ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 9, September 2017

A Phase Of The Intelligence Cycle Associating K-Nearest Neighbors Algorithm With Condensed Nearest Neighbor Algorithm

[1] G.Sujatha M.C.A, [2]K. Madhavan M.C.A M.B.A M.Phil (Phd), [3]Dr.P.Guhan M.C.A. M.Phil. Phd
[1] (M.Phil., Student), Vels University.
[2]Assistant Professor,Cs Dept.,Vels University.
[3]Principal, Jaya College Of Arts And Science-Thiruninravur.

Abstract: **This paper makes clear about two popular algorithms namely k-Nearest Neighbors (K-NN) and Condensed nearest neighbor (CNN) and accomplishes a comparison between them; nowadays classification is a significant in data mining methodology. It can be used for classifying the interested users.**
The classification algorithms have been projected in the earlier eight decades. Every part of them has their own highlights constraint and highlights limitation. All of them have changeable run time, space necessities, consistency and ability for higher performance.
We explore the effect of a level of consciousness on the "nearest neighbor" problem. It also talks about parameter selection, the I-nearest neighbour classifier, nearest neighbor search and various methods in it. This paper also concludes about CNN for data reduction.

I. INTRODUCTION

In pattern detection, the **k-nearest neighbors algorithm** (**k-NN**) is a non-parametric system used for categorization and regression [1].

In both cases, the effort consists of the k closest instruction examples in the quality space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the productivity is a class membership. An object is off the record by a greater part of its neighbors, with the entity being assigned to the class most common in the middle of its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the group of that particular nearest neighbor.
- In k-NN regression, the output is the value for the object. This value is the average of the values of its k nearest neighbors.

II. K-nearest neighbors algorithm (k-NN)

K - NN is a type of instance-based knowledge, or lazy knowledge, where the purpose is only approximated nearby and all computation is delayed until categorization. The *k*-NN algorithm is in the middle of the simplest of all machine knowledge algorithms.

Both for categorization and regression, it can be practical to allocate weight to the assistance of the neighbors, so that the earlier neighbors donate more to the standard than the more remote ones. For example, a frequent weighting technique consists in giving each neighbour a weight of 1/*d*, where *d* is the distance to the neighbor [2].

The neighbors are taken from a set of items for which the class (for *k*-NN classification) or the object assets value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no precise training step is required.

A inadequacy of the *k*-NN algorithm is that its responsive to the local structure of the data. The algorithm is not to be at a complete loss with *k*-means, another popular machine learning method.

III. Parameter selection

The most excellent option of *k* depends upon the data; normally, larger standards of *k* reduce the effect of noise on the categorization [3]. But make limitations between classes is less distinct. A good quality *k* can be preferred by a variety of heuristic techniques parameter optimization. The unusual case where the class is predicted to be the class of the neighboring training example (i.e. when $k = 1$) is called the nearest neighbor algorithm.

The accurateness of the *k*-NN algorithm can be strictly ruined by the occurrence of noisy or unrelated features, or if the characteristic scales are not reliable with their significance. More research attempt has been put into selecting or scaling features to develop categorization. A mainly popular approach is the use of evolutionary algorithms to optimize feature scaling [4]. An additional admired approach is to scale features by the reciprocal information of the training data with the training classes.

In binary categorization problems, it is supportive to choose *k* to be an odd number as this avoids tied votes. One popular way of choosing the relying optimal *k* in this setting is via bootstrap method [5].

3.1. The l-nearest neighbor classifier

The most sensitive nearest neighbour type classifier is the one nearest neighbour classifier that assigns a point *x* to the class of its closest neighbour in the feature space,as the size of preparation data set approaches infinity, the one nearest neighbour classifier guarantees an fault rate which is not inferior than twice the Bayes error rate.

3.2. Nearest neighbor search (NNS)

Nearest neighbor search (NNS), also known as closeness search, is the optimization problem of discovering the point in a known set that is neighboring (or most similar) to a given point. Closeness is typically articulated in terms of a distinction function: the less similar the objects, the larger the function values. Officially, the nearest-neighbor (NN) search difficulty is defined as follows: given a set *S* of points in a space *M* and a query point $q \in M$, find the neighboring point in *S* to *q*.

