IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 11, November 2015*

# A Review on Nearest Neighbour Techniques for Large Data

**Deoyani Sonawane[1], Prof. P. M. Yawalkar[2]**

Student of M.E., Computer Dept, MET BKC Adgaon, Nasik, Savitribai Phule Pune University, Maharashtra, India[1]

Professor, Computer Dept, MET BKC Adgaon, Nasik, Savitribai Phule Pune University, Maharashtra, India[2]

**Abstract:** This for many computer innovation and machine learning problems, key of good performance is large data set. However, in many computer innovation and machine learning algorithms consist of finding nearest neighbour matches in large data set is computationally expansive part. New algorithms are developed for approximate nearest neighbour matching and evaluation and then compare them with preceding algorithms. For matching high dimensional features most efficient algorithms are essential. Perhaps the Locality sensitive hashing (LSH) technique is best known hashing based nearest neighbour technique which requires multiple numbers of hash functions with the property that the hashes of elements that are close to each other are also likely to be close. Variants of LSH such as multi-probe LSH improves the high storage costs by reducing the number of hash tables, and LSH Forest adapts better to the data without requiring hand tuning of parameters. for finding the best algorithm to search a particular data set, Optimal nearest neighbour algorithm and its parameters depend on the large data set characteristics and gives description of automated configuration procedure. In order to scale to very large data sets that would otherwise not fit in the memory. When dealing with such large data, possible solutions include performing some dimensionality reduction on the data, keeping the data on the disk and loading only parts of it in the main memory or distributing the data on several computers and using a distributed nearest neighbour search algorithm.

**Keywords:** Nearest Neighbour Search, Big Data, Approximate Search.

## 1. INTRODUCTION

Searching most similar matches to large data also get called as nearest neighbour matching, and it is most computationally expensive part of many computer innovation algorithms. Having a well-planned algorithm for performing fast nearest neighbour matching in large data sets can bring speed improvements of several orders of magnitude to many applications. Examples of such problems include finding the best matches for local image features in large data sets, clustering local features into visual words using the k-means or similar algorithms, global image feature matching for scene recognition, human pose estimation, matching deformable shapes for object recognition. The nearest neighbour search problem is also of major importance in many other applications, including machine learning, document retrieval, data compression, bio-informatics, and data analysis. It has been shown that using large training sets is key to obtaining good real-life performance from many computer vision methods [8]. Today the Internet is a vast resource for such training data, but for large data sets the performance of the algorithms employed quickly becomes a key issue. When working with high dimensional features, as with most of those encountered in computer vision applications (image patches, local descriptors, global image descriptors), there is often no known nearest-neighbour search algorithm that is exact and has acceptable performance. To obtain a speed improvement, many practical applications are forced to settle for an approximate search, in which not all the neighbours returned are exact, meaning some are approximate but typically still close to the exact neighbours. In practice it is common for approximate nearest neighbour search algorithms to provide more than 95 present of the correct neighbours and still be two or more orders of magnitude faster than linear search. In many cases the nearest neighbour search is just a part of a larger application containing other approximations and there is very little loss in performance from using approximate rather than exact neighbours. Hence work on the most promising nearest neighbour search algorithms in the literature is required and propose new algorithms and improvements to existing ones

## 2. RELATED WORK

Use of the BBD-tree, $(1+\varepsilon)$-approximate nearest neighbour queries for a set of n points in $R^d$ can be answered in O $(C_{d,\varepsilon} \log_n)$ time, where $C_{d,\varepsilon} \leq d\ [1+6_d/\varepsilon\ e]^{d}$ is a constant depending only on dimension and $\varepsilon$. The data structure uses optimal O $(d_n)$ space and can be built in O $(d_n \log_n)$ time. The algorithms have presented are simple (especially the midpoint splitting rule) and easy to implement. Empirical studies indicate good performance on a number of different point distributions. Unlike many recent results on approximate nearest neighbour searching, the pre-processing is independent of $\varepsilon$, and so different levels of precision can be provided from one data structure. Although constant factors in query time grow exponentially with dimension, constant factors in space and pre-processing time grow only linearly in d.
We have also shown that the algorithms can be generalized to enumerate approximate k-nearest neighbours in additional O $(k_d \log_n)$ time. Using auxiliary data structures,

