Survey on KNN Methods in Data Science

Panos K. Syriopoulos(✉), Sotiris B. Kotsiantis, and Michael N. Vrahatis

Computational Intelligence Laboratory, Department of Mathematics,
University of Patras, 26110 Patras, Greece
{p.syriopoulos,sotos,vrahatis}@math.upatras.gr

Abstract. The k-nearest neighbors (KNN) algorithm remains a useful and widely applied approach. In the recent years, we have seen many advances in KNN methods, but few research works give a holistic account of all aspects of KNN and the progress made. This paper is a brief survey on modern KNN methods and their role in data science. Furthermore, we survey: the challenges, how they are approached in the literature, the impact of the distance metric, several KNN variations, as well as query methods.

Keywords: KNN methods · Data science · Instance based learning

1 Introduction

KNN belongs to the family of instance-based learning algorithms, a concept explained by Aha et al. [2]. Simply put, the training instances are stored in memory without explicitly learning a model. The training instances (referred to as "knowledge", training set, or simply dataset) will only be processed in the prediction phase. For each new data instance, a query is made to the knowledge base and, only then, the knowledge returned by the query is processed to produce a prediction. In the simplest case of KNN, the query returns the k nearest data points to the new instance (based on some distance or similarity metric). The intuition is that similar examples are good predictors of unseen examples. Apart from the simplicity of the algorithm, KNN classification enjoys a theoretical guarantee: the probability of error is bounded above by twice the Bayes probability of error (Loizou and Maybank [33]).

In KNN classification, the output is a class membership and is usually determined by the majority class of the instances returned. In KNN regression, the output is determined by the average of neighboring data points. With little to no training time, KNN is a useful tool for "off-the-bat" analysis of data sets. It is versatile, easy to implement and makes no assumptions on the data. An early review of KNN classification is given by Cunningham and Delany [10]. The core issues were: (a) how to determine the appropriate similarity/distance metric, (b) how to determine the user-defined parameter k, (c) how to address computational complexity issues in large datasets. Many works have addressed these issues since then. This survey gives an overview of modern developments

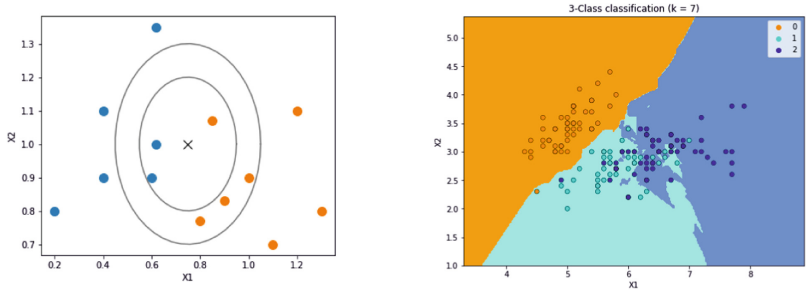
D. E. Simos et al. (Eds.): LION 2022, LNCS 13621, pp. 379–393, 2022.
https://doi.org/10.1007/978-3-031-24866-5_28

with respect to these core issues but also intends to exhibit the utility of KNN in data science in general.

The structure of this paper is as follows: Sects. 2–4 are concerned with the high level concepts of KNN algorithms. Specifically, Sect. 2 is a recap of the challenges related to KNN, and provides descriptions of how certain algorithms tackle these issues. Section 3 is focused on the choice of distance metric, metric learning and the role of feature space transformation. Section 4 describes several recent KNN variations that we find interesting. The rest of the chapters are as follows: Sect. 5 is concerned with prototype selection and generation, and feature selection methods. Section 6 is dedicated to matching issues related to hashing algorithms, partition trees, and graph techniques. A lot of work related to scaling KNN for big data is also presented in Sect. 6. Finally, a synopsis and concluding summary are presented in Sect. 7.

2 Challenges

KNN is a simple and non parametric algorithm (does not make any assumption on the distribution of the training instances). However, proper tuning of the hyper-parameter k is of crucial importance. Figure 1(a) is an example where different parameter values result in different classification outcome. If $k = 3$ the unseen observation (the 'x' mark on the figure) is classified as blue, whereas if $k = 6$ it is classified as orange. Figure 1(b) demonstrates how the decision boundary might look like for a particular choice of k. A comprehensive tutorial is given by Cunningham and Delany [11].



(a) k-nearst neighbors for k=3 and k=6.

(b) Example decision boundary plot.

Fig. 1. In plot (a) the unseen observation is classified as blue for $k = 3$ while it is classified as orange for $k = 6$. Plot (b) exhibits the decision boundary formed in a 3-class example. (Color figure online)

The factors that might affect KNN performance include:

1. **Outliers and Noise:** Outliers have a higher chance of misclassification and noise makes the determination of a suitable k value a more challenging task.

2. **Overlapping class regions:** The region defining a class, or fragments of a class, may overlap with the region of another class. When classes intertwine, the classification decision may be prone to error.
3. **Class imbalance:** Some classes in the data set may contain significantly less observations than other classes. This biases the prediction in favor of the over-represented classes, especially when there is a high degree of overlap.

