Non–Parametric Nearest Neighbor With Local Adaptation

Francisco J. Ferrer–Troyano, Jesús S. Aguilar–Ruiz, and José C. Riquelme

Department of Computer Science, University of Sevilla
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
{ferrer,aguilar,riquelme}@lsi.us.es

Abstract. The k–Nearest Neighbor algorithm (*k-NN*) uses a classification criterion that depends on the parameter k. Usually, the value of this parameter must be determined by the user. In this paper we present an algorithm based on the NN technique that does not take the value of k from the user. Our approach evaluates values of k that classified the *training* examples correctly and takes which classified most examples. As the user does not take part in the election of the parameter k, the algorithm is non–parametric. With this heuristic, we propose an easy variation of the *k-NN* algorithm that gives robustness with noise present in data. Summarized in the last section, the experiments show that the error rate decreases in comparison with the *k-NN* technique when the best k for each database has been previously obtained.

1 Introduction

In Supervised Learning, systems based on examples (CBR, Case Based Reasoning) are object of study and improvement from their appearance at the end of the sixties. These algorithms extract knowledge by means of inductive processes from the partial descriptions given by the initial set of examples or instances. Machine learning process is usually accomplished in two functionally different phases. In the first phase of *Training* a model of the hyperspace is created by the labelled examples. In the second phase of *Classification* the new examples are classified or labelled based on the constructed model. The classifier approached in this paper belongs to the family of the nearest neighbor algorithm (from here on *NN*) where the *training* examples are the model itself. *NN* assigns to each new query the label of its nearest neighbor among those that are remembered from the phase of *Training* (from here on the set T).

In order to improve the accuracy with noise present in data, the *k-NN* algorithm introduces a parameter k so that for each new example q to be classified the classes of the k nearest neighbors of q are considered: q will be labelled with the majority class or, in case of tie, it is randomly broken. Another alternative consists in assigning that class whose average distance is the smallest one or introducing a heuristically obtained threshold $k_1 < k$ so that the assigned class will be that with a number of associated examples greater than this threshold [12].

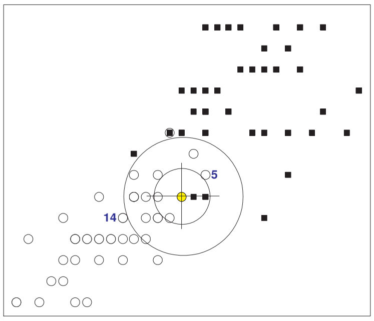

Fig. 1. The chosen value for k is decisive to classify a new example q by *k-NN* when this example is near the decision boundaries. For this example, the smaller value of the parameter k that classifies q correctly is 5.

Extending the classification criterion, the *k-NN*$_{wv}$ algorithms (Nearest Neighbor Weighted Voted) assign weights to the prediction made by each example. These weights can be inversely proportional to the distance with respect to the example to be classified [5, 7]. Therefore, the number k of examples observed and the metric used to classify a *test* example are decisive parameters. Usually k is heuristically determined by the user or by means of cross-validation [11]. The usual metrics of these algorithms are the Euclidean distance for continuous attributes and the Overlap distance for nominal attributes (both metrics were used in our experiments).

In the last years have appeared interesting approaches that test new metrics [15] or new data representations [3] to improve accuracy and computational complexity. Nevertheless, in spite of having a wide and diverse field of application, to determine with certainty when *k-NN* obtains higher accuracy than *NN* [2] and viceversa [8] is still an open problem. In [6] it was proven that when the distance among examples with the same class is smaller than the distance among examples of different class, the probability of error for *NN* and *k-NN* tends to 0 and $\frac{1}{2}$, respectively. But, not always this distribution for input data appears, reason why *k-NN* and *k-NN*$_{wv}$ can improve the results given by *NN* with noise present in the data.

In [13] the experimental results give rise to the two following hypotheses: a) Noisy data need large values for k; b) The performance of *k-NN* is less sensitive to the choice of a metric. Figure 1 illustrates this fact when the values of two attributes from the Iris database are projected on the plane. The X-axis measures the length of the petal and the Y-axis measures the width of the petal.

