Application of k-Dimensional Tree with k-Nearest Neighbor algorithm in Classifying Objects

Annisa Ayu Pramesti - 13518085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518085@std.stei.itb.ac.id

***Abstract*—Objects can be differentiated and classified because they are presented in spatial areas based on their characteristic. Clustering and classification are often used in data mining and machine learning so many algorithms are developed very rapidly. This study applies k-Nearest Neighbor as the searching algorithm and implemented in k-Dimensional Tree structure. The dataset consists of 18 Arabic characters with each character represented by 32 images. The aim of this study is to classify the Arabic characters and then determine the class of a set of given queries.**

***Keywords*—Clustering, k-Dimensional Tree, k-Nearest Neighbor**

I. Introduction

Classification is the process of learning a model that explain different classes of given data. It is a two-step process, consists of a learning step and a classification step. In machine learning and statistics, classification is the problem of identifying the region/class of a new datum. Examples are assigning a given email to the "spam" or "non-spam" class and assigning a diagnosis to a given patient based on observed characteristics of the patient. Classification is an example of pattern recognition and considered an instance of supervised learning. The corresponding unsupervised procedure is known as clustering and involves grouping data into categories based on some measure of inherent similarity or distance. In learning step, a classification model is constructed, and classification step the constructed model is used to prefigure the class labels for given data.

There are many algorithms that can be used for classifying new objects into predetermined classes such as Random Forest, Stochastic Gradient Descent, K-Nearest Neighbor etc. The algorithm that is going to be implemented in this paper is K-Nearest Neighbor using K-Dimensional Tree. A K-D Tree is multidimensional generalization of a binary search tree. It can be used efficiently for range queries and nearest neighbor searches relatively if the provided data is not too big or has too many dimensions. In document analysis problems, the dimension is typically two, so that k-d trees can be a good utility for layout analysis problems.

This study aims to apply the implementation of the k-d trees search algorithm to find K-NN of a given query of an Arabic character so that the class of the character can be known. For every query, the program will determine its class using k-d trees.

II. Clustering, Tree, Graph, K-Dimensional Tree, K-Nearest Neighbor

A. Clustering

1. Partitioning algorithm

The separator algorithm builds attributes of many objects into a set of clusters, it builds clusters in one step that contradicts several steps, and only one set of clusters is built, although several different sets of clusters can be built, although several different sets of clusters can be built internally in various algorithms. The edges of the original chart that cross between groups will produce edges in a partitioned graph. If the number of edges produced is small compared to the original graph, the partitioned graph may be more suitable for analysis and problem solving than the original. Because only one set of clusters is output, the user must enter the desired number of clusters. Each cluster is represented by one of the objects of the cluster located near its center (k-medoid algorithm) or by the center of the cluster (k-means algorithm).

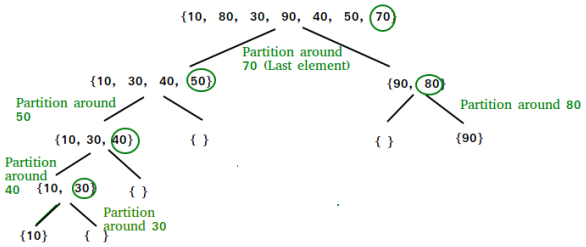

Figure 1. The scheme of partitioning algorithm

2. Hierarchical algorithm

Hierarchical clustering organizes clusters into trees or hierarchical structures that are represented using tree data structures, where the root of this tree is a single cluster that contains all objects. Leaves are a group of single objects. Hierarchy algorithm is run iteratively to make a separation or merge until the criteria stop being fulfilled or all objects have been grouped. Hierarchical algorithms are categorized into agglomerative and divisive algorithms, where the dendrogram can be made from the leaves to the root (Agglomerative approach), or from the root down to the leaves (divisive approach).

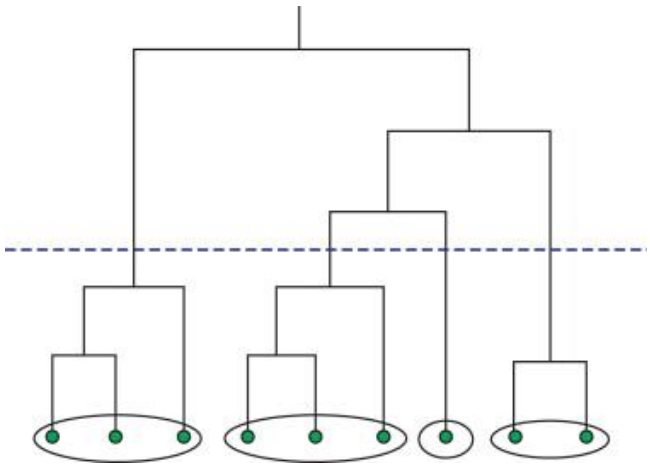

Figure 2. Hierargical clustering

3. Density based algorithm

This algorithm is based on density, as the point is connected to density. The algorithm has many features, such as its ability to find arbitrary form groups, as well as, to distinguish noise, and they require the density parameter as a termination condition.

