Volume 5, Issue 9, September 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Reducing the Dimensional Dependence for Rank-Based Similarity Search

Jeevan Arjun Shelke, Viresh Shivshankar Humnabadkar, Tushar Tinajirao Kale,
Girish Gundopant Kulkarni, Parag. S. Kulkarni
NBN Sinhgad School of Engineering Ambegaon,
Pune, Maharashtra, India

***Abstract:** In this paper we try to introduce a data structure for k-NN search, the Rank Cover Tree (RCT). The pruning tests for RCT rely on the comparison of similarity values not on the other properties of the underlying space, such as the triangle inequality. Objects are selected according to their ranks with respect to the query object, allowing much tighter control on the overall execution costs. Theoretical analysis shows that with very high probability, the RCT returns a correct query result in time that depends very competitively on a measure of the intrinsic dimensionality of the data set. The experimental results for the RCT show that non-metric pruning strategies for similarity search can be practical even when the representational dimension of the data is extremely high. They also show that the RCT is capable of meeting or exceeding the level of performance of state-of-the-art methods that make use of metric pruning or other selection tests involving numerical constraints on distance values.*

***Keywords:** Nearest neighbor search, intrinsic dimensionality, rank-based search.*

I. INTRODUCTION

The fundamental operations in data mining are classification, cluster analysis, and outlier detection, and this might be most widely-encountered is that of similarity search. Similarity search is that the foundation of k-nearest-neighbor (k-NN) classification, which frequently produces competitively low error rates in practice, significantly once the quantity of categories is massive [26]. The error rate of nearest-neighbor classification has been shown to be 'asymptotically optimal' because the training set size will increase [14]. For clustering, several of the foremost effective and commonwaysneed the determination of neighbor sets based mostlyat a considerable proportion of the data set objects [26]: examples includehierarchical(agglomerative)strategieslikeROCK [22] and CURE [23]; density-based strategieslikeDBSCAN [17], OPTICS [3], and SNN [16]; and non-agglomerative shared-neighbor clustering [27]. A content-based filtering strategy for recommender systems and anomaly detection strategies [11] ordinarily build use of k-NN techniques, either through the direct use of k-NN search, or by means that of k-NN cluster analysis. a very common density-based live, the local Outlier factor (LOF), depends heavily on k-NN set computation to see the denseness of the data within the section of the test point [8].

For data mining applications based on similarity search, data objects are usually modeled as feature vectors of attributes that some measure of similarity is defined. Often, the data can be modeled as a subset $S \subset \mathcal{U}$ belonging to a metric space $M = (\mathcal{U}, d)$over some domain$\mathcal{U}$, with distance measure $d: \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}^{+}$satisfying the metric postulates. Given a query point$q \,\epsilon\, \mathcal{U}$, similarity queries over S are of two general types:

- *K-nearest neighbor queries* report a set $\mathcal{U} \subseteq S$ of size k elements satisfying $d(q,u) \leq d(q,v)$for all $u \in \mathcal{U}$ and$v \in S \backslash \mathcal{U}$.
- Given a real value$r \geq 0$, range queries report the set$\{v \in S | d(q,v) \leq r\}$.

While a k-NN query result is not necessarily unique, the range query result clearly is.

Motivated a minimum of in part by the impact of similarity search on issues in data mining, machine learning, pattern recognition, and statistics, the planning and analysis of scalable and effective similarity search structures has been the subject of intensive research for several decades. Till comparatively in recent years, most data structures for similarity search targeted low-dimensional real vector spacer presentations and therefore the Euclidean or alternative L-P distance metrics. However, several public and commercial data sets available these days are additionalnaturallydepictedas vectors spanning several hundreds or thousands of feature attributes, which might be real or integer-valued, ordinal or categorical, or perhaps a mix of those types. This has spurred the development of search structures for additional general metric areas, like the Multi-Vantage-Point Tree [7], the Geometric Near-neighbor Access Tree (GNAT) [9], spatial Approximation Tree (SAT), the M-tree [13], and (more recently) the cover Tree (CT) [6].