A recent survey on the challenges related to KNN, together with experimental results, has been given by Zhang in [66]. The current chapter is complementary by offering insightful descriptions of the challenges and of how they are addressed, together with key results regarding the use of KNN algorithms in general.

The weakness of KNN in overlapping regions can be described in terms of the statistical properties of the classes. Consider two Gaussian distributions with different means and variances, and overlapping density functions. The Gaussian with the smaller variance tends to dominate the decision of KNN, as samples tend to form dense clusters in the region around the mean. In this example, Tang and He [49] demonstrate that Bayesian estimation outperforms regular KNN. To tackle this challenge they propose *Extended-NN* (ENN) which looks at data points that consider the unseen example to be in their k-neighborhood. The method iteratively assigns the unseen observation to classes and calculates a class-wise coherence statistic. Total coherence is calculated by the average number of correct inclusion in the k-neighborhoods of data points (of a given class) that consider the unseen observation to be their k-neighbor, and summing over all the classes. The assignment with largest total coherence wins. ENN is shown to outperform regular KNN consistently. Nonetheless, their paper demonstrates that the incorporation of Bayesian estimation may yield promising results.

Class imbalance phenomena have been widely studied in the machine learning literature (for example He and Garcia [20], Fernández et al. [14]). Traditional pre-processing techniques include re-sampling and cost-sensitive learning. Unfortunately, re-sampling techniques can fail to improve the accuracy of KNN due to the usually sparse nature of minority instances in space. As a response, researchers devise KNN specific approaches, many of which revolve around formulating more appropriate decision rules.

Ando proposes in [3] *Weighted Class-wise Nearest Neighbors* (WCNN). He estimates and compares conditional probabilities of class membership given majority or minority class. The estimates are based on the distance of the unseen observation from: (a) the kth nearest majority observation, (b) the kth nearest minority observation. The intuition is that the distance from the kth minority class observation is generally larger. WCNN biases the decision towards the minority class. The decision function is linear in the k-distances and the weights are optimised for non-linear performance loss in a convex setting. Furthermore, by construction, WCNN can work with distance data. To reduce the perplexity of the distance features, WCNN is paired with a hierarchical clustering algorithm that considers class labels. The major sub-components are regarded as supplementary labels to facilitate optimisation. The key advantages of WCNN

are that (a) data-balancing preprocessing algorithms may be avoided, (b) it works directly with pairwise distance data.

Zhang et al. [70] proposed the *K-Rare Nearest Neighbors* (KRNN) where k is variable so as to encompass at least some examples of the under-represented classes, i.e. the total number of observations in a query's neighborhood is variable. Given an unknown instance, the probability of it belonging to a minority class is modeled with a binomial distribution. The confidence interval for the proportion of minority observations in the data set is calculated, and a modified confidence interval is used for the proportion of minority observations in the neighborhood of the query. Additionally, two modified Laplace estimates for the posterior class probability are formed, and the relative position of the confidence intervals determines which Laplace estimate is used in the decision function. The confidence levels for the global and local confidence intervals are hyperparameters of the algorithm. Experiments indicate a performance improvement over WCNN on the datasets used.

When it comes to noise we can distinguish between noise in the attributes and noise in the prediction variable. When attributes are numeric, noise manifests itself in the statistical properties of the classes. For categorical variables, there could be errors in the attributes and/or misclassified data points. All types of noise deteriorate the performance of KNN. Assuming that correctly classified points dominate their respective regions in feature space, sensitivity to noise can be reduced by choosing a larger k value. Early noise reduction techniques involved disregarding misclassified points in the training set. However, the sensitivity of KNN's performance to noise has motivated KNN-based anomaly detection techniques. The general idea is scoring points based on their similarity to their neighbors, for example Bandaragod et al. [7], Pang et al. [38]. We have not come across a paper that considers correlations between attribute values and noise in a KNN framework.

Apart from competitive performance, KNN based anomaly detection algorithms demonstrate a gravity defiant learning curve. The general understanding is that, for most learning algorithms, the error rate decreases with the size of the dataset, i.e. the more data the better. Contrary to the conventional wisdom, anomaly detection techniques based on KNN reach peak performance when the training dataset is small. Ting et al. [50] use a computational geometry argument to find closed form expressions for the lower and upper bounds of the *area under the receiver operating characteristic curve* for the 1-NN anomaly detection scheme. It is shown that the expected bounds depend on both the size of the dataset and the proportion of anomalies, and that the bounds reach an optimal value at a finite dataset size. Experimental results show that other KNN-based anomaly detectors demonstrate the same behavior. This result consolidates the role of KNN methods in data science as it provides a solid theoretical ground for the use of KNN ensembles in anomaly detection, which would be especially useful in the context of big data. More specifically, KNN detectors can work in parallel on subsamples of the dataset, and achieve maximal expected accuracy. Triguero et al. [52] advocate the use of KNN methods as means of creating smart

data out of big data, the main tools being KNN based noise reduction methods, and missing value imputators. Note that noise reduction methods should be applied prior to imputators to reduce bias.

