How k-Nearest Neighbor Parameters Affect its Performance*

Gustavo E.A.P.A. Batista[1], Diego Furtado Silva[1]

Laboratório de Inteligência Computacional (LABIC)
Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
Caixa Postal 668 – 13560-970 – São Carlos – SP – Brasil
gbatista@icmc.usp.br, diegofurts@grad.icmc.usp.br

Abstract. The k-Nearest Neighbor is one of the simplest Machine Learning algorithms. Besides its simplicity, k-Nearest Neighbor is a widely used technique, being successfully applied in a large number of domains. In k-Nearest Neighbor, a database is searched for the most similar elements to a given query element, with similarity defined by a distance function. In this work, we are most interested in the application of k-Nearest Neighbor as a classification algorithm, i.e., each database element has a label (class) associated, and the main goal of the algorithm is to decide the class of a new case based on the classes of the k most similar database elements. This work provides a discussion and presents empirical evidence of how the main parameters of k-Nearest Neighbor influence its performance. The parameters investigated are the number of nearest neighbors, distance function and weighting function. The most popular parameters choices were evaluated, including nine values for k, three popular distance measures and three well-known weighting functions. Our experiments were performed over thirty-one benchmark and "real-world" data sets. We recommend the use of the inverse weighting function and $k = 5$ for HEOM and HMOM distance functions and $k = 11$ to HVDM distance function.

1 Introduction

The k-Nearest Neighbor is one of the simplest Machine Learning algorithms. Besides its simplicity, k-Nearest Neighbor is a widely used technique, being successfully applied in a large number of domains. Most of the recent interest in the k-Nearest Neighbor search is due to the increasing availability of data represented in new data types such as free text, images, audio, video and so on. In order to search on these data types, traditional *exactly equal* search algorithms are being replaced by *similarity* search algorithms, such as k-Nearest Neighbor.

* This work was partially supported by the Brazilian Research Agencies CNPq and FAPESP.

In k-Nearest Neighbor, a database is searched for the most similar elements to a given query element, with similarity defined by a distance function. In this work, we are most interested in the application of k-Nearest Neighbor as a classification algorithm, i.e., each database element has a label (class) associated, and the main goal of the algorithm is to decide the class of a new case based on the classes of the k most similar database elements.

This work provides a discussion and presents empirical evidence of how the main parameters of k-Nearest Neighbor influence its performance. The parameters investigated are the number of nearest neighbors, distance function and weighting function. The most popular parameters choices were evaluated, including nine values for k, three popular distance measures and three well-known weighting functions.

Our experiments were performed over thirty-one benchmark and "real-world" data sets. In addition, we have improved the implementation of k-Nearest Neighbor algorithm available in Weka[1] to include the parameters employed in this evaluation, such as the inverse weighting and HVDM distance functions. In order to facilitate the reproduction of the results presented in this paper, our implementation of k-Nearest Neighbor is available as Java source code and also embedded in a compiled Weka jar file[2].

Even though it is common-sense in the community that the best combination of parameters values depends on the data sample at hand, a large number of research articles does not perform any parameter tuning at all. This can be partially explained by the fact that most of the parameter estimation approaches, such as cross-validation, are computationally expensive. This fact allied to the problem that the combination of parameters values grows exponentially, and the necessity to evaluate on a large number of data sets, make parameter estimation computationally expensive in several experimental setups.

In this scenario that the contribution of this paper is most valuable. We suggest parameters values that provide the best *mean* performance for a number of data sets. These values can be used as default by researchers and practitioners. One example is the Weka application, whose default value $k = 1$, for its implementation of k-Nearest Neighbor, hardly provides the best results. A second contribution is the understanding of how the parameters influence the performance of the k-Nearest Neighbor algorithm. This knownledge can be useful, for instance, to narrow the range of values to be evaluated by cross-validation, making the parameter estimation process less computationally intensive.

