation. Figure 5 presents the results. We observed the following: (1) With more data, smaller kernels outperform larger ones; (2) larger kernels are more resilient to data

sparseness; (3) when dealing with sparse data, we observe higher variance among different evaluations; and (4) although the KDE uses dynamic bandwidth, it is still unable to perform well on sparse data. Observations 1 and 2 corroborate our observation regarding kernel sizes from Table II.

V. Conclusions and Future Work

We have proposed a novel approach for predicting hotpots using localized kernel density estimation optimized by a genetic algorithm. Our model was inspired by concepts from image processing, and it addresses two primary limitations of traditional kernel density estimation: computational complexity and kernel function choice. Our LKDE method is able to leverage dense data when available and will typically produce focused, more accurate convolution kernels in such situations. As data become sparse, the LKDE method adapts by enlarging the kernel to leverage additional data. Our experiments on real crime data of 17 types from Chicago shows the effectiveness of our approach versus traditional KDE. Peak gains reach 25% (absolute) versus KDE, with many of these gains realized at small area-coverage rates. We have taken a preliminary look at robustness to sparsity, and the results support the use of adaptable convolution kernels.

In the future, we plan to (1) investigate the use of personalized modeling approaches to facilitate learning of different kernels for various sections of the study region; (2) use deep neural networks to learn the localized convolutional kernels; and (3) deploy the LKDE within more comprehensive predictive models that also account for spatial, temporal, and social media data similar to what has been reported by [12].

References

[1] M. Al Boni and M. S. Gerber. Predicting crime with routine activity patterns inferred from social media. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2016.

[2] T. C. Bailey and A. C. Gatrell. *Interactive spatial data analysis*, volume 413. Longman Scientific & Technical Essex, 1995.

[3] K. J. Bowers, S. D. Johnson, and K. Pease. Prospective hot-spotting the future of crime mapping? *British Journal of Criminology*, 44(5):641–658, 2004.

[4] A. A. Braga. Hot spots policing and crime prevention: A systematic review of randomized controlled trials. *Journal of experimental criminology*, 1(3):317–342, 2005.

[5] J. M. Caplan and L. W. Kennedy. *Risk terrain modeling manual: Theoretical framework and technical steps of spatial risk assessment for crime analysis*. Rutgers Center on Public Security, 2010.

[6] J. M. Caplan and L. W. Kennedy. Risk terrain modeling compendium. *Rutgers Center on Public Security, Newark*, 2011.

[7] S. Chainey. Examining the influence of cell size and bandwidth size on kernel density estimation crime hotspot maps for predicting spatial patterns of crime. *Bulletin of the Geographical Society of Liege*, 60:7–19, 2013.

[8] S. Chainey and J. Ratcliffe. *GIS and crime mapping*. John Wiley & Sons, 2013.

[9] S. Chainey, L. Tompson, and S. Uhlig. Utility of hotspot mapping for predicting spatial patterns of crime. *Security Journal*, pages 4–28, 2008.

[10] J. Eck, S. Chainey, J. Cameron, and R. Wilson. Mapping crime: Understanding hotspots. Technical report, National Institute of Justice, 2005.

[11] M. Felson. Those who discourage crime. *Crime and place*, 4:53–66, 1995.

[12] M. S. Gerber. Predicting crime using twitter and kernel density estimation. *Decision Support Systems*, 61:115–125, 2014.

[13] T. Hart and P. Zandbergen. Kernel density estimation and hotspot mapping: examining the influence of interpolation method, grid cell size, and bandwidth on crime forecasting. *Policing: An International Journal of Police Strategies & Management*, 37(2):305–323, 2014.

[14] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[15] N. Levine. The "hottest" part of a hotspot: comments on" the utility of hotspot mapping for predicting spatial patterns of crime". *Security journal*, 21(4):295–302, 2008.

[16] S. McLafferty, D. Williamson, and P. G. McGuire. Identifying crime hot spots using kernel smoothing. *Analyzing crime patterns: Frontiers of practice*, pages 77–85, 2000.

[17] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[18] G. Pezzuchi. A brief commentary on "the utility of hotspot mapping for predicting spatial patterns of crime". *Security journal*, 21(4):291–292, 2008.

[19] B. A. Reaves. Local police departments. *Bureau of Justice Statistics*, 12 2010.

[20] M. Rosenblatt et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[21] L. Shapiro and G. C. Stockman. Computer vision. 2001. *ed: Prentice Hall*, 2001.

[22] M. Wand and M. Jones. Multivariate plug-in bandwidth selection. *Computational Statistics*, 9(2):97–116, 1994.