Most frequently *M* is a metric space and difference is expressed as a distance metric, which is symmetric and satisfies the triangle variation. Even more common, *M* is taken to be the *d*-dimensional vector space where the difference is calculated using the Euclidean distance, Manhattan distance or other distance metric. On the other hand, the difference function can be subjective. One example is the asymmetric Bregman divergences, for which the triangle dissimilarity does not hold [6].

3.2.1. Various Methods of Nearest neighbor search

Various solutions to the NNS obstruction have been anticipated. The value and utility of the algorithms are determined by the time density of queries as fine as the space difficulty of any exploration for data structures that must be maintained. The comfortable examination usually referred to

as the size or extent states that there is no general-purpose accurate solution for NNS in high-dimensional Euclidean space using polynomial preliminary processing and polylogarithmic seek out time.

3.2.1.1. Linear search

The simplest solution to the NNS problem is to calculate the remoteness from the uncertainty point to every other point in the database, maintaining track of the "greatest so far". This algorithm, occasionally referred to as the inexperienced approach, has a running time of $O(dN)$ where N is the cardinality of S and d is the dimensionality of M. There are no search data structures to maintain, so linear exploration has no space complication beyond the storage of the database. Inexperienced search can, on standard, outperform space partitioning approaches on sophisticated dimensional spaces [7].

3.2.1.2 Space partitioning

Since the 1970s, division and bound methodology has been functional to the difficulty. In the case of Euclidean space this advance is known as spatial directory or spatial right to use methods. Several space-partitioning methods have been developed for solving the NNS problem. Possibly the simplest is the k-d tree, which iteratively divides the search space into two regions containing partially the points of the parent region.

Queries are performed via traversal of the tree from the root to a leaf by evaluating the inquiry point at each split. Depending on the remoteness specified in the query, adjoining undergrowth that might contain hits may also require to be evaluated. For constant dimension query point in time, usual difficulty is $O(\log N)$[8] in the case of pointlessly spread points, worst case difficulty is $O(kN^(1-1/k))$[9] .

On the other hand the R-tree data collection was measured to maintain nearest neighbor search in dynamic context, as it has well-organized algorithms for insertions and deletions such as the R tree [10]. R-trees can yield adjacent neighbors not only for Euclidean remoteness, but can also be used with other distances.

In case of all-purpose metric space division and bound approach is known under the name of metric trees. Particular examples include vp-tree and BK-tree.

By means of a set of points in use from a 3-dimensional space and set into a BSP tree, and given a inquiry position taken from the same space, a probable solution to the difficulty of finding the nearest point-cloud position to the query point is given in the subsequent description of an algorithm. Firmly speaking, no such point may exist, because it may not be exclusive. But in practice, usually we only care about ruling any one of the detachment of all point-cloud points that subsist at the shortest distance to a given query position.

The scheme is, for each branching of the tree, guess that the neighboring point in the cloud resides in the half-space containing the inquiry position. This might not be the case, but it is a good quality heuristic. After having recursively moved out all the way through, all the trouble of solving the difficulty for the half-space, now evaluate the remoteness returned by this effect by way of the shortest distance from the query position to the partitioning plane.

This final distance is that which is connecting the query position and the closest possible point so that it could be present in the half-space which is not searched. If this remoteness is superior than

that returned in the previous result, then obviously there is no want to look for the other half-space. If there is such a need, then we must go all the way through the problem of solving the difficulty for the other half space, and then evaluate its result to the earlier result, and then return the proper result.

The performance of this algorithm is nearer to logarithmic point in time than linear moment when the query point is near the cloud, because as the distance involving the query point and the neighboring point-cloud point nears zero, the algorithm needs only a look-up using the query point as a key to get the correct result.

3.2.1.3 Position sensitive hashing

Position sensitive hashing (PSH) is a method is for grouping points in space into 'buckets' based on some remoteness metric working on the points. Points that are close to each other beneath the chosen metric are mapped to the similar bucket with high probability [11].

3.2.1.4 Nearest neighbor search in spaces with small intrinsic dimension

The cover tree has a imaginary 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 growth constant of the dataset.

3.2.1.5 Projected radial search

In the extraordinary case where the data is a intense 3D map of geometric points, the projection geometry of the sensing method can be used to considerably make simpler the search problem. This advance requires that the 3D data is structured by a projection to a two dimensional grid and assumes that the data is spatially leveled across neighboring grid cells with the exclusion of object boundaries.

These assumptions are valid when operating with 3D sensor data in applications such as surveying, robotics and hi-fi vision but may not hold for unorganized data in general. In practice this method 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.

