CLANN: Concept Lattice-based Artificial Neural Network for supervised classification

Norbert Tsopzé[1,2], Engelbert Mephu Nguifo[2], and Gilbert Tindo[1]

[1] Université de Yaoundé I, Faculté des Sciences,
Département d'Informatique BP 812
Yaoundé - Cameroun
[2] CRIL - CNRS, IUT de Lens,
SP 16, Rue de l'Université, 62307 Lens Cedex, France

Abstract. Multi-layer neural networks have been successfully applied in a wide range of supervised and unsupervised learning applications. As they often produce incomprehensible models they are not widely used in data mining applications. To avoid such limitations, comprehensive models have been previously introduced making use of an apriori knowledge to build the network architecture. They permit to neural network methods to deserve a place in the tool boxes of data mining specialists. However, as the apriori knowledge is not always available for every new dataset, we hereby propose a novel approach that generates a concept semi-lattice from initial dataset, to directly build the neural network architecture. Carried out experiments showed the soundness and efficiency of our approach on various UCI.

1 Introduction

Neural network (also called connexionnist network) technique is one of the most used techniques in machine learning. Feed-forward neural networks have been applied to solve many problems: handwritten caracters recognition, molecular biology, etc. There exists a plethora of architectures and algorithms about neural networks. But it is very difficult to choose those which are the best for a given task [7]. Finding the architecture of the network to be used for solving a given problem is a very complicated task. In fact there is no existing exact method for defining the number of layers, the number of neurons in each layer and the connections between neurons [7]. Define the topology of neural network consists of answering certain questions: how many hidden layers? how many neurons per layer? which connection policy? how to define each unit threshold? Moreover, only the decision aspect of the neural model (which consists in using the model in the decision task) is used and no importance is given to its descriptive aspect (how decision is taken). Besides, the intelligibility of learned model is fundamental in datamining [3]; the absence of an explanation capability limits its use [1]. Black box neural network is usually not comprehensible. In [1] authors enhance the transparency of the network in the decompositional extraction rules process.

The topology becomes important for the intelligibility of the neural model and in the process of extracting rules from neural network model.

When tackling a supervised classification problem, the widely used approach (ad'hoc) to define a feed-forward network topology consists in using three layers: input, hidden, output. The number of neurons of the input layer is the number of variables or attributes. The number of neurons of the output layer is the number of classes. The number of neurons in the hidden layer is calculated as the mean of the number of input units and the output units. In the literature, different approaches have been reported to build the network topology. These research works could be divided into two groups:

1. Search an optimal network to minimize the number of units in the hidden layers [11]. This technique brings out a constructive solution to the problem without an apriori knowledge. Its main limitation is the intelligibility of the resulting network, the network is a black box i.e. no semantic is associated to each node.
2. Use a set of apriori knowledge on the problem domain and build neural network from this knowledge [13]. The main advantage here is that the result is a comprehensive network i.e. each node in the network represents one variable in the rules set and each connexion between two nodes represents one dependence between variables. But it is not possible to use it when the apriori knowledge is not available.

The aim of this paper is to introduce CLANN, an approach based on concept lattices which goes beyond the limitations of the existing approaches. Concept lattices is an ordered graph composed by formal concepts. Given an input matrix (also called formal context) specifying a set of objects and their corresponding properties, a formal concept is a pair containing both a subset of objects (X) and a subset of properties (Y), such that Y is the set of all properties shared by the objects of X, and X is the set of all objects that share the properties of Y. In fact concept lattices has been extremely used in supervised classification [10]. The use of the concept lattices [5] in classification proceeds in two phases: the first phase called the training phase consists in selecting interesting concepts among the set of concepts or in extracting rules. Selected concepts and/or rules are then used to take decision about new objects in the second phase [10].

CLANN uses directly the Hasse diagram of the built semi-lattice to define the architecture of the neural network. Within data mining domain, concepts and connexions defined in this diagram are generally used to extract rules into the formal context. Here concepts and connexions are used to define neurons and their connexions. We believe that this approach can be helpful when the apriori knowledge is not available.

