On Optimizing *Locally* Linear Nearest Neighbour Reconstructions Using Prototype Reduction Schemes*

Sang-Woon Kim[1] and B. John Oommen[2]

[1] *Senior Member, IEEE*. Dept. of Computer Science and Engineering, Myongji University, Yongin, 449-728 South Korea
kimsw@mju.ac.kr

[2] *Chancellor's Professor* ; *Fellow : IEEE* and *Fellow : IAPR*. School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. Also an Adjunct Professor with the University of Agder in Grimstad, Norway
oommen@scs.carleton.ca

Abstract. This paper concerns the use of Prototype Reduction Schemes (PRS) to optimize the computations involved in typical k-Nearest Neighbor (k-NN) rules. These rules have been successfully used for decades in statistical Pattern Recognition (PR) applications, and have numerous applications because of their known error bounds. For a given data point of unknown identity, the k-NN possesses the phenomenon that it combines the information about the samples from *a priori* target classes (values) of selected neighbors to, for example, predict the target class of the tested sample. Recently, an implementation of the k-NN, named as the Locally Linear Reconstruction (LLR) [11], has been proposed. The salient feature of the latter is that by invoking a quadratic optimization process, it is capable of systematically setting model parameters, such as the number of neighbors (specified by the parameter, k) and the weights. However, the LLR takes more time than other conventional methods when it has to be applied to classification tasks. To overcome this problem, we propose a strategy of using a PRS to efficiently compute the optimization problem. In this paper, we demonstrate, first of all, that by completely discarding the points not included by the PRS, we can obtain a reduced set of sample points, using which, in turn, the quadratic optimization problem can be computed far more expediently. The values of the corresponding indices are comparable to those obtained with the original training set (i.e., the one which considers all the data points) even though the computations required to obtain the prototypes and the corresponding classification accuracies are noticeably less. The proposed method has been tested on artificial and real-life data sets, and the results obtained are very promising, and has potential in PR applications.

1 Introduction

It is well known that the optimal classifier is the one that invokes the Bayes decision rule. If the *a priori* density functions were easily computable, and the class conditional

* The second author was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada. This work was generously supported by the National Research Foundation of Korea funded by the Korean Government (NRF-2010-0015829).

J. Li (Ed.): AI 2010, LNAI 6464, pp. 153–163, 2010.

densities were truly of a classical well-defined nature (for example, of the exponential family), the tasks of training and testing a pattern recognition/classification system would be trivial. In practice, however, these distributions are far from ideal, and consequently, the science and art of PR has had to develop various non-parametric methods for training and testing. The most elementary of these, and yet the most well-developed, constitute the Nearest Neighbor (NN) family of classifiers[1].

The idea behind the NN rules is age-old and is essentially encapsulated in the axiom that the information about a particular sample point can be gleaned from its nearest neighbors. Traditionally, the consequent decision rule merely performs a majority decision based on the decision of the closest k neighbors. The beauty of such a scheme is that the decision rule asymptotically attains the accuracy of the Bayes rule as the number of neighbors, k, is increased. More recently, to yield even more accurate results (for any given value of k), researchers have proposed that the neighbors need not be assigned equal weights. Rather, the question is that of modeling every feature point as a convex combination of its k neighbors, and from this perspective, the crucial question is that of determining the weights that are to be assigned to these neighbors.

The most important paper in this regard is *probably* the one due to Kang and Cho [11], referred to as the Locally Linear Reconstruction (LLR) method. The fundamental idea behind the LLR, though simple, is quite intriguing, and it involves a quadratic optimization strategy explained presently. The salient feature of this scheme is that by invoking this optimization, one can systematically determine the model parameters, such as the number of neighbors (k) and the corresponding weights. However, the LLR, as proposed in [11], is computationally intensive. This is where our research comes into the picture: To tackle the computational burden, we propose a strategy of using a Prototype Reduction Scheme (PRS) to quickly and efficiently approximately compute the optimization problem. We formulate this in the paragraph below.