This work is organized as follows: Section 2 presents a critical view of parameter setting in Machine Learning; Section 3 discusses the main features of the k-Nearest Neighbor algorithm in our implementation; Section 4 presents our experimental setup and discusses the obtained results; and Section 5 concludes this work.

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] http://www.icmc.usp.br/~gbatista/knn/

2 A Critical View on Parameter Tuning

Parameter tuning is an important but frequently neglected subject in Machine Learning. Algorithms that have a large number of parameters or strongly depend on parameters fine tuning are difficult to evaluate, and results obtained by these algorithms are difficult to reproduce. Parameter setting is usually a very computationally intensive task since parameters usually have a large set of values to be evaluated. In addition, the combination of all possible parameters values to be evaluated usually leads to a combinatorial explosion.

Approaches frequently used in parameter setting, such as cross-validation are also very expensive. In a simple scenario, an algorithm that has three parameters with ten values each, with error estimation and parameter setting by ten-fold cross-validation will have to be executed 100,000 times for each data set. An important note here is that one needs a complete ten-fold cross-validation used to parameter setting for each iteration of the ten-fold cross-validation used to error estimation, since using the same test set for evaluating and parameter tuning is considered a serious methodological mistake [1].

We advocate all proposals of new Machine Learning methods should be accompanied by an evaluation of the methods' parameters. Such evaluation should identify the influence of the parameters on the peformance of the proposed method. This knownledge can be useful, for instance, to narrow the range of values of a parameter to be evaluated by cross-validation, making the parameter estimation process less computationally intensive. In addition, such evaluation allows to propose default parameters values based on the best *mean* performance. These values can be used by researchers and practitioners when one needs to perform a preliminary analysis.

In this paper, we present the results obtained by an evaluation of the influence of some of the main parameters on the performance of the k-Nearest Neighbor algorithm. A detailed explanation of our implementation of this algorithm is presented in the next section.

3 The k-Nearest Neighbor Algorithm

This section discusses the k-Nearest Neighbor algorithm and is partially based on the excellent introdution to instance-based learning given in [2]. All features discussed in this section are present in our implementation.

3.1 The Basic k-Nearest Neighbor Algorithm

The basic version of the k-Nearest Neighbor algorithm assumes that all instances correspond to points in the n-dimensional space $\Re^n$. The nearest neighbors of an instance are often defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance be described by the vector $\langle x, f(x) \rangle$. f is the true concept function that gives the correct class value $f(x)$ for each instance x. The instance x can also be described by the feature vector

$$\langle a_1(x), a_2(x), \ldots a_n(x) \rangle$$

where $a_r(x)$ denotes the value of the rth attribute of instance x.

The distance between two instances x_i and x_j is denoted as $d(x_i, x_j)$. For the standard Euclidean distance, $d(x_i, x_j)$ is defined by Equation 1.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{1}$$

In nearest neighbor learning the concept function can be either discrete-valued (for qualitative class attributes) or real-valued (for quantitative class attributes). In this work, we are most interested in learning discrete-valued concept functions of the form $f : \Re^n \rightarrow V$, where V is the finite set $\{v_1, \ldots v_s\}$ of class values. The k-Nearest Neighbor algorithm is described in Table 1.

Table 1: The basic k-Nearest Neighbor algorithm for approximating discrete-valued functions [2].

```
Training algorithm:
  – For each training example $\langle x, f(x) \rangle$, add the example to the list training_examples
Classification algorithm:
  – Given a query instance $x_q$ to be classified,
      • Let $x_1, \ldots x_k$ denote the $k$ instances from training_examples that are nearest to $x_q$
      • Return
            $\hat{f}(x_q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
        where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.
```

The value $\hat{f}(x_q)$ returned by this algorithm as its estimate of $f(x_q)$ is just the mode (most common value) of the true concept function f among k training examples nearest to x_q. If $k = 1$, then the 1-Nearest Neighbor algorithm assigns to $\hat{f}(x_q)$ the value $f(x_i)$ where x_i is the training instance nearest to x_q. For larger values of k, the algorithm assigns the mode among the k nearest training examples.

