M-CLANN: Multi-class Concept Lattice-Based Artificial Neural Network for Supervised Classification

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

[1] Université de Lille-Nord, Artois
CRIL-CNRS UMR 8188, IUT de Lens
SP 16, Rue de l'Université 62307 Lens Cedex, France
[2] Université de Yaoundé I, Faculté des Sciences,
Département d'Informatique BP 812
Yaoundé - Cameroun

Abstract. Multi-layer neural networks have been successfully applied in a wide range of supervised and unsupervised learning applications. However defining its architecture is a difficult task, and might make their usage very complicated. To solve this problem, a rule-based model, KBANN, was previously introduced making use of an apriori knowledge to build the network architecture. Neithertheless this apriori knowledge is not always available when dealing with real world applications. Other methods presented in the literature propose to find directly the neural network architecture by incrementally adding new hidden neurons (or layers) to the existing network, network which initially has no hidden layer. Recently, a novel neural network approach CLANN based on concept lattices was proposed with the advantage to be suitable for finding the architecture of the neural network when the apriori knowledge is not available. However CLANN is limited to application with only two-class data, which is not often the case in practice. In this paper we propose a novel approach M-CLANN in order to treat multi-class data. Carried out experiments showed the soundness and efficiency of our approach on different UCI datasets compared to standard machine learning systems. It also comes out that M-CLANN model considerably improved CLANN model when dealing with two-class data.

1 Introduction

An artificial neural network (ANN) is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. ANN are useful especially when there is no a priori knowledge about the analyzed data. They offer a powerful and distributed computing architecture, with significant learning abilities and they are able to represent highly nonlinear and multivariable relationships. ANN have been successfully applied to problems in pattern classification, function approximation, optimization, pattern matching and associative memories [13]. Different types of ANN 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. MLP networks trained using the back-propagation learning algorithm are limited to searching for a suitable set of weights in an a priori fixed network topology. The selection of a network architecture for a specific problem has to be done carefully. In fact there are no known efficient methods 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 overfill 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 of the number of input neurons and the number of output neurons. In the literature, different automatic approaches have been reported to dynamically build the network topology. These works could be divided into two groups:

1. Search an optimal network to minimize the number of units in the hidden layers [13]. 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 a larger than necessary network and then eliminate weights and neurons that are deemed redundant. An alternative approach consists of using the genetic approach [4], which is computationally expensive. All these (incremental, pruning, genetic) techniques results to neural network that can be seen as black box system, since no semantic is associated to each hidden neuron. Their main limitation is the intelligibility of the resulting network (black-box prediction is not satisfactory [1,5]).
2. Use a set of an a priori knowledge (set of implicative rules) on the problem domain and derive the neural network from this knowledge [15]. The a priori 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 rules set and each connection between two nodes represents one dependence between variables. The obtained neural network, KBANN (Knowledge-Based ANN), is a comprehensible ANN since each node is semantically meaningful, and the ANN decision is not viewed as deriving from a black-box system, but could be easily explain using a subset of rules from the initial apriori knowledge. But this solution is limited while the apriori knowledge is not available as it might be the case in practice.

There are also in the literature many works which help user to optimise [19] or prune networks by pruning some connections [10] or by selecting some variables [3] among the entire set example variables. But these works do not propose an efficient method to build neural network. 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 an a priori knowledge. M-CLANN is an extended version of the CLANN approach [17]. 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 problem, while CLANN is limited to two-class. Second, the derived topology from the semi-lattice is different in both systems. Our proposed approach presents many advantages: (1) the proposed architecture is a multi-layer feed-forward neural network, such that the use of learning algorithm such as back propagation 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 hidden neurons) in the derived ANN has also 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 system) could be extracted from the training data through the model; (5) It better classifies the dataset after training, as observed during experimentations on some datasets of UCI repository [11].

The paper is organized as follows: Section 2 gives preliminary definitions. Section 3 presents the CLANN model. Section 4 is dedicated to our novel approach M-CLANN. The empirical evidences about the utility of the proposed approach are presented in Section 5.