Rationale for the paper: We start with the premise that it is advantageous to compute the above mentioned optimization. However, we seek a strategy by which the associated computational burden can be reduced. Thus, in this paper, we propose a technique[2] for the fast computation of the reconstruction problem, and in particular, for the various classification applications. We advocate that rather than compute the reconstruction for the entire data set, the data be first reduced into a smaller representative subset using a PRS [2], [6], and that the reconstruction (classification) be achieved by invoking the corresponding method on *this* reduced data set. Thus, by completely discarding the points not included by the PRS, we can obtain a reduced set of sample points, using which, in turn, one can solve the quadratic optimization problem. The reader will observe, at once, that this can reduce the computational burden drastically, because the number of points chosen by the PRS is usually a small *fraction* of the total number of points found in the original data set. Our hypothesis, i.e., that the PRS can be effectively used to noticeably reduce the computations and yet yield almost as accurate results, has been verified by testing on benchmark real-life and artificial data tests, as we

[1] Some strategies for speeding up the kNN have been reported in the literature, e.g., in [14].

[2] As a *prima facie* case, to justify the hypothesis of [11], we only consider the two-class problem. The effective definition and computation of the measures for the multi-class problem are open.

shall presently explain. The geometric aspect of this strategy is the following: Although the reconstructed samples are obtained by using the prototypes procured by invoking a PRS, these reconstructed points do not *individually* "optimally" represent their original counterparts. However, *collectively*, they are the best locations for the k-NNs of the points in the training set, which can, in turn, *collectively* represent the points for testing purposes too. This is truly an interesting feature!

2 An Overview : LLR and PRS

Locally Linear Reconstruction: In this section, we briefly explain the LLR [11] for pattern classification and recognition (as considered for instance-based learning), and in particular for the k-NN. The main idea behind LLR originates from the concept of the locally linear embedding (LLE) [16], which is one of widely-used non-linear dimension reduction schemes. Of course, as mentioned earlier, the premise behind NN learning is that if the input vectors are similar, the targets are also similar with a very high likelihood. In order to realize this premise, researchers have used monotonically decreasing kernel functions, with regard to the distance, to assign weights to the neighbors. Along the same vein, in the case of LLR, we attempt to enforce this general premise in the topological space for the k-NN. Indeed, we argue that if it is possible to accurately describe the input vector for a given query by its neighboring reference patterns, it is also possible to predict (estimate) well the target class (value) of the query with a small error. To initiate discussions in this regard, we first state the notation that we shall use (in a d-dimensional feature space), after which we shall formally describe LLR.

- $\underline{X}_i$ is a "query" (i.e., the testing point) in the feature space, and is a $d \times 1$ vector.
- $\underline{\widehat{X}}_i$ is a re-constructed version of $\underline{X}_i$, and is also a $d \times 1$ vector.
- $\boldsymbol{X}^i_{NN}$ is a $d \times k$ matrix, and contains the d-dimensional k-NNs of $\underline{X}_i$.
- $\underline{W}_{i,NN}$ is a $k \times 1$ vector. It is the corresponding weight vector obtained from $\boldsymbol{X}^i_{NN}$. The matrix $\boldsymbol{W}$, which is the collection of $\underline{W}_{i,NN}$'s, is the set of vectors sought for, and $W_{i,j}$ is the set of weights for $\underline{X}_j$ with regard to the sample point $\underline{X}_i$. Observe that $W_{i,j}$ will be zero if $\underline{X}_j$ is not a neighbor of $\underline{X}_i$.
- The matrix $\mathbf{N}$ is the neighborhood indicator matrix whose element $N_{i,j} = 0$ if $\underline{X}_j$ is not a neighbor of $\underline{X}_i$, and is unity otherwise. For ease of notation, $\mathbf{N}(i)$ will represent the NNs of $\underline{X}_i$.

When a query is given, the method first selects the k-nearest neighbors of the query. Once these NN patterns have been selected, the set of weights corresponding to the neighbor are determined by minimizing the LLR error $Err(\boldsymbol{W})$, defined as the sum of the errors E_i as follows: $\sum_i \left\| \underline{X}_i - \underline{W}^T_{i,NN} \underline{X}_j \right\|^2$, where every $\underline{X}_j$ is a NN of $\underline{X}_i$.

