ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Issue 9, September 2018

Efficient Query Processing in Reverse Nearest Neighbor Search

MS.Poonam Jain , MCA, Asst Proffessor,
Thakur college of science and commerce,Mumbai
DR.SK Singh , Department Head(Information Technology),
Thakur college of science and commerce,Mumbai.

Abstract: Reverse nearest neighbor (RNN) queries are useful in identifying objects that are of significant influence or importance. Existing methods reply on pre-computation of nearest neighbor distances, do not scale well with high dimensionality, or do not produce exact solutions. In this work we motivate and investigate the problem of reverse nearest neighbor search on high dimensional, multimedia data. We propose several heuristic algorithms to approximate the optimal solution and reduce the computation complexity. We propose exact and approximate algorithms that do not require pre-computation of nearest neighbor distances, and can potentially prune off most of the search space. We demonstrate the utility of reverse nearest neighbor search by showing how it can help improve the classification accuracy.

Keywords: Nearest Neighbor, Multidimensional, data heuristic algorithm, Multiple Boolean Ranges.

I. INTRODUCTION

The nearest neighbor (NN) search has long been accepted as one of the classic data mining methods, and its role in classification and similarity search is well documented. Given a query object, nearest neighbor search returns the object in the database that is the most similar to the query object. Similarly, the *k* nearest neighbor search or the *k*-NN search returnsthe *k* most similar objects to the query. NN and *k*-NNsearch problems have applications in many disciplines:information retrieval (find the most similar website to the query website), GIS (find the closest hospital to a certain location), etc. Contrary to nearest neighbor search, less considered is the related but much more computationally complicated problem of reverse nearest neighbor (RNN) search. Our heuristic algorithms include Successive Selection, Greedy and Quasi-Greedy.

Given a query object, reverse nearest neighbor search finds all objects in the database whose nearest neighbors are the query object. Note that since the relation of

NN is not symmetric, the NN of a query object might differ from its RNN(s). Object *B* being the nearest neighbor of object *A*does not automatically make it the reverse nearest neighbor of *A*, since *A* is not the nearest neighbor of *B*. The problem of reverse nearest neighbor search has many practical applications that can be applied to areas such as business impact analysis and customer profile analysis.

II. KNN FOR CLASSIFICATION

Let's see how to use KNN for classification. In this case, we are given some data points for training and also a new unlabelled data for testing. Our aim is to find the class label for the new point. The algorithm has different behavior based on k.

Case 1: k = 1 or Nearest Neighbor Rule

This is the simplest scenario. Let x be the point to be labeled. Find the point closest to x . Let it be y. Now nearest neighbor rule asks to assign the label of y to x. This seems too simplistic and sometimes even counter intuitive. If you feel that this procedure will result a huge error, you are right – but there is a catch. This reasoning holds only when the number of data points is not very large.

If the number of data points is very large, then there is a very high chance that label of x and y are same. An example might help – Let's say you have a (potentially) biased coin. You toss it for 1 million time and you have got head 900,000 times. Then most likely your next call will be head. We can use a similar argument here.

Let me try an informal argument here - Assume all points are in a D dimensional plane. The number of points is reasonably large. This means that the density of the plane at any point is fairly high. In other words, within any subspace there is adequate number of points. Consider a point x in the subspace which also has a lot of neighbors. Now let y be the

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Issue 9, September 2018

nearest neighbor. If x and y are sufficiently close, then we can assume that probability that x and y belong to same class is fairly same – Then by decision theory, x and y have the same class.

The book "Pattern Classification" by Duda and Hart has an excellent discussion about this Nearest Neighbor rule. One of their striking results is to obtain a fairly tight error bound to the Nearest Neighbor rule. The bound is

$$P^* \leq P \leq P^*(2 - \frac{c}{c-1}P^*)$$

Where P^* is the Bayes error rate, c is the number of classes and P is the error rate of Nearest Neighbor. The result is indeed very striking (atleast to me) because it says that if the number of points is fairly large then the error rate of Nearest Neighbor is less than twice the Bayes error rate. Pretty cool for a simple algorithm like KNN. Do read the book for all the juicy details.