The final issue of this chapter concerns the choice of the hyperparameter k. To the best of our knowledge the most straightforward approach seems to be the best in terms of accuracy. Most notably, García-Pedraja et al. [17] propose the assignment of a k value to each observation in the knowledge base, using ten-fold cross-validation. The values are chosen by considering both the local best k and the global best k in order to avoid large deviations in the values assigned to neighboring points. New instances, then, inherit the k value of their nearest neighbor. Experiments indicate improved accuracy over regular KNN. Approaches based on direct (convex) optimization are also present, and can also improve accuracy. Zhang et al. [67] propose a sparse reconstruction of the dataset from itself through multiplication with a weight matrix. Non-negativity is enforced to ensure that points are reconstructed from points with correlated features and l_1 regularization is added to the cost function to ensure sparsity. The number of non-zero entries indicates the suitable k value for each observation. An extra term is added to ensure that points are reconstructed with points with similar features (i.e. are close in the feature space). Their work is extended in Zhang et al. [68] where a decision tree, named k^*-tree, is constructed. The k^*-tree can be searched efficiently and results in faster query times (by storing relevant points in the leafs) for a small loss of accuracy.

3 Choice of Distance Metric

In general, data instances with d features are considered as points within an d-dimensional feature space. Since the prediction is determined by the nearest neighbors, the nature of KNN is such that the relative position of instances is more significant than their absolute positions. Ideally, the distance metric should minimize the distance between similarly classified instances, while maximizing distance between instances of different classes.

Abu Alfeilat et al. [1] tested a large number of distance metrics on real world datasets and found that the choice of metric significantly affects classification performance. Specifically, there is no optimal distance metric that is suitable for all datasets, and metrics from the same families showed similar classification results. Additionally, some distance metrics were found to be more tolerant to noise than others. It could be the case that some application domains favor certain metrics over others. Hu et al. [23] conducts similar experiments for the medical domain and find that the Chi-square distance function performs the best.

While cross validation is an option for finding a suitable distance metric, another approach is distance metric learning. These methods involve applying a (linear or nonlinear) transformation to the feature space in order to maximise classification accuracy. Xing et al. [63] was the first to propose a convex optimization approach. They considered the Mahalanobis distance defined by

$d_A(x, y) = (x-y)^\top A(x-y)$, where $A = W^\top W$ is a positive semi-definite matrix $A \succeq 0$ (here W corresponds to the space transformation). The intuition is: find the optimal matrix A that minimizes distances between similarly labeled points and maximizes distances between points in different classes. To see this, let $y_{ij} = 1$ if the i'th and j'th points in the knowledge base belong to the same class (i.e. $y_i = y_j$), and zero otherwise. The objective function can be written as:

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j}(1 - y_{ij})\sqrt{d_A(x_i, x_j)}, \\ \text{subject to:} \quad & \sum_{i,j} y_{ij} d_A(x_i, x_j) \leqslant 1, \\ & A \succeq 0. \end{aligned} \tag{1}$$

The objective function aims to maximize the distance between points belonging to different classes while the first condition forces points belonging to the same class to remain close to each other. In a similar approach Shalev-Shwartz et al. [42] learns a Mahalanobis metric online and tries to enforce a scalar threshold b such that points in the same class are at most $b-1$ distance apart while points in different classes are at least $b+1$ distance apart. Both these approaches improve the performance of KNN and have the advantage that the optimization problems are convex. However there is a key observation relevant to KNN that exposes a weakness of linear space transformations. Points of the same class may cluster in different locations in feature space and may exhibit different patterns in their relevant positions with points from other classes. Goldberger et al. [18] tries to circumvent this problem by introducing the novel idea of *Neighborhood Component Analysis* (NCA). NCA randomly assigns neighborhoods to points and maximizes the leave-one-out probability that the KNN label is correct. The probability that point j is in the neighborhood of point i is inversely related to their distance in the transformed space. Inspired by the idea that only the distances of neighbors are relevant in the objective function of Eq. (1), Weinberger et al. [59] formulated a convex optimization problem similar to that of Eq. (1) that sums over neighbors (instead of over all pairs of points) and whose resulting transformation is known as the *Large Margin Nearest Neighbor* (LMNN).

It is clear that the choice of metric is relevant in KNN applications and that the methods of this chapter alleviate the adverse effects of data peculiarities. While linear transformations do improve the accuracy of KNN, their weakness comes from the fact that they are global, yet, the distributions of classes differ in different regions of the feature space. As a result, and by extension, all literature on space transformations is relevant for improving KNN classification accuracy.

4 Variations of KNN

Here we briefly recite a selection of recent KNN methods that we find interesting in dealing with the challenges in Sect. 2. Although this chapter provides a less

than complete account, we believe that the our references include a satisfactory number of readings.

