Computing the Concept Lattice using Dendritical Neural Networks

David Ernesto Caro-Contreras, Andres Mendez-Vazquez

Centro de Investigación y Estudios Avanzados del Politécnico Nacional
Av. del Bosque 1145, colonia el Bajío, Zapopan , 45019, Jalisco, México.
dcaro@gdl.cinvestav.mx,amendez@gdl.cinvestav.mx

Abstract. Formal Concept Analysis (FCA) is a new and rich emerging discipline, and it provides efficient techniques and methods for efficient data analysis under the idea of "attributes". The main tool used in this area is the Concept Lattice also named Galois Lattice or Maximal Rectangle Lattice. A naive way to generate the Concept Lattice is by enumeration of each cluster of attributes. Unfortunately the numbers of clusters under the inclusion attribute relation has an exponential upper bound. In this work, we present a novel algorithm, PIRA (PIRA is a Recursive Acronym), for computing Concept Lattices in an elegant way. This task is achieved through the relation between maximal height and width rectangles, and maximal anti-chains. Then, using a dendritical neural network is possible to identify the maximal anti-chains in the lattice structure by means of maximal height or width rectangles.

Keywords: formal concept analysis, lattice generation, neural networks, dendrites, maximal rectangles.

1 Introduction

Formal Concept Analysis (FCA) refers to a mathematized and formal human-centered way to analyze data. It can be described as an ontology-based Lattice Theory. This theory describes ways of visualizing patterns, generate implications, and representing generalizations and dependencies [1]. The theory makes use of a binary relation between objects and attributes, which plays a fundamental role in the understanding of human model conceptualization. FCA formalizes all these ideas, and gives a mathematical framework to work on them [2].

The Concept Lattice is mainly used to represent and describe hierarchies between data clusters (formal concepts or classes), which are inherent in the perceived information. A formal concept can be regarded also as a data cluster, in which certain attributes are all shared by a set of objects. The Concept Lattice is the main subject of the theory of FCA, and it was first introduced in the early eighties by Rudolf Wille [2, 3]. Over time, FCA has become a powerful theory for many interesting applications. Examples of these are in the data analysis of Frequent Closed Itemsets [4], association rules and Functional/Conditional hierarchies discovery [5, 6, 7]. For all these reasons, concept lattice generation

is an important topic in FCA research [8, 9, 10, 11, 12, 13, 14, 15]. However, the main drawback of concept lattices generation is the exponential size of the concept lattice and its generation complexity [16].

In this work, we present a Morphological Neural Network based method to generate the Concept Lattice. This batch method is capable of generating the Hasse diagram, and it is a bottom-up generation algorithm. With that in mind, this work is organized as follows. Section 2 is an elementary description of FCA Theory. Next, section 3 is a short introduction on how the Single Lattice Layer Perceptron (SLLP) works. In section 4, a description on how to compute the Maximal Rectangles is given. In this section, we also define a link between SLLP and the design of a classifier for maximal anti-chains search. Section 5 shows a comparison between PIRA and other known algorithms.

2 Basic Definitions

2.1 Formal Concept Analysis (FCA)

First, it is necessary to define the concept of a formal context [3].

Definition 1. *A binary formal context $\mathbb{K}$ is a triple (G, M, I). In this mathematical structure, G and M are two finite sets, called objects and attributes respectively, and $I \subseteq G \times M$ is a binary relation over G and M, named the incidence of $\mathbb{K}$.*

In order to define formal concepts of the formal context (G,M,I), it is necessary to define two derivation operators, named Galois connectors.

Definition 2. *For an arbitrary subsets $A \subseteq G$ and $B \subseteq M$:*

- $A' := \{m \in M \mid (g, m) \in I, \forall g \in A\}$
- $B' := \{g \in G \mid (g, m) \in I, \forall m \in B\}$

These two derivation operators satisfy the following three conditions over arbitrary subsets $A_1, A_2 \subseteq G$ and $B_1, B_2 \subseteq M$:

1. $A_1 \subseteq A_2$ then $A_2' \subseteq A_1'$ dually $B_1 \subseteq B_2$ then $B_2' \subseteq B_1'$
2. $A_1 \subseteq A_1''$ and $A_1' = A_1'''$ dually $B_1 \subseteq B_1''$ and $B_1' = B_1'''$
3. $A_1 \subseteq B_1' \iff B_1 \subseteq A_1'$

Next, the definition of a formal concept idea, which represents the building unit of FCA.