Case 2: k = K or k-Nearest Neighbor Rule

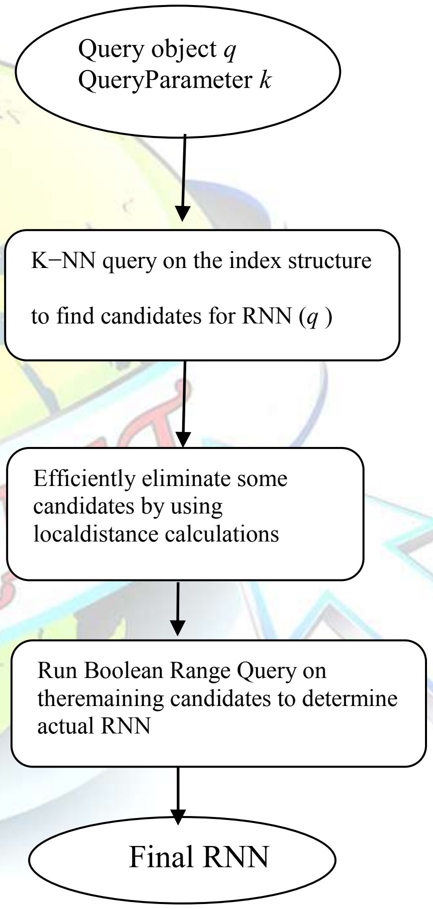
This is a straightforward extension of 1NN. Basically what we do is that we try to find the k nearest neighbor and do a majority voting. Typically k is odd when the number of classes is 2. Lets say k = 5 and there are 3 instances of C1 and 2 instances of C2. In this case, KNN says that new point has to labeled as C1 as it forms the majority. We follow a similar argument when there are multiple classes. One of the straight forward extensions is not to give 1 vote to all the neighbors. A very common thing to do is weighted kNN where each point has a weight which is typically calculated using its distance. For eg under inverse distance weighting, each point has a weight equal to the inverse of its distance to the point to be classified. This means that neighboring points have a higher vote than the farther points. It is quite obvious that the accuracy might increase when you increase k but the computation cost also increases.

III. RNN ALGORITHM

Our general approach for answering RNN queries. The algorithm proceeds in two steps: In the First step, a k-NN search is performed returning the k closest data objects to the query and in the next step these k objects are efficiently analyzed to answer the query. The value of k that should be used in the first step depends on the data set. Based on the values of average recall we can choose the value of k we want to start with. The following subsections explain the second step in detail with necessary optimizations that can be used.

Filtering step I

After the first step of the RNN algorithm we have k-NN of a query point. However, some of them can be ruled out from further elaboration by local distance calculations performed within the k objects, i.e., the candidate set. The principle of the elimination is based on the fact that a point that does not have the query point as the nearest neighbor among the candidate set can never be the RNN of the query point in the whole data set. Therefore a point that has another candidate as its closest in those k-NN points is eliminated from the search (note that only k objects are involved in this test). We refer to this step as filtering step I. Our experiments show that this step can efficiently filter out a significant number of candidates.


Query object q
QueryParameter k
K−NN query on the index structure
to find candidates for RNN (q)
Efficiently eliminate some candidates by using localdistance calculations
Run Boolean Range Query on theremaining candidates to determine actual RNN
Final RNN

Proposed Algorithm for RNN

After filtering step I, we have a set of candidate data points, each of which has a query point as its local NN (considering only the data points in the candidate set). For a point to be a RNN, we need to verify that the query point is the actual NN considering the whole data set. Two different approaches can be followed here. One crude approach to solve the above problem is to and the global NN of each point and check whether it is the query point. If this is the case, then that point is a RNN otherwise it is not an RNN. Another approach, which we refer to as filtering step II isrunning a new kind of query called the Boolean range query, which is covered in the following subsection.

Boolean Range Queries (Filtering step II)

A Boolean range query BRQ (q; r) returns true if the set ft 2 Sjd (q; t) <rg is not empty and returnsfalse otherwise. Boolean range query is a special case of range query whichwill return either true or false depending on whether thereis any point inside the given range or not. Boolean RangeQueries can be naturally handled more efficiently than rangequeries and this advantage is exploited in the proposed schemes. For each candidate point, we define a circular range withthe candidate point as the center and the distance to thequery point as the radius. If this Boolean range query returns false then the point is RNN, otherwise point is not a RNN.

