Extensions of the k Nearest Neighbour Methods for Classification Problems

Zacharias Voulgaris and George D. Magoulas
University of London
School of Computer Science and Information Systems, Birkbeck College
United Kingdom
{zacharias, gmagoulas}@dcs.bbk.ac.uk

ABSTRACT
The k Nearest Neighbour (kNN) method is a widely used technique which has found several applications in clustering and classification. In this paper, we focus on classification problems and we propose modifications of the nearest neighbour method that exploit information from the structure of a dataset. The results of our experiments using datasets from the UCI repository demonstrate that the classifiers produced perform generally better than the classic kNN and are more reliable, without being significantly slower.

KEY WORDS
Nearest neighbour classifier, kNN, classification, pattern recognition, discernibility.

1. Introduction

The k Nearest Neighbour (kNN) is one of the most commonly used methods for pattern recognition [1], and has been applied in a variety of cases [2, 3, 4]. Its simplicity and relatively high convergence speed make it a popular choice. However, in some applications, it may fail to produce adequate results [5], whilst in others its operation may render impractical [1]. Yet, the fact that it has only one parameter, the number of neighbours used (k), makes it easy to fine-tune to a variety of situations. Its main process consists of the following steps: given a set of N points (training set), whose class labels are known, classify a set of n points (testing set) into the same set of classes by examining the k closest points around each point of the testing set and by applying the majority vote scheme.

This process has a few inherent problems, which is why researchers have proposed different extensions of the kNN [1, 2, 3, 4, 5, 6, 7, 8], or even ensemble formulations of kNN classifiers [9]. Most of these approaches have exhibited some interesting and quite promising results and have motivated further research on improving the kNN method.

The contribution of this paper lies in proposing variants of kNN that are eminently suitable for classification problems, as they exploit information inherent in the data sets, such as the dataset structure. These proposed methods are based on new concepts, which are described briefly in this paper, and their potential is numerically confirmed through experiments using datasets of the UCI repository. The paper also proposes and evaluates measures for assessing the performance of the proposed methods.

The rest of the paper is structured as follows. The next section reviews some of the pertinent literature on k-NN extensions, pinpointing their most promising ideas and their drawbacks. Section 2 also introduces some concepts for evaluating the classifiers. In section 3, our proposed variations of kNN are presented and discussed. The results of experiments carried out on these classifiers, as well as on kNN, are presented in section 4. Lastly, in section 5 the findings on the methods' performance are discussed and conclusions stemming from all this research are presented.

2. Background

Although the kNN method when used in classification problems is quite fast, it is often impeded by the size of certain datasets, which is why some researchers have focused on improving its speed. SMART-TV for instance [2] was designed to deal with datasets of high dimensionality by transforming them into a single-dimensional feature space. A similar approach is shared in [3] for spatial data, where this method works best [2, 6]. Yet, these approaches concentrate on high speed mainly and often fail to achieve exceptionally good accuracy rate unless they are applied on particular problems, such as spatial datasets [3].

Other approaches involve feature selection methods [5]. These methods appear to be promising and are partly similar to one of kNN variations we propose later in this paper. Yet, the methods described in the literature are not

always very fast, since it appears to be a trade-off between performance and speed [5].

Changing the way distance is dealt with, in a fundamental level, is an interesting alternative which can improve speed considerably, without significant reductions in accuracy rate [1]. When dealing with complex problems this can be quite fruitful [4], yet these methods appear to be rather cumbersome when applied to other simpler datasets.

Often it is more efficient to combine different classifiers, either by forming a low-level mixture [7] or by building an ensemble [9]. In the first case, it becomes apparent that changes in the structure of the kNN may be essential in order to improve its performance. This idea has triggered our interest in developing new types of kNN-based classifiers, like the ones described later on. In the second case, a creation of uncorrelated classifier is attempted, as negatively correlated classifiers in an ensemble seem to improve the accuracy rate of the whole. Yet, the results although interesting, denote that kNN-based approaches still require much improvement if they are to be used in ensembles to target various classes of problems.

The use of rules in kNN has been researched in [8], where rules have been used as additional attributes with some success. However, in many datasets the creation of rules may be time-consuming and even computationally expensive. Also, in datasets of high dimensionality, the additional cost could make the classification very slow and therefore inefficient.

