# Geographic Boosting Tree: Modeling Non-Stationary Spatial Data

Liangdong Deng, Malek Adjouadi and Naphtali Rishe
*School of Computing and Information Sciences*
*Florida International University*

***Abstract*—Non-stationarity is often observed in Geographic datasets. One way to explain non-stationarity is to think of it as a hidden "local knowledge" that varies across space. It is inherently difficult to model such data as models built for one region do not necessarily fit another area as the local knowledge could be different. A solution for this problem is to construct multiple local models at various locations, with each local model accounting for a sub-region within which the data remains relatively stationary. However, this approach is sensitive to the size of data, as the local models are only trained from a subset of observations from a particular region. In this paper, we present a novel approach that addresses this problem by aggregating spatially similar sub-regions into relatively large partitions. Our insight is that although local knowledge shifts over space, it is possible for multiple regions to share the same local knowledge. Data from these regions can be aggregated to train a more accurate model. Experiments show that this method can handle non-stationary and outperforms when the dataset is relatively small.**

***Index Terms*—spatial non-stationarity, R-tree, heuristic search, boosting**

## I. Introduction

Non-stationarity is often observed in Geographic datasets. With these datasets, a predictive model learned at one location does not necessarily remain effective at other locations. For example, when performing topic modeling on documents, certain topics – such as "politics" – are presented differently in different regions, even where the same language is spoken. This phenomenon is called geographic lexical variation [1] and is a major challenge that one must solve before building any accurate topic models. For such data, a single average global model is not good enough to accurately describe all regions. The source of non-stationarity usually comes from a lack of data. In the topic modeling example, if more data is available regarding the cultural difference among geographic regions, it would help with the model accuracy. In other words, non-stationarity has been reduced. The more data we have, the more stationary it would be. In the extreme case, if the dataset includes all aspects of how every single person talks, it essentially becomes a stationary dataset with which it is possible to build a global model with superior accuracy.

But data – especially Human Geography data – are inherently difficult to collect. In our topic modeling example, it is impossible to collect every person's data in a certain area, no matter how small the area is. Even if one finds a way to collect such data, it is also challenging to quantify certain variables. For example, house sale datasets are typically non-stationary. Models fitted from London would not work in Tokyo. Even in the same city, the model wouldn't be the same for all areas. Safety is one of the factors that affect the sale price of a house. But how does one measure the safety of an area? Besides safety, countless factors can affect the sale price of a house, such as traffic, education (school district), tax implications, etc. None of these are easy to collect or measure. These factors are called "local knowledge" [2] in some research. Local knowledge shifts over space and caused non-stationarity. It is unknown to researchers, but its influence is real and observable.

Historically, researchers were more focused on Physical Geography datasets, which are mostly stationary, as the natural processes of Earth – such as mineral deposits and climate – can be accurately described by environmental factors [3]. If all relevant factors are sampled, the model should work no matter where it is. But with the exponential growth of Human Geography datasets due to the widespread adoption of GPS-enabled personal digital devices [4], non-stationarity in Human Geography data received much more interest from researchers.

To solve non-stationarity, a popular approach is to build multiple local models to account for different regions. The regions need to be small enough that data in them are considered to be stationarity or almost stationary. Studies such as [2], [5] and [6] took this approach and obtained great results. However, the biggest problem with this approach is that local models are only fitted to data within a certain region (called the kernel [7]), which is a relatively small set comparing with all the data available. And studies have shown that small sample size has a negative impact on model accuracy [8]. The impact is huge when the sample size drops to a certain threshold. Thus, all the multiple-local-model approaches must seek a compromise between sample size (favors larger kernel size) and the ability to handle non-stationarity (favors smaller kernel size). For smaller datasets, these approaches essentially deteriorate to a one-global-model approach because kernel sizes must be really large for local models to produce meaningful results.

Here we propose the GB-Tree (Geographic Boosting Tree) as a novel solution to non-stationarity. Our insight is that different regions – even if they are disconnected from each other – could share the same models. For example, if we examine the house sale price models for all regions in the U.S., it would be normal to find multiple regions sharing the same or very similar models. If we aggregate data from similar regions (we call this procedure boosting), the trained models would have much higher accuracy than the individual models due to the increased number of observations. This benefit would be maximized if each of the regions has a small dataset, but combined are large enough to generate an accurate model. Experiments show that GB-Tree out-perform other state-of-the-art methods when the dataset is relatively small.