Many constraints (or heuristics) presented in the literature can be used to prune the concept lattices in order to reduce the number of generated concepts. The CLANN approach has two main advantages: first it finds a comprehensive model when the knowledge is not available, secondly it is possible to justify the built neural network topology. Another advantage of this approach is the

possibility to easily extract rules from the network. Carrying out experimentations of this method using benchmarks from UCI repository shows interesting results in terms of precision, and also that CLANN can be better than many comprehensive models such as $C4.5$ and $IB1$.

The rest of this paper is organized as follows: the next section presents some notions used in Formal Concept Analysis (FCA). The third section presents the CLANN approach. Experimentation and obtained results are described in section four. Related works about extracting rules in trained neural networks end this paper.

2 PRELIMINARIES

2.1 FCA basic notions

This section presents some important notions used in FCA. More notions about FCA could be found in [5].

A **formal context** is a triplet $C = (O, A, I)$ where O is a non empty finite set of objects, A a non empty finite set of attributes (or items) and I is a binary relation between elements of O and elements of A (formally $I \subseteq \{(o, a) / o \in O \text{ and } a \in A\}$). A context C can be represented as binary matrix M (where $M_{i,j} = 1 \iff (o, a) \in I, 0$ otherwise) or transactions database (a row o is the collection of attributes which are verified by the object o). A(resp. O) is also called set of items (resp. transactions). Table 1 is an example of context. We denote in the next sections the set $\{a, b, c, d\}$ $(resp. \{1, 2, 3, 4\})$ as abcd (resp.1234). The objects in table 1 could be divided into two parts: a set of examples O^+ (for instance $O^+ = \{1, 2, 3, 4, 5, 6\}$ for the context table 1) and the set of counterexamples O^- (for instance $O^- = \{7, 8, 9, 10\}$ for the context table 1).

Table 1. Example of context presented as boolean matrix; two-class (+ and -) data indicated by class column.

Objects	a	b	c	d	e	f	class
1	1	1		1	1	1	+
2	1		1		1	1	+
3		1	1	1	1		+
4		1	1	1			+
5	1	1		1	1		+
6	1		1		1		+
7		1		1		1	-
8		1			1	1	-
9			1		1	1	-
10	1			1	1		-

Let f *and* g be two applications defined as follows: $f:\ 2^O \longrightarrow 2^A$, such that $f(X) = X' = \{a \in A \ / \ \forall o \in X\ ,\ (o,a) \in A\}\ ,\ X \subseteq G$ and $g:\ 2^A \longrightarrow 2^O$, such that $g(Y) = Y' = \{o \in O \ / \ \forall a \in Y\ ,\ (o,a) \in I\}, Y \subseteq A$; a pair (X,Y) is called **concept** iff $X = Y'$ *and* $Y = X'$. X (resp. Y) is called **extension** (resp. **intention**) of the concept, an intention of concept is also called description.

Let (X_1,Y_1) and (X_2,Y_2) be two concepts, and $\leq$ a relation defined on a entire set of concepts extracted from the context. $(X_1,Y_1) \leq (X_2,Y_2)$ if $X_1 \subseteq X_2\,(Y_1 \supseteq Y_2)$. (X_1,Y_1) is called **successor** of (X_2,Y_2) and (X_2,Y_2) **predecessor** of (X_1,Y_1) if there is no concepts between them. The relation $\leq$ defines an order relation on the entire set L of concepts [5]; the set of all concepts L with the order relation $\leq$ define the concept lattices.

A lattice is a partially ordered set (or poset) in which every pair of elements has a unique supremum (the elements' least upper bound; called their join) and an infimum (greatest lower bound; called their meet). A semilattice is a partially ordered set (poset) closed under one of two binary operations, either supremum (join) or infimum (meet). Hence we speak of either a join-semilattice or a meet-semilattice. If an ordered set is both a meet- and join-semilattice, it is also a lattice.

