Int. J. Mach. Learn. & Cyber. manuscript No.
(will be inserted by the editor)

Hubness-based fuzzy measures for high-dimensional *k*-nearest neighbor classification

Nenad Tomašev · Miloš Radovanović · Dunja Mladenić · Mirjana Ivanović

Received: date / Accepted: date

Abstract Most data of interest today in data-mining applications is complex and is usually represented by many different features. Such high-dimensional data is by its very nature often quite difficult to handle by conventional machine-learning algorithms. This is considered to be an aspect of the well known *curse of dimensionality*. Consequently, high-dimensional data needs to be processed with care, which is why the design of machine-learning algorithms needs to take these factors into account. Furthermore, it was observed that some of the arising high-dimensional properties could in fact be exploited in improving overall algorithm design. One such phenomenon, related to nearest-neighbor learning methods, is known as *hubness* and refers to the emergence of very influential nodes (hubs) in *k*-nearest neighbor graphs. A crisp weighted voting scheme for the *k*-nearest neighbor classifier has recently been proposed which exploits this notion. In this paper we go a step further by embracing the *soft* approach, and propose several fuzzy measures for *k*-nearest neighbor classification, all based on hubness, which express fuzziness of elements appearing in *k*-neighborhoods of other points. Experimental evaluation on real data from the UCI repository and the image domain suggests that the fuzzy approach provides a useful measure of confidence in the predicted labels, resulting in improvement over the crisp weighted method, as well as the standard *k*NN classifier.

This is an extended version of the paper *Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification*, which was presented at the MLDM 2011 conference [27].

N. Tomašev and D. Mladenić
Institute Jožef Stefan, Artificial Intelligence Laboratory
Jožef Stefan International Postgraduate School
Jamova 39, 1000 Ljubljana, Slovenia
E-mail: nenad.tomasev@ijs.si, dunja.mladenic@ijs.si

M. Radovanović and M. Ivanović
University of Novi Sad, Department of Mathematics and Informatics
Trg D. Obradovića 4, 21000 Novi Sad, Serbia
E-mail: radacha@dmi.uns.ac.rs, mira@dmi.uns.ac.rs

1 Introduction

High-dimensional data is ubiquitous in modern applications. It arises naturally when dealing with text, images, audio, data streams, medical records, etc. The impact of this high dimensionality is manyfold. It is a well known fact that many machine-learning algorithms are plagued by what is usually termed the *curse of dimensionality*. This comprises a set of properties that tend to become more pronounced as the dimensionality of data increases. First and foremost is the unavoidable sparsity of data. In high-dimensional spaces all data is sparse, meaning that there is not enough data to make reliable density estimates. Another detrimental influence comes from the concentration of distances, as all data points tend to become relatively more similar to each other as dimensionality increases. Such a decrease of contrast makes distinguishing between relevant and irrelevant points in queries much more difficult. This phenomenon has been thoroughly explored in the past [1, 10]. Usually, it only holds for data drawn from the same underlying probability distribution. Mixed data is not so severely affected [11], but the effects are still more pronounced than in the lower-dimensional scenarios. The difficulties arising from the influence of high dimensionality on distance measures even led some researchers to question the very notion of a point's nearest neighbor in high-dimensional feature spaces [9].

Regardless of the theoretical considerations above, methods based on nearest neighbors remain in very frequent use throughout various data-mining tasks, such as classification, clustering and information retrieval. This is hardly surprising, given the simplicity of the notion of nearest neighbors: the inferences about the current example are based on the

most similar previously observed points. It is somewhat disheartening to see that even such simple ideas can be, at certain times, severely compromised by the dimensionality curse.

1.1 The hubness phenomenon

In this paper, we will focus only on one phenomenon of interest for nearest-neighbor methods operating in many dimensions. This phenomenon is known as *hubness*. The term was coined after *hubs*, very influential points which arise in high-dimensional spaces. Their influence is measured by the frequency with which they occur as neighbors to other points in the data. In a sense, they are very frequently 'consulted' during inference. If they are not compromised by noise and also contain accurate class-affiliation information, they exhibit a highly beneficial influence and all is well. If, on the other hand, there were some errors in feature values or the attached labels, such points would exhibit a highly detrimental influence and are known as *bad hubs* [16,17,18]. Of course, real-world data is often noisy and not entirely reliable, whether it had been gathered automatically by sensors or input by human operators. Both ways of data acquisition are somewhat uncertain. Consequently, bad hubs are not an uncommon occurrence in practice.

There is more to hubness than just a few frequent neighbors. Denote by $N_k(x)$ the number of k-occurrences of x, i.e., the number of times x appears in k-nearest neighbor lists of other points in the data. The entire distribution of $N_k(x)$ becomes affected and an increasing skewness is usually observed. What this means is that most points very rarely occur as neighbors. Therefore, most of the time when we examine a queried k-neighbor set, it will contain some of the hub-points in the data. We will address these issues in more detail in Section 3. We should point out that hubness is a consequence of high *intrinsic* dimensionality of data (regardless of the nominal number of features in the chosen representation). It is a general property which stems from how the geometry of high-dimensional spaces affects the probability of each point being a k-neighbor (i.e., being among the k closest points to some other point in the data). More specifically, most data sets (approximately) appear as hyperspheres or unions of hyperspheres centered around some distribution means. This positioning renders points closer to the data centers more likely to be included in k-nearest neighbor lists of other data points. This tendency increases with dimensionality.

Hubness was first observed in music retrieval, when some songs were observed as being fetched very frequently in the queries and were determined not to be relevant on average, i.e. the calculated similarity in the feature spaces failed to capture the semantic similarity perceived by people [2, 3]. Even though we mentioned bad hubs as sometimes being caused by noise and errors in the data, it is not entirely so. Many data contain overlapping probability distributions, therefore bad hubness can arise even in error-free data sets. It is equally dangerous in both cases, so the underlying mechanics of hubness will not be given special attention in this paper, as it is a separate topic. What we will do is provide solutions to such cases when bad hubness does appear, as it can not always be avoided.

One simple solution to the problem has already recently been proposed, in form of a weighting scheme for the voting in the k-nearest neighbor (kNN) algorithm [17,19]. We will have a closer look at that weighting in Section 2.1, while we outline the motivation for our fuzzy approach.

Our idea is to extend the class-nonspecific crisp kNN weighting scheme described in [17] to class-specific soft voting in the spirit of the fuzzy k-nearest neighbor (FNN) algorithm [13]. Introducing fuzziness is not only expected to enrich the classification by refining the confidence measures behind each decision but also often improves the overall accuracy. This makes it worth considering.

Other than in classification and retrieval [26], hubness has also been addressed in other data-mining tasks, as for example clustering [28], anomaly detection [24], object recognition in images [23] and instance selection (data reduction) [5].

The fact that hubness is among the most important aspects of the dimensionality curse in nearest-neighbor methods suggests that it certainly needs to be taken into account while designing new approaches. This is why we think that the hubness-aware design of the fuzziness measures for data labels in k-nearest neighbor classification might be advantageous and that is precisely what we will explore in this paper.

The rest of the paper is structured as follows. In Section 2 we present the related work, focused around two major points – the hubness-weighted kNN algorithm, and the FNN algorithm. While observing the former, we outline its weak points and aim our proposed improvements in their direction. The respective hubness-based fuzzy membership functions are presented in Section 3. We go on to evaluate the proposed approach in Section 4. Finally, we give our final remarks and future research directions in Section 5.

2 Related work

2.1 Hubness-weighted kNN

Weighted voting in nearest-neighbor classifiers has become something of a common practice. Weights are usually either based on element position in the k-neighbor list or its distance to the observed data point. Some more robust approaches which take into account the correlation between

these factors have also been recently developed [34]. The hubness-weighting scheme which was first proposed in [16] is a bit more flexible, in a way that the weight associated to x_i is $w(x_i,k)$, meaning that each point in the training set has a unique associated weight, with which it votes whenever it appears in some k-neighbor list, regardless of its position in the list.

