Performance Comparison between Edited kNN and MQ-RBFN for Regression and Classification Tasks

J. E. B. Maia, V. R. S. Laboreiro, F. E. Chaves, F. J. A. Maia, T. G. N. Silva, T. N. Ferreira
Universidade Estadual do Ceará - UECE - Itaperi
Estatística e Ciência da Computação
Email: jose.maia@uece.br, {victor.laboreiro,edvanchaves,felipe.ja.maia,thi.nepo,thiagonascimento.uece,}@gmail.com

***Abstract*—Supervised learning techniques can be roughly grouped into lazy learning or eager learning. Lazy learning and eager learning have very different properties and are suitable for different applications. In this paper we evaluate properties of the two types of learning using a representative distance based algorithm for each class, namely, kNN (k-nearest neighbors) and RBFN (Radial Basis Function Network). In addition, an edition algorithm (SPAM - Supervised Partitioning Around Medoids) is used to reduce the labeled dataset. Our experiments for classification and regression tasks, using 12 public datasets show that prototype selection algorithms typically used with kNN are good alternatives for selection of centers of RBFN when to optimize the number of centers is not the relevant criterion. The experiments also show that the RBFN generally perform better than Edited kNN.**

I. Introduction

The supervised learning techniques (classification or regression) can be roughly grouped into lazy learning or eager learning. In lazy learning [2] little or no effort is expended during the training phase because the effort is postponed until the generalization phase to a new instance. Thus, no predictive model is constructed in advance such as a neural network or a decision tree. In eager learning [2], in turn, focuses on the effort to build a concise and finished predictive model using training cases. Such a model should, to perform well, cover the entire input space focusing to achieve greater accuracy in regions of higher probability. Typical examples of lazy learning and eager learning are, respectively, the kNN [6] and RBFN [9] algorithms.

When the target function is very complex but can be approximated using a combination of several local functions, lazy learners such as k-nearest neighbor (kNN) are known to achieve good results. However, as a downside, lazy learners have high requirements in terms of storage and processing time which may limit their use in real life applications. In order to avoid such problems and possibly improving the algorithm results by avoiding noise and overfitting, an viable approach is to edit the training set. In this paper, we call a kNN algorithm with the training set reduced by some method of redundancy and outliers elimination as Editing kNN.

Radial Basis Function Networks (RBF Networks), on the other hand, have lower classification time requirements because they produce global optimization of the target function during the training stage. Even though the RBFN is an eager learner method, it approximates the target function using a combination of several local functions. Therefore, RBF Networks combine the advantages of lazy and eager learners. Studying this structural property of RBF Networks is the main motivation for this work.

RBF Networks and kNN are Distance-based algorithms. Distance-based algorithms are machine learning algorithms that store internal parameters and a number of exemplars. They compute the output for a given instance exclusively from a combination of the internal parameters and the distance between that instance and each exemplar. To accomplish this task, RBFN uses knowledge of the global structure of the task to build an interpolation while kNN works exclusively with local information. However, it has not been conclusively shown that any global location algorithm consistently outperforms every other algorithm in any specific task [5].

In this study, we investigated the properties and performance of both supervised learning algorithms (Edited kNN and RBFN) on classification and regression tasks. The goal, therefore, is to gain insight on the comparative properties of lazy learning and eager learning.

The remaining sections are organized as follows. The algorithms used in this work are briefly described in Section II. Section III presents the results together with the conceptual discussion of the results. Finally, Section IV concludes this paper.

II. The Algorithms

The methodology of this research is as follows. Each dataset is split into training set and test set. An algorithm (SPAM) is applied to edit instances of the training set. Reduced training sets with 20%, 40%, 60% and 80% of the initial size have been generated. Along with the initial dataset they form five different training sets. The performance of the algorithm kNN for k = 1, 3 and 5, based on each edited set is compared against the performance of RBFN. The RBFN is trained with the initial training set with the RBF centers having the same edited sets used by kNN. The used RBF activation function is the multiquadric. The following subsections describe briefly the algorithms mentioned.

A. kNN - k Nearest Neighbors

The kNN algorithm is one of the most widely used lazy learning approaches [2] that uses a non-parametric method for classifying patterns. This popularity is primarily due to its simplicity and intuitivity. It is a powerful classification algorithm capable of solving complex problems. Given a set of patterns X to be estimated, and $x_i \in X$, the k-Nearest

Neighbor procedure classifies x_i into one of $\omega_C \in \Omega$ classes, is as follows [1]:

- Determine the k-nearest training data vectors to the pattern x_i using an appropriate distance metric, such as, e.g., the Euclidean Distance, Mahalanobis Distance or Manhattan Distance.
- The estimated label $\hat{\omega}_i$ should be assigned as the most frequent label among the k-nearest neighbors of x_i.