2.2 Supervised classification

In a supervised classification process, the system works into two phases [10]: training (learning) phase and testing (evaluation) phase. The dataset is separated into two subsets, the first subset is used to build and train the model during the training phase while the second is used to evaluate the model during the testing phase. For instance, the dataset of table 1 could be divided into two subsets $\{1,2,5,6,9,10\}$ as training set and $\{3,4,7,8\}$ as test set. In the literature, different techniques were reported for supervised classification problem [7]: instance-based learning, decision trees, artificial neural network, support vector machine, bayesian network, ... We are here concerned with artificial neural network.

2.3 Feed-forward neural networks

Neural networks is a set of interconnected neurons (also called units), which exchange information with one to another and communicate with the external environment. Different type of neural networks were reported in the literature [2]: the feed forward neural networks (perceptron, multilayer perceptron, ...) and the reverse feed forward neural networks (for instance the "Adaptive resonance theory" networks). More notions about artificial neural networks can be found in [2]. A feed forward neural network is composed of different layers:

1. The input layer is composed of neurons which receive information from outside.
2. The internal layer is composed of units which make intermediate treatment; it could be composed of many layers.

3. The output layer is formed by units which make decision.

A neuron is a process unit which has an internal memory, communicates with the external environment. Its state is defined as a function of its inputs; this function is called activation function [2]. The connexions between neurons define the architecture of the network. The neuron can also be called unit, it is active if its internal state value is 1 and inactive if this value is 0. After defining the network topology, the connexion weights between units are trained by backpropagation [12]. The connexion weights between two units a and b define the effect of the neuron a on the neuron b.

3 CLANN APPROACH

We describe in this section the different steps of our new approach as shown by figure 1. The process of finding the architecture of neural networks are three-folds: (1) build a join semi-lattice of formal concepts by applying constraints to select relevant concepts; (2) translate the join semi-lattice into a topology of the neural network, and set the initial connections weights; (3) train the neural network.

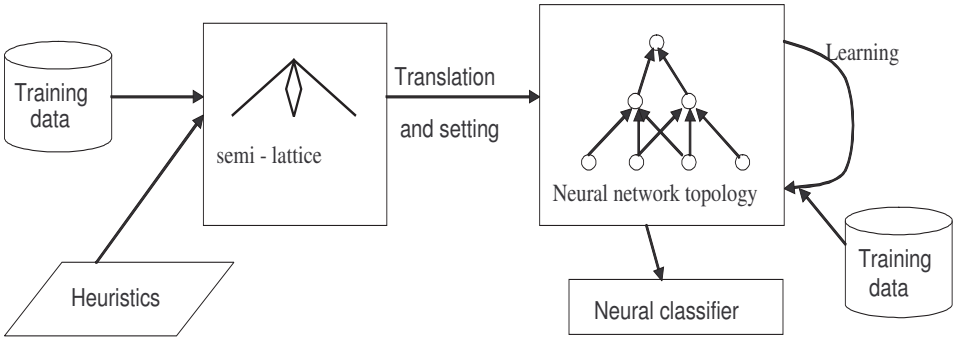

Fig. 1. Neural network topology definition.

3.1 Semi-lattice construction

There are many algorithms [8] which can be used to construct concept lattices; few of them build the Hasse diagram. Lattice could be processed using top-down or bottom-up techniques. In our case, a levelwise approach presents advantage to successively generate concepts of the join semi-lattice and the Hasse diagram. For this reason, we choose to implement the Bordat algorithm [8] which is suitable here. Concepts included in the lattice are only those which satisfy the defined constraints.

Algorithm 1 Modified Bordat algorithm

Require: Binary context K

Ensure: concept lattices (concepts extracted from K) and the Hasse diagram of the order relation between concepts.

