A K-Nearest Classifier Design

Yann Prudent, Abdel Ennaji

Electronic Letters on Computer Vision and Image Analysis 5(2):58-71, 2005

A K Nearest Classifier design

Y. Prudent and A. Ennaji

PSI Laboratory, University and INSA of Rouen, Mont Saint aignan, France
Received 9 December 2004; accepted 15 March 2005

Abstract

This paper presents a multi-classifier system design controlled by the topology of the learning data. Our work also introduces a training algorithm for an incremental self-organizing map (SOM). This SOM is used to distribute classification tasks to a set of classifiers. Thus, the useful classifiers are activated when new data arrives. Comparative results are given for synthetic problems, for an image segmentation problem from the UCI repository and for a handwritten digit recognition problem.

Key Words: Incremental clustering, distributed learning, self-organizing map, pattern recognition

1 Introduction

As document analysis systems grow more sophisticated, it becomes increasingly important to be able to carry out learning and classification tasks more accurately. Machine learning techniques have received a great deal of attention over the last decades. With the emergence of neural networks [Hay95] and Support Vector Machines (SVM) [Vap95], important fundamental developments have led to accurate performances. The reasons for this success essentially come from their universal approximation property and, above all, their good generalisation behavior, which has been proved for many simple applications in recent years. Comparisons of various algorithms have shown that the superiority of one algorithm over another cannot be claimed [GRB00][SBR96]. Performances strongly depend on the characteristics of the problem (number of classes, size of the learning set, dimension of the feature space, etc) and on the efforts devoted to the "design task" of the algorithms (i.e., classifier architecture determination, tuning of learning parameters, etc). Authors in [GRB00] also noticed that a sufficient level of classification accuracy may be reached through a reasonable design effort, and further improvements often require an increasingly expensive design phase. Two other drawbacks can be noticed for the best known methods. The first one concerns the ability of the system to provide an estimation of the confidence of the decision. To achieve good reject behavior most applications in pattern recognition need parameters that can only be defined *a posteriori* according to the results obtained. The second limitation of classical approaches concerns the incremental learning capacity. This is one of the main challenges in the design of evolutionary, efficient and robust decision systems. Indeed, in many applications, new data sets are continuously added to an already huge database. One way to tackle this challenge is to introduce a decision system that operates incrementally (able to learn new data). To overcome these problems, several authors have proposed the idea of developing multi-expert decision systems. This idea is mainly justified by the need to take into account several sources of information -which can be complementary- in order to reach high classification accuracy and

Correspondence to: <yann.prudent@univ-rouen.fr>

Recommended for acceptance by < J.M. Ogier, T. Paquet, G. Sanchez>
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

to make the decisions more reliable, and/or to facilitate the classifier design. Several strategies covering most aspects including the nature of experts, methods of decision combination, etc, have been reported in the literature in recent years [Dui02][Kun02][GS98]. Following this strategy, this paper introduces a new scheme for the general problem of classification task-solving by designing a multi-classifier system. The distribution process respects the data topology in the feature space in order to reach reliable decisions. Extracting the topology of the learning data supposes starting the design process by a clustering phase. To this end and with the objective of designing an incremental decision system, we introduce a new self-organizing map (SOM) clustering algorithm which operates incrementally. During the decision process this SOM is used to activate the appropriate classifiers among a set of committee experts (SVMs). Two levels of reject power are introduced in order to increase the decision reliability. In the next section, a brief review of the most important multiple classifier design approaches and self-organizing maps is given. Section 3 introduces our new incremental clustering algorithm. The design of the decision system is presented in Section 4. Section 5 reports some experimental results and comparisons conducted on the problem of handwritten digit recognition. Finally, concluding remarks and future works are given in Section 6.

2 Related work

2.1 Distributed classification system

Numerous contributions are to be mentioned in the field of the design of multiple and cooperative classifier systems. Three major categories can be identified in this field according to their principles.

The approaches such as Boosting or Bagging are based on principles of data re-sampling, or a re-estimation of the training data weights and attempt to reach an optimal set of classifiers.

The second category of approaches operates only with supervised data [JJNH91]. The classification task is distributed to a committee of experts using only the supervised information. The main disadvantage of such approaches is that no real distribution of the data in the feature space, or particular distribution of classes is taken into account.

The last category concerns the hybrid approaches which attempt to combine both supervised and unsupervised tools.

In this paper we are more particularly interested in this third category, for which little work is reported in the literature. Authors in [GS98] start with an unsupervised training of the learning set by a Kohonen's feature map [Koh82]. This map makes it possible to specialize hidden layer neurons of a Multi-Layer Perceptron (MLP) according to the detected clusters. Because of the use of a Kohonen's map, the user must give the number of clusters *a priori*, which is not easy for real problems and does not seem a suitable choice for an incremental training system design.

[Rib98] also uses an unsupervised analysis of the problem to distribute it on several MLPs. The authors use a hierarchical clustering to determine the clusters in the training data, and an MLP is then associated to each detected cluster. The results obtained are equivalent to a K Nearest Neighbor classifier (KNN) for maximum recognition rate, and much better when the error must be very low. However, hierarchical clustering, although very efficient, presents a significant space and time complexity.