2 Definitions – Preliminaries

A **formal context** is a triplet $K = (O, A, I)$ where O is a not empty finite set of objects, A a not empty finite set of attributes (or items) and I is a binary relation between elements of O and elements of A (formally $I \subseteq O \times A$).

Let f and g be two applications defined as follows: $f:\ 2^O \longrightarrow 2^A$, such that $f(O_1) = O_1' = \{a \in A \ / \ \forall o \in O_1\ ,\ (o,a) \in I\}$, $O_1 \subseteq O$ and $g:\ 2^A \longrightarrow 2^O$, such that $g(A_1) = A_1' = \{o \in O \ / \ \forall a \in A_1\ ,\ (o,a) \in I\}$, $A_1 \subseteq A$; 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 extension (resp. intension) of the concept.

Let L be the entire set of concepts extracted from the context K 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 [7]. If $(O_1, A_1) \leq (O_2, A_2)$ is verified (without intermediated 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 of concepts L. More details on FCA could be found in [7]. Different works in the literature (e.g. [2]) have shown how to derive implication rules or associations rules [8] from this join semi-lattice, sometimes named iceberg concept lattice [16]; or to use the lattice structure in classification [12]. Neithertheless CLANN only treats problem with binary-class data, which is not often the case in practice.

3 The CLANN Model

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.

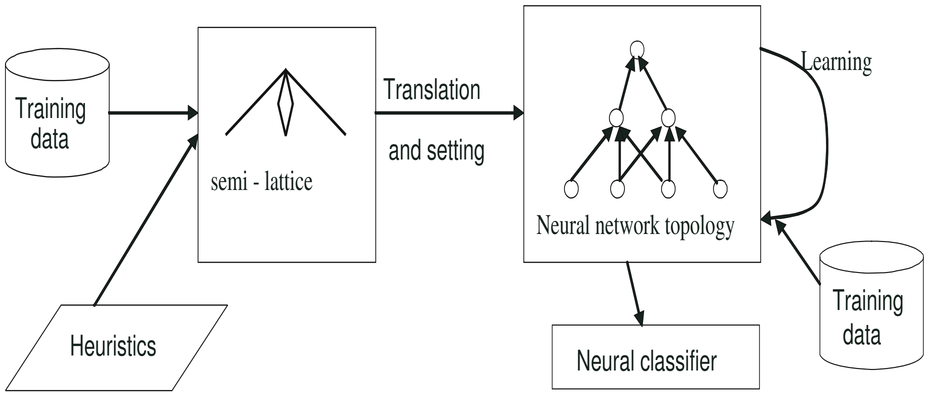

Fig. 1. Neural network topology definition

3.1 Semi-lattice Construction

There are many algorithms [6] 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's algorithm [6] which is suitable here. Concepts included in the lattice are only those which satisfy the defined constraints.

3.2 Constraints

In order to reduce the size of lattice and then the time complexity, we present some 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.

Algorithm 1. Modified Bordat's 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 exist 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 ?? from the concept
    semi-lattice.
```

Validity of concept. Many techniques are used to reduce the size of lattice. The following notions are used in order to 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 constraints, other 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.

4 The M-CLANN Model

As CLANN, M-CLANN defines the topology of a neural network in two phases: in the first phase, a join semi-lattice structure is constructed and in the second one, the join semi-lattice structure is translated into a neural network architecture.

M-CLANN builds the join semi-lattice using a modified version of Bordat's algorithm [6]. The modified algorithm uses monotonic constraints to prune the lattice and then reduce its complexity. The semi-lattice construction process starts by finding the supremum element. The process continues by generating the successors of the concepts that belong to the existing set until there are no concept which satisfies the specified constraints. Algorithm 2 presents the M-CLANN method to translate the semi-lattice into ANN.

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.

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.
```

Among the constraints used in CLANN, the validity of concept is not use in M-CLANN since M-CLANN does not consider the class of each object during the semi-lattice building process. The constraints used in M-CLANN are frequency of concept and the height of the semi-lattice.