The choice of an adequate value for parameter *k* is not trivial since larger values tend to reduce the classifier sensitivity to noise and smaller values may cause the neighborhood to not extend to the domain of other classes. In order to choose an optimal parameter *k*, a cross-validation procedure is usually performed [1].

The kNN can also be used for regression problems. Once determined the vicinity of the point, an local interpolation model is constructed to estimate the output. A simple implementation of kNN regression consists in assigning the property value of the object to be the average of the values of its k-nearest neighbors. Another approach uses an inverse distance weighted average of the k-nearest neighbors. Also it may use more sophisticated methods. For example, local least squares regression [7] or formulating an local optimization problem to determine the output [8]. Regression through kNN may use the same distance functions as the kNN classifier. This study used the inverse distance weighted average.

B. Nearest Neighbor Editing (Prototype Selection)

Often the set of prototypes used in the training contains noise or irrelevant information, which hinder the process of training the classifier. These problematic prototypes can be removed from the training set using a selection process known as prototype selection. The prototype selection can also act by selecting the prototypes most significant for the training process. Selecting a set of examples that generates a template with the best accuracy, for example. The algorithm chosen for this work is the SCE (Supervised Clustering Editing) [4]. The SCE uses the second method mentioned. Selecting a number k from the original examples so that the most significant elements are selected. The SCE algorithm divides the prototypes into clusters using a clustering algorithm, chooses the most significant element of each cluster and these prototypes will be selected by the algorithm. The clustering algorithm selected was the supervised algorithm SPAM (Supervised Partitioning around Medoids) [11] (a variation of the clustering algorithm PAM - Partitioning around Medoids [10]), so we could get the exact number of clusters generated.

C. MQ-RBFN: Multi-Quadric Radial Basis Function Network

The Multi-Quadric Radial Basis Function Network [1] consists in a single hidden layer neural network where each node located in such layer computes a Multi-Quadric activation function. Furthermore, each of the K neurons in the hidden layer is assigned, during the training phase, to a center value μ_k in order to compute the distance $d_{n,k}$ between an input pattern $x_n \in X$ and μ_k.

$$\phi(d_{n,k}) = \sqrt{(d_{n,k}^2 + c^2)} \tag{1}$$

The Multi-Quadric activation function $\phi(d_{n,k})$ (Eq. 1), therefore, outputs the computed value for a given scalar constant c to the output layer. This work adopted c = 0. Finally, the P neurons on the last layer perform a weighted sum of $\phi(d_{n,k})$ with the weights $w_{p,k}$ (Eq. 2).

$$\hat{y}_{n,p} = \sum_{k=0}^{K} w_{p,k}\phi(d_{n,k}) \tag{2}$$

Since the input values are directly sent to the nodes at the hidden layer, training an MQ-RBFN consists of defining the μ and c values for each neuron located at the hidden layer and finding suitable weights $w_{p,k}$. An usual method to define μ is through a clustering algorithm, e.g., k-Means, PAM or SPAM. The last two ones were chosen to perform such task. There are several different approaches detailed in the literature that finds suitable values for the neurons weights. In this paper, Singular Value Decomposition is performed in order to find the Moore-Penrose pseudo-inverse which is used to solve the following linear system:

$$W = \Phi^{+}Y \tag{3}$$

Where N represents the number of training patterns; Φ is a $N \times K$ matrix whose element $\Phi_{i,j}$ represents $\phi_(d_{i,j})$, i.e., the distance between the *i*-th input pattern x_i and the *j*-th center μ_j; W represents a $K \times P$ matrix whose element $w_{i,j}$ represents the weight between the *j*-th neuron at the hidden layer and the *i*-th neuron at the output layer; and Y is a $N \times P$ matrix whose element $\omega_{i,j}$ represents the *j*-th desired output for the *i*-th training pattern x_i.

III. Results and Discussion

The experiments were performed using 6 public datasets for classification and other 6 public datasets for regression. Each dataset was randomly divided in training set (90%) and test set (10%). Then, each training set from each dataset was submitted to a prototype selection algorithm that allows a fixed number K of prototypes to be selected. The values of K were 20%, 40%, 60%, 80% and, of course, 100% of the total of instances in the training set. The experiment was repeated 10 times.

