OR4NN: Preprocess for Scalable K Nearest Neighbour Join

Sheng Zhou[*1] and Jonathan Simmons [†1]

[1]GDS Data Science and Geospatial Insights, Ordnance Survey

GISRUK 2026

Summary

OR4NN is a conceptual framework facilitating efficient scalable KNN join on metric data. Query space is divided into partitions. A candidate set of K objects from the query object set is allocated to each partition. An "object range" is computed based on partition extent and the candidate object set. For any query inside the partition, its KNN will be found from the set of objects intersecting the object range. Practical method for constructing object range for data in 2D Euclidean Space on Cartesian Distance is presented as an example, which yields favorable performance in real world applications.

KEYWORDS: K-Nearest Neighbours, Parallel Processing, Machine Learning, Databricks, Apache Spark

1. Introduction

A KNN (K-Nearest Neighbors) query finds the K objects in an object set closest to a given query location. KNN join finds the K nearest neighbours for each query in a query set (Harvey F and Tulloch D, 2006). KNN query/join enables KNN classification, one of the core machine learning algorithms.

For large scale KNN-Join in a parallel computing environment where the object set is too large for a single computing unit and has to be partitioned, the KNN of a query may spread over multiple partitions. In a worst-case scenario, all partitions have to be examined in order to find the exact KNN results for a query.

In this research we introduce OR4NN, a pre-processing strategy for parallel KNN join on metric datasets. Our method will first divide the query space into several subset regions without gap. Given a pre-defined K, for each region an "object range" which is also as a subset region of the query space is computed. Subsequently, the *k*NN for any queries (for a *k* up to K) in a region will be found from the set of objects intersecting the object range for this region independent of the remainder of the object set, which conforms to the modern shared-nothing architecture perfectly. Additionally, object range information may be stored and re-used for subsequent multiple KNN joins on the same object set.

This method is originated from a previous work on large KNN join and continuous moving object KNN queries in a spatial DBMS environment(Zhou S and Simmons J, 2020).

2. Object Range for Nearest Neighbours (OR4NN)

Given a metric space (X, d) where X is a non-empty set, $d:X\times X\rightarrow R$ is a metric (a.k.a. distance function) defined on X, we define the following notations:

- Dist(q,o)(q,o∈ X): the original distance metric between two points *q* and *o* in X
- **Object** $O=\{o_i|i=1,2,\ldots\}\subset X$ and **Query** $Q=\{q_i|i=1,2,\ldots\}\subset X$
- **Object Set** $O^P=\{O_i|i=1,n\}$ and **Query Set** $Q^P=\{Q_i|i=1,m\}$

[*] Sheng.Zhou@os.uk
[†] Jonathan.Simmons@os.uk

- MinDist(q,O)(q∈X,O⊂X)=inf{Dist(q,o):o∈O}
- MinDist(Q,O)(Q,O⊂X)=inf{Dist(q,o):q∈Q,o∈O}=inf{MinDist(q,O):q∈Q}
- MinDist(Q,O^P)=inf{MinDist(Q,O):O∈O^P}
- MinDist(Q^P,O^P)=inf{MinDist(Q,O^P):Q∈Q^P}
- MaxDist(q,O)(q∈X,O⊂X)=sup{Dist(q,o):o∈O}
- MaxDist(Q,O)=sup{Dist(q,o):q∈Q,o∈O}=sup{MaxDist(q,O):q∈Q}
- MaxDist(Q,O^P)=sup{MaxDist(Q,O):O∈O^P}
- MaxDist(Q^P,O^P)=sup{MaxDist(Q,O^P):Q∈Q^P}
- MaxMinDist(Q,O)=sup{MinDist(q,O):q∈Q}: directional semi-Harsdorff distance
- MaxMinDist(Q,O^P)=sup{MaxMinDist(Q,O):O∈ O^P}
- MaxMinDist(q,O^P)=sup{MinDist(q,O):O∈O^P}
- KNN(p,O^P): the *K* nearest neighbors of *p* w.r.t. O^P