Definition 3. *Let $\mathbb{K}$ be a formal context, $\mathbb{K} := (G, M, I)$. A formal concept C of $\mathbb{K}$ is defined as a pair $C = (A, B)$, $A \subseteq G$ and $B \subseteq M$, where the following conditions are satisfied, $A = B'$ and $A' = B$, where A is named the extent and B is named the intent of the formal concept (A, B).*

Next, we will show some useful notions from Lattice Theory [17, 18] to understand the algebraic structure generated by those derivation operators and the formal concept idea.

Definition 4. *A Partially Ordered Set (Poset) is a set X in which there is a binary relation between elements of X, $\leq$, with the following properties:*

1. $\forall x,\ x \leq x$ (Reflexive).
2. if $x \leq y$ and $y \leq x$, then $x = y$ (Antisymmetric).
3. if $x \leq y$ and $y \leq z$, then $x \leq z$ (Transitive).

Formal Concepts can be partially ordered by inclusion. And for any pair C_i, C_j have a unique greatest lower bound and a unique least upper bound. Then, the set of all formal concept in $\mathbb{K}$, ordered by inclusion, is known as a Concept Lattice. Next definition links some definitions with the Formal Concept notion.

Definition 5. *(Rectangles in $\mathbb{K}$). Let $A \in G$ and $B \in M$, a rectangle in $\mathbb{K}$ is a pair (A, B) such that $A \times B \subseteq I$.*

Given the set of Rectangles in $\mathbb{K}$, a special kind of rectangles are defined as follows:

Definition 6. *(Maximal Rectangles). A rectangle (A_1, B_1) is maximal if and only if there does not exist another valid rectangle (A_2, B_2) in $\mathbb{K}$ such that $A_1 \subseteq A_2$ or $B_1 \subseteq B_2$.*

From here, we have the formal concept as a maximal rectangle.

Theorem 1. *(A, B) is a formal concept of $\mathbb{K}$ if and only if (A, B) is a maximal rectangle in $\mathbb{K}$.*

Now, the following definitions are going to be useful for the proposed bottom-up approach of FCA generation.

Definition 7. *Let $\mathbb{K} := (G, M, I)$ and $A \subseteq G$. A is said to be an object derivative anti-chain set if and only if $A_1' \nsubseteq A_2'$ and $A_2' \nsubseteq A_1'$ for any two distinct $A_1, A_2 \in A$.* Dually, for $B \subset M$ nd $B_1' \nsubseteq B_2'$ for any two distinct B_1', B_2' in B.

We will denote as D a derivative anti-chain and as $\mathfrak{A}(\mathbb{K})$ the set of all antichains in $\mathbb{K}$. All sets in which the super/subconcept order is not satisfied for any distinct elements of the set is called anti-chain set.

Definition 8. *D^+ is a maximal derivative anti-chain iff there does not exist other $D \in \mathfrak{A}(\mathbb{K})$ such that $D \supset D^+$. There exists a maximal derivative anti-chain for objects and one for attributes.*

Finally, we use a simple upward closed set definition.

Definition 9. *Upward closed set. Let $(L, \leq)$ be a poset P. A set $S \subseteq L$ is said to be upward closed if $\forall x, y \in S$ with $y \geq x$ implies that $y \in S$.*

Using these definitions of anti-chains, maximal rectangles and upward closed set, it is possible to device a bottom-up approach by means of dendritic neuronal networks.

3 Lattice-Based Neural Networks (LBNN).

Artificial Neural Networks (ANN) are models of learning and automatic processing inspired by the nervous system. The features of ANN make them quite suitable for applications where there is no prior pattern that can be identified, and there is only a basic set of input examples (previously classified or not). They are also highly robust to noise, failure of elements in the ANN, and, finally, they are parallelizable. As we said early, our work is related with LBNN, which is also considered an Artificial Neural Network, and are inspired in recent advances in the neurobiology and biophysics of neural computation **(author?)** [19, 20].

