M-CLANN: Multiclass Concept Lattice-Based Artificial Neural Network

Engelbert Mephu Nguifo[1,3], Norbert Tsopze[1,2], and Gilbert Tindo[2]

Abstract. Multilayer feedforward neural networks have been successfully applied in different domains. Defining an interpretable architecture of a multilayer perceptron (MLP) for a given problem is still challenging. We propose a novel approach based on concept lattices to automatically design a neural network architecture. The designed architecture can then be trained with the backpropagation algorithm. We report experimental results obtained on different datasets, and then discuss our contribution as a means to provide semantics to each neuron in order to build an interpretable neural network.

1 Introduction

A growing number of real world applications have been tackled with artificial neural networks (ANNs). ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. ANNs offer a powerful and distributed computing architecture, with significant learning abilities and they are able to represent highly nonlinear and multivariable relationships. ANNs have been successfully applied to solve a variety of specific tasks (pattern recognition, function approximation, clustering, feature extraction, optimization, pattern matching

Engelbert Mephu Nguifo and Norbert Tsopze
CRIL CNRS, Artois University, Lens, France
e-mail: `tsopze@cril.univ-artois.fr`

Norbert Tsopze and Gilbert Tindo
Computer Science Department - University of Yaoundé I,
PO Box 812 Yaoundé - Cameroon
e-mail: `tsopze@cril.fr;gtindo@uycdc.uninet.cm`

Engelbert Mephu Nguifo
LIMOS CNRS, Université Blaise Pascal, Clermont Ferrand, France
e-mail: `mephu@isima.fr`

and associative memories) of importance to many applications [28, 39]. ANNs are useful especially when data is plentiful and prior knowledge is limited. Different ANN types have been reported in the literature, among which the multilayer feed-forward network, also called multi-layer perceptron (MLP), was the first and arguably simplest type of ANN devised, and is the main concern of this chapter.

MLP networks trained using the backpropagation learning algorithm are limited to search for a suitable set of weights in an apriori fixed network topology. The selection of a network architecture for a specific problem has to be done carefully. In fact there isn't a fixed and efficient method for determining the optimal network topology of a given problem. Too small networks are unable to adequately learn the problem well while overly large networks tend to overfit the training data and consequently result in poor generalization performance. In practice, a variety of architectures are tried out and the one that appears best suited to the given problem is picked. Such a trial-and-error approach is not only computationally expensive but also does not guarantee that the selected network architecture will be close to optimal or will generalize well. An ad-hoc and simple manner deriving from this approach is to use one hidden layer with a number of neurons equal to the average number of neurons in both input and output layers. In the literature, different automatic approaches have been reported to dynamically build the network topology. These works could be divided into two groups:

1- The first group uses prior knowledge (set of implicative rules) of the application to derive the neural network topology [32]. The prior knowledge is provided by an expert of the domain. The main advantage here is that each node in the network represents one variable in the rule set and each connection between two nodes represents one dependency between variables. The obtained neural network is a comprehensible ANN since each node is semantically meaningful, and the ANN's decision is not viewed as deriving from a black-box system, but could easily be explain using a subset of rules from the prior knowledge. The KBANN system (Knowledge-Based ANN) [32] is an example of such an approach. But this solution is limited while the prior knowledge is not available as might be the case in practice.

2- The second group of techniques searches for an optimal network to minimize the number of units in the hidden layers [28, 34]. These techniques bring out a dynamic solution to the ANN topology problem when a priori knowledge is not available. One technique suggests to construct the model by incrementally adding hidden neurons or hidden layers to the network until the obtained network becomes able to better classify the training data set. Another technique is network pruning which begins by training an oversized network and then eliminate weights and neurons that are deemed redundant. An alternative approach consists of using the linear separability [3] approach or the genetic approach [8] even if the latter is computationally expensive. All these (incremental, pruning, genetic) techniques result to neural networks that can be seen as black box systems, since no meaning is associated to each hidden neuron. Their main limitation is the intelligibility of the resulting network (black-box prediction is not satisfactory [1, 11]).

We propose here a novel solution, M-CLANN (Multi-class Concept Lattices-based Artificial Neural Networks), to build a network topology where each node has an associated semantic without using any prior knowledge. M-CLANN is an extended version of the CLANN approach [35]. Both approaches uses formal concept analysis (FCA) theory to build a semi-lattice from which the NN topology is derived and trained by error backpropagation. The main difference between M-CLANN and CLANN are two-folds. First M-CLANN can deal with multi-class classification problems, while CLANN is limited to two-classes. Second, the derived topologies from the semi-lattice are different in both systems.

Our proposed approach presents many advantages: (1) the generated architecture is a multi-layer feed-forward network, such that the use of the backpropagation algorithm is obvious; (2) each neuron has a semantic as it corresponds to a formal concept in the semi-lattice, which is a way to justify the presence of a neuron; (3) each connection (between input neuron and hidden neuron, and between neurons of different hidden layers) in the derived ANN also has a semantic as it is associated to a link in the Hasse diagram of the semi-lattice; (4) the knowledge for other systems (such as expert systems) could be extracted from the training data through the model; (5) Experimental results have shown the efficiency of the approach compared to other well-known techniques.