1: Init the list L of the concepts $(G, \{\})$ $(L \leftarrow (G, \{\}))$
2: **repeat**
3: **for** concept $c \in L$ such that his successors are not yet been calculated **do**
4: Calculate the successors c' of c.
5: **if** the specified constraint is verified by c' **then**
6: add c' in L as successor of c if c' does not exit in L else connect c' as successor of c.
7: **end if**
8: **end for**
9: **until** no concept is added in L.
10: derive the neural network architecture as described in section 3.3 from the concept semi-lattice.

3.2 Constraints

In order to reduce the size of lattice and then the time complexity, we present a few constraints regularly used to select concepts during the learning process.

Frequency of concept. A concept is frequent if it contains at least α (also called minsupp is specified by the user) objects. The support s of a concept (X, Y) is the ratio between the cardinality of the set X and the total number of objects ($|O|$) ($s = \frac{100 \times |X|}{|O|}\%$). Frequency is an anti-monotone constraint which helps in pruning the lattice and reduce it computational complexity. Support could be seen as the minimal number of objects that the intention of one concept must verified before being taken in the semi-lattice. The figure 3.2 presents a semi-lattice built from the class "+" examples of the table 1 (a) and the equivalent topology of the neural network (b); the specified minsupp value is 30%.

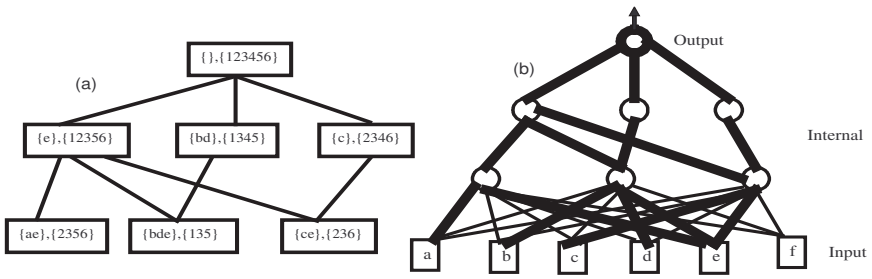

Fig. 2. (a) Join semi-lattice with minsup=30% and (b) network topology.

Validity of concept. Many techniques are used to reduce the size of lattice. The following notions are used in order to the select concepts: a concept (X, Y) is **complete** if Y recognize all examples in dataset. A concept (X, Y) is **consistent** if Y throws back all counterexamples (formally, the set of consistent concept is $\{(X, Y)/Y \cap O^- = \{\}\}$ where $O = O^+ \cup O^-$). To reduce the restriction imposed by these two constrains, others notions are used:

1. **Validity**. A concept (X, Y) is valid if its description recognizes most examples; a valid concept is a frequent concept on the set of examples O^+; formally the set of valid concepts is defined as $\{(X, Y) / |X^+| \geq \alpha\}$ where $0 < \alpha \leq |O^+|$.
2. **Quasi-consistency**. A concept (X, Y) is quasi-consistent is if it is valid and its extension contains few counterexamples. Formally the set of quasi-consistent concepts is defined as $\{(X, Y) / |X^+| \geq \alpha \; and \; |X^-| \leq \beta\}$.

Height of semi-lattice. The level of a concept c is defined as the minimal number of connexions from the supremum concept to c. The height of the lattice is the greatest value of the level of concepts. Using levelwise approach to generate the join semi-lattice, a given constraint can be set to stop concept generation at a fixed level. The height of the lattice could be performed as the depth without considering the cardinality of concepts extension (or intention). In fact at each level, concept extensions (or intentions) do not have the same cardinality. The number of layers of the semi-lattice is a parameter corresponding to the maximum level (height) of the semi-lattice.

3.3 From Semi-Lattice to Neural network

Mapping Hasse diagram of lattice into neural networks is described as following:

- Each lattice concept becomes one unit (or node) in the neural network; a node is thus seen as a group of objects which verify a given set of attributes.
- A layer of n units (n is the number of attributes) is created as the input layer of the system; each neuron of this layer is connected to any neuron of the first hidden layer (concepts in the semi-lattice, with no successor) inside internal layers.
- The supremum concept of lattice is translated into the neural unit representing the output of the network; this concept is the one whose extension contains all the training set of objects.
- Other concepts are rewritten as hidden units (or hidden layers). The units which do not have successor constitute the first hidden layer.
- There is a connection between two neurons if and only if there is a link between their associated concepts in the join semi-lattice.

