



Nearest neighbor search

Nearest neighbor search (NNS), as a form of **proximity search**, is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point. Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values.

Formally, the nearest neighbor (NN) search problem is defined as follows: given a set S of points in a space M and a query point $q \in M$, find the closest point in S to q . Donald Knuth in volume 3 of *The Art of Computer Programming* (1973) called it the **post-office problem**, referring to an application of assigning to a residence the nearest post office. A direct generalization of this problem is a k -NN search, where we need to find the k closest points.

Most commonly M is a metric space and dissimilarity is expressed as a distance metric, which is symmetric and satisfies the triangle inequality. Even more common, M is taken to be the d -dimensional vector space where dissimilarity is measured using the Euclidean distance, Manhattan distance or other distance metric. However, the dissimilarity function can be arbitrary. One example is asymmetric Bregman divergence, for which the triangle inequality does not hold.^[1]

Applications

The nearest neighbor search problem arises in numerous fields of application, including:

- Pattern recognition – in particular for optical character recognition
- Statistical classification – see k-nearest neighbor algorithm
- Computer vision – for point cloud registration^[2]
- Computational geometry – see Closest pair of points problem
- Cryptanalysis – for lattice problem^[3]
- Databases – e.g. content-based image retrieval
- Coding theory – see maximum likelihood decoding
- Semantic search
- Data compression – see MPEG-2 standard
- Robotic sensing^[4]
- Recommendation systems, e.g. see Collaborative filtering
- Internet marketing – see contextual advertising and behavioral targeting
- DNA sequencing
- Spell checking – suggesting correct spelling
- Plagiarism detection
- Similarity scores for predicting career paths of professional athletes.
- Cluster analysis – assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense, usually based on Euclidean distance

- [Chemical similarity](#)
- [Sampling-based motion planning](#)

Methods

Various solutions to the NNS problem have been proposed. The quality and usefulness of the algorithms are determined by the time complexity of queries as well as the space complexity of any search data structures that must be maintained. The informal observation usually referred to as the [curse of dimensionality](#) states that there is no general-purpose exact solution for NNS in high-dimensional Euclidean space using polynomial preprocessing and polylogarithmic search time.

Exact methods

Linear search

The simplest solution to the NNS problem is to compute the distance from the query point to every other point in the database, keeping track of the "best so far". This algorithm, sometimes referred to as the naive approach, has a [running time](#) of $O(dN)$, where N is the [cardinality](#) of S and d is the dimensionality of S . There are no search data structures to maintain, so the linear search has no space complexity beyond the storage of the database. Naive search can, on average, outperform space partitioning approaches on higher dimensional spaces.^[5]

The absolute distance is not required for distance comparison, only the relative distance. In geometric coordinate systems the distance calculation can be sped up considerably by omitting the square root calculation from the distance calculation between two coordinates. The distance comparison will still yield identical results.

Space partitioning

Since the 1970s, the [branch and bound](#) methodology has been applied to the problem. In the case of Euclidean space, this approach encompasses [spatial index](#) or spatial access methods. Several space-partitioning methods have been developed for solving the NNS problem. Perhaps the simplest is the [k-d tree](#), which iteratively bisects the search space into two regions containing half of the points of the parent region. Queries are performed via traversal of the tree from the root to a leaf by evaluating the query point at each split. Depending on the distance specified in the query, neighboring branches that might contain hits may also need to be evaluated. For constant dimension query time, average complexity is $O(\log N)$ ^[6] in the case of randomly distributed points, worst case complexity is $O(kN^{(1-1/k)})$ ^[7] Alternatively the [R-tree](#) data structure was designed to support nearest neighbor search in dynamic context, as it has efficient algorithms for insertions and deletions such as the [R* tree](#).^[8] R-trees can yield nearest neighbors not only for Euclidean distance, but can also be used with other distances.

In the case of general metric space, the branch-and-bound approach is known as the [metric tree](#) approach. Particular examples include [vp-tree](#) and [BK-tree](#) methods.