The KNN of a multi-point query is in the union of the KNN of its constituent points:

$$KNN(\{p,q\},O^P)\subseteq KNN(p,O^P)\cup KNN(q,O^P) \quad (1)$$

$$KNN(Q=\{p_i|i=1,n\},O^P)\subseteq \cup(KNN(p_i,O^P)|i=1,n) \quad (2)$$

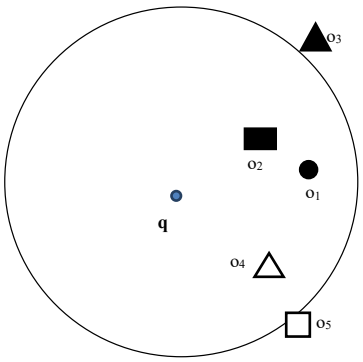

Figure 1 Object range for a point query {q} w.r.t. $O^K=\{o_1,o_2,o_3\}$

2.1. Object Range for Point Queries

Given O^P with *n* objects, for 1≤K≤ *n*, a point query $Q_q=\{q\}$, and a set of objects $O^K=\{O_i|i=1,K\}\subseteq O^P$ as the **candidate object set** ($\{o_1,o_2,o_3\}$ in figure 1, K=3), we may define a set on X as the **object range** for Q_q at *k*≤K w.r.t. O^K:

$$OR(Q_q=\{q\},O^K)=\{p|Dist(p,q)\leq MaxMinDist(q,O^K)\} \quad (3)$$

Subsequently, we have:

$$KNN(Q_q=\{q\}, O^P) \subseteq OR(Q_q, O^K) \quad (4)$$

That is, although there may be other objects from O^P (e.g. o_4, o_5 in figure 1) closer to the query than some or all objects in O^K, O^K guarantees there are at least K objects inside $OR(Q_q, O^K)$ so that the KNN query will always succeed. Note that this is the minimum object range of the query point which takes the form of a disc. Any set on X that contains $OR(Q_q, O^K)$ is also an object range of it.

And for multi-point query $Q_m=\{q_i|i=1,2,\ldots\}$:

$$KNN(Q_m,O^P)\subseteq\cup(OR(\{q_i\},O^K)|i=1,2,\ldots) \quad (5)$$

$$OR(Q_m,O^K)=\cup(OR(\{q_i\},O^K)|i=1,2,\ldots) \quad (6)$$

For a subset $Q_n\subseteq Q_m$ we have:

$$OR(Q_n,O^K)\subseteq OR(Q_m,O^K) \quad (7)$$

$$KNN(Q_n,O^P)\subseteq QR(Q_n,O^K)\subseteq OR(Q_m,O^K) \quad (8)$$

2.2. Object Range as a Facility for Scalable KNN

If we view Q_m as a region in X and $Q_n\subseteq Q_m$ a query in Q_m, the KNN of Q_n may be found in $OR(Q_m,O^K)$. That is, the object range for Q_m is computed as the **union** of object range of all points in Q_m, and the KNN of any query Q_n defined in Q_m are contained in the object range of Q_m.

This observation leads to a generic conceptual framework for handling large-scale KNN join:

1) Divide the query space into a set of partitions $R^P=\{R_i|i=1,n\}$
2) Given a pre-defined K, for each partition R_i, select and assign a candidate object set $O^K_i\subseteq O^P$
3) Compute $OR(\{r_j\}|r_j\in R_i,O^K_i)$ for all points $r_i\in R_i$ and union the resulting object ranges to create $OR(R_i,O^K_i)= \cup(OR(\{r_j\}|r_j\in R_i,O^K_i))$.
4) At query time, for each partition R_i, extract $Q^R_i=\{Q|Q\in Q^P,Q\cap R_i\neq\emptyset\}$ from query set Q^P as queries intersecting the query partition R_i, and extract $O^R_i=\{O|O\in O^P,O\cap OR(R_i,O^K_i)\neq\emptyset\}$ from the object set O^P as objects intersecting the query partition's object range $OR(R_i,O^K_i)$.
5) For each pair of (Q^R_i,O^R_i), $KNN(Q|Q\in Q^R_i,O^P)$ may be found from O^R_i using any preferred local KNN algorithms. In other words, each pair of (Q^R_i, O^R_i) may be joined locally and independently, no multi-pass or cross-communication is required.

