# Prototype Selection: Combining Self-Generating Prototypes and Gaussian Mixtures for Pattern Classification

Cristiano de S. Pereira and George D. C. Cavalcanti

***Abstract*— This paper presents an investigation into prototype-based classifiers. Different methods have been proposed to deal with this problem. There are two main classes of prototype-selection algorithms. The first is merely selective, in which the resulting set of prototypes is formed by well-chosen samples from the training set. The second is known as the creative class of algorithms. This strategy creates new instances and performs adjustments of the prototypes during training. Two methods of the creative strategy are presented here: a self-generating prototype scheme and a fuzzy variation of Nearest Prototype Classification, which uses a Gaussian Mixture ansatz. The respective advantages and problems are discussed. A hybrid method is proposed to overcome difficulties and improve accuracy. The hybrid strategy obtained better results in the experiments when compared to each of two basic approaches and the classic K-Nearest Neighbor.**

## I. Introduction

Prototype selection is a machine learning scheme, the main purpose of which is to choose pattern vectors from the training dataset that achieve a better fit to data point distributions and maintain class separation. The aim of this kind of technique is to find the smallest possible prototype set that minimizes the classification error rate. Classification can usually be achieved by using the nearest prototype strategy. An advantage to prototype-based classifiers is the low demand for storage space. The computer resources required in classification tasks are also considerably fewer. Such methods are known as Prototype Reduction Schemes.

Kim and Oommen [1] introduced a taxonomy for reduction schemes. The authors separate these strategies into two classes: *selective* and *creative* schemes. The first scheme is purely selective, i.e, the resulting set of this kind of scheme is completely formed by data points from the original data set. In *creative* schemes, new prototypes are created during the reduction process by combining data points and making adjustments during supervised training.

Different reduction techniques have been proposed and good results have been obtained in different domains. Wilson and Martinez [2] investigated different reduction techniques, comparing performances in 31 distinct classification tasks. The authors propose a set of six algorithms (DROP1-DROP5 and DEL) that seem more robust with regard to the presence of noise data. Batista et al. [3] applied Tomek Links [4], Condensed Nearest Neighbor Rule (CNN) [5], One-Sided Selection (OSS) [6] and Neighborhood Cleaning Rule [7] as reduction techniques in unbalanced classes.

Learning Vector Quantization (LVQ) is very popular algorithm of creative schemes, introduced by Kohonem [8]. This algorithm starts with an initial set of prototypes and makes adjustments in the prototype locations in order to locally approximate the classification boundaries. This training is supervised. However, the heuristics of the selection strategy makes the creation of an optimization process more difficult, which is a disadvantage of LVQ.

Models [9], [10] have been proposed to improve the performance of the classic LVQ algorithm. Such models assign discriminant and cost functions to the training, but also use heuristics.

In order to overcome difficulties in estimating discriminant functions and achieve an optimization process based on the descendent gradient method, Seo et al. [11] introduced the Soft Nearest Prototype Classification (SNPC) model. This model combines an explicit *ansatz* to describe the probability density function of the data points in the classes with a selection criterion that minimizes the classification error rate. This is an improved LVQ derivation that has the advantage of easy adaptation by changing the discriminant function and distance measure. The SNPC classification strategy is inspired by Bayesian Theory, which seeks to construct the classification boundaries using the criterion of maximum *a posteriori* (MAP) probability.

However, SNPC has an important: the model parameters. For example, the initial position of the prototypes is one of these parameters. The parameters are very important for the success of the experiments and it is necessary to find better values for eachnew problem. The parameter values that produce the best results are known as hyperparameters. Experiments carried out with SNPC demonstrate obtaining these hyperparameters can be an exhaustive task.

Fayed et al. [12] propose an approach for prototype selection based on self-generation, known as Self-Generating Prototypes (SGP). The main advantage of SGP over SNPC is the absence of parameters. This method does not require an initial prototype position. It generates prototypes using an easily understood set of rules that divide, combine and remove datapoint groups, thereby acting as a reduction scheme. Other property of SGP demonstrated in [12] is the short time elapsed in the training process.

The work reported in this paper aims to combine SGP and SNPC in order to form an improved hybrid scheme with a better performance regarding error classification rates than these techniques separately. The main idea is to use SGP

This work was supported in part by the Brazilian National Research Council CNPq (Proc. 478534/2006-0).