Another method encountered in the literature questions the efficiency of the voting scheme of kNN [10], and proposes an alternative measure for determining how each class is related to a test point. This approach is taken one step further in some of the kNN variations proposed in this paper, since the use of only one measure (distance) to assess the relationship to a class is often insufficient.

Of the methods described above, [10] goes beyond the simple counting of neighbours by evaluating them as well. We share this philosophy and in this paper we bring forward some kNN-based alternatives, which either assign a quality index to each element of the dataset, or a different k value. Also, similarly to [5], we introduce a classifier that makes use of different weights for the various features of the dataset. However, this is done in a fast and quite efficient way.

Since the methods we propose in this paper might be used in ensemble formulations we discuss below two measures that are used later on in our experiments to estimate the performance of the classifiers produced and can provide insight on the performance of these methods in ensemble formulations. The first one yields a sign of how "certain" the classifiers are for each classification performed and is called *degree of certainty*, and the second one measures how related the certainty of the classification is with the correctness of it, and is called *net reliability.*

2.1 Degree of Certainty

The Degree of Certainty is a generalisation of the Certainty Factor (CF), which according to [11] is defined for a type of classifier as in Eq. 3.1:

$$CF_i = \frac{final\ vote(i)}{\sum_{c=1}^{\#\ of\ classes} final\ vote(c)}, \qquad (2.1)$$

where i denotes the i-th classified and c the class number.

By substituting *final vote(c)* for the classification score of class, and *final vote(i)* for *max(final vote)*, we obtain the Degree of Certainty (DC), an index of assuredness which is compatible with all types of classifiers (Eq. 3.2). This measure yields information about how confident a classifier is for a particular classification and is in the form of a vector.

$$DC_i = \frac{\max(classification\ score)}{\sum_{c=1}^{\#\ of\ classes} classification\ score(c)}, \qquad (2.2)$$

where i denotes the i-th pattern classified, c the class number and *classification_score* is the score determining the classification output of the classifier.

2.2 Net Reliability

Net Reliability (NR) is a measure, primarily developed for evaluating how trustworthy the Degree of Certainty of a classifier is. This is because it has been observed that there are cases where a classifier has high DC for a classification which later proves to be wrong, while a low DC is yielded for a classification which is later found to be correct. In other words, it is similar to a correlation measure between Accuracy and DC. It takes values in the interval [-1, 1] and is calculated by Eq. (3.3). Apparently, the higher the Net Reliability of classifier is, the better for the classifier (usually anything positive is good).

$$NR = \frac{1}{n}\sum_{i=1}^{n}\left[(2v_i - 1)\cdot DCy_i\right], \qquad (2.3)$$

where v is the classification validity vector (a binary vector depicting the correct classifications as 1 and the wrong ones as 0), *Dcy* the Degree of Certainty vector, i denotes the pattern classified, and n is the total number of elements in the test set.

Since *NR* greatly depends on the classifier as well as on the dataset, it is often useful to calculate it every time and

to regard it as a significant measure of its performance, if we are to make use of the *DC* of the classifier.

3. Classification Approaches Based on kNN

3.1 The Density Based kNN Classifier – DB-kNN

As the mere counting of the neighbours appears to be insufficient for determining the class of a test element [10], we altered the kNN classifier in a way that another factor would be taken into account, namely that of density. In our work this is defined by the name Structural Density as it reveals information about the structure of a dataset and is inspired from physics.

Structural Density (SD) is defined as the number of points in the neighbourhood of an element over the volume of this neighbourhood. The parameter involved is that of the radius (r) defining the neighbourhood, which is determined as follows. First, we calculate the density of all the elements as a function of r and the average density of the whole set (as the total number of elements over the total volume of the dataset, something irrelevant to r). We then search for a value of r such that the mean of the individual densities is equal to the average density calculated earlier.

The Density Based kNN (DB-kNN) classifier has been created by taking into account the structural density concept for evaluating the significance of each neighbour, along with the distances. Initially the densities of all the elements are computed for each one of the classes of the dataset. Then, they are normalised to [0, 1] since the relative densities appear to have more meaning than the absolute ones. Based on these densities, each neighbouring element is assessed regarding its role as "core" element of its class by measuring its relative structural density with regard to the class. By dividing it by its Euclidean distance a score is obtained for each neighbour. By taking a biased mean (which is influenced more by the larger numbers) of these scores for each one of the classes, we end up with q voting scores, where q is the number of classes. From these final scores the classification of each test point is derived, as well as the Degree of Certainty. In the not so common case that the *DC* of the classification of a particular test element is below 0.667 (i.e. the classification is considered unreliable), the classic kNN method is applied for that instance.