Despite their various advantages, spatial and metric search structures are both limited by an effect often referred to as the curse of dimensionality. One way in which the curse may manifest itself is in a tendency of distances to concentrate strongly around their mean values as the dimension increases. Consequently, most pairwise distances become difficult to distinguish, and the triangle inequality can no longer be effectively used to eliminate candidates from consideration along search paths. Evidence suggests that when the representational dimension of feature vectors is high (roughly 20 or more [5]), traditional similarity search accesses an unacceptably-high proportion of the data elements, unless the underlying data distribution has special properties [6], [12]. Even though the local neighborhood information employed by data mining applications is useful and meaningful, high data dimensionality tends to make this local information very expensive to obtain.

The performance of similarity search indices depends crucially on the way in which they use similarity information for the identification and selection of objects relevant to the query. Virtually all existing indices make use of numerical constraints for pruning and selection. Such constraints include the triangle inequality (a linear constraint on three distance values), other bounding surfaces defined in terms of distance (such as hyper cubes or hyperspheres) [25], [32], range queries involving approximation factors as in Locality-Sensitive Hashing (LSH) [19], [30], or absolute quantities as additive distance terms [6]. One serious drawback of such operations based on numerical constraints such as the triangle inequality or distance ranges is that the number of objects actually examined can be highly variable, so much so that the overall execution time cannot be easily predicted.

In an attempt to improve the scalability of applications that depend upon similarity search, researchers and practitioners have investigated practical methods for speeding up the computation of neighborhood information at the expense of accuracy. For data mining applications, the approaches considered have included feature sampling for local outlier detection [15], data sampling for clustering and approximate similarity search for k-NN classification (as well as in its own right). Examples of fast approximate similarity search indices include the BD-Tree, a widely-recognized benchmark for approximate k-NN search; it makes use of splitting rules and early termination to improve upon the performance of the basic KD-Tree. One of the most popular methods for indexing, Locality-Sensitive Hashing [19], [30], can also achieve good practical search performance for range queries by managing parameters that influence a tradeoff between accuracy and time. The spatial approximation sample hierarchy (SASH) similarity search index [29] has had practical success in accelerating the performance of a shared-neighbor clustering algorithm [27], for a variety of data types.

In this paper, we propose a new similarity search structure, the Rank Cover Tree (RCT), whose internal operations completely avoid the use of numerical constraints involving similarity values, such as distance bounds and the triangle inequality. Instead, all internal selection operations of the RCT can be regarded as ordinal or rank-based, in that objects are selected or pruned solely according to their rank with respect to the sorted order of distance to the query object. Rank thresholds precisely determine the number of objects to be selected, thereby avoiding a major source of variation in the overall query execution time. This precision makes ordinal pruning particularly well-suited to those data mining applications, such as k-NN classification and LOF outlier detection, in which the desired size of the neighborhood sets is limited. As ordinal pruning involves only direct pairwise comparisons between similarity values, the RCT is also an example of a combinatorial similarity search algorithm [21].

The main contributions of this paper are as follows:

- A similarity search index within which only ordinal pruning is employed for node selection—no use is formed of metric pruning or of different constraints involving distance values.
- Experimental proof indicating that for practical k-NN search applications, our rank-based technique is extremely competitive with approaches that build specific use of similarity constraints. Specially, it comprehensively outperforms progressive implementations of each LSH and therefore the BD-Tree, and is capable of achieving practical speedups even for data sets of extraordinarily high representational dimensionality.
- A formal theoretical analysis of performance showing that RCT k-NN queries efficiently manufacture correct results with very high chance. The performance bounds are} expressed in terms of a measure of intrinsic spatiality (the expansion rate [31]), independently of the complete objective dimension of the information set. The analysis shows that by accepted polynomial sublinear dependence of question value in terms of the quantity of data objects n, the dependence on the intrinsic dimensionality is less than the other known search index achieving sublinear performance in n, whereas still achieving terribly high accuracy.

To the best of our knowledge, the RCT is the first practical similarity search index that both depends solely on ordinal pruning, and admits a formal theoretical analysis of correctness and performance. A preliminary version of this work has appeared in [28].

The remainder of this paper is organized as follows. Section 2 briefly introduces two search structures whose designs are most-closely related to that of the RCT: the rank-based SASH approximate similarity search structure [29], and the distance-based Cover Tree for exact similarity search [6]. Section 3 introduces basic concepts and tools needed to describe the RCT. Section 4 presents the algorithmic details of the RCT index.