Using a set of points taken from a 3-dimensional space and put into a [BSP tree](#), and given a query point taken from the same space, a possible solution to the problem of finding the nearest

point-cloud point to the query point is given in the following description of an algorithm.

(Strictly speaking, no such point may exist, because it may not be unique. But in practice, usually we only care about finding any one of the subset of all point-cloud points that exist at the shortest distance to a given query point.) The idea is, for each branching of the tree, guess that the closest point in the cloud resides in the half-space containing the query point. This may not be the case, but it is a good heuristic. After having recursively gone through all the trouble of solving the problem for the guessed half-space, now compare the distance returned by this result with the shortest distance from the query point to the partitioning plane. This latter distance is that between the query point and the closest possible point that could exist in the half-space not searched. If this distance is greater than that returned in the earlier result, then clearly there is no need to search the other half-space. If there is such a need, then you must go through the trouble of solving the problem for the other half space, and then compare its result to the former result, and then return the proper result. The performance of this algorithm is nearer to logarithmic time than linear time when the query point is near the cloud, because as the distance between the query point and the closest point-cloud point nears zero, the algorithm needs only perform a look-up using the query point as a key to get the correct result.

Approximation methods

An approximate nearest neighbor search algorithm is allowed to return points whose distance from the query is at most c times the distance from the query to its nearest points. The appeal of this approach is that, in many cases, an approximate nearest neighbor is almost as good as the exact one. In particular, if the distance measure accurately captures the notion of user quality, then small differences in the distance should not matter.^[9]

Greedy search in proximity neighborhood graphs

Proximity graph methods (such as navigable small world graphs^[10] and HNSW^{[11][12]}) are considered the current state-of-the-art for the approximate nearest neighbors search.

The methods are based on greedy traversing in proximity neighborhood graphs $G(V, E)$ in which every point $x_i \in S$ is uniquely associated with vertex $v_i \in V$. The search for the nearest neighbors to a query q in the set S takes the form of searching for the vertex in the graph $G(V, E)$. The basic algorithm – greedy search – works as follows: search starts from an enter-point vertex $v_i \in V$ by computing the distances from the query q to each vertex of its neighborhood $\{v_j : (v_i, v_j) \in E\}$, and then finds a vertex with the minimal distance value. If the distance value between the query and the selected vertex is smaller than the one between the query and the current element, then the algorithm moves to the selected vertex, and it becomes new enter-point. The algorithm stops when it reaches a local minimum: a vertex whose neighborhood does not contain a vertex that is closer to the query than the vertex itself.

The idea of proximity neighborhood graphs was exploited in multiple publications, including the seminal paper by Arya and Mount,^[13] in the VoroNet system for the plane,^[14] in the RayNet system for the \mathbb{E}^n ,^[15] and in the Navigable Small World,^[10] Metrized Small World^[16] and HNSW^{[11][12]} algorithms for the general case of spaces with a distance function. These works were preceded by a pioneering paper by Toussaint, in which he introduced the concept of a *relative neighborhood graph*.^[17]

Locality sensitive hashing

Locality sensitive hashing (LSH) is a technique for grouping points in space into 'buckets' based on some distance metric operating on the points. Points that are close to each other under the chosen metric are mapped to the same bucket with high probability.^[18]

Nearest neighbor search in spaces with small intrinsic dimension

The cover tree has a theoretical bound that is based on the dataset's doubling constant. The bound on search time is $O(c^{12} \log n)$ where c is the expansion constant of the dataset.