Joining results from all pairs of (Q^R_i, O^R_i) are returned and post-processed to produce the final output.

3. Object Range in 2D Euclidean Space on Cartesian Distance

Section 2 defines object range on generic metric data. It is normally not feasible to create object range directly from the definition. For specific data types such as 2D geometric data or network data, more efficient methods are available to create object ranges.

In order to handle multi-point objects/queries (e.g. multi-point, linestring or polygon), we replace MaxMinDist in (3) with MaxDist to define the **extended object range** of $Q_q=\{q\}$ w.r.t. O^K as:

$$OR^E(Q_q,O^K)=\{p|Dist(p,q)\leq MaxDist(q, O^K)\} \quad (9)$$

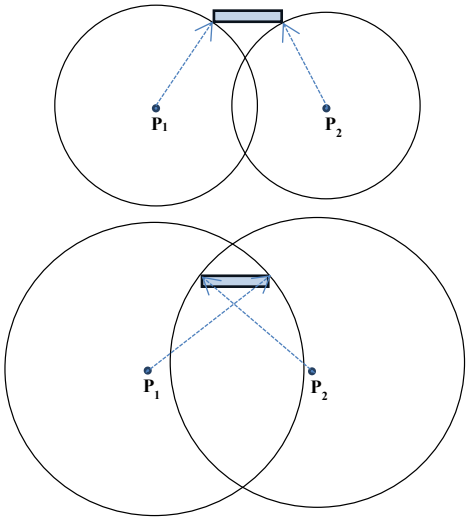

Figure 2 Original (top) and extended (bottom) object ranges for two query points p_1 and p_2

A property of O^K may be derived from (9): the candidate object set O^K is contained in the intersection of the extended object ranges of all query points $Q_m=\{p_i|i=1,m\}$.

$$O^K \subseteq \cap(OR^E(\{p_i\},O^K)) \tag{10}$$

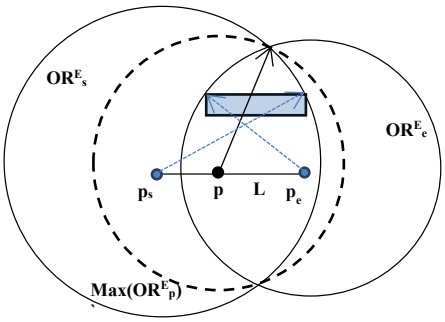

Figure 3 Extended object range for an interior point p on a line segment

Given a line segment $L(p_s,p_e)=\{p|Dist(p,p_s)+Dist(p,p_e)=Dist(p_s,p_e):p\in X\}$ with endpoints p_s and p_e, it can be proven that the KNN of an interior point *p* of L is in the union of the object ranges of the two end points:

$$OR^E(p\in L,O^K) \subseteq U(OR^E_s,OR^E_e) \tag{11}$$

$$KNN(p\in L,O^P) \subseteq U(OR^E_s,OR^E_e) \tag{12}$$

Similarly, given three non-collinear points p_1, p_2, and p_3 (i.e. $Dist(p_1,p_2)+Dist(p_2,p_3)>Dist(p_1,p_3)$), and a point *p* inside triangle $T(p_1, p_2, p_3)$, we may prove:

$$OR^E(p\in T,O^K) \subseteq U(OR^E_1,OR^E_2,OR^E_3) \tag{13}$$

$$KNN(p\in T,O^P) \subseteq U(OR^E_1,OR^E_2,OR^E_3) \tag{14}$$

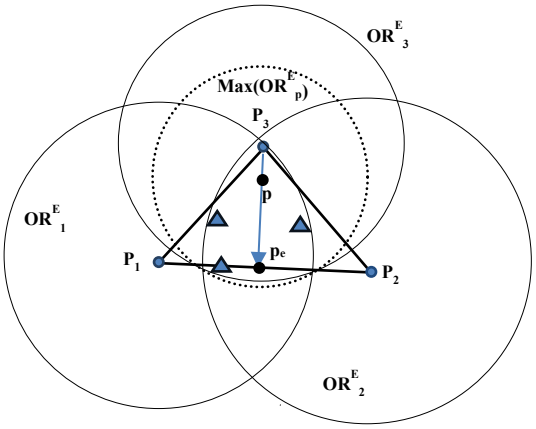

Figure 4 Extended object range for an interior point p inside a triangle

Furthermore, given a polygon PLG(p_1,p_2, …,p_n) which may be triangulated into a collection of multiple triangles, the object range of a polygon is in the union of the object range of all its **boundary vertices** (Figure 5). The KNN query as a point set in the polygon can be found from the set of objects intersecting the object range of the polygon.