## II. Background

Non-stationarity has been studied for a long time. Early in 1996, Brunsdon, Fotheringham, and Charlton proposed the GWR (Geographically Weighted Regression) algorithm for this problem. Its "main characteristic is that it allows regression coefficients to vary across space, so the values of the parameters can vary between locations" [9]. GWR is designed to allow relationships between features and labels to differ across space, which is substantially equivalent to building many local models for each of the sub-region.

Many later studies inherited this multiple-local-model idea. [5] changed the base modeling algorithm – the method used to build local models – from Ordinary Least Squares (OLS), which was used by GWR, to Random Forests (RF) and observed substantial improvements. Part of the reason why this method was successful is that RF can capture some of the non-stationarity, thus benefits more from larger sub-regions. And [6] further extended the idea by building local models at multiple spatial scales, which make it more flexible and suitable for datasets where non-stationarity is observed at various scales.

However, all these methods suffer from the previously mentioned problem that when the dataset is relatively small, it is impossible to build effective local models from each of the sub-region, which would be even smaller. Some of them do adapt to dataset size to a certain degree. But in this case, they would simply adopt huge kernel sizes that it is no different from building a single global model with the base modeling method. Here, we present a new approach that solves this problem by allowing sub-regions to be aggregated. The aggregated sub-regions (called partitions) will have a boosted accuracy comparing with treating each of the sub-regions individually, thus resulting in improved overall performance.

## III. Study Area

Throughout the paper, we use the Melbourne Housing Market data (obtained from [10]), which contains 8,841 real estate transaction records from the city of Melbourne in Australia. The records span from 2016 to 2018, during which a housing bubble was observed in the area. There are 21 features in the dataset that can be categorized into location, transaction, and house related features. Among all of them, the sale price is the target variable to be modeled and predicted.

This is a classic human geography dataset in which data availability varies depending on the amount of human activity. As illustrated in Figure 1, the downtown area had a lot of real estate transactions during that period, whereas rural regions only have scattered data points. If local models are to be built for each of the rural regions, there is certainly not enough data points to train any meaningful model.

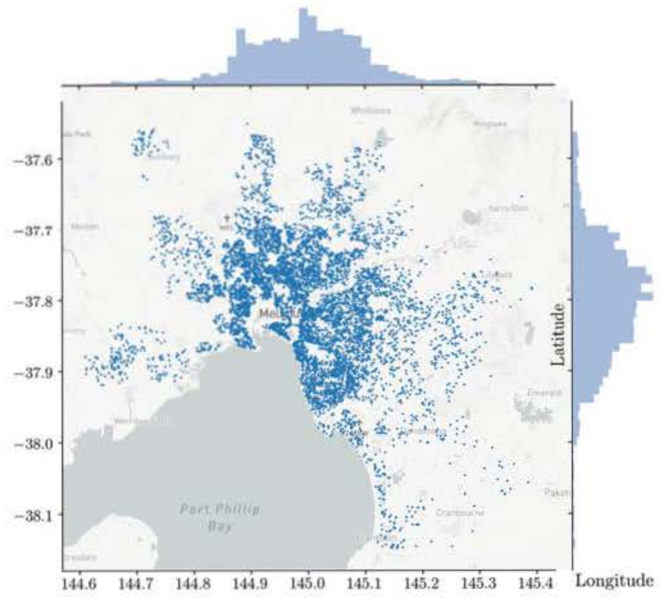

Fig. 1. Melbourne Housing Market Data.

## IV. Geographic Boosting Tree

In this section we detail the design and key steps in GB-Tree's training and predicting process.

### A. Defining Spatial Similarity

To aggregate similar sub-regions, we first need to define what regions are considered to be similar to each other. Here, we introduce the concept of "Spatial Similarity". A region of a geographic dataset is said to be spatially similar to another region if both regions can be explained by the same or similar local models.

