Knowledge-Based Systems 90 (2015) 99–110

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Location difference of multiple distances based *k*-nearest neighbors algorithm

Shuyin Xia [a,*], Zhongyang Xiong [a], Yueguo Luo [a,c], Limei Dong [b], Guanghua Zhang [a]

[a] College of Computer Science, Chongqing University, Chongqing 400044, China
[b] Institute of Electrical Engineering and Information, Sichuan University, Chengdu 400015, China
[c] Network Information Center, Yangtze Normal University, Chongqing 408100, China.

ARTICLE INFO

Article history:
Received 17 March 2015
Revised 22 September 2015
Accepted 25 September 2015
Available online 3 October 2015

Keywords:
Location difference of multiple distances
k-nearest neighbors
Tree structure

ABSTRACT

k-nearest neighbors (kNN) classifiers are commonly used in various applications due to their relative simplicity and the absence of necessary training. However, the time complexity of the basic algorithm is quadratic, which makes them inappropriate for large scale datasets. At the same time, the performance of most improved algorithms based on tree structures decreases rapidly with increase in dimensionality of dataset, and tree structures have different complexity in different datasets. In this paper, we introduce the concept of "location difference of multiple distances, and use it to measure the difference between different data points. In this way, location difference of multiple distances based nearest neighbors searching algorithm (LDMDBA) is proposed. LDMDBA has a time complexity of O(log*dn*log*n*) and does not rely on a search tree. This makes LDMDBA the only kNN method that can be efficiently applied to high dimensional data and has very good stability on different datasets. In addition, most of the existing methods have a time complexity of O(*n*) to predict a data point outside the dataset. By contrast, LDMDBA has a time complexity of O(log*d*log*n*) to predict a query point in datasets of different dimensions, and, therefore, can be applied in real systems and large scale databases. The effectiveness and efficiency of LDMDBA are demonstrated in experiments involving public and artificial datasets.

1. Introduction

kNN algorithms are used to find *k*-nearest neighbors of data points in a dataset. These algorithms are used in many fields including feature selection [1], pattern recognition [2–3], clustering [4], classification noise detection [5], and classification [6–10]. The basic method of finding *k*-nearest neighbors of a point is to compute all Euclidean distances from the query point to all other data points. This method is known as the full search algorithm (FSA). The FSA has a complexity of O(n^2) so that it is very time consuming. To reduce the computation complexity, two classes of algorithms [11–27] were proposed.

The first class of algorithms creates a search tree to store data points. In these algorithms, the search strategy is bounded by branches of the search tree. Fukunaga and Narendra [11] used the hierarchical clustering technique to decompose data points and represented results using Ball tree. The structure of Ball tree is highly influenced by clustering algorithms [12]. To further improve the performance, fiveBall tree construction methods were introduced by Omohundro [13]. Friedman et al. [14] used spatial decomposition to generate a balanced *k*-dimensional tree. A refined version of *k*–*d* tree method was introduced by Sproull [15]. The algorithm based on *k*–*d* tree method performs better than the Ball tree on datasets with a small dimensionality. Kim and Park [16] used the ordered partition method to create a multiple branch tree. Mico et al. [17] used a prestored distance table to eliminate more impossible nodes. McNames [18] proposed a method based on a principal axis search tree (PAT). Wang and Gan [19] combined projected clusters and the PAT algorithm to reduce the computation time. Chen et al. [20] used winner update search method and a lower-bound tree (LB tree) to speed up the algorithm.

The other class of algorithms does not create a tree structure, but uses different methods. Cheng et al. [21] reduced the number of multiplication operations by using the min–max method. Bei and Gray [22] introduced the method of partial distortion to reduce the time of distance calculations. Ra and Kim [23] utilized the difference between mean values of the query point and other data points to eliminate impossible data points. Tai et al. [24] eliminated impossible data points using the projection values of data points. Nene and Nayar used a

* Corresponding author. Tel: +86 13883929561.
E-mail addresses: syxia@cqu.edu.cn, 380835019@qq.com (S. Xia), zyxiong@cqu.edu.cn (Z. Xiong), 362011523@qq.com (Y. Luo), 37638721@qq.com (L. Dong), guanghua0307@qq.com (G. Zhang).

http://dx.doi.org/10.1016/j.knosys.2015.09.028
0950-7051/

projection value to limit the distance from a query point [25]. Lu et al. [26] used the norm, mean value and variance to eliminate impossible data points. Lai et al. [27] utilized triangle inequality and projection values to accelerate the algorithm, which is now referred to as FkNNUPTI. These methods can speed up the process of finding nearest neighbors to some extent, but their time complexities have not been reduced any more so that they are still not enough efficient for different datasets.

Among all available methods, the PAT algorithm [18,19,27] and algorithm based on an orthogonal search tree (OST) [28] have good performance for many types of benchmark datasets. PAT method creates a principal axis search tree according to projection values of data points onto principal axes of tree nodes. The principal axis of a node is evaluated using data points in the node and the principal component analysis (PCA) [18]. OST method chooses an orthogonal vector from the orthonormal basis for a node. The orthogonal vector selected for the node is perpendicular to orthogonal vectors chosen for ancestors of the node. As the number of orthogonal vectors is small, this method only requires little computation time to calculate projection values. Furthermore, the method uses an additional inequality to eliminate impossible data points, which cannot be deleted using the node elimination inequality. However, the performance of the algorithm deteriorates with the increase in dimensions, which is shown in [28] and our experiments. The reason is that higher dimensions lead to higher complexity of tree structures. Thus, as our experiments show, the performance of various kNN methods using various tree structures reduces significantly. In addition, as the complexity of the tree structure corresponding to various datasets is different, the stability of such methods is poor as well.