The weights, $\boldsymbol{W}$, which minimize the reconstruction error, can be obtained by solving the above minimization problem. Also, since the constraints on the optimization problem differ depending on whether the learning task is a classification or regression problem, the corresponding procedures for solving them are different as well. In particular, for classification tasks, we need to impose two additional constraints on $\boldsymbol{W}$, namely that all the weights must be non-negative, and that the sum of the neighbors' weights must be *unity* for every query. Thus,

$$Err(\boldsymbol{W}) = \frac{1}{2}\sum_i \left\| \underline{X}_i - \underline{W}_{i,NN}^T \boldsymbol{X}_{NN}^{i\ T} \right\|^2 \tag{1}$$
$$= \frac{1}{2}\sum_i \left\{ \underline{X}_i^T \underline{X}_i - 2\underline{X}_i^T \mathbf{X}_{NN}^i \underline{W}_{i,NN} + \underline{W}_{i,NN}^T \boldsymbol{X}_{NN}^{i\ T} \boldsymbol{X}_{NN}^i \underline{W}_{i,NN} \right\}.$$

By examining Eq. (1), we see that we can obtain the weights for the k-NNs of $\underline{X}_i$, $\underline{W}_{i,NN}$, by solving the following optimization problem[3]:

$$Min\ Err(\underline{W}_{i,NN}) = \frac{1}{2}\underline{W}_{i,NN}^T \boldsymbol{X}_{NN}^{i\ T} \boldsymbol{X}_{NN}^i \underline{W}_{i,NN} - \underline{X}_i \boldsymbol{X}_{NN}^i \underline{W}_{i,NN}, \tag{2}$$
$$such\ that\ \ \underline{W}_{i,NN} \geq 0,\ \ \sum_j W_{i,j} = 1\ \ \forall i.$$

After obtaining the weights assigned, we can reconstruct a sample point, $\widehat{\underline{X}}_i$, corresponding to the query $\underline{X}_i$ by a weighted sum of the samples of $\underline{X}_i$'s NNs as follows:

$$\widehat{\underline{X}}_i = \sum_{\underline{X}_j \in \boldsymbol{N}(i)} W_{i,j} \boldsymbol{X}_j. \tag{3}$$

As the reader will observe, although this strategy is expedient, it involves the unavoidable non-trivial computationally intensive optimization. But our position is that it need not be done for all the sample points, but merely for a smaller subset of points which *represent* them - i.e., those obtained by a PRS.

Prototype Reduction Schemes: In non-parametric pattern classification which uses the NN or the $k-$NN rule, each class is described using a set of sample prototypes, and the class of an unknown vector is decided based on the identity of the closest neighbor(s) which are found among all the prototypes. To reduce the number of training vectors, various PRSs have been reported in the literature - two excellent surveys are found in [2], [6]. Rather than embark on yet another survey of the field, we mention here a *few* representative methods of the "zillions" that have been reported. One of the first of its kind is the Condensed Nearest Neighbor (CNN) rule [10]. The reduced set produced by the CNN, however, customarily includes "interior" samples, which can be completely eliminated, without altering the performance of the resultant classifier. Accordingly, other methods have been proposed successively, such as the Reduced Nearest Neighbor (RNN) rule, the Prototypes for Nearest Neighbor (PNN) classifiers [5], the Selective Nearest Neighbor (SNN) rule [15], two modifications of the CNN [18], the Edited Nearest Neighbor (ENN) rule [7], and the non-parametric data reduction method [9]. Besides these, the Vector Quantization (VQ) and the Bootstrap techniques have also been reported as being extremely effective approaches to data reduction. Recently, Support Vector Machines (SVM) [4] have proven to possess the capability of extracting

[3] The quadratic programming problem, $\min \frac{1}{2}\underline{U}^T \boldsymbol{H}\underline{U} + \underline{B}^T\underline{U}$, such that $\boldsymbol{A}\underline{U} \leq 0$, $\boldsymbol{A}_{eq}\underline{U} = \underline{b}_{eq}$, and $\underline{l}_b \leq \underline{U} \leq \underline{u}_b$, (where $\boldsymbol{H}$, $\boldsymbol{A}$, and $\boldsymbol{A}_{eq}$ are matrices, and $\underline{B}$, $\underline{b}_{eq}$, $\underline{l}_b$, $\underline{u}_b$, and $\underline{U}$ are vectors) defines a set of lower and upper bounds on the design variables, $\underline{U}$, so that the solution is in the range $\underline{l}_b \leq \underline{U} \leq \underline{u}_b$.

vectors that support the boundary between any two classes. Thus, they have been used satisfactorily to represent the global distribution structure.