This weighting is based on the interpretation of how the hubness phenomenon affects kNN performance. As was mentioned before, hubness of an element x_i is the number of its k-occurrences in neighbor lists of other elements, and is denoted by $N_k(x_i)$. This can be decomposed into two parts: $N_k(x_i) = GN_k(x_i) + BN_k(x_i)$, where $GN_k(x_i)$ is the number of *good* k-occurrences and $BN_k(x_i)$ is the number of *bad* k-occurrences. Good occurrences are those when the label of x_i matches the label of the element in whose k-neighbor list x_i is observed. Bad occurrences are characterized by a mismatch of labels. Elements with high bad hubness are often found in neighbor lists of elements belonging to other categories in the data. This means that bad hubs exhibit a detrimental influence on k-nearest neighbor classification, because their vote often gives misleading information. Fig. 1 illustrates this point in a simple binary classification scenario. The aforementioned weighting scheme reduces these bad influences directly. Standardized bad hubness is defined as $h_b(x_i,k) = (BN_k(x_i) - \mu_{BN_k})/\sigma_{BN_k}$, where μ_{BN_k} is the mean bad hubness and σ_{BN_k} the standard deviation. The two parameters of the bad occurrence distribution are simply estimated from the training set as $\mu_{BN_k} = \frac{1}{N}\sum_{x_i \in D} BN_k(x_i)$ and $\sigma_{BN_k} = \sqrt{\frac{1}{N}\sum_{x_i \in D}(BN_k(x_i) - \mu_{BN_k})^2}$. The weight associated to x_i is then $w(x_i,k) = e^{-h_b(x_i,k)}$. It was shown that this often leads to significant improvements in high-dimensional settings where hubs naturally appear as an artefact of dimensionality. The amount of improvement depends on the distribution of bad hubness within the data.

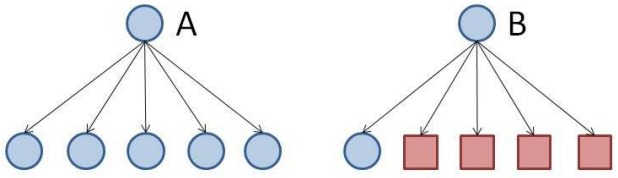


Fig. 1 An illustrative binary classification example. The instances of the two classes are depicted as circles and squares. An arrow indicates a nearest-neighbor relation, so that if it points from x_1 to x_2 this means that x_1 is a neighbor of x_2. We see that the two focal points, A and B, have the same overall hubness, $N_1(A) = N_1(B) = 5$. The nature of the influence of the two points is, on the other hand, entirely different. Point A seems to be a neighbor only to the points that share its own label, so we can be quite confident in its future votes. Point B behaves in quite the opposite way, and we can not be confident in its future votes. This is why the hubness-based weighting scheme is useful.

What the described approach disregards completely is the structure of bad hubness. In non-binary classification, when a label mismatch occurs, it can occur in any of the class neighborhoods. Instead of observing $N_k(x_i)$ as a sum of good and bad hubness, we could decompose it with respect to individual classes into $N_k(x_i) = \sum_{c=1}^{n_c} N_{k,c}(x_i)$, where each $N_{k,c}(x_i)$ is the number of k-occurrences of x_i in neighborhoods of elements of class c, and n_c is the total number of classes. Good hubness is just the special case when $c = y_i$, y_i being the label of x_i in the data set. Therefore, instead of using the hubness information only to reduce the votes of bad hubs, it is possible to take into account the structure of bad hubness, which can be used to decompose the crisp vote given by x_i into a fuzzy vote relying on all $N_{k,c}(x_i)$. There already exists a framework that can assist in achieving this goal, referred to as the fuzzy nearest-neighbor classifier.

2.2 Fuzzy nearest neighbor algorithm

Fuzzy sets are based on a notion of inherent ambiguity in the data, meaning that a single element can be viewed as partially belonging to several different categories at the same time [31]. This ambiguity is often problem-specific and the set membership function is then provided by the domain experts. However, there are also ways of deducing some sort of fuzziness automatically from the data. Denote by $u_{ci} = u_c(x_i)$ the degree of membership of x_i in class c. The following properties must hold in order for u_c to define a fuzzy split on the data set:

$$\sum_{c=1}^{n_c} u_{ci} = 1,$$
$$0 < \sum_{i=1}^{n} u_{ci} < n,$$
$$u_{ci} \in [0,1].$$

The second and the third condition might seem equivalent, but in fact they are not, due to the strict inequality in the second condition, which essentially guarantees that each class is non-empty. As for the first condition, that all class memberships for a given data point should sum up to 1, it is merely a convenience of scaling and u_c has been defined in such a way in previous work on using fuzzy labels in k-nearest neighbor classification [13]. We will, therefore, not argue with this definition, even though it is certainly possible to work with fuzzy measures where the first condition does not hold.

Let x be a newly observed data instance for which we wish to perform classification. Let $D_k(x) = x_1,..x_k$ be its k

nearest neighbors. The degree of membership of x in each class c is then defined as

$$u_c(x) = \frac{\sum_{i=1}^{k} u_{ci}(\|x - x_i\|^{-(2/(m-1))})}{\sum_{i=1}^{k} (\|x - x_i\|^{-(2/(m-1))})}, \quad (1)$$

where $\|\cdot\|$ denotes the Euclidean norm. The parameter m in Eq. 1 determines how heavily the distance is weighted when calculating contributions from each neighbor. For large values of m, neighbors are weighted more equally, while low values of m favor closer neighbors. The most commonly used default value for this parameter is $m = 2$, so that fuzzy votes are weighted by the reciprocal of the distance.

There exist many ways for automatically generating suitable fuzzy measures from the data. This is not only used for class membership fuzziness, but also for fuzzifying attributes. A range of techniques can be used, including genetic algorithms, clustering, neural networks, entropy, and others [8]. In the original fuzzy nearest-neighbor article [13], some simple ways to achieve this were also proposed, one of which was to observe k nearest neighbors of x_i and count the percentages of them coming from any particular class. The final measure was a linear combination of the element's label and these percentages, normalized so as to fall in the desired $[0,1]$ range.

Apart from applying the fuzzy approach to specific domains [6,12,20,21,30], most attention has been given lately to the issues of scalability in terms of achieving speedup in fuzzy nearest-neighbor search [4,33], as well as improving the weighting scheme [15].

3 Proposed hubness-based fuzzy measures

The basis of our motivation was already mentioned in Section 2.1 while discussing the properties of hubness-weighted kNN. Instead of using *good* and *bad* hubness, we propose to use *class hubness* $N_{k,c}(x_i)$ defined uniquely for each element in the training set. It is immediately apparent that this measure can be fit into the fuzzy nearest-neighbor framework. Contrary to the more usual fuzzy measures, it does not represent inherent fuzziness of an element's label, but instead measures the fuzziness of an *appearance* of elements in k-neighbor sets, based on the training data. Regardless of the semantic difference between the two, their form remains the same.

There are, however, some difficulties with using hubness as a fuzzy measure. For small values of k, there are many elements in the data which have zero hubness. This becomes even more pronounced in high dimensions due to the mentioned skew of the distribution of k-occurrences. Also, in non-binary classification problems, we need even more hubness information in order to be able to properly estimate the partial memberships for all the existing categories. This poses a serious limit on using class hubness for calculating fuzziness. We would be forced to use very high k values, which could be detrimental in cases when best kNN classification is achieved for smaller neighborhood sizes, as is often the case for non-noisy small or medium-sized data sets.

We propose to handle the problems outlined above by only using hubness of the elements which exhibit hubness greater than some predefined threshold. This in fact separates the data for which it is possible to make reliable fuzzy estimates from those that exhibit hubness too low to be of any use in such a manner. For the data below the threshold, we propose to use a different fuzzy estimate. We explore four such approaches and discuss the pros and cons of their use in the rest of this section, as well as analyze the fruitfulness of their application in Section 4 when presenting the results of the experimental evaluation. Let X be the training set and Y the set of corresponding labels. The hybrid fuzzy measure which we will be considering in the rest of the paper takes the following form:

$$u_c(x_i) = \begin{cases} p_k(y = c|x_i) \approx \frac{N_{k,c}(x_i)+\lambda}{N_k(x_i)+n_c\lambda}, & \text{if } N_k(x_i) > \theta, \\ f_k(c, x_i), & \text{if } N_k(x_i) < \theta. \end{cases}$$

The term $p_k(y = c|x_i)$ denotes the conditional probability of element x being of class c if element x_i appears in its k-neighbor set. For elements which exhibit hubness above a certain threshold, this can be estimated by dividing the class hubness by total hubness. The λ factor is a Laplace estimator, which is used for smoothing to prevent any probability from being estimated as zero. By observing the formula for the conditional probability, one can notice that the label y_i of x_i is not used at all when casting the vote of x_i! This is indeed a very peculiar property. Even though it is possible to work with fuzziness defined in such a way, we wanted to make the fuzziness also dependent on the element's label, so we included each x_i in its own neighbor list at the 0th position. For high-hubness elements, this does not make a large difference, but by doing so we implicitly express a certain degree of confidence in label y_i.

The value of $f_k(c, x_i)$ for low-hubness elements should, ideally, represent a kind of estimate of the actual conditional probability. Since this is not easy to achieve, alternative nearest-neighbor based fuzzy estimates pose themselves as viable alternatives.

