Learning Task-Specific City Region Partition

Hongjian Wang*
Twitter Inc.
hongjianw@twitter.com

Porter Jenkins
Penn State
prj3@ist.psu.edu

Hua Wei
Penn State
hzw77@ist.psu.edu

Fei Wu
Penn State
fxw133@psu.edu

Zhenhui Li
Penn State
jessieli@ist.psu.edu

ABSTRACT

The proliferation of publicly accessible urban data provide new insights on various urban tasks. A frequently used approach is to treat each region as a data sample and build a model over all the regions to observe the correlations between urban features (e.g., demographics) and the target variable (e.g., crime count). To define regions, most existing studies use fixed grids or pre-defined administrative boundaries (e.g., census tracts or community areas). In reality, however, definitions of regions should be different depending on tasks (e.g., regional crime count prediction vs. real estate prices estimation). In this paper, we propose a new problem of task-specific city region partitioning, aiming to find the best partition in a city w.r.t. a given task. We prove this is an NP-hard search problem with no trivial solution. To learn the partition, we first study two variants of Markov Chain Monte Carlo (MCMC). We further propose a reinforcement learning scheme for effective sampling the search space. We conduct experiments on two real datasets in Chicago (i.e., crime count and real estate price) to demonstrate the effectiveness of our proposed method.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**; **Data mining**.

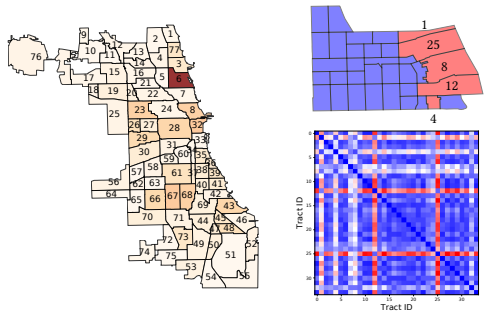
KEYWORDS

Region partition; reinforcement learning; crime prediction

ACM Reference Format:
Hongjian Wang, Porter Jenkins, Hua Wei, Fei Wu, and Zhenhui Li. 2019. Learning Task-Specific City Region Partition. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313704

1 INTRODUCTION

Given the growing amount of urban data, a number of data-driven models have been developed to provide insights for urban problems. A common approach is to treat each region as a data sample, take the region properties as features, and build a model to learn the correlation between region features and a target variable. Crime prediction is a good example. Criminologists are interested in knowing the correlation between demographics and crime [17, 21]. Each region i is taken as a data sample, with X_i as its demographic feature and Y_i as its crime count. A model (e.g., linear regression or negative binomial model) is built to estimate crime count vector Y using the feature matrix X. If some features (e.g., disadvantage index) show a significant correlation with crime count, researchers could relate this empirical result with criminology theory [8] and policy makers could further propose corresponding policies to address crime issues.



(a) Prediction errors (b) Community #6

Figure 1: (a) Heatmap of crime prediction error at community level in Chicago. Darker color means higher prediction error. The community area #6 is an outlier with the largest error. (b) Community #6 consists of 34 tracts (top) and their pair-wise similarity (bottom). It is clear that 5 tracts (red) on the east side are different from others.

Existing studies often use pre-defined administrative boundaries (e.g., street block, census tract, or community area) to define a region [17, 24]. A plot of 77 administrative community areas of Chicago is shown in Figure 1. However, such administrative boundaries might be too rigid and do not reflect the true spatial regions with regards to the targeted urban issues. One can alternatively propose to study the problem at the point level or small grid cell level (e.g., 10 meters by 10 meters grids). However, this could lead to data sparsity issues and jeopardize the integrity of the statistical model. Consequently, any inference made from the model would be at risk of bias. These concerns have both theoretical and policy implications. We use the following example to further illustrate the issue of using pre-defined community areas to study crime.

Example 1.1. Following previous work [17], we replicate a negative binomial model to predict crime count using demographic features by treating each community area as a data sample. Figure 1 plots the crime prediction error for each community area of Chicago. Community area #6 (i.e., Lake View area) shows an abnormally high error. In order to explain this outlier, we further investigate the internal structure of this area. Community #6 consists of 34 census tracts as shown in the top of Figure 1(b). In the bottom of Figure 1(b), we visualize pair-wise Euclidean distance between tracts based on demographic features. There are five tracts

*The work was done when the author was a student at Penn State University.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313704

that are different from others, and they are all located on the east side of community #6. These five tracts, when mixed with other tracts in this community, lead to inferior performance of crime count prediction in that area.

The observation above motivates us to learn a better region partition for crime study. In this paper, we propose a new problem of *task-specific region partitioning*. Given spatial variables and a selected model (e.g., linear regression), we aim to partition the city into regions such that the model trained by taking regions as data samples achieves optimal results.