The learning sets are normalized using z-score, and the test sets are normalized using the corresponding mean and variance from the learning set. Statistical evaluation of the results used the two-sample t-test [1] with a significance level equal to 5%. The null hypothesis is that the MSE or Accuracy distributions are independent random samples from normal distributions with equal means and equal but unknown variances. The alternative hypothesis is that the means are not equal.

After that, the evaluated algorithms were executed for each value of K to generate the statistical values that can be seen in these sections. A description of the datasets can be found in the Tables I and II. The datasets can be found in http://sci2s.ugr.es/keel/datasets.php.

Tables III and IV presents the results for each of the classification and regression tasks, respectively, for $K/N = 0.2$. Each table entry shows three values: the mean, the standard deviation and the *p*-value obtained with the t-test. For all tests the reference sample is that one achieved by RBFN-PAM, i.e., the t-test of each column (except the one from RBFN-PAM) always matches with the RBFN-PAM result. In order to facilitate the tables comprehension, each table cell indicates, using a symbol, if the hypothesis has been rejected ($\times$) or not ($\checkmark$).

Observing the classification results at Table III, it is possible to notice that RBFN performance is almost unaltered when its centers are selected using an unsupervised method (PAM) or an supervised one (SPAM). Furthermore, as shows the statistic avaliation presented in the last 3 columns, the null hyphotesis is rejected only in 3 of 18 cases indicating that RBFN performance is superior, in such experiments, than the Edited kNN using SPAM.

On regression experiments, Table IV shows that the null hypothesis is not rejected in 6 of the 18 presented cases. These 6 cases focuses in two datsets: ForestFires and MachineCPU. Therefore, the RBFN mean performance is also superior than the one achieved by kNN for regression datasets. This table also shows that, in this case, the RBFN performance does not change notably when using unsupervised or supervised center selection approaches.

The plots at Figures 1 to 4 shows the effect of varying the ratio K/N. Since the same behavior is noticeable for the other datasets, they were ommited in this paper for the sake of brevity. Figures 1 and 2 shows the regression results of Stocks and Treasaury datasets. The RBFN sistematically achieves better performance than the kNN. Notice that for small values of K/N, the RBFN MSE is almost one order of magnitude lower than the 1NN.

The classification results for the Glass and Dermatology datasets are presented at Figures 3 and 4. For both cases, the RBFN accuracy keeps with small variation along the entire range of K/N. On the other hand, as the ratio K/N increases, the kNN accuracy improves considerably at the Glass dataset, but, at the Dermatology dataset, its accuracy suffers no considerable changes for the same range.

When analysing only the Edited kNN algorithm, it is possible to see that the parameter *k* is hard to be tuned. While its performance improves as the *k* increases on classification tasks such as Glass or Yeast, however, a considerable performance reduction is noticeable at the MachineCPU regression dataset. Inverse behaviors can also be found on both tables. Notice that the RBFN behavior presents a very regular performance whether is at classification or regression tasks, and whether is at for all datasets or even at the entire range of K/N.

IV. Conclusion

The second and third columns of the Tables III and IV confirm the null hypothesis that the RBFNs with centers selected by the supervised algorithm (SPAM) or unsupervised (PAM) have statistically comparable performance. On the other hand, the three last columns of Tables III and IV show that the RBFN performance is consistently greater than kNN. It is important to note that the use of 20% of the training data as RBFN centers is only suitable when the number of RBFs is not a relevant restriction. Note, however, that the adopted approach used the Multi-Quadric RBFN which has lower time

TABLE I. Datasets used for classification

	#Attributes	**#Instances**	**#Classes**
dermatology	34	358	6
glass	9	214	7
heart	13	270	2
wdbc	30	569	2
wine	13	178	3
yeast	8	1484	10

TABLE II. Datasets used for regression

	#Attributes	**#Instances**
autoMPG8	7	392
baseball	16	337
forestFires	12	517
machineCPU	6	209
stock	9	950
treasury	15	1049

TABLE III. Comparison between Edited kNN and MQ-RBFN for Classification (K/N = 0.2)

