2010

An Attempt to Find Neighbors

Yong Shi
Kennesaw State University, yshi5@kennesaw.edu

Ryan Rosenblum
Kennesaw State University

Recommended Citation

Yong Shi, Ryan Rosenblum, "An Attempt to Find Neighbors," cyberc, pp.318-320, 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2010

2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery

An Attempt to Find Neighbors

Yong Shi and Ryan Rosenblum
Department of Computer Science and Information Systems
Kennesaw State University
1000 Chastain Road
Kennesaw, GA 30144

***Abstract*—In this paper, we present our continuous research on similarity search problems. Previously we proposed *PanKNN*[18] which is a novel technique that explores the meaning of K nearest neighbors from a new perspective, redefines the distances between data points and a given query point *Q*, and efficiently and effectively selects data points which are closest to *Q*. It can be applied in various data mining fields. In this paper, we present our approach to solving the similarity search problem in the presence of obstacles. We apply the concept of obstacle points and process the similarity search problems in a different way. This approach can assist to improve the performance of existing data analysis approaches.**

I. Introduction

Huge amount of data have been generated in many disciplines nowadays. The similarity search problem has been studied in the last decade, and many algorithms haves been proposed to solve the K nearest neighbor search[13], [17], [2], [12], [9]. We previously proposed *PanKNN*[18] which is a novel technique that explores the meaning of K nearest neighbors from a new perspective, redefines the distances between data points and a given query point *Q*, and efficiently and effectively selects data points which are closest to *Q*. In this paper, we first give a brief introduction about our previous work on PanKNN and discuss the Fuzzy concept; then, we propose to use the Fuzzy concept to design OPanKNN algorithm that targets solving the nearest neighbors problems in the presence of obstacles.

II. Related work

The similarity between two data points used to be based on a similarity function such as Euclidean distance which aggregates the difference between each dimension of the two data points in traditional nearest neighbor problems. In those applications, the nearest neighbor problems are solved based on the distance between the data point and the query point over a fixed set of dimensions (features). However, such approaches only focus on full similarities, i.e., the similarity in full data space of the data set. Also early methods [1], [6], [21] suffer from the ”cure of dimensionality”. In a high dimensional space the data are usually sparse, and widely used distance metric such as Euclidean distance may not work well as dimensionality goes higher. Recent research [7] shows that in high dimensions nearest neighbor queries become unstable: the difference of the distances of farthest and nearest points to some query point does not increase as fast as the minimum of the two, thus the distance between two data points in high dimensionality is less meaningful. Some approaches [14], [4], [3] are proposed targeting partial similarities. However, they have limitations such as the requirement of the fixed subset of dimensions, or fixed number of dimensions as the input parameter(s) for the algorithms.

There are quite a few approached designed to detect clusters in the presence of obstacles and facilitators. For example, COD_CLARANS [5] is a modified version of the CLARANS [15] partitioning algorithm which performs clustering processes in the presence of obstacles. AUTOCLUST+ [11] is a version of AUTOCLUST [10] enhanced to handle obstacles, which does not require parameters. DBRS+ [19] is derived from DBRS [20], and it handles both obstacles and facilitators. However, none of these algorithms considers detecting outliers simultaneously with clustering process. In many cases, outliers are as important as clusters, such as credit card fraud detection, discovery of criminal activities, discovery of computer intrusion, and etc. These approaches do not consider the presence of obstacles as well.

III. Fuzzy concept

Various data sets in the real world are not naturally well organized and fuzzy concept can be applied to further improve the data analysis approaches. The concept of fuzzy sets was first introduced by Zadeh [23] to represent vagueness. The use of fuzzy set theory is becoming popular because it produces not only crisp decision when necessary but also corresponding degree of membership. Usually, membership functions are defined based on a distance function, such that membership degrees express proximities of entities to cluster centers. In conventional clustering, sample is either assigned to or not assigned to a group. Assigning each data point to exactly one cluster often causes problems, because in real world problems a crisp separation of clusters is rarely possible due to overlapping of classes. Also there are exceptions which cannot be suitably assigned to any cluster. Fuzzy sets extend to clustering in that objects of the data set may be fractionally assigned to multiple clusters, that is, each point of data set belongs to groups by a membership function. This allows for ambiguity in the data and yields detailed information about the structure of the data, and the algorithms adapt to noisy data and classes that are not well separated. Most fuzzy cluster analysis methods optimize a subjective function that evaluates a given fuzzy assignment of data to clusters.