Projected radial search

In the special case where the data is a dense 3D map of geometric points, the projection geometry of the sensing technique can be used to dramatically simplify the search problem. This approach requires that the 3D data is organized by a projection to a two-dimensional grid and assumes that the data is spatially smooth across neighboring grid cells with the exception of object boundaries. These assumptions are valid when dealing with 3D sensor data in applications such as surveying, robotics and stereo vision but may not hold for unorganized data in general. In practice this technique has an average search time of $O(1)$ or $O(K)$ for the k -nearest neighbor problem when applied to real world stereo vision data.^[4]

Vector approximation files

In high-dimensional spaces, tree indexing structures become useless because an increasing percentage of the nodes need to be examined anyway. To speed up linear search, a compressed version of the feature vectors stored in RAM is used to prefilter the datasets in a first run. The final candidates are determined in a second stage using the uncompressed data from the disk for distance calculation.^[19]

Compression/clustering based search

The VA-file approach is a special case of a compression based search, where each feature component is compressed uniformly and independently. The optimal compression technique in multidimensional spaces is Vector Quantization (VQ), implemented through clustering. The database is clustered and the most "promising" clusters are retrieved. Huge gains over VA-File, tree-based indexes and sequential scan have been observed.^{[20][21]} Also note the parallels between clustering and LSH.

Variants

There are numerous variants of the NNS problem and the two most well-known are the k -nearest neighbor search and the ϵ -approximate nearest neighbor search.

k -nearest neighbors

k -nearest neighbor search identifies the top k nearest neighbors to the query. This technique is commonly used in predictive analytics to estimate or classify a point based on the consensus of its neighbors. k -nearest neighbor graphs are graphs in which every point is connected to its k

nearest neighbors.

Approximate nearest neighbor

In some applications it may be acceptable to retrieve a "good guess" of the nearest neighbor. In those cases, we can use an algorithm which doesn't guarantee to return the actual nearest neighbor in every case, in return for improved speed or memory savings. Often such an algorithm will find the nearest neighbor in a majority of cases, but this depends strongly on the dataset being queried.

Algorithms that support the approximate nearest neighbor search include locality-sensitive hashing, best bin first and balanced box-decomposition tree based search.^[22]

Nearest neighbor distance ratio

Nearest neighbor distance ratio does not apply the threshold on the direct distance from the original point to the challenger neighbor but on a ratio of it depending on the distance to the previous neighbor. It is used in CBIR to retrieve pictures through a "query by example" using the similarity between local features. More generally it is involved in several matching problems.

Fixed-radius near neighbors

Fixed-radius near neighbors is the problem where one wants to efficiently find all points given in Euclidean space within a given fixed distance from a specified point. The distance is assumed to be fixed, but the query point is arbitrary.

All nearest neighbors

For some applications (e.g. entropy estimation), we may have N data-points and wish to know which is the nearest neighbor *for every one of those N points*. This could, of course, be achieved by running a nearest-neighbor search once for every point, but an improved strategy would be an algorithm that exploits the information redundancy between these N queries to produce a more efficient search. As a simple example: when we find the distance from point X to point Y , that also tells us the distance from point Y to point X , so the same calculation can be reused in two different queries.

Given a fixed dimension, a semi-definite positive norm (thereby including every L^p norm), and n points in this space, the nearest neighbour of every point can be found in $O(n \log n)$ time and the m nearest neighbours of every point can be found in $O(mn \log n)$ time.^{[23][24]}

See also

- | | |
|-----------------------------------------|--------------------------------------|
| ▪ <u>Ball tree</u> | ▪ <u>Curse of dimensionality</u> |
| ▪ <u>Closest pair of points problem</u> | ▪ <u>Digital signal processing</u> |
| ▪ <u>Cluster analysis</u> | ▪ <u>Dimension reduction</u> |
| ▪ <u>Content-based image retrieval</u> | ▪ <u>Fixed-radius near neighbors</u> |

- [Fourier analysis](#)
- [Instance-based learning](#)
- [k-nearest neighbor algorithm](#)
- [Linear least squares](#)
- [Locality sensitive hashing](#)
- [Maximum inner-product search](#)
- [MinHash](#)
- [Multidimensional analysis](#)
- [Nearest-neighbor interpolation](#)
- [Neighbor joining](#)
- [Principal component analysis](#)
- [Range search](#)
- [Similarity learning](#)
- [Singular value decomposition](#)
- [Sparse distributed memory](#)
- [Statistical distance](#)
- [Time series](#)
- [Voronoi diagram](#)
- [Wavelet](#)

