# Graph of neural networks for pattern recognition

Hubert Cardot and Olivier Lezoray
LUSAC, IUT SRC, 120, Rue de l'exode, 50000 Saint-Lo, FRANCE
E-mail: Olivier.Lezoray@info.unicaen.fr,hubert.cardot@chbg.unicaen.fr

## Abstract

*This paper presents a new architecture of neural networks designed for pattern recognition. The concept of induction graphs coupled with a divide-and-conquer strategy defines a Graph of Neural Network (GNN). It is based on a set of several little neural networks, each one discriminating only two classes. The principles used to perform the decision of classification are : a branch quality index and a selection by elimination. A significant gain in the global classification rate can be obtained by using a GNN. This is illustrated by tests on databases from the UCI machine learning database repository. The experimental results show that a GNN can achieve an improved performance in classification.*

## 1. Introduction

Data classification is a central problem in the field of pattern recognition. A lot of methods have been proposed and some of them have become classical ones (decision trees), Bayesian approach, Fuzzy clustering, cluster analysis). They have led to numerous industrial applications. These years, neural networks and more particularly Multi-Layer Perceptrons [3] have received a great deal of attention. Reasons for this success essentially come from their universal approximation capabilities. In this paper, a new strategy for building a neural classifier is introduced. The latter redefines the learning task of a classical large neural network in several simpler ones. Using simpler networks can lead to good generalization abilities without requiring human assistance. The redefinition of the learning task into several smaller ones follows a divide-and-conquer strategy. The paper is organized as follows. Section 2 recalls the basis of induction graphs. Section 3 details the construction of Graph of Neural Network (GNN). From the latter, class discrimination is performed as it will be further discussed. In the last section, we present experimentations on UCI repository datasets [1] and on our own works on in microscopic imaging [4].

## 2. Induction graphs

Decision tree [2] is a non-parametric classification method widely used in pattern recognition. It uses decision functions which partitions the feature space into two regions to determine the identity of an unknown input pattern. These decision functions are organized in such a way that the outcome of successive decision functions refines the decision of classification. The result of the learning process is represented by a tree the nodes of which specify decision functions on attributes values and which leaves correspond to sets of input examples with the same class or to elements in which no more attributes are available. This data classification method is widely used in induction graph theory [8]. Induction graphs are a generalization of decision trees. In a decision tree, the classification decision is made from the root towards the leaves without the possibility of going backward from a node to one lower or higher level node in the tree. Induction graphs enable to introduce links between different level nodes and thus constitute a graph structure. A certain number of articles can be found in literature using a tree structure to build either a neural tree [7] (the nodes of the tree being neurons which are used as non-linear binary decision functions), or neural networks trees [6] (nodes of the tree being neural networks which are used as non-linear n-ary decision functions). We propose to define a new structure based on a graph of neural networks. Unlike the usual methods, we do not build a neural networks tree but a structure whose nodes are neural networks and are completely connected, namely, a neural network induction graph.

## 3. Graph of neural networks

### 3.1. Construction

The construction of the GNN is supervised and based on divide-and-conquer strategy. It builds a set of neural networks (a graph of neural networks). When there is a large number of classes labelling the data, the classification by only one large network can be difficult. This large neural network presents difficulties with generalization. What we

suggest consist in using only small neural networks : when we say "small", we mean simple structure. Since the capacity of generalization of a network is weakened by its complexity, to simplify the problem we reduce the number of classes to recognize : each network has to classify only two classes. Therefore, to discriminate more than two classes, several networks are needed. Our architecture arises in the following way. The neural networks used in this paper are Multi-Layer networks with back-propagation of the gradient error (MLP). The GNN construction is done in three steps : 1) The construction of the neural networks, knowing the number of classes of objects to be separated, 2) the training of each neural network, 3) the construction of the GNN.