The rest of this chapter is organized as follows: the next section provides an overview of some related works. Section three recalls some background knowledge on formal concept analysis theory and supervised classification; the fourth section describes our approach M-CLANN (Multiclass CLANN). Experimental studies are reported in section five. Section six discusses the soundness and efficiency of our approach.

2 Related Works

Research works about neural network architecture design could be divided into two groups as mentioned above. The first group uses prior knowledge to propose a MLP topology, while the second group searches for an optimal topology minimizing the number of hidden neurons and layers.

2.1 Defining Neural Topology Using Prior Knowledge

An interesting framework is proposed in [32] to design the ANN topology using the domain theory represented as a set of rules. The derived system

KBANN (Knowledge Based Artificial Neural networks) [32] use a set of rules represented as a set of Horn clauses. From these rules an items hierarchy is defined and the architecture of the neural network is derived from this hierarchy. The hierarchy between items is defined using the following equivalences:

1. Final conclusions $\Leftrightarrow$ output units;
2. Intermediate conclusions $\Leftrightarrow$ internal units;
3. Hypothesis $\Leftrightarrow$ input units;
4. Dependency between items $\Leftrightarrow$ connection links.

The different steps of KBANN are as follows:

1. Rewriting. This step consists of writing the rules such that disjuncts are expressed as a set of rules (each rule has only one antecedent).
2. Mapping. The hierarchy between items is defined and directly mapped to the network.
3. Labeling. Each unit is numbered by its level.
4. Adding new hidden units. In order to make the network able to learn derived features not specified in the initial rule set, it is advised to add new units in the hidden layer.
5. Adding input units. Some relevant features which are referred to by the initial rule set are added.
6. Adding links. Links are added to connect each unit numbered $n-1$ to each unit numbered n. The connection weights of these links are set to 0.
7. Perturbing. A small random number is added to each weight.
8. Initialization of connection weights and ANN training by error backpropagation.

The connection weights and the bias neurons are initialized as follows:

- w for the positive antecedents
- $-w$ for the negated antecedents.
- The bias on the unit corresponding to the rules consequent to $(p-1/2)w$ where p is the number of positive antecedents of the unit.

w is a positive number having 4 (empirically defined) as the default value.

Example 1. *Figure 1 presents a simplified example of defining the neural topology by KBANN approach. In the first column, are the initial rules which are rewritten and presented in the second column. And the final network is presented in the third column.*

2.2 Defining Neural Topology without Prior Knowledge

When the prior knowledge is unavailable, it is not possible to use KBANN. To avoid this, there are reported methods that directly define the ANN topology

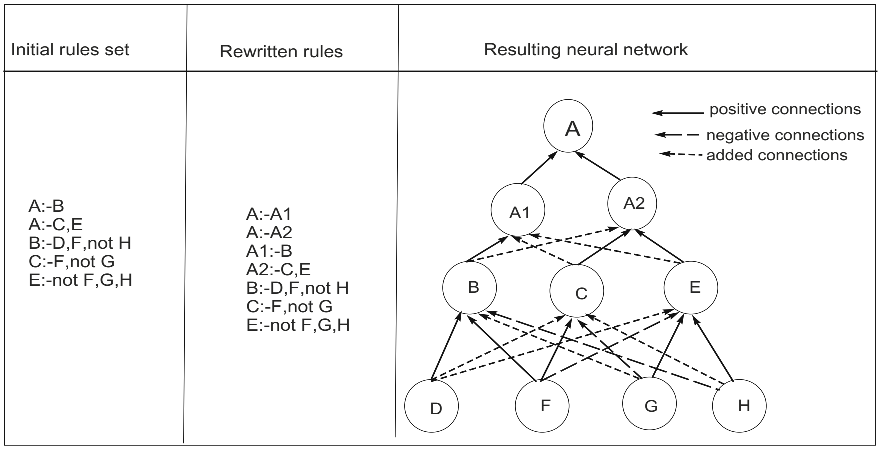

Fig. 1 Example of ANN topology definition with KBANN.

from the data. These methods start with a small network and dynamically grow the network by adding and training neurons as needed until better classification is achieved. These methods can be divided into two subgroups: those with many hidden layers [28] and those with only one hidden layer [38].

2.2.1 Many Hidden Layers

In [28] the authors provide a survey of these methods including MTiling, MUpstart, and MTower. The new added neuron is trained using a variant of perceptron similar to the pocket perceptron with rachet modification [15]. The process adds layers in the existing network until better classification is achieved or the maximum number of layers (user specified value) is attempted.

1. MTiling. It constructs a strictly layered network of threshold neurons. Apart from the most top layer (which is also the output layer) which receives inputs from the layer immediately bellow it and to the inputs neurons, each layer receives input from the layer immediately bellow it. Two kinds of neurons are distinguished: the master unit and the ancillary neurons that are added and trained to ensure a faithful representation of the training data. After training, some ancillary neurons could be pruned to minimize the network structure.
2. MUpstart. The network is constructed as a binary tree. Two kinds of errors are defined: wrongly off (*output* = 0 while the *target* = 1) and wrongly on (*output* = 1 while the *target* = 0). In case of wrongly off (rep. on), one left (resp. right) child neuron is added to the wrong neuron and trained to correct this error.