In selecting prototypes, vectors near the boundaries between the classes have to be considered to be more significant, and the created prototypes need to be adjusted towards the classification boundaries so as to yield a higher performance. Based on this philosophy, Kim and Oommen [12], [13] proposed a new hybrid approach (HYB) that involved two distinct phases, namely, those of selecting and adjusting [12]. To overcome the computational burden for "large" datasets, they also proposed a recursive HYB in [13]. In [13], the data set is sub-divided recursively into smaller subsets to filter out the "useless" internal points. Subsequently, a conventional PRS (i.e., HYB) processes the smaller subsets of data points that effectively sample the entire space to yield *subsets* of prototypes – one set of prototypes for each subset. The prototypes, which result from each subset, are then coalesced, and processed again by the PRS to yield more refined prototypes. In this manner, prototypes which are in the interior of the Voronoi boundaries, and are thus ineffective in the classification, are eliminated at the subsequent invocations of the PRS, *noticeably* reducing the PRS's processing time.

This overview of the state-of-the-art of PRSs should be sufficient to help us proceed in formulating our solution to the problem at hand.

3 Schema for the Proposed Solution

Our goal is to "quickly" find out the class of a query point in the input feature space after reconstructing an approximated version of the corresponding sample using its NNs. However, rather than reconstruct the approximated data sample using the entire training set, we advocate that the data be first reduced into a smaller representative subset using a PRS, and that the data point be estimated by invoking a reconstruction scheme on *this* reduced data set. Thereafter, the classification accuracy of the $k-$NN classifier is compared. Thus, the proposed scheme can be formalized as follows:

Algorithm 1. PRS_LLR

Input: The original Training Set, T.

Output: Testing by utilizing a fast reconstruction of the approximated query point using a reduced data set rather than the entire training set.

Assumption 1: The algorithm has access to a PRS such as the CNN, PNN or HYB.

Assumption 2: The algorithm has access to the LLR algorithm mentioned previously.

Method:

Step 1: Select the representative set, Y, from the training set T by resorting to a PRS.

Step 2: Find the closest neighbors, $\boldsymbol{X}_{NN}^{i}$, for a query $\underline{X}_i$ from Y, rather than from T.

Step 3: Compute corresponding weight vector, $\underline{W}_{i,NN}$, using LLR and a k_1-NN rule.

Step 4: Reconstruct $\widehat{\underline{X}}_i$ with LLR using $\boldsymbol{X}_{NN}^{i}$ and $\underline{W}_{i,NN}$, and the k_1-NN rule.

Step 5: Classify $\widehat{\underline{X}}_i$ by comparing it with the elements of Y using the best k_2-NN rule.

End Algorithm PRS_LLR

We would like to emphasize that there are a few fundamental differences between what we propose and the original LLR method proposed in [11]. First of all, we observe that the computation of the LLR weights does not involve the entire training set T, but

a representative set, Y, derived from it using a PRS. Secondly, we note that the weights that are computed for the LLR involve a NN rule, using k_1 neighbors, where the latter is the pre-determined degree of the NN classifier used for the training phase. But once the reconstructed point is obtained, we now have the freedom of testing it using the most suitable NN classifier, which may not necessarily be a k_1-NN classifier. Indeed, as in any PR problem, given a training set, the practitioner has the freedom to choose the best NN classifier that suits his application. In the same vein, in our case, we choose the best "Testing" NN classifier (a k_2-NN classifier) for the application domain, using the modified "Training" set, Y, and the modified testing sample, $\underline{\widehat{X}}_i$. It turns out that usually, k_2 is quite distinct from k_1!

We shall now demonstrate the power of **Algorithm PRS_LLR**.

4 Experimental Set-Up, Results and Evaluation

Experimental Data: The proposed scheme has been tested and compared with the conventional LLR method reported in the literature. This was done by performing experiments on both "artificial" and "real-life" data sets [4]. In each case, the sample vectors of each data set was divided into two subsets of equal size $T1$ and $T2$ (typically, used for training and validation, alternatively). The computation was done on each subset and subsequently averaged.

