Hindawi
Complexity
Volume 2021, Article ID 2011738, 9 pages
https://doi.org/10.1155/2021/2011738

WILEY | Hindawi

Research Article

Using Proximity Graph Cut for Fast and Robust Instance-Based Classification in Large Datasets

Stanislav Protasov and Adil Mehmood Khan

Machine Learning and Knowledge Representation Lab, Innopolis University, Innopolis 420500, Innopolis, Russia

Correspondence should be addressed to Stanislav Protasov; s.protasov@innopolis.ru

Received 17 May 2021; Accepted 29 October 2021; Published 29 November 2021

Academic Editor: Siew Ann Cheong

K-nearest neighbours (kNN) is a very popular instance-based classifier due to its simplicity and good empirical performance. However, large-scale datasets are a big problem for building fast and compact neighbourhood-based classifiers. This work presents the design and implementation of a classification algorithm with index data structures, which would allow us to build fast and scalable solutions for large multidimensional datasets. We propose a novel approach that uses navigable small-world (NSW) proximity graph representation of large-scale datasets. Our approach shows 2–4 times classification speedup for both average and 99th percentile time with asymptotically close classification accuracy compared to the 1-NN method. We observe two orders of magnitude better classification time in cases when method uses swap memory. We show that NSW graph used in our method outperforms other proximity graphs in classification accuracy. Our results suggest that the algorithm can be used in large-scale applications for fast and robust classification, especially when the search index is already constructed for the data.

1. Introduction

Proximity graphs are a practical class of graphs with applications in multiple areas. For example, they are used for motion planning, as rapidly exploring random trees in [1, 2] and minimum spanning trees in clustering [3]. Most importantly, they lay in the core of $\log(|V|)$ search time data structures for large-scale multidimensional data indexing, where $|V|$ stands for dataset cardinality.

Instance-based classification (IbC) methods store items (instances) from the training dataset as part of the classifier. Unlike other methods such as decision trees and artificial neural networks, the IbC algorithms do not estimate the classifier function from the training data in advance; instead, they store training data and derive a class label from an examination of the unseen sample's nearest neighbours at test time [4]. Such methods easily adopt to unseen data by extending the list of stored samples.

Among pure IbC methods, we can identify k-nearest neighbours (kNN) with different variations [5–7], piecewise functions (e.g., splines [8]), and kernel approximators, such as radial basis function (RBF) interpolation methods. Splines and kernel approximation are frequently used in numerical methods for equation solving. At the same time, kNN is considered both a good basis for novel machine learning approaches [5] and useful tool for complex applied machine learning tasks [9].

Decision trees, support vector machines (SVM), self-organizing maps [10], learning vector quantization [11], and RBF networks [12] can also be attributed to instance-based methods. However, we avoid such wide interpretation, as these methods do not require storing original samples for classification.

In this paper, we address the problem of classification speed in the context of IbC using proximity graphs. Large datasets often appear in content recommendation tasks of internet services: search, online shops, streaming, or social networks. Classification accompanies recommendations in such tasks as sentiment analysis [13] or auto-labelling [14]. As such systems serve millions or even billions of requests per day, this makes a millisecond algorithm overhead scale into hours and days of CPU time every day. This is a notable financial load.

The K-nearest neighbours method estimates the class label of a test sample based on the labels of its closest neighbours from the training set. Distance is defined with some metric function. To avoid computing the distance of the test sample to every item in the training data, indexing is employed. This allows achieving sublinear classification time with various data structures such as trees, graphs, and inverted indices.

Graph-based indexing utilizes the idea that a dataset can often be represented in a metric space. Thus, adding a distance metric for nodes and requiring edges to represent close neighbourhoods, we can benefit from greedy-like search algorithms, traversing the graph with preliminary knowledge about the desired direction towards a query sample.

In instance-based methods, algorithm execution time depends on the number of stored instances, while model-based methods depend on the number of model parameters. Thus, IbC should offer both asymptotically and practically fast methods even for very large datasets, which requires constructing additional structures to navigate the data, such as search indices. We consider the case where index creation is indispensable and try to reduce the classification wall time. More specifically, we show how navigable small-world (NSW) [15] and hierarchical navigable small-world (HNSW) graphs [16] properties can be utilized in machine learning. We propose an improvement to NSW and HNSW index data structures, which results in 2–4 times sustainable speedup on average compared to 1-NN classification baseline.

The contribution of this work can be summarized as follows:

(i) We propose a new instance-based classification approach, which utilizes properties of NSW and HNSW index data structures to achieve 2–4 times 1-NN classification speedup.

(ii) Our proposed methods show a 2-order time improvement when used with a memory swap file.