Task-specific city region partitioning is a challenging problem. The key challenge lies in that the region properties (both features and target variable) and the model coefficients change simultaneously when we change the region partition. We prove that this is an NP-hard problem. We adopt a stochastic sampling approach to tackle our NP-hard problem. We start from a pre-defined region partition (e.g., community areas), and generate a new partition sample by flipping the membership of a smaller area (e.g., a tract). First, two variants of MCMC methods are proposed to generate new samples. Finally, we employ reinforcement learning to automatically learn how to generate the next sample that is more likely to improve the prediction performance.

We evaluate our method on two real datasets, i.e. crime counts and real estate prices. The learned region partitions are shown to consistently outperform the administrative boundaries and spatial clustering method. For example, our methods, on average, outperform the administrative boundary by 67% in MAE for a crime count prediction task.

To summarize, the contributions of this paper are three fold. First, we propose a novel problem on task-specific region partitioning, which is motivated by real-world urban studies [17]. Second, we prove the problem is NP-hard and use MCMC and reinforcement learning sampling strategies to solve the problem. Finally, we validate our method through extensive experiments on two real datasets.

2 REGION PARTITION PROBLEM

The input of our problem is a set of minimum spatial units. Without loss of generality, we use tract as the unit of study in this paper, and the set of tracts is denoted as $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$. Within each tract t_i, the following information are available $\langle p_i, y_i, x_i \rangle$, where p_i is a sequence of GPS coordinates representing tract exterior boundary, y_i is the target variable of interest, such as crime count and average house price, and $x_i \in \mathcal{R}^d$ is a d-dimension contextual feature vector. Note that each tract is a polygon with one connected component.

A city is partitioned into different community areas. We use community area as the proper unit to study the correlation between y and x. Each community area consists of several adjacent tracts, denoted as $Z_j = \{t_1^j, t_2^j, \cdots\}$. We derive $\langle P_j, Y_j, X_j \rangle$ for each community area Z_j by aggregating $\{\langle p_i, y_i, x_i \rangle | t_i \in Z_j\}$, where P_j is the exterior boundary, Y_j is the prediction target, and X_j is the contextual features of Z_j.

Definition 2.1 (Partition). A partition over $\mathcal{T}$ consists of a set of m non-overlapping community areas, denoted as $\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_m\}$, satisfying the following four conditions

(1) (subset) $\forall j, Z_j \subset \mathcal{T}$;
(2) (non-overlapping) $\forall p, q, Z_p \cap Z_q = \emptyset$;
(3) (completeness) $\bigcup_{j=1}^{m} Z_j = \mathcal{T}$;
(4) (spatial-continuity) $\forall j$, P_j defines a polygon with exact one connected component.

A task is to learn a model f that takes input X and predicts Y on a given partition $\mathcal{Z}$. In order to learn a good model f, task-specific region partition problem is defined as follows.

Definition 2.2 (Task-specific region partition). Given a set of tracts $\mathcal{T}$ and a task f, find a partition $\mathcal{Z}$ with m components, such that the task error is minimized. Formally, we have

$$\arg\min_{\mathcal{Z}, f} \sum_{j=1}^{m} \Big(||Y_j - f(X_j)||_2 + G(Z_j) \Big), \tag{1}$$

where $G(\mathcal{Z}) = \sum_{j=1}^{m} G(Z_j)$ is a constraint on partition. For simplicity, we use $\mathcal{F}(\mathcal{Z}, f) = \sum_{j=1}^{m} \mathcal{F}(Z_j, f)$ as the quality measure of a partition $\mathcal{Z}$, where $\mathcal{F}(Z_j, f) = ||Y_j - f(X_j)||_2 + G(Z_j)$. Hence, the objective becomes

$$\arg\min_{\mathcal{Z}, f} \mathcal{F}(\mathcal{Z}, f). \tag{2}$$

The constraint function G ensures the partition $\mathcal{Z}$ has desirable property, such as small variance in community populations or balanced size in terms of community area.

NP-Hardness. The problem in Definition 2.2 is a combinatorial optimization problem. The decision version of the problem is to find a partition $\mathcal{Z}$, such that $\mathcal{F}(\mathcal{Z}, f) \leq \epsilon$, where ϵ is a constant. Here we prove such decision problem is NP-complete, and therefore the optimization version is NP-hard.