978-0-7695-4235-5/10 $26.00 © 2010 IEEE
DOI 10.1109/CyberC.2010.64

One of the classic fuzzy clustering approach is the Fuzzy C-means Method designed by Bezdek, J. C [8]. In brief, for a data set X with size of n and cluster number of c, it extends the classical within groups sum of squared error objective function to a fuzzy version by minimizing the objective function with weighting exponent m, $1 \leq m < \infty$. On the other hand, the fuzzy C-Means (FCM) uses an iterative optimization of the objective function, based on the weighted similarity measure between x_k and the cluster center v_i.

IV. Solving similarity problems

We will briefly introduce our previous work on PanKNN[18] in this section. PanKNN is a novel approach to nearest neighbor problems in which we also analyze the nearest neighbor problems for a new perspective. We define the new meaning for the K nearest neighbors problem, and design algorithms accordingly. The similarity between a data point and a query point is not based on the difference aggregation on all the dimensions. We propose self-adaptive strategies to dynamically select dimensions based on the different situation of the comparison.

For a given data point X_i, and a given query point Q, we call the distance between X_i and Q as Pan-distance $PD(X_i, Q)$. $PD(X_i, Q)$ does not calculate the aggregated differences between X_i and Q on all dimensions. Instead, it only take into account those dimensions on which X_i is close enough to Q, and sum them up. This strategy not only avoids the negative impacts from those dimensions on which X_i is far to Q, but also eliminate the curse of dimensionality caused by similarity functions such as Euclidean distance which calculates the square root of the sum of squares of distances on each dimensions. On more dimensions X_i is close (within the sets of K nearest neighbor) to Q, the smaller Pan-distance X_i has to Q. If we have two data points X_i and X_j, we judge which data point is closer to Q based on how many dimensions on which they are close enough (within dimension-wise K nearest neighbors) to Q, as well as their average distances to Q on such dimensions.

Given a data set DS, we first calculate the difference δ_{il} of each data point X_i to the query point Q on each dimension D_l. Then we sort the ids on each dimension D_l based on δ_{il}, and select the first K ids on each dimension D_l and put them into KS_l. We put all the ids in all KS_l to the set GS, and calculate the PD(X_i, Q) for each data point if its id is in GS. Finally, we sort the ids based on the Pan-distance and select the first K ids in the sorted list as the ids of K nearest neighbors of Q. We do not need to calculate the difference using different number of dimensions. The *number* of dimensions and the *subset* of dimensions associated with data point X_i are both dynamically decided depending on the values of X_i and their rankings on different dimensions.

V. Searching nearest neighbors

The PanKNN algorithm solves the similarity search problems in a new perspective efficiently and effectively. However, it does not consider the cases where there are obstacles in the date sets from which we try to find the nearest neighbors for a given query point Q (an example is shown in figure **??**). In this section we propose to design an algorithm in the presence of obstacles, which will be referred to as OPanKNN.

Let n denote the total number of data points and d be the dimensionality of the data space. Let D_l be the lth dimension, where l = 1, 2, ..., d. Let the input d-dimensional data set be **X**

$$\mathbf{X} = \{X_1, X_2, ..., X_n\} \tag{1}$$

which is normalized to be within the hypercube $[0, 1]^d \subset R^d$. Each data point X_i is a d-dimensional vector:

$$X_i = [x_{i1}, x_{i2}, ..., x_{id}]. \tag{2}$$

Data point X_i has the id number i. Let Q be the query point: $Q = [q_1, q_2, ..., q_d]$. Let $\Delta_i = [\delta_{i1}, \delta_{i2}, ..., \delta_{id}]$ as the array of differences between the data point X_i and the query point Q on each dimension. There are obstacles existing in the data set as well. Obstacles can be represented in various ways. One simple and efficient way is to represent them as multi-dimensional points like the data points in the data set and the query point Q. Let m be the total number of obstacle points, and we can represent the set of obstacle points as:

