CSC730: Report for Assignment 3

South Dakota School of Mines and Technology

Carson Price, Kris Jensen

February 4, 2024

1 Introduction

Our task with assignment three is to describe the OptiGrid algorithm, analyze the Optigrid code available on Github [1], in great detail. After analyzing the algorithm and the code, we will modify the demo example to accept a 2D dataset of 30000 points and create visualizations of the data points and cutting planes. If time permits, we can earn extra credit by extending the modifications and visualization to a 3D dataset of 30000 points.

2 Description of OptiGrid

The OptiGrid algorithms works on a dataset by first determining if the dataset can be contracted, dimensionally reduced, by projecting the data onto a d-1 space. This means that if the dataset can be divided into two seperate spaces by a plane, then the dataset can be seperated into at least two classes. Simply drawing a plane through the data is insufficient to determine the best plane to divide the data.

The algorithm will determine if a plane exists to divide the data by calculating the score of the plane. The score is determined by applying a kernel density estimation function to the data in the current dataset. Depending on the kernel density esimation function and bandwidth, the score will contains some number of peaks. If there are zero or one peaks, then the splitting for this dataset is complete. If the score is above a certain threshold, then the plane is considered a good plane to divide the data. The data is then labeled as being left or right, or above or below the plane.

We will refer to left or top as A and right or bottom as B. The total dataset, D, is the union of A and B. Upon succesfull splitting of A and B, then the algorithm will recursively apply the same process to A and B. The algorithm will continue to split the dataset until the dataset is no longer able to be split.

3 Analysis of OptiGrid Code

Let's begin by setting the stage for the OptiGrid code, by analyzing the pseudocode set out by Hinneburg and Keim [2]. Then the structure of the code will be outlined, followed by a detailed analysis of each function and its purpose.

OptiGrid(dataset D, q, min_cut_score)

```
1.  Determine a set of contracting projections P = {$P_0$, . . ., $P_k$}
2.  Calculate all projections of the dataset $D \rightarrow P_0(D)$, . . ., $P_k(D)$
3.  Initialize a list of cutting planes $BEST\_CUTS \leftarrow \emptyset, CUT \leftarrow \emptyset$
4.  FOR i=0 TO k Do
      a. CUT ← Determine best_local_cuts($P_i(D)$)
      b. CUT_SCORE ← score_best_local_cuts($P_i(D)$)
      c. Insert all cutting planes with a score ≥ min_cut_score into $BEST\_CUTS$
    END FOR
5.  IF $BEST\_CUT = \emptyset$ THEN RETURN $D$ as a cluster
6.  Determine the $q$ cutting planes with highest score from $BEST\_CUTS$ and delete the rest
7.  Construct a Multidimensional Grid $G$ defined by the cutting planes in $BEST\_CUTS$ and insert all data points $x \in D$ into $G$
8.  Determine clusters, i.e. determine the highly populated grid cells in G and add them to the set of cluster C
9.  REFINE(C)
10. FOREACH Cluster $C_i \in C$ DO
    OptiGrid($C_i$, q, min_cut_score)
```

3.1 Functions of class OptiGrid

1. *__init__*
   The *__init__* functions acts as the class constructor. It initializes the class variables and sets the default values for the parameters. The parameters include dataset dimension (**d**), number of cuts per iteration (**q**), the max cut score density of a plane (**max_cut_score**), noise level for dataset(**noise_level**), several parameters related to the kernel density estimation including bandwidth (**kde_bandwidth**), grid ticks (**kde_grid_ticks**), sample size

(**kde_num_samples**), tolerance (**kde_atol**) and (**kde_rtol**), and finally an argument for turning on or off output (**verbose**) . This function sets the initial conditions of the OptiGrid algorithm.

2. fit
3. *_iteration*
4. *_fill_grid*
5. *_create_cuts_kde*
6. *_find_best_cuts*
7. *_find_peaks_distribution*
8. *_estimate_distribution*
9. *score_samples*
10. *_score_sample*

3.2 Functions of class GridLevel

1. *__init__*
2. *add_subgrid*
3. $get_sublevel$

4 Results

5 Discussion

6 Conclusion

7 References

[1] mihailescumihai/optigrid. (2019). GitHub. https://github.com/mihailescum/Optigrid
[2] Hinneburg, A., & Keim, D. A. (1999). Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In Proceedings of the 25th International Conference on Very Large Data Bases (pp. 506-517). Morgan Kaufmann Publishers Inc.