A Fast Nearest-Neighbor Algorithm Based on a Principal Axis Search Tree

James McNames, *Member, IEEE*

Abstract—A new fast nearest-neighbor algorithm is described that uses principal component analysis to build an efficient search tree. At each node in the tree, the data set is partitioned along the direction of maximum variance. The search algorithm efficiently uses a depth-first search and a new elimination criterion. The new algorithm was compared to 16 other fast nearest-neighbor algorithms on three types of common benchmark data sets including problems from time series prediction and image vector quantization. This comparative study illustrates the strengths and weaknesses of all of the leading algorithms. The new algorithm performed very well on all of the data sets and was consistently ranked among the top three algorithms.

Index Terms—Nearest neighbor, vector quantization encoding, principal components analysis, closest point, intrinsic dimension, post office problem.

1 Introduction

Given a data set of n_p points, $\{x_1, x_2, \ldots, x_{n_p}\}$, the k-nearest-neighbors problem is to find the k points that are closest to a query point, q, where $q, x_i \in \Re^{n_d}\ \forall i$. This problem is encountered in a wide range of applications including density estimation, pattern recognition [1], clustering [2], function approximation [3], time series prediction [4], document retrieval [5], optical character recognition [6], and vector quantization [2], [7]. In many of these applications, the computational cost of finding the nearest neighbors imposes practical limits on the data set size and the rate at which the application can operate. This has motivated the development of many fast nearest-neighbor algorithms.

This article introduces a new algorithm based on principal axis trees called PAT and compares it with other leading algorithms. Only algorithms that use the Euclidean metric,

$$D^2(x, q) \equiv \sum_{i=1}^{n_d} (x_i - q_i)^2,$$

to measure the distance between two points, are included in this study.

The performance of nearest-neighbor algorithms is usually specified by the preprocessing time, memory required, and average query time. For most applications, a moderate amount of preprocessing and memory allocation are acceptable if the average query time is small enough.

The following section reviews some of the most popular and effective methods of reducing query time. Section 3 discusses how the dimension of the data set affects these methods. Section 4 describes principal axis trees (PAT) in detail. Section 5 specifies the other algorithms included in a comparative empirical study, discusses how the query time was measured, and describes how user-specified parameters were chosen. Section 6 reports the performance of PAT and the other algorithms on a variety of benchmark problems. PAT is shown to have excellent performance across a broad range of data sets.

2 Elimination Criteria

Many nearest-neighbor algorithms have been proposed to overcome the large computational cost of the obvious brute force approach. Typically, these algorithms employ one or more elimination criteria. Each estimates a lower bound on the distance between a query point and a point, or set of points, in the data set. If the lower bound is greater than the distance to the kth nearest neighbor found so far, the point can be eliminated without explicitly calculating the distance to that point. This section describes three of these elimination criteria that are included in principal axis trees (PAT).[1]

2.1 Partial Distance Search

Full-search improvements apply an elimination criterion to every point in the data set. Cheng et al. [7] originally proposed an algorithm called partial distance search (PDS) that is arguably the most popular full-search improvement. The algorithm consists of a simple modification of the brute force search: during the calculation of the distance, if the partial sum of square differences exceeds the distance to the nearest neighbor found so far, the calculation is aborted. Like the brute force search, PDS does not require any preprocessing or storage and the performance of PDS is almost always substantially better.

• *The author is with the Department of Electrical & Computer Engineering, Portland State University, PO Box 751, Portland State University, Portland, OR 97207-0751. E-mail: mcnames@ee.pdx.edu.*

Manuscript received 24 Jan. 2000; revised 13 Oct. 2000; accepted 20 Feb. 2001.
Recommended for acceptance by C. Brodley.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 111291.

1. More thorough reviews of common elimination criteria are given in [8], [9].