3.2 Distance-Weighted Nearest Neighbor Algorithm

One refinement to the basic k-Nearest Neighbor algorithm is to weight the contribution of each of the k neighbors according to their distance to the query instance x_q, giving greater weight to closer neighbors. Two widely used approaches to calculate such weights are: (1) to weight the vote of each neighbor according to the inverse square of its distance from x_q, and; (2) to weight the vote of each neighbor according to its similarity with x_q, where similarity is defined as $1 - d(x_q, x_i)$.

This can be accomplished by replacing the final line of the algorithm of Table 1 by

$$\hat{f}(x_q) = \max_{v \in V} \sum_{i=1}^{k} \omega_i \delta(v, f(x_i)) \tag{2}$$

where

$$\omega_i = \frac{1}{d(x_q, x_i)^2} \tag{3}$$

or

$$\omega_i = 1 - d(x_q, x_i) \tag{4}$$

In the case of Equation 3, in order to accommodate the case where the query point x_q exactly matched one of the training instances x_i and the denominator $d(x_q, x_i)^2$ is therefore zero, $\hat{f}(x_q)$ can be assigned to $f(x_i)$ in this case. If there are several such training examples, $\hat{f}(x_q)$ can be assigned to the majority classification among them.

3.3 VDM, HEOM and HVDM Distance Functions

The Euclidean distance is widely used and it is well suited for quantitative attributes. However, one criticism about this distance is that it does not handle naturally qualitative attributes. An approach to handle data sets with both qualitative and quantitative attributes is to use a heterogeneous distance function that uses different attribute distance functions for different types of attributes. For instance, one might use the *overlap* metric for qualitative attributes and the normalized Euclidean distance for quantitative attributes. This approach is known as *Heterogeneous Euclidean-Overlap Metric (HEOM)* [3], and the distance between two instances x_i and x_j is defined by Equation 5.

$$HEOM(x_i, x_j) = \sqrt{\sum_{r=1}^{n} d_a(a_r(x_i), a_r(x_j))^2} \tag{5}$$

where the distance function $d_a(\cdot, \cdot)$ is defined by Equation 6.

$$d_a(a_r(x_i), a_r(x_j)) = \begin{cases} 1, \text{ if } a_r(x_i) \text{ or } a_r(x_j) \text{ is unknown; otherwise} \\ \mathit{overlap}(a_r(x_i), a_r(x_j)), \text{ if } a_r \text{ is qualitative} \\ \mathit{range\_norm\_diff}(a_r(x_i), a_r(x_j)), \text{ if } a_r \text{ is quantitative} \end{cases} \tag{6}$$

Unknown attribute values are handled by returning an attribute distance value of 1 (*i.e.*, maximal distance) if either of the attribute values is unknown. The function *overlap* and the $range\_normalized\_diff$ are defined by Equation 7 and Equation 8, respectively.

$$overlap(a_r(x_i), a_r(x_j)) = \begin{cases} 0, & \text{if } a_r(x_i) = a_r(x_j) \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

$$range\_norm\_diff(a_r(x_i), a_r(x_j)) = \frac{|a_r(x_i) - a_r(x_j)|}{max_{a_r} - min_{a_r}} \tag{8}$$

where max_{a_r} and min_{a_r} are the maximum and minimum values, respectively, observed in the training set for attribute a_r. This means that it is possible for a new (unseen) input vector to have a value outside this range, producing a difference value greater than one. However, such cases are rare, and when they occur, a larger difference may be acceptable anyway.

Other heterogeneous distances can de defined in a similar way. For instance, all L^p norm distances can be composed with overlap metric using the same approach. In particular, *Heterogeneous Manhattan-Overlap Metric (HMOM)* is a distance that uses Manhattan instead of Euclidean distance.