Let:

- $R_i$ denotes a sub-region in a geographic dataset.
- $X(R_i)$ denotes observations within $R_i$.
- $M(R_i)$ denotes the model fitted to $X(R_i)$ using any underlying algorithm.
- $E(M(R_i), R_j)$ denotes the prediction error of applying $M(R_i)$ to $R_j$

Then, $R_i$ and $R_j$ are said to be spatially similar if:

$$E(M(R_i), R_i) \approx E(M(R_i), R_j)$$
$$\text{and, } E(M(R_j), R_i) \approx E(M(R_j), R_j)$$

With spatial similarity defined, our goal can be described as finding all the regions that are similar to each other and aggregate them as Partitions (denoted as $P$). For each $P_1, ..., P_n$, we will build a corresponding Boosted Model ($BM$). The final model is then comprised of all $BM$, which will predict unknown observations together.

### B. Testing Spatial similarity

To verify if the idea of spatial similarity works, a test is designed to check if there is any said spatial similarity in

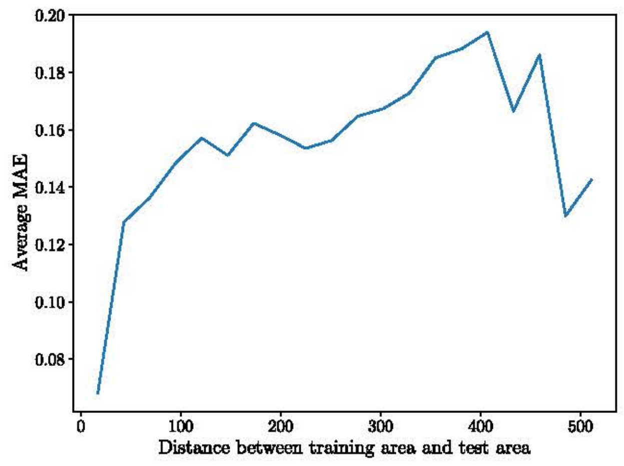

Fig. 2. Spatial similarity test (line graph).

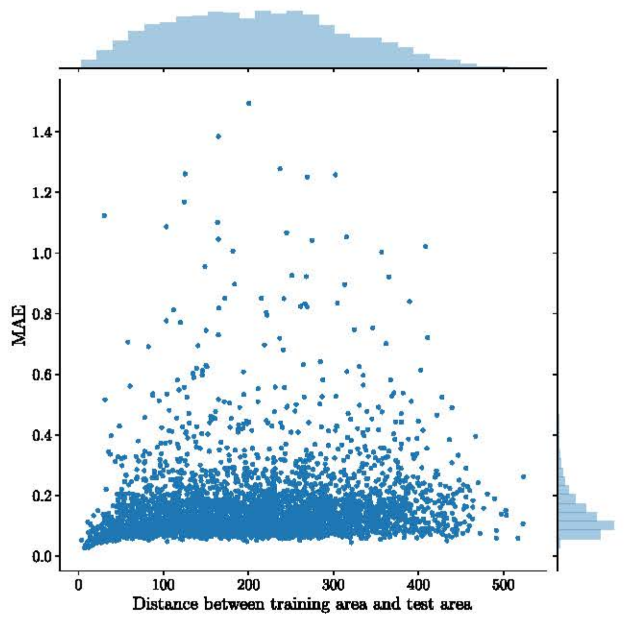

Fig. 3. Spatial similarity test (scatter plot).

the Melbourne House Price dataset. Each time, two random regions $R_i$ and $R_j$ (they have the same size) are selected from the dataset. $M(R_i)$ and $M(R_j)$ are then trained from each of the region, using the Random Forests algorithm. After that, $M(R_i)$ is used to predict data points in $R_j$ and the prediction error (measured by $MAE$, Mean Absolute Error) is added to the final result set. Same procedure is also applied for $M(R_j)$ and $R_i$. This procedure is repeated enough times (5000 times) to reduce sampling bias.

The result is shown in figure 2. The distance is the Euclidean Distance between latitude and longitude of two region's geographical centers multiplied by $10^4$ for the sake of simplicity. From the graph, we can see that $MAE$ is only low when the two regions are very close to each other. In this case, the low $MAE$ is actually caused by $R_i$ and $R_j$ overlapping each other, thus models trained from $R_i$ and $R_j$ are also similar. After a certain distance, the $MAE$ doesn't grow any more and stays unchanged no matter how the distance increases.

When rendering the test result as a scatter plot, as shown in figure 3, we have additional observations. The fluctuation of $MAE$ in figure 2 actually comes from a few number of outliers, which may formed by various reasons. One possible reason is that data points are unevenly distributed in the dataset space. Thus some sub-regions may happen to have too few observations in them to create any accurate models. If these outliers are excluded from the result, we can say that $MAE$ stays almost the same no matter how distance changes.