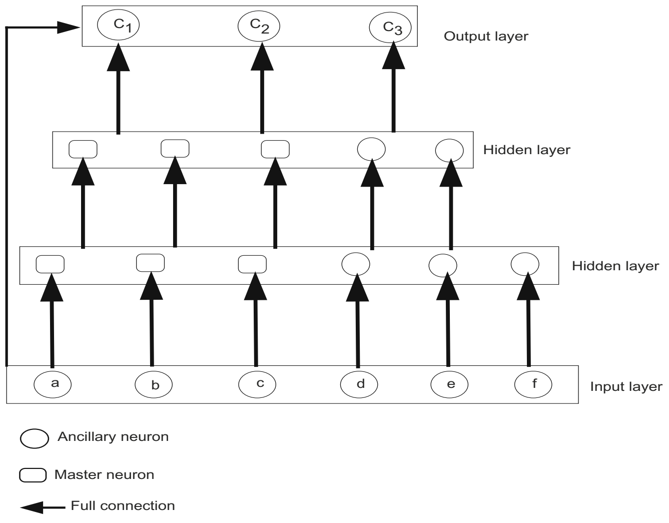

Fig. 2 Example of neural network topology definition by MTiling method.

3. MTower. The resulting network is like a tower. It successively adds new layers in the network until better classification is achieved. The newly added layer is fully connected to the input layer and to the output layer. After connecting this layer, it becomes the new output layer.

Example 2. *Figure 2 is an example of a network constructed by MTiling.*

Recently an approach based on linear separability was introduced in [3] which relies on barycentric correction procedure algorithm for training the individual threshold logic unit.

2.2.2 One Hidden Layer

The Distal method [38] belongs to this category. It builds a 3 layer neural network. Each neuron of the input layer is linked to an attribute. Each neuron of the output layer is associated to a predefined class. The process essentialy consists of defining the hidden layer. Distal clusters training data in disjoint subsets and represents each subset in the hidden layer by one neuron.

Example 3. *Figure 3 presents a neural network defined by Distal. (a) is the initial state of the network and (b) is the generated network.*

There are also in the literature many works which help the user to optimize [37] or prune networks by pruning some connections [23] or by selecting some variables [5] among the entire set of initial variables, or by detecting and filtering noisy examples [34]. These works do not propose an efficient method to build neural network topology, but they can be classified in the second

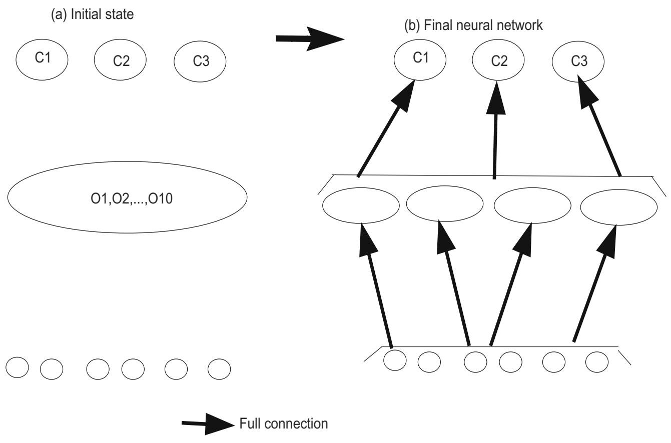

Fig. 3 Example of neural network topology definition by Distal method.

group, since by reducing the number of input neurons, the number of neurons in the hidden layer could also vary.

3 Background - Classification and Formal Concept Analysis

In this section we recall what is supervised classification task, then define basic notions of FCA, and finally presents constraints that can be used to prune the concept lattice in supervised classification.

3.1 Classification

The classification task consists of labelling unknown patterns into a predefined class. The classification process builds a model and trains it for making this model able to affect the unseen patterns to one of the output classes. Here, each known pattern is presented as a pair (x, y) where x is the vector containing different values taken by the pattern on different attributes and y is its class value represented by a particular attribute. The training data is divided into two sets: the training set and the test set. The system operates in two phases: the training phase consists in designing the model while the second step evaluates the trained model.

For instance, in the data table 1, objects or patterns 1 to 6 can be associated to the positive class (+), while patterns 7 to 10 can be associated to the negative class (-).

The model evaluation (or test) consists of calculating its accuracy rate as the ratio between the number of well classified patterns and the total number of patterns. There are many techniques to determine accuracy rate among which:

1. K-fold cross validation. The training data is divided into k disjoint subsets and the model is trained and tested k times. At each iteration i, the i^{th} subset is used to test the model built and trained using the other $k-1$ subsets (all other subsets except i^{th} subset). The accuracy rate is calculated as the average of the different accuracy rates obtained at each iteration. Empirically it is advised to take $k = 10$.
2. Leave-one-out is a variant of k-fold cross validation where k is to the number of patterns on the training set.
3. Holdout. The training set is randomly separated into two disjoint subsets. One of these subsets is used to build and train the model while the other is used for test.

Classification as well as supervised learning are more detailed in [7, 20].

3.2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a mathematical framework that models the world as being composed of objects and attributes, describing an application [40, 16].