One criticism about the overlap metric is that it fails to make use of additional information available about qualitative attributes. An approach to overcome this limitation is the *Value Difference Metric (VDM)* proposed in [4]. The VDM metric considers the classification similarity for each possible value of an attribute to calculate the distances between these values. As a result, a matrix of distances is created from the training set for each attribute. The distance $VDM_a(a_r(x_i), a_r(x_j))$ between two values $a_r(x_i)$ and $a_r(x_j)$ of a given qualitative attribute a_r, is defined by Equation 9.

$$VDM_a(a_r(x_i), a_r(x_j)) = \sum_{l=1}^{s} \left| \frac{N_{a_r(x_i),v_l}}{N_{a_r(x_i)}} - \frac{N_{a_r(x_j),v_l}}{N_{a_r(x_j)}} \right|^c \tag{9}$$

where

- $N_{a_r(x_i)}$ is the number of instances in the training set that have value $a_r(x_i)$ for attribute a_r;
- $N_{a_r(x_i),v_l}$ is the number of instances in the training set that have value $a_r(x_i)$ for attribute a_r and class value v_l;
- s is the number of class values in the data set;
- c is a constant, usually 1 or 2.

The VDM metric considers two values similar if they have similar classifications (i.e., similar correlations with the output classes), regardless of the values ordering. For example, if an attribute *color* has three values *red*, *green* and *blue*, and the application is to identify whether or not an object is an apple, *red* and *green* would be considered closer than *red* and *blue* because the former two have similar correlations with the output class *apple*.

If VDM is used directly on quantitative attributes, most of the values might be unique, in which case $N_{a_r(x_i)}$ is 1, and $N_{a_r(x_i),v_l}$ is 1 for a value v_l and 0 for all other attribute class values. In addition, query instances are likely to have unique values not present in the training set. In this case $N_{a_r(x_i),v_l}$ will be 0 for all $v_l \in V$ and $N_{a_r(x_i)}$ (which is the sum of $N_{a_r(x_i),v_l}$ for all classes) will also be 0, resulting in a division by zero. Even if all quantitative attribute values are not unique, there are often too many different values, thus the statistical sample is unreliably small for each value, and the distance measure is still untrustworthy. Because of these problems, it is inappropriate to use the VDM directly on quantitative attributes.

One approach to overcome the problem of using VDM on quantitative attributes is *discretization.* A quantitative attribute can be discretized and treated as a qualitative attribute. However, discretization can lose much of the important information available in quantitative values.

The *Heterogeneous Distance Function (HVDM)* [3] is a distance function similar to HEOM, except that it uses VDM instead of an overlap metric for qualitative attributes and it also normalizes differently. This distance function is defined by Equation 10.

$$HVDM(x_i, x_j) = \sqrt{\sum_{r=1}^{n} d_a(a_r(x_i), a_r(x_j))^2} \tag{10}$$

where n is the number of attributes in the data set. $d_a(a_r(x_i), a_r(x_j))$ is a distance between two values $a_r(x_i)$ and $a_r(x_j)$ of a given attribute a_r, and is defined as

$$d_a(a_r(x_i), a_r(x_j)) = \begin{cases} 1, \text{ if } a_r(x_i) \text{ or } a_r(x_j) \text{ is unknown; otherwise} \\ \mathit{normalized\_vdm}(a_r(x_i), a_r(x_j)), \text{ if } a_r \text{ is qualitative} \\ \mathit{normalized\_diff}(a_r(x_i), a_r(x_j)), \text{ if } a_r \text{ is quantitative} \end{cases} \tag{11}$$

Since 95% of the values in a normal distribution fall within two standard deviations of the mean, the difference between numeric values is divided by 4 standard deviations to scale each value in a range that is usually of width 1. The function $\mathit{normalized\_diff}$ is therefore defined as

$$\mathit{normalized\_diff}(a_r(x_i), a_r(x_j)) = \frac{(|a_r(x_i) - a_r(x_j)|)}{4\sigma_{a_r}} \tag{12}$$

where σ_{a_r} is the standard deviation of the numeric values of attribute a_r.