The rest of the paper is organized as follows. Section 2 discusses different indexing strategies for large multidimensional datasets. Section 3 covers both algorithm construction and theoretical justification of the proposed idea. Section 4 describes the experimental setup, datasets, hardware, and ways of comparison of our method to other approaches. Section 5 is devoted to the numerical results of our experiments. In this section, a proposed classifier is assessed in terms of speed and accuracy, and the NSW graph is compared to other proximity graphs. Section 6 analyses obtained numbers and state conditions in which using our method is beneficial and discusses interesting properties. Section 7 closes the paper with a highlight of major outcomes.

Our experiments, results, and code are available in the GitHub repository (https://github.com/IUCVLab/proximity-cut).

2. Related Work

This section overviews how a problem of large dataset indexing is solved in the industry right now and considers indexing application to instance-based learning.

The problem of large-scale indexing for multidimensional data arose together with efficient methods of document embedding using artificial neural networks [17–19]. The Internet became an endless source of data, including web pages, Wikipedia articles (https://dumps.wikimedia.org/), scientific papers (https://en.wikipedia.org/wiki/Web_of_Science) and images (https://en.wikipedia.org/wiki/Google_Photos) which form collections with 10^5 to 10^{10} items in each. Contemporary research in natural language processing also requires bigger datasets to prove robustness [20]. As there is no exact borderline, we address these sizes as large. Search on such a scale can no longer be exhaustive. To be practical, it requires sublinear time. For an unsorted collection, this means that, on classical computers, we need to use approximate methods, also known as approximate nearest neighbour search (ANNS). Exact and approximate nearest neighbour searches are the core tool of many metric-based machine learning algorithms, including kNN classification, k-means, k-centroids, and DBSCAN clustering. We discuss three approaches to building large dataset indices to guarantee fast ANNS. In this paper, we assume that the data can be represented in a metric or in a vector space depending on indexing method.

2.1. Tree Based. The invention of AVL-trees and B-trees made search trees a powerful tool to build $\mathcal{O}(\log(|V|))$ indices for numerical data. Quad-trees [21] and KD-trees [22] have been used for indexing multidimensional vector data. Unfortunately, their usage is limited to low dimensions because they suffer from the curse of dimensionality. For example, indexing 10^9 items with KD-tree will utilize at most $\log_2(10^9) \approx 30$ first dimensions of the vector, while contemporary deep models produce 100–1000 dimensional vectors, such as 768-dimensional BERT embeddings as in [17]. For such big vectors, the search procedure will not account the majority of dimensions. Thus, it cannot guarantee a low distance from the query to the obtained "neighbours." To solve this problem, authors of annoy (https://github.com/spotify/annoy) apply random projections instead of predefined vector dimensions and multiple search trees, which is proven an efficient way of reducing data dimensions for large datasets [23]. A collection of trees can achieve high ANNS accuracy with a small search time. However, it comes with a significant memory overhead as each tree consumes memory proportional to dataset size.

2.2. Inverted Index Based. An inverted index file (IVF) is an efficient method for text indexing, as it utilizes statistical properties of human language and discrete word representation. Since multidimensional vector data is continuous, various metric-based discretization approaches, such as vector quantization and vector clustering, are used to prepare the so-called vocabularies—finite collections of vectors, representing data clusters [24, 25]. Current works discuss methods to avoid the problem from which IVF suffers in natural language processing. Word frequencies in natural language are different, leading to a skewed index. The proposed methods include different k-means implementations to form a vocabulary and product quantization

technique for a better space partitioning as in [24]. Though IVF is a fast and scalable method with promising search speed and ANNS accuracy, it requires significant additional memory [26].

2.2.1. Proximity Graph-Based. A proximity graph is a graph with a distance metric defined for vertices [27]. In practice, the metric can be defined not for all pairs of vertices, and edge in such a graph exists if and only if (or with higher probability if) its vertices satisfy particular geometric requirements; for example, if they are close in metric space. Building a proximity graph with dataset items as vertices can be understood as building a road network. It allows the search algorithm to travel starting from the arbitrary vertex in the direction of the search query by following some greedy strategy.

There are multiple types of deterministic and probabilistic proximity graphs, including minimum spanning trees (MST), relative neighbourhood graphs (RNG), Gabriel graphs, and Delaunay triangulations [28]. Among them, there are a group of data structures based on the idea of small-world graphs. A major feature of small-world (SW) networks [29] compared to other graphs is that together with edges connecting tight neighbourhoods (compare with local roads), they also include "distant" edges (compare with flights). In this example, "distant" means the edge which connects near-clique clusters which do not share any nodes. The existence of such "distant" edges leads to $\mathbb{E}(\log(|V|))$ expected shortest path length (edges count) between arbitrary pair of vertices which was proven in [30]. NSW and HNSW graphs [16] place graph vertices into a metric space, introducing a highly efficient greedy-like algorithm to traverse the graph. The authors claim that their data structure approximates Delaunay triangulations in high dimensions and propose a novel method of constructing SW graphs in metric space, which has $\mathcal{O}(|V|\log|V|)$ construction time complexity and $|V| * d$ memory overhead, where d represents the number of dimensions in data vectors.