The pseudo code forBoolean Range Query is given in Figure.

```
Procedure BooleanRangeQuery (Noden; pointq; radius)
BEGIN
Input:
Node n to start the search
Point q is query point
radius is equal to distance between
query point and candidate set point
Output:
True =) At least one point inside search region
False =) No point inside the search region
If n is a leaf node do:
check if there exists a point p such that
dist (p; q) <= q then return TRUE;
If n is an internal node, then for each branch do:
If any node (MBR) intersects such that at least
one edge of MBR is inside search region, i.e.,
minmaxdist<=radius then return TRUE;
else
Sort the intersecting MBRs wrt. some criterion
such as mindist and traverse the MBRs wrt. it.
If any of the MBRs has a point then return TRUE;
else return FALSE;
END
```

Cases for Boolean range query intersecting with a region

Boolean Range Query versus Range Query

In a range query, typically multiple MBRs intersect the query region. Especially when the number of dimensions ishigh, the number of MBRs intersecting the query region is also high. For range queries we have to access all the MBRs that intersect with the search region. The main strength of Boolean range query over traditional range query is that even multiple MBRs intersect a search region we do not need to access all of the MBRs. AboveFigure shows five possible cases where a circular search region can overlap partially or fully for range queries. Searchregion is denoted by circle and MBRs are denoted by A, B, C, D and E.

1. MBR A: fully contained in search region. This means that search region has at least one point.
2. MBR B: only one vertex of B is in search region.
3. MBR C: two of C's vertices are in search region. This means that an entire edge of MBR is in search region,which guarantees that at least one data point is in search region.
4. MBR D: no vertices of D are in search region.
5. MBR E: Search region is fully contained in the MBR E.

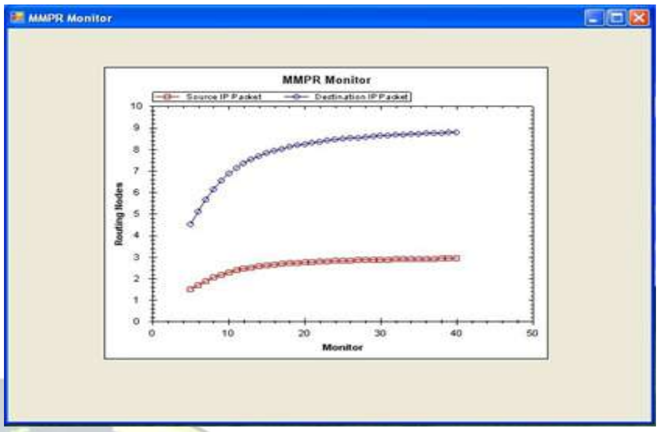
For a Boolean range query, if we can guarantee that atleast one edge of MBR is inside the search region then wedon't need any page access (case A and C in above Figure). Thiscan be easily checked by calculating the minmaxdist, which is defined as minimum of the maximum possible distancesbetween a point (here candidate point or center of searchregion) to the face of intersecting MBR. If minmaxdistis less than or equal to the radius of the search region thanthere is at least a point in the intersecting MBR so we donot need any page access. For example, we found that forisolet data set, 35% of intersecting MBRs have this property.For other cases B, D, and E we cannot say whether thereis a point contained both in search region and MBR. So,traversal of the corresponding branch may be necessary toanswer the query if no relevant point is found previously.When a range query intersects a number of MBRs, it hasto retrieve all of them and make sure that there is a datapoint in them. The answer of the range query is all thepoints that lie in the range. Even if the query asks onlythe average of data objects in a given range, i.e. aggregation queries, it needs to read all the data points with in theMBRs and check whether they lie within the given range.However a Boolean range query can be answered withoutany page accesses as described above or with minimal number of page accesses. If a single relevant point is found inone of the MBRs, then we do not have to look for otherpoints and we can safely say that the corresponding point isnot an RNN. So, for the case of multiple MBRs intersectingthe query region, a decision has to be made to maximize the chance of finding a point in the MBR so that the querycan be answered with minimal number of MBRs accesses.In choosing the MBRs, there could be a number of possiblechoices:

1. Sort MBRs wrt. Overlap with search region and thenchoose MBR with maximum overlap.
2. Sort MBRs wrt. Mindist and then choose MBR withminimum mindist.
3. Choose randomly.We tried experimentally all the above three possibilitiesand found that choice 1 and 2 are comparable and both arebetter than choice 3. Most of the time the set of MBRsthat are retrieved to the memory in the first phase of thealgorithm is enough to guarantee that a point in a candidateset is not an RNN. In this case, no additional I/O is neededin the second phase. Therefore, almost with no additionaloverhead on the k-NN algorithm we can simply answer RNNqueries.