In the NP-completeness proof, we first approximate the decision problem above with an easier problem. The fact, that X_i, Y_i, and f are dynamically changing according to $\mathcal{Z}$, complicates the original problem. Therefore, we replace the jointly learned optimal f with a fixed f_0. Since f minimizes Equation 2 while f_0 does not, we have $\mathcal{F}(\mathcal{Z}, f) < \mathcal{F}(\mathcal{Z}, f_0)$. Next, we use the largest task error on one community area $\sup\{\mathcal{F}(Z_j, f_0)\}$ to approximate the overall task error $\mathcal{F}(\mathcal{Z}, f_0)$, where sup calculates supremum of a set. Namely, $\mathcal{F}(\mathcal{Z}, f) < \mathcal{F}(\mathcal{Z}, f_0) < m \cdot \sup\{\mathcal{F}(Z_j, f_0)\}$, because there are m community areas. Therefore, if we prove $\exists \mathcal{Z}$, such that $m \cdot \sup\{\mathcal{F}(Z_j, f_0)\} \leq \epsilon$, then we have $\mathcal{F}(\mathcal{Z}, f) \leq \epsilon$.

Now we prove the approximated decision problem is NP-complete. First, such approximated decision problem is NP, because given a partition $\mathcal{Z}_0$, we are able to validate $m \cdot \sup\{\mathcal{F}(Z_j, f_0)\} \leq \epsilon$ in polynomial time. Next, we prove the NP-hardness of this problem by reducing the (k, v)-balanced partitioning problem to the approximated decision problem. The (k, v)-balanced partition problem [2] is a proved NP-complete problem, which partition graph into k disjoint components of size at most $v\frac{n}{k}$, while the capacity of edge cut is less than ϵ. We construct the adjacency graph of all tracts $\mathcal{T}$, with weight $\sup\{\mathcal{F}(Z_j, f_0)\}$ on each edge. A solution to (m, v)-balanced partition problem on such adjacency graph is a solution to the approximated decision problem. The balanced partition problem achieve $k \cdot \sup\{\mathcal{F}(Z_j, f_0)\} \leq \epsilon$, where k is the number of edges to cut the graph. It is clear that $k > m$, and therefore we find a partition satisfying $m \cdot \sup\{\mathcal{F}(Z_j, f_0)\} \leq \epsilon$.

3 METHODS

We employ a stochastic search process to find approximate solutions due to the NP-hard nature of our problem. First, we propose two variants of Markov Chain Monte Carlo (MCMC) method with different sample proposal strategy. Then, we use reinforcement learning to automatically learn the sample strategy.

3.1 Markov Chain Monte Carlo

The optimal task function f in Equation (2) can be easily learned, when a partition $\mathcal{Z}$ is given. Therefore, the challenge lies in searching through the partition space. Toward this goal, we first adopt the MCMC method, or more specifically the Metropolis-Hastings algorithm to optimize $\mathcal{Z}$.

Markov Chain Monte Carlo [3] is a stochastic algorithm that constructs a Markov chain that will ultimately converge to p through stochastic sampling [11]. In our case, the state space is all possible partitions $\mathcal{Z}$, and the distribution $p(\mathcal{Z})$ defines how likely that $\mathcal{Z}$ is optimal. Clearly, it is difficult to calculate $p(\mathcal{Z})$, but a partition $\mathcal{Z}$ with lower $\mathcal{F}(\mathcal{Z})$ value is more likely to be optimal.

In addition to the quality function $\mathcal{F}$, MCMC employs a proposal function $q(\mathcal{Z}'|\mathcal{Z})$, which defines the transition probability from state $\mathcal{Z}$ to $\mathcal{Z}'$. The Markov chain moves toward $\mathcal{Z}'$ with acceptance probability $\gamma = \min\left[1, \frac{p(\mathcal{Z}')q(\mathcal{Z}|\mathcal{Z}')}{p(\mathcal{Z})q(\mathcal{Z}'|\mathcal{Z})}\right]$. And $p(\mathcal{Z}) = \frac{e^{-\mathcal{F}(\mathcal{Z})/T}}{P}$ is the Boltzmann distribution, where P is the normalization constant, and T is the temperature parameter. The proposal function q is very flexible, if not limitless. In practice, the choice of q generally affects how quickly the algorithm converges.

The MCMC algorithm is described below. First, we initialize the partition with the existing administrative boundary, denoted as $\mathcal{Z}_0$. Within each step, we draw $u \in [0, 1]$ from uniform distribution $\mathcal{U}_{[0,1]}$, draw the next partition $\mathcal{Z}'$, and calculate acceptance probability γ. Only if u is smaller than γ, we accept the new partition $\mathcal{Z}'$. We repeat the process above, until $\mathcal{F}(\mathcal{Z})$ converges.

MCMC with Naive Proposal. We begin by establishing a baseline MCMC with the simplest q we can devise. Namely, we generate a new partition by *uniformly randomly* selecting one tract $t_i \in \mathcal{T}$ that is on the boundary of some community area Z_j, and then flip t_i to the adjacent community area. With naive proposal strategy, q is symmetric, because $q(\mathcal{Z}'|\mathcal{Z})$. Therefore, the acceptance probability $\gamma = \min[1, \frac{p(\mathcal{Z}')}{p(\mathcal{Z})}]$.