The state of the art seems to revolve around meaningful compositions of existing methods and ideas. The works cited in this paragraph have been used in conjunction with each other to produce better KNN algorithms. Pascal and Yoshua [54] devised a variant of KNN based on k-local hyperplanes. The intuition is that classes lie on non-linear manifolds on the feature space. The core idea is to linearly approximate the manifold locally, and assign unseen observations based on their distance from the (approximate) manifold of each class. Particularly, suppose a set of class labels $\mathcal{C} = \{1, 2, \ldots, N_C\}$ where N_C is the number of classes. For each unseen data point, locate the k nearest neighbors from each class (for a total of kN_C nearest neighbors). Say $\mathcal{N}_{C_i} = \{x_1, x_2, \ldots, x_k\}$ are the k closest points with class label i. The local approximation of the corresponding manifold is a linear combination of the points in $\mathcal{N}_{C_i}$. The coefficients of the linear combinations are optimised to minimize the distance to the unseen observation for each i in $\mathcal{C}$, and the point is assigned to the class of the closest linear combination. Another important "building block" is the family of fuzzy KNN algorithms. We direct the reader to Derrac et al. [12] for a taxonomy and experimental analysis. Meanwhile, several researchers have proposed the use of different k-values for different regions of the feature space, i.e. Wettschereck and Dietterich [60], Wang et al. [56], Garcia et al. [17]. Among others, Garcia et al. [17] proposed a tenfold cross validation scheme for the assignment of different values of k to each point in the knowledge base. Unclassified examples would inherit the k value of their closest neighbor. Furthermore, several researchers adopted weighting schemes to weight the labels of the k-nearest neighbors (usually according to distance), i.e. Dudani [13], Liu and Chawla [32], Gou et al. [19].

Recently, researchers combine two or several of the known approaches, together with their own innovation, and produce algorithms that are less vulnerable to noise and outliers, class imbalance, and class region distribution. Susan and Kumar [47] applied a linear transformation (such as LMNN or NCA, see Sect. 3) to the feature space and adopted a new decision rule. They split the k nearest neighbors in two clusters. One consists of the neighbors that are closest to the farthest neighbor, and the other consists of the neighbors that are closest to the closest neighbor. The latter cluster, then, determines the final classification. The rule is resilient to outliers, noise, and class imbalances because the feature space transformation has expanded distances between instances of different classes. Yu et al. [65] combined k-local hyperplane distances with a fuzzy relative transform decision rule to tackle class imbalances. Zhang et al. [69] employed the locality preserving projection (by He and Niyogi [22]) to reconstruct the test sample from the training sample, resulting in a weighting of nearest neighbors. These are but a few examples of successful combinations of ideas.

5 Feature Selection and Data Reduction

This chapter is concerned with data reduction techniques. These methods revolve around disregarding irrelevant and/or redundant dimensions of a dataset, or,

disregarding unnecessary data points. These methods are valuable in practice as they may result in greater accuracy, reduced runtimes, and reduced memory requirements.

The accuracy of distance based algorithms can be severely degraded with high-dimensional data due to the curse of dimensionality. Feature selection (FS) techniques aim at reducing the dimensionality of the data, thus, improving the accuracy of the learning algorithm. The goal is to identify a small subset of features that maximizes a measure of accuracy. A broad categorization of FS methods includes filter-based methods, wrapper-based methods, embedded methods, and hybrid methods. For more information we refer the reader to Li et al. [30]. Lets consider a general strategy for FS that includes (a) a feature subset selector, (b) a feature subset evaluator. Usually, in wrapper-based methods, a machine learning model is re-trained and tested with the dataset projected on the current feature subset (in order to evaluate its performance). In these cases, KNN offers a significant speedup when the distance function is calculated recursively. For example, the d-dimensional Euclidean distance satisfies:

$$d_E(x, x')^2 = \sum_{i=1}^{d-1} (x_i - x'_i)^2 + (x_d - x'_d)^2.$$

Wang et al. take the aforementioned approach in [55]. Many approaches in the literature select KNN for subset evaluation, e.g. Tahir et al. [48] propose a Tabu search strategy for subset generation. A challenge in the FS literature is instability. In many cases the number of features far exceeds the number of observations. As a result, the features selected by certain algorithms (most notably the random forest approach) highly depend on the initial data sample. Ensemble KNN wrapper methods are believed to be able to tackle instability issues (Li et al. [31], Park and Kim [39]). In the context of filter-based methods, a comparative study by Rogati and Yang [41] showed that KNN methods were amongst the top three performers at year 2001, indicating that KNN can benefit greatly from FS methods.

In a work more intimately related to KNN, Xiao and Chaovalitwongse [62] showed that the FS problem can be cast as a convex optimisation problem if the decision is based on the distance to the centroids of each class. The idea is to learn a Mahalanobis matrix (similarly to what is shown in Sect. 3) and add a l_1 regularization term in the loss function. l_1 regularization promotes sparsity for the Mahalanobis matrix, which effectively nullifies certain dimensions of the data.