In [14] a study of the different situations in which *k-NN* improves the results of *NN* is exposed, and four classifiers are proposed (*Locally Adaptive Nearest Neighbor*, $localKNN_{ks}$) where for each new example q to be classified the parameter k takes a value k_q which is *similar* to the values that classified the M nearest neighbors e_q of q. Using a similar approach to Wettschereck's, we propose a classification criterion for new examples by taking different values for k according to the most frequent ones that classified the original examples correctly. To calculate such values the proximity of each example to its enemy is analysed, being the *enemy* the nearest neighbor with different class. A priori, if the example to be classified is near examples having different classes among them, it might be classified by few values. But if this example is surrounded by neighbors with the same class, it could be classified by different values.

This paper presents a method that reduces and classifies according to such a criterion with no need to analyse each particular case. In this way, the impure regions and the border instances are better analysed in order to provide greater robustness with noise present in the data.

In section 2 the proposed method and their computational complexity are detailed. Section 3 describes the experiments and the results from the UCI repository [4] and section 4 summarizes the conclusions.

2 Description of the Algorithm

2.1 Approach

By means of the *k-NN* algorithm, if a new example is near the decision boundaries, the resulting class depends on the parameter k. At worst, the percentages of examples of each class are similar at these regions. In such situation, the set formed by classifying values k_{e_i} associated with each example e_i at this region can be large or zero, i.e. some examples will not have any associated value k_{e_i} which classify it correctly by *k-NN*. So, this information (the classifying values associated with the nearest neighbors of a new query q) can be not relevant to classify a new query q by *k-NN*.

We not assume that it is possible to determine the values of the parameter k which allow to classify the examples in overlapped regions. However, we assume that it is possible to improve the accuracy if several times the *k-NN* algorithm is evaluated on these regions. The idea is as simple as to give more than one opportunity to the example that is to be classified . If the example to be classified is a central example this criterion will not have any effect. If it is a border example, the accuracy can improve. Thus, the disturbing effect caused by the noise and the proximity to enemies can be smoothed. The consequences of such a bias can be explained in three cases:

- If q is a central example, the majority class might almost always be the same one for each evaluation.
- If q is a noise example, either there will not be an associated value k_q that classifies q correctly or k_q will be large.
- If q is a border example, several evaluations can avoid the errors of classification.

Figures 2 and 3 illustrate these facts by means of projections on the plane of the values of two attributes of the Horse-Colic database. In the first case (Figure

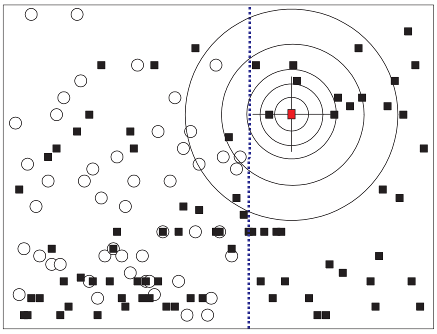

Fig. 2. Horse Colic database. If the new example to be classified is central, the majority class in each evaluation might be the same almost always.

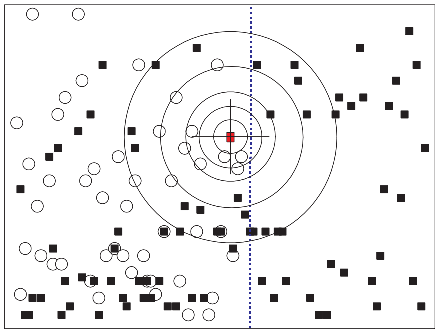

Fig. 3. Horse Colic database. If the new example to be classified is a border example, several evaluations of the *k-NN* algorithm can avoid some errors.

2) the value of k is slight relevant whereas in the second case (Figure 3) such value is critical. Therefore, our problem is to find either the limits $[k_{q_{min}}, k_{q_{max}}]$ between which the *k-NN* algorithm will be applied for each new example q or the values, which are not necessarily continuous, $\{k_{q_1}, k_{q_2}, \ldots\}$ from which is calculated the majority class. The method has been denominated fNN (k–Frequent Nearest Neighbors) since it takes the most frequent values of k among those that classified correctly the examples of each database. In this process, there are no parameters given by the user since these values are calculated locally for each database.