In [HPG99] the authors try to bring a solution to these disadvantages with the use of an unsupervised SOM model : the *Growing Neural Gas* network [Fri95b] which does not require any preliminary knowledge of the problem such as the number of clusters.This SOM is also used to specialize the hidden layer of an MLP according to the clustering result. This approach can be considered as an extension of [GS98].

All the approaches previously presented use an unsupervised algorithm to distribute the classification tasks either to several neurons of the same MLP or to a population of MLPs. None of them attempts to exploit the

diversity of the classifiers proposed in the literature review and/or to use different strategies of resolution.

2.2 An overview of Self-organizing Maps

We first would like to state the properties shared by all the models described below. The network structure is a graph consisting of a set of nodes (units or neurons) A and a set of edges N connecting the nodes. Each unit c has an associated position (or reference vector) w_c in the input space. Adaptation, during learning, of the reference vectors is done by moving the position of the nearest (or the "winning") unit (neuron) c_1 and its topological neighbors in the graph toward the input signal. For an input signal ξ the nearest unit c_1 is :

$$c_1 = \min_{c \in A} dist(\xi, w_c) \tag{1}$$

and the position is updated as follows :

$$\Delta w_c = \epsilon(t) h_{c,c_1} \|\xi - w_c\| \tag{2}$$

where $\epsilon(t)$ is the adaptation step and h_{c,c_1} is a neighborhood function.
We can differentiate two kinds of SOM, Static Self-organizing Maps and Growing Self-organizing Maps. The Static SOMs have a pre-defined structure which is chosen *a priori* and does not change during the parameter adaptation. Growing SOMs, however, have no pre-defined structure; this is generated by successive additions (and possibly deletions) of nodes and/or connections.

• **Static Self-organizing Maps**

The *Kohonen's Self-Organizing Feature Map* (SOFM) [Koh82] method considers a hyper-rectangular structure on the graph. Adaptation steps as described above are performed with:

$$\epsilon(t) h_{c,c_i} = \begin{cases} f(t) & \text{if } c \in N_{c_1}(t) \\ 0 & \text{otherwise} \end{cases}$$

where $0 \leq f(t) \leq 1$ is a decreasing monotonous function and:

$$N_{c_1}(t) = \{c \in A / \text{there is a path between } c_1 \text{ and } c \text{ that is smaller than } R_{c_1}(t)\}$$

$R_{c_1}(t)$ is a decreasing integer function.

The decreasing behavior of both function f and neighborhood area of the winning unit during learning, makes it possible to roughly explore the input space at the beginning of the training process, and to carry out a refinement of the unit position in the final phase.

Neural Gas (NG) [MS91] is a pure vector quantization method which does not define any topology among the units. Rather, adaptations are done based on the distance computation in the input space.

The main principle of NG is to adapt the k nearest units for each input signal ξ. During the learning, k decreases from a large initial to a small final value.

This approach supposes a constant number of units, and it is necessary to predefine the total number of adaptation steps due to decreasing parameters during learning. Indeed, a large initial value of k causes adaptation of a large number of units. Then k is decreased up to a final value, namely value one, and so only the nearest center for each input signal is adapted.

However, the NG model may be combined with Competitive Hebbian Learning [Mar93] to build up a topology during self-organization. The principle of the Competitive Hebbian Learning (CHL) method is simply to create an edge between the winning and the second winning unit at each adaptation step. The graph generated is a subgraph of the Delaunay triangulation corresponding to the reference vectors. The NG/CHL combination has been called "topology-representing networks" [MS94].

• **Self-organizing Map generated by a constructive process**

The *Growing Cell Structures* (GCS) model [Fri94] has a structure consisting of *Hypertetrahedrons* with a dimensionality chosen in advance. A k-dimensional hypertetrahedron is a k-polyhedron having only $k+1$ vertices. For example, k is respectively 1, 2 and 3 for lines, triangles and tetrahedrons. The model is initialized with one hypertetrahedron. Adaptation steps, as described above, are performed with:

$$\epsilon(t)h_{c,c_i} = \begin{cases} \epsilon_b & \text{if } c = c_i \\ \epsilon_n & \text{if there is an edge between } c \text{ and } c_i \\ 0 & \text{otherwise} \end{cases}$$

At each adaptation step local error information is accumulated in the winning unit c_1.

$$\Delta E_{c_1} = \|w_{c_1} - \xi\|^2$$

After a given number λ of these adaptation steps, unit q with the maximum accumulated error is determined and a new unit is inserted by splitting the longest edge emanating from q. Moreover, additional edges are inserted in order to have a structure with hypertetrahedrons. The GCS model has a fixed dimensionality, so it carries out a dimensionality-reducing mapping from the input space into a k-dimensional space. This can be used to visualize data.