3. Methodology

This work is dedicated to an improvement of the IbC methods. Given a big multidimensional dataset, we can achieve good results with the kNN classifier: using existing search indices, we can guarantee $\log(|V|)$ search time without sacrificing accuracy. These methods are competitive and are used in recent applied research as in [5, 9]. Still, we must consider speed in terms of both theoretical complexity and wall time, as large-scale services are sensitive to even a millisecond overhead in a single function. Proximity graphs (NSW and HNSW in our case) built upon the unlabelled collection achieve expected logarithmic nearest neighbours search time with a greedy-like algorithm. Therefore using a graph-based index, the kNN classifier can run in logarithmic time, which can hardly be improved in terms of theoretical complexity. On the contrary, we concentrate our efforts on utilizing label information to reduce practical computation time and preserve classification accuracy.

Our work is limited to the assumption that the dataset has a property of a metric space, with high probability nearest neighbours (in terms of the metric) of an item belonging to the same class as the item. This assumption is sometimes called the compactness hypothesis [31]. This assumption is general for all metric-based machine learning methods, including both unsupervised (e.g., k-means and DBSCAN) and supervised (e.g., kNN and linear models) approaches.

The core theoretical idea of the proposed method lies in the fact that a proximity *graph cut* can be used to approximate the class boundaries. A graph cut is a *set of edges where the source and destination vertices belong to different classes*. A graph cut example is given in Figure 1.

The outline of our methodology is the following:

(i) The same as in kNN classification, we accept the compactness hypothesis. This allows making an assumption that classes are closed volumes in $\mathbb{R}^n$.

(ii) kNN classifier assigns a class to an unseen sample based on implicit class border estimation with neighbours voting. Border estimation can be replaced with faster border crossing detection, based on The Jordan Curve theorem [32] and its extension [33].

(iii) The proposed border crossing detection technique is based on traversing NSW and HNSW graphs with a greedy algorithm (beam search). This algorithm produces the near-shortest path between the starting point and unseen sample, which is shown to be $\log(|V|)$ long [30].

Further paragraphs expand the listed ideas.

The Jordan curve theorem guarantees that if there are two classes in $\mathbb{R}^2$ where one class is surrounded by a closed curve, then an arbitrary path between two points belonging to different classes intersects this border an odd number of times and at least once. We apply multidimensional consequence [33] of the theorem following the compactness hypothesis: for an arbitrary path (edges sequence) in proximity graph, single class border crossing can be used to indicate class change. An exact crossing point location is not needed. It is enough to account edges where vertices have different labels, that is, which belong to the graph cut. The method also works even if the class is not a single cluster but a set of disconnected clusters.

Speed characteristics of our implementation are derived from two properties of NSW and HNSW graphs. Firstly, in small-world graphs (by definition), the shortest path between two arbitrary vertices has expected logarithmic length. Thus, any query search algorithm can start from a random graph node and find the nodes closest to the query node in $\log(|V|)$ time on average if the shortest path is known. Secondly, the greedy beam search algorithm in a dense enough NSW graph produces a path with $\mathcal{O}(\log(|V|))$ edges with a probability of $1 - o(1)$, which is shown by the experiments in [15] and theoretical proof in [30]. Greediness here is defined as selecting the next vertex from the neighbours, such that it is the closest (in metric space) to the

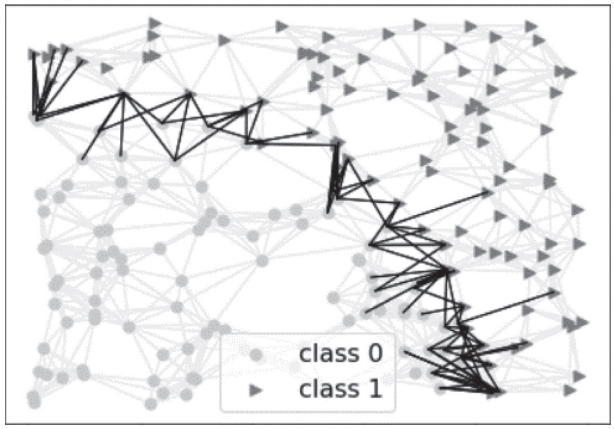

Figure 1: NSW index graph is built on 2 data classes (circles and triangles) in metric space. Black edges represent graph cut. They have source and destination vertices in different classes.

destination. Euclidean and angular metrics are the most popular for vector space datasets. In other words, if algorithms search for query node q starting in node n, at each step, it should move to such a neighbor n' of n, which has the smallest distance to q in terms of metric. The aforementioned properties guarantee that this search, on average, will successfully converge in $\log(|V|)$ time.