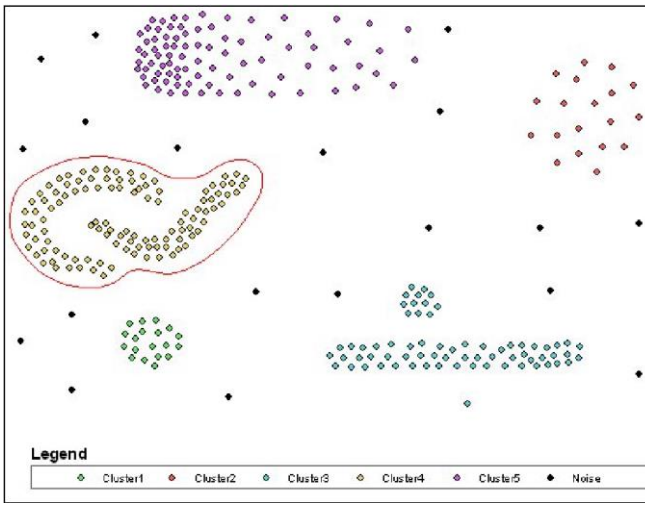

Figure 3. Illustration of density based clustering

4. Grid based algorithm

The grid-based algorithm depends on the size of the grid rather than the data object, uses a single uniform grid mesh to partition the entire problem domain into cells, and data objects in cells are represented by cells using a set of statistical attributes of the object. Grid-based algorithm makes groupings on grid cells, not on the database itself. In conclusion this algorithm faster than the others.

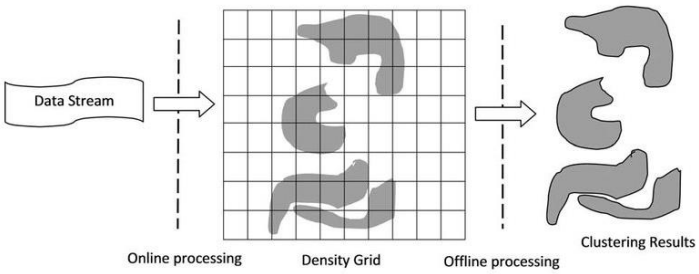

Figure 4. Example of grid base clustering

B. Tree

A tree is a simple, undirected, connected, acyclic graph (or, equivalently, a connected forest). A common tree or tree is defined as a limited set of non-empty elements called nodes or nodes that have the property that each node can have a minimum degree of 1 and a maximum degree of n. It can be partitioned into separate n + 1 subsets so that the first subset contains the root of the tree and the rest of the subset n includes elements from the subtree n.

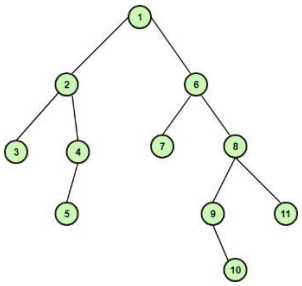

Figure 5. Structure of tree

Let G = (V, E) be a simple non-directed graph, the number of vertices is n, and is a tree. Each pair of vertices in G is connected by a single path and has m = n - 1 side. G does not contain a circuit and the addition of one side to the graph will make only one circuit. G is connected and all sides are bridges.

Each connected graph has at least one spanning tree. Spanning tree from a connected graph is a spanning graph in the form of a tree. Spanning trees are obtained by breaking the circuit in the graph. The unconnected graph with k components has k spanning forests called spanning forests.