3.2.1.6 Vector approximation files

In elevated dimensional spaces, tree indexing structures turn out to be useless because an increasing proportion of the nodes need to be examined anyway. To fasten up linear search, a compressed account of the feature vectors stored in RAM is used to pre-filter the datasets in a first run. The concluding candidates are strong-minded in a second stage using the uncompressed data from the disk for remote calculation [12].

3.2.1.7 Compression/clustering based search

The VA-file is in motion in the direction of is a particular case of a firmness based search, where each characteristic module is condensed consistently and separately. The optimal firmness method in multidimensional spaces is Vector Quantization (VQ), implemented all the way through clustering. The database is clustered and the greater parts of “capable” clusters are retrieved. Massive gains over VA-File, tree-based indexes and sequential scan have been experiential [13][14]. Also note the parallels connecting clustering and LSH.

3.2.1.8 Unappeasable walk search in small-world graphs

One probable way to solve NNS is to create a graph, where every point is exclusively connected with the highest point. The exploration for the point in the set *S* closest to the query *q* takes the outline of the search of highest point in the graph. One of the basic highest point explore algorithms in graphs with metric items is the greedy search algorithm. It starts from the random highest point.

The algorithm computes a distance value from the uncertainty q to each highest point from the neighborhood of the current highest point, and then selects a highest point with the minimal distance value. If the distance value between the query and the selected highest point is smaller than the one between the query and the current element, then the algorithm moves to the selected highest point, and it becomes new current highest point.

The algorithm stops when it reaches a local minimum: a highest point whose neighborhood does not contain a highest point that is closer to the query than the highest point itself. This idea was exploited in the VoroNet system for the plane' in the RayNet system for the ,[15] and for the general metric space in the Metrized Small World algorithm.[16]

IV. CNN FOR DATA REDUCTION

Condensed nearest neighbor (CNN, the *Hart algorithm*) is an algorithm designed to reduce the data set for *k*-NN categorization [17]. It selects the set of prototypes *U* from the instruction data, such that 1NN with *U* and can categorize the examples almost as exactly as 1NN does with the whole data set.

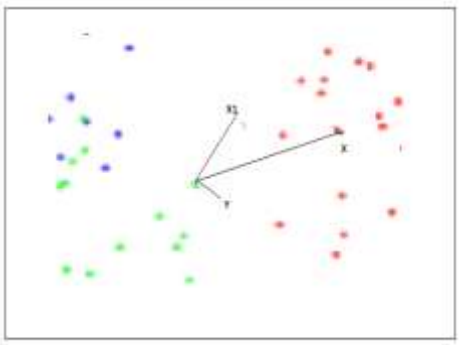

Figure 1 : Calculation of the border ratio.

Three types of points: prototypes, class-outliers, and absorbed points.

1. **Outliers**: Points which would not be documented as the correct type if added to the database later.
2. **Prototypes**: The lowest set of points required in the training set for all the other non-outlier points to be correctly documented.
3. **Absorbed points:** Points which are not outliers, and would be properly recognized based just on the set of prototype points.

Given a exercise set *X*, CNN works iteratively:

- Scan all elements of *X*, looking for an component *x* whose nearest prototype from *U* has a different label than *x*.
- Remove *x* from *X* and add it to *U*
- Replicate the scan until no more prototypes are added to *U*.

Use *U* as an alternative of *X* for categorization. The examples that are not prototypes are called "absorbed" points.

It is well-organized to scan the training examples in order of decreasing border ratio[18].The limit ratio of a training example *x* is defined as

$$a(x) = \frac{x' - y}{x - y}$$

where ||*x*-*y*|| is the distance to the neighboring example *y* having a dissimilar color than *x*, and ||*x'*-*y*|| is the remoteness from *y* to its neighboring example *x'* with the same label as *x*.

The border ratio is in the period [0,1] since ||*x'*-*y*||never exceeds ||*x*-*y*||.

This ordering gives first choice to the borders of the classes for addition in the set of prototypes *U*. A point of a dissimilar label than *x* is called external to *x*. The statistics points are identified by colors: the initial point is *x* and its label is red. External points are blue and green. The point neighboring to *x* is external point *y*. The closest to *y* red point is *x'* . The limit ratio *a*(*x*) = ||*x'*-*y*|| / ||*x*-*y*||is the characteristic of the initial point *x*.