Authors are with the Center of Informatics (CIn), Federal University of Pernambuco (UFPE), P.O. Box 7851, Cidade Universitaria, Cep: 50.740-530 Recife PE Brazil (corresponding authors to provide phone:55-81-21268430; e-mails: cdsp@cin.ufpe.br; gdcc@cin.ufpe.br).

as a pre-processing task in order to create the initial set of prototypes. SNPC is then applied using the prototypes obtained by SGP as a fine-tuning step.

The next section is dedicated to SNPC, presenting its mathematical bases and formalizing the cost function and equations used for the training step and classification. Section III describes SGP and presents its training rules. Reasons for the creation of a new method by combining SGP and SNPC are explained in Section IV. Section V describes the experiments, databases and main results. Finally, Section VI presents the conclusions regarding this work.

## II. SNPC

Classic NPC operates in hard mode classification, i.e, a new data point is assigned to the prototype class nearest to it. This is the "winner-takes-all" strategy. Note that when using this strategy, if a new data point is placed at the classification boundaries, only the closest prototype defines the class of this point. Although this point is in questionable area, the other neighbor prototypes are not considered when making a decision.

SNPC introduces fuzzy assignment probabilities to the data points in relation to the prototypes. Each class may be represented by one or more prototypes. Thus, for each data point, the degree of of the point to all classes in the problem is defined. Once the assigned probabilities are defined for the prototypes, all prototypes are moved proportionally to these probabilities. The prototypes that have the same class of the current point are moved to near this point and the other prototypes are repulsed. This occurs during the learning process in order to minimize the classification error for points placed next to classification boundaries.

In order to guide the learning and evaluate the generalization performance, a cost function (1) has been introduced.

$$E = \frac{1}{N}\sum_{k=1}^{N}\sum_{\{j:c_j\neq y_k\}}^{M} P(j|x_k) \tag{1}$$

where $N$ is the number of data points, $M$ is the number of prototypes, $x_k$ is a sample point, $y_k$ is the true class of $x_k$ and $c_j$ is the j-prototype class. Evaluating this equation from the NPC standpoint, $P(j|x_k)$ is 1 if $j$ is the nearest prototype to $x_k$ and is 0, otherwise. The cost is the number of misclassified points divided by the size of the data point set.

In SNPC, Equation (1) has a different behavior and the a posteriori probability $P(j|x_k)$ is defined as:

$$P(j|x_k) = \frac{exp(-d(x_k,\theta_j))}{\sum_{m=1}^{M} exp(-d(x_k,\theta_m))} \tag{2}$$

where $d(x_k,\theta_j)$ is the distance from $x_k$ to prototype $\theta_j$. Equation (2) represents the fuzzy assignment probabilities of the Seo model. Note that (2) is a continuous function and a gradient descendent-based optimization procedure is therefore possible [13] [14]. One positive aspect is the lower influence of the data points that are placed near class boundaries in prototype adjustments. This avoids prototype oscillations and leads to faster convergence.

The individual cost of a sample is the sum of its assignment probabilities to all prototypes of the incorrect classes and the total classifier cost is the sum of all individual costs. From Equation (1), the individual cost $ls_k$ for each point is obtained as follows:

$$ls_k = \sum_{\{j:c_j\neq y_k\}}^{M} P(j|x_k) \tag{3}$$

Prototype positions are changed following equation (4).

$$\theta_l(t+1) = \theta_l(t) - \alpha(t)\frac{\partial ls_t}{\partial \theta_l} \tag{4}$$

The learning rule (6) was obtained from (4) and (1) and a formal proof can be viewed in [11].

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) - \alpha(t)\Delta\theta_l(t) \qquad (5)\\ \Delta\theta_l(t) &= \begin{cases} P(l|x_t)ls_t\frac{\partial d(x,\theta_l)}{\partial\theta_l}, (if) & c = y \\ -P(l|x_t)(1-ls_t)\frac{\partial d(x,\theta_l)}{\partial\theta_l}, (if) & c \neq y \end{cases} \end{aligned}$$