The *Growing Neural Gas* (GNG) model [Fri95b] does not impose any explicit constraints on the graph topology. The graph is generated and continuously updated by competitive Hebbian Learning [Mar93]. The adaptation of reference vectors is identical in GNG and GCS. After a fixed number λ of adaptation steps, unit q with the maximum error is determined and a new unit is inserted between q and its neighbor p in the graph that has the maximum error. Error variables of q and p are re-distributed with the new unit. The topology of a GNG network reflects the topology of the input data distribution and can have different dimensionalities in different parts of the input space. For this reason it is only possible to visualize low-dimensional input data.

The *Growing Grid* (GG) method [Fri95a] supposes a hyper-rectangular structure on the graph. Stated otherwise, the graph is a rectangular grid of a certain dimensionality k which is chosen *a priori*. The starting configuration is a k dimensional hypercube.

For example, a 2×2-grid for $k = 2$ and a $2 \times 2 \times 2$-grid for $k = 3$. To keep the integrity of this structure, it is necessary to always insert complete rows or columns. Except for its structure, the Growing Grid has the same algorithm as the Growing Cell Structures network. A new row or column is inserted at each λ adaptation step and the accumulated error determination is used to determine where to insert this row or column.

The *Growing Self-Organizing Map* (GSOM) [BV97] combines the reference vector adaptation of the SOFM with a constructive procedure for hypercubal output space. It starts from an initial 2-neuron configuration, learns according to the SOFM-algorithm, adds neurons to the output space until a specified maximum number of units is reached. GSOM can grow by adding nodes in one of the directions already spanned in the output space, or by adding a new dimension. It uses the back-propagation of the winning unit error along the different directions in order to determine how to add nodes and in which direction.
Some of the neural network models presented in this section make it possible to learn new data dynamically, but do not guarantee the preservation (stability) of old knowledge as shown in the result section. It is the most important drawback when dealing with real and complex problems of data mining and knowledge discovery.

3 Contributions

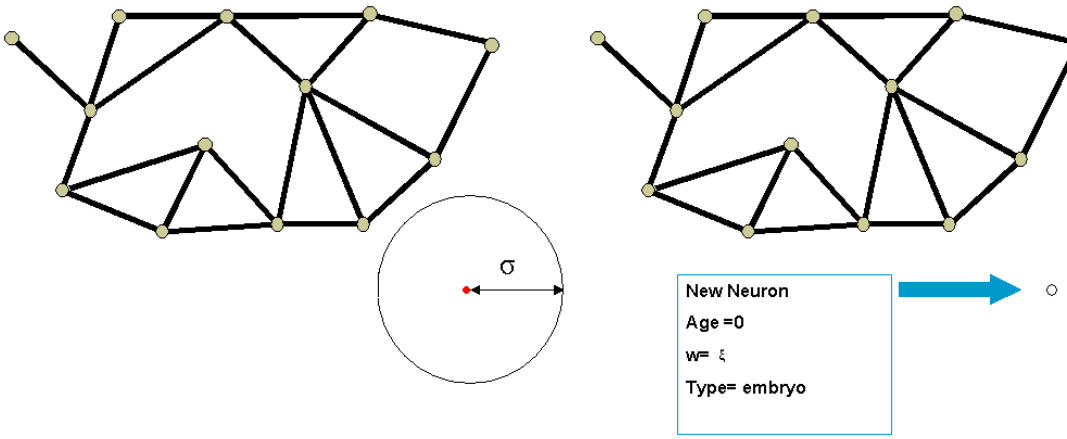
We point out that our long-term aim is to obtain an incremental learning system. Thus, for our purposes, the algorithm for building a self-organizing map must be incremental. Therefore, in this part, we describe a new training algorithm for a self-organizing map before presenting a possible use of the map for the design of a distributed and multi-classifier system.

3.1 Incremental Growing Neural Gas

Our model is like the Growing Neural Gas model in that it does not impose any explicit constraint on the graph, which is generated and continuously updated by competitive Hebbian Learning; however unlike the networks described here, our approch distinguishes two kinds of neurons : *mature* neurons, and *embryo* neurons. In addition, connections or edges between neurons are created dynamically during learning following the principles of the CHL algorithm. And both neurons and edges are associated to an age which is set to zero when created and updated during learning. When a new neuron is inserted, it is an *embryo* neuron and its age is set to zero. Initially, the graph is empty. At each iteration, we search for the winning unit following (eq. 1). Then, as in the Adaptive Resonance Theory (ART), we perform the vigilance test of eq. 3 in order to decide if the winning unit is close enough to the input signal.

$$dist(\xi, w_c) \leq \sigma \tag{3}$$

If the graph is empty, or the winning unit does not satisfy the vigilance test, we add a new *embryo* neuron with $w_{new} = \xi$ and the iteration is finished. This case is shown in figure 1. Figure 1(a) illustrates an IGNG network with new data to be learned (in dimension 2). This data is too far from the winning unit, so a new *embryo* neuron (fig 1(b)) is created. The reference vector of this new unit is the position of the data in the input space. If the vigilance test is satisfied for the winning unit, we apply it to the second-nearest unit. If there is only one