II. RELATED WORK

This paper are going to be involved with two recently-proposed approaches that on the surface appear quite dissimilar: the SASH heuristic for approximate similarity search [29], and the cover Tree for exact similarity search [6]. Of the two, the SASH is often considered combinatorial, whereas the cover Tree makes use of numerical constraints. Before formally stating the new results, we have a tendency to first provide an outline of each the SASH and also the cover Tree.

2.1. SASH

For large data sets, we should use data structures that permit far better performance on the amount N of items within the database. This will be illustrated by the role that R-Trees play in the efficiency DBSCAN. To manage very massive data sets, we've SASH. The SASH makes minimal assumptions concerning the nature of the metric for associative queries. It additionally doesn't enforce a partition of the search space, as for instance R-Trees do. For approximate k-NN (k-ANN) queries on the large sets, the SASH systematically returns a high proportion of truth k-NNs at speeds of roughly two orders of magnitude quicker than sequential search. The SASH has already been successfully applied to clustering and navigation of very large, very high dimensional text data sets. The SASH internally uses a k-NN query on small sets. A (SASH) may be a multi-level structure recursively constructed by building a SASH on a half-sized random sample S'⊂S of the object set S, so connecting every object remaining outside S' to many of its approximate nearest neighbors from at intervals S'. Queries are processed by initial locating approximate neighbors at intervals sample S', so exploitation the pre-established connections to find neighbors at intervals the rest of the information set. The SASH index depends on a pairwise distance measure, however otherwise makes no assumptions relating to the illustration of the information, and doesn't use constellation difference for pruning of search ways.

SASH construction is in batch fashion, with points inserted in level order. Every node υ seems at one level of the structure: if the leaf level is level 1, the probability of υ being allotted to level $j$is $\frac{1}{2^j}$. Every node υ is connected to at the most p parent nodes, for a few constant $p \geq 1$, chosen as approximate nearest neighbors from among the nodes at one level more than υ. The SASH guarantees a continuing degree for every node by guaranteeing that every will function the parent of at the most $c = 4p$ children; any decide to attach more than c children to an individual parent w is resolved by accepting solely the c nearest children to w, and reassigning rejected kids to close surrogate oldsters whenever necessary.

Similarity queries are performed by establishing an upper limit $k_j$on the number of neighbor candidates to be retained from level $j$of the SASH, dependent on both $j$and the number of desired neighbors'k. The search starts from the root and progresses by successively visiting all children of the retained set for the current level, and then reducing the set of children to meet the quota for the new level, by selecting the k_j elements closest to the query point.

2.2. Cover Trees and the Expansion Rate

In [31], Karger and Ruhl introduced a measure of intrinsic dimensionality as a means of analyzing the performance of a local search strategy for handling nearest neighbor queries. In their method, a randomized structure resembling a skip list is used to retrieve pre-computed samples of elements in the vicinity of points of interest. Eventual navigation to the query is then possible by repeatedly shifting the focus to those sample elements closest to the query, and retrieving new samples in the vicinity of the new points of interest.

The expansion rate of S is the minimum value of δsuch that the above condition holds, subject to the choice of minimum ball set size *b*.

The Cover Tree can also be used to answer approximate similarity queries using an early termination strategy. However, the approximation guarantees only that the resulting neighbors are within a factor of $1 + \varepsilon$of the optimal distances, for some error value ε> 0.

III. RANK COVER TREE

Some of the design features of the SASH similarity search structure and the Cover Tree are used in proposed Rank Cover Tree. Like the SASH (and unlike the Cover Tree) we shall see that its use of ordinal pruning allows for tight control on the execution costs associated with approximate search queries. By restricting the number of neighboring nodes to be visited at each level of the structure, the user can reduce the average execution time at the expense of query accuracy.