To sum up, for a classification task, class boundary estimation is not needed. Instead, it is enough to detect the event of boundary-crossing. This useful observation allows reducing computational overhead, which is valuable for large systems. For implementation, we use both properties of NSW graphs to efficiently obtain a path in a graph and combine them with the Jordan curve theorem.

A formal description of our method is as follows: let a class be a set of volumes in a multidimensional metric space. As we mentioned earlier, we approximate class boundaries with a graph cut. Boundary-crossing occurs if an edge in a path belongs to a cut. Thus, we propose the following algorithm of classification. Given a sample vector that needs to be classified, a search is started from a graph vertex with a random index taken from a uniform distribution. A vertex choice procedure does not influence the result as any shorted path in the NSW graph has a logarithmic length. Then, graph nodes are greedily traversed towards the given vector, and the algorithm stops if it cannot find a closer neighbour. If class labels are available for all vertices, only the last border crossing is needed (if any happened) to assign a class to the sample.

The algorithm works for both binary and multiclass problems. Generalization to multiclass comes from applying the one-vs-all technique: the last border crossing can be considered as moving from the united "all" class to "one" class.

The proposed approach for NSW graphs is summarized in algorithm 1, which is an approximate equivalent to 1-NN classification. The only difference for HNSW implementation requires to start at the top level of the graph and repeat the same algorithm at the lower levels until 0^{th} convergence. This also means that the choice of vertex v for HNSW graphs is deterministic.

The Euclidean distance for normalized features is used as a metric in tests if other is not mentioned explicitly. This choice is reasonable in many practical applications as it captures the human perception of "closeness": a significant change in the value of one feature or insignificant changes in multiple features should not dramatically influence the distance metric.

NSW graph allows using the proposed algorithm together with the kNN voting procedure. That is, classification can be run multiple times to achieve better accuracy. In the case of the HNSW graph, the search procedure always uses a predefined starting point. Thus applying voting will not bring any benefit.

4. Experiments

We implement our method to improve the original NSW and HNSW graph search procedure. Our experiments study our method from three points of view:

(i) NSW graph ANNS quality compared to other proximity graphs,
(ii) Classification accuracy compared to 1-NN,
(iii) Time improvement compared to baseline 1-NN classification with HNSW.

We understand *graph quality* criterion as an ability to provide a better approximation for the ANNS problem. Application of proximity graphs is always a trade-off between speed of neighbourhood exploration and percentage of actual neighbours retrieved (which can be referred to as recall metric). Experiments show that graph choice is good. The other two criteria are devoted to the method assessment for both accuracy and time. 1-NN classification is used as a baseline. The first reason is that a proposed method is an approximation for this classification technique, so we assess our solution compared to the best achievable nonexhaustive 1-NN classification method done with NHSW. Our target is to achieve better practical time with acceptable accuracy loss. The second reason is that based on 1-NN results, one can easily extrapolate the time cost for an arbitrary kNN classification method.

To compare a *graph type* used in our method with all graph types presented in survey [28], we run experiments with 3 UCI datasets mentioned in a paper: Dermatology (https://archive.ics.uci.edu/ml/datasets/dermatology), Isolet (https://archive.ics.uci.edu/ml/datasets/isolet), and Image Segmentation (https://archive.ics.uci.edu/ml/datasets/Image + Segmentation). Non-normalized Euclidean distance is used for Isolet and Image Segmentation to reproduce original accuracy results. For the Dermatology dataset, as it contains both categorical and numerical features, we implement and use the Heterogeneous Value Difference Metric (HVDM) defined in [34] (implemented in https://github.com/IUCVLab/proximity-cut/blob/master/modules/tools/hvdm.py). For this set of experiments, NSW implementation from our repository was used.

By construction, the HNSW graph contains the NSW graph as a subset on level 0. Thus, all remaining experiments are conducted with hnswlib implementation where NSW graphs are extracted from the parent HNSW graph.

The *speed and accuracy* of classification are compared to the 1-NN classifier on the medium-size road signs dataset [35] with 43 classes (images are resized to 256-dimensional representation, 10% test set) and two large binary classification datasets HIGGS ($1.1 * 10^7$ items) and SUSY ($5 * 10^6$ items) [36] (5% test set). Detailed speedup statistics are measured using another medium-size Cover Type dataset [37].

In this work, we do not claim to invent or improve existing classification algorithm(s). These types of work require exhaustive testing for all marginal cases. We aim to apply state-of-the-art indexing infrastructure and show what can we gain from it (in speedup) and at what cost (in accuracy). As NSW and HNSW's time complexity characteristics have already been studied [15, 16] and proven [30], in this paper, we focus on practical improvement. Since algorithm time is shown to depend on dataset dimensionality and size, we cover both aspects.