(a) A new training sample

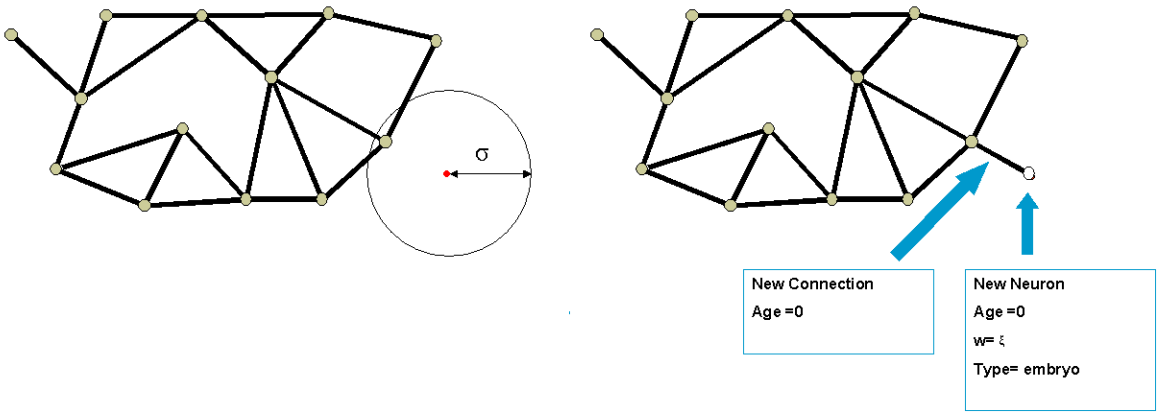
(b) Creation of a new neuron

Figure 1: insertion of a new neuron when the new data is too far from its winning unit

unit in the graph, or if the test is not satisfied for the second nearest unit, a new *embryo* neuron is added with $w_{new} = \xi$. Figure 2 shows this case. In this figure the new input satisfies the vigilance test, but it is too far from the second-nearest neuron, so we create a new *embryo* neuron which is connected to the winning unit (fig 2(b)). The reference vector of the new unit is the position of the data in the input space. Finally, when the two nearest units satisfy the vigilance test, the reference vectors of the units are adapted as in equation 2 with:

$$\epsilon(t)h_{c,c_i} = \begin{cases} \epsilon_b & \text{if } c = c_i \\ \epsilon_n & \text{if there is an edge between } c \text{ and } c_i \\ 0 & \text{otherwise} \end{cases}$$

The learning algorithm is then continued throughout the adaptation of both neurons and edges. If the connection between the two nearest units does not exist, an edge is created. Otherwise, the age of this edge is set to zero. On the other hand, the age of all the other edges connected to the winning unit is incremented.



(a) A new training sample (b) Creation of a new neuron

Figure 2: insertion of a new neuron when the new data is too far from its second-winning unit

Algorithm 1: The incremental growing neural gas algorithm

Data : **int** a_{mature}, **float** σ,**Training Database** S
Result : An IGNG network
begin
  while *a stopping criterion is not fulfilled* **do**
    Choose an input signal $\xi \in S$;
    Find the winning unit c_1;
    if *the graph is empty or* $dist(\xi, w_{c_1}) \geq \sigma$ **then**
      insert a new *embryo* neurons with $w_{new} = \xi$;
    else
      Find the second-nearest unit c_2;
      if *there is only one unit or* $dist(\xi, w_{c_2}) \geq \sigma$ **then**
        insert a new *embryo* neurons with $w_{new} = \xi$;
        create a connection between c_1 and ξ;
      else
        Increment the age of all edges emanating from c_1;
        $w_{c_1} += \epsilon_b(\xi - w_{c_1})$;
        $w_n += \epsilon_n(\xi - w_n)$; //($n$ are the directs neighbors of c_1)
        if c_1 *and* c_2 *are connected by an edge* **then**
          $age_{c_1 \to c_2} = 0$;
        else
          create a connection between c_1 and c_2;
        Remove edges with an age higher than a_{max}
        if *this results in mature neurons having no emanating edges* **then**
          remove them as well;
        Increment the age of all direct neighbors of c_1;
        foreach *embryo neuron c* **do**
          if *age(c)*$\geq a_{mature}$ **then**
            c becomes a mature neuron
end

Afterward, we remove edges with an age higher than a_{max}, and if this leads to isolated *mature* neurons (without connections), we remove them as well. Then, we increment the age of all direct neighbor neurons of the winning unit. Consequently, if a given *embryo* neuron has an age higher than a_{mature}, this unit becomes a mature neuron. The final graph is only made up of mature neurons, as *embryo* neurons are useful only for the training.
This process is detailed in algorithm 1.