It should be noted that representing the neighbor occurrence fuzziness strictly in form of conditional probabilities is not entirely necessary. Fuzzy measures are in general not meant to be interpreted as probabilities. They are used to model uncertainty and are simply more adequate for modeling certain types of uncertainty than the probability theory [22][29]. In the context of neighbor occurrence models, this would imply that one can more easily extend the fuzzy

framework, for instance by assigning different weights to different individual occurrences. The class specific weighted hubness then becomes $N_{k,c}(x_i) = \sum_{x:x_i \in D_k(x)} w_k(x,x_i)$ and the total weighted occurrence sum $N_k(x_i) = \sum_{c \in n_c} N_{k,c}(x_i)$. The weighting can be performed based on the distance between the neighbor points, as was recently demonstrated [25], but is not limited to that. Such weighted occurrence models are genuinely 'fuzzy', as they no longer try to estimate the $p_k(y = c|x_i)$.

We focused on four different ways of dealing with low hubness: a crisp estimate method, a global estimate method, as well as two different local estimates.

- What we refer to as the *crisp estimate* (CE) is the simplest and least flexible way of handling low hubness, which is not in itself necessarily bad – to use the element's own label. In this scenario, low-hubness elements vote the same way they would vote in kNN, with no attached fuzziness. Smoothing is performed by using the same λ value as before.
- *Global estimate* (GE) is more flexible, but introduces the risk of adding more fuzziness than necessary. We compute the GE of the conditional probability as defined in Eq. 2. The denominator represents the summation of $\sum_{(x,y)\in(X,Y)|y=y_i} \sum_{c=1}^{n_c} N_{k,c}(x)$. This is a sensible approach, but it remains questionable just how much is lost and how much is gained by employing it. Even though it does give a global conditional probability of elements from a particular class being included in neighbor sets of another class, there is no guarantee that locally, in the observed part of the data set, this estimate holds.

$$f_k(c,x_i) = \frac{\lambda + \sum_{(x,y)\in(X,Y)|y=y_i} N_{k,c}(x)}{n_c\lambda + \sum_{(x,y)\in(X,Y)|y=y_i} N_k(x)} \tag{2}$$

- If the global estimate fails to capture the proportions contained in the underlying conditional probability for a specific data instance, using a local fuzziness estimate is a possible alternative. Since we already have the k-neighbor lists, it seems natural to take advantage of this when trying to estimate an element's fuzziness. Here we depart from trying to estimate the actual conditional probability and experiment with a more usual approach. Let $\{x_{i1} \dots x_{ik}\}$ be the k nearest neighbors of x_i and for convenience denote x_i also as x_{i0}, since we insert each element into its neighbor list at the 0th position. The *local estimate* (LE_1) is then given by Eq. 3, where $\delta_{cy_{ij}}$ is Kronecker's delta function ($\delta_{cy_{ij}} = 1$ if $c = y_{ij}$ and 0 otherwise). This way, the $f_k(c,x_i)$ is defined as the proportion of examples from class c in the vicinity of the observed point, somewhat smoothed (λ). In a sense, it is a class density estimate. It is not entirely clear which value of k would work best in practice, as this depends on the local structure of the data. In our experiments we used a default neighborhood size of $k = 10$ when calculating the local estimate.

$$f_k(c,x_i) = \frac{\lambda + \sum_{j=0}^{k} \delta_{cy_{ij}}}{n_c\lambda + k + 1} \tag{3}$$

- There is an alternative way to define local fuzziness based on nearest neighbors and this was in fact one of the methods from the original FNN paper [13]. It is based on LE_1, but made so as to emphasize the label of an element, as in the CE method. In fact, it represents a linear combination of the two approaches. We will denote it LE_2, as defined in the following equation:

$$f_k(c,x_i) = \begin{cases} 0.51 + 0.49 \cdot \frac{\lambda + \sum_{j=1}^{k} \delta_{cy_{ij}}}{n_c\lambda + k + 1}, & \text{if } c = y_i, \\ 0.49 \cdot \frac{\lambda + \sum_{j=1}^{k} \delta_{cy_{ij}}}{n_c\lambda + k + 1}, & \text{if } c \neq y_i. \end{cases}$$

The factor of 0.51 was used for the label information simply to guarantee that $f_k(y_i,x_i) > f_k(y_j,x_i)$ for $i \neq j$. Any other $\alpha \in (0,1)$ could have in principle been used instead, whereas any $0.5 < \alpha < 1$ would have ensured that the majority of information comes from the label. This makes the LE_2 anti-hub estimate somewhat less fuzzy, but that is not necessarily a bad thing, as the primary goal is to ensure good classification accuracy.

Apart from testing these fuzzy measures separately, we have also merged them into a single hybrid hubness-based fuzzy nearest-neighbor algorithm which we present in Algorithm 1. Given the training data set, we use the leave-one-out procedure to try classifying each point x from the training data by observing the remaining $n-1$ elements. Such a classification is attempted for each element and for all the k values in a given range, as well as different threshold values and different $f_k(c,x_i)$. The configuration leading to the highest accuracy on the training data is then selected for use on the test set.

The time complexity of this approach is in fact completely comparable to the one of hubness-weighted kNN, with the bottleneck being the computation of k-neighbor sets. Fast approximate algorithms for calculating all k-neighbor sets do exist, one of the most recent being the one presented by Chen et al. [7]. This approximate algorithm runs in $\Theta(dn^{(1+\tau)})$ time, where $\tau \in (0,1]$ is a parameter used to set a trade-off between speed and accuracy. This makes hubness-based algorithms potentially feasible for use on large-scale data sets. We will present our initial results on the scalability of the proposed approach in Section 4.3.

We tested two versions of the algorithm shown in Algorithm 1. The first version uses the distance-based fuzzy vote weighting described in Eq. 1, which we denote by *dwh-FNN*. As an alternative we also tested a version of the algorithm where no distance-based weighting is performed, and fuzzy voting is achieved simply by summing all the respective u_{ci} for every class. This will be referred to as *h-FNN*

Algorithm 1 Hubness-based Fuzzy Nearest Neighbor: Training

```
int[][] NNs = computeNearestNeighborLists(k_min, k_max);
float[][][] classHubnessAllK = computeElementToClassHubness(NNs);
float[][][] GEAllK = computeGlobalEstimates(NNs);
float[][] LE_1 = computeLE1(NNs);
float[][] LE_2 = computeLE2(NNs);
float[][] CE = computeCE(NNs);
float maxAcc = 0;
int bestK, bestTheta;
for all θ = θ_min; θ ≤ θ_max; θ++ do
    for all k = k_min; k ≤ k_max; k++ do
        float GEAcc, LE1Acc, LE2Acc, CEAcc = 0;
        for all i = 0; i < trainingData.length; i++ do
            if votebyGE(x_i, GEAllK, ClassHubnessAllK, NNs) == x_i.label then
                GEAcc++;
            end if
            if votebyLE1(x_i, LE_1, ClassHubnessAllK, NNs) == x_i.label then
                LE1Acc++;
            end if
            if votebyLE2(x_i, LE_2, ClassHubnessAllK, NNs) == x_i.label then
                LE2Acc++;
            end if
            if votebyCE(x_i, CE, ClassHubnessAllK, NNs) == x_i.label then
                CEAcc++;
            end if
        end for
        updateMaxAccAndBestConfiguration(GEAcc, LE1Acc, LE2Acc, CEAcc);
    end for
end for
return The best parameter configuration and all the hubness estimates
```

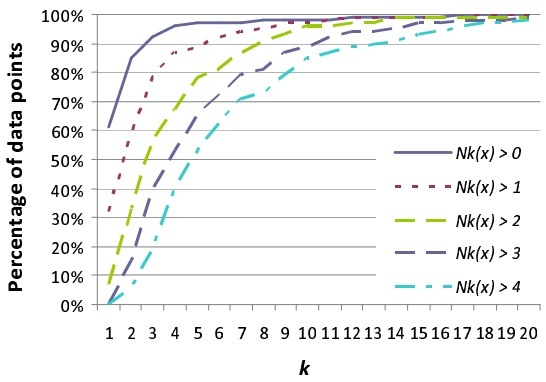
in the rest of the text. The parameter m from Eq. 1 was set to 2 by default, this being the value which is most frequently used.

4 Experimental evaluation

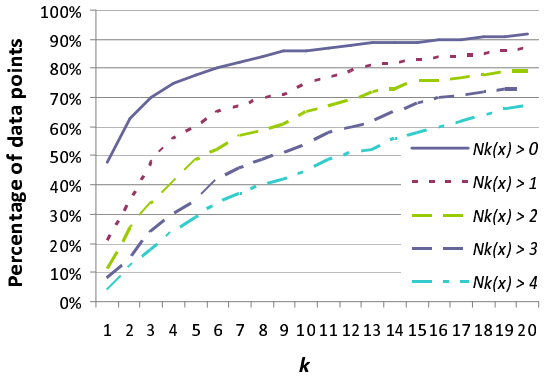
This section presents the results of experiments that compare the standard k-nearest neighbor classifier and the hubness-weighted kNN with the two proposed hubness-based fuzzy approaches h-FNN and dwh-FNN. Section 4.1 deals with data sets of various dimensionalities from the established UCI repository, while Section 4.2 focuses on high-dimensional data from the image domain.