The DB-kNN provides a new look at the use of the neighbours in the kNN, because it explores the potential of evaluating them rather than merely counting them, which, in our view, is an important step forward. Also, the distance is used, making the evaluation of the neighbours more refined.

3.2. The Variable k Nearest Neighbour Classifier – V-kNN

Since the value of the k parameter often influences the classification results, sometimes significantly, we devised another classification algorithm that overcomes this issue. By making use of the *DC* concept, it estimates the optimum k for each classification.

The Variable kNN (V-kNN) classifier works as follows. First for each one of the training set elements a classification of it is performed based on various neighbourhoods. The k value that maximises the *DC* of each classification is found. Therefore, for each training set there corresponds a particular k value which is considered the best available. Afterwards, for each unknown element, the nearest neighbour is found and its k value is assumed (based on the "optimum" k array). Then, the kNN classifier is applied on that test element, using that k value. As a concept, this is something similar to one of the ideas presented in [3].

It is noteworthy that apart from its performance as a classifier, the V-kNN approach yields some useful information about the dataset: the average optimum k value, which for kNN-type classifiers is very useful to know as it improves their performance, particularly for the classical kNN classifier. However, for very sparse datasets the optimum k found may not be valid and the results may not be better than those of kNN.

3.3. The Weighted kNN Classifier – W-kNN

Similarly to the Density Based kNN classifier, the Weighted kNN (W-kNN) performs an evaluation but this time on the features instead of on the patterns. Each feature is evaluated and assigned a weight based on how useful this feature is for discerning the classes of the dataset. To do this a new concept is introduced here, namely that of the Index of Discernibility.

The Index of Discernibility (ID) is a measure developed for assessing how easily distinguishable the classes of a dataset are. Originally we discerned classes using boxes containing them, but this appeared to be non-sensitive to the class structures and not computationally effective. In this version of the Index of Discernibility we make use of (hyper)spheres; this attempt assumes a fixed radius around each element of the dataset, which corresponds to the average distance between this and the rest of the elements of that class. Note that the radius depends on the class structure, so elements belonging to different classes may have different radii. Once the radius of an element is established, elements of the same class as the examined element that belong to its (hyper)sphere are identified and counted. The discernibility of that element is calculated by dividing the number of these elements by the number of total elements in the (hyper)sphere (see Figure 1). The Index of Discernibility of the whole dataset is calculated

as the number of elements having discernibility higher than 0.5 divided by the total number of elements.

The Index of Discernibility is also used for evaluating individual features by applying it on single dimensions of the dataset.

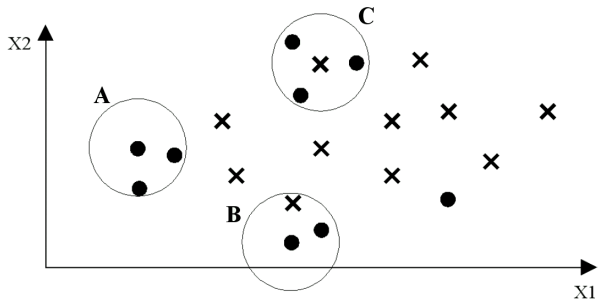

Figure 1 – Illustration of the Index of Discernibility for a simple dataset with only two features, X1 and X2. In this example, the discernibility of the element in the centre of circle A is ID1 = 2 / 2 = 1, that of the element in the centre of circle B is ID2 = 1 / 2 = 0.5, while that of the element in the centre of circle C is ID3 = 0 / 3 = 0.

The W-kNN classifier works as follows. First each one of the features of the training set are evaluated using the ID. The weights are then obtained by normalising the IDs. Lastly, the weights are applied on both the training and the testing set and the kNN classifier operates on the now transformed dataset.

3.4. The Class Based kNN Classifier – CB-kNN

The Class Based kNN (CB-kNN) is somewhat different as a kNN extension. It was developed because often the datasets are unbalanced as regards their class structure, so it may be the case that one class has too few elements to "win" the vote of a classification of the kNN classifier.