Data reduction methods attempt to reduce the number of training instances. The idea is to select, or artificially generate prototypes that faithfully represent the given target concept. Ideas include retaining instances that are close to the decision boundary, or, retaining instances near the centers of the class clusters. Wrapper-based subset search and evaluation strategies are also present. A complete taxonomy, comparison, and extensive experimentation is given in Garcia et al. [16]. A valuable insight is that efficient query methods, such as approximate nearest neighbors (described in the next chapter), can compete in run-time

performance with regular KNN even when the initial dataset is reduced in size. Another observation is that the time complexity of these algorithms were generally $O(n^2d^2)$ or higher, where n is the number of data points and d the number of features, together with high storage requirements. Arnaiz-González et al. [5] propose a linear complexity algorithm using locality sensitivity hashing (LSH). Even though the data reduction rates and the resulting classification accuracy were not among the top performers, it is a solution for extremely large datasets, and their paper provides descriptions of many other data reduction algorithms. Additionally, Triguero et al. [51] provide a distributed solution using the MapReduce framework. Recent publications, among others, include prototype selection for imbalanced datasets: Sisodia and Sisodia [45], prototype selection with local feature weighting: Zhang et al. [71], prototype selection for KNN regression: Song et al. [46].

6 Nearest Neighbor Matching Algorithms

It is important to note that when working with high dimensional features, there is no known exact nearest-neighbor search algorithm with acceptable efficiency. To enhance the speed of queries, most practical applications settle for approximate search. The neighbors returned by approximate search techniques may not be the k-nearest, but they are typically close to the k-nearest neighbors. Regardless of whether KNN is used as a classifier, or as a tool for aforementioned purposes, this chapter aims to show that KNN can be a scalable solution.

When the data set is large, $O(dn)$ query time-complexity renders KNN algorithms intractable for certain applications. Several solutions have been studied. When the number of training instances is large, special data structures (partitioning trees, neighboring graph techniques) can enhance query speeds significantly. The main disadvantage of such methods is how they scale with the dimensionality of the data. Techniques such as hashing counteract this problem.

A simple and popular hashing method is the locality sensitive hashing (LSH), Indyk and Motwani [25]. The idea is to hash the data so that the probability that hashes coincide is much higher for points that are close together. Formally, locality sensitivity is defined by four parameters, (r_1, r_2, p_1, p_2). Given some distance function d, a family of hash functions $\mathcal{H}$ is $(r1, r2, p1, p2)$-sensitive if for any two points p, q in the data set the following conditions are fulfilled:

1. if $p \in B(q, r_1)$, then $\Pr_{h\in\mathcal{H}}[h(p) = h(q)] \geqslant p_1$,

2. if $p \notin B(q, r_2)$, then $\Pr_{h\in\mathcal{H}}[h(p) = h(q)] \leqslant p_2$,

where $B(p, r)$ denotes the hypersphere centered at point p with radius r. For this to be meaningful, it is essential that probabilities p_1, p_2 satisfy $p_1 > p_2$, and $r_1 < r_2$. The gap between p_1 and p_2 can be amplified by concatenating several hash functions. Practically, a collection of hashes split the data points into several partitions. To process a query, brute force search is applied to elements of the partitions with corresponding hash values. LSH is grounded in the theory

of random projections. In the simplest case, points are projected on random lines passing through the origin. These lines are then discretized into small line segments each with a corresponding id. Thus, a table of hash codes is created, each entry having pointers to the corresponding data points. This allows for the creation of data structures that can be searched efficiently. A generalization named *Density Densitive Hashing* (DSH) also exploits the distribution of the data [28].

Hashing methods result in fast and effective queries. Query times achieved are sublinear. In general, the quality of the hash functions determine the quality of the method (for details see Muja and Lowe [36]). In fact, the space partitions produced by random projections had been widely studied in the average case. In the last decade a lot of work has been dedicated in studying worst case scenarios. To illustrate what is meant by average and worst case scenario, consider the random projection method described in the previous paragraph. If a dataset consists of points sparsely distributed around the origin, the probability of collision of far away points is small. On the contrary, if there is a dense cluster of points in the dataset (and away from the origin), these points are likely to collide with far away points. For this reason, data-dependent hashing methods have been developed. These methods aim for data optimal (approximate) nearest neighbors search, for details see He et al. [21], Xu et al. [64], Iwamura et al. [26], Andoni and Razenshteyn [4]. In the recent years, the literature on hashing methods has grown and we direct the reader to the survey by Wang et al. [57].

In the family of partitioning trees, the *kd*-tree (Friedman et al. [15]), has been one of the best known neighbor matching algorithms with logarithmic time complexity, but it scales poorly with the number of dimensions (and in fact is comparable to exhaustive search when the number of dimensions is high, see Indyk [24]). Several authors have improved *kd*-trees in order to speed up KNN search. To name a few, Beygelzimer et al. [8] proposed cover-trees, Silpa-Anan and Hartley [44] proposed optimised *kd*-trees. Nister and Stewenius [37] proposed the vocabulary tree, which uses hierarchical k-means. In a comparison by Muja and Lowe [35] it was shown that the multiple randomized trees are the most effective for high dimensional data.