Guided MCMC with Softmax Proposal. We propose another MCMC approach with a more intelligent proposal strategy than uniformly random selection. This approach is a greedy strategy that we prefer to adjust the community area with the highest prediction error to improve current partition.

Given this intuition, we heuristically design our guided MCMC method to sample a community area with large error first. To achieve this, we apply the softmax function over the prediction errors on different community areas to derive the sample probability of each community area, i.e. $p(Z_j) = \frac{\exp(||Y_j - f(X_j)||_2)}{\sum_{k=1}^{m} \exp(||Y_k - f(X_k)||_2)}$. Note that under the softmax proposal approach, the proposal function q is not symmetric. Therefore, we have to explicitly calculate the values for q functions.

Discussion: we acknowledge that there are plenty of other strategies to build random walks for MCMC. However, there is always a trade-off between efficiency and effectiveness. The naive MCMC is the simplest approach to conduct MCMC, while being slow to converge. The guided MCMC employs human heuristics to improve the efficiency at the expense of potentially converging to a worse local optima, because the greedy strategy may prune the search space too aggressively.

3.2 Reinforcement Learning

There are two main drawbacks of the MCMC method. First, when drawing a new sample, the MCMC methods do not account for any information from previous samples. Namely, the naive MCMC could repeatedly reject the same sample. Intuitively, all the samples we observed so far should carry certain information to assist sampling in the future. Second, the Markov chain-based stochastic search strategy is more likely to get stuck on a local optima, because the nature of MCMC sampling follows a depth first search. It is very likely that another chain leads to a better local optima, but the algorithm is not able to explore the alternative outcome.

To address these two issues of MCMC method, we propose a reinforcement learning (RL) [15] scheme for generating new samples. RL differs from standard supervised learning in that the correct input/output pairs are not presented at the same time. Instead, RL is concerned with how agents ought to take actions in an environment so as to maximize cumulative gain.

In what follows, we map the RL components to our problem. The set of tracts consists of the environment, and their community area assignment is the state. An action is to re-assign some tract t_i from Z_j to Z_p, denoted as tuple $\langle t_i, Z_p \rangle$. The immediate reward of such transition is defined by $\Delta\mathcal{F} = e^{-\mathcal{F}(\mathcal{Z}')} - e^{-\mathcal{F}(\mathcal{Z})}$, where the exponential function coverts the loss into gain. We define the cumulative gain Q as a function of the current state $\mathcal{Z}$ and action $\langle t^k, Z^k \rangle$ at step k, and Q satisfies the following condition

$$Q(\mathcal{Z}, \langle t^k, Z^k \rangle) = \Delta\mathcal{F} + \delta \cdot \sum_{a \in \{\langle t^{k+1}, Z^{k+1} \rangle\}} Q(\mathcal{Z}', a), \tag{3}$$

where δ is the discount factor on future reward and $\{\langle t^{k+1}, Z^{k+1} \rangle\}$ is the set of all possible actions given $\mathcal{Z}'$. Given Q function, at each state $\mathcal{Z}$, we are able to find the best action with the highest cumulative gain through

$$\arg\max_{\langle t, Z \rangle} Q(\mathcal{Z}, \langle t, Z \rangle). \tag{4}$$

In our problem, a reinforcement learning scheme faces the following three challenges.

Huge state space. The state space is exponential to the number of tracts. Consequently, we cannot track the exact reward for each state and actions. Instead, we rely on Deep Q-learning (DQN) [16], where a deep neural network approximates the Q function.

Our neural network architecture is shown in Figure 2. The input of our DQN model includes the partition $\mathcal{Z}$ and an action tuple $\langle t, Z \rangle$. Since the input consists of either indexes of community area or index of a tract, we employ an embedding layer to encode these positive integers into dense vectors. We concatenate $\mathcal{Z}$ and Z and apply the CA embedding layer $embed_{CA}$ to obtain d_{CA}-dimension

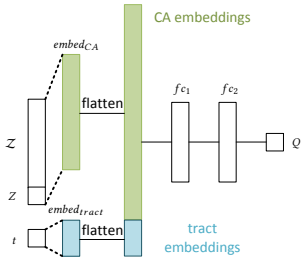


Figure 2: The Deep Q-learning neural network architecture. The input contains current partition $\mathcal{Z}$ and action $\langle t, Z\rangle$. We employ two embedding layers $embed_{CA}$ and $embed_{tract}$ for community area and tract respectively. The embeddings of all inputs are flattened and concatenated together as the intermediate representation of the partition and action. Multiple fully connected layers, $\{fc_1, fc_2, \cdots\}$, are used for predicting the final output Q.