Definition 1. *A* ***formal context*** *is a triplet $C = (O, A, I)$ where O is a non empty finite set of objects, A is a non empty finite set of attributes (or items) and I is a binary relation between O and A (formally $I \subseteq O \times A$).*

The formal context (binary) C could be represented as a binary matrix such that $C_{ij} = 1$ if the object represented in row i verifies the attribute represented in column j and 0 if not.

Example 4. *Table 1 is an example of a binary formal context.*
$O = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ is the set of objects or patterns while $A = \{a, b, c, d, e, f\}$ is a set of attributes.

The fundamental intuition of FCA relies on the fact that a concept is represented by an intent and an extent.

Definition 2. *Let f and g be two applications defined as follows:*

- $f : 2^O \longrightarrow 2^A$, *s.t.* $f(O_1) = O_1' = \{a \in A \;/\; \forall o \in O_1\;,\; (o, a) \in I\}$, $O_1 \subseteq O$;
- $g : 2^A \longrightarrow 2^O$, *s.t.* $g(A_1) = A_1' = \{o \in O \;/\; \forall a \in A_1\;,\; (o, a) \in I\}$, $A_1 \subseteq A$;

Table 1 Example of a formal context presented as boolean matrix.

O/A	a	b	c	d	e	f
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	

A pair (O_1, A_1) *is called* ***formal concept*** *iff* $O_1 = A_1'$ *and* $A_1 = O_1'$*.* O_1 *(resp.* A_1*) is the extent (resp. intent) of the concept.*

Example 5. *From table 1,* $(\{1, 2, 5, 6, 10\}, \{a, e\})$ *is a formal concept where* $\{1, 2, 5, 6, 10\}$ *is the extent and* $\{a, e\}$ *is the intent. While* $(\{1, 2, 5\}, \{a, e\})$ *is not a formal concept since* $\{1, 2, 5\}$ *is not the largest set for which each object verifies all attributes of the set* $\{a, e\}$*.*

Definition 3. *Let* L *be the entire set of concepts extracted from the formal context* C *and* $\leq$ *a relation defined as* $(O_1, A_1) \leq (O_2, A_2) \Rightarrow (O_1 \subset O_2)$ *(or* $A_1 \supset A_2$*). The relation* $\leq$ *defines the order relation on* L *[16].*

If $(O_1, A_1) \leq (O_2, A_2)$ *is verified (without intermediate concept) then the concept* (O_1, A_1) *is called the successor of the concept* (O_2, A_2) *and* (O_2, A_2) *the predecessor of* (O_1, A_1)*.*

The **Hasse diagram** is the graphical representation of the relation *successor/predecessor* on the entire set L of concepts.

The fundamental theorem of FCA [40] states that the set of formal concepts of a formal context forms a complete lattice, called a concept lattice. A complete lattice is a partial order in which the greatest lower bound and least upper bound of any subset of the elements in the lattice must exist.

FCA have shown to be useful in data mining for generating concise representations of implicative rules [19] or association rules [2, 18], but also for supervised classification [14]. More details on FCA could be found in [16, 4].

3.3 Constraints

In order to reduce the size of concept lattice and consequently the time complexity, we introduce some constraints regularly used to select concepts during the learning process.

3.3.1 Frequency of Concept

A concept is frequent if it contains at least α (also refered to as 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 prune the lattice and reduce its computational complexity. Minimum support is the minimal number of objects that the intent of a concept must verify to be selected.

3.3.2 Validity of Concept

A concept (X, Y) is **complete** if Y recognizes all positive examples. A concept (X, Y) is **consistent** if Y throws back all counter examples or negative examples (formally, the set of consistent concepts is $\{(X, Y)/Y \cap O^- = \{\}\}$ where $O = O^+ \cup O^-$). Both completeness and consistency constraints are restrictive and can lead to overfitting. Other weak constraints are then introduced:

1. **Validity.** A concept (X, Y) is valid if its description recognizes many 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 if it is valid and its extent contains few counter examples. Formally, the set of quasi-consistent concepts is defined as $\{(X, Y) \,/\, |X^+| \geq \alpha \textit{ and } |X^-| \leq \beta\}$.

3.3.3 Height of a Semi-lattice

The level of a concept c is defined as the minimal number of connections from the supreme concept to c. The height of a semi-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 concept extents (or intents). In fact at each level, concept extents (or intents) 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.

4 Concept Lattice-Based Artificial Neural network

We describe in this section the different steps of our new approach, M-CLANN, as shown by figure 4. The process of finding the architecture of neural networks has three steps: (1) build a joint semi-lattice of formal concepts by applying constraints to select relevant concepts [24, 33]; (2) translate

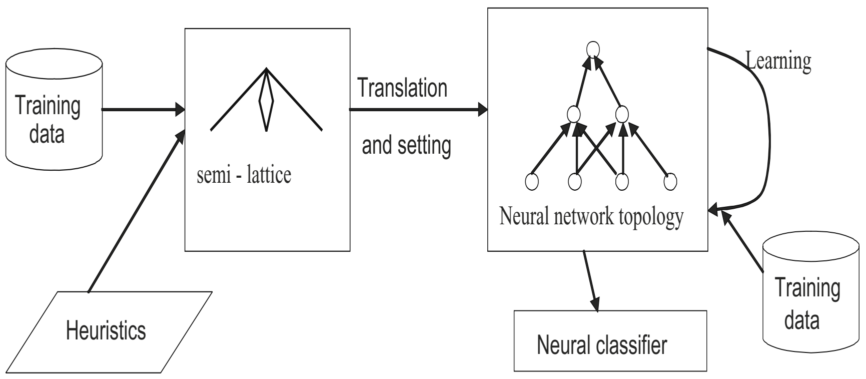

Fig. 4 Neural network topology definition.

the join semi-lattice into a topology of the neural network, and set the initial connections weights; (3) train the neural network.