Jegou et al. [27] proposed a product quantization approach in which the feature space is decomposed into low dimensional subspaces in which the data points are represented by compact codes. Babenko and Lempitsky [6] proposed the inverted multi-index, obtained by replacing the standard quantization with product quantization. A more in-depth analysis is given in review papers by Vasuki and Vanathi [53], Wu and Yu [61].

Nearest neighbor graph methods build graphs where vertices are data points or subsets of data points. In this case the query is an effective exploration of the graph. Empirical results place graph methods in the current state of the art for query methods. We direct the reader to recent surveys by Wang et al. [58] and Shimomura et al. [43].

Finally, a lot of work has been done for decentralized framework solutions. Chatzimilioudis et al. [9] developed Spitfire, a high performance distributed algorithm. Gieseke et al. [40] presented a GPU based algorithm for *kd*-trees. Kim et al. [29] propose parallel KNN using MapReduce. Maillo et al. [34] also provided a solution for exact k-nearest neighbor classification based on Spark.

7 Synopsis and Concluding Remarks

We have studied a variety of aspects related to KNN. Challenges related to: (a) noise and outliers, (b) overlapping class regions, and (c) class imbalanced data have been analyzed through key works in the literature. The role of KNN in data pre-processing, including de-noising and missing value imputations has been presented. The choice of hyperparameter k has also been analyzed. The impact of the distance metric used has been explored, together with metric-learning techniques. The merit of KNN in feature selection algorithms has been discussed, together with the effect of such methods on KNN's accuracy. We further explored data reduction techniques. Moreover, query methods, together with mentions of parallel and distributed solutions have been mentioned.

We would like to point out here that KNN is a non parametric, instance based algorithm that makes no assumptions about the underlying data distribution. This trait is crucial due to the fact that real world data rarely obey typical theoretical assumptions. KNN is easy to implement and its basic principle is easy to understand. Some challenges regarding the choice of the right distance metric have been overcome with data driven approaches like metric learning. Localization of the k-value, and different decision rules (subspace distances, fuzzy criteria, feature weighting etc.) have also increased classification accuracy.

The state-of-the-art seems to revolve around the meaningful composition of ideas that make the algorithm more resilient to class imbalances, noise, and outliers. The two major drawbacks are the storage requirements and the query run-time complexity. Methods of prototype selection and generation can reduce the number of training instances required without hindering classification accuracy greatly. Methods of feature selection can reduce the dimensionality of the data by discarding irrelevant/redundant features. KNN based methods have been developed for noise reduction, outlier detection, and missing value imputation among other applications. Theoretical results indicate that KNN anomaly detectors demonstrate a gravity defiant learning curve. These issues enable the usage of KNN anomaly detectors ensembles in the context of large datasets.

Approximate nearest neighbor methods significantly reduce query run-times, allowing for applications on data intensive domains. Hashing, quantization, neighboring graph techniques, and indexing methods all contribute to the diverse literature that enables the use of KNN in large data domains. GPU based distributed algorithms and adaptations for data streams are a testament to KNN's utility as a classification method or as part of larger machine learning models.

References

1. Alfeilat, H.A., et al.: Effects of distance measure choice on K-nearest neighbor classifier performance: a review. Big Data, **7** (2019)
2. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. (1991)
3. Ando, S.: Classifying imbalanced data in distance-based feature space. Knowl. Inf. Syst. **46** (2016)
4. Andoni, A., Razenshteyn, I.: Optimal data-dependent hashing for approximate near neighbors. In: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2015, pp. 793–801. Association for Computing Machinery, New York, NY (2015). ISBN 9781450335362
5. Arnaiz-González, Á., Díez-Pastor, J.-F., Rodríguez, J.J., García-Osorio, C.: Instance selection of linear complexity for big data. Knowl.-Based Syst. **107**, 83–95 (2016)
6. Babenko, A., Lempitsky, V.: The inverted multi-index. IEEE Trans. Pattern Anal. Mach. Intell. **37**(6), 1247–1260 (2014)
7. Bandaragoda, T.R., Ting, K.M., Albrecht, D., Liu, F.T., Wells, J.R.: Efficient anomaly detection by isolation using nearest neighbour ensemble. In: 2014 IEEE International Conference on Data Mining Workshop, pp. 698–705. IEEE (2014)
8. Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 97–104 (2006)
9. Chatzimilioudis, G., Costa, C., Zeinalipour-Yazti, D., Lee, W.-C., Pitoura, E.: Distributed in-memory processing of all k nearest neighbor queries. IEEE Trans. Knowl. Data Eng. **28**(4), 925–938 (2015)
10. Cunningham, P., Delany, S.: k-nearest neighbour classifiers. Mult Classif. Syst. **54**, 04 (2007)
11. Cunningham, P., Delany, S.J.: k-nearest neighbour classifiers - a tutorial. ACM Comput. Surv. (CSUR) **54**(6), 1–25 (2021)
12. Derrac, J., García, S., Herrera, F.: Fuzzy nearest neighbor algorithms: taxonomy, experimental analysis and prospects. Inf. Sci. **260**, 98–119 (2014)
13. Dudani, S.A.: The distance-weighted k-nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. **SMC-6**(4), 325–327 (1976)
14. Fernández, A., del Río, S., Chawla, N.V., Herrera, F.: An insight into imbalanced big data classification: outcomes and challenges. Complex Intell. Syst. **3**(2), 105–120 (2017)
15. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. (TOMS) **3**(3), 209–226 (1977)
16. Garcia, S., Derrac, J., Cano, J., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012)
17. García-Pedrajas, N., Romero del Castillo, J.A., Cerruela-García, G.: A proposal for local k values for k -nearest neighbor rule. IEEE Trans. Neural Netw. Learn. Syst. **28**(2), 470–475 (2017)
18. Goldberger, J., Hinton, G.E., Roweis, S., Salakhutdinov, R.R.: Neighbourhood components analysis. In: Saul, L., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17. MIT Press, Cambridge (2004)