As a conclusion, two regions, even if they are far away from each other, could be spatially similar enough that they can explained by the same model. By this observation, it is possible for us to aggregate spatially similar regions into partitions.

### C. Creating Regions

As the first step of our algorithm, the data space needs to be divided into regions. Theoretically any spatial dividing algorithm can accomplish the task, such as Grid method, Quadtree [11], and K-D Tree [12]. But to get optimal results, we choose to use a customized R-tree to perform the task of dividing space. R-tree was proposed by Antonin Guttman in the year 1984 [13]. Its main idea is to use a minimum bounding rectangle (MBR) to group objects within an area, and organize them into a hierarchical structure. Unlike many other hierarchical trees, R-tree is constructed from bottom up. Here, we adopt R-tree for several reasons:

- A minimum and maximum number of children can be specified for each of the node. This flexibility makes it possible to create larger MBRs in sparse areas and smaller MBRs in dense areas.
- The choose leaf procedure in the R-tree algorithm is highly customizable. In fact, many R-tree variants were developed over the years,, like R*-tree [14], R+ tree, Hilbert R-tree [15] and so on.

The problem with the classic R-tree is that it could generate very narrow MBRs. We want to avoid such situation because far away data points are less likely to be accounted for by the same model. The famous first law of geography says "everything is related to everything else, but near things are more related than distant things" [16]. In accordance with the idea, we should try our best to assign close data points to the same regions. Thus, squared MBRs are preferable than narrow MBRs.

For this reason, we customize the $ChooseLeaf$ procedure of the classic R-tree as follows:

The $ChooseLeaf$ procedure will traverse the tree to find the best placement strategy for the newly inserted Entry. If the Entry to be inserted can fit any existing node, it will be inserted into that node, obviously. But when inserting the Entry must cause enlargement of an existing node, multiple candidates

```
// Select the leaf node that balances
   between enlargement and squareness
Function ChooseLeaf(e)
    $n = root\_node$
    while n is not a leaf node do
        Initialize best_placement
        foreach c in n.children do
            if n is the direct parent of a leaf node then
                if placing e in c has less impact score
                 than best_placement then
                    $best\_placement = c$
                end
            else
                if place e in c cause less enlargement
                 than best_placement then
                    $best\_placement = c$
                end
            end
        end
        $n = best\_placement$
    end
end
```

will be compared to decide where is the best fit. For the classic R-tree, the simplest and good enough strategy is to place the new Entry where would cause the lease enlargement, which makes total sense as with smaller nodes, the R-tree will query faster. And the original purpose of inventing the classic R-tree is to generate a spatial index which can locate spatial objects quickly.

But in our situation, we don't really care about R-tree's query performance, but use it as a spatial dividing method. Thus we modify the $ChooseLeaf$ procedure to find out the *best_placement* that has the least *impact_score*, whereas *impact_score* is defined as:

$$impact\_score = enlargement * (\text{Ratio of MBR})^2 \quad (1)$$

The *impact_score* is designed to penalize MBRs basing on how narrow they are. And penalties grow exponentially as they become narrower. As a result, this algorithm will generate R-tree with more squared MBRs, as shown in figure 4.

## D. Creating Partitions

After regions are generated, it is now time to find spatially similar regions and group them into partitions. In mathematics, the definition of partition is that "a partition of a set is a grouping of its elements into non-empty subsets, in such a way that every element is included in exactly one subset" [17]. The theoretical number of possible ways to partition a set of size $n$ is called a Bell Number, which is named after mathematician Eric Temple Bell who studied this number in the 1930s. A Bell Number satisfies a recurrence relation [18] :

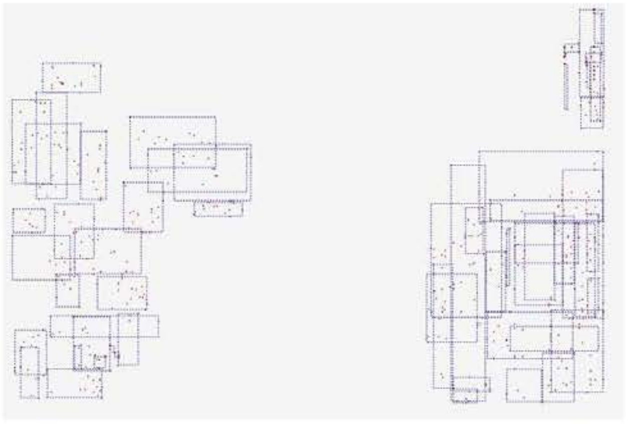

Fig. 4. Regions generated by customized R-tree.