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 is 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 intention) is 1.

5 Experimentations

To examine the practical aspect of the approach presented above, we run the experiments on the data available on UCI repository [11]. The characteristics of these data are collected in table 1 which contains the name of the dataset (dataset label), 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). Those attributes were binarized by the Weka [18] binarization procedure "Filters.NominalToBinary". The diversity of these data (from 24 to 3196 training patterns; from 2 to 19 output classes) helps in revealing the behaviour of each model in many situations. There is no missing values in these datasets.

Table 1. 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
Monsk1	124	432	2	6	15
Monsk2	169	432	2	6	15
Monsk3	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

The two constraints presented above have been applied in selecting concepts during experimentation. In the first step we separately use each constraint and we combine them in the second step during the join semi-lattice construction process.

The experiment results are obtained from the model trained by backpropagation algorithm [14] and validated by 10-cross validation or holdout [9]. The parameters of the learning algorithm 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. In the result table, the symbol "-" indicates that no concept satisfies the constraint and the process has not converged. Table 2 presents the accuracy rate (percentage) obtained with data in table 1 and those obtained using other classifiers. These classifiers are MLP (a WEKA implementation of the multilayer perceptron classifier), C4.5 (a decision tree based classifier), IB1 (a case based learning classifier model). The result presented is the accuracy rate. In this table, MCL1 is M-CLANN built using only one level of the semi-lattice; MCL30 and MCL20 are M-CLANN built using respectively 30 and 20 percent as frequency of the concepts. CLANN column represents the accuracy rate obtained using the original version of CLANN (with a constraint of one level; x indicates that it is not possible to use CLANN since it is a multi-class dataset). Finally MC1-30 and MC1-20 are respectively M-CLANN built with a combination of one level lattice and 30% minimum support and M-CLANN built with a combination of one level semi-lattice and 20% minimum support as constraints. The best results (accuracy rate) of M-CLANN are obtained with the α value equal to 20% (MCL20). With high minimum support values, the semi-lattice does sometimes 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.

M-CLANN was not compared with KBANN because we haven't an apriori knowledge about these data. The goal of this comparison is to see the behaviour (on the supervised classification problems) of M-CLANN regarding those of other learning models. From the average of accuracy rate presented in the table 2, MCL20 is the best classifier in average. Using different parameters settings, M-CLANN is still better than other classifiers. MLP of Weka platform is the best one compared to C4.5 and IB1. Another advantage of M-CLANN over MLP is that each neuron is meaningful and this can be used to explain its decision. During the experimentations, the running times of MLP and M-CLANN are much more greater than that of C4.5 and IB1. The difference between the running times of M-CLANN and those of MLP is not significative.

Table 2. Accuracy rate of used classifiers with data of table 1

Dataset	CLANN	MCL1	MCL30	MCL20	MC1-30	MC1-20	MLP	C4.5	IB1
Bal	x	99,76	-	99,89	-	99,89	98,40	77,92	66,72
Chess	93,60	99,87	93,60	93,78	99,87	99,87	99,30	98,30	89,9
Hayes	x	75,72	78,58	85,72	78,57	85,71	82,15	89,28	75,00
Tic	94,45	89,64	99,67	99,86	99,32	100	96,86	93,21	81,63
Spect	93,90	72,74	92,56	96,73	73,66	77,57	65,77	66,70	66,31
Monsk1	82,70	91,67	91,17	91,17	91,67	91,71	100	100	89,35
Monsk2	78,91	100	100	100	100	99,67	100	70,37	66,89
Monsk3	83,61	93,51	91,17	93,52	92,59	93,52	93,52	100	81,63
Lympho	x	80,78	84,67	88,91	85,71	92,56	81,76	74,32	80,41
Solar1	x	79,42	78,67	69,58	71,10	71,10	72,79	74,30	68,39
Solar2	x	75,00	76,71	70,91	75,34	78,95	68,11	69,97	66,56
Soyb	x	81,33	89,34	86,95	83,11	84,04	92,02	88,83	89,89
Lenses	x	98,67	100	99,87	98,67	99,87	95,83	91,67	100
Average	86,05	86,62	89,67	90,53	87,57	90,37	88,57	84,22	78,67