All experiments were conducted at a 64-bit Windows 10 laptop using a single CPU core. The laptop has AMD Rizen 3 3200U chip with 2.6 GHz frequency and 2 physical cores. 6 GB RAM is installed in the machine with 3.5 GB available for experiments. Python implementations were launched at Jupyter notebooks with Python 3.7.4. C++ implementation was compiled with GCC 7.4.0 using Windows Subsystem for Linux (Ubuntu 18.04).

5. Results

5.1. Graphs Comparison. The choice of NSW graph was validated by comparing accuracy results with other proximity graphs, namely, relative neighbourhood graphs (RNG), Gabriel graphs, and minimum spanning trees (MST). We compare our results against implementations from [28] on UCI datasets proposed in the paper. The authors intentionally focused only on classification accuracy and omitted speed comparison. Thus, we can compare our results by accuracy only. On Isolet, our method outperformed RNG graphs classification with 88.5% accuracy against 88.1%. With Dermatology data, it achieved the same 95.65% accuracy as RNG graphs, which can be the result of very small dataset and almost complete graph. With Image Segmentation data, our method achieved 87.5% accuracy which is only slightly worse than 1-NN (90.3%) and RNG (88.8%). Detailed results are given in Table 1.

5.2. Average Classification Accuracy and Time. In NSW and HNSW graphs, the construction phase depends on hyperparameter M, which linearly influences the number of graph edges. According to original papers, increasing this parameter can bring better accuracy results paying with additional index memory. We compared how this parameter influences baseline 1-NN classification and the proposed method on two large datasets. Results are provided in Figure 2.

We also studied how dataset size and graph edges density controlled by NSW hyperparameter M influence average classification time and accuracy. We compared baseline 1-NN classification with the proposed method on three datasets with different hyperparameter values. With comparable accuracy number, our method showed sustainable speedup on both graphs. Time and accuracy results are given in Table 2.

NSW and HNSW graphs are built by a deterministic procedure, but their properties depend on the order of inserting and the structure of the dataset itself. Search and classification time for such graphs can only be estimated in terms of expected values. We used a medium-size Cover Type dataset to compare classification times distributions. While both baseline 1-NN classification and NSW-based proposed method show comparable time spread growth, the HNSW-based method shows extremely good numbers. For visual comparison, please refer to Figure 3.

5.3. Service Reliability Comparison. Indexing structures are used in different search tasks to improve the quality of service. Service reliability is frequently assessed in terms of 95^{th} or 99^{th} percentiles. Thus, we prepared a percentile-based comparison of the proposed method to the 1-NN baseline, which shows 1.5–2 times speedup for NSW-based implementation and 4+ times comparison for NSW-based. The numbers are given in Table 3.

6. Discussion

The experiments on graph comparison show that NSW-based classification accuracy outperforms sparse Gabriel and MST graphs in all experiments. Also, the resulting classifier shows a behavior very similar to RNG-based implementation for each of the experiments. Considering this, we refer to [38], which states that although 2-dimensional case RNG construction requires $\Theta(|V|\log|V|)$ operations, the k-dimensional and non-Euclidean metric spaces will require $\mathcal{O}(|V|^3)$ operations. NSW graphs are constructed in $\mathbb{E}(|V|\log|V|)$ which is a significant gain for large-scale datasets.

Comparison of the proposed method accuracy to the baseline 1-NN classifier shows that the proposed method is slightly under the baseline, HNSW-based implementation in all tests 0–7 percent points worse than 1-NN. But we also observed that NSW-based implementation asymptotically tends to the baseline (see Figure 2) with the growth of graph density, defined by M hyperparameter. For the small size of the dataset, the NSW-based implementation showed significant speedup, thus using k–NN classifiers where $k < speedup$ built upon the proposed method will achieve asymptotically better accuracy for the smaller time. HNSW-based implementation at the same time shows consistent speedup for all graph sizes and densities. The loss in accuracy can be explained by the fact that the final step of our method returns only an approximation of the nearest neighbour. Thus, in future studies, improvement of the algorithm for a better approximation at the last search iterations can be addressed to compete with the baseline in accuracy while preserving similar time. We can also say that speedup was observed at all experiments for all sizes of the datasets and graph densities. A general observation is that speedup tends

Input: *NSW* - dataset index; x - sample to be classified
Result: *class* label
$v \longleftarrow$ random vertex from *NSW*;
$d_{new} \longleftarrow distance(v, x)$;
$class_{new} \longleftarrow v.class$
repeat
  $d \longleftarrow d_{new}$;
  $class \longleftarrow class_{new}$;
  // closest to x neighbours of d
  $v \longleftarrow closest(v.neighbours \mid x)$;
  $d_{new} \longleftarrow distance(v, x)$;
  $class_{new} \longleftarrow v.class$;
  // until we can't get closer to x
until $d_{new} > d$;
return *class*

Algorithm 1: Greedy graph-based in NSW for multiclass classification.