19. Gou, J., Du, L., Zhang, Y., Xiong, T.: A new distance-weighted k-nearest neighbor classifier. J. Inf. Comput. Sci. **9**, 1429–1436 (2012)
20. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. (2009)
21. He, J., Liu, W., Chang, S.-F.: Scalable similarity search with optimized kernel hashing. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1129–1138 (2010)
22. He, X., Niyogi, P.: Locality preserving projections. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge (2003)
23. Hu, L.-Y., Huang, M.-W., Ke, S.-W., Tsai, C.-F.: The distance function effect on k-nearest neighbor classification for medical datasets. Springerplus **5**(1), 1–9 (2016). https://doi.org/10.1186/s40064-016-2941-7
24. Indyk, P.: Nearest neighbors in high-dimensional spaces (2004)
25. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Conference Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 604–613, October 2000
26. Iwamura, M., Sato, T., Kise, K.: What is the most efficient way to select nearest neighbor candidates for fast approximate nearest neighbor search? In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3535–3542 (2013)
27. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2010)
28. Jin, Z., Li, C., Lin, Y., Cai, D.: Density sensitive hashing. IEEE Trans. Cybern. **44**(8), 1362–1371 (2013)
29. Kim, W., Kim, Y., Shim, K.: Parallel computation of k-nearest neighbor joins using mapreduce. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 696–705. IEEE (2016)
30. Li, J., et al.: Feature selection: a data perspective. ACM Comput. Surv. (CSUR) **50**(6), 1–45 (2017)
31. Li, S., Harner, E.J., Adjeroh, D.A.: Random KNN feature selection - a fast and stable alternative to random forests. BMC Bioinform. **12**(1), 1–11 (2011)
32. Liu, W., Chawla, S.: Class confidence weighted kNN algorithms for imbalanced data sets. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 6635, pp. 345–356. Springer, Heidelberg (2011). ISBN 978-3-642-20847-8. https://doi.org/10.1007/978-3-642-20847-8_29
33. Loizou, G., Maybank, S.J.: The nearest neighbor and the Bayes error rates. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-9**(2), 254–262 (1987)
34. Maillo, J., Ramírez, S., Triguero, I., Herrera, F.: KNN-IS: an iterative spark-based design of the k-nearest neighbors classifier for big data. Knowl.-Based Syst. **117**, 3–15 (2017)
35. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1), 2 (331–340), 2 (2009)
36. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2227–2240 (2014)
37. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), vol. 2, pp. 2161–2168. IEEE (2006)
38. Pang, G., Ting, K.M., Albrecht, D.: LeSiNN: detecting anomalies by identifying least similar nearest neighbours. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 623–630. IEEE (2015)