$$\mathbf{C} = \{C_1, C_2, ..., C_m\} \tag{3}$$

which is also normalized to be within the hypercube $[0, 1]^d \subset R^d$. Each obstacle point C_h is a d-dimensional vector:

$$C_h = [c_{h1}, c_{h2}, ..., c_{hd}]. \tag{4}$$

Each value c_{hl} where h=1,2,...,m and l=1,2,...,d represents a obstacle point on dimension D_l where values on the two different sides of c_{hl} are obstructed to be in the same segment (zone).

Since the full data space is normalized, the value range of the data points on each dimension D_l, where l = 1, 2, ..., d should be within the interval [0,1], as well as the value range of the obstacle points. On dimension D_l, the values of all the obstacle points are:

$$c_{1l}, c_{2l}, ..., c_{ml} \tag{5}$$

We sort them in ascending order

$$c_{1l}', c_{2l}', ..., c_{ml}' \tag{6}$$

where $c_{1l}' >= 0$ and $c_{ml}' <= 1$.

For the purpose of consistency, let c_{0l}' represent 0, and let $c_{m+1,l}'$ represent 1. Thus the value range on dimension D_l can be divided into m+1 zones (segments):

$$[c_{0l}', c_{1l}'), [c_{1l}', c_{2l}'), ..., [c_{ml}', c_{m+1,l}'] \tag{7}$$

We use Z_{l0}, Z_{l1}, ..., $Z_{l,m+1}$ to represent them respectively. For a given query point, $Q = [q_1, q_2, ..., q_d]$, suppose its value q_l on $D_l \in [c_{kl}', c_{k+1l}')$, or Z_{lk} where k=0,1,...,m . For each data point X_i in **X**, on each dimension D_l, where l=1,2,...,d, we not only check if its value x_{il} on D_l is close to q_l which is the value of Q on D_l, but also check if x_{il} is in the same segment of q_l on D_l. If x_{il} is not in the same segment of q_l , even if x_{il} is one of the K closest value to q_l, we still can not say it is very close Q on D_l. On the other hand, it is also inappropriate to completely discard x_{il} in the following calculation. Here we adopt the fuzzy concept to determine the weight x_{il} should have when we calculate the distance between X_i and Q.

Given a data set DS of n data points $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ with *d* dimensions D_1, D_2, ..., D_d, a query point Q, and a set of obstacle points $\mathbf{C} = \{C_1, C_2, ..., C_m\}$ in the same data space, we first sort the data points on each dimension D_l, l=1, 2, ..., d, based on δ_{il} which is the difference between data point X_i and Q on dimension D_l. On each dimension D_l, l=1, 2, ..., d, let KS_l be the set which contains the ids of the first K data points in the sorted list. We call these first K data points as *dimension-wise K nearest neighbor* to Q on D_l.

Those K data points which ids are in KS_l, however, might not be in the same segment (zone) with q_l. This is due to the possibility that Z_{lk} which q_l belongs to contains less than K data points.

For each data point X_i, i=1, 2, ... n, let F_i be an array containing values in the range of [0,1]: $F_i = [f_{i1}, f_{i2}, ..., f_{id}]$ in which f_{ij} represents the degree of the possibility that dimension j should be considered when we calculate the distance between X_i and Q.

Given two d-dimensional points $X_i = [x_{i1}, x_{i2}, ..., x_{id}]$ and $Q = [q_1, q_2, ..., q_d]$, with the existence of obstacle points $\mathbf{C} = \{C_1, C_2, ..., C_m\}$, and D_l as the dimension l, l=1, 2, ..., d, the *Pan-distance* of X_i to Q in the presence of obstacles

$$PDO(X_i, Q) = \frac{\sum_{l=1}^{d} \delta_{il} * f_{il}}{(\sum_{l=1}^{d} b_{il})^2} \tag{8}$$

where δ_{il} is the difference between X_i and Q on D_l, f_{il} is a value between [0,1] depending on whether $i \in KS_l$ and whether x_{il} is in the same segment with q_l on D_l. PDO(X_i, Q) can also be defined as the product of the average distance of X_i to Q on those dimensions on which X_i is within the set of *dimension-wise K nearest neighbor* to Q, and the weight to the average difference based on on how many dimensions on which X_i is within the set of K nearest neighbor to Q.