Dataset	RBFN (PAM)	RBFN (SPAM)	1NN (SPAM)	3NN (SPAM)	5NN (SPAM)
Dermathology	**97.18%**	96.92%	92.75%	94.42%	93.33%
	2.68	2.81	4.41	4.16	4.57
		✓(.833)	×(.023)	✓(.135)	×(.051)
Glass	**70.91%**	69.57%	54.35%	51.30%	43.91%
	11.70	9.61	9.45	9.57	7.79
		✓(.781)	×(.002)	×(e-4)	×(e-6)
Heart	**84.44%**	84.07%	78.89%	78.52%	79.26%
	7.96	7.62	9.88	6.00	9.91
		✓(.916)	✓(.206)	×(.088)	✓(.240)
WDBC	**97.71%**	97.37%	93.68%	92.98%	92.46%
	2.36	2.07	4.39	4.22	5.61
		✓(.735)	×(.032)	×(.011)	×(.024)
Wine	**98.33%**	**98.33%**	91.67%	92.22%	92.78%
	2.68	2.68	7.05	7.03	6.44
		✓(1.0)	×(.017)	×(.025)	×(.027)
Yeast	59.10%	**59.66%**	47.25%	53.22%	53.83%
	3.43	2.73	3.98	2.17	2.32
		✓(.690)	×(e-7)	×(e-5)	×(e-5)

TABLE IV. Comparison between Edited kNN and MQ-RBFN for Regression (K/N = 0.2)

Dataset	RBFN (PAM)	RBFN (SPAM)	1NN (SPAM)	3NN (SPAM)	5NN (SPAM)
AutoMPG8	7.6	**7.3**	1.4e+1	1.3e+1	1.3e+1
	2.5	2.8	4.2	3.5	3.8
		✓(.811)	×(e-04)	×(.001)	×(e-04)
Baseball	5.1e+5	**5.0e+5**	1.2e+6	7.6e+5	7.4e+5
	1.4e+5	2.0e+5	4.4e+5	3.1e+5	2.4e+5
		✓(.841)	×(e-04)	×(.038)	×(.023)
ForestFires	5.0e+3	**4.3e+3**	1.1e+4	5.6e+3	4.9e+3
	7.2e+3	7.4e+3	1.1e+4	7.1e+3	7.4e+3
		✓(.887)	✓(.150)	✓(.690)	✓(.860)
MachineCPU	3.8e+3	**3.5e+3**	5.0e+3	6.1e+3	8.6e+3
	4.9e+3	3.6e+3	4.0e+3	6.7e+3	9.9e+3
		✓(.880)	✓(.400)	✓(.300)	✓(.160)
Stock	**5.4e-1**	8.8e-1	1.6	2.2	3.2
	1.2e-1	2.0e-1	5.6e-1	9.6e-1	1.6
		×(e-4)	×(.003)	×(.002)	×(.001)
Treasury	**4.0e-2**	4.2e-2	1.5e-1	1.5e-1	1.7e-1
	2.0e-2	1.9e-2	7.8e-2	8.6e-2	9.1e-2
		✓(.812)	×(.002)	×(.003)	×(.002)

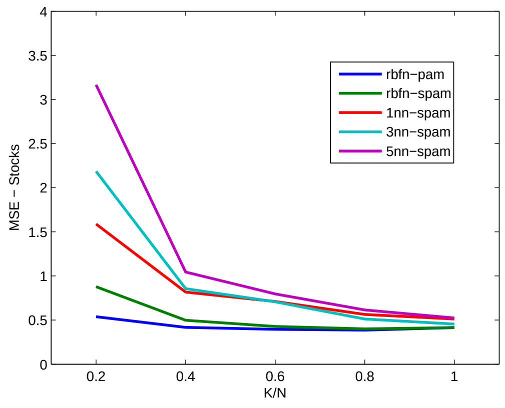

Fig. 1. MSE results for Stocks dataset for K/N = 0.2 to 1.0.

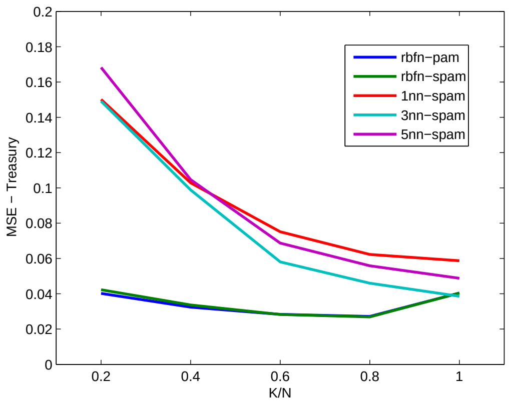

Fig. 2. MSE results for Treasury dataset for K/N = 0.2 to 1.0.