Table 1: Comparison of classification accuracy for different proximity graphs against SVM classifier with polynomial kernel.

Dataset	Accuracy (%)					
	SVM	kNN	RNG	MST	Gabriel	NSW
Dermatology ($k = 10$)	96.9	98.6	95.6	86.7	65.4	**95.65**
Isolet ($k = 7$)	96.0	91.9	88.1	N/A	N/A	**88.5**
Image Segmentation ($k = 1$)	92.9	90.3	88.8	76.7	82.1	**87.5**

k refers to the best kNN hyperparameter. NSW is our choice. Bold shows our study implementations.

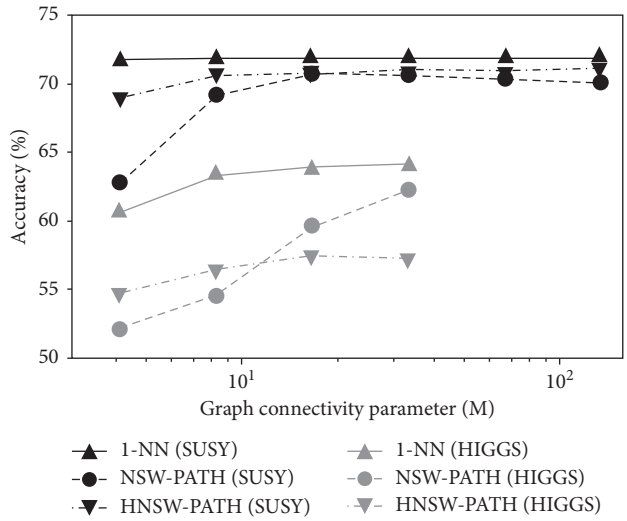

Figure 2: Accuracy comparison of the proposed NSW-PATH and HNSW-PATH classifiers with 1-NN baseline at two large-scale datasets SUSY (5M items) and HIGGS (11M items).

to be bigger for smaller datasets, but for 10^7-scale datasets, it remains on a significant level.

We separately stop on the speedup value for the SUSY dataset with high density ($M = 128$), which shows 1.17 times improvement for NSW and enormous 148.1 times improvement for HNSW (see Table 2). The experiment was conducted multiple times with different system parameters showing the same result. We found out that this behavior fully depends on using swap memory. For the small Road Signs dataset, doubling graph density (parameter M) implies linear absolute time growth for 1-NN classification, whereas for SUSY, we observe 3-order time growth compared to 16 times density growth ($M = 8, 128$). We detected that, at the test machine, the process could only allocate ~ 3.5 GB of physical RAM, while the data structure required almost 5 GB of virtual memory. Thus, a significant part of the data was dumped into HDD. This slowed down both the index construction phase and classification. The HNSW graph architecture uses exponentially smaller parts of memory for higher graph levels according to the construction process.

Table 2: Time and accuracy comparison of a proposed graph-based method implemented with NSW and HNSW graphs. Graph parameters are given as follows: dataset name (number of items, HNSW connectivity parameter *M)*.

Graph parameters	Time (ms), accuracy (%)						Speedup NSW vs. 1-NN	Speedup HNSW vs. 1-NN
	1-NN		NSW-PATH		HNSW-PATH			
Road signs (10K, $M = 8$)	209	92.57%	25.2	92.57%	72.4	92.57%	**8.3**	2.89
Road signs (10K, $M = 64$)	1615	82.57%	151	82.57%	186	82.57%	**10.7**	8.68
SUSY (5M, $M = 4$)	78.16	71.74%	22.41	62.74%	15.01	69.02%	3.49	**5.21**
SUSY (5M, $M = 8$)	89.5	71.87%	49.43	69.19%	21.43	70.58%	1.81	**4.18**
SUSY (5M, $M = 128$)	8538	71.86%	7271	71.12%	57.66	70.1%	1.17	**148.1**
HIGGS (11M, $M = 32$)	596.08	64.09%	253.61	62.27%	218.11	57.14%	2.35	**2.73**

Bold values refer to the best achieved speedup.