References

Citations

1. Cayton, Lawrence (2008). "Fast nearest neighbor retrieval for bregman divergences". *Proceedings of the 25th International Conference on Machine Learning*. pp. 112–119. doi:10.1145/1390156.1390171 (<https://doi.org/10.1145/1390156.1390171>). ISBN 9781605582054. S2CID 12169321 (<https://api.semanticscholar.org/CorpusID:12169321>).
2. Qiu, Deyuan, Stefan May, and Andreas Nüchter. "GPU-accelerated nearest neighbor search for 3D registration." (<https://core.ac.uk/download/pdf/22872975.pdf>) International conference on computer vision systems. Springer, Berlin, Heidelberg, 2009.
3. Becker, Ducas, Gama, and Laarhoven. "New directions in nearest neighbor searching with applications to lattice sieving." (<https://eprint.iacr.org/2015/1128.pdf>) Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (pp. 10-24). Society for Industrial and Applied Mathematics.
4. Bewley, A.; Upcroft, B. (2013). *Advantages of Exploiting Projection Structure for Segmenting Dense 3D Point Clouds* (<http://www.araa.asn.au/acra/acra2013/papers/pap148s1-file1.pdf>) (PDF). Australian Conference on Robotics and Automation.
5. Weber, Roger; Schek, Hans-J.; Blott, Stephen (1998). "A quantitative analysis and performance study for similarity search methods in high dimensional spaces" (<http://www.vldb.org/conf/1998/p194.pdf>) (PDF). *VLDB '98 Proceedings of the 24rd International Conference on Very Large Data Bases*. pp. 194–205.
6. Andrew Moore. "An introductory tutorial on KD trees" (<https://web.archive.org/web/20160303203122/http://www.autonlab.com/autonweb/14665/version/2/part/5/data/moore-tutorial.pdf?branch=main&language=en>) (PDF). Archived from the original (<http://www.autonlab.com/autonweb/14665/version/2/part/5/data/moore-tutorial.pdf?branch=main&language=en>) (PDF) on 2016-03-03. Retrieved 2008-10-03.
7. Lee, D. T.; Wong, C. K. (1977). "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees". *Acta Informatica*. **9** (1): 23–29. doi:10.1007/BF00263763 (<https://doi.org/10.1007/BF00263763>). S2CID 36580055 (<https://api.semanticscholar.org/CorpusID:36580055>).