3.4 Connection weights and threshold

Connection weights are initialized as follows:

– Connection between two nodes, that derived directly from the semi-lattice is initially weighted to 1. This implies that when the node (or neuron) is active, all its predecessors are active too.
– Connection between an input layer unit and hidden layer unit is weighted as follows: 1 if the attribute associated by the input layer node appears in the intention Y of concept associated to the hidden layer node, and -1 otherwise.
– Threshold is set to zero for all nodes (or units).

These connection weights are modified during the training process, using the learning algorithm (which is the backpropagation algorithm [2] in our case). The activation function is also a parameter of CLANN approach. We choose to implement the sigmoid function among those reported in the literature [2].

4 EXPERIMENTATIONS AND RESULTS

This section presents the experimental comparisons conducted with our method. CLANN has been implemented in C++, and run on a personal computer Pentium IV (1,8Ghz, 1Go RAM and 80Go Hard Disk) in the Linux fedora core environment. Table 2 describes the two-class datasets with contain nominal attributes, taken from UCI repository [3]. We use the Weka binarization procedure "NominalToBinary" to convert multivalued attribute to binary one.

Dataset	#Nominal attributes	#Binary attributes	training	test	data-size
Spect	23	23	80	187	267
Chess end-of-game	36	38	10-CV	10-CV	3196
Tic-tac-toe	9	27	10-CV	10-CV	958
Monsk1	7	15	124	432	556
Monsk2	7	15	168	432	600
Monsk3	7	15	122	432	554

Table 2. Experimented Datasets; '10-CV' indicates that training and test data are defined by 10-fold cross validation technique

Neural network obtained from the Hasse diagram of the join semi-lattice are trained with backpropagation algorithm using 500 iterations. Activation function used in the experiments is the sigmoid function ($f(x) = \frac{1}{1+\exp(x)}$). The experimentations were done using 10-fold cross validation for datasets with none already defined test set as Chess and Tic-tac-toe. For other datasets, our model is built and trained using provided training dataset, and evaluate on test set. CLANN results are depicted by three tables : 3, 4 and 5. These tables present the accuracy rate which is the percentage of correctly classified test objects. Table 3 shows the results obtained on one dataset (SPECT) with the different constraints and various parameter values. Considering the frequency or validity

[3] available on the web site http://www.ics.uci.edu/AI/ML/MLDBRepository.html

constraint, when the minimum support (δ or α) decreases, the accuracy rate increases. This is certainly due to the fact that the number of hidden nodes also increases when the minimum support decreases. With a high number of hidden nodes, the neural network model produces good perfomance. Considering the quasi-consistency constraint, when its minimum value decreases, the accuracy rate slightly decreases. The effect of this constraint is not so significant in our case. Considering the semi-lattice height constraint, the accuracy rate slightly increases until level 2, and then significantly decreases. Thus it is not necessary to build more than two levels from of the semi-lattice. We choose to use this constraint (setting the default value to 1) in order to compare CLANN to other machine learning classifiers.

Table 3. Results obtained from SPECT database using different constraints.

Constraints	Generalization	Constraints	Generalization
α=10, β=10	93,74%	δ=20	93,59%
α=20, β=10	92,60%	δ=30	93,59%
α=30, β=10	91,30%	δ=40	90,65%
α=40, β=10	90,4%	δ=50	89,84%
α=10, β=50	93,90%	$height$=1	93,60%
α=20, β=50	93,90%	$height$=2	93,90%
α=30, β=50	92,40%	$height$=3	92,40%
α=40, β=50	88,40%	$height$=4	90,65%