Given a data set DS of n data points $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ with D_l as the dimension l, l=1, 2, ..., d, a query point Q in the same data space, and a set of obstacle points $\mathbf{C} = \{C_1, C_2, ..., C_m\}$, we try to find a set PKS which consists of k data points from DS so that for any data point $X_i \in PKS$ and any data point $X_j \in DS - PKS$, the PDO(X_i, Q) is less than or equal to PDO(X_j, Q). The set PKS is the Pan-K Nearest Neighbor set of Q in DS in the presence of obstacles.

VI. Conclusion

In the paper we present our strategy to design the similarity search approaches in the presence of obstacles. On each dimension we divide the value range into segments based on the obstacle points and conduct our OPanKNN algorithm to find K nearest neighboring points for a given query point Q. In the future work, we will conduct more experiments on synthetic and real data sets to test and demonstrate the efficiency and effectiveness of our approach.

References

[1] White D.A. and Jain R. Similarity Indexing with the SS-tree. In *Proceedings of the 12th Intl. Conf. on Data Engineering*, pages 516–523, New Orleans, Louisiana, February 1996.

[2] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *SIGMOD '06*, pages 515–526, New York, NY, USA, 2006. ACM.

[3] C. C. Aggarwal. Towards meaningful high-dimensional nearest neighbor search by human-computer interaction. In *ICDE*, 2002.

[4] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973, 2001.

[5] Anthony K.H. Tung, Jean Hou and Jiawei Han. Spatial clustering in the presence of obstacles. In *ICDE '01: Proceedings of the 17th International Conference on Data Engineering*, page 359, Washington, DC, USA, 2001. IEEE Computer Society.

[6] D. A. Berchtold S., Keim and H.-P. Kriegel. The X-tree : An index structure for high-dimensional data. In *VLDB'96*, pages 28–39, Bombay, India, 1996.

[7] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *International Conference on Database Theory 99*, pages 217–235, Jerusalem, Israel, 1999.

[8] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[9] B. Cui, H. Shen, J. Shen, and K. Tan. Exploring bit-difference for approximate KNN search in high-dimensional databases. In *Australasian Database Conference, 2005.*, 2005.

[10] V. Estivill-Castro and I. Lee. Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets. In *In Proceedings of the 5th International Conference on Geocomputation*, pages 23–25, 2000.

[11] V. Estivill-Castro and I. Lee. Autoclust+: Automatic clustering of point-data sets in the presence of obstacles. In *TSDM '00: Proceedings of the First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining-Revised Papers*, pages 133–146, London, UK, 2001. Springer-Verlag.

[12] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation, 2003.

[13] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *The VLDB Journal*, pages 518–529, 1999.

[14] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *The VLDB Journal*, pages 506–515, 2000.

[15] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[16] Rothman, Milton A. *The laws of physics*. New York, Basic Books, 1963.

[17] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. *SIGMOD Rec.*, 27(2):154–165, 1998.

[18] Y. Shi and L. Zhang. A dimension-wise approach to similarity search problems. In *the 4th International Conference on Data Mining (DMIN'08)*, 2008.

[19] O. Z. University and O. R. Zaane. Clustering spatial data when facing physical constraints. In *In Proc. of the IEEE International Conf. on Data Mining*, pages 737–740, 2002.

[20] X. Wang and H. J. Hamilton. Dbrs: A density-based spatial clustering method with random sampling. In *PAKDD*, pages 563–575, 2003.

[21] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 194–205, 24–27 1998.

[22] Yong Shi, Yuqing Song and Aidong Zhang. A shrinking-based clustering approach for multidimensional data. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, pages 1389–1403, 2005.

[23] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.