In this paper, we introduce the concept of location differences and location difference based algorithm (LDMDBA) is proposed. The LDMDBA has a time complexity of $O(\log d n \log n)$ that is far less than FSA and most of other algorithms. The algorithm does not rely on any tree structure so that it can run efficiently on datasets of high dimensionality and has very good stability in various datasets. Furthermore, the algorithm has a time complexity of $O(\log d \log n)$ for predicting a data point outside datasets with different dimensionality. Therefore, unlike most existing kNN methods, the proposed algorithm can be applied in real systems and large scale databases. The performance of our method is compared with FSA and other six algorithms on datasets generated from different distributions and public benchmark datasets.

The main contributions of this paper can be summarized as follows:

- The concept of "location difference of multiple distances" and a new kNN algorithm LDMDBA are proposed. The LDMDBA has good prediction accuracy. In addition, its time complexity is equal to $O(\log d n \log n)$, which is no larger than existing algorithms.
- Unlike most existing fast kNN algorithms, our method does not rely on tree structures, so that its efficiency is not affected by dimensionality, and it can perform well on different datasets.
- The LDMDBA has a time complexity of $(\log d \log n)$ for predicting a new data point so that it is more suitable for large databases than existing algorithms.
- The effectiveness and robustness of the LDMDBA are explored and demonstrated by experiments on artificial and public real datasets.

The rest of this paper is organized as follows. In Section 2, our proposed method is presented and described in detail. Experimental results are given in Sections 3 and 4 and conclusions in Section 5.

2. Location difference of multiple distances based nearest neighbors searching algorithm

2.1. Location difference of multiple distances based factor (LDMDBF)

The proposed algorithm introduces a new measurement to measure location difference among different points. It requires only n (i.e. the number of data points) assignments of values of the measurement to each point. At the same time, the FSA usually need n^2 calculations to compute the Euclidean distance between each pair of points. Therefore, by avoiding these calculations, the efficiency of the algorithm can be improved considerably.

The method is based on the idea that neighbors have similar information about their location. The idea can be illustrated by the following example. Suppose, the information about the location of some people, including A, only contains the distances from these people to the North Pole. The problem is to find the people living near A. Although the spatial coordinates of these people and the Euclidean distances between them are unknown, they can be approximately estimated by the known information, i.e. the distances from those people to the North Pole. Let the distance from A to the North Pole be 100 m. The person whose distance to the North Pole is equal to 200 m is more likely to live near A than the person whose distance is equal to 10,000 m. This phenomenon can be further described using the example in Figs. 1 and 2.

As is shown in Fig. 1(a), O is set as a reference point. The distance from point A to point B is denoted as AB. As the point B is a near neighbor of point A, the distance BO is close to the distance AO. On the other hand, the point C is located farther from point A compared with point B, and the distance CO is much larger than the distances AO and BO. Thus, the distances from the reference point can be used to measure the location difference instead of the Euclidean distance and find nearest neighbors. However, in some cases, the distance to a single reference point is not sufficient to find nearest neighbors accurately. As shown in Fig. 1(b), although the point A' is located far from point A, the points are located on a same circle and they have approximately same distances to the center of the circle O (the reference point). Thus, multiple distances to some reference points can be used to measure location difference instead of the standard Euclidean distance and effectively find nearest neighbors. This method avoids computing the distances between each pair of points, and considerately improves the efficiency of the algorithm. In addition, it does not rely on any kind of tree structure. However, the neighbors found by multiple distances from some reference points are approximate. Therefore, in the proposed algorithm, the Euclidean distances between some data points are computed to further guarantee high accuracy.

Based on the above analysis, the location difference of multiple distances is defined as follows:

Definition 1. ***LDMDBF***

Given a database D, a point $A \in D$, and the norm denoted as $\|\cdot\|$: $R_{d^-} > R$, the distance from O_1 to A is denoted by $\mathrm{dis}(O_1 A)$. Higher dimensions need more reference data points to guarantee good prediction accuracy; at the same time, too many reference data points may lead to increase in computation time. Therefore, the number of reference data points is taken as $\log_2 d$. The values of the first i dimensions of the ith reference point O_i can be set to -1 and the other values are set to 1 (i.e. $O_i = (-1, -1, -1 \ldots -1, 1, 1, 1, \ldots, 1)$, where the number of values -1 is equal to i). The neighbors of point A found using the ith reference point are denoted by $\mathrm{neighbors}_i(A)$, and label (D') represents labels of all the points in D'. The label of A is determined by the sum of label $(\mathrm{neighbors}_i(A))$. Thus, the Location Difference of Multiple Distances Based Factor denotes the label of A that is computed using its neighbors found by the proposed method. LDMDBF(A) is equal to the