2.2 The Algorithm

Let n be the number of examples of the *Training* set T. Let $kNN(e,i)$ the i^{th} nearest neighbor of an example e within T. We denote $majorityClass(e, i..j)$ as the majority class between the i^{th} and the j^{th} neighbors of e. *fNN* associates with each example e_i two values:

1. $kCMin_i$: The smallest k that classifies correctly the example e_i by using the *k-NN* algorithm, such that (see Figure 4):

$$\forall j \in [1, kCMin_i) \;|\; Class\,(kNN\,(e_i, j)) = Class\,(e_i) \Rightarrow$$
$$\Rightarrow majorityClass\,(e_i, 1..j) \neq Class\,(e_i) \tag{1}$$

2. $kCMax_i$: If $kCMin_i$ was found, then $kCMax_i \geq kCMin_i$ and:

$$\forall j \in [kCMin_i, kCMax_i] \Rightarrow Class\,(e_i) = Class\,(kNN\,(e_i, j)) \tag{2}$$

With these values, a new value $kLim$ is calculated which satisfies the following property:

$$\forall e_i \in T, j \in [\min(kCMin_i), kLim] \Rightarrow \exists e_k \in T \;|\; kCMin_k = j \tag{3}$$

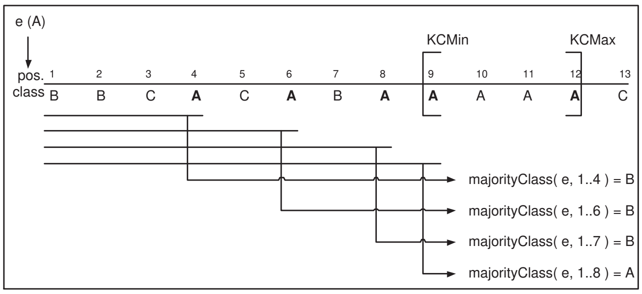

Fig. 4. Example of $kCMin$ and $kCMax$ for an example e with class A. The ties are broken with the nearest class. Majority class is not A from $k = 1$ to $k = 8$. When $k = 9$ the majority class is A ($kCMin_e = 9$). All the examples have class A from $k = 9$ to $k = 12$. Finally, the thirteenth neighbor of e has class C, so that $kCMax_e = 12$.

where $min(kCMin_i)$ is the least $kCMin_i$ value of each example.

Those examples e_i that either have not any associated value $kCMin_i$ or have an associated value $kCMin_i > kLim$ are considered outliers and they are removed. The resulting reduced set (from here on T_f) will be used as *Training* model for our algorithm.

In order to classify a new example q, the *k-NN* algorithm is applied several times to the same example q by varying the value of k. These values belong the interval $[\min(kCMin_i), kLim]$. The assigned label will be that among the nearest to q that is most frequent in the k_T evaluations of *k-NN*. Thus, the computational complexity of *fNN* is: $\Theta\left(n^2 \cdot (\log n + 1) + n \cdot (\log n + \delta + 2) + kLim^2\right)$.

In the first phase the set T_f is generated. The $n - 1$ nearest neighbors for the n examples of the *Training* set are ordered $(\Theta\,(n \cdot (n + n \cdot \log n)))$. So, for each example it is computed the distance to all the neighbors $(\Theta\,(n))$ and then the associated list is ordered $(\Theta\,(n \log n))$. After this procedure $kLim$ is found $(\Theta\,(n \cdot \delta)\,,\ \min(kCMin_i) \leq \delta \leq kLim)$ and the outliers are removed $(\Theta\,(n))$.

In the second phase a *test* example is classified $(\Theta\left(n \cdot (\log n + 1) + kLim^2\right))$. The pseudo code for the *fNN* algorithm is shown in Figure 5 where n is the original number of examples and c the number of different labels for the class.