In our experiments, the four artificial data sets "Non_normal 2, 3" and "Non_linear 2, 3", were generated with different sizes for the testing and training sets, and had cardinalities of 500 and $5,000$ respectively. The data sets "Ionosphere", "Sonar", and "Arrhythmia", which are real benchmark data sets, are cited from the UCI Machine Learning Repository [3].

The data set named "Non_normal" (in short, "Non_n"), which has also been employed as a benchmark experimental data set for numerous experimental set-ups, was generated from a mixture of four 8-dimensional Gaussian distributions as desribed in detail in [8]). The data set named "Non_linear" (in short, "Non_l") which has a strong non-linearity at its boundary, was generated artificially from a mixture of four normal variables as described in [13].

Experimental Parameters: Choosing the parameters of PRSs play an important role in determining the quality of the solution. The parameters[5] for the PRSs[6] were: Since the number of prototypes depends on the characteristics of the data set, the number of

[4] More extensive results for other data sets are available, but omitted here in the interest of space.

[5] These parameters are included here for the sake of researchers who would like to duplicate the results.

[6] The reader should observe that, as mentioned previously, any PRS can be employed to obtain the reduced set, Y. In the present paper, only three methods, namely CNN, PNN, and HYB have been used in the testing. The main reason for choosing these is as follow: First of all, the prototype vectors obtained with CNN and PNN are *selected* and *created*, respectively. On the other hand, for HYB, the prototypes are initially *selected*, after which they are adjusted. Finally, for all the methods, the final number of prototypes is not a quantity that is controlled or determined automatically.

iterations is predetermined by the size of T. Hence, CNN and PNN had no parameters. In HYB, we invoked a hybridized version of the SVM and an LVQ3-type algorithm, both of which are available in publicly distributed packages. The SVM was employed to determine the initial code book vectors for the LVQ3. The parameters for the LVQ3 learning are specified in [12]. For instance, the parameters for the data set "Adult4" were $\alpha = 0.05$, $\epsilon = 0.06$, $w = 0.35$, $\eta = 5{,}600$.

Selecting Prototype Vectors: In order to evaluate the proposed classification mechanisms, we first selected the prototype vectors from the experimental data sets using the CNN, PNN, and HYB algorithms. In HYB, we selected initial prototypes using a SVM algorithm. After this selection, we invoked a phase in which the optimal positions (i.e., with regard to classification) were learned with an LVQ3-type scheme. For the SVM and LVQ3 programs, we utilized publicly-available software packages. For example, we can see that the numbers of selected prototype vectors of the "Non_n2" dataset with CNN are $(64, 66)$, $(56, 380)$, and $(63, 57)$, respectively. Each of them is considerably smaller than the size of the original data set, $(500, 500)$. Using the selected vectors as a representative of the training data set, we can significantly reduce the cardinality of the dataset (and the consequential computations) without noticeably degrading the performance. The reduction of the classification processing time follows as a natural consequence. As an observation, we also mention that the reduction rate increased dramatically as the size of the data sets was increased.

Experimental Results: To illustrate the method, consider Figure 1 which shows the plots of the 2-dimensional data set $\left\{(x_2^i, x_4^i)^T\right\}_{i=1}^{50}$ projected from the original four dimensional "Iris2" data set. The figure on the left is the original data set, where the points of two classes are represented by '$\times$' and '$+$', respectively. The figure on the right is the reconstructed data set from the prototypes, rather than the entire samples using LLR. Here, the prototypes are extracted by CNN and represented by '$\otimes$' and '$\oplus$', respectively. The reader should observe the non-intuitive properties of the scheme by studying Figure 1. Although the samples shown in the figure on the right (given by '$\times$' and '$+$' respectively), are reconstructed by using the prototypes '$\otimes$' and '$\oplus$', respectively from the figure on the left, the reconstructed points do not *individually* "optimally" represent their original counterparts. However, *collectively*, they are the best locations for the k_1-NNs which can, in turn, *collectively* represent the points.