### 3.2. Two-class neural networks

For a classification problem with $n$ classes, a set of unconnected networks is built, each one being in charge of separating elements from two distinct classes. The set of different classes is denoted by $C = \{C_1, C_2, ..., C_n\}$ and $|C| = n$. For $n$ classes, that leads to have $(n * (n-1))/2$ neural networks being used for classification. The set of neural networks is given by $\Re = \{\Re_{c_1,c_2}; \Re_{c_1,c_3}; ...; \Re_{c_{n-1},c_n}\}$. The training of each neural network is processed in a sequential and unordered way. That does not have any influence on the training of each network since there is no connection between them at this time. The difficulty in separating $n$ classes is simplified by the specialization of each network, because a network is interested only in the separation of two classes. Consequently, during the learning step, each network learns to recognize only examples of these two classes. This is an advantage compared to only one MLP, since the set of data to be learned is restricted. This implies, on the one hand to simplify the training and on the other hand, to make easier the discrimination between these two classes since the network learnt how to recognize only those. The global training dataset containing patterns of all the different classes is denoted by $S_T$. The latter is divided in several subsets for each neural network. $S_T(c_i, c_j)$ is the dataset which corresponds to the neural network which differenciates the classes $C_i$ and $C_j$ and contains patterns of only those two classes. The initial training data ($S_T(c_i, c_j)$) associated to each neural network is split into two subsets : a learning set ($S_L(c_i, c_j)$) and a validation set ($S_V(c_i, c_j)$). The latter consists in 20% of $S_T(c_i, c_j)$ and the learning set in 80% of $S_T(c_i, c_j)$. The learning of a neural network is performed on $S_L(c_i, c_j)$ and the $S_V(c_i, c_j)$ validation set is used to evaluate the classification rate of the network during the training. Therefore the validation set is not learnt by the neural networks. The structure of the neural networks used is the following one : a layer of inputs containing as many neurons as the number of attributes associated with the object to be classified, a hidden layer containing a variable number of neurons and one output neuron. The value of the output neuron is in the interval $]-1, 1[$. According to the sign of the result associated with this single neuron, an object is classified in one of the two classes that the network separates. The neural networks used by our architecture are very simple (only one hidden layer, only one neuron of output). This has several advantages. The simplicity of the task associated to each neural network simplifies the convergence of the training as well as the search for a simple structure. The generalization of their structure can be made in a dynamic way very easily. Therefore, an automatic method is used to find the number of hidden neurons that gives the best classification rate [3]. Once the training of a $\Re_{c_i,c_j}$ network is carried out, the $Q(\Re_{c_i,c_j})$ classification rate of this network is available. The latter is obtained on the $S_V(c_i, c_j)$ validation dataset and thus relates only to data that have not been learnt. Once all the neural networks are created and trained independently, the GNN is built. Each neural network is connected to all the other ones. This produces a graph of fully connected neural networks. The GNN defines an unweighted and unoriented graph with a structure that enables to know which network is directly reachable from one node of the graph.

## 4. Classification

Once a GNN has been created, each network discriminating only two classes, the problem of the choice of the identity of an input pattern (its final class) arises.

### 4.1. Selection by elimination

Indeed, if an object is proposed to a neural network of the GNN and if the unknown input pattern does not belong to one of the two classes discriminated by a neural network, the answer is not significant and that is likely to distort the decision of classification. To avoid this, we set up a selection by elimination. By selection we mean determination of the identity (i.e. class) of an input pattern and by elimination we indicate the way to choose the class. If a network of the GNN is used, the latter will classify the object in one of the two classes that it differentiates ($C_i$ and $C_j$). If this network indicates the object as belonging to the class $C_i$ then $C_j$ is eliminated and reciprocally. It is the principle of elimination. To classify a pattern X by a GNN, successive interrogations of the neural networks progressively eliminate the possible classes until only one final class is available giving the identity of the unknown input pattern. However, the choice and the sequence of the networks which will carry out the decision of classification has to be precised. This principle of elimination assumes that each interrogation of a neural network eliminates a class and implies that, for $n$

classes to differenciate, a branch in the GNN has a length of $(n-1)$ neural networks. We have now to state how to choose a branch in the GNN that gives the classification decision.