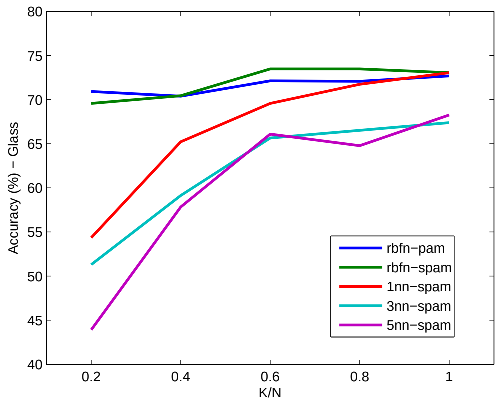

Fig. 3. Accuracy results for Glass dataset for K/N = 0.2 to 1.0.

complexity than the usual Gaussian RBF.

When varying the selected prototypes set cardinality and using them for training the edited kNN and also for defining the RBFs centers in the RBFN, the RBFN performance was slightly different. The MSE graphs for regression (Figures 1 and 2) shows that for smaller edited datasets the RBFN performance is clearly better than the ones achieved by kNN.

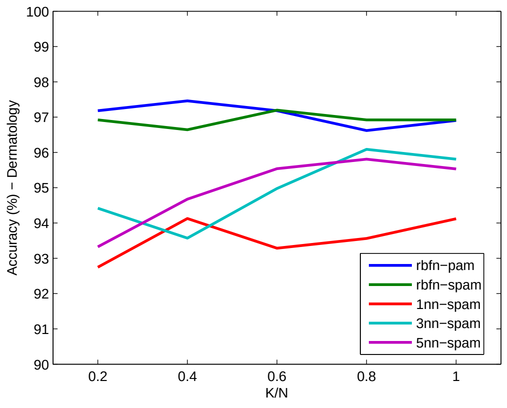

Fig. 4. Accuracy results for Dermatology dataset for K/N = 0.2 to 1.0.

As the edited set cardinality increases, the performance gap between the kNN and RBFN gets smaller. In contrast, the average accuracy of RBFN for the classification (Figures 3 and 4) is greater than kNN for the full scale datasets in analysis.

One can thus conclude that the local approximation properties of the kNN (a lazy method) were not enough to overcome the global approximation properties of RBFN (an eager method) when the number of RBFs is great.

Note in Tables III and IV that, although RBFN has the overall best performance, the kNN classifier, in some cases, was able to achieve similar performance. The next step in this study is to extend to larger and broader datasets.

References

[1] A. R. Webb and K. D. Copsey, *Statistical Pattern Recognition*, 3rd Edition, John Wiley & Sons, Ltd.,2011.

[2] D. W. Aha, *Lazy learning: special issue editorial*, Artificial Intelligence Review, 11, pp. 710, 1997.

[3] B. V. Dasarathy, *Nearest Neighbor Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, Los Alamitos, CA, 1991.

[4] C. F. Eick, N. Zeidat and R. Vilalta, *Using Representative-Based Clustering for Nearest Neighbor Dataset Editing*, 4th IEEE International Conference on Data Mining, Brighton, UK, pp. 375-378, 2004.

[5] D. Wettschereck, *A study of distance-based machine learning algorithms*, Doctoral dissertation, Department of Computer Science, Oregon State University, Corvallis, OR, 1994.

[6] E. K. Garcia, S. Feldman, M. R. Gupta and S. Srivastava , *Completely Lazy Learning*, IEEE Trans. on Knowledge and Data Engineering, vol. 22, no. 9, pp. 1274-1285, 2010.

[7] M. Birattari, G. Bontempi and H. Bersini, *Lazy learning meets the recursive least squares algorithm*, Proc. of the 1998 conference on Advances in neural information processing systems II, pp. 375-381, 1999.

[8] G. Bontempi, M. Birattari and H. Bersini, *Lazy learning for modeling and control design*, Internat. J. Control, 72(7/8), pp. 643-658, 1999.

[9] J. M. Valls, I. M. Galvan and P. Isasi, *Learning radial basis neural networks in a lazy way: A comparative study*, Neurocomputing 71, pp. 2529-2537, 2008.

[10] L. Kaufman, P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, 2005.

[11] C. F. Eick, N. Zeidat and Z. Zhao, *Supervised Clustering Algorithms and Benefits*, 16th IEEE Int. Conf. on Tools with Artificial Intelligence ICTAI 2004, 2004