it is possible to handle point insertions and deletions in O($\log_n$) time each. There are a number of important open problems that remain. One is that of improving constant factors for query time. Given the practical appeal of a data structure of optimal O($d_n$) size for large data sets, an important question is what lower bounds can be established for approximate nearest neighbor searching using data structures of this size. Another question is whether the approximate kth nearest neighbour can be computed in time that is polylogarithmic in both n and k [1]. In the context of nearest neighbour query in a high dimensional space with a structured data set SIFT descriptors in 128 dimensions in application, application have demonstrated that various randomisation techniques give enormous improvements to the performance of the KD-tree algorithm. The basic technique of randomisation is to carry out simultaneous searches using several trees, each one constructed using a randomly rotated (or more precisely, reflected) dataset. This technique can lead to an improvement from about 75% to 88% in successful search rate, or a 3 times search speed-up with the same performance. Best results are obtained by combining this technique with a rotation of the data set to align it with its principal axis directions using PCA and then applying random House holder transformations that preserve the PCA subspace of appropriate dimension. This leads to a success rate in excess of 95%. Tests with synthetic high-dimensional data led to even more dramatic improvements, with up to 7-times diminished error rate with the NKD-tree algorithm alone [2]. Overview of efficient algorithms for the approximate and exact nearest neighbour problems is described. The goal is to pre-process a dataset of objects (e.g., images) so that later, given a new query object, one can quickly return the dataset object that is most similar to the query. The problem is of significant interest in a wide variety of areas.

The goal is twofold. Survey a family of nearest neighbour algorithms that are based on the concept of locality sensitive hashing. Many of these algorithms have already been successfully applied in a variety of practical scenarios. Describe a recently discovered hashing-based algorithm, for the case where the objects are points in the d-dimensional Euclidean space. As it turns out, the performance of this algorithm is provably near-optimal in the class of the locality-sensitive hashing algorithms. LSH family achieves a near-optimal separation between the collision probabilities of close and far points. An interesting feature of this family is that it effectively enables the reduction of the approximate nearest neighbour problem for worst-case data to the exact nearest neighbour problem over random (or pseudorandom) point configuration in low-dimensional spaces. Currently, the new family is mostly of theoretical interest. This is because the asymptotic improvement in the running time achieved via a better separation of collision probabilities makes a difference only for a relatively large number of input points. Nevertheless, it is quite likely that one can design better pseudorandom point configurations which do not suffer from this problem [3].Automatic algorithm configuration allows a user to achieve high performance in approximate nearest neighbour matching by calling a single library routine. The user need only provide an example of the type of dataset that will be used and the desired precision, and may optionally specify the importance of minimizing memory or build time rather than just search time. All remaining steps of algorithm selection and parameter optimization are performed automatically. Experiments, have found that either of two algorithms can have the best performance, depending on the dataset and desired precision. One of these is an algorithm have developed that combines two previous approaches: searching hierarchical k-means trees with a priority search order. The second method is to use multiple randomized kd trees. Have demonstrated that these can speed the matching of high-dimensional vectors by up to several orders of magnitude compared to linear search.