### 4.2. Branch quality index

After the training step, each neural network holds a classification rate (denoted by $Q(\Re_{c_i,c_j})$). When a neural network classifies a new pattern $X$, it gives the value of the output neuron $O(\Re_{c_i,c_j}, X)$. The sign of this value gives the class of $X$ denoted by $C(\Re_{c_i,c_j})$. It will be noted thereafter that if a neural network separates two classes $C_i$ and $C_j$, an input pattern $X$ is considered as class $C_i$ if $O(\Re_{c_i,c_j}, X) < 0$ and $C_j$ if $O(\Re_{c_i,c_j}, X) >= 0$. However, it might be beneficial to move the decision threshold of each neural network according to different factors. Three elements act upon the result given by a network : its potential of classification (the classiciation rate), its decision (the value of the output neuron) and the representativity of the dataset used for the training. Using only the value of the output neuron to assess the relevance of a classification performed by a neural network may cause dubious decisions. We suggest to use a Quality Index which makes a trade-off between all these parameters. For a neural network $\Re_{c_i,c_j}$ and an input pattern $X$, we define

$$QI(\Re_{c_i,c_j}, X) = |O(\Re_{c_i,c_j}, X)| * Q(\Re_{c_i,c_j}) * \frac{|E(\Re_{c_i,c_j})|}{|log(n)|}$$

$$\text{with} \quad E(\Re_{c_i,c_j}) = \frac{|S_L(c_i,c_j)|}{|S_L|} * log\left(\frac{|S_L(c_i,c_j)|}{|S_L|}\right)$$

$|S_L|$ and $|S_L(c_i,c_j)|$ respectively denote the size of the global learning dataset and the specific one associated with the $\Re_{c_i,c_j}$ neural network. $QI$ quantifies the relevance of a given neural network of the GNN for a classification decision. A GNN is a set of connected neural networks, it is therefore possible to define, as for classical graphs, a branch in the GNN. A branch in the GNN is an ordered set of neural networks following the connections between one network to another. A branch in the GNN graph is defined by successive interrogations of neural networks. Since a Quality Index is computed with each network of a branch, we can define a Branch Quality Index which gives the relevance of the set of neural networks used. The Branch Quality Index is defined as the sum of all the Quality Index of the neural networks of the branch and formally given by $BQI(\Omega, X) = \sum_{i=1}^{|\Omega|} QI(\Omega_i, X)$ where $\Omega$ is an ordered set of neural networks and $\Omega_i$ the $i$th network of $\Omega$. However we do not consider as valuable all the different possible branches in the GNN. For each network, the adjacent networks which can be considered in a given branch must not discriminate one of the previously eliminated classes since they have already been eliminated. While building a branch, at a given depth $i$, there are only $\frac{(n-i)(n-i-1)}{2}$ possible adjacent neural networks. Therefore, as stated before, the length of a branch is of $(n-1)$.

### 4.3. BQI criterion maximisation

To reach the best decision of classification using a GNN, one has to use the branch of neural networks which maximizes the BQI criterion. Two strategies can be used. One can choose at each level of classification the adjacent network maximizing the $QI$ value (strategy 1 : figure 1 for a 4 classes problem) . One can also find the global maxima of the whole search-space. This can be done by a genetic algorithm (strategy 2) : a chromosome is defined by a sequence of $(n*(n-1))/2$ gene, each one associated to one neural network. A network is considered as used if the corresponding gene is activated in the chromosome. The genetic algorithm has to find the chromosome maximising the $BQI$ criterion with $(n-1)$ networks on a branch. The fitness function used is the following. For a chromosome $m$, we define $f(m) = g(m) * \sum_i QI(R_i)$ with $g(m) = (-m^2 + 2*(n-1)*m)/(n-1)^2$. The function $g$ is used to ensure that the final number of active genes corresponds exactly to $(n-1)$ which is the wanted length of the branch. The fitness function is therefore maximum with exactly $(n-1)$ networks having the highest $BQI$ sum.

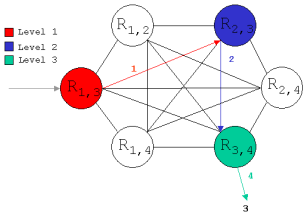


**Figure 1. A branch in the GNN for a 4 classes problem.**

## 5. Experimental results

The databases for which results will be presented here are real data bases coming from the Machine Learning Data Repository of the University of California at Irvine (UCI) [1] and also from our own works on microscopical imaging [4].