4.1 UCI data sets

Hubness-based fuzzy measures that we proposed are of a hybrid nature, since in each case they combine two different estimates. In order to see how different estimates might be applied, we calculated on each data set, for a range of neighborhood sizes, the percentage of data points which have hubness below/above a given threshold. For two of the used data sets, the plots of several lower thresholds for hubness can be seen in Fig. 2. Naturally, great variation of behavior can be observed across different data sets, since it is related to the aforementioned skew of the hubness distribution in high dimensions. In other words, we expect for highly skewed data sets the term $f_k(c,x_i)$ to play a more important role than in the case of low to medium-skewed data with respect to hubness. It is precisely for these data sets that the mentioned



(a) Iris data set



(b) Dexter data set

Fig. 2 Percentage of elements with hubness exceeding a certain threshold, for neighborhood sizes $k \in \{1..20\}$

estimates of actual hubness may become as important as hubness itself. From Fig. 2, however, the difference becomes quite clear. For the less skewed data sets, if a good classification can be achieved for a neighborhood size of $k \in [10,20]$ or above, then there is probably enough hubness information to allow for its straightforward use as a fuzzy measure. If, on the other hand, the nature of the data is such that the best results are obtained for low k values, ranging maybe from 1 to 5, the situation is reversed. When dealing with highly skewed data, such as in the case of the Dexter data set, influence of $f_k(c,x_i)$ is non-negligible even when considering higher k values.

The first round of testing was performed on 15 data sets taken from the UCI data repository. The used data sets are of various sizes and dimensionalities, and are summarized in Table 1, with the first six columns denoting data-set name, size, dimensionality (d), number of classes (n_c), and the observed skewness of the distributions of N_1 and N_{10} (S_{N_1}, $S_{N_{10}}$).[1] For each data set, the skew of the distribution of k-

[1] Skewness, the standardized 3rd moment of a probability distribution, is 0 if the distribution is symmetrical, while positive (negative) values indicate skew to the right (left).

Table 1 Summary of UCI datasets

Data set	size	d	n_c	S_{N_1}	$S_{N_{10}}$
colonTumor	62	2000	2	1.04	1.06
dexter	300	20000	2	2.95	3.33
diabetes	768	8	2	0.73	0.15
ecoli	336	7	8	0.62	0.37
glass	214	9	6	0.58	0.23
ionosphere	351	34	2	2.17	1.71
iris	150	4	3	0.46	0.03
isolet-1	1560	617	26	1.30	1.20
mfeat-fourrier	2000	76	10	1.20	0.75
ozone-eighthr	2534	72	2	1.31	0.70
page-blocks	5473	10	5	0.79	0.11
parkinsons	195	22	2	0.39	−0.19
segment	2310	19	7	0.70	0.16
vehicle	846	18	4	0.92	0.44
yeast	1484	8	10	0.78	0.27

Table 2 Classification accuracy of *k*NN, hubness-weighted *k*NN (hw-*k*NN), h-FNN and dwh-FNN on UCI data sets. The symbols ○/● denote statistically significant better/worse performance than dwh-FNN

Data set	*k*NN	hw-*k*NN	h-FNN	dwh-FNN
colonTumor	65.1±19.6 ●	72.5±20.6	74.9±20.0	74.5±20.0
dexter	60.1±18.2 ●	72.5± 7.9 ○	68.6± 8.3	68.5± 8.3
diabetes	76.5± 4.1 ○	72.0± 4.6 ●	74.2± 4.9	74.2± 4.9
ecoli	85.4± 6.0	84.5± 6.4	83.6± 6.4	84.3± 6.3
glass	70.5± 9.3 ○	67.6±10.0 ○	65.4± 9.9 ○	63.8±10.0
ionosphere	89.7± 5.2	87.5± 5.7 ●	89.9± 5.5	90.0± 5.6
iris	96.9± 4.0 ○	95.3± 4.8	95.1± 4.7	94.7± 4.8
isolet-1	90.0± 2.6 ○	81.3± 3.4 ●	81.2± 3.8 ●	82.3± 3.6
mfeat-fourier	77.5± 2.9 ●	80.3± 2.6 ●	81.0± 2.6 ●	81.9± 2.6
ozone-eighthr	76.8± 2.5 ●	93.4± 1.8	93.4± 1.3	93.6± 1.3
page-blocks	93.5± 1.0 ●	96.0± 0.8	96.1± 0.8	96.2± 0.8
parkinsons	82.7± 7.7 ●	92.1± 5.8	92.5± 5.2	92.7± 5.2
segment	89.9± 1.7 ●	91.2± 1.7	90.8± 1.8 ●	91.2± 1.8
vehicle	60.7± 5.7 ●	66.6± 5.1	64.4± 4.9	65.2± 5.6
yeast	59.0± 4.1 ○	52.3± 4.1 ●	55.1± 3.8	55.5± 3.8
Average	78.29	80.34	80.41	80.57

occurrences was calculated for various k values, to indicate the degree of hubness of the data. Euclidean distance was used in all the experiments.

On the described UCI data sets, *k*NN, hubness-weighted *k*NN, h-FNN and dwh-FNN were tested. In all the algorithm tests, 10 runs of 10-fold cross-validation were performed. All algorithm parameters were set automatically, separately on each fold during the training phase, based on the training set. Neighborhood sizes were tested in the range $k \in [1,20]$ and thresholds $\theta \in [0,10]$. Classification accuracies achieved by the classifiers are given in Table 2. The corrected resampled t-test [14] was used to test for statistical significance of differences in accuracy for each data set. Differences which were found to be significant with $p < 0.01$ compared to dwh-FNN are denoted by symbols ○/● in the table.

Table 3 Pairwise comparison of classifiers on UCI data: number of wins (with the statistically significant ones in parenthesis)

	*k*NN	hw-*k*NN	h-FNN	dwh-FNN
*k*NN	–	8 (8)	9 (8)	9 (8)
hw-*k*NN	7 (6)	–	9 (4)	10 (5)
h-FNN	6 (6)	6 (2)	–	11 (3)
dwh-FNN	6 (5)	5 (2)	4 (1)	–

The dwh-FNN classifier was selected as the baseline for statistical comparison in Table 2 since we determined that it generally outperformed all other classifiers. To provide a more detailed pairwise classifier comparison, Table 3 shows the number of wins of classifiers signified by the column label, over classifiers denoted by the row labels, with statistically significant wins given in parenthesis.

Overall improvement over *k*NN is apparent already from the shown average scores over all data sets in Table 2, as well as Table 3. Particular improvements vary and there do exist data sets for which none can be observed, as well as some where performance degradation is present. Hubness-weighted *k*NN, h-FNN and dwh-FNN exhibit similar improvement patterns, which makes sense given that they aim at exploiting the same phenomenon. Improvement over the standard *k*NN classifier signifies that there is a lot of usable bad-hubness information in the data. Fuzzy approaches appear to offer additional improvement over hw-*k*NN, justifying our approach and the need to differentiate between classes when employing bad hubness for nearest-neighbor classification. The cases where standard *k*NN is significantly better than hubness-based approaches most probably stem from the difficulties of estimating $p_k(y = c|x_i)$, which requires more data in the case of non-binary classification, as well as $f_k(c,x_i)$ occasionally being an inappropriate substitute in cases of low hubness.

It appears that the distance-based weighting from Eq. 1 does not bring drastic overall improvement to the hubness-based fuzzy membership functions that are used in the h-FNN algorithm, at least not for the default value of the m parameter. This is not all that surprising, though. As was stated in previous discussion, the semantics of hubness-based fuzziness differs slightly from that of more usual fuzzy measures. This is due to the fact that class hubness marks the fuzziness of the elementary event that point x_i appears in a k-neighbor set of an element of some specific category. This hubness is estimated by previous appearances of that element in k-neighbor sets of various other elements in the training data. Among these occurrences, x_i may be located at either place within each observed k-neighbor set. In other words, hubness is a measure which is for a fixed k independent of which positions in k-neighbor sets an element takes. If these lists

were to undergo a random permutation, the hubness for that fixed neighborhood size would have remained unchanged.

Let us assume that we wish to determine the label of a new example x by using h-FNN. The contribution of those x_i closer to x stems not only from previous events when they were also close to the observed element, but also from previous events when they were much farther away. The same holds for farther elements in the k-neighbor set. This is why a linear combination of class hubness contributions is sufficient and any additional distance-based weighting seems superfluous. On the other hand, due to the fact that we can not calculate proper class-hubness probabilities for low-hubness elements, this is only partially true. In cases where fuzziness is estimated for low-hubness x_i, distance-based weighting might bring some improvement by emphasizing more important votes. In practice, most k-neighbor sets will probably contain a mixture of these cases.