3 Results

To carry out the method and the test, the Euclidean distance for continuous attributes and the Overlap distance for the nominal attributes were used. The values of the continuous attributes were normalized in the interval [0,1]. Examples with missing-class was removed and attributes with missing-values was treated with the mean or mode, respectively. *fNN* was tested on 20 databases from the Machine Learning Database Repository at the University of California, Irvine [4]. In order to reduce statistical variation, each experiment was executed

```
function fNN(Train:SET; query:INSTANCE) return (label:Integer)
var
  T_f: Set;  k,k_Lim,label: Integer;  k_Vect: VECTOR[n][2] Of Integer
  frequencies: VECTOR[c] Of Integer
beginF
  // Calculating kCMin, kCMax for each example.
  Calculate_MinMax_Values_K(Train, k_Vect)
  // Finding the limits for the evaluations of kNN.
  Calculate_Limits(k_Vect,k_Lim)
  // Removing the examples whose kCMin does not belong to [1,k_Lim].
  T_f:= Delete_Outliers(Train, k_Lim)
  for k:= Min(kCMin) to k_Lim
   label:= Classify_KNN(query,T_f,k)
   frequencies[label]:= frequencies[label]+ 1/Average_Dist(query,label,k)
  return(frequencies.IndexOfMaxElement())
endF
```

Fig. 5. Pseudo code for *fNN*.

by means of 10-folds cross-validation. *fNN* was compared with *k-NN* using 25 different values of k (the odd numbers belonging to interval [1, 51]). This limit was fixed after observing for all databases how the accuracy decreased from a value near the best k for each database (being 33 the maximum value for *Heart Cleveland*) database. In Table 1 is reported the main results obtained. The average-accuracy with the associated standard deviation and the computational cost by means of *fNN* is showed in Columns 2a and 2b respectively. The *k-NN* algorithm is included for comparison using the best k for each database (Column 3a) and the best average-value (k=1) for all databases (Column 1a). Both computational cost for *k-NN* were very similar and they are showed in Column 1b. Column 3b shows the best value of k for each database by *k-NN*. Column 2c show the size of T_f regarding the *Training* set and Column 2d show the values of $kLim$ for each database, i.e. the limit for k by *fNN*. The databases marked with * mean an improvement of *fNN* regarding *1-NN* by means of t-Student statical test using $\alpha = 0.05$. We can observe in Table 1 that *fNN* obtained better precision than *1-NN* for 13 databases where the best k for the *k-NN* algorithm was a high value, so that:

- If $KLim < k_{best}$ for *k-NN*, then *fNN* provides higher accuracy than *1-NN*.
- The percentage of examples that are excluded from T_f is a minimum error bound for *k-NN*.

4 Conclusions

An easy variation of the *k-NN* algorithm has been explained and evaluated in this paper. Experiments with commonly used databases indicate that exits do-

Table 1. Results for 20 databases from the UCI repository by using 10-folds cross-validation. Column 1 shows the average accuracy with the standard deviation and the computational cost by *k-NN* with k=1 (the best value for all databases). Column 2 shows the same percentages obtained by *fNN*, the percentage of examples retained from the *Training* set and the value of kLim, i.e. the limit for *k* by *fNN* . Column 3 shows the best accuracy with the standard deviation by the *k-NN* algorithm when the best *k* is found. This best *k* was looked for in the odd numbers belonging to interval [1,51].