The $\mathit{normalized\_vdm}$ is defined as

$$\mathit{normalized\_vdm}(a_r(x_i), a_r(x_j)) = \sqrt{\sum_{l=1}^{s} \left| \frac{N_{a_r(x_i),v_l}}{N_{a_r(x_i)}} - \frac{N_{a_r(x_j),v_l}}{N_{a_r(x_j)}} \right|^2} \tag{13}$$

4 Experimental Analysis

The experimental analysis was carried out using thirty-one benchmark and "real world" data sets. In order to facilitate reproduction of our results, we employed twenty-eight benchmark data sets from University of California at Irvine Repository [5]. The other three data sets were donated by other researchers or were employed in previous researches of the first author: the Oil Spill data set [6] which purpose is to build a classifier to recognize oil spills in satellite radar images; the Mammography data set [7] which objective is to classify pixels in mammography images as possibly cancerous; and the Hoar-frost data set [8] which objective is to predict the possibility of a hoar-frost within a 24-hour window.

Table 2 summarizes the data sets used in this study. It shows, for each data set, the data set name, a shorter data set identifier used in the text, the number of examples, the number of attributes and the distribution of quantitative and qualitative attributes, the number of classes, the presence of missing data and the majority class error.

Table 2: Data sets summary description.

Data Set Name	Identifier	#Examples	#Attributes (quanti., quali.)	#Classes	Missing Data	Majority Class Error
1984 United States Congressional Voting Records Database	Vote	435	16 (0,16)	2	Yes	38,621%
Abalone Data Set	abalone	4176	8 (7,1)	2	No	2,717%
Annealing Data Set	Anneal	898	38 (6,32)	6	No	23,831%
Breast Cancer Data Set	Breast-cancer	286	9 (0,9)	2	Yes	29,720%
Credit Approval	Credit-a	690	15 (6,9)	2	Yes	44,493%
Final Settlements in Labor Negotitions in Canadian Industry	Labor-neg-data	57	16 (8,8)	2	Yes	35,088%
German Credit Data Set	Credit-g	1000	20 (7,13)	2	No	30,000%
Hepatitis Domain	Hepatitis	155	19 (14,5)	2	Yes	20,645%
Hoar-frost Detection Data Set	Hoar-frost	3043	236 (200,36)	2	No	6,112%
Horse Colic Database	Horse-Colic	368	22 (7,15)	2	Yes	36,957%
Hypothyroid Database	Hypothyroid	3772	29 (7,22)	4	Yes	7,715%
Johns Hopkins University Ionosphere Database	Ionosphere	351	34 (34,0)	2	No	35,897%
Landsat Satellite	Satimage	6435	36 (36, 0)	2	No	9,730%
Large Soybean Database	Soybean	683	35 (0,35)	19	Yes	86,53%
Letter Recognition	Letter	20000	16 (16, 0)	26	No	95,935%
Liver Disorders	Bupa	345	6 (6, 0)	2	No	41,861%
MAGIC Gamma Telescope Data Set	Magic 04	19019	10 (10,0)	2	No	35,16%
Microcalcifications in Mammography Data Set	Mammography	11182	6 (6,0)	2	No	2,316%
Nursery Database	Nursery	12960	8 (8, 0)	5	No	6,666%
Oil Spills Detection in Satellite Radar Images Data Set	Oil-spill	936	48 (48,0)	2	No	4,380%
Optical Recognition of Handwritten Digits Data Set	Optdigits	5619	63 (63,00)	2	No	9,913%
Page Blocks Classification Data Set	Page-blocks	5472	10 (10,0)	2	No	10,216%
Pen-Based Recognition of Handwritten Digits Data Set	Pen-digits	10991	16 (16,0)	2	No	10,409%
Pima Indians Diabetes	Diabetes	768	8 (8, 0)	2	No	34,896%
Protein Localization Sites	Ecoli	336	7 (7,0)	8	No	57,441%
Thyroid Disease Data Set	Sick	3772	29 (7,22)	2	Yes	6,124%
Sonar, Mines vs. Rocks	Sonar	208	60 (60,0)	2	No	46,635%
SPAM E-mail Database	Spam	4601	57 (57,0)	2	No	39,405%
Splice-junction Gene Sequences	Splice	3176	60 (0, 60)	3	No	48,120%
Tokyo SGI Server Performance Data	Tokyo1	958	43 (43,0)	2	No	36,117%