Initial comparisons between the different hubness-based fuzzy membership functions proposed in Section 3 were also performed. Experiments were rerun without automatic parameter selection on the folds, so that the algorithms were trained once for every combination of $k \in [1,20]$ and $\theta \in [0,4]$, for every proposed fuzzy scheme. We extracted the parameter values from the range where the algorithms achieved highest accuracy scores, based again on the 10 times 10-fold cross-validation procedure, for every data set. Averages of k values for which the best results were obtained are shown for every used fuzzy scheme in Fig. 3. For each fuzzy approach, lower k values were selected on average if no distance-based vote weighting was performed. This suggests that if the distance weighting is performed, more neighbors are required to convey the same amount of information, due to some votes being downgraded. Different measures attain their best scores at different k-values, as suggested by the observed frequencies. In particular, the global hubness-based fuzziness (GE) finds its maximum at lower k-values than other measures. It is a useful property, as less time is required to perform all the computations when k is smaller. However, the average best accuracies for all the approaches were basically the same. This suggests that hubness itself is still the most important part of the hybrid fuzziness and that anti-hubs can be handled in any of the proposed ways, without significantly affecting the overall performance, at least in medium hubness data (UCI). We will re-evaluate the differences between the anti-hub estimates on high-hubness image data in Section 4.2. As for the threshold parameter, the average θ value for which the best accuracy was achieved was around 1.5 for all approaches. This means that more often than not, class hubness was to be preferred to any of the $f_k(c,x_i)$ terms, even when based only on 3 or 4 k-occurrences.

The frequencies of the selected neighborhood size falling in one of the four ranges: $[1,5]$, $[6,10]$, $[11,15]$, $[16,20]$, are shown in Fig. 4. Two ranges are preferred more often, namely $k \in [1,5]$ and $k \in [11,15]$. By examining all the results, we found that in cases of the more tangible accuracy improvements, larger k values ($k > 10$) were selected, while lower k values usually signified equal or only slightly better performance. This can be seen as natural, since larger k values provide the algorithm with more hubness information and hence better probability estimates, on which the used fuzziness was based. However, not all data sets are such that high k values make sense, since in some it may induce a larger breach of locality. This is why hubness-based approaches are not expected to lead to an improvement over all data sets. This is their inherent limitation. Of course, this also depends heavily on the size of a particular data set. With more data, higher k values can be observed more safely. In high-dimensional spaces this is also affected by the curse of dimensionality because the data is always sparse.

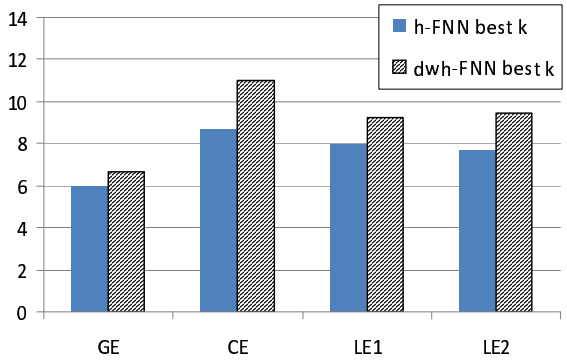


Fig. 3 Average best k values for different hubness-based fuzzy approaches, according to the results from tests on UCI data

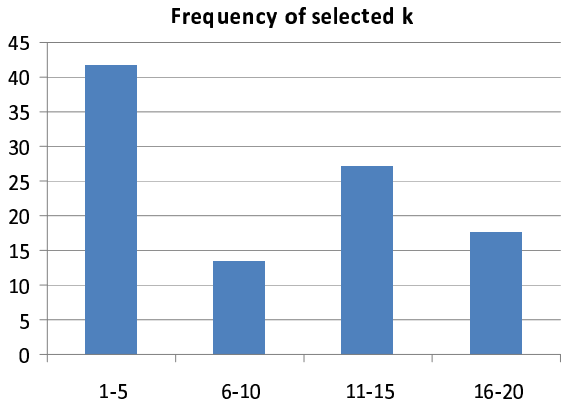


Fig. 4 Frequency of the selected best k values, based on the results from tests on UCI data

Table 4 Class structure of the used ImageNet data subsamples

Data set	Classes
subs-3	sea moss, fire, industrial plant
subs-4	cloud, butterfly orchid, herbaceous plant, bird
subs-5	bird, fire, tracked vehicle, people, compass flower
subs-6	fish, industrial plant, wind turbine, compass flower, butterfly orchid, evergreen plant
subs-7	football, worm, sea star, night club, cloud, orchidaceous plant, mountain range

Table 5 Summary of ImageNet data sets

Data set	size	d	n_c	S_{N_1}	$S_{N_{10}}$
subs-3	2731	416	3	15.85	6.19
subs-4	6054	416	4	8.87	6.32
subs-5	6555	416	5	26.08	11.88
subs-6	6010	416	6	13.19	6.23
subs-7	8524	416	7	5.62	4.60

4.2 ImageNet data

The ImageNet database (`http://www.image-net.org/`) is a large repository containing over 12 million images organized in more than 17000 synsets (classes). Images are intrinsically high-dimensional data, and are therefore quite suitable for testing hubness-based approaches. Out of synsets from the ImageNet hierarchy we constructed five image data sets for testing, with the used classes summarized in Table 4. Some of them combine more easily distinguishable images, as *subs-3*, while some are made more difficult by containing several different plant types in different categories, as in *subs-6*. SIFT features and color histograms were extracted for each image [32]. A codebook of 400 most representative SIFT features was obtained by clustering from a large sample. Each image was thus represented by a 400-dimensional array of codebook frequencies, as well as a 16-dimensional color histogram. We used the Manhattan distance on this group of data sets. No feature weighting was performed, meaning that color and texture information was given equal significance. This may not be optimal, but we were not interested in performing optimal image classification, since our goal was only to compare the approaches under consideration on high-dimensional data. As in the previous section, Table 5 gives an overview of the obtained data sets. Note that this data exhibits a much higher skew of the distribution of k-occurrences than most UCI data sets from Table 1.

On each of the subsamples we performed 10 times 10-fold cross-validation. The value of k was chosen automatically from the range $k \in [1,10]$ on each fold. Average accuracies of the classifiers are given in Table 6. Statistically significant differences ($p < 0.05$) compared to dwh-FNN are denoted by symbols ○/●. Pairwise classifier comparison is shown in Table 7.

Table 6 Classification accuracy of kNN, hubness-weighted kNN (hw-kNN), h-FNN and dwh-FNN on ImageNet data sets for $k \in [1,10]$. The symbol ● denotes statistically significant worse performance compared to dwh-FNN

Data set	kNN	hw-kNN	h-FNN	dwh-FNN
subs-3	78.29±2.38 ●	81.51±3.34	82.16±2.26	82.34±2.23
subs-4	54.68±2.02 ●	65.91±1.82	64.83±1.62	64.87±1.61
subs-5	50.80±2.08 ●	58.06±3.80 ●	61.54±1.93	61.81±1.95
subs-6	63.09±1.81 ●	70.10±1.68	68.84±1.58	69.04±1.64
subs-7	46.71±1.63 ●	51.99±4.68 ●	58.85±1.60	59.04±1.59
Average	54.71	65.51	67.24	67.42

Table 7 Pairwise comparison of classifiers on ImageNet data: number of wins (with the statistically significant ones in parenthesis)

	kNN	hw-kNN	h-FNN	dwh-FNN
kNN	–	5 (5)	5 (5)	5 (5)
hw-kNN	0 (0)	–	3 (2)	3 (2)
h-FNN	0 (0)	2 (0)	–	5 (0)
dwh-FNN	0 (0)	2 (0)	0 (0)	–

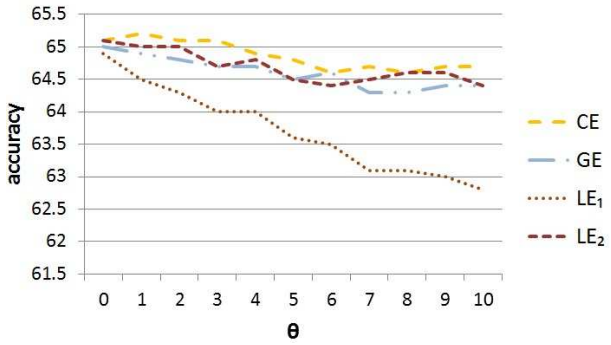


Fig. 5 A comparison between the fuzzy measures on subs-4.