Tables 1 and 2 show the run-time characteristics of the proposed scheme for the artificial data sets and the other benchmark data sets. With regard to notation, in these tables, the abbreviations WHL, CNN, PNN, and HYB correspond to the experimental methods employed for the WHoLe data set, and the prototypes extracted with the CNN, PNN, and HYB methods, respectively. Analogously, in the case of WHL, the data complexities (classification accuracies) and the corresponding processing CPU-times were measured for the whole data set, and for CNN, PNN, and HYB, the measures were computed for the corresponding extracted prototypes.

By examining the results, it is clear that the classification accuracies, for the benchmark databases can be measured quite efficiently and fairly accurately by first invoking the corresponding PRS techniques. To clarify this, consider, for example, the accuracies obtained for the samples reconstructed with NN = 5 for "Non_n2". The classification

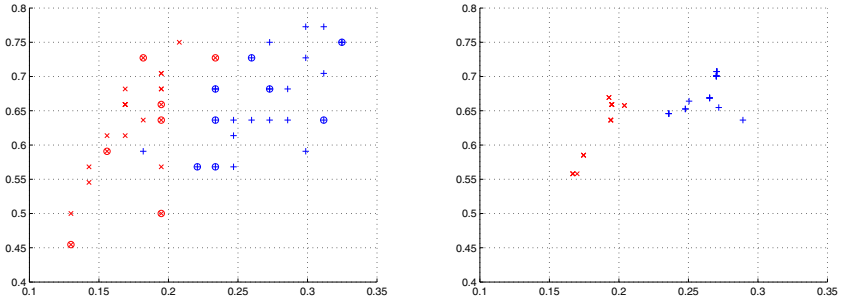

Fig. 1. Plots of the 2-dimensional vectors $\left\{ \binom{x_2^i}{x_4^i} \right\}_{i=1}^{50}$ generated from the original 4-dimensional "Iris2" data set. The details of the picture are discussed in the text.

Table 1. A comparison of classification accuracies (%) for the samples locally reconstructed with the experimental data sets and their prototypes extracted with CNN, PNN, and HYB, where each evaluation sample was reconstructed with the k_1 nearest neighbors of cardinalities *1, 3, 5, 7, 9, 11, 13*. The number in parenthesis in each entry represents the "order" k_2, of the corresponding "Testing" classifier, using which the respective accuracy was obtained.

Datasets	PRS	NN: k_1=1	NN: k_1=3	NN: k_1=5	NN: k_1=7	NN: k_1=9	NN: k_1=11	NN: k_1=13
Non_n3	WHL	94.50 (11)	94.52 (13)	94.54 (13)	94.54 (13)	94.54 (13)	94.50 (13)	94.52 (13)
	CNN	94.50 (11)	94.62 (11)	94.56 (13)	94.57 (13)	94.58 (13)	94.55 (13)	94.54 (13)
	PNN	94.40 (7)	94.53 (13)	94.56 (13)	94.56 (13)	94.53 (13)	94.60 (13)	94.50 (13)
	HYB	71.70 (9)	42.27 (1)	42.30 (1)	42.29 (1)	42.27 (1)	42.24 (1)	42.34 (1)
Non_l3	WHL	91.08 (11)	91.16 (13)	91.07 (13)	91.14 (13)	91.12 (13)	91.12 (13)	91.20 (11)
	CNN	90.46 (9)	90.06 (7)	89.96 (9)	89.83 (9)	89.74 (9)	89.71 (9)	89.70 (9)
	PNN	87.83 (9)	88.67 (7)	88.86 (7)	89.06 (7)	89.10 (7)	89.09 (7)	89.12 (7)
	HYB	88.04 (13)	88.36 (13)	88.18 (13)	88.18 (13)	88.12 (13)	88.20 (13)	88.24 (13)
Ionos	WHL	78.69 (1)	77.55 (1)	76.98 (1)	76.13 (1)	76.42 (1)	75.85 (1)	75.85 (1)
	CNN	81.81 (1)	80.39 (1)	77.84 (1)	77.27 (1)	75.85 (1)	74.43 (1)	74.14 (1)
	PNN	82.67 (1)	83.52 (3)	82.95 (3)	83.23 (3)	82.38 (3)	82.38 (3)	81.81 (3)
	HYB	83.23 (1)	80.96 (1)	77.27 (1)	78.12 (3)	78.97 (3)	78.40 (3)	78.69 (3)
Sonar	WHL	82.21 (1)	83.65 (3)	84.61 (3)	84.13 (3)	84.13 (3)	83.65 (3)	83.65 (3)
	CNN	79.80 (1)	79.80 (1)	78.36 (1)	79.32 (1)	78.36 (1)	79.32 (1)	79.80 (1)
	PNN	82.69 (1)	82.21 (1)	81.25 (1)	81.73 (1)	81.25 (1)	79.80 (1)	79.80 (1)
	HYB	80.76 (1)	79.80 (1)	79.80 (1)	79.32 (1)	79.32 (1)	78.36 (1)	78.84 (1)
Arrhy	WHL	97.56 (1)	97.56 (1)	97.56 (1)	97.78 (1)	97.78 (1)	97.34 (1)	97.34 (1)
	CNN	96.46 (1)	96.46 (1)	96.46 (1)	96.01 (1)	95.79 (1)	94.91 (1)	95.13 (1)
	PNN	99.11 (1)	99.11 (1)	98.89 (1)	98.67 (1)	–	–	–
	HYB	99.11 (1)	98.89 (1)	98.67 (1)	98.45 (1)	98.45 (1)	98.89 (1)	99.11 (1)