	1-NN		**fNN**				**best k-NN**	
Domain	Pred. Acc.	Time	Pred. Acc.	Time	%Ret.	kLim	Pred. Acc.	best k
Anneal	91.76 ±2.4	1.10	90.32 ±1.8	15.7	96.5	9	91.76 ±2.4	1
Balance Scale*	77.76 ±4.8	0.25	89.44 ±1.5	4.0	89.7	11	89.76 ±1.4	21
B. Cancer (W)	95.56 ±2.2	0.20	96.57 ±1.7	2.91	92.1	9	96.85 ±2.0	17
Credit Rating*	80.72 ±2.2	0.56	87.11 ±1.8	7.59	91.9	7	87.68 ±1.6	13
German Credit	72.29 ±2.9	1.42	74.61 ±3.4	18.2	89.1	21	73.09 ±4.2	17
Glass	70.19 ±2.0	0.04	69.16 ±1.3	0.64	84.4	9	70.19 ±2.0	1
Heart D. (C)*	74.92 ±2.5	0.09	81.52 ±2.0	1.32	89.3	13	83.17 ±2.7	33
Hepatitis*	81.29 ±0.8	0.03	87.11 ±0.9	0.43	88.5	9	85.16 ±1.0	7
Horse Colic	67.93 ±3.0	0.22	69.02 ±1.6	2.90	83.7	11	70.38 ±2.6	7
Ionosphere	86.61 ±1.9	0.29	85.18 ±2.0	3.66	91.1	13	86.61 ±1.9	1
Iris	95.33 ±0.8	0.01	96.00 ±0.8	0.29	95.6	1	97.33 ±0.8	15
Pima Diabetes	71.21 ±5.0	0.56	74.09 ±3.9	7.62	87.7	15	75.52 ±4.2	17
Primary Tumor*	35.69 ±1.3	0.05	42.19 ±1.5	0.81	54.9	11	43.07 ±1.5	29
Sonar	86.54 ±1.5	0.16	86.06 ±1.2	1.91	94.1	5	86.54 ±1.5	1
Soybean	90.92 ±3.5	0.65	91.07 ±2.9	8.42	95.9	13	90.92 ±3.5	1
Vehicle	70.06 ±3.0	1.12	69.97 ±2.9	13.5	89.9	13	70.06 ±3.0	1
Voting	92.18 ±1.6	0.07	92.64 ±1.3	1.18	95.1	3	93.56 ±1.0	5
Vowel	99.39 ±0.9	1.19	98.79 ±1.3	16.1	99.1	1	99.39 ±0.9	1
Wine	96.07 ±1.1	0.03	96.63 ±0.7	0.54	98.1	5	97.75 ±0.7	31
Zoo*	97.03 ±0.6	0.01	93.07 ±1.1	0.19	95.6	1	97.03 ±0.6	1
Average	81.67 ±2.2	0.40	83.53 ±1.8	5.39	90.11	9	84.29 ±2.0	11

mains where classification is very sensitive to the parameter k by using the *k-NN* algorithm. For these input data, we could summarize several aspects:

- Without the need of parameter, *fNN* is a reduction and classification technique that keeps the average accuracy of the *k-NN* algorithm.
- *kLim* and the size of T_f compared to the size of T are an approximated indicator for the percentage of examples that cannot be correctly classified by the *k-NN* algorithm.
- The reduction of the database is very similar to the reduction that makes *CNN* [15], so that *fNN* is less restrictive than *CNN*. With large databases, this reduction can accelerate the learning process for the *k-NN* algorithm.

5 Future Work

Actually we are testing *fNN* with other classifiers. Particularly, we have chosen two systems, C4.5 [9] and HIDER [10], which generate decision trees and axis–parallel decision rules, respectively. Due to *fNN* makes a previous reduction, we have chosen the method EOP [1], which reduces databases conserving the decision boundaries that are parallel to the axis.

6 Acknowledgments

The research was supported by the Spanish Research Agency CICYT under grant TIC99-0351. We thank the anonymous reviewers for providing helpful advice and suggestions that improved this work.

References

1. J. S. Aguilar, J. C. Riquelme, and M. Toro. Data set editing by ordered projection. In *Proceedings of the* 14^{th} *European Conference on Artificial Intelligence*, pages 251–255, Berlin, Germany, August 2000.
2. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
3. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for nearest neighbor searching. In *Proceedings of 5th ACM SIAM Symposium on discrete Algorithms*, pages 573–582, 1994.
4. C. Blake and E. K. Merz. Uci repository of machine learning databases, 1998.
5. S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
6. T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, 1967.
7. S.A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6, 4:325–327, 1975.
8. R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11:63–91, 1993.
9. J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, California, 1993.
10. J. C. Riquelme, J. S. Aguilar, and M. Toro. Discovering hierarchical decision rules with evolutive algorithms in supervised learning. *International Journal of Computers, Systems and Signals*, 1(1):73–84, 2000.
11. M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36:111–147, 1974.
12. I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, June 1976.
13. C. Wettschereck. *A Study of Distance-Based Machine Learning Algorithms*. PhD thesis, Oregon State University, 1995.
14. D. Wettschereck and T.G. Dietterich. Locally adaptive nearest neighbor algorithms. *Advances in Neural Information Processing Systems*, (6):184–191, 1994.
15. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificiall Intelligence Research*, 6(1):1–34, 1997.