Once the learning process is finished, a new data point is classified using:

$$c = \underset{c'}{\text{argmax}} \sum_{\{j:c_j=c'\}} P(j|x_k). \tag{6}$$

### A. Gaussian Mixture Ansatz

Seo et al. [11] assumes a D-dimensional Gaussian Mixture Ansatz to describe point distribution around the prototypes. Thus, the probability density is given by:

$$p(x|j) = \frac{1}{(2\pi\sigma^2)^{D/2}} exp\left(-\frac{(x-\theta_j)^2}{2\sigma^2}\right) \tag{7}$$

and the assumed distance measure is:

$$d(x|\theta_j) = \frac{(x-\theta_j)^2}{2\sigma^2} \tag{8}$$

From these assumptions and using (3) and (5), it is possible to obtain a new assignment probability function (9) and new learning rule (11).

$$P(j|x) = \frac{exp\left(-\frac{(x-\theta_j)^2}{2\sigma^2}\right)}{\sum_k exp\left(-\frac{(x-\theta_k)^2}{2\sigma^2}\right)} \tag{9}$$

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) - \alpha(t)\Delta\theta_l(t) \qquad (10)\\ \Delta\theta_l(t) &= \begin{cases} -P(l|x_t)ls_t(x_t-\theta_l), \quad if & c = y \\ P(l|x_t)(1-ls_t)(x_t-\theta_l), if & c \neq y \end{cases} \end{aligned}$$

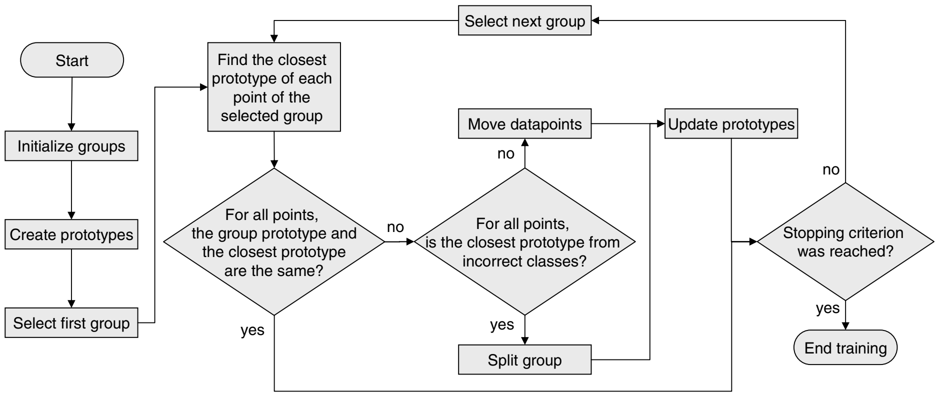

Start
Initialize groups
Create prototypes
Select first group
Find the closest prototype of each point of the selected group
Select next group
For all points, the group prototype and the closest prototype are the same?
no
yes
Move datapoints
no
For all points, is the closest prototype from incorrect classes?
yes
Split group
Update prototypes
no
Stopping criterion was reached?
yes
End training

Fig. 1. Workflow chart of the SGP training process

## III. SGP

Like other prototype-based machine learning schemes, SNPC depends on the correct choice of the number of prototypes per class and their initial locations. It is often difficult to find the optimal values for these parameters and the algorithm must run a number of times using different sets of parameters to do so. This takes a long time and increases the computational overhead. The Self-Generating Prototype (SGP) [12] method attempts to overcome these problems. The main advantage of this method is that the number of prototypes and their locations are obtained during the training process without much human intervention. Another advantage is the simplicity of its prototype selection strategy. The data points that have the same class label are grouped and the mean of each group represents the initial prototype set, resulting in one prototype per group. During the training process, the following operations are performed: groups are divided; data points are changed from one group to another; some groups are merged; and some groups are removed after a pruning step.

Figure 1 shows a workflow chart for the SGP training process. The initial groups are formed by data points from the same class and the initial prototypes are the mean of these groups. One by one, the groups are selected and evaluated as described in the chart. In the Split Group step, the selected group is divided in two subgroups. The separation boundary is the hyperplane $H$, which passes through the original mean of the group and is perpendicular to the first principal component of data points in the original group. The hyperplane can be created using:

$$H : \{x | x - \mu_x \cdot \gamma_1 = 0\} \tag{11}$$

where $x$ represents the data points of the original group, $\mu_x$ is the mean of this group and $\gamma_1$ is the first principal component (this step can be seen in Figure 2). In the Move Data Points step, the points in which the closest prototype is from another group of the correct class are moved to the group of the closest prototype. The points that have a closest prototype of the incorrect class are removed from the original group to form a new group. In the Update Prototypes step, the mean of the changed groups are recomputed as new positions of the prototypes of these groups. The Stopping Criterion is reached when no change occurs to either the groups or to the prototypes.