accuracies of WHL, CNN, PNN, and HYB are 94.50, 94.50, 94.60, and 71.90 (%), respectively, where the quantities mentioned in parenthesis in each row represent the classification accuracies that are obtained with the *11*-NN, *9*-NN, *7*-NN, and *5*-NN classifiers, respectively. But with regard to computation, the processing CPU-times of these

Table 2. A comparison of the processing CPU-times (seconds) required for the samples locally reconstructed with the experimental data sets and their prototypes. Here, the prototypes were extracted with CNN, PNN, and HYB, respectively. Thereafter, each evaluation sample was reconstructed with the nearest neighbors of cardinalities *1, 3, 5, 7, 9, 11, 13*.

Datasets	PRS	NN: k_1=1	NN: k_1=3	NN: k_1=5	NN: k_1=7	NN: k_1=9	NN: k_1=11	NN: k_1=13
Non_n3	WHL	91.22	112.30	117.02	128.06	148.27	152.55	167.41
	CNN	80.38	76.70	75.30	83.92	94.55	99.50	111.22
	PNN	114.22	139.23	135.19	152.86	183.89	174.75	192.83
	HYB	81.32	92.41	100.21	111.61	120.64	125.44	138.72
Non_l3	WHL	121.36	130.36	141.44	151.47	185.13	208.06	227.08
	CNN	58.42	65.59	71.67	82.78	98.50	108.94	125.95
	PNN	80.56	88.03	95.75	110.23	125.33	138.28	136.41
	HYB	90.53	97.38	105.88	126.09	151.14	167.52	188.38
Ionos	WHL	2.92	6.88	6.55	6.63	7.78	7.21	7.52
	CNN	2.85	6.72	6.60	6.57	6.94	6.96	7.29
	PNN	2.79	6.79	6.33	6.80	6.80	7.07	7.24
	HYB	2.87	6.85	6.57	6.66	6.82	7.07	7.13
Sonar	WHL	1.86	3.56	3.85	3.90	4.01	4.06	4.24
	CNN	2.00	3.46	3.90	3.87	4.01	4.18	4.21
	PNN	1.92	3.42	3.85	4.04	4.17	4.13	4.32
	HYB	1.90	3.60	3.92	3.95	4.13	4.21	4.34
Arrhy	WHL	98.94	107.27	109.28	114.02	119.34	123.80	130.98
	CNN	67.86	74.30	77.22	79.20	83.77	87.66	95.20
	PNN	64.36	70.67	73.05	76.17	–	–	–
	HYB	71.88	85.20	83.27	90.88	92.61	96.69	101.89

methods are 22.28, 20.28, 20.73, and 19.56 seconds, respectively[7] – which represents an advantage of about 12%. The effect is more marked in the case of large data sets. For example, in the case of the "Non_l3" data set, the accuracy measures of WHL, CNN, PNN, and HYB are 91.07, 89.96, 88.86, and 88.18 $(\%)$, respectively, while the processing times involved by using the PRSs are much smaller – namely 71.67, 95.75, and 105.88 seconds respectively, instead of 141.44 seconds required for the entire data set. Similar observations can also be made for the other benchmark data sets. But, in general, as an overall conclusion we believe that we can assert that a PRS can be effectively invoked to optimize the Locally Linear Reconstruction process for PR applications.