Hubness-based algorithms show an obvious improvement on all subsets over the standard kNN classifier. As the number of classes increases, improvement of h-FNN and dwh-FNN over hubness-weighted kNN becomes more prominent, which is consistent with observations on UCI data.

In Section 4.1 we reported a brief comparison of the proposed fuzzy measures on medium-hubness UCI data, which revealed that all of them attain similar best accuracies, though for different k-values, when averaged over all the datasets. In Fig. 5 we focus on the comparison when varying the values of the threshold θ parameter. Higher θ values increase the influence of the $f_k(c,x_i)$ terms, while the lower threshold values emphasize the original point class-hubness frequencies, even when derived from very few occurrences. A comparison is shown on subs-4, one of the high-hubness ImageNet datasets that we have analyzed.

As in the previous case of medium-hubness data, the best accuracies are observed for low θ parameter values and the best results for all measures are very similar. However, more

differences can be observed as the θ is slowly increased and the $f_k(c,x_i)$ terms get more frequently used during voting. First of all, one notices that there is a clear difference between the performance of the two local estimates (LE_1 and LE_2) on this particular image dataset. In fact, LE_1 seems to be clearly inferior to LE_2, which is not surprising, given that it relies more on the crisp label than LE_1, and the crisp handling of the anti-hubs (CE) works best on this dataset.

The fact that all the measures achieve similar best scores and that this always takes place for low θ values makes the task of choosing the appropriate measure and the appropriate threshold much easier in practice. By setting $\theta = 0$ or $\theta = 1$ and by using either of the CE, GE or LE_2 estimate methods, one could hope to achieve very good results. This is important, as it essentially removes the need for performing cross-validation when doing the search for the best parameter configuration. It is a very time consuming step and removing it helps speed up the algorithm. This may not be very important for small datasets, but most real-world datasets are quite large and scalability is certainly important.

Noisy and compromised data, on the other hand, need to be handled somewhat more carefully. Most measurement errors, noisy points and outliers tend to be anti-hubs, though the reverse implication does not necessarily hold. This means that unreliable points would tend to have low hubness in most complex, real-world datasets. The negative influence of erroneous points could be reduced by setting a slightly higher threshold ($\theta > 1$) and relying more on the global class-to-class hubness estimate (GE) for handling such anti-hubs. It ought to be more reliable in noisy data scenarios than CE or LE_1 and LE_2, as the labels of such potentially incorrect data points are often wrong and the neighbors might be quite distant and less relevant for estimating the local occurrence fuzziness. If the data is not prohibitively large, it is still advisable to look for the best parameter configuration automatically during the training phase, as outlined in the algorithm 1.

4.3 Scalability

One of the most important issues in modern data-mining tasks is scalability, since we are mostly faced with problems involving big data. Algorithms that perform well in terms of accuracy, but scale poorly, are not really useful in most practical applications. This is why we decided to test how the proposed h-FNN and dwh-FNN perform under approximate *k*NN set calculation, which is used to speed up the procedures. We chose the approximate *k*NN graph construction algorithm described in [7], which is a divide and conquer method based on recursive Lanczos bisection. As mentioned before, the time complexity of the procedure is $\Theta(dn^{1+\tau})$, where $\tau \in (0,1]$ reflects the quality of the approximation. The main question which arises is: for which values of τ could we still retain the good performance observed on the actual *k*NN sets? Fig. 6 shows some encouraging results.

We see that the hubness-based approaches seem to be quite robust to approximate calculation of the *k*NN sets on the data, at least we could say that is the case for this particular employed approximate algorithm [7]. Improvements over the baseline *k*NN remain even for $\tau = 0$, which essentially means that the total overhead over the basic *k*NN can be reduced to linear time complexity, which is excellent. We also see that dwh-FNN remains better than hw-*k*NN in all approximate cases, which implies that the relative strengths and weaknesses of the approaches remain unchanged under such conditions.

The exact algorithm (with no approximations) is of the squared time complexity (and memory requirements) which makes it applicable to most medium-to-large real world datasets, though it may have difficulties handling *very* large datasets. On the other hand, constructing a *k*NN graph (in order to calculate the hubness scores) is among those tasks that can be easily solved by distributed computing. We are also using some initial multi-threaded implementations. As for the single-threaded implementation, the overall performance of h-FNN and dwh-FNN is essentially the same as in hw-*k*NN, since both algorithms spend most of the training time on calculating the distance matrix and all the *k*NN sets. The additional time required to summarize the class-hubness scores and/or make some local or global estimates for anti-hubs is negligible when compared to the two main sub-tasks.

In order to compare the approaches, we have generated a series of synthetic 100-dimensional Gaussian mixtures and we have measured the training time of each of the methods separately. According to Fig. 7, h-FNN and hw-*k*NN take almost the same amount of time for the training phase, while the most time consuming approach is to use the cross-validation in h-FNN or dwh-FNN in order to try and find the best fuzzy measure and the best parameter configuration (θ,

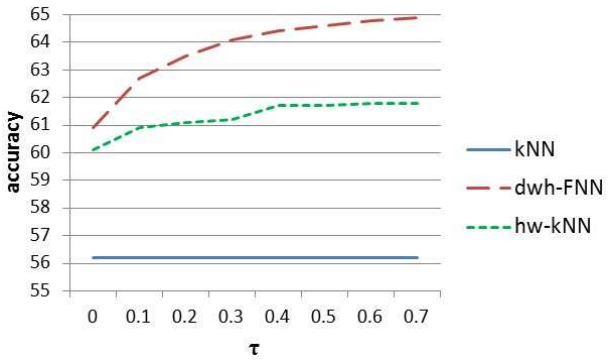


Fig. 6 The accuracy of the hubness-based approaches on subs-4 when the occurrence model is inferred from the approximate *k*NN graph generated by [7]. We see that there are significant improvements even for $\tau = 0$

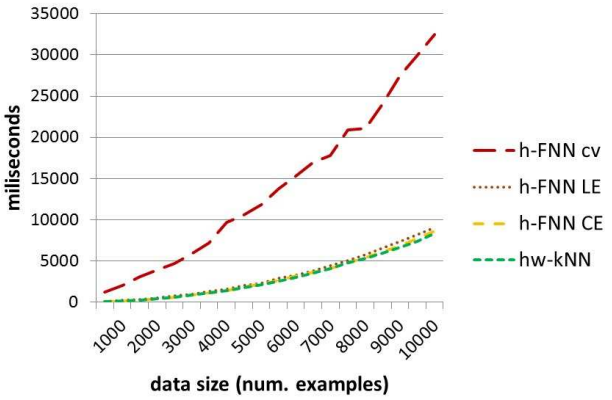


Fig. 7 The execution times of the training phase of hw-*k*NN, h-FNN employing CE/GE or LE_1/LE_2 and h-FNN performing cross-validation on the training set to decide on the best fuzzy measure and parameter set-up. $k = 5$ was used in the experiment. *k*NN is omitted, as it requires no training. All experiments were performed on a computer with an i7 Intel processor and 8Gb RAM.

M). Fortunately, as we have already discussed in Section 4.2, it seems that this is not really necessary in practice and that it is not so difficult to come up with good *default* parameters which ought to work well on most datasets. All curves in Fig. 7 do not intersect and the ordering remains the same as data size is increased: t(h-FNN cv) $>$ t(h-FNN LE) $>$ t(h-FNN CE) $>$ t(hw-*k*NN), though the differences between the last three are apparently minor. In other words, the improvement that h-FNN and dwh-FNN achieve over hw-*k*NN is essentially *free*, from the perspective of time complexity.

4.4 Probability landscapes

When estimating the potential benefits of using a particular classification algorithm, accuracy is not the only quantity of interest. We would also like the algorithm to be able to provide us with decent confidence measures behind its label assignments, which would provide the experts using the system with valuable additional information. Fuzzy approaches are, well, more 'fuzzy' and 'soft' to begin with, so they always do output some sort of a confidence measure alongside the final vote. The question remains: how *good* are these associated numbers?

A complete analysis of the associated probabilities in all conceivable scenarios would be quite difficult and is certainly beyond the scope of this paper. We will, however, shed some light on the quality of the employed fuzzy measures by analyzing a couple of illustrative examples. We will consider the two-dimensional synthetic data sets shown in Fig. 8. We opted for 2D data so that we can easily visualize the results.

For the two data sets DS_1 and DS_2, shown in Fig. 8, we computed the probability landscapes in the following way: we performed a fifth-order Voronoi tessellation in the plane ($k = 5$) and then assigned a class probability to every pixel in each of the obtained cells by each of the considered algorithms (*k*NN, hw-*k*NN, h-FNN, dwh-FNN).