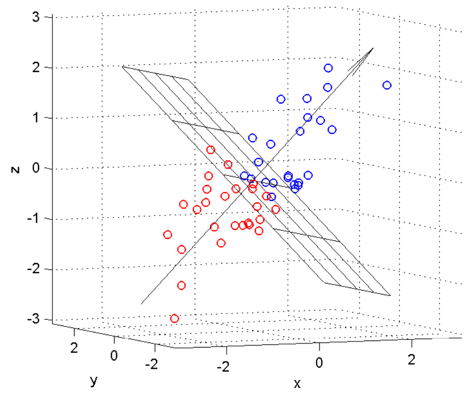
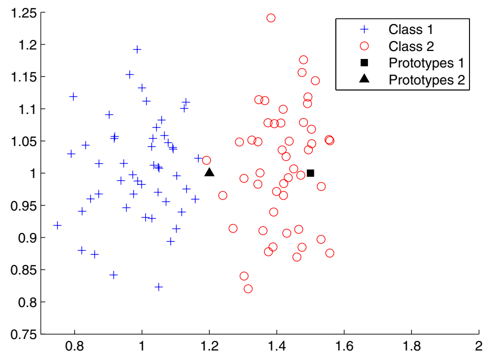
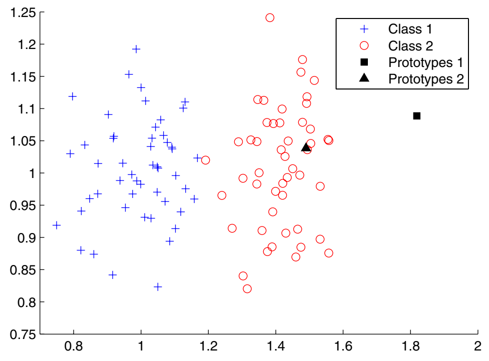


Fig. 2. The Split Group step. The arrow represents the first principal component.

Two strategies are introduced as post-processing tasks to remove redundant prototypes and reduce the number of prototypes: merging and pruning. After training, if there are two groups of the same class where the second closest prototype to all data points in the first group is the prototype of the second group and the other way around, the groups are merged. In pruning, if the second closest prototype of all data points of a group has the same class, the group and its prototype are removed.



(a) Before training.



(b) After training.

Fig. 3. Simulation of a SNPC problem with 2-D artificial data points.

## IV. Reasons for Combining SGP and SNPC

Despite their qualities, both SGP and SNPC have aspect that need improving. The SGP algorithm can present problems when applied to unbalanced domains. This has been shown in experiments with the Thyroid database from Proben1 benchmark problems in [12]. SGP reduced the number of prototypes of the classifier to one. Obviously, the resulting prototype was from the majority class. However, using the traditional K-NN approach, it is not possible to solve a two-class problem with a single prototype. All the points of the minor class were misclassified. A similar effect can be viewed in Figure 4 using artificial data.

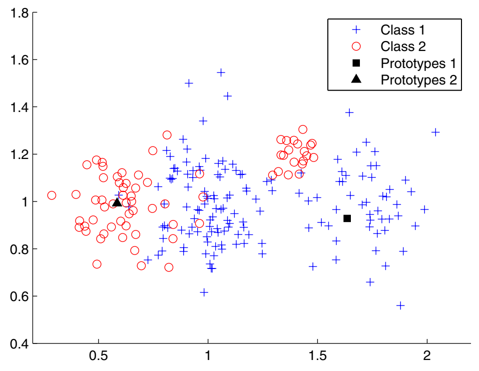


Fig. 4. Simulation of SGP problem with 2-D artificial data points

Figure 4 illustrates a set of artificial 2-D data points from a simulated two-class problem and the resulting prototypes generated by the SGP algorithm. Crosses and circles represent the patterns of the two classes. Squares and triangles are used to represent the prototypes of these classes, respectively. The data points are plotted in such a way that a smaller cluster of circles is intentionally located between two clusters of opposite classes. SGP was applied to this set of data points and only generated two prototypes. Note that there is no prototype to represent the small cluster of circles. The circle cluster between the cross clusters was probably removed during the SGP training. Despite the fact that this cluster did not represent a noise pattern, all the new data points falling within this cluster region arew misclassified. It is well known that the classifier needs to fit the prototypes better. Moreover, this shows that an entire class can be removed if this class is formed by small clusters.