vectors. The output from $embed_{CA}$ has a dimension of $(n+1)$-by-d_{CA}, where n is the number of tracts or the length of $\mathcal{Z}$. The index of target tract t is encoded into d_{tract}-dimension vector through tract embedding layer $embed_{tract}$. The output from $embed_{tract}$ has dimension of 1-by-d_{tract}. The embeddings of all inputs are flattened and concatenated together as the intermediate representation of the partition and action. Multiple fully connected layers, $\{fc_1, fc_2, \cdots\}$, are used for the prediction task. Finally, the output layer predicts the sigmoid of Q, which is the reward if the given action is taken on given partition. The detailed parameter setting of our DQN model is given later.

Large and dynamic action space. It is difficult to directly calculate the summation of future Q values in Equation (3), and to find the maximum over all actions in Equation (4). The reason is that for different partition states, the tracts on the boundaries are different, and thus the set of possible actions is also different. In this paper, we set the discount factor $\delta = 0$ to ease the calculation of Q. When searching for the best action with Equation (4), we sample a subset of $m = 32$ actions, and find the best action within such subset.

It is worthy to mention that when the discount factor is set to 0, the DQN only accounts for immediate reward instead of the total reward. Even so, the DQN method is still intuitively better than the MCMC method. The reason is that an MCMC approach cannot keep track of the information from rejected historical samples. The DQN method on the other hand can account for the similarity between partitions, and thus potentially avoid to repeatedly try the same action on similar partitions. The DQN is able to learn extra information from historical samples in the form of immediate reward, if not the total reward.

Training overhead is high. To train the Deep Q-learning model, we perform a sequence of random actions to generate a batch of training data. Such training overhead is significantly higher than the MCMC method. To improve the efficiency, we save our DQN model across different tasks and different rounds. The neural network is only updated when the found action cannot improve the cumulative reward.

4 EXPERIMENTS

4.1 Experiment Setting

4.1.1 Data description. The fundamental geographic unit of study in this paper is a tract, which is a small area established by the U.S. Census Bureau. Analysis at the tract level is attractive because it offers the finest level of granularity where demographic data are available.

We collect the *demographic data* of 801 tracts in Chicago from the 2010 US census survey [1]. The demographic data provide the raw count of households over 100 different categories, include ethnicity, income, education, etc. The *geographic boundary information* for 801 tracts are available through the U.S. census survey [1] as well. Each boundary is defined by a polygon with a sequence of GPS coordinates. The Chicago Data Portal [12] makes the Chicago *crime data* publicly available online. Our study focuses on data from 2010 and 2011 in which over 700,000 incidents of crime were recorded. For each crime incident, the date, location, and incident type are reported. *House Price Data* in the Chicago metropolitan area are obtained from Zillow, a popular real estate website [29]. We collect the last sale price, floor size, latitude, and longitude information for over 44, 000 properties that were sold between January 2015 and December 2017.

4.1.2 Prediction Tasks. We employ a negative binomial regression model [17] as our prediction task, namely

$$\mathbf{E}(Y) = \exp(\alpha X),$$

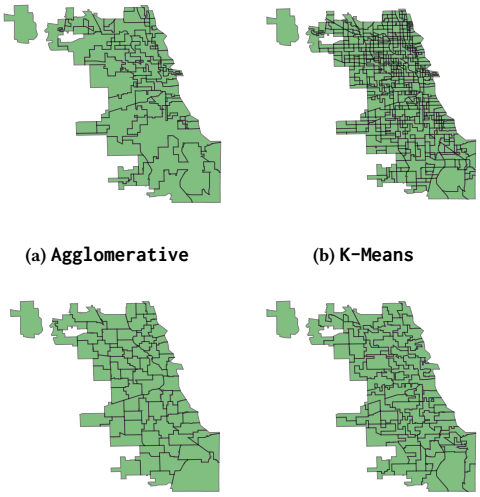
where $\mathbf{E}$ calculates the expectation. The link function used in the regression is a negative binomial distribution. The reason to use negative binomial regression is that 1) it is suitable for non-negative value prediction, and 2) it solves the over-dispersion problem by allowing the variance to be larger than the mean.

The demographics features at the community level are pre-processed as contextual features, X_i, including total population, percentage of household in each income category, diversity of ethnicity, etc.

Two prediction tasks are used to evaluate our region partition methods in the experiments. Both tasks use the same processed demographic features, X. The first task is crime count prediction, where we aggregate the total number of crime in a community as Y. The crime count in year 2010 is used as training, and 2011 data are used as testing. The second task is house price prediction, where we use the average price per square foot in a community as Y. The houses that are sold before August 1st, 2016 are used as training, while the rest are used as testing. The split point makes the training and testing data have roughly equal number of samples.