Table 4 presents accuracy rates of CLANN and four other classifiers taken from the Weka datamining tool [4] among which two decision tree method (C4.5 and ID3), one instance based learning method (IB1), and one neural network method (MLP). MLP is built with the ad-hoc method as described in the section 1 and trained using gradient backpropagation algorithm [12]. Considering the accuracy rate average over the datasets used, CLANN is never the last classifier when ranking them on the accuracy rate. It is always better than at least one other classifier. MLP outperforms all the other classifier in terms of accuracy rates, however it is not a comprehensible model for datamining. CLANN results are comparable to decision-tree ones, and CLANN outperforms IB1. Considering the SPECT dataset, CLANN significantly outperform all those classifiers.

Table 5 presents comparative results between CLANN and other constructive neural networks models [11] among which Tiling, Upstart, Tower, Distal. Those constructive neural networks are still non compresensible model for data mining. Our comparison is only to analyse the soundness of our approach. The three first algorithms construct the neural model by successive addition of neurons until the network has a maximum number (specified by the user) of layers or the desired accuracy is obtained. The parameters used are described as following: the maximum number of layer is 10 and the desired accuracy is 100% (default value

[4] available on the web site http://www.cs.waikato.ac.nz/ml/weka/

Table 4. Accuracy rate of CLANN versus other standard models

Dataset	CLANN	MLP	C4.5	ID3	IB1
Spect	93,90%	65,77%	66,7%	65,24%	66,31%
Chess	93,60 %	99,30%	98,30%	97,80%	89,90%
Monsk1	82,70%	100%	100%	92,59%	89,35%
Monsk2	78,91%	100%	70,37%	86,57%	66,89%
Monsk3	83,61%	93,52%	100%	89,81%	81,63%
Tic-tac-toe	83,57%	96,86%	93,21%	93,84%	81,63%
Average	86,05%	92,56%	88,10%	87,64%	79,29%

used in their implementation), added unit is trained by Pocket Perceptron with racket modification [11]. Distal uses one hidden layer and clusters the training set into disjoint groups and each group becomes one neural unit of the hidden layer. Distal is similar to our approach. They differ by the fact that: Distal uses only one hidden layer, Distal uses disjoint group while the node extension in CLANN may overlap and furthermore each node in CLANN model is clearly characterized by an intention.

Table 5. Classification rate of CLANN versus other constructive neural network models

Dataset	CLANN	Tiling	Upstart	Tower	Distal
Spect	93,90%	89,60%	83,29%	71,40%	93,90%
Chess	93,60%	93,90%	90,65	92,40%	89,74%
Monsk1	82,70%	83,13%	77,78%	85,85%	90,23%
Monsk2	78,91%	77,13%	87,30%	66,90%	89,10%
Monsk3	83,61%	74,91%	87,22%	82,08%	86,46%
Tic-tac-toe	83,57%	98,40%	99,89%	100%	95,85%
Average	86,05%	86,17%	87,67%	83,11%	90,88%

From table 5, considering the accuracy rate average, Distal outperforms all the classifiers, CLANN outperforms Tower and is comparable to Tiling and Upstart. As shown in the previous comparison, CLANN is better on the SPECT dataset. CLANN outperforms Distal on CHESS dataset. In addition of these results, CLANN built a comprehensive model and is more suitable for data mining than the other constructive approaches. In fact concept lattices can be useful in rules extraction process [6]. The other constructive neural network model as well as MLP are seen as black box by the user, and their interpretation remains a difficult task. Providing explanation when dealing with datamining applications brings up an added-value [1, 3], which will be easily obtained with CLANN than other methods.

KBANN [13] is not compared to CLANN because the apriori knowledge about the previous dataset (table 2) are not available.

5 RELATED WORKS

To the best of our knowledge, KBANN [13] is the only algorithm able to build comprehensive neural network. KBANN derives from the knowledge provided by the user as set of rules, the network topology. Unlike KBANN, CLANN does not need apriori knowledge, its model is based on semi-lattice built directly from data.