SNPC, on the other hand, is very dependent on initial prototypes and other parameters. To illustrate this, Figure 3 displays a simulation of the SNPC training process with a set of 2-D artificial data points. The set is formed by crosses (Class 1) and circles (Class 2). Figure 3(a) shows the state before running the SNPC training. The distribution of points and initial prototypes can be seen in this picture. Figure 3(b) presents the location of the prototypes after training. The main purpose of this experiment was to evaluate the SNPC behavior when executing under adverse conditions. The prototype from Class 1 was positioned in the Class 2 cluster. Note that SNPC was not able to move Prototype 1 to the correct side of the space. On the contrary, the prototype was moved away from the cross cluster. The attraction force of the cross cluster in Prototype 1 was not larger than the repulsion force of the opposite class cluster. There was divergence of the sequence of adjustments for Prototype 1 generated by the SNPC process. This confirms that SNPC depends on the initial set of prototypes.

To avoid such problems, this present paper suggests a strategy that combines SGP and SNPC. The idea is to use SGP to generate a good set of initial prototypes, followed by SNPC to better fit these prototypes to the data point distribution and minimize the misclassification rate. This strategy intents to join the qualities of the two models into a hybrid model.

The next section is dedicated to experiments carried out with real-world datasets, presenting the databases, describing the experiments and commenting on the results.

## V. Experiments

### A. Databases

To validate the proposed approach, the hybrid model was tested in real-world datasets from different domains. The datasets are from Proben1 [15] and the UCI Machine Learning Repository [16]. These databases are well-known benchmarks for pattern classification.

*1) Cancer:* Diagnose breast cancer. The records of this dataset are formed by cell descriptions obtained through microscopic examination. Based on these attributes, the aim is to classify a tumor as either benign or malignant. This dataset is composed of 699 records with 9 inputs and 2 outputs. 65.5% of the examples are benign.

*2) Card:* Predict the approval or non-approval of a credit card for a customer based on 51 personal attributes. The dataset contains 690 examples and 44% of the samples are positive.

*3) Diabetes:* Predict whether an individual Pima indian is diabetes positive or not. The attributes are personal data (age, number of times pregnant) and medical examination results (e.g., blood pressure, body mass index, result of glucose tolerance test, etc.) The records are formed by 8 inputs and 2 outputs. The data set has 768 examples and 65.1% of these examples are diabetes negative.

*4) Gene:* Detect intron/exon boundaries (slice junctions) in nucleotide sequences. From a window of 60 DNA sequence elements (nucleotides), decide whether the middle is either an intron/exon boundary (donor), an exon/intron boundary (acceptor) or neither of these. This database is composed of 3175 examples. Each example is formed by 120 inputs and 3 outputs. There are 25% donors and 25% acceptors in the dataset. The remaining 50% are neither.

*5) Glass:* This dataset contains attributes obtained from chemical analyses of glass splinters. There are examples of six types of glass. The dataset has 214 examples with 9 inputs and 6 output. The class sizes are 70, 76, 17, 13, 9 and 29 instances. The objective is to classify the glass types.

*6) Satimage:* Classify satellite images. The database consists of multi-spectral values of pixels in 3x3 neighborhoods in a satellite image and the classification is associated with the central pixel in each neighborhood. There are 36 attributes and the examples are distributed in 6 decision classes (1 = red soil; 2 = cotton crop; 3 = grey soil; 4 = damp grey soil; 5 = soil with vegetation stubble; 6 = mixture class (all types present); 7 = very damp grey soil).

*7) Segment:* Image segmentation data. The instances were drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a classification for each pixel. This dataset has 2310 examples equally distributed in seven classes (1 = brickface; 2 = sky; 3 = foliage; 4 = cement; 5 = window; 6 = path; 7 = grass).

*8) Vehicle:* Classify a given silhouette as one of four types of vehicle (Opel, Saab, bus and van), using a set of 18 features extracted from the silhouette. The features are geometric measures. This dataset is composed of 240 examples from the Opel class, 240 examples from the Saab class, 240 examples from bus class and 226 examples from van class.

*9) Wine:* Classify wine types. These data are the results from a chemical analysis of wines. The analysis determined the quantities of 13 constituents found in each of the three types of wines. There are 59 examples of Class 1, 71 examples of Class 2 and 48 examples of Class 3.