Variables used in the algorithms defined in this section are : C is a formal context (dataset); L is the semi-lattice built from the training dataset K; c and c' are formal concepts; n is the number of attributes in each training pattern; m is the number of output classes in the training dataset; c a formal concept, element of L; NN is the comprehensive neural network build to classify the data.

4.1 Semi-lattice Construction

There are different algorithms [22] which can be used to generate formal concepts; only a 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 [22] which is suitable here. Concepts included in the lattice are only those which satisfy the defined constraints.

In order to prune the concept lattices, we can use one or multiple constraints to select concepts during this step. The constraints used in M-CLANN are frequency of concept and the height of the semi-lattice. For example it is possible to combine frequency and height constraints, or to use only one of them. The semi-lattice construction process starts by finding the supreme element. The process continues by generating the successors of the concepts that belong to the existing set until there are no concepts which satisfies the specified constraints.

Algorithm 1. Modified Bordat algorithm

```
Require: Binary context $C$
Ensure: concept lattices (concepts extracted from $C$) and the Hasse diagram of
    the order relation between concepts.
 1: Init the list $L$ of the concepts $(O, \{\})$ $(L \leftarrow (O, \{\}))$
 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 from the concept semi-lattice.
```

4.2 Generation of ANN Topology

In the second step, the join semi-lattice is translated into a neural network architecture. Algorithm 2 presents the M-CLANN method to translate the semi-lattice into ANN.

Example 6. *Figure 5 presents an example ANN topology designed with M-CLANN. In this figure, (a) is the semi-lattice while (b) is the corresponding neural network topology.*

Objects used in this algorithm are defined as follows: K is a formal context (dataset); L is the semi-lattice built from the training dataset K; c and c' are formal concepts; n is the number of attributes in each training pattern; m is the number of output classes in the training dataset; c a formal concept, element of L; NN is the comprehensive neural network build to classify the data.

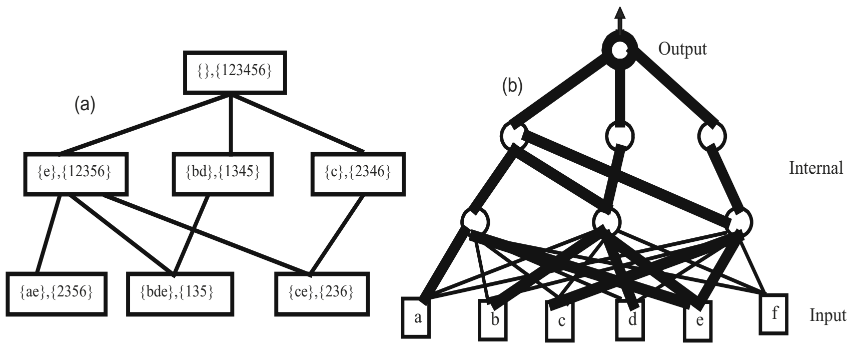

Fig. 5 Example of ANN architecture design using M-CLANN

Algorithm 2. Translation of semi-lattice into ANN topology

```
Require: $L$ a semi-lattice structure built using specified constraints.
Ensure: $NN$ initial topology obtained from the semi-lattice $L$
1: for each concept $c \in L$ do
2:     if the set of predecessor of $c$ is empty mark its successor as "last hidden
       neuron";
3:     Else $c$ becomes neurons and add to $NN$ with the successor and predecessor
       as in $L$; if the set of successor of $c$ is empty then mark $c$ as "first hidden
       neuron".
4:     Endif
5: end for
6: Create a new layer of $n$ neurons and connect each neuron of this layer to the
   neurons marked as "first hidden neuron" in $NN$.
7: Create a new layer of $m$ neurons and connect each neuron of this layer to the
   neurons marked as "last hidden neuron" in $NN$.
8: Initialize connection weights and train them.
```

Threshold is zero for all units and the connection weights are initialized as follows:

- Connection weights between neurons derived directly from the lattice is initialized to 1. This implies that when the neuron is active, all its predecessors are active too.
- Connection weights between the input layer and hidden layer are initialized as follows: 1 if the attribute represented by the input appears in the intention Y of the concept associated to the ANN node and -1 otherwise. This implies that the hidden unit connected to the input unit will be active only if the majority of its input (attributes including in its intent) is 1.

4.3 Training the Generated Topology

The last step of M-CLANN is to train the obtained neural network. This is done using the error backpropagation algorithm [30]. This algorithm searches the appropriate connection weights between the different units by propagating the input signals through the network and backpropagating the error from the output units to the input units. This is done by minimizing the quadratic sum of the error.