Research works about rules extraction from neural networks presented two main algorithms for this task. These algorithms [1] execute on neural model according to the assumptions which stipulate that neurons are maximally active (activation near 1 or 0) and that training does not significantly alter the meaning of the neuron, implicite rules present in the model after mapping Hasse diagram into neural network will not significantly change after training. These algorithms are:

1. Subset. Its principle [1] consists of clustering internal links of hidden and output units, into subsets of links, and then using only the subsets of links whose the sum of weights is greater than the unit threshold. Extracted rules are in written in the form "If...then..."
2. MofN. It assumes that individual links are not important and weights form small number of clusters. It clusters the connections for each hidden and output neuron and delete those clusters which can not affect the unit result. It finds the average weight for each remaining cluster and retrain the bias of each cluster. Finally a single rule for each hidden and output neuron X is derived. The rules are in the form "If N of the M antecedents are active then X is active".

Each of these algorithms could be applied to extract rules in CLANN model.

As the architecture of CLANN model is based on semi-lattice, extracting rules using concept lattices approaches [9] could also be applied. Each node of CLANN model contains a closed set and the connexion weight between two nodes could help to weight the extracted rules.

6 CONCLUSION

This work proposes an approach to define a constructive comprehensive neural network without an apriori knowledge. Experiments conducted has shown the soundness and efficiency of our approach.

CLANN is limited to the binary data set. Binarization techniques could be applied on dataset such that CLANN is able to handle nominal, discrete and continuous values. Various approaches are proposed in the literature to deal with many-valued context directly when generating concept lattice. Further research will focus on those approaches. Many constraints are also presented but it is not easy to define the default values. Extensive experimental study is currently ongoing to find out the default values. Computational complexity of CLANN is similar to those of Weka MLP, this complexity will be further explicitly explored.

CLANN is also limited to two-class supervised classification problem. Ongoing research is dealing with multi-class design, constraint criteria setting and rules extraction from CLANN topology.

Acknowledgment

This research is partially supported by the French Embassy service SCAC (Service de la Coopération et d'Action Culturelle) in Yaounde. Thanks to the anonymous reviewers for their helpful comments.

References

1. R. Andrews, J. Diederich, A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 8(6): 373-389, 1995.
2. Y. Benani. Apprentissage Connexionniste, Editions Herms Science, Paris 2006.
3. M.W. Craven and J.W. Shavlik. Using Neural Networks for Data Mining. Future Generation Computer Systems, 13(2-3): 211-229, 1997.
4. S. Gallant. Perceptron based learning algorithms. IEEE Transactions Neural Networks, 1:179-191, 1990.
5. B. Ganter and R.Wille. Formal Concepts Analysis: Mathematical foundations. Springer - Verlag,1999.
6. G. Gasmi, S. Ben Yahia, E. Mephu Nguifo, Y. Slimani. A new informative generic base of association rules. Advances in Knowledge Discovery and Data Mining, 3518:81-90,2005.
7. J. Han, M. Hamber. Datamining: Concepts and Techniques. Morgan Kauffman Publishers, London, 2001.
8. S. Kuznetsov, S. Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices, JETAI 14(2/3):189-216,2002.
9. J.L Guigues and Vincent Duquenne, 'Familles minimales d'implications informatives resultant d'un tableau de donnes binaires, Mathmatiques et sciences sociales, 95:5-18 1986.
10. E. Mephu Nguifo, P. Njiwoua. Treillis de concepts et classification supervisée. Techniques et Sciences Informatiques, 24:449-488, 2005.
11. R. Parekh, J. Yang, V. Honavar. Constructive Neural-Network Learning Algorithms for Pattern Classification. IEEE Transactions on Neural Networks, 11(2): 436-451, 2000.
12. D.E. Rumelhart, G. E. Hinton, R. J. Williams. Learning representations by back-propagating errors. Nature,323: 318-362, 1986.
13. J.W. Shavlik, G. Towell. Kbann: Knowledge based articial neural networks, Artificial Intelligence, 70:119-165, 1994.