Three k-Nearest Neighbor parameters were evaluated. The first one is the number of nearest neighbors (k). In total, nine values were evaluated: 1, 3, 5, 7, 9, 11, 15, 21 and 27 neighbors. The second parameter is the distance function. Three distance functions were evaluated: HEOM (Heterogeneous Euclidean-Overlap Metric), HVDM (Heterogeneous Euclidean-VDM metric) and HMOM (Heterogeneous Manhattan-Overlap Metric). The third parameter is the nearest neighbors weighting function. Three weighting functions were evaluated: no weighting function, inverse of squared distance – namely simply **inverse** in this work – (Equation 3) and similarity (Equation 4).

Ten-fold cross-validation was used to partition the data sets into training and test sets. In order to reduce results variance by chance we have repeated this process ten times, i.e., we performed ten × ten-fold cross-validation. Due to the combinatorial explosion of the number of parameters values, data sets and resampling iterations, the k-Nearest Neighbor algorithm was run more than one-quarter million times.

As a consequence of this massive number of executions, tables with numerical results are not shown here[3]. In order to analyze whether there is a statistically significant difference among the parameters values, we ran the Friedman test[4]. Friedman test was run with three different null-hypotheses: (1) that the performance of the three weighting functions is comparable; (2) that the performance of all number of nearest neighbors is comparable; (3) that the performance of all distance functions is comparable. When the null-hypothesis is rejected by the Friedman test, at 95% confidence level, we can proceed with a post-hoc test to detect which differences among the methods are significant. We ran the Bonferroni-Dunn multiple comparisons with a control test.

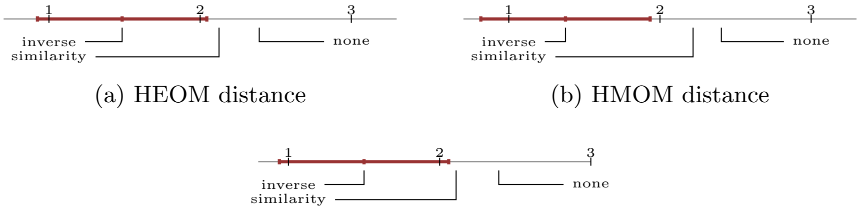
Regarding the performance of the three weighting functions, the inverse function outperformed the other two functions on more than 85% of the results, considering all k values[5] and the three distance function. Due to this expressive performance, we decide to compare the performance of the weighting functions first. As mentioned before, the first null-hypothesis is that the performance of all weighting functions is comparable. This null-hypothesis was divided into three, one for each distance function. For all distance functions, this null-hypothesis was rejected, at 95% confidence, by the Friedman test. Figure 1 shows the results of the Bonferroni-Dunn test. Inverse function significantly outperforms the other two weighting functions for all three distance functions.

The second null-hypothesis is that the performance of all k values are comparable. This null-hypothesis was divided into nine, one for each combination of distance and weighting functions. Friedman test rejected all but one null-hypothesis, at 95% confidence level. The only null-hypothesis not rejected was for inverse weighting and HVDM distance functions. In order to summarize the results, Figure 2 shows four graphs. The bars correspond to mean accuracy for all thirty-one data sets, and indicate a common pattern. Performance increases as k increases up to a maximum between $k = 5$ and $k = 11$. Then, for higher

[3] All tabulated results can be found at `http://www.icmc.usp.br/~gbatista/knn/`

[4] The Friedman test is a nonparametric equivalent of the repeated-measures ANOVA. See [9] for a thorough discussion regarding statistical tests in machine learning research.