3.2 K Nearest Classifier design

The topological structure represented by the *Incremental Growing Neural Gas* (IGNG) network is used to distribute a classification problem to a set of classifiers. We associate each mature neuron n_i of the IGNG network with a subset s_i of the training database such that :

$$s_i = \{\xi \in S / d(\xi, w_{n_i}) \leq \sigma_i\}$$

σ_i is currently given *a priori*, but it should be estimated during the IGNG training phase in a future and more elaborate version of our system. As we can see on fig.3, each neuron is thus associated with a hyper-spherical zone of influence of radius σ_i.

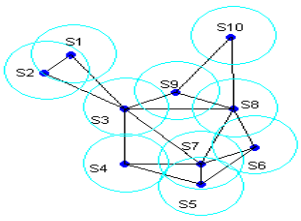
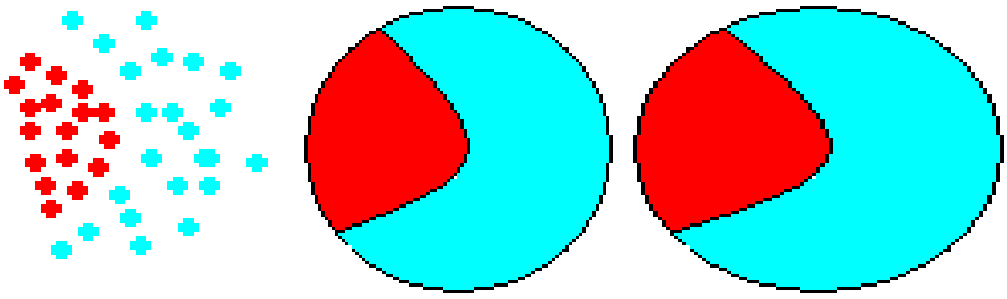


Figure 3: Distribution of the training set S on several subsets s_i

This method has the advantage of closing the decision frontiers generated by the classifiers (cf fig.4).



(a) Training data

(b) Decision frontiers generated by a SVM

(c) Decision frontiers generated by our approach

Figure 4: Closing of the decision frontiers

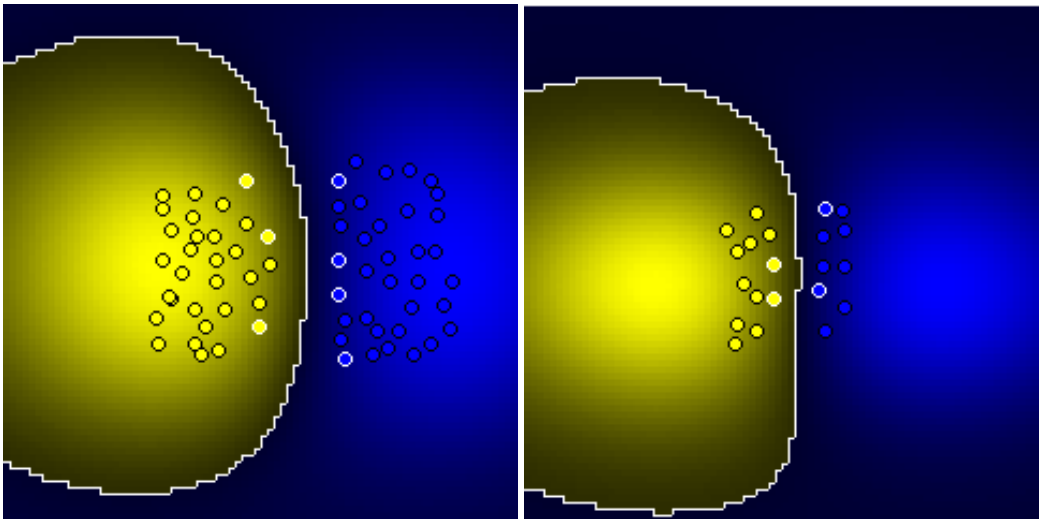


Figure 5: The frontiers designed by SVM with a Gaussian kernel is based only on the selected support vectors instead of a real class distribution

Indeed, the generation of open decision frontiers is one of the principal drawbacks of many classifiers such as MLPs or SVMs [GS98]. Thus, although efficient, they cannot reject efficiently and become not appropriate for real and complex pattern recognition problems. Such behaviour is well known for MLPs, which have another drawback : the so-called moving-target problem [GS98]. Concerning SVMs, their good generalization capabilities, which have been proved for many simple applications in recent years, do not give a convincing response to this challenge. Indeed, as illustrated in figure 5 on a very simple problem, the frontiers designed by SVM with a Gaussian kernel (the most reliable kernel to deal with this problem) confirm the statement that SVMs tend to draw unreliable (for reject) separation frontiers in the input data space (based only on the selected support vectors instead of a real class distribution). In our system, the parameter σ_i introduces a first level of reject. Indeed the union of all the subsets s_i is not necessarily equivalent to the whole training set S.

Our system considers the data that are not in this union as noise or as unreliable and they are not learned. When the system has to classify new data that does not belong to any area of neuron influence, a reject decision is made.