C. Graph

A graph G consists of 2 finite sets, namely a set of non-empty points (symbol V (G)) and a set of lines (symbol E (G)).Each line corresponds to one or two points. These points are called End Points. Lines that only correspond to one endpoint are called loops. Two different lines connecting the same point are called Parallel Lines.
Two points are said to be adjacent (adjacent) if there is a line connecting the two. Points that do not have a line associated with them are called Isolating Points
A graph that has no dots (so no lines) is called a Blank Graph. If all the lines are directed then the graph is called the Directed Graph, or often abbreviated as Digraph. If all the lines are not directed, then the graph is called an Undirected Graph. In this material, if only graphs are mentioned, then what is meant is undirected graphs.
Sometimes a graph is represented by an image. Draw a graph G consists of a set of point-views V (G), a set of lines E (G) connecting these points (along with the direction of the line on directed graphs), and labels on their lines (if any). The length of the line, the curvature of the lines, and the location of the points have no effect in a graph.

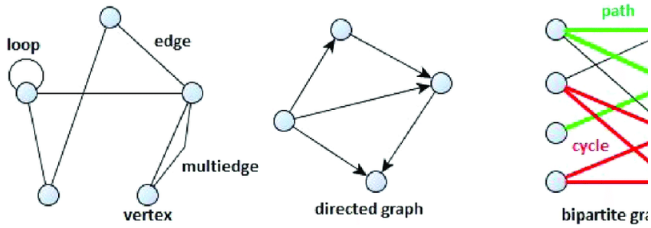

Figure 6. Example of graph.

Graph G = (V, E), which in this case:

1. V = non-empty set of vertices
   = {v1, v2, ..., vn}
2. E = edge set connecting a pair of vertices
   = {e1, e2, ..., en}

D. K-Dimensional Tree

K Dimensional tree (or k-d tree) is one of various tree data structure. It is used to represent points in the k dimensional space. This data structure is famous of its various applications such as the closest point (in k-dimensional space), efficient spatial data storage, search range etc.

The implementation is very different from other tree structure data such as quadtree or octree. Each internal node in this structure divides space into 2 half. The left child of the node represents the left half while the right child represents the right half. Space is divided into 2 parts regardless of the number of dimensions. To be more accurate, each internal node represents a hyperplane which cuts space into 2 parts. For 2-dimensional space, it is a line and for 3-dimensional space, it is a plane.

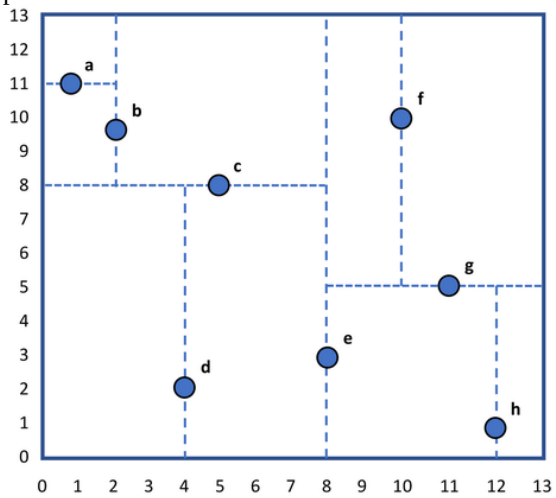

Figure 6. Illustration of K-D Tree

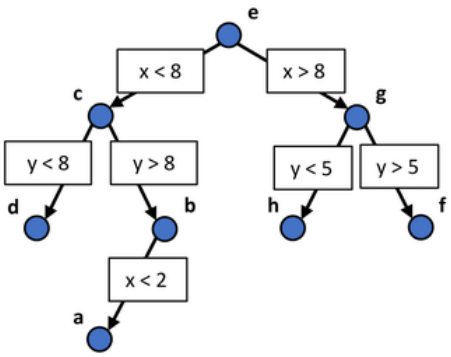

Figure 7. The tree structure based on figure 2

The purpose of this structure data is always to hierarchically decompose space into a relatively small number of cells such that no cell contains too many input objects. This provides a fast way to access any input object by position. However, this may not work well in a very big dimension but relatively good in low dimensions.

E. K-Nearest Neighbor

An object is classified by the most similar from its neighbors, assigned to the most common class among K's closest neighbors measured by the distance function. If K = 1, then a simple case is assigned to the nearest neighbor class. With the given data, K-NN can classify new, unlabelled data by analysis of the k number of the nearest data points. Thus, the variable k is considered to be a parameter that will be established by the machine learning engineer.