### B. Experimental Results

The tests in real-world datasets were carried out using the methods described in this paper (SGP and SNPC), the classic K-Nearest Neighbor (KNN) and the hybrid strategy proposed here.

Two versions of the SGP algorithm were used in these tests. SGP1 is a basic version of SGP without the two Reduce Prototypes steps. SGP2 is the SGP algorithm with the Merging and Pruning steps. To avoid overfitting, Fayed et al. [12] introduced two parameters to SGP. These parameters are called $R_{min}$ and $R_{mis}$. $R_{min}$ represents a threshold for the relative size of a group in relation to the largest group. For example, if $R_{min}$ is 0.10 and the size of largest group is 200, all the groups with sizes less than 20 ($0.10 \times 200$) are discarded. The second parameter $R_{mis}$ is a threshold for misclassification rate of the group. If the number of misclassified patterns in a group divided by the size of the group is less than $R_{mis}$, the group remains unchanged. These parameters reduce the number of prototypes and improve the generalization capability. In the experiments, the values of these parameters were varied in range $[0.01 - 0.2]$. Five-fold cross-validation was used to estimate the best values for the $R_{min}$ and $R_{mis}$ parameters.

The initial prototypes for SNPC were positioned around the centroids $\mu_l$ of each class. The prototypes $\theta_l$ were generated using $\theta_l = \mu_l + 0.2 \times \sigma_l \times \zeta \times \xi$. $\sigma_l$ is a vector of the standard deviation of the data points from Class $l$ along the coordinate axis; $\zeta$ is the number of prototypes per class; and $\xi$ is a random number generated from a uniform distribution limited by -1 and 1. Five-fold cross-validation was used to estimate the best parameters. The algorithm used for the SNPC training requires a validation set. Thus, for SNPC and SGP2+SNPC in each cross-validation iteration, part of the data reserved for training was re-split into a new training set and a validation set. The new training set was formed by $75\%$ of the data points of original training set and the remaining $25\%$ was used for validation. Cross-validation and all the other splitting operations were stratified.

After a large number of preliminary experiments with K-NN, the decision was made to use the value 7 to set $K$ for all datasets.

Table I displays the classification error rate obtained for each method in each real-world dataset. Standard deviations are also displayed. The results in the table are the averages of

TABLE I
CLASSIFICATION ERROR RATES

| | **k-NN** | **SGP1** | **SGP2** | **SNPC** | **SGP2+SNPC** | **Average** |
|---|---|---|---|---|---|---|
| **Cancer** | $3.58 \pm 1.53$ | **3.11** $\pm 0.07$ | $3.27 \pm 0.08$ | $4.47 \pm 0.21$ | $\underline{3.19} \pm 0.37$ | 3.52 |
| **Card** | $16.08 \pm 1.67$ | $\underline{15.94} \pm 2.43$ | $16.09 \pm 1.71$ | $16.79 \pm 3.47$ | **14.21** $\pm 2.84$ | 15.82 |
| **Diabetes** | $27.08 \pm 3.46$ | $26.29 \pm 1.29$ | $\underline{26.27} \pm 2.36$ | $31.62 \pm 3.27$ | **21.47** $\pm 3.39$ | 26.55 |
| **Gene** | $21.98 \pm 2.09$ | $\underline{15.25} \pm 0.93$ | $\underline{15.25} \pm 0.93$ | **14.99** $\pm 1.71$ | $16.92 \pm 1.03$ | 16.88 |
| **Glass** | $35.48 \pm 7.62$ | $31.25 \pm 2.21$ | $29.93 \pm 2.93$ | $\underline{27.98} \pm 3.54$ | **27.52** $\pm 2.91$ | 30.43 |
| **Satimage** | **9.50** $\pm 0.75$ | $13.08 \pm 1.29$ | $14.31 \pm 3.51$ | $14.29 \pm 3.65$ | $\underline{11.06} \pm 2.38$ | 12.45 |
| **Segment** | **6.84** $\pm 1.32$ | $15.46 \pm 2.23$ | $14.98 \pm 2.11$ | $15.32 \pm 1.93$ | $\underline{11.76} \pm 1.84$ | 12.83 |
| **Vehicle** | $\underline{36.67} \pm 1.95$ | $37.58 \pm 2.18$ | $37.58 \pm 1.11$ | $42.21 \pm 3.02$ | **34.74** $\pm 2.17$ | 37.76 |
| **Wine** | $27.56 \pm 7.87$ | $26.56 \pm 8.22$ | $25.40 \pm 5.68$ | **21.60** $\pm 4.75$ | $\underline{24.99} \pm 5.28$ | 25.22 |
| **Average** | 20.53 | 20.50 | 20.34 | 21.03 | **18.43** | |