Once the training set S is broken up into several subsets s_i, for each subset we have the possibility of training several classifiers of various kinds (MLP, SVM, KNN, etc.), or with different training parameters (kernel, number of neurons, etc.). Hereafter, we will call the association "neuron, classifiers, learning subset" a GNeuron. A decision taken locally by a given GNeuron results from the combination function of the classifiers associated with it.

When the system has to classify a new input ξ, all the GNeurons that respect the condition given in equation 4 are selected.

$$d(\xi - w_i) \leq \sigma_i \tag{4}$$

Like in the KNN classifier, the system uses the classifiers of the K Nearest selected GNeurons, with a defined combination rule in order to take a decision.

In order to balance the vote of each Gneuron, we introduce a coefficient that takes into account two factors that seem significant to us : the size of the learning subset of the GNeuron ($|s_i|$), and the distance between the neuron and the data to be classified ξ. The coefficient given in equation 5 is an empirical proposition to introduce a weighting of the vote of each GNeuron.

$$v = \frac{|s_i|}{|s_i| + \beta d(\xi, w_{n_i})} \tag{5}$$

β is a constant and defines a compromise between the significance of the distance of the data ξ and the neuron, and the significance of the learning subset size of each neuron.

4 Experimental results

The aim of these experiments is to compare our method with general well known classification methods in several fields (image segmentation, handwritten digits recognition), and not with dedicated methods in handwritten digits recognition such as [La03, Ba01].

4.1 Incremental Growing Neural Gas

In this section, we will report some experimental results to demonstrate the general behavior and performances of the IGNG network. To visualize data, we use two synthetic problems with a low-dimensional input data space. The first one has been proposed by Martinetz and Schulten [MS91] to demonstrate the non-incremental "Neural Gas" model and was also used by Fritzke [Fri95b] to validate the "Growing Neural Gas" model. We propose a second synthetic problem which allows us to show the limits of the GNG model in incremental learning. In order to validate our model for high-dimensional problems, we report some results obtained for the handwritten digit recognition problem over a subset of the NIST database 3. The feature vector considered is composed of the 85 (1+4+16+64) gray levels of a 4-level-resolution pyramid [BB82]. Fig.6a gives an example of digits from the NIST database while Fig.6b shows an example (digit 2) of the representation retained.

(a) Figures of NIST database (b) Pyramid resolution of digit "2"

Figure 6: Examples of digits in the Nist database and the retained vector representation

In this experiment, each neuron is labeled for a classification task using a simple majority vote rule of the labelled learning data that have activated each neuron during the training process. Two subsets from the NIST database of respectively 2626 digits for training and 2619 other digits for testing are used in this experiment.

4.1.1 IGNG vs GNG in offline learning

• **Martinetz Distribution** The data distribution given in fig. 7 has been proposed by Martinetz and Schulten and used by Fritzke [Fri95b] to show that his model quickly learns the important and complex topological

relations in this distribution by forming structures of different dimensionalities.

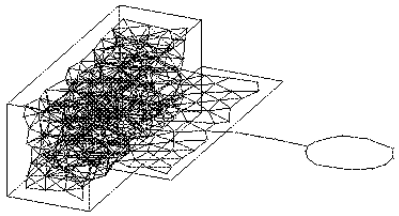


Figure 7: Topological map structure obtained over Martinez distribution

As we can see, our approach performs well in this example. The topological structure of the data is well reproduced in the map topology. Neurons and connections retained in the network represent the real distribution of the learning set data.

• **Handwritten Digit Recognition**

This experiment is done to compare our approach to the GNG and LVQ approaches for a classification problem. The three networks are compared using part of the NIST database described below. A learning cycle corresponds to a presentation to the network of all the training database samples. Table 1 shows that our

	Cycles	λ	ϵ_b	ϵ_n	σ	a_{mature}	Units	Recognition
LVQ	50	-	-	-	-	-	330	89.65%
GNG	50	400	0.1	0.006	-	-	330+0	91.44%
IGNG	**10**	-	0.01	0.002	2.7	450	313+9	**91.71**%

Table 1: Recognition rate of the three networks

approach leads to faster learning (5 times faster) than the GNG network for a comparable accuracy, and with a fewer number of neurons. Note that in this experiment, 9 *embryo* neurons are generated during the learning process.

4.1.2 IGNG vs GNG in incremental learning

• **A Synthetic Problem**

The synthetic problem we consider here is composed of two classes. The first class has a spherical distribution, while the second one has a cubic shape. The network was first trained with the spherical class. Then, to produce incremental learning, a data subset with only samples from the second class was presented to the trained network. Fig.8 shows the result obtained with the GNG and our IGNG models. On this figure, we can observe the degradation of the topology learned with the GNG model, reflecting a deterioration of the previously learned class. Such a phenomenon is avoided with our approach which allows a good behavior concerning the stability/plasticity dilemma in an incremental learning context.

• **Handwritten Digit Recognition**

For the digit recognition problem, the portion of the NIST database used was split into four parts. Each model was then trained on one part after another. Results reported on table 2 confirm the previously observed behavior.