The CB-kNN algorithm deals with these datasets working in the following way. For every test element, the *k* nearest elements of each class are taken. The value of *k* is automatically selected by the classifier, so as to maximise the *DC* of the classification. Afterwards, the harmonic mean of the distances of these neighbours is calculated (so that it is not influenced so much by the most distant elements). Finally, these means are compared and the class yielding the lowest value is chosen for the classification.

3.5. The Discernibility kNN Classifier – D-kNN

Inspired by the W-kNN method, we decided to make use of the discernibility concept once again. This time we devised an algorithm similar in structure to the original kNN extension of the DB-kNN. The aim was to make an algorithm that is quite fast, without losing in accuracy.

Similarly to DB-kNN, the Discernibility kNN (D-kNN) takes into account the distance of each neighbour. Yet, instead of using the structural density it takes the discernibility of each element (through which the Index of Discernibility is computed in other cases). By dividing the discernibility by the distance, a score is produced, for each one of the neighbours. Then, the scores for each class are averaged to produce one classification score for each one of the classes. The class yielding the highest classification score is selected for the classification.

4. Experimental Results

4.1 Description of the Experiments

The experiments carried out included 50 rounds of 10-fold cross-validation (500 classifications altogether, for each one of the classifiers). They were performed on 6 datasets obtained from the UCI repository [12]. The characteristics of these datasets can be seen in Table 1. The classifiers that required a *k* value used the one shown at the rightmost column of the table.

Dataset Characteristics			
Name	**Attributes**	**Patterns**	**K***
Bupa Liver	6	345	7
Pima Indians	8	768	6
Breast Cancer W.	9	683	2
Heart Disease	13	270	6
Vehicle	18	846	5
Boston Housing	13	506	6

Table 1 – Characteristics of the datasets used in the experiments. K* denotes the "optimum" *k* calculated by the Variable *k* Nearest Neighbour algorithm.

4.2 Results

The performance of the kNN-based classifiers was evaluated using the Accuracy Rate and the Net Reliability (discussed in section 2.2). Also, the average CPU time of the training part of the classification was taken into account. For each one of the datasets a series of experiments took place and based on the above criteria the winner, i.e. the best available classifier, was found. These results are shown in Table 2.

Dataset	**Accuracy Rate**	**Net Reliability**	**CPU Time (2nd) in sec.**
Bupa Liver	D-kNN (66.31%)	D-kNN (0.2572)	W-kNN (0.0226)
Pima Indians	V-kNN (74.55%)	V-kNN (0.4707)	W-kNN (0.1020)
Breast Cancer W.	CB-kNN (96.93%)	V-kNN (0.9285)	W-kNN (0.0920)
Heart Disease	D-kNN (81.31%)	D-kNN (0.5926)	W-kNN (0.0332)
Vehicle	CB-kNN (70.89%)	D-kNN (0.4009)	V-kNN (0.1230)
Boston Housing	CB-kNN (69.03%)	W-kNN (0.3587)	W-kNN (0.0789)

Table 2 – Winners based on average performance over the 50 rounds. The winning performance metric is shown inside brackets. The Net Reliability is calculated by means of the classification and the Degree of Certainty vectors at the end of each experiment.

Afterwards, a one-to-one comparison was made for each pair of classifiers, showing how many times (rounds) one

classifier outperformed the other. Then these scores were added up for each classifier. The final sum revealed the relative performance of each classifier, shown in Table 3.

Dataset	Accuracy Rate	Net Reliability	CPU Time (2^{nd}) in sec.
Bupa Liver	D-kNN (232)	D-kNN (241)	W-kNN (200)
Pima Indians	V-kNN (219)	V-kNN (250)	V-kNN (193)
Breast Cancer W.	V-kNN (230)	V-kNN (237)	CB-kNN (200)
Heart Disease	D-kNN (193)	D-kNN (226)	V-kNN (200)
Vehicle	CB-kNN (218)	D-kNN (231)	V-kNN (200)
Boston Housing	W-kNN (210) & CB-kNN (209)	W-kNN (214)	V-kNN (200)

Table 3 – Winners based on the relative performance, pairwise, over 50 rounds. The numbers in brackets show the sum of the times the winning classifier was better than the others in terms of a particular performance measure, for each dataset.

In is noteworthy that in all of the six datasets, kNN was outperformed by one of the variations introduced in this paper. Also, the only criterion where it actually performed well was speed, since it required no training.