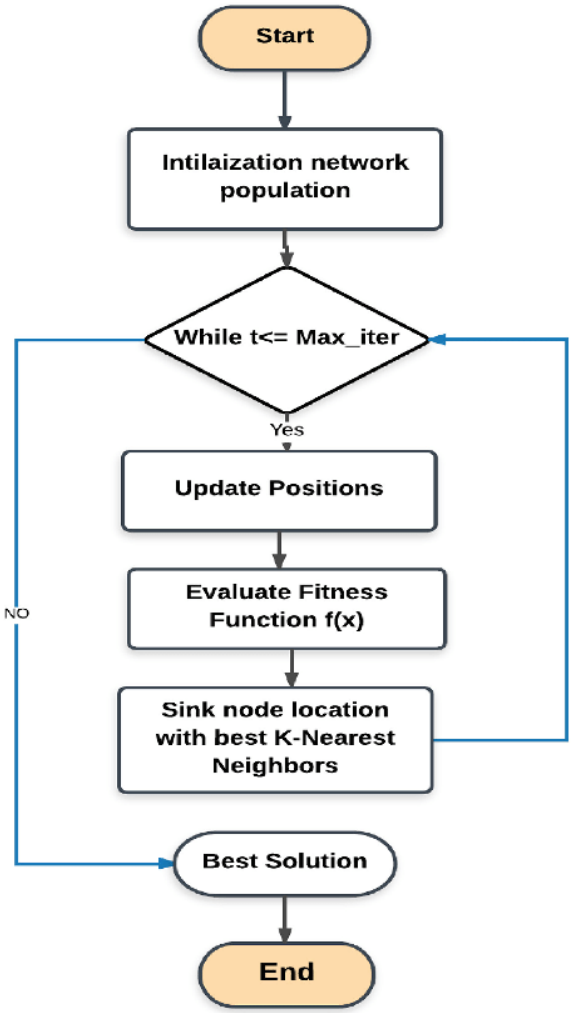

Figure 8. Flowchart of k-Nearest Neighbor

The algorithm goes pretty simple. Let m be the number of training data samples and p be an unknown point. The training samples need to be stored in an array of data points arr []. This means each element of this array represents a tuple (x, y). After that we can iterate i from 0 to n and calculate Euclidean distance d (arr [i], p). Thus, we make set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point. The function then return the majority label among S.

Choosing the optimal value for K is best done by first inspecting the data so that it can satisfy the need. In general, a large K value is more precise as it reduces the overall noise but usually needed more running time. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. As the K increasing, the smoother and the more defined boundaries among different classes.

III. Arabic Character Recognition

A. Arabic Character Dataset

In this paper, the dataset consist of images of Arabic characters. Every character has 32 train images that are all different to accommodate various queries. The data are stored in a folder and each image has a unique name that contain the label of the character that is being represented.

Figure 9. Example train image of Arabic character

The algorithm used for extraction is OpenCV Kaze and the results of the extraction are stored in a list along with the categories.

B. Building K-Dimensional Tree

The main function that is used to build K-Dimensional tree is insert. If we have a k-d tree consists of root node and a current axis, the left subtree will contain all points whose coordinates in the axis are smaller than that of root node. Similarly, the right subtree will contain all points whose coordinates in that axis are greater-equal to that of root node.

```
function kdtree (list of points pointList, int depth)
{
    // Select axis based on depth so that axis cycles through all valid values
    var int axis := depth mod k;

    // Sort point list and choose median as pivot element
    select median by axis from pointList;

    // Create node and construct subtree
    node.location := median;
    node.leftChild := kdtree(points in pointList before median, depth+1);
    node.rightChild := kdtree(points in pointList after median, depth+1);
    return node;
}
```

Figure 10. Example algorithm uses a median-finding sort to construct a balanced k-d tree *(Source: Wikipedia)*

Each level of K-d tree partitions the space into two partitions, the partitioning is done along one dimension of the node at the top level of the tree, along another dimension in nodes at the next level, and so on, iterating through the dimensions. The partitioning proceeds in such a way that, at each node, approximately one half of the points stored in the subtree fall on one side, and one half fall on the other. Partitioning stops when a node has less than a given maximum number of points.

C. Finding Nearest Neighbor

```
knnQuery(queryVector) {
  while (not nodeQueues.allEmpty) {
    node = pollNodeQueues()
    if (not node.isLeafNode) {
      if (isCandidate(node.leftChild, queryVector)) {
        addToNodeQueue(node.leftChild)
      }
      if (isCandidate(node.rightChild, queryVector)) {
        addToNodeQueue(node.rightChild)
      }
    } else {
      addToResultList(node)
    }
  }
  return resultList
}

node pollNodeQueues() {
  int rand = random(0, numberOfNodeQueues - 1)
  if (not nodeQueues.get(rand).isEmpty) {
    return nodeQueues.get(rand).poll()
  } else {
    for (int i from 0 to numberOfNodeQueues - 1) {
      if (not nodeQueues.get(i).isEmpty) {
        return nodeQueues.get(i).poll()
      }
    }
  }
  return lowPriorityQueue.poll()
}
```