[5] Actually, all k values but 1. Weighting functions cannot be applied when $k = 1$.



(a) HEOM distance

(b) HMOM distance

(c) HVDM distance

Fig. 1: Results of the Bonferroni-Dunn test considering the weighting functions. The thick line marks the interval of one critical difference, at 95% confidence level.

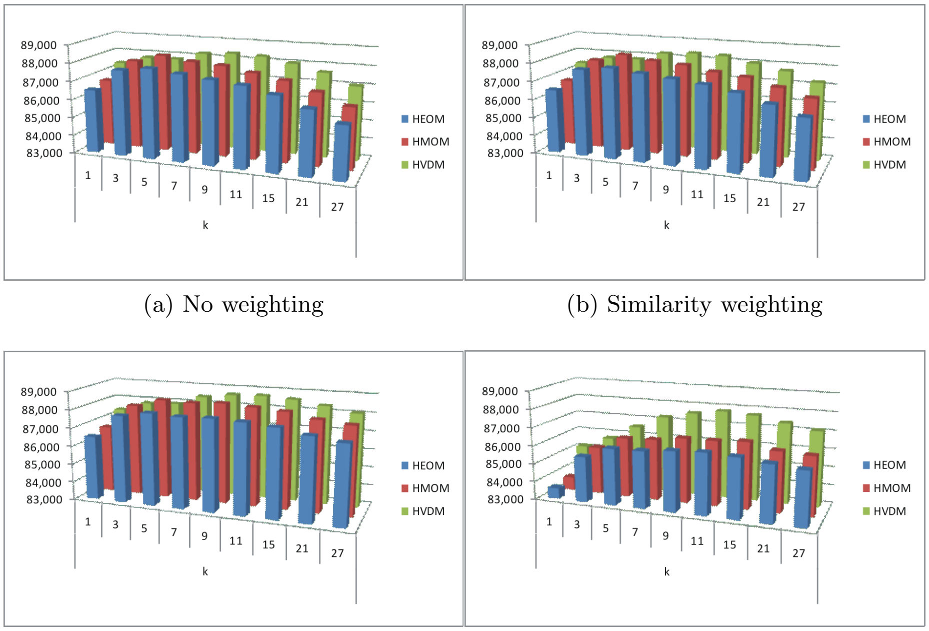
values of k the performance decreases. This curve was observed for all weighting and distance functions. Actually, as the inverse function severely penalizes the most distant nearest neighbors due to the square factor, the mean performance for higher k values does not decrease as steeper as for the other two weighting functions.

Therefore, the benefits of using inverse weighting function are two-fold. This weighting scheme is able to significantly increase the performance of k-Nearest Neighbor algorithm as showed by the rejection of the first null-hypothesis, and it can reduce performance losses caused by using higher k values, smoothing the performance of k-Nearest Neighbor across all k values.

Table 3 summarizes the results of all nine null-hypothesis. This table shows the weighting and distance functions, whether the null-hypothesis was reject by the Friedman test at 95% confidence level, the best ranked k value, and the k values outperformed by the best k value as pointed out by the Bonferroni-Dunn test. Note that the best k values are around 5 and 7. In addition, the values 1, 15, 21 and 27 are frequently outperformed by the best k value with 95% confidence level. Since we recommend the use of the inverse weighting function, $k = 5$ presented the best *mean* performance for HEOM and HMOM, and $k = 11$ presented the best *mean* performance for HVDM. It is important to note that even though $k = 11$ showed the best mean performance for HVDM, this parameter value was not able to significantly outperform the other k values for this specific combination of weighting and distance functions.

The third null-hypothesis that the performance of all distance functions is comparable was not rejected by the Friedman test. However, since HVDM can only contribute in the presence of qualitative attributes, we selected the subset of all data sets with at least one qualitative attribute. Once again, Friedman test did not reject the null-hypothesis. This is clearly a negative result for HVDM which has a more complex implementation and present a higher time complexity. Figure 2d illustrates graphically the performance of the three distance functions considering only the subset of data sets with at least one qualitative attribute.