5. Conclusions and Discussion

In this paper, an investigation on extensions of the kNN method was conducted. Our approach which uses properties of the dataset led to the development of effective modifications of the kNN method for classification problems. The proposed kNN variants were tested in classification problems from the UCI repository.

Based on the experiments carried out, the DB-kNN method worked much slower than kNN due to the structural density calculation. However, its performance was generally better than that of kNN.

The V-kNN classifier was very fast (often the fastest of all) and generally performed better than kNN.

The W-kNN method was exceptionally fast, since the operations required for computing and applying the weights are very simple and without any complications. Its performance is generally better than that of kNN and the CPU time required is minimal for various values of *k*.

With regards to the CB-kNN classifier, the large number of computations involved in the process makes the overall CPU time naturally longer than that of the other classifiers. Yet, it outperforms kNN significantly, as it was the most accurate classifier in three of the datasets.

The D-kNN classifier achieved its aim to a great extent as it was quite fast (though not the fastest), while at the same time it generally outperformed kNN and many of the other kNN extensions. Moreover, it proved to be quite reliable (in terms of Net Reliability).

Summing up, all of the introduced classifiers seem to have a good performance overall, though most of them excel in one or two datasets. Also, even though they are all slower than kNN since they require a training phase, they are generally quite reliable in terms of Net Reliability. Our future work involves further testing at a larger scale to fully investigate the advantages of the proposed methods and identify their limitations.

Acknowledgements

The authors would like to thank Prof. Boris Mirkin for his valuable suggestions during the initial stages of this paper, as well as the insightful supervision and guidance he provided in the experiments involved in this work.

References

[1] Moreno-Seco, F., Mico, L. and Oncina, J. A Modification of the LAESA Algorithm for Approximated k-NN Classification. *Pattern Recognition Letters 24* (2003), pp. 47–53

[2] Abidin, T. and Perrizo, W. SMART-TV: A Fast and Scalable Nearest Neighbor Based Classifier for Data Mining. *Proceedings of ACM SAC-06*, Dijon, France, April 23-27, 2006. ACM Press, New York, NY, pp. 536-540

[3] Khan, M., Ding, Q. and Perrizo, W. K-Nearest Neighbors Classification of Spatial Data Streams using P-trees. *Proceedings of the PAKDD*, 2002, pp. 517-528

[4] Yu, K. and Ji, L. Karyotyping of Comparative Genomic Hybridization Human Metaphases Using Kernel Nearest-Neighbor Algorithm, *Cytometry*, 48, 202–208, 2002.

[5] Sotoca J.M., Sanchez J.S., and Pla F, Estimating Feature Weights for Distance-Based Classification. *Pattern Recognition in Information Systems PRIS2003*, Angers (France), pp.156-166, Ed. ICEIS PRESS, ISBN: 972-98816-3-4, 2003

[6] Mainar-Ruiz, G. and Juan Carlos Pérez- Cortes Approximate Nearest Neighbor Search Using a Single Space-filling Curve and Multiple Representations of the Data Points. *Proc. 18th International Conference on Pattern Recognition (ICPR 2006)*, 20-24 August 2006, Hong Kong, China: 502-505

[7] Hendrickx, I. and Antal van den Bosch. Maximum-Entropy Parameter Estimation for the k-nn Modified Value-Difference Kernel. *Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence*, Groningen, The Netherlands, 2004

[8] Antal van den Bosch Feature Transformation Through Rule Induction: A Case Study with the k-NN Classifier. In J. Fürnkrantz (Ed.), *Proceedings of the ECML/PKDD 2004 Workshop on Advances in Inductive Rule Learning*, Pisa, Italy, September 2004, pp. 1-16

[9] Domeniconi, C., and Yan, B. On Error Correlation and Accuracy of Nearest Neighbor Ensemble Classifiers.

Proceedings of the SIAM International Conference on Data Mining, Newport Beach, California, April 21-23, 2005

[10] Wang, H. and Bell, D. Extended k-Nearest Neighbours Based on Evidence Theory. *The Computer Journal*, Vol. 47 (6) Nov. 2004, pp. 662-672.

[11] Aydin, T. and Guvenir, H. A. Modeling Interestingness of Streaming Classification Rules as a Classification Problem. *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, ISSN 1611-349 (Online), Volume 3949, 2006

[12] UCI Machine Learning Repository, available on line at the University of California, Irvine http://www.ics.uci.edu/~mlearn/MLSum mary.html