(a) GNG (b) IGNG

Figure 8: GNG (a) and IGNG (b) behavior in an incremental learning

	Cycles	λ	ϵ_b	ϵ_n	σ	a_{mature}	Units	Recognition
GNG	50	400	0.1	0.006	-	-	328+0	81.29%
IGNG	**10**	-	0.01	0.002	2.7	450	244+16	**90.18%**

Table 2: Recognition rate on test datasets in incremental learning

4.2 K Nearest Classifiers (KNC)

As this part is still under development, the results presented in this section are only preliminary results. For this reason, the classifiers used for GNeurons are only SVMs and the vote rule used for classification is a unanimous vote. Our system, called KNC (K Nearest Classifiers) was compared to well-known classifiers, the KNN (K Nearest Neighbors) and the SVM (Support Vector Machine). The SVM used was libsvm [CCCJ04] (pairewise svm). The kernel used for the SVM was determined by a *try/error* process with the following kernels: linear kernel ($K(x, y) = x * y$), polynomial kernel ($K(x, y) = (\gamma xy + c_0)^{degree}$), Gaussian kernel ($K(x, y) = \exp(-\gamma|x - y|^2)$), and sigmoid kernel ($K(x, y) = \tanh(\gamma xy + c_0)$), while varying γ between 0 and 1 per step of 0.1, and $degree$ between 2 and 5 per step of 1, we took $c_0 = 0$.
In this section, we report some results obtained for two problems : the image segmentation problem of the UCI dataset and the handwritten digit recognition problem over the NIST database 3.

• **Image segmentation of UCI data**

This problem is composed of a training set of 210 instances and of a test set of 2100 instances. The feature space is of dimension 19 for a-class problem (BRICKFACE, SKY, FOLIAGE, CEMENT, WINDOW, PATH, GRASS). A linear kernel is used for the SVM. Table 3 shows the results obtained.

classifier	recognition(%)	error(%)
KNN	87.57	12.43
SVM	93.95	6,05
KNC	**94.34**	5.66

Table 3: Recognition and error rate of the three classifiers

• **Handwritten Digit Recognition**

As in section 4.1, two subsets of the NIST database of respectively 2626 digits for training and 2619 other digits for testing are used in this experiment. For these subsets we consider the feature vector constituted by the 85 (1+4+16+64) gray levels of a 4-level-resolution pyramid [BB82]. To improve our experiment we use two other subsets of the NIST database of respectively 10000 digits for training and 60000 other digits for testing. For this problem, the feature space is composed of 33 Fourier-Mellin based invariants [AOC+00]. The kernel used for the SVM is a Gaussian one with $\gamma = 0.2$.

classifier \ feature	85	33
KNN	96.83%	85.72%
SVM	**97.90%**	90.15%
KNC	**97.90%**	**90.79%**

Table 4: Recognition rate on the NIST database

classifier \ error	Max	$\leq$ 1.2%	$\leq$0.4%	$\leq$ 0.2%
KNN	96.83%	93.5%	81.59%	73.73%
SVM	**97.90%**	-	-	-
KNC	**97.90%**	**96.06%**	**90,34%**	**81.6%**

Table 5: Recognition rate/Error rate on the NIST database

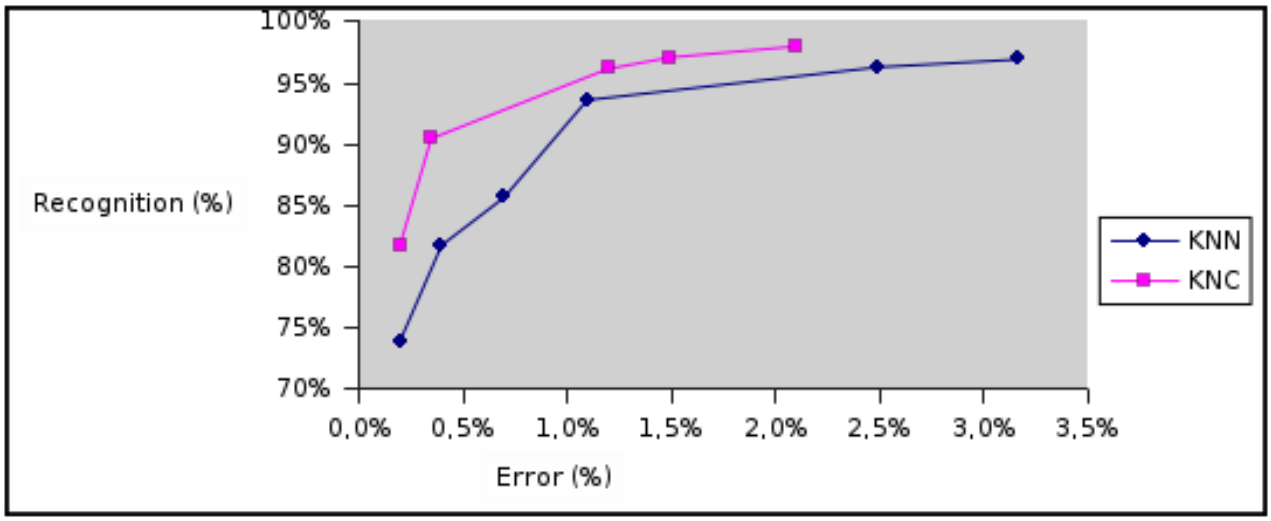


Figure 9: Error/Recognition for the KNN and the KNC for NIST database

The results of table 5 and the curves of figure 9 present the evolution of the recognition rate as a function of the error rate. The results of table 5 were obtained for each classifier in the following way:

-The results of the KNN were obtained by a unanimous vote of the K Nearest neighbors by increasing K until the required error rate is reached (the maximum recognition rate is obtained with k=1).