The theory of LBNN is actively used in classification [21, 20, 22], clustering [23], associative memories [24, 25], fuzzy logic [26], among others[27, 28]. Basically, in the LBNN model, an input layer receives external data, and subsequent layers perform the necessary functions to generate the desired outputs. Single Lattice Layer Perceptrons (SLLP), also named Dendritic Single Layer Perceptron, are basically a classifier in which there exists a set of input neurons, a set of output neurons, and a set of dendrites growing from the output neurons. Those dendrites are connected with the input set by some axons from those input neurons. A training set configures those outputs based on the maxima $\bigvee$ and minima $\bigwedge$ operations derived from the morphological algebra $(\mathbb{R}, +, \bigvee, \bigwedge)$. FCA and Mathematical Morphology common algebraic framework: Erosion, dilatation, morphological operators, valuations, an many other functions in concept lattices have been previously studied [29].

In SLLP, a set of n input neurons $N_1, \ldots, N_n$ accepts input $x = (x_1, ..., x_n) \in \mathbb{R}^n$. An input neuron provides information through axonal terminals to the dendritic trees of output neurons. A set of O output neurons is represented by $O_1, ..., O_m$. The weight of an axonal terminal of neuron N_i connected to the k_{th} dendrite of the O_j output neuron is denoted by w^{ℓ}_{ijk}, in which, the superscript $\ell \in \{0, 1\}$ represents an excitatory $\ell = 1$ or a inhibitory $\ell = 0$ input to the dendrite. The k_{th} dendrite of O_j will respond to the total value received from the N input neurons set, and it will accept or reject the given input. Dendrite computation is the most important operation in LBNN. The following equation $\tau^j_k(x)$, from SLLP, corresponds to the computation of the k_{th} dendrite of the j_{st} output neuron[21].

$$\tau^j_k(x) = p_{jk} \bigwedge_{i \in I(k)} \bigwedge_{\ell \in L} (-1)^{1-\ell}(x_i + w^{\ell}_{ijk})$$

Where x is the input value of neurons $N_1, ..., N_n$ and x_i is the the value of the input neuron N_i. $I(k) \subseteq 1, ..., n$ represents the set of all input neurons with synaptic connection on the k_{th} dendrite of O_j. The number of terminal axonal fibers on N_i that synapse on a dendrite of O_j is at most two, since $\ell(i) \subseteq \{0, 1\}$. Finally, the last involved element is $p_{jk} \in \{-1, 1\}$ and it denotes the excitatory ($p_{jk} = 1$) or inhibitory ($p_{jk} = -1$) response of the k_{th} dendrite of O_j to the received input. All the values $\tau^j_k(x)$ are passed to the neuron cell body. The

value computation of O_j is a function that computes all its dendrites values. The total value received by O_j is given by[21]:

$$\tau^j(x) = p_j * \bigvee_{k=1}^{K_j} \tau_k^j(x)$$

In this SLLP model, K_j is the set of all dendrites of O_j, $p_j = \pm 1$ represents the response of the cell body to the received input vector. At this point, we know that $p_j = 1$ means that the input is accepted and $p_j = -1$ means that the cell body rejects the received input vector. The last statement related with O_j correspond to an activation function f, namely $y_j = f\left[\tau^j(x)\right]$.

$$f\left[\tau^j(x)\right] = \begin{cases} 1 \iff \tau(x) \geq 0 \\ 0 \iff \tau(x) < 0 \end{cases} \tag{1}$$

As, we mention early, dendrites configurations is computed using a training set. For this, they use merge and elimination methods [17].

4 PIRA Algorithm.

In PIRA-LBNN model for finding upward-closed set elements, a set of binary patterns are represented by G'. Thus, the binary representation of a formal context is itself a set of patterns, in which the derivative of each object is an element $x \in G'$. Then, we can define each element $x = (x_1, ..., x_n) \in G'$ as a binary vector. This allows us to define a simple class classification rule to find maximal rectangles, and in an equivalent way the maximal anti-chains.

In PIRA algorithm we search all the rectangles with maximum width or height from each formal concept founded. There are two ways to achieve our goal. It means that $\ell = 1$ or $\ell = 0$ depending on whether it is calculating, a supremum or an infimum, by using excitatory or inhibitory dendrites. Thus, p_{jk} is also a constant, $p_{jk} = 1$ or $p_{jk} = -1$ and it denotes the excitatory or inhibitory response of the k_{th} dendrite of M_j to the received input, and another remarkable statement is the fact that we only need to connect zeros as axonal branches. The simplest way to compute the value of the k_{th} dendrite derived from the SLLP equations, is:

$$\tau_k^j(x) = \bigvee_{i \in I(k)} (x_i) \tag{2}$$

This equation, where $\tau_k^j(x)$ is the value of the computation of the k_{th} dendrite of the j_{th} output neuron given a input x, $I(k) \subseteq \{1, ..., n\}$ is the set of input neurons with terminal fibers that synapse the k_{th} dendrite of our output neuron. We realize that all weights w_{ijk}^{ℓ} are equal to zero, this is, for our upward closed set classifier, we only need to store zero values from the input patterns in the training step. Our goal is that the output of our classification neuron be $x \in C_1$