The probability landscapes generated for DS_1 are shown in Fig. 9. It is immediately apparent that *k*NN produces a fractured landscape, which indicates over-fitting. When there are many more points and a higher k value can be safely used, this is less of a problem. Real-world data, however, are not two-dimensional and are hence always sparse, much more so than in the considered DS_1 data set. This suggests that the basic *k*NN can not be expected to give reasonable probability estimates in such scenarios. The hubness-based weighting apparently helps, even though there is no hubness in two dimensions. However, it still reduces the votes of some less reliable borderline points. The hubness-based fuzzy approaches produce landscapes that are even more smooth, which seems like a nice property for a model of the data.

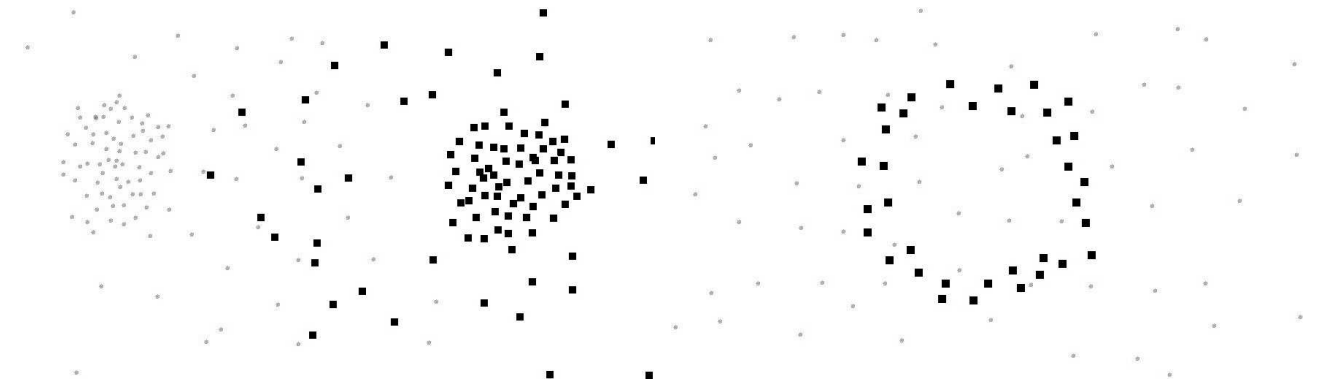
As for the second, ring-shaped data set, the associated probability landscapes are shown in Fig. 10. Once again we see that the basic *k*NN classifier over-fits on certain points and fails to detect a common theme. The hubness-based fuzzy *k*-nearest neighbor classifier (h-FNN) gives the most reasonably-looking result and hw-*k*NN lies somewhere in between the two.

The hubness-based *k*NN algorithms discussed in this paper are not designed to model the data directly, but are able to capture some of the underlying regularities in the data by virtue of building an occurrence model. We have observed some encouraging results on two-dimensional synthetic data sets. However, investigating the overall performance in the general case is not as easy, therefore we can only assume for now that these observations may generalize to the high-dimensional case as well. In a sense, it can be considered a reasonable assumption, since both of these algorithms have been tailored specifically for high-dimensional data in the first place and the very fact that they perform very well in the low-dimensional case is an unexpected beneficial property of the algorithms.

5 Conclusions and future work

We have proposed several ways of incorporating hubness into fuzzy membership functions for data points in *k*NN classification. This was meant as a generalization of the previous hubness-weighted *k*NN approach. The fuzzy *k*-nearest neighbor classification offers better confidence measures for label assignments, which is a highly desirable property.

Several hybrid fuzzy membership functions were tested and evaluated. The fuzzy *k*-nearest neighbor classifier employing these fuzzy measures outperforms the basic *k*NN classifier and also offers improvement over the crisp hubness-weighted *k*NN. The accuracy improvement thus achieved



(a) DS_1 (b) DS_2

Fig. 8 Two 2D synthetic binary datasets. The first one depicts a case of two overlapping Gaussian-like distributions with dense central regions and sparse border regions. The second example shows a ring-like distribution immersed into a rather uniform background distribution, which could even be interpreted as noise

may not be large on average, but the main advantage of the fuzzy approach lies in the mentioned interpretability of the results, and the fact that the approach takes advantage of high intrinsic dimensionality of the data instead of being hampered by it, taking a step closer to mitigating the curse of dimensionality.

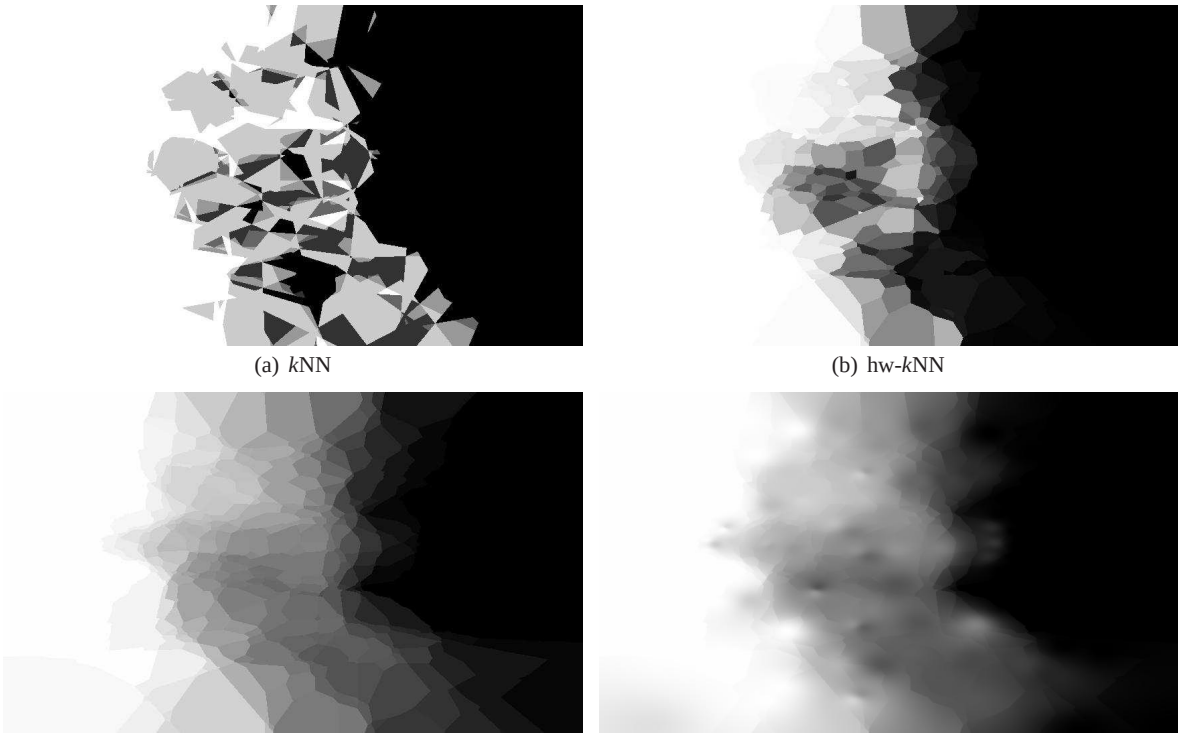
The approach seems to be quite scalable when the approximate *k*NN sets are used instead. Most of the original accuracy is retained when opting for such speedup in computation.

These fuzzy measures represent but one way of exploiting the hubness phenomenon for classification. There are yet other paths to follow and we intend to address many of the related issues in our future work.

Acknowledgements This work was supported by the bilateral project between Slovenia and Serbia "Correlating images and words: Enhancing image analysis through machine learning and semantic technologies," the Slovenian Research Agency, the Serbian Ministry of Education and Science through project no. OI174023, "Intelligent techniques and their integration into wide-spectrum decision support," and the ICT Programme of the EC under PASCAL2 (ICT-NoE-216886) and PlanetData (ICT-NoE-257641).