39. Park, C.H., Kim, S.B.: Sequential random k-nearest neighbor feature selection for high-dimensional data. Expert Syst. Appl. **42**(5), 2336–2342 (2015)
40. Patwary, M.M.A., et al.: Panda: extreme scale parallel k-nearest neighbor on distributed architectures. In: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 494–503. IEEE (2016)
41. Rogati, M., Yang, Y.: High-performing feature selection for text classification. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management, pp. 659–661 (2002)
42. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004, pp. 94. Association for Computing Machinery, New York (2004)
43. Shimomura, L.C., Oyamada, R.S., Vieira, M.R., Kaster, D.S.: A survey on graph-based methods for similarity searches in metric spaces. Inf. Syst. **95**, 101507 (2021)
44. Silpa-Anan, C., Hartley, R.: Optimised KD-trees for fast image descriptor matching. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
45. Sisodia, D., Sisodia, D.S.: Quad division prototype selection-based k-nearest neighbor classifier for click fraud detection from highly skewed user click dataset. Int. J. Eng. Sci. Technol. **28**, 101011 (2022)
46. Song, Y., Liang, J., Lu, J., Zhao, X.: An efficient instance selection algorithm for k nearest neighbor regression. Neurocomputing **251**, 26–34 (2017)
47. Susan, S., Kumar, A.: DST-ML-EkNN: data space transformation with metric learning and elite k-nearest neighbor cluster formation for classification of imbalanced datasets. In: Chiplunkar, N.N., Fukao, T. (eds.) Advances in Artificial Intelligence and Data Engineering. AISC, vol. 1133, pp. 319–328. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-3514-7_26
48. Tahir, M.A., Bouridane, A., Kurugollu, F.: Simultaneous feature selection and feature weighting using hybrid Tabu search/K-nearest neighbor classifier. Pattern Recogn. Lett. **28**(4), 438–446 (2007)
49. Tang, B., He, H.: ENN: extended nearest neighbor method for pattern recognition [research frontier]. IEEE Comput. Intell. Mag. **10**(3), 52–60 (2015)
50. Ting, K.M., Washio, T., Wells, J.R., Aryal, S.: Defying the gravity of learning curve: a characteristic of nearest neighbour anomaly detectors. Mach. Learn. **106**(1), 55–91 (2017)
51. Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F.: MRPR: a mapreduce solution for prototype reduction in big data classification. Neurocomputing **150**, 331–345 (2015)
52. Triguero, I., García-Gil, D., Maillo, J., Luengo, J., García, S., Herrera, F.: Transforming big data into smart data: an insight on the use of the k-nearest neighbors algorithm to obtain quality data. WIREs Data Min. Knowl. Discov. **9**(2) (2019)
53. Vasuki, A., Vanathi, P.: A review of vector quantization techniques. IEEE Potentials **25**(4), 39–47 (2006)
54. Vincent, P., Bengio, Y.: K-local hyperplane and convex distance nearest neighbor algorithms. In: Dietterich, T., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Information Processing Systems, vol. 14. MIT Press, Cambridge (2001)
55. Wang, A., An, N., Chen, G., Li, L., Alterovitz, G.: Accelerating wrapper-based feature selection with k-nearest-neighbor. Knowl.-Based Syst. **83**, 81–91 (2015)
56. Wang, J., Neskovic, P., Cooper, L.N.: Neighborhood size selection in the k-nearest-neighbor rule using statistical confidence. Pattern Recogn. **39**(3), 417–423 (2006)

57. Wang, J., Zhang, T., Song, J., Sebe, N., Shen, H.T.: A survey on learning to hash. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 769–790 (2018)
58. Wang, M., Xu, X., Yue, Q., Wang, Y.: A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. arXiv preprint arXiv:2101.12631 (2021)
59. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification, January 2005
60. Wettschereck, D., Dietterich, T.: Locally adaptive nearest neighbor algorithms. In: Cowan, J., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems, vol. 6. Morgan-Kaufmann, Burlington (1993)
61. Wu, Z., Yu, J.: Vector quantization: a review. Front. Inf. Technol. Electron. Eng. **20**(4), 507–524 (2019). https://doi.org/10.1631/FITEE.1700833
62. Xiao, C., Chaovalitwongse, W.A.: Optimization models for feature selection of decomposed nearest neighbor. IEEE Trans. Syst. Man Cybern. Syst. **46**(2), 177–184 (2016)
63. Xing, E., Jordan, M., Russell, S.J., Ng, A.: Distance metric learning with application to clustering with side-information. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, vol. 15. MIT Press, Cambridge (2002)
64. Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., Yu, N.: Complementary hashing for approximate nearest neighbor search. In: 2011 International Conference on Computer Vision, pp. 1631–1638 (2011)
65. Yu, Z., Chen, H., Liu, J., You, J., Leung, H., Han, G.: Hybrid k-nearest neighbor classifier. IEEE Trans. Cybern. **46**(6), 1263–1275 (2016)
66. Zhang, S.: Challenges in KNN classification. IEEE Trans. Knowl. Data Eng. 1 (2021)
67. Zhang, S., Li, X., Zong, M., Zhu, X., Cheng, D.: Learning k for KNN classification. ACM Trans. Intell. Syst. Technol. (TIST) **8**(3), 1–19 (2017)
68. Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R.: Efficient KNN classification with different numbers of nearest neighbors. IEEE Trans. Neural Netw. Learn. Syst. **29**(5), 1774–1785 (2017)
69. Zhang, S., Cheng, D., Deng, Z., Zong, M., Deng, X.: A novel KNN algorithm with data-driven k parameter computation. Pattern Recogn. Lett. **109**, 44–54 (2018). Special Issue on Pattern Discovery Multi-Source Data (PDMSD)
70. Zhang, X., Li, Y., Kotagiri, R., Wu, L., Tari, Z., Cheriet, M.: KRNN: k rare-class nearest neighbour classification. Pattern Recogn. **62**, 33–44 (2017)
71. Zhang, X., Xiao, H., Gao, R., Zhang, H., Wang, Y.: K-nearest neighbors rule combining prototype selection and local feature weighting for classification. Knowl.-Based Syst. **243**, 108451 (2022)