Algorithm 1 addDendrite

```
INPUT: NeuralNetwork P, Pattern x
OUTPUT: Updated P
  Dendrite k = addNewDendrite(P)
  FOR EACH element in x
    IF getValue(element) = 0
      i = getPosition(element)
      addAxonalBranch(k, i)
    END
  END
END
```

if the input $x \in D^+$. The training step is to find the set D^+. This is achieved by processing elements from higher to lower cardinality,

Specifically, each dendrite k corresponds with one maximal antichain intent to be tested, $I(k)$ is the incidence set of positions where the value of the maximal rectangle is zero for the pattern represented by the k dendrite. We get the state value of M_j computing the minimum value of all it's dendrites. Again, as the SLLP, each $\tau_k^j(x)$ is computed, and it is passed to the cell body of M_j. Then we can get the total value received by our output neuron as follows:

$$\tau^j(x) = \bigwedge \tau_k^j(x) \tag{3}$$

We realize that the activation function is not required since $f[\tau^j(x)] = \tau^j(x)$ where $\tau^j(x) = 1$ if $x \not\leq y$ for all $y \in C_1$ and $\tau^j(x) = 0$ if $x \leq y$ for some $y \in C_1$. As we mentioned above x is a maximal rectangles if and only if $x \not\leq y$ and $y \not\leq x$ for all $y \in C_1$. Using the previous statement we can ensure that half of the work is done, and the second test, $y \not\leq x$, will be performed by processing data in a particular order. In our case, we use cardinality order ensuring that each new computed row is not a superset of the previously computed rows.

As we said before, the idea is to use our LBNN structure to classify maximal rectangles. When we start computing a formal concept, our structure is empty, this means there are not dendrites or axonal connections. So, the first step is to add each element with the maximal cardinality as a pattern to learn. Algorithm 1 shows how an element is added to our LBNN for maximal rectangles learning. First, algorithm 1 receives as parameter the LBNN which is being trained and a binary vector. As we will see below, this binary vector has been proven as an antichain element. Algorithm 1, first grows a new dendrite k_{th} in our output neuron O_j. Every column in x is checked, if that property is not contained by the object x, then an axonal branch grows from the i position of the input neuron set to the new dendrite. This operation is represented by addAxonalBranch calling. We can assure that we will not misclassify formal concepts in the processed context. The algorithm 2 shows how to compute the Concept Lattice by computing Maximal Antichain Sets recursively.

Algorithm 2 Compute Maximal Rectangles

```
INPUT: A Binary Context K:(G,M,I), K–Supremum, K–Infimum,
    Lattice
OUTPUT: Intent HashSet Maximal_Rectangles
STEP 1:
  Init:
  LBNN Upward Structure
  Maximal_Rectangles
STEP 2:
  Sort G by derivative higher to lower Cardinality
STEP 3:
  IF maximal cardinality is equal to K–Supremum intention
      cardinality
    Add Link from K–Supremum to K–Infimum
    RETURN
  Add, as positive dendrites, all elements with maximal
      cardinality in G' to LBNN.
STEP 4:
  Foreach remaining element in G
    IF Maximal_Rectangles does not contains element' AND
        Upward evaluation element' is 1 then:
      add, as dendrite, element to LBNN
      create Formal Concept with:
        element union K–infimum as extent
        element' as intent
      add this new formal concept to:
        Maximal_Rectangles
    OTHER IF Maximal_Rectangles contains element
      add element to previously Formal Concept created.
STEP 5:
  Foreach Rectangle in Maximal_Rectangles
    IF Lattice does not contains Rectangle
      add Rectangle to Lattice
      add a Link from Rectangle to K–Infimum
      Compute Maximal Rectangles with:
        G = G / Rectangle extent
        M = Rectangle intent
        I = Projection
        K–Supremum
        Rectangle as a K–Infimum
        Lattice
    ELSE
      add a Link from previously created Rectangle in
          Global_Maximal_Rectangles to K–Infimum
```