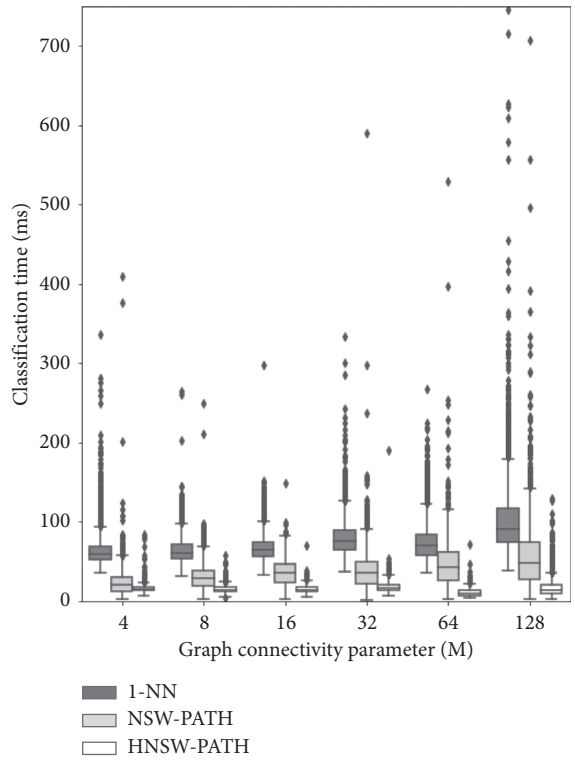

Figure 3: Comparison of classification time distribution on medium-size Cover Type dataset.

Table 3: Algorithm speedup comparison with respect to different graph densities (tuned by NSW *M* hyperparameter) and percentile. Speedup values compared to baseline HNSW-based 1-NN classification are provided for 99^{th} percentile (p99), 95^{th} percentile (p95), and average classification time (Avg).

M	NSW-PATH			HNSW-PATH		
	p99	p95	Avg	p99	p95	Avg
4	2.44	2.17	2.85	5.59	4.55	4.16
8	1.56	1.71	2.12	4.07	4.36	4.34
16	1.47	1.42	1.84	4.31	4.18	4.45
32	1.55	1.55	2.10	4.51	4.14	4.62
64	1.32	1.26	1.61	**6.44**	**6.50**	**7.10**
128	1.45	1.42	1.82	4.11	3.18	5.30

Thus, only the last steps of the algorithm require 0-level graph traversal, while the level with higher numbers can easily fit into physical RAM. This makes HNSW-based classification very promising in cases of low physical RAM.

Considering potential service quality, we state that both NSW and HNSW-based implementations of the proposed method show comparable speed improvement for the average and high-percentile classification time. We also observe that this speedup does not depend on graph density as shown in Table 3.

To sum up, we define two potential applications for our proposed method. Firstly, both NSW- and HNSW-based algorithms can be used as dedicated classifiers for datasets of all sizes to improve absolute classification time still sacrificing a few percent points of accuracy. NSW-based classifier, in this case, will offer asymptotically growing accuracy for denser graphs, HNSW-based version will be extremely time-efficient when RAM does not fit the dataset, and a swap file is used. Secondly, for small- and medium-sized datasets, NSW-based kNN implementation can offer better accuracy for the same classification time.

We want to mention that although we discussed competing indexing approaches (IVF, trees) in Section 2, we cannot implement our methods with these data structures, as they do not produce the graph cut we use in both our methods.

7. Conclusion

In this paper, we introduced the novel approach to instance-based classification. The approach improves existing NSW and HNSW data structures for faster classification of unseen items. It simplifies the original search algorithm and connects it with the Jordan curve theorem. The method achieved sustainable 4x speedup at real medium-scale datasets and more than 2x speedup for a large dataset using production hnswlib C++ library while preserving asymptotically close accuracy. It also showed extremely good time improvement if used with swap file.

We analysed our solution execution time, and we can say that it provides significantly better reliability in terms of 95^{th} and 99^{th} classification time percentile compared to the 1-NN classification baseline.

Our future research can target improvement of nearest neighbour estimation in the end of the proposed search algorithm which will improve classification accuracy while keeping time smaller.

Data Availability

The machine learning classification datasets (Dermatology, Isolet, Image Segmentation, Cover Type, SUSY, and HIGGS) used to support the findings of this study have been deposited in the UCI repository (https://archive.ics.uci.edu/ml/datasets/dermatology, https://archive.ics.uci.edu/ml/datasets/isolet, archive.ics.uci.edu/ml/datasets/Image + Segmentation, https://archive.ics.uci.edu/ml/datasets/HIGGS, https://archive.ics.uci.edu/ml/datasets/SUSY, https://archive.ics.uci.edu/ml/datasets/covertype). These prior studies (and datasets) are cited at relevant places within the text as references [28, 36, 37]. The machine learning classification dataset with road signs used to support the findings of this study has been deposited in the INI Benchmark Website https://benchmark.ini.rub.de. These prior studies (and datasets) are cited at relevant places within the text as references [35]. The code related to synthetic dataset generation used to support the findings of this study has been deposited in the GitHub repository https://github.com/IUCVLab/proximity-cut.

Conflicts of Interest

Adil Mehmood Khan acts as an editor of Complexity and Robustness Trade-Off for Traditional and Deep Models special issue.