5 Experimentations and Results

5.1 Data

To examine the practical aspect of the approach presented above, we run the experiments on the data available on the UCI repository [26]. The

Table 2 Experimental data sets

Dataset	#Train	#Test	#Class	#Nom	#Bin
Balance-scale (Bal)	625	0	3	4	20
Chess	3 196	0	2	36	38
Hayes-roth (Hayes)	132	28	3	5	15
Tic-tac-toe (Tic)	958	0	2	9	26
Spect	80	187	2	22	22
Monks1	124	432	2	6	15
Monks2	169	432	2	6	15
Monks3	122	432	2	6	15
Lymphography (lympho)	148	0	3	18	51
Solar-flare1 (Solar1)	323	0	7	12	40
Solar-flare2 (Solar2)	1066	0	7	12	40
Soybean-backup (Soyb)	307	376	19	35	151
Lenses	24	0	3	4	12

characteristics of this data is shown in the table 2 which contains the name of the dataset, the number of training patterns (#Train), the number of test patterns (#Test), the number of output classes (#Class), the initial number of (nominal) attributes in each pattern (#Nom), the number of binary attributes obtained after binarization (#Bin). Attributes were binarized by the Weka [36] binarization procedure “Filters.NominalToBinary”. The diversity of this data (from 24 to 3196 training patterns; from 2 to 19 output classes) helps in revealing the behavior of each model in many situations. There are no missing values in these datasets.

Two constraints presented above (frequency and height) have been applied in selecting concepts during experimentation. We first separately use each of them and then we combine them.

5.2 Results

Experimental results are obtained from the model trained by error back-propagation [30] and validated by 10-fold cross-validation or holdout [20]. The learning parameters are the following: as activation function, we use the sigmoid ($f(x) = \frac{1}{1+\exp x}$), 500 iterations in the weight modification process and 1 as learning rate.

Table 3 presents the accuracy rate (percentage) obtained with data in table 2. In table 3, the symbol ”-” indicates that no formal concept satisfies the constraints and the process was stopped. The symbol ”x” indicates that the classifier CLANN was not applied for those multiclass problem.

In this table, MCL1 is M-CLANN built from a semi-lattice with one level while MCL30 and MCL20 are M-CLANN built using respectively 30 and 20

Table 3 Accuracy rates of MCLANN classifier with some varied input parameters.

Dataset	CLANN	MCL1	MCL2	MCL30	MCL20	MC1-30	MC1-20
Bal	x	99,76	96,23	-	99,89	-	99,89
Chess	93,60	99,87	91,70	93,60	93,78	99,87	99,87
Hayes	x	75,72	76.85	78,58	85,72	78,57	85,71
Tic	94,45	89,64	90.21	99,67	99,86	99,32	100
Spect	93,90	72,74	72,56	92,56	96,73	73,66	77,57
Monks1	82,70	91,67	95,56	91,17	91,17	91,67	91,71
Monks2	78,91	100	98,65	100	100	100	99,67
Monks3	83,61	93,51	100	91,17	93,52	92,59	93,52
Lympho	x	80,78	90,24	84,67	88,91	85,71	92,56
Solar1	x	79,42	78,87	78,67	69,58	71,10	71,10
Solar2	x	75,00	72,62	76,71	70,91	75,34	78,95
Soyb	x	81,33	79,01	89,34	86,95	83,11	84,04
Lenses	x	98,67	90,00	100	99,87	98,67	99,87
Average	86,05	86,62	86,97	89,67	90,53	87,57	90,37

percent as frequency threshold. MC1-30 (respectively MC1-20) is M-CLANN built with a combination of semi-lattice height equals to 1 and 30% (resp. 20%) as frequency threshold. CLANN column represents the precision rate obtained using the original version of CLANN (with lattice height threshold equals to one).

With high minimum support values, sometimes the semi-lattice does not contain sufficient concepts to better classify the data. For instance, with the minimum support value set to 35%, the semi-lattice built from Balance-scale is empty. The best results (accuracy rate) of M-CLANN are obtained with the α value equal to 20% (MCL20). These results are comparable to those of other classifiers as shown in table 4 using some standard machine learning classifiers or some constructive multilayer perceptrons. In table 4, the symbol "x" indicates that the classifier does not converge. The standard classifiers are taken from the WEKA platform [36] and are MLP (a multilayer perceptron classifier), C4.5 (a decision tree based classifier), IB1 (a case based learning classifier model). The constructive multilayer perceptrons are the original versions of author's implementation of Mtiling, Mtower, Mupstart and Distal.

M-CLANN was not compared with KBANN because we have no prior knowledge about this data. The goal of this comparison is to see the behavior (on the supervised classification problems) of M-CLANN regarding those of other standard learning models.

Using different parameters settings, M-CLANN outperfomed standard machine learning classifiers in terms of accuracy on the experimental datasets. MLP is better than C4.5 and IB1.

Table 4 Accuracy rate of other classifiers.