Algorithm 3 Main Function

```
INPUT: BinaryContext(G,M,I)
OUTPUT: Lattice L
Step 1: Get Maximum and Infimum elements.
  FormalConcept max = getMax(G,M,I)
  FormalConcept min = getMin(G,M,I)
  addConcept(L,max), addConcept(L,min)
STEP 2: Get maximal Rectangles From min
  MaxRectangles = Compute Maximal Rectangles with:
    G/min.extent,
    min.intent,I-Proj,
    max,
    min,
    L
```

In algorithm **2**, the first step creates a new dendritic neural network. It is initialized with one output neuron, $n = |intent|$ and $k = 0$, where the number of input neurons is n, and each input neuron represents one attribute element in M. The second step is used to get the G' set ordered by attribute cardinality. Next, in a third step, if the maximal cardinality of the elements in G' is equal to the Supremum intent cardinality, it adds a link between Supremum and Infimum, and stops. Otherwise, it adds all the elements in G' with the maximal cardinality, to the dendritical neural network structure. Those elements are maximal rectangles in the given binary context. The fourth step is used to check the remaining elements. If an element derivative does not exist already, as an intent, and the evaluation 3 says that it is a maximal rectangle, then, it adds the object and its derivative as a new maximal rectangle. Otherwise, if the element derivative already exists, it is added to the previously formal concept as its extent. In the last step, it processes each maximal rectangle that has been found. If that element is already contained in the lattice, it adds a link between Infimum and the previous element created in the lattice. Otherwise, it adds that link and processes that formal concept recursively. Then, it adds this new element to the lattice structure.

At this point, it computes all the concepts given by the binary context, but $\mathfrak{B}(\mathbb{K})$ cardinality is bounded by an exponential complexity. A simple way to avoid this issue is to use a Binary Tree to store and recover all elements in$\mathfrak{B}(\mathbb{K})$. Basically, this binary tree works as a hash function using the concept of intent in their binary representation. Then, it searches, finds and adds any intent in $|M|$ steps. In addition, the processing order enables us to generate the edges of the Hasse diagram without additional computing steps. Therefore, the process of generating the concept lattice and Hasse diagram presented in this paper has an expected polynomial delay time [16] for maximal anti-chains search.

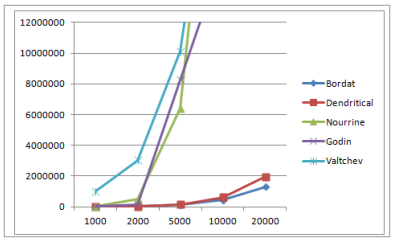
5 Computational Experiments.

As shown in [16], many parameters are involved in the performance and running time of an algorithm. For our tests, we considered the number of objects, the number of attributes, the density of the context and the worst case for contexts of $M \times M$. Where density is the least percentaje of binary relations for each object in $\mathbb{K}$. Those parameters were tested independently. Four algorithms were selected to compare the Dendritical algorithm performance. Implementations were performed in java.

Figure 1 shows the execution time behavior for a formal context with 10% density and 100 attributes. Here, the V-axis represent the running time of each algorithm in ms, and the H-axis the total number of objects under constant number of attributes. The test was performed by increasing the number of objects from one thousand to twenty thousand, with a random distribution and 10% of density.

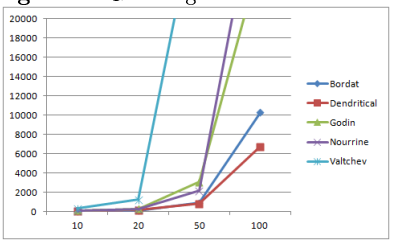
We can verify that Bordat algorithm and our algorithm have the best performance when the number of objects increase with respect to the attributes.

Figure 1. Growing objects number.



Execution time when the number of objects grows from 1000 to 20000, with 100 attributes and 10% of density.

Figure 2. Growing attributes number

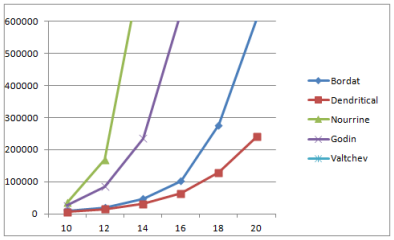


Execution time when the number of attributes from 10 to 100 with 1000 objects and 10% density case.