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \quad (2)$$

According to [19], the Bell Number grows exponentially:

$$B_n \sim \frac{1}{\sqrt{n}} \left( \frac{n}{W(n)} \right)^{n+\frac{1}{2}} \exp\left( \frac{n}{W(n)} - n - 1 \right) \quad (3)$$

In the formula above, $W(n)$ is a Lambert W function which has a growth rate of logarithm. With this growth rate of $B_n$, it is impractical to iterate all the possible combinations of partitioning. Thus a heuristic search is needed.

If we train a global model $M_0$ on the entire dataset, it is likely to fit some regions, but doesn't work so well for the rest of regions. To improve $M_0$, we can create a partition $P_0$ from the regions that fit $M_0$, and train another model $M_1$. Although $M_1$ is fitted to $P_0$, there is chance some regions in $P_0'$ (the complement of $P_0$, defined as: $P_0' = \{block \notin P_0\}$) that may be left out, which could also fit $M_1$. Next, we test $M_1$ on $R_{all}$ (all the regions) to see which regions fit $M_1$ best (likewise, let's call them $P_1$). This is essentially a recursion in which both $M_i$ and $P_i$ can keep improving to a certain degree. Ideally, the recursion stops when $P_{i+1}$ is the same with $P_i$. In practice, this may never happen, causing the recursion to run indefinitely. Thus we set the stop condition as when $P_{i+1}$ converges with $P_i$. And the convergence threshold can be a constant or predetermined number. To avoid the situation in which $P_i$ fails to converge, a cap should be put on the maximum number of loops.

So far we only discussed how to generate one partition. To generate all partitions, we could repeat the procedure again and again until all regions are assigned to partitions. But here we have a problem that the aforementioned heuristic search process does not guarantee it will eventually cover all unassigned regions. Some regions may be too much of an outlier that they're never picked no matter how many times we run the heuristic search process.

Hence, we introduce two changes to the algorithm. First, we allow one region assigned to multiple partitions. This is because a region should not be excluded from the next heuristic search simply because it has already been included in a previous generated partition, as there is a chance this region may fit a better partition than the one it's already assigned to. What's more, the heuristic search will choose regions with least $mae$ values with a hard cutoff point. Regions with $MAE$ values slightly greater than the cutoff line may still fit the partition. Thus, the second change we introduce is a concept of similarity score:

$$Score(R, P) = \frac{(MAE(R, M(P)))^{\alpha}}{C(R)} \quad (4)$$

In the equation above, $C(R)$ is the number of times Region $R$ has been included in any partition. As a region being included by more and more partitions, its similarity score to any partition will decrease, causing it much less likely to be included by a lot of partitions. On the other hand, regions which have never been included in an partition would have a higher chance to be eventually selected by a partition. And $\alpha$ is a negative number as the higher $MAE$ it is, the less similar a region is to a certain partition. $\alpha$ is designed to add flexibility to the similarity score. It is generally recommended to have a value between $0$ and $-1$, but the optimal value can be learned by an exhaustive grid search process.

Now the algorithm is almost complete except the number of partitions (denoted as $K$) is not determined yet. Ideally, the optimal $K$ value should be the same as how many different models are needed to fully describe the entire dataset. For some datasets, there might be prior knowledge available to help determine the value of $K$. When such prior knowledge doesn't exist, one should start with $\sqrt{N(R)}$ (square root of the number of regions) and perform a grid search to find out the best $K$ value.

## *E. Predicting*

The final model of the Geographic Boosting Tree is comprised of multiple boost models, with each of the boost model representing a Partition which consists of multiple R-tree regions.

For prediction, one should first use the location of the unknown observation to determine which region(s) it belongs to. Here, one can do a brute force search, or use the traditional R-tree search algorithm to find the correct regions. Remember in R-trees MBRs can overlap each other, so it's possible for one location to belong to multiple regions. If it doesn't fall into any region, then the nearest regions will be used. After the regions are determined, one or many partitions that these regions are associated to can also be determined. And the final predication result should be a majority vote (for classification tasks) or an average (for regression tasks) of predictions from all models.