Dataset	MCL20	MLP	C4.5	IB1	MTiling	MUpstart	MTower	Distal
Bal	99,89	98,40	77,92	66,72	94,27	100	95,16	96,77
Chess	93,78	99,30	98,30	89,90	96,24	97,18	96,87	89,74
Hayes	85,72	82,15	89,28	75,00	89,29	90,01	78,57	54,32
Tic	99,86	96,86	93,21	81,63	75,52	73,03	64,21	61,23
Spect	96,73	65,77	66,70	66,31	89,60	83,29	71,40	83,90
Monks1	91,17	100	100	89,35	81,71	77,21	78,01	90,23
Monks2	100	100	70,37	66,89	85,42	82,43	77,87	89,10
Monks3	93,52	93,52	100	81,63	100	89,42	91,21	86,46
Lympho	88,91	81,76	74,32	80,41	85,71	78,57	78,57	86,45
Solar1	69,58	72,79	74,30	68,39	100	100	98,89	x
Solar2	70,91	68,11	69,97	66,56	96,88	93,75	96,88	68,23
Soyb	86,95	92,02	88,83	89,89	83,23	85,45	84,34	x
Lenses	99,87	95,83	91,67	100	99,50	99,00	98,50	99,88
Average	90.59	88,57	84,22	78,67	90,81	87,39	84,43	88.90

Another advantage of M-CLANN over MLP is that each neuron has a semantic as it is associated to each intent of a formal concept. During the experimentations, the running time of MLP and M-CLANN are similar but much higher compared to that of C4.5 and IB1.

MTiling has the best average accuracy rate over the whole dataset, even if this accuracy rate is only slightly greater than that of MCLANN.

6 Discussion

As presented in the previous section, there exists many algorithms which could be used to define the neural network architecture. Each of those algorithms present advantages but they also have issues:

1. **Input data.** Many algorithms could not process other data than numeric. Apart from Distal where the authors have defined the distance between symbolic data, all others only treat numeric data. In addition of the training data, using KBANN method requires a domain theory which is not always available. The choice of the method could hardly be influenced by the input data.
2. **Interpretability of ANN.** It is well known that the ANN is one of the most commonly used methods in classification. As it is seen as 'Black box', it is not used in the domain where result explanations are important. Among the previous methods, only M-CLANN and KBANN present interpretable architectures. So, it could not be advised to use other approaches than M-CLANN and KBANN, while in M-CLANN, each node is associate to one formal concept and each formal concept is formed by

a set of objects (extent) and and a set of attributes (intent) shared by these objects; in KBANN, each node is associated to one variable on the rules set.

3. **Choice of the algorithm's parameters.** One problem with ANN topology design algorithm is the choice of network and training parameters. This problem is avoided in KBANN method where only the maximum number of iterations is needed. In addition to the maximum iterations number (500 as default value), M-CLANN needs to define the constraints value. Other constructive algorithms (except Distal) need to define the maximum layers number, the choice of training algorithm, the maximum iterations number in the training process.

Recently different works showing links between FCA and ANN were reported in the literature. Except from our previous method CLANN, those are different from MCLANN. [12] uses the FCA approach to encode the neural network function, while [31] proposes two ways of directly encoding closure operators on finite sets in a 3 layered feed forward neural network.

7 Conclusion

In this chapter, a new approach of finding the ANN topology is presented. This method is based on concept lattices and is able to define an interpretable ANN topology without any prior domain knowledge. This proposal extends our previous method CLANN in order to treat multi-class supervised classification problems.

Some empirical classification results presented above show its efficiency compared to standard machine learning classification and other constructive multilayer perceptrons.

The extension of this approach will consist of extracting rules from the network after training and the treatment of multivalued context. A more theoretical study of [29] discusses the fact some neural networks compute and others don't. We will explore the link with our proposal.

References

1. Andrews, R., Diederich, J., Tickle, A.: Surevy and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 8(6), 373–389 (1995)
2. Bastide, Y., Pasquier, N., Taouil, R., Stumme, G., Lakhal, L.: Mining minimal non-redundant association rules using frequent closed itemsets. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 972–986. Springer, Heidelberg (2000)