4.1.3 Compared Methods. The existing administrative boundary is a clear baseline partition. Since our region partition problem is similar to clustering problem, we employ various conventional clustering methods to derive different clustering partitions as alternative baselines. More specifically, we compare the following region partition methods.

- Administrative boundary **(`Admin`)** uses the existing administrative boundary defined by US Bureau of Census [1]. A visual depiction of this partition can be seen in Figure 1.
- Various clustering methods. **`K-Means`** clustering separates tracts into n groups of equal variance. **`Agglomerative`** clustering performs a hierarchical clustering using a bottom up approach. The ward linkage function, along with a tract adjacency graph, are used to guarantee spatial continuity. **`Spectral`** clustering does a low-dimensional embedding of the affinity matrix between



(a) **Agglomerative** (b) **K-Means**

(c) **Spectral** (d) **DQN**

Figure 3: (a - c) The clustering results for three clustering baselines. (d) The learned partition from DQN method for crime prediction task. Note that K-means generate non-connected communities, and thus smaller regions. In all cases $m = 77$.

samples first, and then applies the K-means method. Note that Spectral also takes the tract adjacency graph as the affinity graph.

- Two variants of MCMC method. **Naive** MCMC with a straightforward uniform proposal. **Softmax** MCMC with softmax proposal is another variant, which uses human heuristics.
- Q-learning is our proposed reinforcement learning method to search for an optimal partition. In our **DQN** model, the embedding dimension of community area d_{CA} is 2, while d_{tract} is 4. We employ two fully connected layers for the prediction task, where fc_1 has 200 neurons and fc_2 has 100 neurons.

Since Naive, Softmax, and DQN all learn a region partition based on a supervised task, these methods utilize information from both the feature matrix, X, and the target vector, Y. To ensure a fair comparison between these methods and the conventional clustering methods (Agglomerative, K-means, and Spectral), we run the latter under three different settings: X only, Y only, and both X and Y. In each instance, we set the number of clusters as $m = 77$, which equals the number of community areas in Admin. Note that Agglomerative is a stable clustering method and produces the exact same clustering result each time it is run. Meanwhile, the K-Means and Spectral methods do not always converge to the same partition, and therefore we run each 10 times. Similar to K-means and Spectral, our proposed Naive, Softmax, and DQN may converge to different local optimal partitions, and thus we run our methods 10 times as well. Finally, we compute the mean and standard deviations of their corresponding error measures.

4.1.4 Evaluation Metrics. Given a partition $\mathcal{Z}$, we evaluate the quality of this partition on the testing data set. Mean Absolute Error (MAE) and standard deviations are used to measure the performance of prediction tasks. We compute MAE metric using Leave-one-out Cross-validation.

Table 1: MAE and corresponding standard deviation of various partition methods, averaged over 10 runs. For all deterministic methods, we did not report the standard deviation.

Method	Crime	House price
Admin	1,826.84	31.29
Agglomerative X only	60,126.28	51.28
Agglomerative Y only	14,069.28	41.87
Agglomerative X and Y	147,465.30	51.32
K-means X only	2,513.35 (433.74)	31.74 (1.17)
K-means Y only	1,048.59 (99.47)	32.75 (1.03)
K-means X and Y	2,779.78 (307.68)	32.98 (1.76)
Spectral X only	1,373.07 (36.82)	30.85 (1.36)
Spectral Y only	1,356.17 (48.73)	31.35 (2.09)
Spectral X and Y	1,407.57 (60.21)	32.30 (1.12)
Naive	697.40 (60.21)	33.21 (13.53)
Softmax	671.90 (62.91)	32.99 (4.48)
DQN	**602.93** (91.65)	**22.64** (2.24)

4.2 Quantitative Evaluations

4.2.1 Effectiveness Study. In Table 1, we report the evaluation results of the various partition methods. The partitions from different baselines are visualized in Figure 3(a-c). We run each method 10 times, and report the MAE and corresponding standard deviations in the table. We hold constant the number of MCMC iterations at approximately 1,500. Early stopping is allowed if a convergence criterion (discussed below) is met. The final partition of DQN is visualized in Figure 3(d). Overall, we have the following three observations.