$$OR^E(p \in PLG, O^K) \subseteq U(OR^E_i \mid i = 1, n) \tag{15}$$

$$KNN(p \in PLG, O^P) \subseteq U(OR^E_i \mid i = 1, n) \tag{16}$$

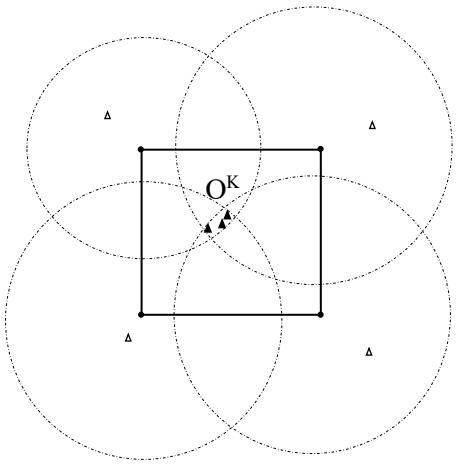

Figure 5 Extended object range for a square partition w.r.t. O^K(K=3). Hollow triangles are other objects inside the OR^E.

4. Discussion

For a 2D Euclidean query space, partitions may be created in multiple ways, e.g. a regular grid or Voronoi diagram from sampled objects.

The minimum object range of a vertex on the boundary of a partition is a disc/circle. For practical implementation, the circumscribed polygon (e.g. bounding box or octagon) of the circle may be used instead to simplify the representation.

A smaller object range for a partition will retrieve fewer in-range objects and is more efficient for subsequent KNN queries on this partition. Object range size depends on the assigned candidate object set O^K. It may be possible to find an optimal solution but in principle it should be sufficient to select the K objects closest to the centre of the partition as the candidate object set.

Both the object and query sets may contain arbitrary combination of points, lines and polygons. For a query intersecting two or more partitions, the query may be performed on each partition and all KNN results will be collected and re-ranked to produce the final KNN for the query.

The introduction of extended object range also enables the use of other distance metrics (MaxDist or MaxMinDist) in KNN query, in addition to the conventional MinDist.

A prototype of OR4NN for 2D geometric data has been implemented on Databricks/Apache Spark platform using Java/Scala. Applications of the prototype on multiple national coverage datasets yield superior performance over several commercial packages which may fail to complete, sometime return approximate results only, or is unable to handle linear/areal queries/objects.

The prototype and a detailed technical report containing the proofs for the key claims (11) and (13) and the algorithm for KNN on network data are available at:

https://github.com/OrdnanceSurvey/OR4NN

Acknowledgements

A patent application based on this research has been filed (UK Patent Office No. 2411135.3). The authors would like to thank Dr. Chris Jochem and Dr. Kate New of Ordnance Survey GDS Data Science and Geospatial Insights team for experimenting and evaluating the OR4NN prototype on real world datasets.

References

Bohm C and Krebs F (2004). The k-Nearest Neighbour Join: Turbo Charging the KDD Process. *Knowledge and Information Systems*, 6(6), 728–749.

Zhou S and Simmons J (2020). Data Pre-process Facilitating Efficient K-NN Queries in Spatial Database. Proceedings of Geographical Information Science Research UK (GISRUK) 2020.

Biographies

Sheng Zhou is a Senior Data Scientist at Ordnance Survey. He obtained his PhD in GIS/Urban Planning from Cardiff University and participated in research projects on multiscale spatial database and multi-agent system for active maps at Glamorgan and Cardiff universities. His research interests include machine learning algorithms and their applications on geospatial data, computational geometry, spatial databases and map generalisation.

Jonathan Simmons is the Head of Data Science and Management at Ordnance Survey.