### 5.1. UCI databases

These databases are used in various articles on classification. This will enable us to compare the performances of

our architecture with the traditionnal MLP neural network approach. Table 1 describes the different UCI databases

**Table 1. Data bases used for the tests.**

| Database | $n$ | $\|S_T\|$ | $\|S_{Test}\|$ | MLP | GNN |
|---|---|---|---|---|---|
| Wine | 3 | 145 | 35 | 97.14 | 97.14 |
| Vehicle | 4 | 680 | 253 | 66.67 | 69.64 |
| PageBlocks | 5 | 4383 | 1092 | 84.80 | 88.55 |
| Segment | 7 | 176 | 36 | 80.56 | 91.67 |
| Glass | 7 | 176 | 40 | 52.50 | 67.50 |
| Shuttle | 7 | 31501 | 12000 | 79.15 | 95.75 |
| PenDigits | 10 | 7495 | 3499 | 83.82 | 89.03 |
| OptDigits | 10 | 3065 | 760 | 90.39 | 91.05 |
| Letter | 26 | 20001 | 5000 | 62.45 | 76.97 |

and presents the results obtained. Among all the different tests we carried out, it did not appear that there was any difference between the two strategies maximazing the BQI criterion along the branches. This enables to state that the first strategy seems enough general to be used for classification problems. One can note that for all the bases, the GNN makes it possible to obtain better results in all the cases ranging from simpler (few classes) to more complex (many classes) data bases. The GNN performs between 0 and 16.6% better than a traditional MLP. The divide-and-conquer strategy employed coupled with a graph of neural networks thus proves to be very efficient for the classification of data.

### 5.2. Microscopical imaging

To illustrate the ability of the GNN for pattern recognition, it is applied to the recognition of cells in serous cytology. We suggest to use the GNN architecture to build an automatic cellular sorting system for serous cytology. Images are previously segmented (see Figure 2) and the obtained regions are described by 46 attributes ranging of size, shape, color and texture [4].

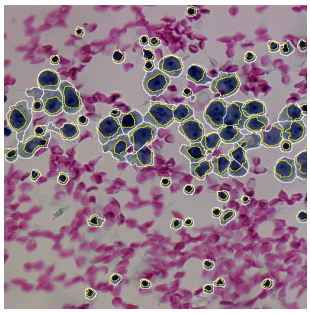

**Figure 2. A cytological color image and the corresponding segmentation.**

For that experimentation, the various types of objects that can be met in serous cytology have been indexed. That represents a rather important number of classes of cells to be recognized (18) and a GNN can help in the recognition. We are therefore interested in the recognition of cells. They must be distributed in the 18 different classes of objects (ranging from normal to abnormal). The isolated cells are classified by a GNN, each network beeing improved by the SFFS attribute selection method [5]. The training of our architecture is carried out on a learning database of 3870 cells and tested on a database of 1967 cells. The total rate of recognition of the GNN after the learning with attribute selection by the SFFS wrapper method is 83.54% for the cells of the test data base (the recognition rate of the GNN without attribute selection is 72.36% which is higher than the one obtained using a single large neural network : 55.90%).

## 6. Conclusion

We have suggested a new neural network architecture based on an induction graph of two classes neural networks. The properties of a GNN for classification and pattern recognition problems has been studied and this new architecture has proved its superiority compared to a traditionnal MLP. In addition to its strength of classification, another interest of the GNN is their particular properties for incremental learning : the whole inducer is not totally rebuilt each time new training data are available.

## References


[1] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[2] L. Breiman, J. Friedman, and R. Olshen. *Classification and regression trees*. Wadsworth and Brooks, California, 1984.

[3] C. Campbell. *Constructive Learning Techniques for Designing Neural Network Systems*. San Diego: Academic Press, 1997.

[4] O. Lezoray, A. Elmoataz, H. Cardot, and M. Revenu. Arctic : An automatic system for cellular sorting by image analysis. In *Proceedings of Vision Interface*, pages 312–319, 1999.

[5] P. Pudil, F. Ferri, J. Novovicovà, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.

[6] A. Ribert, A. Ennaji, and Y. Lecourtier. Building and evaluation of a distributed neural classifier. In *Proceedings of Vision Interface*, pages 582–585, 1999.

[7] A. Sirat and J. Nadal. Neural trees : a new tool for classification. *Network*, 1:198–209, 1990.

[8] A. Zighed, R. Rakotomalala, and S. Rabasda. A discretization method of continuous attributes in induction graphs. In *Proceedings of the 13th European Meetings on Cybernetics and System Research*, pages 997–1002, 1996.