Clustering methods overall perform poorly. This is likely due to the fact that the clustering methods do not fully consider the task information in a supervised way. Even when we consider both $\mathbf{X}$ and $\mathbf{y}$, the clustering methods are outperformed by the proposed methods. In general, Agglomerative results in the highest prediction errors for both crime prediction and house price prediction task. This is likely due to the fact that Agglomerative utilizes tract connectivity as a hard constraint. As a result, the generated communities have a large variance in their sizes, as shown in Figure 3(a). K-Means gives worse result than that of Admin in almost all cases. While the number of communities, m is held constant in all cases, the generated partition of K-Means appears to be greater than $m = 77$ in Figure 3(b) because K-Means does not incorporate spatial continuity constraint. Consequently, one community can consist of several disconnected components. Spectral tends to yield the best results in both tasks among these baselines because it accounts for the affinity of tracts and generates communities with similar sizes. However, it is worth mentioning that Spectral method cannot guarantee the spatial continuity of generated communities. From Figure 3(c), we can also see that Spectral results in a similar partition as the original administrative boundary, shown in Figure 1(a). That is also why it achieves similar prediction accuracy as Admin.

The proposed MCMC method outperforms the baselines. Both variants of MCMC methods significantly outperform Admin and Spectral baselines. Such observations validate the effectiveness

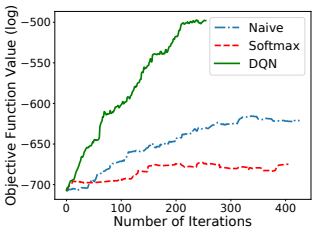


Figure 4: Convergence plots for proposed methods on house price prediction task.

of the MCMC strategy in searching for optimal solutions. On average, `Softmax` tends to outperform `Naive` on both tasks studied. However, it is inconclusive if this is a general result due to the incremental performance difference between the two in certain cases.

`DQN` *method performs the best among all.* It is clear that `DQN` finds a better local optimal solution than that of MCMC methods. On the crime prediction task, the average MAE is 602.93, which represents a 67% improvement over `Admin` baseline. On the house price prediction task, `DQN` consistently gives the best performance. The reason is that `DQN` explores over a subset of actions and picks the best one at each step, compared to the MCMC method, which searches the partition space in a depth first search fashion. Comparing the final partition of `DQN` and `Spectral`, we notice that `Spectral` partition is quite similar to the original administrative boundary in Figure 1. As a consequence, `Spectral` has similar MAE to `Admin`, but is much worse than `DQN`.

4.2.2 Convergence Study. In addition to discussing the proposed methods in the context of model accuracy, we also briefly investigate their performance in terms of convergence. In our method, we track the standard deviation of the $\mathcal{F}$ values from the last 50 iterations. When such standard deviation is less than a pre-defined threshold, we stop. In Figure 4 we visualize the log of quality measure, i.e. $-\mathcal{F}(\mathcal{Z})$, against the number of iterations for three proposed methods on the house price task.

We perform only one run of each method, which could result in slightly different results than what we observe on average. However, Figure 4 is common enough to understand typical convergence behavior. We observe that `DQN` finishes in a less than 300 iterations, while `Naive` and `Softmax` both take more than 400 iterations. Clearly, we observe that `DQN` converge to a better optimal solution with higher training gain. In this specific run, `Naive` is slower to converge than `Softmax`, but also finds a more optimal solution. There is more evidence that these methods may give very similar results, on average.

5 RELATED WORK

Urban Data Heterogeneity. As we collect more types of new urban data, we are able to solve a wide spectrum of urban problems. For example, recent studies use various urban data, such as POI and crowd mobility [19, 20], to estimate air quality [26], noise pollution [27], real estates values [7] and crime [17, 21].

These existing works focus on mining the subtle correlations across different domains of data. We generalize the urban problems above as a learning task, f, which maps some urban features to a target variable of interest. We study how to find a proper region partition as the domain of urban problem, f, so that the learned correlation is consistent and significant.

Traditional Region Partition Methods. There are mainly four types of region partition methods that are widely used in existing urban computing literature. 1) A fixed sized grid [18, 22, 26], 2) existing administrative boundaries [17], 3) clustering of point-wise urban data to get regions [10], and 4) other specifically designed partitions, such as road networks partition [24], cellular tower coverage partition [23], and fan-shaped partitions [28].

It is worthy mentioning that most existing partition methods are purely based on cartographic information, and do not make use of the urban data properties. In this paper, we try to partition the city with an explicit objective, so that the partition methods explicitly take the learning task f into consideration.

Partition Problem in Other Contexts. In the field of community detection problem [5, 6], the partition problem is well studied as well. While most of these works do not have an extra objective for partition, a few of them involves additional property as constraint. For example, the homophilic network decomposition [13] partitions the networks while characterize the degree of homophily of its nodes. The authors assign a dominant label within each group. Hence, the problem is essentially a classification problem, which is fundamentally different from ours.

Another general problem is to partition a set into k groups and fit a model within each group [25]. Such problem definition aims to show that the correlations between features and target vary over the space (e.g., in some area, disadvantage index correlates with crime; while in some other areas, it does not). A simplified version of such problem is segmented regression in 2D space, which could be solved efficiently with dynamic programming [4]. Compared to our problem, such problem formulation has different goal while being easier to solve.