3. Bertini Jr., J.R., do Carmo Nicoletti, M.: MBabCoNN - A Multiclass Version of a Constructive Neural Network Algorithm Based on Linear Separability and Convex Hull. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 723–733. Springer, Heidelberg (2008)
4. Carpineto, C., Romano, G.: Concept Data Analysis: Theory and Applications. John Wiley and Sons, Chichester (2004)
5. Cibas, T., Fogelman, F., Gallinari, P., Raudys, S.: Variable Selection with Optimal Cell Damage. In: International conference on Artificial Neural Network (ICANN 1994), Part I, pp. 727–730 (1994)
6. Cornuéjols, A., Miclet, L.: Apprentissage Artificiel: Concepts et algorithmes, Eyrolles (2002)
7. Mitchell., T.M.: Machine Learning. McGraw-Hill, New York (1997)
8. Curran, D., O'Riordan, C.: Applying Evolutionary Computation to designing Neural Networks: A study of the State of the art department of Information Technology, Technical report, NUI Galway (2002)
9. Darbari, A.: Rule extraction from trained ANN: A Survey, Technical report, Institute of Artificial Intelligence, Dept. of Computer Science, TU Dresden, Germany (2001)
10. Dreyfus, G., Samuelides, M., Martinez, J.M., Gordon, M., Badran, F., Thiria, S., Hérault, L.: Réseaux de Neurones: Méthodologie et applications. Eyrolles (2002)
11. Duch, W., Setiono, R., Zurada, J.M.: Computational intelligence methods for understanding of data. Proceedings of the IEEE 92(5), 771–805 (2004)
12. Endres, D., Fldiák, P.: An application of formal concept analysis to Neural Decoding. In: Belohlavek, E.R., Kuznetsov, S.O. (eds.) proceedings of sixth Intl. Conf. on Concept Lattices and Applications (CLA), pp. 181–192 (2008)
13. Frean, M.: The Upstart algorithm: A method for constructing and training feed forward neural networks. Neural computation (4), 198–209 (1992)
14. Fu, H., Fu, H., Njiwoua, P., Nguifo, E.M.: A comparative study of FCA-based supervised classification algorithms. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 313–320. Springer, Heidelberg (2004)
15. Gallant, S.: Perceptron based learning algorithms. IEEE Transactions Neural Networks 1, 179–191 (1990)
16. Ganter, B., Wille, R.: Formal Concepts Analysis: Mathematical foundations. Springer, Heidelberg (1999)
17. Garcez d'Avila, A.S., Broda, K., Gabbay, D.M.: Symbolic knowledge extraction from trained neural networks: a sound approach. Artificial Intelligence 125, 155–207 (2001)
18. Gasmi, G., Ben Yahia, S., Mephu Nguifo, E., Slimani, Y.: A new informative generic base of association rules. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 81–90. Springer, Heidelberg (2005)
19. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de donnes binaires. Mathmatiques et sciences sociales 95, 5–18 (1986)
20. Han, J., Kamber, M.: Datamining: Concepts and Techniques. Morgan Kauffman Publishers, San Francisco (2001)
21. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the theory of neural computation. Lecture Notes, Santa Fe Institute. Addison Wesley Publishing, Reading (1991)
22. Kuznetsov, S., Obiedkov, S.: Comparing Performance of Algorithms for Generating Concept Lattices. JETAI 14(2/3), 189–216 (2002)

23. Le Cun, Y., Denker, J.S., Solla, S.A.: Optimal Brain Damage. In: Advances in Neural Information Processing Systems, vol. 2, pp. 598–605. Morgan Kaufmann Publishers, San Francisco (1990)
24. Mephu Nguifo, E.: Une nouvelle approche base sur le treillis de Galois pour l'apprentissage de concepts. Mathématiques, Informatique, Sciences Humaines 134, 19–38 (1994)
25. Mephu Nguifo, E., Tsopzé, N., Tindo, G.: M-CLANN: Multi-class concept lattice-based artificial neural network for supervised classification. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 812–821. Springer, Heidelberg (2008)
26. Newmann, D.J., Hettich, S., Blake, C.L., Merz, C.J.: (UCI)Repository of machine learning databases, Dept. Inform. Comput. Sci. Univ. California, Irvine, CA (1998), `http://www.ics.uci.edu/AI/ML/MLDBRepository.html`
27. Parekh, R., Yang, J., Honavar, V.: Constructive Neural Networks Learning Algorithms for Multi-Category Classification. Department of Computer Science Lowa State University Tech. Report ISU CS TR 95-15 (1995)
28. Parekh, R., Yang, J., Honavar, V.: Constructive Neural-Network Learning Algorithms for Pattern Classification. IEEE Transactions on neural networks 11(2), 436–451 (2000)
29. Piccinini, G.: Some neural networks compute, others dont. Neural Network 21 (special issue), 311–321 (2008)
30. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by backpropagating errors. Nature (323), 318–362 (1986)
31. Rudolph, S.: Using FCA for Encoding Closure Operators into Neural Networks. In: Priss, U., Polovina, S., Hill, R. (eds.) ICCS 2007. LNCS (LNAI), vol. 4604, pp. 321–332. Springer, Heidelberg (2007)
32. Shavlik, W.J., Towell, G.G.: Kbann: Knowledge based articial neural networks. Artificial Intelligence (70), 119–165 (1994)
33. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing Iceberg concept lattices with TITANIC. Journal on Knowledge and Data Engineering (KDE) 2(42), 189–222 (2002)
34. Subirats, J.L., Franco, L., Molina Conde, I., Jerez, J.M.: Active learning using a constructive neural network algorithm. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008,, Part II. LNCS, vol. 5164, pp. 803–811. Springer, Heidelberg (2008)
35. Tsopze, N., Mephu Nguifo, E., Tindo, G.: CLANN: Concept-Lattices-based Artificial Neural Networks. In: Diatta, J., Eklund, P., Liquire, M. (eds.) Proceedings of fifth Intl. Conf. on Concept Lattices and Applications (CLA 2007), Montpellier, France, October 24-26, 2007, pp. 157–168 (2007)
36. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
37. Yacoub, M., Bennani, Y.: Architecture Optimisation in Feedforward Connectionist Models. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327. Springer, Heidelberg (1997)
38. Yang, J., Parekh, R., Honavar, V.: Distal: An Inter-pattern Distance-based Constructive Learning Algorithm: Intell. Data Anal. 3, 55–73 (1999)
39. Werbos, P.J.: Why neural networks? In: Fiesler, E., Beale, R. (eds.) Handbook of Neural Computation, pp. A2.1:1–A2.3:6. IOP Pub., Oxford University Press, Oxford (1997)
40. Wille, R.: Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. In: Rival, I. (ed.) Ordered Sets, pp. 445–470 (1982)