5 Conclusions

In this paper, we have considered how we can use the principles of Prototype Reduction Schemes (PRSs) to optimize the computations involved in the well-known families of k-Nearest Neighbor (k-NN) rules. Although k-NN rules have been extensively studied, recently, an implementation of the k-NN, named as the Locally Linear Reconstruction

[7] The times recorded are the times required for the MATLAB computation on a PC with a CPU speed of 2.40GHz and RAM 2GB, and operating on a Windows platform.

(LLR) [11], which invokes a quadratic optimization process has been proposed. The latter method is capable of systematically setting model parameters, such as the number of neighbors (k) and the weights. Our aim, in this paper, was to optimize the computation time required for LLR by using a PRS. We have proposed a strategy of using a PRS to efficiently compute the optimization problem. We have demonstrated that by completely discarding the points not included by the PRS, we can obtain a reduced set of sample points, using which, in turn, the quadratic optimization problem can be computed. The accuracies of proposed method is comparable to those obtained with the original training set (i.e., the one which considers all the data points) even though the computations required are noticeably less (the proposed method sometimes requiring only about 50% of the time). The proposed method has been tested on artificial and real-life data sets, and the results obtained are quite promising, and could have potential in PR applications. An avenue for further research involves developing alternate stochastic learning methods by which the query sample can be estimated accurately and quickly.

References

1. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning. Artifical Intelligence Review 11(5), 11–73 (1997)
2. Bezdek, J.C., Kuncheva, L.I.: Nearest prototype classifier designs: An experimental study. International Journal of Intelligent Systems 16(12), 1445–11473 (2001)
3. Blake, C.L., Merz, C.J.: UCL Machine Learning Databases. University of California, Department of Information and Computer Science, Irvine, CA, Can also be downloaded from `http://www.ics.uci.edu/mlearn/MLRepository.html`
4. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2), 121–167 (1998)
5. Chang, C.L.: Finding prototypes for nearest neighbor classifiers. IEEE Trans. Computers 23(11), 1179–1184 (1974)
6. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos (1991)
7. Devijver, P.A., Kittler, J.: On the edited nearest neighbor rule. In: Proc. 5th Int. Conf. on Pattern Recognition, Miami, Florida, pp. 72–80 (1980)
8. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press, San Diego (1990)
9. Fukunaga, K., Mantock, J.M.: Nonparametric data reduction. IEEE Trans. Pattern Anal. and Machine Intell. 6(1), 115–118 (1984)
10. Hart, P.E.: The condensed nearest neighbor rule. IEEE Trans. Inform. Theory 14, 515–516 (1968)
11. Kang, P., Cho, S.: Locally linear reconstruction for instance-based learning. Pattern Recognition 41, 3507–3518 (2008)
12. Kim, S.-W., Oommen, B.J.: Enhancing prototype reduction schemes with LVQ3-type algorithms. Pattern Recognition 36(5), 1083–1093 (2003)
13. Kim, S.-W., Oommen, B.J.: Enhancing prototype reduction schemes with recursion: A method applicable for "large" data sets. IEEE Trans. Systems, Man, and Cybernetics - Part B 34(3), 1384–1397 (2004)

14. Liu, T., Moore, A., Gray, A.: Efficient exact k-NN and nonparametric classification in high dimensions. In: Proc. of Neural Information Processing Systems (2003)
15. Ritter, G.L., Woodruff, H.B., Lowry, S.R., Isenhour, T.L.: An algorithm for a selective nearest neighbor rule. IEEE Trans. Inform. Theory 21, 665–669 (1975)
16. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500), 2323–2326 (2000)
17. Roweis, S.T., Saul, L.K.: Think globally, fit locally: unsupervised learning of nonlinear manifolds. Journal of Machine Learning Research 4, 119–155 (2003)
18. Tomek, I.: Two modifcations of CNN. IEEE Trans. Syst., Man and Cybern. 6(6), 769–772 (1976)