Acknowledgments

This research has been financially supported by The Analytical Center for the Government of the Russian Federation (Agreement No. 70-2021-00143 dd. 01.11.2021, IGK 000000D730321P5Q0002).

References

[1] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Technical Report, Iowa State University, Ames, Iowa, 1998.

[2] V. Massagué Respall, D. Devitt, and R. Fedorenko, "Unmanned aerial vehicle path planning for exploration mapping," Technical Report, Innopolis University, Innopolis, Russia, 2020.

[3] M. A. Carreira-Perpiñán and R. S. Zemel, "Proximity graphs for clustering and manifold learning," in *Proceedings of the 17th International Conference on Neural Information Processing Systems, NIPS'04*, pp. 225–232, MIT Press, Cambridge, MA, USA, December 2004.

[4] E. Keogh, *Instance-Based Learning*, pp. 549-550, Springer US, Boston, MA, USA, 2010.

[5] O. Anava and K. Levy, "k*-nearest neighbors: from global to local," in *Advances in Neural Information Processing Systems*, pp. 4916–4924, 2016.

[6] J. Gou, W. Qiu, Z. Yi, Y. Xu, Q. Mao, and Y. Zhan, "A local mean representation-based K -nearest neighbor classifier," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 3, pp. 1–25, 2019.

[7] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[8] L. Bookstein Fred, "Principal warps: thin-plate spline and decomposition of deformations," *Transictions on pattern analysis and machine intelligence*, vol. 11, 1992.

[9] C. Sitawarin and D. Wágner, "Defending against adversarial examples with k-nearest neighbor," 2019, https://arxiv.org/abs/1906.09525.

[10] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[11] T. Kohonen, "Learning vector quantization," in *Self-organizing Maps*, pp. 175–189, Springer, Berlin, Germany, 1995.

[12] D. Lowe, D. Broomhead, R. Signals, and M. U. K. Radar Establishment, *Radial Basis Functions Multivariable Functional Interpolation and Adaptive Networks. No. RSRE-MEMO-4148*, Royal Signals And Radar Establishment, Malvern, UK, 1988.

[13] M. Rezwanul, A. Ali, and A. Rahman, "Sentiment analysis on twitter data using knn and svm," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 6, 2017.

[14] K. Shah, H. Patel, D. Sanghvi, and M. Shah, "A comparative analysis of logistic regression, random forest and knn models for the text classification," *Augmented Human Research*, vol. 5, no. 1, 2020.

[15] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Approximate nearest neighbor algorithm based on navigable small world graphs," *Information Systems*, vol. 45, pp. 61–68, 2014.

[16] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," 2018, http://arxiv.org/abs/1810.04805.

[18] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM '13*, pp. 2333–2338, Association for Computing Machinery, California, CA, USA, November 2013.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[20] A. Onan, "Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach," *Computer Applications in Engineering Education*, vol. 29, no. 3, pp. 572–589, 2020.

[21] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[22] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[23] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary Mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[24] A. Babenko and V. Lempitsky, "The inverted multi-index," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 6, pp. 1247–1260, 2015.

[25] D. Baranchuk, A. Babenko, and Y. Malkov, "Revisiting the inverted indices for billion-scale approximate nearest neighbors," in *Computer Vision – ECCV 2018*, V. Ferrari,

M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., Springer International Publishing, Cham, Switzerland, pp. 209–224, 2018.

[26] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg, "Searching in one billion vectors: re-rank with source coding," 2011, https://arxiv.org/abs/1102.3828.

[27] L. Mathieson and P. Moscato, "An introduction to proximity graphs," Springer International Publishing, Cham, Switzerland, pp. 213–233, 2019.

[28] G. T. Toussaint and C. Berzan, "Proximity-graph instance-based learning, support vector machines, and high dimensionality: an empirical comparison," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, Ed., Springer, Berlin, Heidelberg, pp. 222–236, 2012.

[29] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[30] L. Prokhorenkova, "Graph-based nearest neighbor search: From practice to theory," 2019, http://arxiv.org/abs/1907.00845.

[31] A. Arkadev and E. Braverman, *Computers and Pattern Recognition*, Thompson Book Co. Inc., Washington, DC, USA, 1966.

[32] C. Jordan, *Cours d'analyse de l'École polytechnique*, Gauthier-Villars et fils, vol. 1, pp. 587–594, 1887.

[33] E. Spanier, *Algebraic Topology*, p. 198, McGraw-Hill, New York, NY, USA, 1966.

[34] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[35] V. N. Sichkar and S. A. Kolyubin, "Effect of various dimension convolutional layer filters on traffic sign classification accuracy," *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 19, no. 3, pp. 546–552, 2019.

[36] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature Communications*, vol. 5, no. 1, p. 4308, 2014.

[37] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[38] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognition*, vol. 12, no. 4, pp. 261–268, 1980.