Discrete Optimizations. Our problem is a discrete optimization problem. The objective is easy to derive but difficult to find an optimal solution for. MCMC sampling has been shown to be effective in optimizing discrete structures [14]. To improve the efficiency of MCMC, the learning to optimize technique [9] employs reinforcement learning framework to learn where to draw the next sample. Our solution is mostly inspired by this work.

6 CONCLUSION

In this paper, we proposed a new problem called task-specific region partition. The problem is motivated by the drawbacks of existing administrative boundaries. The task-specific region partition problem is NP-hard, and hence searching for a global optimum is difficult. Two variants of MCMC methods and a reinforcement learning method are proposed to efficiently search for an optimal solution. Our methods are evaluated on two prediction tasks using two evaluation metrics. The learned predictions consistently outperform the administrative boundaries in both tasks.

ACKNOWLEDGMENTS

The work was supported in part by NSF awards #1652525, #1618448, and #1639150. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] 2010. United States Census Bureau. Demographics Survey. http://www.census.gov

[2] Konstantin Andreev and Harald Racke. 2006. Balanced graph partitioning. *Theory of Computing Systems* 39, 6 (2006), 929–939.

[3] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. 2003. An introduction to MCMC for machine learning. *Machine learning* 50, 1-2 (2003), 5–43.

[4] Richard Bellman and Robert Roth. 1969. Curve fitting by segmented straight lines. *J. Amer. Statist. Assoc.* 64, 327 (1969), 1079–1084.

[5] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[6] Santo Fortunato and Marc Barthelemy. 2007. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007), 36–41.

[7] Yanjie Fu, Yong Ge, Yu Zheng, Zijun Yao, Yanchi Liu, Hui Xiong, and Jing Yuan. 2014. Sparse real estate ranking with online user reviews and offline moving behaviors. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 120–129.

[8] Corina Graif, Andrew S Gladfelter, and Stephen A Matthews. 2014. Urban poverty and neighborhood effects on crime: Incorporating spatial and network perspectives. *Sociology compass* 8, 9 (2014), 1140–1155.

[9] Ke Li and Jitendra Malik. 2016. Learning to optimize. *arXiv preprint arXiv:1606.01885* (2016).

[10] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 33.

[11] Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, Massachusetts.

[12] City of Chicago data portal. 2015. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2.

[13] Giulio Rossetti, Luca Pappalardo, Riivo Kikas, Dino Pedreschi, Fosca Giannotti, and Marlon Dumas. 2016. Homophilic network decomposition: a community-centric analysis of online social services. *Social Network Analysis and Mining* 6, 1 (2016), 103.

[14] Malcolm Strens. 2003. Evolutionary MCMC sampling and optimization in discrete spaces. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 736–743.

[15] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[16] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning.. In *AAAI*, Vol. 16. 2094–2100.

[17] Hongjian Wang, Daniel Kifer, Corina Graif, and Zhenhui Li. 2016. Crime rate inference with big data. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 635–644.

[18] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A simple baseline for travel time estimation using large-scale trip data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 61.

[19] Hongjian Wang and Zhenhui Li. 2017. Region Representation Learningvia Mobility Flow. In *In Proceedings of CIKM'17*. CIKM'17, 10 pages.

[20] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A Simple Baseline for Travel Time Estimation Using Large-scale Trip Data. *ACM Trans. Intell. Syst. Technol.* 10, 2, Article 19 (Jan. 2019), 22 pages. https://doi.org/10.1145/3293317

[21] Hongjian Wang, Huaxiu Yao, Daniel Kifer, Corina Graif, and Zhenhui Li. 2017. Non-Stationary Model for Crime Rate Inference Using Modern Urban Data. *IEEE Transactions on Big Data* (2017).

[22] Fei Wu, Hongjian Wang, and Zhenhui Li. 2016. Interpreting traffic dynamics using ubiquitous urban data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 69.

[23] Fengli Xu, Yong Li, Huandong Wang, Pengyu Zhang, and Depeng Jin. 2017. Understanding mobile traffic patterns of large scale cellular towers in urban environment. *IEEE/ACM Transactions on Networking* 25, 2 (2017), 1147–1161.

[24] Jing Yuan, Yu Zheng, and Xing Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 186–194.

[25] Bin Zhang. 2003. Regression clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 451–458.

[26] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. 2013. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1436–1444.

[27] Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric Chang. 2014. Diagnosing New York city's noises with ubiquitous data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 715–725.

[28] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. 2015. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2267–2276.

[29] Zillow.com. 2017. Real Estate Value in Chicago. https://www.zillow.com/.