References

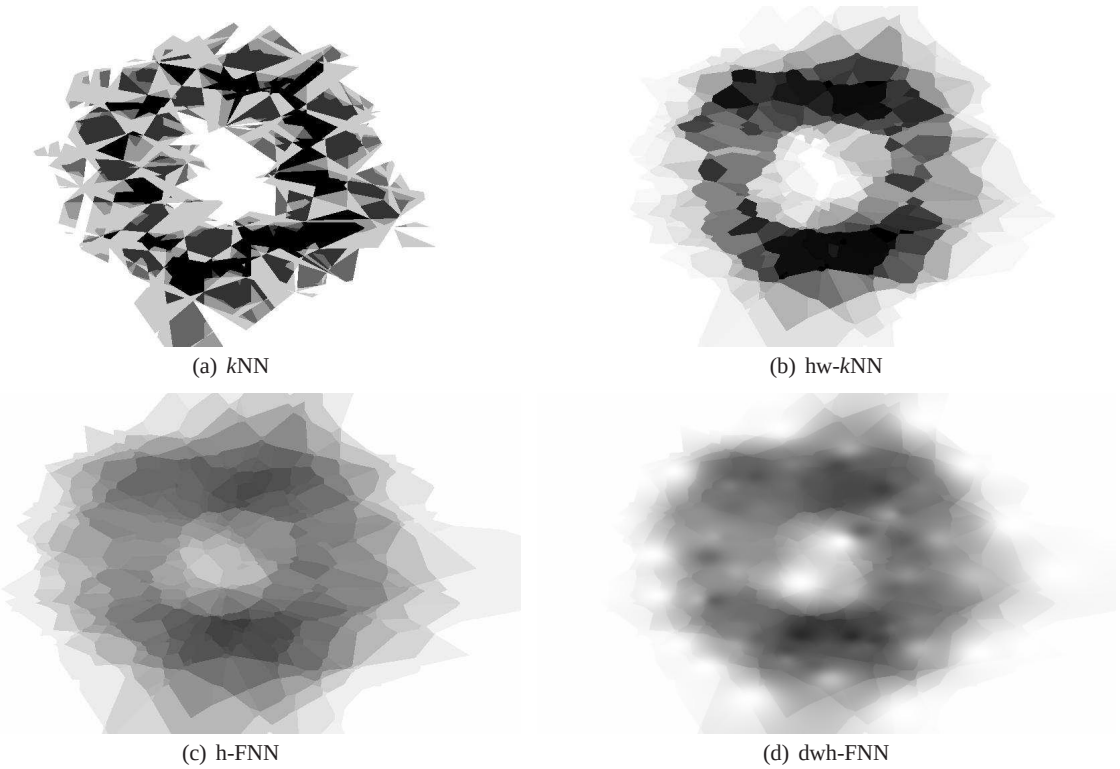
1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional spaces. In: Proceedings of the 8th International Conference on Database Theory (ICDT), *Lecture Notes in Computer Science*, vol. 1973, pp. 420–434. Springer (2001)
2. Aucouturier, J.J.: Ten experiments on the modelling of polyphonic timbre. Ph.D. thesis, University of Paris 6 (2006)
3. Aucouturier, J.J., Pachet, F.: Improving timbre similarity: How high is the sky? Journal of Negative Results in Speech and Audio Sciences **1** (2004)
4. Babu, V.S., Viswanath, P.: Rough-fuzzy weighted k-nearest leader classifier for large data sets. Pattern Recognition **42**(9), 1719–1731 (2009)
5. Buza, K., Nanopoulos, A., Schmidt-Thieme, L.: INSIGHT: Efficient and effective instance selection for time-series classification. In: Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Part II, *Lecture Notes in Artificial Intelligence*, vol. 6635, pp. 149–160. Springer (2011)
6. Cabello, D., Barro, S., Salceda, J.M., Ruiz, R., Mira, J.: Fuzzy k-nearest neighbor classifiers for ventricular arrhythmia detection. International Journal of Bio-Medical Computing **27**(2), 77–93 (1991)
7. Chen, J., Fang, H., Saad, Y.: Fast approximate *k*NN graph construction for high dimensional data via recursive Lanczos bisection. Journal of Machine Learning Research **10**, 1989–2012 (2009)
8. Cintra, M.E., Camargo, H.A., Monard, M.C.: A study on techniques for the automatic generation of membership functions for pattern recognition. In: Congresso da Academia Trinacional de Ciências (C3N), vol. 1, pp. 1–10 (2008)
9. Durrant, R.J., Kabán, A.: When is 'nearest neighbour' meaningful: A converse theorem and implications. Journal of Complexity **25**(4), 385–397 (2009)
10. François, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. IEEE Transactions on Knowledge and Data Engineering **19**(7), 873–886 (2007)
11. Houle, M.E., Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Can shared-neighbor distances defeat the curse of dimensionality? In: Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), *Lecture Notes in Computer Science*, vol. 6187, pp. 482–500. Springer (2010)
12. Huang, W.L., Chen, H.M., Hwang, S.F., Ho, S.Y.: Accurate prediction of enzyme subfamily class using an adaptive fuzzy k-nearest neighbor method. Biosystems **90**(2), 405–413 (2007)
13. Keller, J.E., Gray, M.R., Givens, J.A.: A fuzzy k-nearest neighbor algorithm. IEEE Transactions on Systems, Man and Cybernetics **15**(4), 580–585 (1985)
14. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning **52**(3), 239–281 (2003)
15. Pham, T.D.: An optimally weighted fuzzy k-NN algorithm. In: Proceedings of the 3rd International Conference on Advances in Pattern Recognition (ICAPR), Part I, *Lecture Notes in Computer Science*, vol. 3686, pp. 239–247. Springer (2005)
16. Radovanović, M., Nanopoulos, A., Ivanović, M.: Nearest neighbors in high-dimensional data: The emergence and influence of hubs. In: Proceedings of the 26th International Conference on Machine Learning (ICML), pp. 865–872 (2009)



(a) *k*NN (b) hw-*k*NN

(c) h-FNN (d) dwh-FNN

Fig. 9 The probability landscapes of the analyzed algorithms on DS_1 for $k = 5$. The difference between different approaches is quite obvious, as both hw-*k*NN and the fuzzy hubness-based approach produce much smoother landscapes than the basic *k*NN



(a) *k*NN (b) hw-*k*NN

(c) h-FNN (d) dwh-FNN

Fig. 10 The probability landscapes of the analyzed algorithms on DS_2 for $k = 5$

17. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: Popular nearest neighbors in high-dimensional data. Journal of Machine Learning Research **11**, 2487–2531 (2010)
18. Radovanović, M., Nanopoulos, A., Ivanović, M.: On the existence of obstinate results in vector space models. In: Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 186–193 (2010)
19. Radovanović, M., Nanopoulos, A., Ivanović, M.: Time-series classification in many intrinsic dimensions. In: Proceedings of the 10th SIAM International Conference on Data Mining (SDM), pp. 677–688 (2010)
20. Shen, H.B., Yang, J., Chou, K.C.: Fuzzy KNN for predicting membrane protein types from pseudo-amino acid composition. Journal of Theoretical Biology **240**(1), 9–13 (2006)
21. Sim, J., Kim, S.Y., Lee, J.: Prediction of protein solvent accessibility using fuzzy k-nearest neighbor method. Bioinformatics **21**(12), 2844–2849 (2005)
22. Singpurwalla, N., J.M., B.: Membership functions and probability measures of fuzzy sets. Journal of the American Statistical Association **99**, 867 – 877 (2004)
23. Tomašev, N., Brehar, R., Mladenić, D., Nedevschi, S.: The influence of hubness on nearest-neighbor methods in object recognition. In: Proceedings of the 7th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 367–374 (2011)
24. Tomašev, N., Mladenić, D.: Exploring the hubness-related properties of oceanographic sensor data. In: Proceedings of the 14th International Multiconference on Information Society (IS), vol. A, pp. 149–152 (2011)
25. Tomašev, N., Mladenić, D.: The influence of weighting the k-occurrences on hubness-aware classification methods. In: Proceedings of 14th International Multiconference on Information Society (2011)
26. Tomašev, N., Mladenić, D.: Nearest neighbor voting in high dimensional data: Learning from past occurrences. Computer Science and Information Systems **9**(2) (2012)
27. Tomašev, N., Radovanović, M., Mladenić, D., Ivanović, M.: Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification. In: Proceedings of the 7th International Conference on Machine Learning and Data Mining (MLDM), *Lecture Notes in Artificial Intelligence*, vol. 6871, pp. 16–30. Springer (2011)
28. Tomašev, N., Radovanović, M., Mladenić, D., Ivanović, M.: The role of hubness in clustering high-dimensional data. In: Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Part I, *Lecture Notes in Artificial Intelligence*, vol. 6634, pp. 183–195. Springer (2011)
29. Wang, X.Z., He, Y.L., Dong, L.C., Zhao, H.Y.: Particle swarm optimization for determining fuzzy measures from data. Information Sciences **181**(19), 4230 – 4252 (2011)
30. Yu, S., Backer, S.D., Scheunders, P.: Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. Pattern Recognition Letters **23**(1–3), 183–190 (2002)
31. Zadeh, L.A.: Fuzzy sets. Information and Control **8**(3), 338–353 (1965)
32. Zhang, Z., Zhang, R.: Multimedia Data Mining. Chapman and Hall (2009)
33. Zheng, K., Fung, P.C., Zhou, X.: K-nearest neighbor search for fuzzy objects. In: Proceedings of the 36th ACM SIGMOD International Conference on Management of Data, pp. 699–710 (2010)
34. Zuo, W., Zhang, D., Wang, K.: On kernel difference-weighted k-nearest neighbor classification. Pattern Analysis and Applications **11**, 247–257 (2008)