Figure 2 shows the increase in execution time when the number of attributes increases. All datasets for this test has a 10% density and 1000 objects, and the number of attributes is increasing from ten to one hundred, and, the number of formal concepts is growing too. Here we can notice that Bordat and Dendritical algorithms, are faster than Godin, Nourrine and Valtchev algorithms. We can also notice that Dendritical execution time behavior is faster when the number of objects grows.

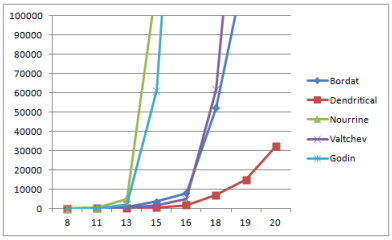
Figure 3 shows the increase in execution time when the density percentage becomes higher. All datasets has 1000 objects and 100 attributes and when the density grows the number of formal concepts grows too. As we mention, density is mainly the percentage of attributes for all the objects in the formal context.

Figure 3. Growing density percentage



Execution time when density increases from 10% to 20%. 1000 objects and 100 attributes case.

Figure 4. MxM worst case contexts



Algorithms running diagonal contexts in which number of attributes are growing from 8 to 20 attributes.

Here, we notice that when the density grows Dendritical is closer and even faster than the other algorithms. Figure 4 shows running time for zero diagonal contexts where |G| = |M| and yields the complete lattice, which means $2^{|M|}$ formal concepts. In this kind of formal contexts, we can see a clear running time superiority of the dendritical algorithm. Finally, table 1 shows some examples of time complexity at each algorithm step.

Table 1. Execution time examples

No. Obj	No. Att	Density	Concepts	Query Time	Ordering	Dendritical	Total
1000	*36*	*17*	*8377*	722ms	20ms	49ms	791ms
1000	*49*	*14*	*14190*	924ms	67ms	101ms	1092ms
1000	*81*	*11*	*26065*	1853ms	124ms	201ms	2178ms

Algorithms time when density becomes higher and the number of attributes become lower. Those tests shows the increase in execution when attributes and $I \subseteq G \times M$ are modified.

6 Conclusion

In this paper, we presented an algorithm based on the main idea of maximal rectangles, using the cardinality notion and the dendritical classifier. We have also compared it with some known algorithms for the construction of Concept Lattices.

From the tests presented in this paper, we can see that as the number of objects grows, our algorithm time execution is higher than Bordat or other methods, but it could be a better choice when the density or the number of attributes is high. Also, the results shows a good performance for the $M \times M$ contexts in the worst case scenario, which demonstrates the feasibility of our algorithm on some kinds of datsets.

Bibliography

[1] Belohlavek, R., Beydoun, G.: Formal Concept Analysis With Background Knowledge: Attribute Priorities. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **5465** (2009) 109–117

[2] Hereth, J., Stumme, G., Wille, R., Wille, U.: Conceptual knowledge discovery: a human-centered approach. Journal of Applied Artificial Intelligence **17** (2003) 281–302

[3] Wille, R.: Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts. In: ICFCA. (2009) 314–339

[4] Valtchev, P., Missaoui, R., Godin, R., Meridji, M.: Generating frequent itemsets incrementally: two novel approaches based on Galois lattice theory. J. Exp. Theor. Artif. Intell. **14**(2-3) (2002) 115–142

[5] Maddouri, M.: A Formal Concept Analysis Approach to Discover Association Rules from Data. R. Belohlavek. V. Snasel CLA **1** (2005) 10–21

[6] Agrawal, R.: Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, **1** (1994) 487–499

[7] Lakhal, L., S.G.: Efficient Mining of Association Rules Based on Formal Concept Analysis. Ganter et al. Springer. **LNAI 3626** (2005) 180–195

[8] Merwe, D., Obiedkov, S., Kourie, D.: AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In Eklund, P., ed.: Concept Lattices. Volume 2961 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2004) 372–385

[9] Lv, L., Zhang, L., Jia, P., Zhou, F.: A Bottom-Up Incremental Algorithm of Building Concept Lattice. In Wu, Y., ed.: Software Engineering and Knowledge Engineering: Theory and Practice. Volume 115 of Advances in Intelligent and Soft Computing. Springer Berlin Heidelberg (2012) 91–98

[10] Valtchev, P., Missaoui, R., Lebrun, P.: A partition-based approach towards constructing Galois (concept) lattices. Discrete Math. **256**(3) (2002) 801–829