8. Roussopoulos, N.; Kelley, S.; Vincent, F. D. R. (1995). "Nearest neighbor queries". *Proceedings of the 1995 ACM SIGMOD international conference on Management of data - SIGMOD '95*. p. 71. doi:10.1145/223784.223794 (<https://doi.org/10.1145%2F223784.223794>). ISBN 0897917316.
9. Andoni, A.; Indyk, P. (2006-10-01). "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions". *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. pp. 459–468. CiteSeerX 10.1.1.142.3471 (<https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.3471>). doi:10.1109/FOCS.2006.49 (<https://doi.org/10.1109%2FFOCS.2006.49>). ISBN 978-0-7695-2720-8.
10. Malkov, Yury; Ponomarenko, Alexander; Logvinov, Andrey; Krylov, Vladimir (2012), Navarro, Gonzalo; Pestov, Vladimir (eds.), "Scalable Distributed Algorithm for Approximate Nearest Neighbor Search Problem in High Dimensional General Metric Spaces" (http://link.springer.com/10.1007/978-3-642-32153-5_10), *Similarity Search and Applications*, vol. 7404, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 132–147, doi:10.1007/978-3-642-32153-5_10 (https://doi.org/10.1007%2F978-3-642-32153-5_10), ISBN 978-3-642-32152-8, retrieved 2024-01-16
11. Malkov, Yury; Yashunin, Dmitry (2016). "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs". arXiv:1603.09320 (<https://arxiv.org/abs/1603.09320>) [cs.DS (<https://arxiv.org/archive/cs>)].
12. Malkov, Yu A.; Yashunin, D. A. (2020-04-01). "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **42** (4): 824–836. arXiv:1603.09320 (<https://arxiv.org/abs/1603.09320>). Bibcode:2020ITPAM..42..824M (<https://ui.adsabs.harvard.edu/abs/2020ITPAM..42..824M>). doi:10.1109/TPAMI.2018.2889473 (<https://doi.org/10.1109%2FTPAMI.2018.2889473>). ISSN 0162-8828 (<https://search.worldcat.org/issn/0162-8828>). PMID 30602420 (<https://pubmed.ncbi.nlm.nih.gov/30602420>).
13. Arya, Sunil; Mount, David (1993). "Approximate Nearest Neighbor Queries in Fixed Dimensions". *Proceedings of the Fourth Annual {ACM/SIGACT-SIAM} Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas.*: 271–280.
14. Olivier, Beaumont; Kermarrec, Anne-Marie; Marchal, Loris; Rivière, Etienne (2006). "Voro Net: A scalable object network based on Voronoi tessellations" (<https://hal.inria.fr/inria-00071210/PDF/RR-5833.pdf>) (PDF). *2007 IEEE International Parallel and Distributed Processing Symposium*. Vol. RR-5833. pp. 23–29. doi:10.1109/IPDPS.2007.370210 (<https://doi.org/10.1109%2FIPDPS.2007.370210>). ISBN 1-4244-0909-8. S2CID 8844431 (<https://api.semanticscholar.org/CorpusID:8844431>).
15. Olivier, Beaumont; Kermarrec, Anne-Marie; Rivière, Etienne (2007). "Peer to Peer Multidimensional Overlays: Approximating Complex Structures". *Principles of Distributed Systems*. Lecture Notes in Computer Science. Vol. 4878. pp. 315–328. CiteSeerX 10.1.1.626.2980 (<https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.626.2980>). doi:10.1007/978-3-540-77096-1_23 (https://doi.org/10.1007%2F978-3-540-77096-1_23). ISBN 978-3-540-77095-4.

16. Malkov, Yury; Ponomarenko, Alexander; Krylov, Vladimir; Logvinov, Andrey (2014). "Approximate nearest neighbor algorithm based on navigable small world graphs". *Information Systems*. **45**: 61–68. doi:10.1016/j.is.2013.10.006 (<https://doi.org/10.1016%2Fj.is.2013.10.006>). S2CID 9896397 (<https://api.semanticscholar.org/CorpusID:9896397>).
17. Toussaint, Godfried (1980). "The relative neighbourhood graph of a finite planar set". *Pattern Recognition*. **12** (4): 261–268. Bibcode:1980PatRe..12..261T (<http://ui.adsabs.harvard.edu/abs/1980PatRe..12..261T>). doi:10.1016/0031-3203(80)90066-7 (<https://doi.org/10.1016%2F0031-3203%2880%2990066-7>).
18. A. Rajaraman & J. Ullman (2010). "Mining of Massive Datasets, Ch. 3" (<http://infolab.stanford.edu/~ullman/mmds.html>).
19. Weber, Roger; Blott, Stephen. "An Approximation-Based Data Structure for Similarity Search" (<https://web.archive.org/web/20170304043243/https://pdfs.semanticscholar.org/83e4/e3281411ffef40654a4b5d29dae48130aefb.pdf>) (PDF). S2CID 14613657 (<https://api.semanticscholar.org/CorpusID:14613657>). Archived from the original (<https://pdfs.semanticscholar.org/83e4/e3281411ffef40654a4b5d29dae48130aefb.pdf>) (PDF) on 2017-03-04. {{cite journal}}: Cite journal requires |journal= (help)
20. Ramaswamy, Sharadh; Rose, Kenneth (2007). "Adaptive cluster-distance bounding for similarity search in image databases". *ICIP*.
21. Ramaswamy, Sharadh; Rose, Kenneth (2010). "Adaptive cluster-distance bounding for high-dimensional indexing". *TKDE*.
22. Arya, S.; Mount, D. M.; Netanyahu, N. S.; Silverman, R.; Wu, A. (1998). "An optimal algorithm for approximate nearest neighbor searching" (<https://web.archive.org/web/20160303232202/http://www.cse.ust.hk/faculty/arya/pub/JACM.pdf>) (PDF). *Journal of the ACM*. **45** (6): 891–923. CiteSeerX 10.1.1.15.3125 (<https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3125>). doi:10.1145/293347.293348 (<https://doi.org/10.1145%2F293347.293348>). S2CID 8193729 (<https://api.semanticscholar.org/CorpusID:8193729>). Archived from the original (<http://www.cse.ust.hk/faculty/arya/pub/JACM.pdf>) (PDF) on 2016-03-03. Retrieved 2009-05-29.
23. Clarkson, Kenneth L. (1983), "Fast algorithms for the all nearest neighbors problem", *24th IEEE Symp. Foundations of Computer Science, (FOCS '83)*, pp. 226–232, doi:10.1109/SFCS.1983.16 (<https://doi.org/10.1109%2FSFCS.1983.16>), ISBN 978-0-8186-0508-6, S2CID 16665268 (<https://api.semanticscholar.org/CorpusID:16665268>).
24. Vaidya, P. M. (1989). "An $O(n \log n)$ Algorithm for the All-Nearest-Neighbors Problem" (<https://doi.org/10.1007%2FBF02187718>). *Discrete and Computational Geometry*. **4** (1): 101–115. doi:10.1007/BF02187718 (<https://doi.org/10.1007%2FBF02187718>).