the classification error obtained for the best configurations of each pair $\langle$technique,dataset$\rangle$. The last line and column of the table are the general averages of the techniques and datasets, respectively. Bold numbers represent the lowest classification error and underlined numbers indicate the second lowest classification error.

Note that the hybrid method SGP2+SNPC obtained the best results in the tests with the Diabetes, Glass, Vehicle, and Card datasets. For example, the classification error rate achieved by the hybrid method for the Diabetes database was $21.47\%(\pm 3.39)$. This is nearly $5\%$ lower the second best method. Moreover, no other method achieved the lowest error rates in 4 of the 9 databases used. In the experiments with the Cancer, Satimage, Wine and Segment datasets, the SGP2+SNPC achieved the second lowest classification error. Comparing the two last columns of the table, the error rates of the SGP2+SNPC method are lower than the mean of all techniques for the respective datasets, except the for Gene dataset. The last line of the table displays the mean of the K-NN, SGP1, SGP2, SNPC and SGP2+SNPC for all datasets and confirms that the method proposed in the present work was better than the other methods.

## VI. CONCLUSIONS

This paper presented a hybrid pattern classification model that combines two methods with distinct characteristics. The scheme of the first model is based on self-generating prototypes - SGP. The second is a training algorithm for NPC that uses a Gaussian Mixture ansatz to describe the data point distributions and uses a method based on stochastic gradient descent to adjust the prototypes in the training algorithm. The contribution of this work was to formulate an improved method to overcome problems encountered in SGP and SNPC. SGP is first executed to generate the initial prototypes and the SNPC training algorithm is then applied to these prototypes to minimize the misclassification rate. This hybrid model does not depend on a good set of initial prototypes and has few parameters. The experiments performed in different domains demonstrate that the combination of SGP2 and SNPC achieved superior performance in terms of classification accuracy when compared to the other methods evaluated here.

## REFERENCES

[1] S.-W. Kim and B. J. Oommen, "A Brief Taxonomy and Ranking of Creative Prototype Reduction Schemes," *Pattern Analysis and Applications*, vol. 6, no. 3, pp. 232–244, 2003.

[2] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[3] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[4] I. Tomek, "Two modifications of cnn," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, pp. 769–772, 1976.

[5] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. IT-14, pp. 515–516, 1968.

[6] M. Kubat and S. Matwin, "Addressing the course of imbalanced training sets: One-sided selection," in *International Conference in Machine Learning*, pp. 179–186, 1997.

[7] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Proceedings of the 8th Conference on Artificial Intelligence in Medicine*, pp. 63–66, 2001.

[8] T. Kohonen, "Learning vector quantization for pattern recognition." Technical Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986.

[9] C.-H. L. Shigeru Katagiri and B.-H. Juang, "New discriminative training algorithms based on the generalized probabilistic descent method," in *Proceedings of the IEEE Workshop Neural Networks Signal Processing*, p. 299308, 1991.

[10] E. McDermott and S. Katagiri, "Prototype-based minimum classification error/generalized probabilistic descent training for various speech units," *Computer Speech and Language*, vol. 8, no. 4, pp. 351–368, 1994.

[11] M. B. Sambu Seo and K. Obermeyer, "Soft nearest prototype classification," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 390–398, 2003.

[12] S. R. H. Hatem A. Fayed and A. F. Atiya, "Self-generating prototypes for pattern classification," *Pattern Recognition*, vol. 40, no. 5, pp. 1498–1509, 2007.

[13] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[14] L. Bottou, "Online learning and stochastic approximations." Online Learning in Neural Networks, D. Saad, Ed., Cambridge: Cambridge University Press, 1998.

[15] L. Prechelt, "Proben1 a set of neural-network benchmark problems." University of Karlsruhe, Germany, 1994.

[16] A. Asuncion and D. Newman, "UCI machine learning repository." University of California, Irvine, School of Information and Computer Sciences, 2007.