-In the same way, the results of the KNC were obtained by a unanimous vote of the K Nearest Gneuron classifiers by increasing K until the error required rate is respected (the maximum recognition rate is obtained for k=1). SVMs are not able to reject efficiently, thus it is impossible to obtain a curve in figure 9 for this classifier.

5 Conclusion

A new approach to designing a multi-classifier system is presented. This approach uses an incremental self-organizing map, introduced in this paper, in order to distribute several experts over the feature space. Preliminary results obtained with our approach are promising. The performances are equivalent to the other classifiers for a maximum error rate, but appear to be better when a low error rate is required. Indeed, the results obtained up to now show that if a maximum rate of recognition is necessary, our system is at least as powerful as SVMs. However, if the error rate must be low, the KNC obtains much better results, whereas the SVMs are not, to our knowledge, able to reject data efficiently. Our approach thus makes it possible to combine the recognition performances of classifiers such as SVMs and an effective rejection ability. The choice of the IGNG network, and the total freedom in the choice of classifiers, enable us to see this system as a new contribution toward the design of an incremental and robust decision system. Finally, only *Support Vector Machines* are associated each neuron of the IGNG network. Multiplying the classifiers on this level will probably make it possible to improve the results obtained and allow us to consider various and cooperative decision task strategies for complex problems.

References

[AOC+00] S. Adam, JM. Ogier, C. Cariou, R. Mullot, J. Labiche, and J. Gardes. Symbol and character recognition : application to engineering drawings. *International Journal on Document Analysis and Recognition*, pages 89–101, 2000.

[Ba01] R. Britto and al. A two-stage hmm based system for recognizing handwritten numeral strings. In *Proc. 6th ICDAR*, pages 396–400, 2001.

[BB82] D.H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.

[BV97] H. U. Bauer and T. Villmann. Growing an hypercubal output space in a self-organizing feature map. *IEE TNN*, (8):218–226, 1997.

[CCCJ04] C. Chih-Chung and L. Chih-Jen. Libsvm – a library for support vector machines. Technical report, National Tawain University, 2004.

[Dui02] R.P.W Duin. The combining classifier: to train or not to train. In IEEE Computer Society, editor, *ICPR*, pages 765–771, Quebec, 2002.

[Fri94] B. Fritzke. Growing cell structures — A self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[Fri95a] B. Fritzke. Growing grid - a self organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letter*, (2):9–13, 1995.

[Fri95b] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. 1995.

[GRB00] G. Giacinto, F. Roli, and L. Bruzzone. Combination of neural and statistical algorithms for supervised classification of remote-sensing images. *Pattern Recognition Letters*, 2000.

[GS98] M. Gori and F. Scarselli. A multilayer perceptron adequate for pattern recogition and verification. *IEE Transactions on pattern analysis and machine intelligence*, 20:1121–1132, 1998.

[Hay95] S. Haykin. *Neural Networks - A Comprehensive Fundations"*. IEE Press, 1995.

[HPG99] Jean-François Hébert, Marc Parizeau, and Nadia Ghazzali. An hybrid architecture for active and incremental learning: The self-oganizing perceptron (sop) network. In *IJCNN99*, Washington, DC, USA, July 1999.

[JJNH91] R.A. Jacobs, M.J. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Kun02] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume Pattern Recognition, Feb 2002.

[La03] C.L. Liu and al. Handwitten digit recognition : benshmarking of state-of-the -art techniques. *Pattern Recognition*, pages 2271–2285, 2003.

[Mar93] T. Martinetz. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *Proc. ICANN'93*, pages 427–434. Springer, 1993.

[MS91] T. Martinetz and K. Schulten. A "Neural-Gas" network learns topologies. In *Proc. ICANN*, volume I, pages 397–402, Amsterdam, Netherlands, 1991.

[MS94] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7(2), 1994.

[Rib98] A. Ribert. *Structuration Evolutive de données: Application à la construction de classifieurs distribués*. PhD thesis, Université de Rouen, 1998.

[SBR96] S.B. Serpico, L. Bruzzone, and F. Roli. An experimental comparison of neural and statistical nonparametric algorithms for supervised classification of remote-sensing images. *Pattern Recognition Letters*, 1996.

[Vap95] V.N Vapnik. *The nature of statistical theory*. Springer, 1995.