In the RCT, a tree T imposed on a random leveling $\mathcal{L}$ of the data set S. Given any$0 \leq l \leq h$, the subtree$T_l \subset T$spanning only the level sets $L_l, L_{l+1}, \ldots \ldots, L_{h-1} \in \mathcal{L}$is also a Rank Cover Tree for the set L_l; T_l will be referred to as the partial Rank Cover Tree of *T* for level *l*. Now, for any $u \in L_l$ and any choice of *l* < *j* < *h*, we define $a_j(u) \in L_j$to be the unique ancestor of *u* in T_l at level *j*.

RCT search proceeds from the root of the tree, by identifying at each level *j* a set of nodes V_j (the cover set) whose subtrees will be explored at the next iteration. For an item *u* to appear in the query result, its ancestor at level *j* must appear in the cover set associated with level *j*. V_j is chosen so that, with high probability, each true k-nearest neighbor *u* satisfies the following conditions: the ancestor $u_j = a_j(u)$ of u is contained in V_j, and for any query point q, the rank $\rho_j\ (q, u_j)$ of u_j with respect to Lj is at most a level-dependent coverage quota$k_j = \omega\, max \left\{\frac{k}{\Delta^j}, 1\right\}$.

The real-valued parameter ω is the coverage parameter. It influences the extent to which the number of requested neighbors k impacts upon the accuracy and execution performance of RCT construction and search, while also establishing a minimum amount of coverage independent of k.

Offline construction of the RCT is performed by level ordered insertion of the items of the random leveling, with the insertion of node $v \in L_j$ performed by first searching for its nearest neighbor $w \in L_{j+1}$using the partial Rank Cover TreeT_{j+1}, and then linking *v* to *w*.

A Rank Cover Tree T will be said to be well-formed if for all nodes $u \in T$ with parent $v \neq u$, the parent *v* is the nearest neighbor of *u* from among the nodes at that level—that is, if $\rho_{\lambda(u)+1}(u, v) = 1$. In the analysis to follow, we determine conditions upon the choice of the coverage parameter ω for which the construction algorithm produces a well-formed RCT with high probability. We also derive bounds on the error probability for k-NN queries on a well-formed RCT.

IV. CONCLUSION

We try introducing a replacement data structure for K-NN, the Rank cover Tree, that uses direct comparisons between distance values. The RCT construction andqueryexecutioncosts is freelanceontherepresentationaldimension of the data, however are oftenanalyzedprobabilistically in terms of a measure of intrinsicdimensionality.The theoretical analysis, shows that the RCT outperforms its twonearest relatives—the cover Tree and SASH structures -in several cases, and consistentlyoutperforms the E2LSH implementation of LSH, classical indices such as the KD-Tree and BD-Tree, and—for data sets of high (but sparse) dimensionality— the KD-Tree ensemble technique FLANN.

REFERENCES

[1] Michael E. Houle And Michael Nett,” Rank-Based Similarity Search: ReducingThe Dimensional Dependence”, IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 37, No. 1, January 2015.

[2] I. Abraham, D. Malkhi, and O. Dobzinski, “LAND: Stretch (1 + epsilon) locality-aware networks for DHTs,” in Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithm, 2004, pp. 550–559.

[3] A. Andoni and P. Indyk. (2005). E2LSH 0.1: User Manual. [Online]. Available: www.mit.edu/andoni/LSH/, 2005.

[4] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “OPTICS: Ordering points to identify the clustering structure,” in Proc. ACM SIGMOD Int. Conf. Manag. Data, 1999, pp. 49–60.

[5] A. Asuncion and D. J. Newman. (2007). UCI machine learning repository. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[6] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ”nearest neighbor” meaningful?” in Proc. 7th Int. Conf. Database Theory, 1999, pp. 217–235.

[7] A. Beygelzimer, S. Kakade, and J. Langford, “Cover trees for nearest neighbor,” in Proc. 23rd Int. Conf. Mach. Learn., 2006, pp. 97– 104.

[8] T. Bozkaya and M. Ozsoyoglu, “Indexing large metric spaces for similarity search queries,” ACM Trans. Database Syst., vol. 24, no. 3, pp. 361–404, 1999.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying density-based local outliers,” SIGMOD Rec., vol. 29, no. 2, pp. 93–104, 2000.

[10] S. Brin, “Near neighbor search in large metric spaces,” in Proc. 21th Int. Conf. Very Large Data Bases, 1995, pp. 574–584.