As can be noted on Figure 2d, HVDM outperformed HEOM and HMOM for all values of parameter k. The explanation for this apparent inconsistency between Friedman test and Figure 2d relies on the accuracies obtained in each



(a) No weighting

(b) Similarity weighting

(c) Inverse weighting

(d) Inverse - data sets with qualitative attributes

Fig. 2: Mean accuracy for all data sets.

data set. HVDM performance has a high variance. For instance, using the results obtained with $k = 5$ for the splice data set, HEOM and HMOM obtained a 85.58% accuracy and HVDM obtained 94.14%, an expressive improvement. However, for the labor-neg-data data set, the performance of HEOM is 91.73% and for HMOM is 90.23% and for HVDM is only 83.47%. Therefore, even though HVDM can improve the mean performance for data sets with at least one qualitative attribute, we cannot recommend it in general. However, this measure was able to significantly increase the accuracy for some data sets, and it is certainly worth evaluating it for a data set with qualitative attributes at hand. Finally, we would like to comment that we analyzed the possibility that the performance improvement of HVDM is dependent of the percentage of qualitative attributes, in a way that data sets with many qualitative attributes would be greatly benefited by HVDM. However, this behavior was not characterized by the data sets used in the experiment.

5 Conclusion

This work analyses how the parameters affect the behavior of the k-Nearest Neighbor algorithm. As a consequence of the obtained results, we recommend

Table 3: Summary of statistical test results for the second null-hypothesis.

Weighting	Distance	Null-hypothesis	Best k value	Outperformed k values
None	HEOM	Rejected	7	1, 15, 21, 27
None	HMOM	Rejected	5	1, 15, 21, 27
None	HVDM	Rejected	7	27
Similarity	HEOM	Rejected	7	1, 15, 21, 27
Similarity	HMOM	Rejected	5	1, 15, 21, 27
Similarity	HVDM	Rejected	7	27
Inverse	HEOM	Rejected	5	1, 27
Inverse	HMOM	Rejected	5	1, 21, 27
Inverse	HVDM	Not rejected	11	-

the use of the inverse weighting function. This weighting function has two major benefits: it has an excellent *mean* performance, statistically outperforming the other two weighting functions; and it causes the parameter k to have a smooth influence over the classification performance of the algorithm, since it penalizes distant neighbors. We would like to emphasize that the highest accuracy results were obtained for k between 5 and 11 nearest neighbors. Since we recommend the use of the inverse weighting function, $k = 5$ presented the best *mean* performance for HEOM and HMOM, and $k = 11$ presented the best *mean* performance for HVDM. Finally, we were not able to find significant differences among the results presented by the three distance functions: HEOM, HMOM and HVDM. Since HVDM can only contribute to classification in the presence of qualitative attributes, we selected the subset of data sets with at least one qualitative attribute. However, no significant differences were observed in this data set subset as well.

References

1. Salzberg, S.: On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery **1**(3) (1997) 317–328
2. Mitchell, T.: Machine Learning. McGraw Hill, New York, NY, USA (1997)
3. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. Journal of Artificial Intelligence Research (JAIR) **6** (1997) 1–34
4. Stanfill, C., Waltz, D.: Instance-based Learning Algorithms. Communications of the ACM **12** (1986) 1213–1228
5. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
6. Kubat, M., Holte, R.C., Matwin, S.: Machine learning for the detection of oil spills in satellite radar images. Machine Learning **30**(2-3) (1998) 195–215
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal Artificial Intelligence Research (JAIR) **16** (2002) 321–357
8. Bucene, L.C.: Mineração de Dados Climáticos para Alertas de Geada e Deficiência Hídrica (2008) Tese de Doutorado, FEAGRI/UNICAMP.
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30