## *F. Choice of the Underlying Model*

The GB-Tree is actually a framework which can use any regular machine learning algorithm as the underlying method to train boost models. But unless there is special considerations, we recommend using the Random Forests [20] algorithm to create the boost models.

As suggested by the name, Random Forests algorithm train multiple decision trees which are then used to perform predictions. Only part of the data is used to train each of the decision trees thus they're generated differently. What's more, each decision tree node are created from a random subset of features as a further step of randomization. The theoretical basis of RF is that the bagging of multiple randomly generated trees will keep bias the same but decrease the variance of the overall model, thus producing a better result than any of the individual trees.

We choose RF as the underlying model because RF is based on decision trees, which is inherently able to capture some non-stationarity within any spatial data. For many other models, latitude and longitude is tough to be dealt with because they have no way to treat them together as a coordinate. And a lot of information would be lost if latitude and longitude are treated as separate variables. But decision trees are totally different. Any leaf node of a decision tree has a set of ancestor nodes with different criteria. If latitude or longitude appears in one or many of the ancestor nodes, the leaf node can be thought as operating within the region defined by all the ancestor nodes with latitude or longitude as the criteria. From this aspect, a decision tree can actually divide the space into many smaller regions and learn models from them. This procedure is not guaranteed and the decision tree is not necessarily dividing the space in the best way, but this trait is positively enhanced when multiple decision trees are trained to construct a Random Forest.

## *G. Results*

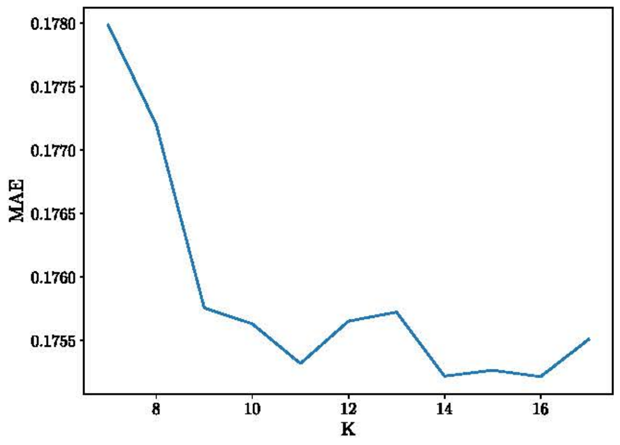

Fig. 5. MAE under different values of K.

We run the GB-Tree algorithm on the Melbourne Housing data. A grid search is performed in order to find how the algorithm performs under different values of $K$. The results are shown in figure 5 as a line graph. When $K$ is small enough, the algorithm essentially becomes the equivalent of the underlying algorithm. And when $K$ is too large, it will try to create a local model for every single model, which is definitely not the desired result. According to the results, $K = 14$ is a good balance and there is no point in increasing $K$ further as the $MAE$ stays pretty much the same when $K$ is greater than 11.

TABLE I
RESULTS FROM OTHER ALGORITHMS ON THE SAME DATASET.

| | MAE |
|---|---|
| Linear regression | 0.308 |
| Neural Network | 0.317 |
| Random Forests | 0.187 |
| MGWR | 0.186 |
| RFsp | 0.191 |

To compare with the others, we also run several other algorithms using the same data. Results are listed in table I. Algorithms not designed to handle non-stationary spatial data, like Linear Regression and Neural Network, produced poor results without surprise. Two state-of-the-art algorithms, RFsp [21] and MGWR [6], which are specifically designed to handle such data, actually performed similarly to the Random Forests algorithm. This is due to the fact that Melbourne Housing dataset is relatively small and the multiple-local-model approach is not as effective in this case. The GB-Tree, however, is still able to capture non-stationarity in such a situation.

## V. CONCLUSION

This paper presents GB-Tree, which solves the challenge of unable to create accurate local models due to the limited number of nearby observations by creating partitions that aggregate spatially similar regions. GB-Tree first divides the data space into multiple regions with a customized R-tree algorithm for which MBRs are designed to be as square as possible. It then groups spatially similar regions into partitions using a heuristic search process. Next, boost models are trained from each of the partitions, using Random Forests as the underlying modeling method. At last, a grid search process is added to find the optimal number of partitions.