[11] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou, “On hierarchical routing in doubling metrics,” in Proc. 15thAnnu. ACMSIAM Symp. Discrete Algorithm, 2005, pp. 762–771.

[12] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, 2009.

[13] E. Ch_avez, G. Navarro, R. Baeza-Yates, and J. L. Marroqu_ın, “Searching in metric spaces,” ACM Comput. Surv., vol. 33, no. 3, pp. 273–321, 2001.

[14] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,” in Proc. 23rd Int. Conf. Very Large Data Bases, 1997, pp. 426–435.

[15] T. Cover, and P. Hart, “Nearest neighbor pattern classification,” IEEE Trans. Inf. Theory, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[16] T. de Vries, S. Chawla, and M. E. Houle, “Finding local anomalies in very high dimensional space,” in Proc. IEEE Int. Conf. Data Mining, 2010, pp. 128–137.

[17] L. Ert€oz, M. Steinbach, and V. Kumar, “Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data,” in Proc. 3rd SIAM Int. Conf. Data Mining, 2003, p. 1.

[18] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proc. 2nd Int. Conf. Knowl. Discov. Data Mining, 1996, pp. 226–231.

[19] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, “The Amsterdam library of object images,” Int. J. Comput. Vis., vol. 61, no. 1, pp. 103–112, 2005.

[20] A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in Proc. 25th Int. Conf. Very Large Data Bases, 1999, pp. 518–529.

[21] J. E. Goodman and J. O’Rourke, Eds., Handbook of Discrete and Computational Geometry. Cleveland, OH, USA: CRC Press, 1997.

[22] N. Goyal, Y. Lifshits, and H. Sch€utze, “Disorder inequality: A combinatorial approach to nearest neighbor search,” in Proc. Intern. Conf. Web Search Web Data Mining, 2008, pp. 25–32.

[23] S. Guha, R. Rastogi, and K. Shim, “ROCK: A robust clustering algorithm for categorical attributes,” Inf. Syst., vol. 25, no. 5, pp. 345–366, 2000.

[24] S. Guha, R. Rastogi, and K. Shim, “CURE: An efficient clustering algorithm for large databases,” Inf. Syst., vol. 26, no. 1, pp. 35–58, 2001.

[25] A. Gupta, R. Krauthgamer, and J. R. Lee, “Bounded geometries, fractals, and low-distortion embeddings,” in Proc. 44th Annu. IEEE Symp. Foundations Comput. Sci., 2003, p. 534.

[26] A. Guttman, “R-trees: A dynamic index structure for spatial searching,” in Proc. Annu. Meeting, 1984, pp. 47–57.

[27] J. Han, and M. Kamber, Data Mining: Concepts and Techniques, San Francisco, CA, USA: Morgan Kaufmann, 2006.

[28] M. E. Houle, “The relevant set correlation model for data clustering,” Statist. Anal. Data Mining, vol. 1, no. 3, pp. 157–176, 2008.

[29] M. E. Houle and M. Nett, “Rank cover trees for nearest neighbor search,” in Proc. Int. Conf. Similarity Search Appl., 2013, pp. 16–29.

[30] M. E. Houle and J. Sakuma, “Fast approximate similarity search in extremely high-dimensional data sets,” in Proc. 21st Intern. Conf. Data Eng., 2005, pp. 619–630.

[31] P. Indyk, and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in Proc. 30th ACM Symp. Theory Comput., 1998, pp. 604–613.

[32] D. R. Karger, and M. Ruhl, “Finding nearest neighbors in growthrestricted metrics,” in Proc. 34th ACM Symp. Theory Comput., 2002, pp. 741–750.

[33] N. Katayama and S. Satoh, “The SR-tree: An index structure for high-dimensional nearest neighbor queries,” in Proc. ACM SIGMOD Int. Conf. Manag. Data, May 1997, pp. 369–380.

[34] R. Krauthgamer and J. R. Lee, “The black-box complexity of nearest neighbor search,” in Proc. 31st Int. Colloquium Automata, Lang. Programm., 2004, pp. 858–869.

[35] R. Krauthgamer and J. R. Lee, “Navigating nets: Simple algorithms for proximity search,” in Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms, 2004, pp. 798–807.

[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.