The use of automated algorithm configuration will make it easy to incorporate any new algorithms that are found in the future to have superior performance for particular datasets. It can be seen that performance improves with the number of randomized tree sup to a certain point(about20randomtrees in this case) and that increasing the number of random trees further leads to static or decreasing performance. The memory over head of using multiple random trees increases linearly with the number of trees. The hierarchical k-means tree algorithm has the highest performance for some datasets. However, one disadvantage of this algorithm is that it often has a higher tree-build time than the randomized kd-trees. The build time can be reduced significantly by doing a small number of iterations in the k-means clustering stage instead of running it until convergence [4]. Similarity search (nearest neighbour search) is a problem of pursuing the data items whose distances to a query item are the smallest from a large database. Various methods have been developed to address this problem, and recently a lot of efforts have been devoted to approximate search. Present a survey on one of the main solution called as hashing, which has been widely studied since the pioneering work locality sensitive hashing. Divide the hashing algorithms in to two main categories: locality sensitive hashing, which designs hash functions without exploring the data distribution and learning to hash, which learns hash functions according the data distribution, and review them from various aspects, including hash function design and distance measure and search scheme in the hash coding space. The problem of sub linear time approximate similarity search for a class of learned metrics is found. While randomized algorithms such as LSH have been employed extensively to mitigate the time complexity of identifying similar examples, particularly in vision their use has been restricted to generic measures for which the appropriate hash functions are already defined; that is, direct application to learned metrics was not possible [5]. The problem of approximate nearest neighbour (ANN) search for visual descriptor indexing. Most spatial partition trees, such as KD trees, VP trees, and so on, follow the hierarchical binary space partitioning framework. The key effort is to design different partition functions (hyper plane or hyper sphere) to divide the points so that 1) the data points can be well grouped to support effective NN candidate location and 2) the partition functions can be quickly evaluated to support efficient NN candidate location. Design a trinary-projection direction-based partition function. The trinary-projection direction is defined as a combination of a few coordinate axes with the weights being 1 or -1. Pursue the projection direction using the widely adopted maximum variance criterion to guarantee good space partitioning and find fewer coordinate axes to guarantee efficient partition function evaluation. Present a coordinate-wise enumeration algorithm to find the principal trinary-projection direction. In addition, provide an extension using multiple randomized trees for improved performance.Present a novel hierarchical spatial partition tree for approximate nearest neighbour search. The key idea is using a trinary projection direction, a linear combination of a few coordinate axes with weights being 1 or -1, to form the partition hyper plane. The superiority of our approach comes from two aspects: 1) fast projection operation at internal nodes in traversing, only requiring a few addition/subtraction operations, which leads to high search efficiency, and 2) good space partition guaranteed by a large variance along the projection direction for partitioning data points, which results in high search accuracy. [6]Introduced a method to enable efficient approximate similarity search for learned metrics, and experiments show good results for a variety of data sets, representations, and base metrics. Main contribution is a new algorithm to construct theoretically sound locality-sensitive hash functions for both implicit and explicit parameterizations of a Mahalanobis distance. For high-dimensional data, derive simultaneous implicit updates for both the hash function and the learned metric. Experiments demonstrate our technique's accuracy and flexibility for a number of large-scale search tasks. In future work, intend to explore online extensions to our algorithm that will allow similarity constraints to be processed in an incremental fashion, while still allowing intermittent queries. also interested in considering generalizations of our implicit hashing formulation to accommodate alternative kernelized metric learning algorithms, and in pursuing active constraint selection methods within our framework.[7]. Large training sets are important for many computer vision and machine learning problems. Finding nearest neighbour matches to high dimensional vectors representing training data is most computationally expensive. Thus a new algorithm is proposed for approximate nearest neighbour matching and is evaluated and compared it with previous algorithms. For matching high dimensional features, two algorithms are most efficient: the randomized k-d forest and a new algorithm proposed the priority search k-means tree. Also a new algorithm is proposed for matching binary features by searching multiple hierarchical clustering trees. In order to scale to very large data sets, distributed nearest neighbour matching framework is proposed that can be used with any of the algorithms described in the paper. All this research has been released as an open source library called Fast library for approximate nearest neighbours (FLANN). The problem of fast nearest neighbour search in high dimensional spaces is addressed a core problem in many computer vision and machine learning algorithms and which is the most computationally expensive part of these algorithms. Present and compare the algorithm have found to work best at fast approximate search in high dimensional spaces: the randomized k-d trees and a newly introduced algorithm, the priority search k-means tree. [8]

## 3. CONCLUSION

The nearest neighbour search problem has major importance in many applications, including machine learning, document retrieval, data compression, bio-informatics, and data analysis. For that Use of large training sets is key to obtain good real-life performance from many computer vision methods. Today the Internet is a vast resource for such training data but for large data sets the performance of the algorithms employed quickly becomes a key issue. Hence designing new algorithms and improvements to existing ones are needed. Variations of the k-d tree using non-axis-aligned partitioning hyper planes have shown trinary projection tree, but trinary projection tree are less efficient as compared to randomized k-d tree decomposition.Applying hashing technique in the implementation of nearest neighbour may certainly help in improving the results over K-d tree decomposition.

## REFERENCES

[1]S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," J. ACM, vol. 45, no. 6, pp. 891– 923, 1998.
[2] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2008, Pp.1–8.
[3] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbour in high dimensions," Commun. ACM, vol. 51, no. 1, pp. 117–122, 2008.
[4]M. Muja and D.G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in Proc. Int. Conf. Computer Vis. Theory Appl., 2009, pp. 331–340.
[5] Brian Kulis, Prateek Jain, and Kristen Grauman, "Fast Similarity Search for Learned Metrics"IEEE Transaction on pattern analysis and machine intelligence vol. 31, no.12, December 2009.
[6] Jingdong Wang, Naiyan Wang, You Jia, Jian Li, Gang Zeng, HongbinZha, , and Xian-Sheng Hua, "Trinary-Projection Trees for Approximate Nearest Neighbor Search", IEEE, Transaction on pattern analysis and machine intelligence vol. 36, no. 2, February 2014.
[7] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and JianqiuJi "Hashing for Similarity Search: A Survey"August 13, 2014.
[8] Marius Muja and David G. Lowe, "Scalable Nearest Neighbour Algorithms for High Dimensional Data" IEEE Transaction on pattern analysis and machine intelligence, vol. 36, no. 11, November 2014.