Multiple Boolean Range Queries

In the filtering step II we need to run multiple Boolean range queries since there are multiple points that remain after the filtering step I. Since these are candidates that are very close to each other, multiple queryoptimization becomes very natural and effective in this framework. Multiple queries are defined as the set of queries issued simultaneously as opposed to single queries which are issued independently. There have been a significant amount of research that investigated this approach for other kind of queries. In our algorithm, a Boolean range query needs to be executed for each point in the candidate set. Since the radius of the queries are expected to be very close to each other (because they are all in the k-NN of the query), executing multiple queries simultaneously reduces the I/O cost significantly. First we read a single page for the whole set of queries. The criteria for deciding which page to access first is to retrieve the page that has the most number of intersections with the queries. The performance gain of this approach is discussed next.

K-Best Algorithm Graph



IV. Conclusion

In this paper, we wish to find the objects similar to our query point, often our search needs to expand beyond the simple notion of nearest neighbor or *k*-nearest neighbors. For example, we might be interested in objects that may not be in the *k*-NN search neighborhood, but find our query to be closest to them. Furthermore, in all cases above, it's difficult to specify a range for similarity (as in the case for range queries), or a desired number of matching objects (i.e. *k* for *k*-NN) to be returned. They are typically very high dimensional, many mining algorithms or dimensionality reduction techniques are proposed to cope with the high dimensionality. To date, existing algorithms for exact RNN search work for only low-dimensional data. Some algorithms are proposed for arbitrary dimensionality, but they rely on indices such as R-tree or its variants, which do not scale well for high dimensionality. Our goal is thus to design algorithms that can scale to high dimensionality.

References

[1]. Norbert Beckmann , Hans-Peter Kriegel , Ralf Schneider , Bernhard Seeger, The R*-tree: an efficient and robust access method for points and rectangles, Proceedings of the 1990 ACM SIGMOD international conference on Management of data, p.322-331, May 23-26, 1990, Atlantic City, New Jersey, USA [doi>10.1145/93597.98741]

[2]. RimantasBenetis , S. Jensen , GytisKarĉiauskas , SimonasŜaltenis, Nearest and reverse nearest neighbor queries for moving objects, The VLDB Journal — The International Journal on Very Large Data Bases, v.15 n.3, p.229-249, September 2006 [doi>10.1007/s00778-005-0166-4]

[3]. Mark de Berg , Marc van Kreveld , Mark Overmars , Otfried Schwarzkopf, Computational geometry: algorithms and applications, Springer-Verlag New York, Inc., Secaucus, NJ, 1997.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Issue 9, September 2018

[4]. Muhammad Aamir Cheema , Xuemin Lin , Ying Zhang , Wei Wang , Wenjie Zhang, Lazy updates: an efficient technique to continuously monitoring reverse kNN, Proceedings of the VLDB Endowment, v.2 n.1, August 2009.

[5]. E. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1:269--271, 1959.

[6]. Yunjun Gao , Baihua Zheng, Continuous obstructed nearest neighbor queries in spatial databases, Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, June 29-July 02, 2009, Providence, Rhode Island, USA [doi>10.1145/1559845.1559906]

[7]. Yunjun Gao , Baihua Zheng , Gencai Chen , Wang-Chien Lee , Ken C. K. Lee , Qing Li, Visible Reverse k-Nearest Neighbor Query Processing in Spatial Databases, IEEE Transactions on Knowledge and Data Engineering, v.21 n.9, p.1314-1327, September 2009 [doi>10.1109/TKDE.2009.113]

[8]. Yunjun Gao , Baihua Zheng , Gencai Chen , Qing Li , XiaofaGuo, Continuous visible nearest neighbor query processing in spatial databases, The VLDB Journal — The International Journal on Very Large Data Bases, v.20 n.3, p.371-396, June 2011 [doi>10.1007/s00778-010-0200-z]

[9]. Flip Korn , S. Muthukrishnan, Influence sets based on reverse nearest neighbor queries, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, p.201-212, May 15-18, 2000, Dallas, Texas, USA [doi>10.1145/342009.335415]

[10]. Chuanwen Li , Yu Gu , Fangfang Li , Mo Chen, Moving K-Nearest Neighbor Query over Obstructed Regions, Proceedings of the 2010 12th International Asia-Pacific Web Conference, p.29-35, April 06-08, 2010 [doi>10.1109/APWeb.2010.28]