Figure 11. Pseudo code of modified kNN search algorithm[1]

The kNN search algorithm presented in traverses the k-d tree recursively. For a given node, the search function executes the following steps: (1) If its point is closer to the query point than any other within a global list if the k best result candidates, update the list with that point. (2) If the node is not a leaf node, the function is recursively called with the child node closer to the query point. (3) The minimum possible distance between the 576 query point and the other, more distant child node is calculated. If this distance is larger than the maximum distance for a candidate (calculated from the result list), no point in that node or its descendants can be a result candidate. Otherwise, the search function is called with that node.

D. Testing

To test the data, the dataset is splitted into 2 parts, train data and test data. Test data takes 25% of the dataset and the train data takes 75%. From the tests conducted, the 87.3% query image results were obtained in accordance with the desired category. However, the execution time is relatively slow so the algorithm need to be improved in order to lessen the running time.

IV. Conclusion

A program has been created using the KNN algorithm with the k-d tree data structure. This program is used to group and test datasets in the form of Arabic letters and produce an accuracy of 87.3%. The overall algorithm is pretty fast but as the dimensions go bigger the execution time goes slower.

V. Future Research

There are many researches that can be developed based on this proposition such as creating descriptor of each key point obtained or simply developing the algorithm so it can be much faster and efficient. This algorithm can also be used in various researches in many fields such as medical field, natural language processing, etc, as long as the data can be represented by multidimensional points.

VII. Acknowledgment

The author would like to express her gratitude to God Almighty for his guidance, to her parents for their eternal support, love and education, and to the Mr. Rinaldi Munir for his teachings of Discrete Mathematics in Class K1 to complete this paper.

References

[1] Hering, Tim, “Parallel Execution of kNN-Queries on in-memory K-D Trees”. Magdeburg: University Otto-von-Guericke.
[2] K.H. Rosen, Discrete Mathematics and its Applications, 7th ed. New York: McGraw-Hill, 2012, ch 10-11.
[3] Otair, Mohammed. “Approximate K-Nearest Neighbour based Spatial Clustering using K-D Tree”. International Journal of Database Management Systems (IJDMS) Vol.5, No.1, February 2013
[4] https://github.com/ppplinday/Picture-Classification-PCA-KNN. accessed on 5 December 2019.
[5] Fu, Cong and Deng Cai. “EFANNA : An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph”. arXiv:1609.07228v3 [cs.CV] 3 Dec 2016
[6] Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching". Communications of the ACM. 18 (9): 509–517. doi:10.1145/361002.361007.
[7] Jacob E. Goodman, Joseph O'Rourke and Piotr Indyk (Ed.) (2004). "Chapter 39 : Nearest neighbours in high-dimensional spaces". Handbook of Discrete and Computational Geometry (2nd ed.). CRC Press.
[8] Rosenberg, J. B. (1985). "Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 4: 53–67. doi:10.1109/TCAD.1985.1270098.
[9] Houthuys, P. (1987). "Box Sort, a multidimensional binary sorting method for rectangular boxes, used for quick range searching". The Visual Computer. 3 (4): 236–249. doi:10.1007/BF01952830.
[10] Feldmann, Andreas Emil (2012). "Fast Balanced Partitioning is Hard, Even on Grids and Trees". Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science. arXiv:1111.6745. Bibcode:2011arXiv1111.6745F.
[11] Garey, Michael R.; Johnson, David S. (1979). Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman & Co. ISBN 978-0-7167-1044-8.
[12] Hendrickson, B.; Leland, R. (1995). A multilevel algorithm for partitioning graphs. Proceedings of the 1995 ACM/IEEE conference on Supercomputing. ACM. p. 28.
[13] https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/. accessed on 5 December 2019.
[14] https://www.pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/. accessed on 5 December 2019.
[15] http://code.activestate.com/recipes/577497-kd-tree-for-nearest-neighbor-search-in-a-k-dimensi/. accessed on 5 December 2019.
[16] http://pointclouds.org/documentation/tutorials/kdtree_search.php. accessed on 5 December 2019.
[17] https://gist.github.com/tompaton/863301. accessed on 5 December 2019.
[18] https://www.kaggle.com/sabermalek/plf50. accessed on 5 December 2019.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 6 Desember 2019

Annisa Ayu Pramesti - 13518085