[11] Nourine, L., Raynaud, O.: A Fast Algorithm for Building Lattices. Inf. Process. Lett. **71**(5-6) (1999) 199–204

[12] Nourine, L., Raynaud, O.: A fast incremental algorithm for building lattices. J. Exp. Theor. Artif. Intell. **14**(2-3) (2002) 217–227

[13] Farach-Colton, M., Huang, Y.: A Linear Delay Algorithm for Building Concept Lattices. In Ferragina, P., Landau, G., eds.: Combinatorial Pattern Matching. Volume 5029 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2008) 204–216

[14] Godin, R., Missaoui, R., Alaoui, H.: Incremental concept formation algorithms based on Galois (Concept) lattices. Computational Intelligence **11**(2) (1995) 246–267

[15] Ganter, B.: Two basic algorithms in concept analysis. In: Proceedings of the 8th international conference on Formal Concept Analysis. ICFCA-10, Berlin, Heidelberg, Springer-Verlag (2010) 312–340
[16] Kuznetsov, S.O., Obiedkov, S.A.: Comparing performance of algorithms for generating concept lattices. Journal of Experimental and Theoretical Artificial Intelligence **14**(2-3) (2002) 189–216
[17] Kaburlasos, V.G., Ritter, G.X.: Computational Intelligence Based on Lattice Theory. Volume 67 of Studies in Computational Intelligence. Springer (2007)
[18] B. A. Davey, H.A.P.: Introduction to lattices and orders. 2nd edn. Press Sindicate H. Cambridge University (2002)
[19] Ritter, G.X., Iancu, L., Urcid, G.: Neurons, Dendrites, and Pattern Classification. In: CIARP. (2003) 1–16
[20] Urcid, G., Ritter, G.X., Iancu, L.: Single Layer Morphological Perceptron Solution to the N-Bit Parity Problem. In: CIARP. (2004) 171–178
[21] Ritter, G., Urcid, G.: Learning in Lattice Neural Networks that Employ Dendritic Computing. In Kaburlasos, V., Ritter, G., eds.: Computational Intelligence Based on Lattice Theory. Volume 67 of Studies in Computational Intelligence. Springer Berlin / Heidelberg (2007) 25–44
[22] Barmpoutis, A., Ritter, G.X.: Orthonormal Basis Latice Neural Networks. In Computational Intelligence Based on LatticeTheory, V. Kaburlasos and G. X. Ritter **1** (Springer-Verlag, Heidelberg, Germany, 2007) 43–56
[23] Kaburlasos, V.: Granular Enhancement of Fuzzy ART-SOM Neural Classifiers Based on Lattice Theory. In Kaburlasos, V., Gerhard, R., eds.: Computational Intelligence Based on Lattice Theory. Volume 67 of Studies in Computational Intelligence. Springer Berlin / Heidelberg (2007) 3–23
[24] Aldape-Perez, M., Yanez-Marquez, C., Camacho-Nieto, O., J.Arguelles-Cruz, A.: An associative memory approach to medical decision support systems. Comput. Methods Prog. Biomed. **106**(3) (2012) 287–307
[25] Ritter, G.X., Chyzhyk, D., Urcid, G., Grana, M.: A Novel Lattice Associative Memory Based on Dendritic Computing. In: HAIS. (2012) 491–502
[26] Kaburlasos, V.G., P.V.: Fuzzy lattice neurocomputing (fln) models. Neural Networks **13 (10)** (2000) 1145–1170.
[27] Witte, V., Schulte, S., Nachtegael, M., Malange, T., Kerre, E.: A Lattice-Based Approach to Mathematical Morphology for Greyscale and Colour Images. In Kaburlasos, V., Ritter, G., eds.: Computational Intelligence Based on Lattice Theory. Volume 67. Springer Berlin / Heidelberg (2007) 129–148
[28] Urcid, G., Nieves-Vazquez, J.A., Garcia-A., A., Valdiviezo-N., J.C.: Robust image retrieval from noisy inputs using lattice associative memories. In: Image Processing: Algorithms and Systems. (2009)
[29] Atif, J., B.I.D.F.H.C.: Mathematical morphology operators over concept lattices. . In: Cellier, P., Distel, F., Ganter, B. (Eds.) ICFCA 2013, (Springer-Verlag Berlin Heidelberg) **LNAI 7880** (2013) 28–43.