By aggregating spatially similar regions into partitions, GB-Tree can train boosted models from larger datasets with better accuracy, in contrast to the traditional multiple-local-model methods that only train one local model from one region. This method provides a novel way to deal with non-stationarity. It can handle non-stationary datasets and outperform other state-of-art algorithms when the dataset is relatively small.

## ACKNOWLEDGMENT

This material is based in part upon work supported by the National Science Foundation under Grant Nos. MRI CNS-2018611, I/UCRC IIP-1338922, MRI CNS-1532061, MRI CNS-1920182.

## REFERENCES

[1] J. Eisenstein, B. O'Connor, N. A. Smith, and E. Xing, "A latent variable model for geographic lexical variation," in *Proceedings of the 2010 conference on empirical methods in natural language processing*, pp. 1277–1287, 2010.

[2] C. Brunsdon, A. S. Fotheringham, and M. E. Charlton, "Geographically weighted regression: A method for exploring spatial nonstationarity," *Geographical Analysis*, vol. 28, no. 4, pp. 281–298, 1996.

[3] D. Massey, "Space-time, 'science' and the relationship between physical geography and human geography," *Transactions of the Institute of British Geographers*, vol. 24, no. 3, pp. 261–276, 1999.

[4] C. Beath, I. Becerra-Fernandez, J. Ross, and J. Short, "Finding value in the information explosion," *MIT Sloan Management Review*, vol. 53, pp. 18–20, 06 2012.

[5] S. Georganos, T. Grippa, A. N. Gadiaga, C. Linard, M. Lennert, S. Vanhuysse, N. Mboga, E. Wolff, and S. Kalogirou, "Geographical random forests: a spatial extension of the random forest algorithm to address spatial heterogeneity in remote sensing and population modelling," *Geocarto International*, pp. 1–16, 2019.

[6] A. S. Fotheringham, W. Yang, and W. Kang, "Multiscale geographically weighted regression (mgwr)," *Annals of the American Association of Geographers*, vol. 107, no. 6, pp. 1247–1265, 2017.

[7] S. Kalogirou, "Destination choice of athenians: An application of geographically weighted versions of standard and zero inflated poisson spatial interaction models," *Geographical Analysis*, vol. 48, no. 2, pp. 191–230, 2016.

[8] J. Morgan, R. Dougherty, A. Hilchie, and B. Carey, "Sample size and modeling accuracy with decision tree based data mining tools," *Acad Inf Manag Sci J*, vol. 6, 01 2003.

[9] J. Mateu, "Comments on: A general science-based framework for dynamical spatio-temporal models," *Test*, vol. 19, pp. 452–455, 11 2010.

[10] T. Pino, "Melbourne housing market data." https://www.kaggle.com/anthonypino/melbourne-housing-market, 2018.

[11] R. Finkel and J. Bentley, "Quad trees: A data structure for retrieval on composite keys.," *Acta Inf.*, vol. 4, pp. 1–9, 03 1974.

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, p. 509–517, Sept. 1975.

[13] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, (New York, NY, USA), p. 47–57, Association for Computing Machinery, 1984.

[14] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, (New York, NY, USA), p. 322–331, Association for Computing Machinery, 1990.

[15] I. Kamel and C. Faloutsos, "Hilbert r-tree: An improved r-tree using fractals," *Proc. Twentieth Int. Conf. Very Large Databases*, 10 1999.

[16] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Economic Geography*, vol. 46, pp. 234–240, 1970.

[17] Wikipedia, "Partition of a set." https://en.wikipedia.org/wiki/Partition_of_a_set, 2020.

[18] N. Asai, I. Kubo, and H.-H. Kuo, "Bell numbers, log-concavity, and log-convexity," *Acta Applicandae Mathematica*, vol. 63, 05 2001.

[19] L. Lovász, *Combinatorial problems and exercises*. Amsterdam: North-Holland, 2. ed. ed., 1993.

[20] L. Breiman, "Random forests," in *Machine Learning*, pp. 5–32, 2001.

[21] T. Hengl, M. Nussbaum, M. N. Wright, G. B. Heuvelink, and B. Gräler, "Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables," *PeerJ*, vol. 6, p. e5518, Aug. 2018.