CNN is also quite affected by noise in the preparation set. In order to examine this, three data sets were produced as usual, and for each one the number of noise points was improved, and CNN was run, recording the percentage of points assigned to each type. The results from averaging these three different data sets are shown in the table below:

These data are also accessible as a graph in Figure 1 for clarity. As can easily be seen, ever-increasing the number of random noise points affected the results of the CNN algorithm in three main ways:

1. The percentage of points classed as outliers improved considerably.

2. The percentage of points classed as captivated decreased.

3. The percentage of points classed as prototypes improved slightly. All of these areas would be predictable.

The proportion of outliers increases because there are more and more many noise points of the other color in each cluster, which will lead them to be mis-classified. The percentage of points deemed to be prototypes increases because our data set now has a much more complex arrangement, once we have included all these random noise points. The percentage of absorbed points therefore must reduce since the other two types are increasing.

V. CONCLUSION AND FUTURE ENHANCEMENT

The CNN algorithm can take a long time to run, mainly on very large data sets, as described. There are different methods for reducing dimensionality, which are described. Using CNN can cause our system to give us dissimilar classifications than if we just used k-NN on the raw data.

We can use the k-NN algorithm to categorize new points, by giving the same categorization as those which the new point is close to I For large datasets, we can apply CNN data reduction to get a lesser training set to work with I Tests after CNN will be obtained more rapidly, and will usually have the same result

These have been *proposed* to decrease the training set to gain on speed and space efficiencies. To ensure the minimalistic of this preparation set we presented some recent proposals using meta heuristics to check the optimality of the consequential set of some KNN reduction techniques. Note that each method is very efficient in a specific area and in special circumstances.

References

1. Altman, N. S. (1992). " An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician . 46(3):175 – 185. doi : 10.1080 / 00031305 . 1992 . 10475879.

2. Stone C. J. (1977). "Consistent nonparametric regression". Annals of Statistics. 5 (4) : 595 – 620. doi : 10.1214 /aos/1176343886.

3. Everitt, B. S., Landau, S., Leese, M. and Stahl, D. (2011) Miscellaneous Clustering Methods, in Cluster Analysis, 5th Edition, John Wiley & Sons, Ltd, Chichester, UK.

4. Nigsch F, Bender A, van Buuren B, Tissen J, Nigsch E, Mitchell JB (2006). "Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization". Journal of Chemical Information and Modeling. 46 (6): 2412– 2422. do i : 10 . 1021 / ci060149f . PMID 17125183.

5. Hall P, Park BU, Samworth RJ (2008). "Choice of neighbor order in nearest-neighbor classification". Annals of Statistics. 36 (5): 2135–2152. doi:10.1214/07-AOS537.

6. Cayton, Lawerence (2008). "Fast nearest neighbor retrieval for bregman divergences.". Proceedings of the 25th international conference on Machine learning: 112–119.

7. Weber, Schek, Blott. "A quantitative analysis and performance study for similarity search methods in high dimensional spaces" (PDF).

8. Andrew Moore. "An introductory tutorial on KD trees" (PDF).

9. 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.

10. 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. ISBN 0897917316.

11. A. Rajaraman & J. Ullman (2010). "Mining of Massive Datasets, Ch. 3.".

12. Weber, Blott. "An Approximation-Based Data Structure for Similarity Search".

13. Ramaswamy, Rose, ICIP 2007. "Adaptive cluster-distance bounding for similarity search in image databases".

14. Ramaswamy, Rose, TKDE 2010. "Adaptive cluster-distance bounding for high-dimensional indexing".

15. Olivier, Beaumont; Kermarrec, Anne-Marie; Marchal, Loris; Rivière, Etienne (2006). "VoroNet: A scalable object network based on Voronoi tessellations". INRIA. RR-5833 (1): 23–29. doi:10.1007/BF00263763.

16.Olivier, Beaumont; Kermarrec, Anne-Marie; Rivière, Etienne (2007). "Peer to Peer Multidimensional Overlays: Approximating Complex Structures". Principles of Distributed Systems**.** 4878 (.): 315 –328 . doi : 10.1007 /978-3-540-77096-1_23. ISBN 978-3-540-77095-4.

17. P. E. Hart, The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory 18 (1968) 515–516. doi: 10.1109/TIT.1968.1054155

18. E. M. Mirkes, KNN and Potential Energy: applet. University of Leicester, 2011.