6 Conclusion

This paper shows how concept lattices can help to build a comprehensible ANN for a given task, by providing clear significance to each neuron. In this work, we use some monotonic constraints to reduce the size of the lattice and also the training time. Particularly for the classification tasks, looking at the experiment results, our model is better in average than many other classifiers in different test cases. M-CLANN model also uses binary data set. Our ongoing research firstly consists of extending M-CLANN to nominal and numeric data. Many attempts exist in the literature to build concept lattices for such data sets. We will also compare extracted rules from our model to those of other models as KBANN, or decision trees.

References

1. Andrews, R., Diederich, J., Tickle, A.B.: Survey 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: CL 2000. LNCS (LNAI), vol. 1861, pp. 972–986. Springer, Heidelberg (2000)
3. Cibas, T., Fogelman, F., Gallinari, P., Raudys, S.: Variable Selection with Optimal Cell Damage. In: International conference on Artificial Neural Network (ICANN 1994), vol. 1, pp. 727–730 (1994)
4. Curran, D., O'Riordan, C.: Applying Evolutionary Computation to designing Neural Networks: A study of the State of the art. Technical report NUIG-IT-111002, department of Information Technology, NUI of Galway (2002)
5. Duch, W., Setiono, R., Zurada, J.M.: Computational intelligence methods for understanding of data. Proceedings of the IEEE 92(5), 771–805 (2004)
6. Kuznetsov, S., Obiedkov, S.: Comparing Performance of Algorithms for Generating Concept Lattices. JETAI 14(2/3), 189–216 (2002)
7. Ganter, B., Wille, R.: Formal Concepts Analysis: Mathematical foundations. Springer, Heidelberg (1999)
8. Gasmi, G., Ben Yahia, S., Mephu Nguifo, E., Slimani, Y.: A new informative generic base of association rules. In: Advances in Knowledge Discovery and Data Mining (PAKDD), vol. 3518, pp. 81–90 (2005)
9. Han, J., Hamber, M.: Data Mining: Concepts and Techniques. Morgan Kauffman Publishers, San Francisco (2001)
10. Le Cun, Y., Denker, J.S., Solla, S.A.: Optimal Brain Damage. In: Advances in Neural Information Processing Systems 2, pp. 598–605. Morgan Kaufmann Publishers, San Francisco (1990)
11. 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`
12. Fu, H., Fu, H., Njiwoua, P., Mephu Nguifo, E.: Comparative study of FCA-based supervised classification algorithms. In: The proc of 2nd ICFCA, Sydney, pp. 313–320 (2004)

13. Parekh, R., Yang, J., Honavar, V.: Constructive Neural-Network Learning Algorithms for Pattern Classification. IEEE Transactions on Neural Networks 11, 436–451 (2000)
14. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 318–362 (1986)
15. Shavlik, J.W., Towell, G.G.: Kbann: Knowledge based articial neural networks. Artificial Intelligence 70, 119–165 (1994)
16. 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)
17. Tsopze, N., Mephu Nguifo, E., Tindo, G.: CLANN: Concept-Lattices-based Artificial Neural Networks. In: Diatta, J., Eklund, P., Liquiére, M. (eds.) Proceedings of fifth Intl. Conf. on Concept Lattices and Applications (CLA 2007), Montpellier, France, October 24-26, 2007, pp. 157–168 (2007)
18. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)
19. Yacoub, M., Bennani, Y.: Architecture Optimisation in Feedforward Connectionist Models. In: 8th International Conference on Artificial Neural Networks (ICANN 1998), Skövde, Sweden (1998)