Sources

- Andrews, L. (November 2001). "A template for the nearest neighbor problem" (<http://www.ddj.com/architect/184401449>). *C/C++ Users Journal*. **19** (11): 40–49. ISSN 1075-2838 (<https://search.worldcat.org/issn/1075-2838>).
- Arya, S.; Mount, D.M.; Netanyahu, N. S.; Silverman, R.; Wu, A. Y. (1998). "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions". *Journal of the ACM*. **45** (6): 891–923. CiteSeerX 10.1.1.15.3125 (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3125>)

[ps://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3125](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.3125)).
[doi:10.1145/293347.293348](https://doi.org/10.1145/293347.293348) (<https://doi.org/10.1145/293347.293348>).
[S2CID 8193729](https://api.semanticscholar.org/CorpusID:8193729) (<https://api.semanticscholar.org/CorpusID:8193729>).

- Beyer, K.; Goldstein, J.; Ramakrishnan, R.; Shaft, U. (1999). "When is nearest neighbor meaningful?". *Proceedings of the 7th ICDT*.
- Chen, Chung-Min; Ling, Yibei (2002). "A Sampling-Based Estimator for Top-k Query". *ICDE*: 617–627.
- Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann. ISBN 978-0-12-369446-1.
- Zezula, P.; Amato, G.; Dohnal, V.; Batko, M. (2006). *Similarity Search – The Metric Space Approach*. Springer. ISBN 978-0-387-29146-8.

Further reading

- Shasha, Dennis (2004). *High Performance Discovery in Time Series*. Berlin: Springer. ISBN 978-0-387-00857-8.

External links

- [Nearest Neighbors and Similarity Search \(http://simsearch.yury.name/tutorial.html\)](http://simsearch.yury.name/tutorial.html) – a website dedicated to educational materials, software, literature, researchers, open problems and events related to NN searching. Maintained by Yuri Lifshits
 - [Similarity Search Wiki \(https://archive.today/20130222061350/http://sswiki.tierra-aoi.net/\)](https://archive.today/20130222061350/http://sswiki.tierra-aoi.net/) – a collection of links, people, ideas, keywords, papers, slides, code and data sets on nearest neighbours
-

Retrieved from "https://en.wikipedia.org/w/index.php?title=Nearest_neighbor_search&oldid=1320333629"